
Using Driverless AI

Release 1.9.0.1

H2O.ai

Aug 11, 2020

RELEASE NOTES

1 Change Log	3
1.1 Version 1.9.0.1 (Aug 10, 2020)	3
1.2 Version 1.9.0 (July 27, 2020)	3
1.3 Version 1.8.7.2 LTS (July 13, 2020)	6
1.4 Version 1.8.7.1 LTS (June 23, 2020)	6
1.5 Version 1.8.7 LTS (June 15, 2020)	6
1.6 Version 1.8.6 LTS (Apr 30, 2020)	7
1.7 Version 1.8.5 LTS (Mar 09, 2020)	9
1.8 Version 1.8.4.1 LTS (Feb 4, 2020)	10
1.9 Version 1.8.4 LTS (Jan 31, 2020)	10
1.10 Version 1.8.3 LTS (Jan 22, 2020)	11
1.11 Version 1.8.2 LTS (Jan 17, 2020)	11
1.12 Version 1.8.1.1 (Dec 21, 2019)	13
1.13 Version 1.8.1 (Dec 10, 2019)	13
1.14 Version 1.8.0 (Oct 3, 2019)	15
1.15 Version 1.7.1 (Aug 19, 2019)	17
1.16 Version 1.6.4 LTS (Aug 19, 2019)	18
1.17 Version 1.7.0 (Jul 7, 2019)	19
1.18 Version 1.6.3 LTS (June 14, 2019)	20
1.19 Version 1.6.2 LTS (May 10, 2019)	21
1.20 Version 1.6.1.1 LTS (Apr 24, 2019)	21
1.21 Version 1.6.1 LTS (Apr 18, 2019)	21
1.22 Version 1.6.0 LTS (Apr 5, 2019)	22
1.23 Version 1.5.4 (Feb 24, 2019)	22
1.24 Version 1.5.3 (Feb 8, 2019)	23
1.25 Version 1.5.2 (Feb 2, 2019)	23
1.26 Version 1.5.1 (Jan 22, 2019)	24
1.27 Version 1.5.0 (Jan 18, 2019)	24
1.28 Version 1.4.2 (Dec 3, 2018)	25
1.29 Version 1.4.1 (Nov 11, 2018)	25
1.30 Version 1.4.0 (Oct 27, 2018)	25
1.31 Version 1.3.1 (Sep 12, 2018)	26
1.32 Version 1.3.0 (Sep 4, 2018)	26
1.33 Version 1.2.2 (July 5, 2018)	27
1.34 Version 1.2.1 (June 26, 2018)	27
1.35 Version 1.2.0 (June 11, 2018)	28
1.36 Version 1.1.6 (May 29, 2018)	29
1.37 Version 1.1.4 (May 17, 2018)	29
1.38 Version 1.1.3 (May 16, 2018)	29
1.39 Version 1.1.2 (May 8, 2018)	30

1.40	Version 1.1.1 (April 23, 2018)	30
1.41	Version 1.1.0 (April 19, 2018)	30
1.42	Version 1.0.30 (April 5, 2018)	31
1.43	Version 1.0.29 (April 4, 2018)	31
1.44	Version 1.0.28 (April 3, 2018)	31
1.45	Version 1.0.27 (March 31, 2018)	31
1.46	Version 1.0.26 (March 28, 2018)	31
1.47	Version 1.0.25 (March 22, 2018)	32
1.48	Version 1.0.24 (March 8, 2018)	32
1.49	Version 1.0.23 (March 7, 2018)	32
1.50	Version 1.0.22 (Feb 23, 2018)	33
1.51	Version 1.0.21 (Feb 21, 2018)	33
1.52	Version 1.0.20 (Feb 17, 2018)	33
1.53	Version 1.0.19 (Jan 28, 2018)	34
1.54	Version 1.0.18 (Jan 24, 2018)	34
1.55	Version 1.0.17 (Jan 23, 2018)	34
1.56	Version 1.0.16 (Jan 22, 2018)	34
1.57	Version 1.0.15 (Jan 11, 2018)	35
1.58	Version 1.0.14 (Jan 11, 2018)	35
1.59	Version 1.0.13 (Jan 10, 2018)	35
1.60	Version 1.0.11 (Dec 12, 2017)	36
1.61	Version 1.0.10 (Dec 4, 2017)	36
1.62	Version 1.0.9 (Nov 29, 2017)	36
1.63	Version 1.0.8 (Nov 21, 2017)	37
1.64	Version 1.0.7 (Nov 17, 2017)	37
1.65	Version 1.0.5 (Oct 24, 2017)	37
1.66	Version 1.0.4 (Oct 19, 2017)	37
1.67	Version 1.0.3 (Oct 9, 2017)	37
1.68	Version 1.0.2 (Oct 5, 2017)	38
1.69	Version 1.0.1 (Oct 4, 2017)	38
1.70	Version 1.0.0 (Sep 24, 2017)	38
2	Introduction to H2O Driverless AI	39
2.1	Architecture	41
2.2	Roadmap	42
3	Why Driverless AI?	43
4	Key Features	45
4.1	Flexibility of Data and Deployment	45
4.2	NVIDIA GPU Acceleration	45
4.3	Automatic Data Visualization (Autovis)	45
4.4	Automatic Feature Engineering	45
4.5	Automatic Model Documentation	46
4.6	Time Series Forecasting	46
4.7	NLP with TensorFlow	46
4.8	Automatic Scoring Pipelines	46
4.9	Machine Learning Interpretability (MLI)	46
4.10	Automatic Reason Codes	47
4.11	Custom Recipe Support	47
5	Supported Algorithms	49
5.1	Constant Model	49
5.2	Decision Tree	49
5.3	FTRL	49

5.4	GLM	49
5.5	Isolation Forest	50
5.6	LightGBM	50
5.7	PyTorch Models	50
5.8	RuleFit	50
5.9	TensorFlow	50
5.10	XGBoost	51
5.11	Zero-Inflated Models	51
5.12	References	51
6	Driverless AI Workflow	53
7	Driverless AI Licenses	55
7.1	About Licenses	55
7.2	Adding Licenses for the First Time	55
7.3	Updating Licenses	58
8	Supported Environments	63
8.1	Linux	63
8.2	Windows 10 Pro, Enterprise, or Education	63
8.3	Mac OS X	63
8.4	IBM Power	64
9	Before You Begin Installing or Upgrading	65
9.1	Supported Browsers	65
9.2	To sudo or Not to sudo	65
9.3	Note about Docker Configuration (ulimit)	65
9.4	Note about nvidia-docker 1.0	65
9.5	Deprecation of nvidia-smi	66
9.6	New nvidia-container-runtime-hook Requirement for PowerPC Users	66
9.7	Note About CUDA Versions	66
9.8	Note About Authentication	66
9.9	Note About Shared File Systems	66
9.10	Note About the Master Database File	67
9.11	Backup Strategy	67
9.12	Upgrade Strategy	67
10	Sizing Requirements	69
10.1	Sizing Requirements for Native Installs	69
10.2	Sizing Requirements for Docker Installs	69
10.3	GPU Sizing Requirements	69
10.4	Sizing Requirements for Storing Experiments	69
10.5	Virtual Memory Settings in Linux	69
11	Installing and Upgrading Driverless AI	71
11.1	Linux X86_64 Installs	71
11.2	IBM Power Installs	136
11.3	Mac OS X	154
11.4	Windows 10	159
12	Using the config.toml File	165
12.1	Configuration Override Chain	165
12.2	Sample Config.toml File	166
13	Environment Variables and Configuration Options	201

13.1	Setting Environment Variables and Configurations Options	201
14	Multinode Training (Alpha)	203
14.1	Understanding Multinode Training	203
14.2	Requirements	204
14.3	Description of Configuration Attributes	204
14.4	Multinode Setup Example	205
15	Enabling Data Connectors	207
15.1	S3 Setup	208
15.2	HDFS Setup	211
15.3	Azure Blob Store Setup	217
15.4	BlueData DataTap Setup	223
15.5	Google BigQuery Setup	228
15.6	Google Cloud Storage Setup	231
15.7	Hive Setup	233
15.8	JDBC Setup	236
15.9	kdb+ Setup	243
15.10	Minio Setup	247
15.11	Snowflake Setup	249
15.12	Data Recipe URL Setup	252
15.13	Data Recipe File Setup	254
16	Configuring Authentication	257
16.1	Client Certificate Authentication Example	257
16.2	LDAP Authentication Example	261
16.3	Local Authentication Example	264
16.4	mTLS Authentication Example	265
16.5	OpenID Connect Authentication Examples	268
16.6	PAM Authentication Example	272
17	Enabling Notifications	275
17.1	Script Interfaces	275
17.2	Example	275
18	Export Artifacts	279
18.1	Enabling Artifact Exports	279
18.2	Exporting an Artifact	280
19	Changing the Language in the UI	283
19.1	Examples	283
20	Launching Driverless AI	285
20.1	Resources	287
20.2	Messages	288
21	The Datasets Page	291
21.1	Supported File Types	291
21.2	Adding Datasets	292
21.3	Renaming Datasets	293
21.4	Dataset Details and Rows	293
21.5	Downloading Datasets	294
21.6	Splitting Datasets	295
21.7	Visualizing Datasets	295

22 Experiments	301
22.1 Before You Begin	301
22.2 New Experiments	357
22.3 Completed Experiment	363
22.4 Model Insights	365
22.5 Model Scores	367
22.6 Experiment Graphs	369
22.7 Experiment Summary	374
22.8 Viewing Experiments	378
23 Diagnosing a Model	383
23.1 Classification Metric Plots	383
23.2 Regression Metric Plots	385
24 Project Workspace	387
24.1 Linking Datasets	387
24.2 Linking Experiments	389
24.3 Experiments List	391
24.4 Unlinking Data on a Projects Page	394
24.5 Deleting Projects	394
25 Leaderboards	397
25.1 Creating a Leaderboard	397
25.2 Leaderboard Models	399
25.3 Regular Experiments	399
25.4 Time Series Experiments	399
26 MLI Overview	401
27 The Interpreted Models Page	403
28 MLI for Regular (Non-Time-Series) Experiments	405
28.1 Interpreting a Model	405
28.2 Understanding the Model Interpretation Page	409
28.3 Viewing Explanations	431
28.4 General Considerations	432
29 MLI for Time-Series Experiments	433
29.1 Multi-Group Time Series MLI	433
29.2 Single Time Series MLI	436
30 Score on Another Dataset	439
31 Transform Another Dataset	441
32 Scoring Pipelines Overview	443
33 Visualizing the Scoring Pipeline	445
34 Which Pipeline Should I Use?	449
35 Driverless AI Standalone Python Scoring Pipeline	451
35.1 Python Scoring Pipeline Files	451
35.2 Quick Start	452
35.3 The Python Scoring Module	455
35.4 The Scoring Service	456

35.5	Python Scoring Pipeline FAQ	458
35.6	Troubleshooting Python Environment Issues	458
36	Driverless AI MLI Standalone Python Scoring Package	461
36.1	MLI Python Scoring Package Files	461
36.2	Quick Start	462
36.3	MLI Python Scoring Module	465
36.4	K-LIME vs Shapley Reason Codes	465
36.5	MLI Scoring Service Overview	466
37	MOJO Scoring Pipelines	469
37.1	Driverless AI MOJO Scoring Pipeline - Java Runtime	469
37.2	Driverless AI MOJO Scoring Pipeline - C++ Runtime with Python and R Wrappers	475
37.3	MOJO2 Javadoc	478
38	Deploying the MOJO Pipeline	479
38.1	Deployments Overview Page	479
38.2	Available Deployments	479
38.3	Amazon Lambda Deployment	480
38.4	REST Server Deployment	485
39	What's Happening in Driverless AI?	491
40	Data Sampling	493
40.1	Imbalanced Sampling Methods	494
41	Driverless AI Transformations	497
41.1	Available Transformers	497
41.2	Example Transformations	502
42	Internal Validation Technique	505
43	Missing and Unseen Levels Handling	507
43.1	How Does the Algorithm Handle Missing Values During Training?	507
43.2	How Does the Algorithm Handle Missing Values During Scoring (Production)?	508
43.3	What Happens When You Try to Predict on a Categorical Level Not Seen During Training?	508
43.4	What Happens if the Response Has Missing Values?	509
44	Imputation in Driverless AI	511
44.1	Enabling Imputation	511
44.2	Running an Experiment with Imputation	511
45	Time Series in Driverless AI	515
45.1	Understanding Time Series	515
45.2	Rolling-Window-Based Predictions	521
45.3	Time Series Constraints	522
45.4	Time Series Use Case: Sales Forecasting	522
45.5	Using a Driverless AI Time Series Model to Forecast	525
45.6	Time Series Expert Settings	527
45.7	Additional Resources	527
46	NLP in Driverless AI	529
46.1	NLP Feature Engineering	529
46.2	n-gram	530
46.3	Truncated SVD Features	531
46.4	Linear Models for TF-IDF Vectors	531

46.5	Word Embeddings	531
46.6	PyTorch Transformer Architecture Models (eg. BERT) as Modeling Algorithms	534
46.7	NLP Naming Conventions	535
46.8	NLP Expert Settings	536
46.9	A Typical NLP Example: Sentiment Analysis	537
47	Image Processing in Driverless AI	541
47.1	Uploading Data for Image Processing	541
47.2	Modeling Images	541
48	Queuing	545
49	The Python Client	547
49.1	Installing the Python Client	547
49.2	Driverless AI: Credit Card Demo	549
49.3	Driverless AI - Training Time Series Model	566
49.4	Driverless AI - Time Series Recipes with Rolling Window	572
49.5	Driverless AI NLP Demo - Airline Sentiment Dataset	574
49.6	Driverless AI Autoviz Python Client Example	577
50	The R Client	581
50.1	Installing the R Client	581
50.2	R Client Tutorial	583
51	Monitoring Pending Jobs	591
51.1	Failed Jobs	592
52	Driverless AI Logs	593
52.1	While Visualizing Datasets	593
52.2	While an Experiment is Running	594
52.3	After an Experiment has Finished	595
52.4	During Model Interpretation	596
52.5	After Model Interpretation	597
53	Sending Logs to H2O.ai	599
53.1	Dataset Failures	599
53.2	Experiments	599
53.3	MLI	599
53.4	Custom Recipes	599
54	System Logs	601
54.1	Docker Installs	601
54.2	Native Installs	601
55	Driverless AI Security	603
55.1	Objective	603
55.2	Important things to know	603
55.3	User Access	604
55.4	Data Security	604
55.5	Client-Server Communication Security	607
55.6	Web UI Security	609
55.7	Custom Recipe Security	609
55.8	Baseline Secure Configuration	609
56	FAQ	611
56.1	General	614

56.2	Installation/Upgrade/Authentication	615
56.3	Data	619
56.4	Connectors	619
56.5	Recipes	620
56.6	Experiments	620
56.7	Feature Transformations	630
56.8	Predictions	630
56.9	Deployment	631
56.10	Time Series	632
56.11	Logging	634
57	Tips ‘n Tricks	635
57.1	Pipeline Tips	635
57.2	Time Series Tips	636
57.3	Scorer Tips	637
57.4	Knob Settings Tips	637
57.5	Tips for Running an Experiment	638
57.6	Expert Settings Tips	638
57.7	Checkpointing Tips	638
57.8	Text Data Tips	638
58	Appendix A: Custom Recipes	639
58.1	Additional Resources	639
58.2	Examples	639
59	Appendix B: Third-Party Integrations	643
59.1	Instance Life-Cycle Management	643
59.2	API Clients	643
59.3	Scoring	644
59.4	Storage	644
59.5	Data Sources	644
60	References	645

H2O Driverless AI is an artificial intelligence (AI) platform for automatic machine learning. Driverless AI automates some of the most difficult data science and machine learning workflows such as feature engineering, model validation, model tuning, model selection, and model deployment. It aims to achieve highest predictive accuracy, comparable to expert data scientists, but in much shorter time thanks to end-to-end automation. Driverless AI also offers automatic visualizations and machine learning interpretability (MLI). Especially in regulated industries, model transparency and explanation are just as important as predictive performance. Modeling pipelines (feature engineering and models) are exported (in full fidelity, without approximations) both as Python modules and as Java standalone scoring artifacts.

Driverless AI runs on commodity hardware. It was also specifically designed to take advantage of graphical processing units (GPUs), including multi-GPU workstations and servers such as [IBM's Power9-GPU AC922 server](#) and the [NVIDIA DGX-1](#) for order-of-magnitude faster training.

This document describes how to install and use Driverless AI. For more information about Driverless AI, see <https://www.h2o.ai/products/h2o-driverless-ai/>.

For a third-party review, see <https://www.infoworld.com/article/3236048/machine-learning/review-h2oai-automates-machine-learning.html>.

Have Questions?

If you have questions about using Driverless AI, post them on Stack Overflow using the driverless-ai tag at <http://stackoverflow.com/questions/tagged/driverless-ai>.

You can also post questions on the H2O.ai Community Slack workspace in the **#driverlessai** channel. If you have not signed up for the H2O.ai Community Slack workspace, you can do so here: <https://www.h2o.ai/community/>.

CHANGE LOG

1.1 Version 1.9.0.1 (Aug 10, 2020)

- Bug Fixes:
 - Fix migration for certain time-series experiments
 - Fix missing files for automatic image model
 - Fix MLI job status for PDP/ICE
 - Fix handling of ID column for MLI kernel shapley
 - Fix exception handling for startup failures
 - Constrain Python environment for standalone scoring package

1.2 Version 1.9.0 (July 27, 2020)

- New Features:
 - *Multinode training* (alpha)
 - *Queuing of experiments* to avoid system overload
 - *Automatic Leaderboard*: Single-button creation of a project with a series of diverse experiments
 - *Multi-layer hierarchical feature engineering*:
 - * Allow optional pre-processing layer for specific custom data cleanup/conversions
 - * Subsequent layers take each previous layer's output as input (can be numeric or categorical/string)
 - PyTorch deep learning backend in addition to TensorFlow
 - *Image classification and regression* with pre-trained and fine-tuned state-of-the-art Deep Learning models:
 - * Image data ingest from binary archives
 - Archives can contain (one) optional .csv file with mapping of image paths to target (regression/classification)
 - Automatic training dataset creation and label creation (from directory structure) if no .csv provided
 - * Image Transformers (for converting image path columns
 - “densenet121”, “efficientnetb0”, “efficientnetb2”, “inception_v3”, “mobilenetv2”, “resnet34”, “resnet50”, “seresnet50”, “seresnext50”, “xception”

- Optional fine-tuning
 - Optional GPU acceleration (strongly recommended when enabling fine-tuning)
 - Pretrained and fine-tunable ImageVectorizer transformer with automatic dimensionality reduction
 - Images can be provided either as zipped archives, or as paths to local or remote locations (URIs)
 - Automatic image labeling when importing zipped archives of images (based on folder names and structure)
 - Can handle multiple image columns with URIs in a tabular dataset
 - Single experiment can combine image, NLP and tabular data
 - MOJO support (also for CPU-only systems)
- * Automatic Image model
 - End-to-end model training, no tuning needed
 - State-of-the-art results with grandmaster techniques
 - Neural architecture search based on pretrained and fine-tuned TensorFlow models
 - Multi-GPU training
 - Visual insights in GUI (losses, sample images, augmentation, Grad-CAM visual explanations)
 - * MLI is not available for image experiments and is a work in progress
- *PyTorch BERT NLP* pre-trained and fine-tuned state-of-the-art Deep Learning models:
 - * “bert-base-uncased”, “distilbert-base-uncased”, “xlnet-base-cased”, “xlm-mlm-enfr-1024”, “roberta-base”, “albert-base-v2”, “camembert-base”, “xlm-roberta-base”
 - * Optional GPU acceleration (strongly recommended)
 - * MOJO support (also for CPU-only systems)
 - * BERT transformers (for converting text columns into numeric features for other models like GBMs)
 - * BERT models (when only have one text column)
 - AutoReport now includes the following:
 - * Information about the time series validation strategy
 - * Experiment lineage (model lineage plot)
 - * NLP/Image architecture details
 - *Zero-inflated regression models* for insurance use cases (combination of classification + regression models)
 - *Time series centering and de-trending transformations*:
 - * Inner ML model is trained on residuals after fitting and removing trend from target signal (per time-series group)
 - * Support for constant (centering), linear and logistic trends
 - * SEIRD model for epidemic modeling of (S)usceptible, (E)xposed, (I)nfected, (R)ecovered and (D)eceased, fully configurable lower/upper bounds for model parameters
 - *Graphical config.toml editor for expert settings*
 - *Empiric prediction intervals* for regression problems with user-defined confidence levels (based on holdout predictions)

- *Insights tab* with helpful visualizations (currently only for time-series and image problems)
- For binary classification problems with F05, F1, F2, MCC scorers, use the same metric for optimal threshold determination
- *Custom data recipes* can now be part of the experiment's modeling pipeline, and will be part of the Python scoring package
- Custom visualizations in AutoViz following the Grammar of Graphics
- Pass data to (custom) scorers, so can access other columns, not only actual and predicted values
- Added many new *scorers for common regression and classification metrics* out of the box
- Added holiday calendar for 24 more countries, allow user to select list of countries to create is-holiday features for.
- Added identity_no_clip target transformer for regression problems that never clips the predictions to observed ranges and allows extrapolation
- *MLI*:
 - * New GUI/UX for MLI
 - * Added Kernel Explainer for original feature Shapley importance
 - * Added ability to download Shapley values for original features from UI as CSV file
 - * Added intercept column to k-LIME output CSV file
 - * Added ability to run surrogate models on DAI model residuals to help debug model errors
 - * Added ability to export Decision Tree Surrogate model rules as text and Python code
 - * Added Decision Tree Surrogate model for multinomial experiments
 - * Added Leave One Covariate Out (LOCO) for multinomial experiments
 - * Added two traditional fair lending metrics for Disparate Impact Analysis (DIA): Standardized Mean Difference (SMD) and Marginal Error (ME)
 - * Added two interpretable model recipes to <https://github.com/h2oai/driverlessai-recipes>: GA2M and XNN (<https://github.com/h2oai/driverlessai-recipes/tree/master/models/mli>)
 - * Display prediction label for binary classification experiments in MLI summary page
- Improvements:
 - Improved parsability (machine readability) of log files
 - Custom recipes are now only visible to the user that created them, previously created custom recipes remain globally visible
 - Faster time-series experiments
 - Improve preview to show more details about modeling part of final pipeline
 - Improved notifications system
 - Reduced size of MOJO
 - Only allow imbalanced sampling techniques when data is larger than user controllable threshold
 - Upgraded to latest H2O-3 backend for custom recipes
 - Faster feature selection for large imbalanced datasets
- Documentation updates:

- Added animated GIFs
- Added tabbed content
- Added more details for *imbalanced sampling methods* for binary classification
- New content (refer to above linked topics)
- Bug fixes:
 - Various bug fixes

1.3 Version 1.8.7.2 LTS (July 13, 2020)

- Bug Fixes:
 - Add and pass authentication_method parameter to use proper get_true_username and start_session
 - SQL-like connector: strip unnecessary semi-colon from the end of query
- Documentation updates:
 - Document use of hive_app_jvm_args

1.4 Version 1.8.7.1 LTS (June 23, 2020)

- New Features:
 - Add ability to push artifacts to a Bitbucket server
 - Add per-feature user control for monotonicity constraints for XGBoostGBM, LightGBM and DecisionTree models
- Bug Fixes:
 - Fix Hive kerberos impersonation
 - Fix a DTap connector issue by using the proper login username for impersonation
 - Fix monotonicity constraints for XGBoostGBM, LightGBM and DecisionTree models

1.5 Version 1.8.7 LTS (June 15, 2020)

- New Features:
 - Add intercept term to k-LIME csv
 - Add control of default categorical & numeric feature rendering in DAI PD/ICE
 - Add ability to restrict custom recipe upload to a specific git repository and branch
 - Add translations for Korean and Chinese
 - Add ability to use multiple authentication methods simultaneously
- Improvements:
 - Improve behavior of systemctl in the case Driverless AI fails to start
 - Improve logging behavior for JDBC and Hive connectors

- Improve behavior of C++ scorer, fewer unnecessary files saved in tmp directory
- Improve Docker image behavior in Kubernetes
- Improve LDAP authentication to allow for anonymous binding
- Improve speed of feature selection for experiments on large, wide, imbalanced datasets
- Improve speed of data import on busy system
- Bug fixes:
 - Fix automatic Kaggle submission and score retrieval
 - Fix intermittent Java exception seen by surrogate DRF model in MLI when several MLI jobs are run concurrently
 - Fix issue with deleting Deployments if linked Experiment was deleted
 - Fix issue causing Jupyter Notebooks to not work properly in Docker Image
 - Fix custom recipe scorers not being displayed on Diagnostics page
 - Fix issue with AWS Lambda Deployment not handling dropped columns properly
 - Fix issue with not being able to limit number of GPUs for specific experiment
 - Fix in-server scoring inaccuracies for certain models built in 1.7.1 and 1.8.0 (standalone scoring not affected)
 - Fix rare datatable type casting exception
- Documentation updates:
 - The “Maximum Number of Rows to Perform Permutation-Based Feature Selection” expert setting now has a default value of 500,000
 - Improved Hive and Snowflake connector documentation
 - Updated the Main.java example in the Java Scoring Pipeline chapter
 - Added documentation describing how to change the language in the UI before starting the application
 - Added information about how custom recipes are described and documented in the Autoreport
 - Updated the LDAP authentication documentation
 - Improved the Linux DEB and RPM installation instructions
 - Improved the AWS Community AMI installation instructions
 - Improved documentation for the Reproducible button

1.6 Version 1.8.6 LTS (Apr 30, 2020)

- New Features:
 - Add expert setting to reduce size of MOJO scoring pipelines (and hence reduce latency and memory usage for inference)
 - Enable Lambda deployment for IBM Power
 - Add restart button for Deployments
 - Add automatic Kaggle submission for supported datasets, show private/public scores (requires Kaggle API Username/Key)

- Show warning if single final model is worse on back-testing splits (for time series) or cross-validation folds (for IID) than the fold models (indicates issue with signal or fit)
- Update R client API to include autodoc, experiment preview, dataset download, autovis functions
- Add button in expert settings that toggle some effective settings to make a small MOJO production pipeline
- Add an option to upload artifacts to S3 or a Git repository
- Improvements:
 - Improve experiment restart/refit robustness if model type is changed
 - Extra protection against dropping features
 - Improve implementation of Hive connector
- Bug fixes:
 - Upgrade datatable to fix endless loop during stats calculation at file import
 - Web server and UI now respect dynamic base URL suffix
 - Fix incorrect min_rows in MLI when providing weight column with small values
 - Fix segfault in MOJO for TensorFlow/PyTorch models
 - Fix elapsed time for MLI
 - Enable GPU by default for R client
 - Fix python scoring h2oai ModuleNotFound error
 - Update no_drop_features toml and expert button to more generally avoid dropping features
 - Fix datatable mmap strategy
- Documentation updates:
 - Add documentation for enabling the Hive data connector
 - Add documentation for updating expired DAI licenses on AWS Lambda deployments using a script
 - Documentation for uploading artifacts now includes support for S3 and Git in the artifacts store
 - Improve documentation for one-hot encoding
 - Improve documentation for systemd logs/journalctl
 - Improve documentation for time series ‘unavailable columns at prediction time’
 - Improve documentation for Azure blob storage
 - Improve documentation for MOJO scoring pipeline
 - Add information about reducing the size of a MOJO using a new expert setting

1.7 Version 1.8.5 LTS (Mar 09, 2020)

- New Features:
 - Handle large (up to 10k) multiclass problems, including GUI improvements in such cases
 - Detect class imbalance for binary problems where target class is non-rare
 - Add feature count to iteration panel
 - Add experiment lineage pdf in experiment summary zip file
 - Issue warnings if final pipeline scores are unstable across (cross-)validation folds
 - Issue warning if Constant Model is improving quality of final pipeline (sign of bad signal)
 - Report origin of leakage detection as from model fit (AUC/R2), GINI, or correlation
- Improvements:
 - Improve handling of ID columns
 - Improve exception handling to improve stability of raising python exceptions
 - Improve exception handling when any individual transformer or model throw exception or segfaults
 - Improve robustness of restart and refit experiment to changes in experiment choices
 - Improve handling of missing values when transforming dataset
 - Improve robustness of custom recipe importing of modules
 - Improve documentation for installation instructions
 - Improve selection of initial lag sizes for time series
 - Improve LightGBM stability for regression problems for certain mutation parameters
- Documentation updates:
 - Improved documentation for time-series experiments
 - Added topics describing how to re-enable the Data Recipe URL and Data Recipe File connectors
 - For users running older versions of the Standalone Python Scoring Pipeline, added information describing how to install upgraded versions of outdated dependencies
 - Improved the description for the “Sampling Method for Imbalanced Binary Classification Problems” expert setting
 - Added constraints related to the REST server deployments
 - Noted required vs optional parameters in the HDFS connector topics
 - Added an FAQ indicating that MOJOs are thread safe
 - On Windows 10, only Docker installs are supported
 - Added information about the Recommendations AutoViz graph
 - Added information to the Before you Begin Installing topic that master.db files are not backward compatible with earlier Driverless AI versions
- Bug fixes:
 - Update LightGBM for bug fixes, including hangs and avoiding hard-coded library paths
 - Stabilize use of psutil package

- Fix time-series experiments when test set has missing target values
- Fix python scoring to not depend upon original data_directory
- Fix preview for custom time series validation splits and low accuracy
- Fix ignored minimum lag size setting for single time series
- Fix parsing of Excel files with datetime columns
- Fix column type detection for columns with mostly missing values
- Fix invalid display of 0.0000 score in iteration scores
- Various MLI fixes (don't show invalid graphs, fix PDP sort order, overlapping labels)
- Various bug fixes

1.8 Version 1.8.4.1 LTS (Feb 4, 2020)

Available here

- Add option for dynamic port allocation
- Documentation for AWS community AMI
- Various bug fixes (MLI UI)

1.9 Version 1.8.4 LTS (Jan 31, 2020)

Available here

- New Features:
 - Added ‘Scores’ tab in experiment page to show detailed tuning tables and scores for models and folds
 - Added Constant Model (constant predictions) and use it as reference model by default
 - Show score of global constant predictions in experiment summary as reference
 - Added support for setting up mutual TLS for the DriverlessAI
 - Added option to use client/personal certificate as an authentication method
- Documentation Updates:
 - Added sections for enabling mTLS and Client Certificate authentication
 - Constant Models is now included in the list of Supported Algorithms
 - Added a section describing the Model Scores page
 - Improved the C++ Scoring Pipeline documentation describing the process for importing datatable
 - Improved documentation for the Java Scoring Pipeline
- Bug fixes:
 - Fix refitting of final pipeline when new features are added
 - Various bug fixes

1.10 Version 1.8.3 LTS (Jan 22, 2020)

Available here

- Added option to upload experiment artifacts to a configured disk location
- Various bug fixes (correct feature engineering from time column, migration for brain restart)

1.11 Version 1.8.2 LTS (Jan 17, 2020)

Available here

- New Features:
 - Decision Tree model
 - Automatically enabled for accuracy ≤ 7 and interpretability ≥ 7
 - Supports all problem types: regression/binary/multiclass
 - Using LightGBM GPU/CPU backend with MOJO
 - Visualization of tree splits and leaf node decisions as part of pipeline visualization
 - Per-Column Imputation Scheme (experimental)
 - Select one of [const, mean, median, min, max, quantile] imputation scheme at start of experiment
 - Select method of calculation of imputation value: either on entire dataset or inside each pipeline's training data split
 - Disabled by default and must be enabled at startup time to be effective
 - Show MOJO size and scoring latency (for C++/R/Python runtime) in experiment summary
 - Automatically prune low weight base models in final ensemble (based on interpretability setting) to reduce final model complexity
 - Automatically convert non-raw github URLs for custom recipes to raw source code URLs
- Improvements:
 - Speed up feature evolution for time-series and low-accuracy experiments
 - Improved accuracy of feature evolution algorithm
 - Feature transformer interpretability, total count, and importance accounted for in genetic algorithm's model and feature selection
 - Binary confusion matrix in ROC curve of experiment page is made consistent with Diagnostics (flipped positions of TP/TN)
 - Only include custom recipes in Python scoring pipeline if the experiment uses any custom recipes
 - Additional documentation (New OpenID config options, JDBC data connector syntax)
 - Improved AutoReport's transformer descriptions
 - Improved progress reporting during Autoreport creation
 - Improved speed of automatic interaction search for imbalanced multiclass problems
 - Improved accuracy of single final model for GLM and FTRL
 - Allow config_overrides to be a list/vector of parameters for R client API

- Disable early stopping for Random Forest models by default, and expose new ‘rf_early_stopping’ mode (optional)
- Create identical example data (again, as in 1.8.0 and before) for all scoring pipelines
- Upgraded versions of datatable and Java
- Installed graphviz in Docker image, now get .png file of pipeline visualization in MOJO package and Autoreport. Note: For RPM/DEB/TAR SH installs, user can install graphviz to get this optional functionality
- Documentation Updates:
 - Added a simple example for modifying a dataset by recipe using live code
 - Added a section describing how to impute datasets (experimental)
 - Added Decision Trees to list of supported algorithms
 - Fixed examples for enabling JDBC connectors
 - Added information describing how to use a JDBC driver that is not tested in house
 - Updated the Missing Values Handling topic to include sections for “Clustering in Transformers” and “Isolation Forest Anomaly Score Transformer”
 - Improved the “Fold Column” description
- Bug Fixes:
 - Fix various reasons why final model score was too far off from best feature evolution score
 - Delete temporary files created during test set scoring
 - Fixed target transformer tuning (was potentially mixing up target transformers between feature evolution and final model)
 - Fixed tensorflow_nlp_have_gpus_in_production=true mode
 - Fixed partial dependence plots for missing datetime values and no longer show them for text columns
 - Fixed time-series GUI for quarterly data
 - Feature transformer exploration limited to no more than 1000 new features (Small data on 10/10/1 would try too many features)
 - Fixed Kaggle pipeline building recipe to try more input features than 8
 - Fixed cursor placement in live code editor for custom data recipe
 - Show correct number of cross-validation splits in pipeline visualization if have more than 10 splits
 - Fixed parsing of datetime in MOJO for some datetime formats without ‘%d’ (day)
 - Various bug fixes
- Backward/Forward compatibility:
 - Models built in 1.8.2 LTS will remain supported in upcoming versions 1.8.x LTS
 - Models built in 1.7.1/1.8.0/1.8.1 are not deprecated and should continue to work (best effort is made to preserve MOJO and Autoreport creation, MLI, scoring, etc.)
 - Models built in 1.7.0 or earlier will be deprecated

1.12 Version 1.8.1.1 (Dec 21, 2019)

Available here

- Bugfix for time series experiments with quarterly data when launched from GUI

1.13 Version 1.8.1 (Dec 10, 2019)

Available here

- New Features:
 - Full set of scoring metrics and corresponding downloadable holdout predictions for experiments with single final models (time-series or i.i.d)
 - MLI Updates:
 - * What-If (sensitivity) analysis
 - * Interpretation of experiments on text data (NLP)
 - Custom Data Recipe BYOR:
 - * BYOR (bring your own recipe) in Python: pandas, numpy, datatable, third-party libraries for fast prototyping of connectors and data preprocessing inside DAI
 - * data connectors, cleaning, filtering, aggregation, augmentation, feature engineering, splits, etc.
 - * can create one or multiple datasets from scratch or from existing datasets
 - * interactive code editor with live preview
 - * example code at <https://github.com/h2oai/driverlessai-recipes/tree/rel-1.8.1/data>
 - Visualization of final scoring pipeline (Experimental)
 - * In-GUI display of graph of feature engineering, modeling and ensembling steps of entire machine learning pipeline
 - * Addition to Autodoc
 - Time-Series:
 - * Ability to specify which features will be unavailable at test time for time-series experiments
 - * Custom user-provided train/validation splits (by start/end datetime for each split) for time-series experiments
 - * Back-testing metrics for time-series experiments (regression and classification, with and without lags) based on rolling windows (configurable number of windows)
 - MOJO:
 - * Java MOJO for FTRL
 - * PyTorch MOJO (C++/Py/R) for custom recipes based on BERT/DistilBERT NLP models (available upon request)
- Improvements:
 - Accuracy:
 - * Automatic pairwise interaction search (+,-,*,/.) for numeric features (“magic feature” finder)
 - * Improved accuracy for time series experiments with low interpretability

- * Improved leakage detection logic
- * Improved genetic algorithm heuristics for feature evolution (more exploration)
- Time-Series Recipes:
 - * Re-enable Test-time augmentation in Python scoring pipeline for time-series experiments
 - * Reduce default number of time-series rolling holdout predictions to same number as validation splits (but configurable)
- Computation:
 - * Faster feature evolution part for non-time-series experiments with single final model
 - * Faster binary imbalanced models for very high class imbalance by limiting internal number of re-sampling bags
 - * Faster feature selection
 - * Enable GPU support for ImbalancedXGBoostGBMModel
 - * Improved speed for importing multiple files at once
 - * Faster automatic determination of time series properties
 - * Enable use of XGBoost models on large datasets if low enough accuracy settings, expose dataset size limits in expert settings
 - * Reduced memory usage for all experiments
 - * Faster creation of holdout predictions for time-series experiments (Shapley values are now computed by MLi on demand by default)
- UX Improvements:
 - * Added ability to rename datasets
 - * Added search bar for expert settings
 - * Show traces for long-running experiments
 - * All experiments create a MOJO (if possible, set to ‘auto’)
 - * All experiments create a pipeline visualization
 - * By default, all experiments (iid and time series) have holdout predictions on training data and a full set of metrics for final model
- Documentation Updates:
 - Updated steps for enabling GPU persistence mode
 - Added information about deprecated NVIDIA functions
 - Improved documentation for enabling LDAP authentication
 - Added information about changing the column type in datasets
 - Updated list of experiment artifacts available in an experiment summary
 - Added steps describing how to expose ports on Docker for the REST service deployment within the Driverless AI Docker container
 - Added an example showing how to run an experiment with a custom transform recipe
 - Improved the FAQ for setting up TLS/SSL

- Added FAQ describing issues that can occur when attempting **Import Folder as File** with a data connector on Windows
- Bug Fixes:
 - Allow brain restart/refit to accept unscored previous pipelines
 - Fix actual vs predicted labeling for diagnostics of regression model
 - Fix MOJO for TensorFlow for non target transformers other than identity
 - Fix column type detection for Excel files
 - Allow experiments with default expert settings to have a MOJO
 - Various bug fixes

1.14 Version 1.8.0 (Oct 3, 2019)

Available here

- Improve speed and memory usage for feature engineering
- Improve speed of leakage and shift detection, and improve accuracy
- Improve speed of AutoVis under high system load
- Improve speed for experiments with large user-given validation data
- Improve accuracy of ensembles for regression problems
- Improve creation of Autoreport (only one background job per experiment)
- Improve sampling techniques for ImbalancedXGBoost and ImbalancedLightGBM models, and disable them by default since can be slower
- Add Python/R/C++ MOJO support for FTRL and RandomForest
- Add native categorical handling for LightGBM in CPU mode
- Add monotonicity constraints support for LightGBM
- Add Isolation Forest Anomaly Score transformer (outlier detection)
- Re-enable One-Hot-Encoding for GLM models
- Add lexicographical label encoding (disabled by default)
- Add ability to further train user-provided pretrained embeddings for TensorFlow NLP transformers, in addition to fine-tuning the rest of the neural network graph
- Add timeout for BYOR acceptance tests
- Add log and notifications for large shifts in final model variable importances compared to tuning model
- Add more expert control over time series feature engineering
- Add ability for recipes to be uploaded in bulk as entire (or part of) github repository or as links to python files on page
- Allow missing values in fold column
- Add support for feature brain when starting “New Model With Same Parameters” of a model that was previously restarted

- Add support for toggling whether additional features are to be included in pipeline during “Retrain Final Pipeline”
- Limit experiment runtime to one day by default (approximately enforced, can be configured in Expert Settings -> Experiment or config.toml ‘max_runtime_minutes’)
- Add support for importing pickled Pandas frames (.pkl)
- MLI updates:
 - Show holdout predictions and test set predictions (if applicable) in MLI TS for both metric and actual vs. predicted charts
 - Add ability to download group metrics in MLI TS
 - Add ability to zoom into charts in MLI TS
 - Add ability to use column not used in DAI model as a k-LIME cluster column in MLI
 - Add ability to view original and transformed DAI model-based feature importance in MLI
 - Add ability to view Shapley importance for original features
 - Add ability to view permutation importance for a DAI model when the config option *autodoc_include_permutation_feature_importance* is set to *on*
 - Fixed bug in binary Disparate Impact Analysis, which caused incorrect calculations amongst several metrics (ones using false positives and true negatives in the numerator)
- Disable NLP TensorFlow transformers by default (enable in NLP expert settings by switching to “on”)
- Reorganize expert settings, add tab for feature engineering
- Experiment now informs if aborted by user, system or server restart
- Reduce load of all tasks launched by server, giving priority to experiments to use cores
- Add experiment summary files to aborted experiment logs
- Add warning when ensemble has models that reach limit of max iterations despite early stopping, with learning rate controls in expert panel to control.
- Improve progress reporting
- Allow disabling of H2O recipe server for scoring if not using custom recipes (to avoid Java dependency)
- Fix RMSPE scorer
- Fix recipes error handling when uploading via URL
- Fix Autoreport being spawned anytime GUI was on experiment page, overloading the system with forks from the server
- Fix time-out for Autoreport PDP calculations, so completes more quickly
- Fix certain config settings to be honored from GUI expert settings (woe_bin_list, ohe_bin_list, text_gene_max_ngram, text_gene_dim_reduction_choice, tensorflow_max_epochs_nlp, tensorflow_nlp_pretrained_embeddings_file_path, holiday_country), previously were only honored when provided at startup time
- Fix column type for additional columns during scored test set download
- Fix GUI incorrectly converting time for forecast horizon in TS experiments
- Fix calculation of correlation for string columns in AutoVis
- Fix download for R MOJO runtime

- Fix parameters for LightGBM RF mode
- Fix dart parameters for LightGBM and XGBoost
- Documentation updates:
 - Included more information in the Before You Begin Installing or Upgrading topic to help making installations and upgrades go more smoothly
 - Added topic describing how to choose between the AWS Community and AWS Marketplace AMIs
 - Added information describing how to retrieve the MOJO2 Javadoc
 - Updated Python client examples to work with Driverless AI 1.7.x releases
 - Updated documentation for new features, expert settings, MLI plots, etc.
- Backward/Forward compatibility:
 - Models built in 1.8.0 will remain supported in versions 1.8.x
 - Models built in 1.7.1 are not deprecated and should continue to work (best effort is made to preserve MOJO and Autoreport creation, MLI, scoring, etc.)
 - 1.8.0 upgraded to scipy version 1.3.1 to support newer custom recipes. This might deprecate custom recipes that depend on scipy version 1.2.2 (and experiments using them) and might require re-import of those custom recipes. Previously built Python scoring pipelines will continue to work.
 - Models built in 1.7.0 or earlier will be deprecated
- Various bug fixes

1.15 Version 1.7.1 (Aug 19, 2019)

Available here

- Added two new models with internal sampling techniques for imbalanced binary classification problems: ImbalancedXGBoost and ImbalancedLightGBM
- Added support for rolling-window based predictions for time-series experiments (2 options: test-time augmentation or re-fit)
- Added support for setting logical column types for a dataset (to override type detection during experiments)
- Added ability to set experiment name at start of experiment
- Added leakage detection for time-series problems
- Added JDBC connector
- MOJO updates:
 - Added Python/R/C++ MOJO support for TensorFlow model
 - Added Python/R/C++ MOJO support for TensorFlow NLP transformers: TextCNN, CharCNN, BiGRU, including any pretrained embeddings if provided
 - Reduced memory usage for MOJO creation
 - Increased speed of MOJO creation
 - Configuration options for MOJO and Python scoring pipelines now have 3-way toggle: “on”/“off”/“auto”
- MLI updates:
 - Added disparate impact analysis (DIA) for MLI

- Allow MLI scoring pipeline to be built for datasets with column names that need to be sanitized
- Date-aware binning for partial dependence and ICE in MLI
- Improved generalization performance for time-series modeling with regularization techniques for lag-based features
- Improved “predicted vs actual” plots for regression problems (using adaptive point sizes)
- Fix bug in datatable for manipulations of string columns larger than 2GB
- Fixed download of predictions on user-provided validation data
- Fix bug in time-series test-time augmentation (work-around was to include entire training data in test set)
- Honor the expert settings flag to enable detailed traces (disable again by default)
- Various bug fixes

1.16 Version 1.6.4 LTS (Aug 19, 2019)

Available [here](#)

- ML Core updates:
 - Speed up schema detection
 - DAI now drops rows with missing values when diagnosing regression problems
 - Speed up column type detection
 - Fixed growth of individuals
 - Fixed n_jobs for predict
 - Target column is no longer included in predictors for skewed datasets
 - Added an option to prevent users from downloading data files locally
 - Improved UI split functionality
 - A new “max_listing_items” config option to limit the number of items fetched in listing pages
- Model Ops updates:
 - MOJO runtime upgraded to version 2.1.3 which supports perpetual MOJO pipeline
 - Upgraded deployment templates to version matching MOJO runtime version
- MLI updates:
 - Fix to MLI schema builder
 - Fix parsing of categorical reason codes
 - Added ability to handle integer time column
- Various bug fixes

1.17 Version 1.7.0 (Jul 7, 2019)

Available here

- Support for Bring Your Own Recipe (BYOR) for transformers, models (algorithms) and scorers
- Added protobuf-based MOJO scoring runtime libraries for Python, R and Java (standalone, low-latency)
- Added local REST server as one-click deployment option for MOJO scoring pipeline, in addition to AWS Lambda endpoint
- Added R client package, in addition to Python client
- Added Project workspace to group datasets and experiments and to visually compare experiments and create leaderboards
- Added download of imported datasets as .csv
- Recommendations for columnar transformations in AutoViz
- Improved scalability and performance
- Ability to provide max. runtime for experiments
- Create MOJO scoring pipeline by default if the experiment configuration allows (for convenience, enables local/cloud deployment options without user input)
- Support for user provided pre-trained embeddings for TensorFlow NLP models
- Support for holdout splits lacking some target classes (can happen when a fold column is provided)
- MLI updates:
 - Added residual plot for regression problems (keeping all outliers intact)
 - Added confusion matrix as default metric display for multinomial problems
 - Added Partial Dependence (PD) and Individual Conditional Expectation (ICE) plots for Driverless.ai models in MLI GUI
 - Added ability to search by ID column in MLI GUI
 - Added ability to run MLI PD/ICE on all features
 - Added ability to handle multiple observations for a single time column in MLI TS by taking the mean of the target and prediction where applicable
 - Added ability to handle integer time column in MLI TS
 - MLI TS will use train holdout predictions if there is no test set provided
- Faster import of files with “%Y%m%d” and “%Y%m%d%H%M” time format strings, and files with lots of text strings
- Fix units for RMSPE scorer to be a percentage (multiply by 100)
- Allow non-positive outcomes for MAPE and SMAPE scorers
- Improved listing in GUI
- Allow zooming in GUI
- Upgrade to TensorFlow 1.13.1 and CUDA 10 (and CUDA is part of the distribution now, to simplify installation)
- Add CPU-support for TensorFlow on PPC
- Documentation updates:

- Added documentation for new features including
 - * Projects
 - * Custom Recipes
 - * C++ MOJO Scoring Pipelines
 - * R Client API
 - * REST Server Deployment
- Added information about variable importance values on the experiments page
- Updated documentation for Expert Settings
- Updated “Tips n Tricks” with new Scoring Pipeline tips
- Various bug fixes

1.18 Version 1.6.3 LTS (June 14, 2019)

Available here

- Included an Audit log feature
- Fixed support for decimal types for parquet files in MOJO
- Autodoc can order PDP/ICE by feature importance
- Session Management updates
- Upgraded datatable
- Improved reproducibility
- Model diagnostics now uses a weight column
- MLI can build surrogate models on all the original features or on all the transformed features that DAI uses
- Internal server cache now respects usernames
- Fixed an issue with time series settings
- Fixed an out of memory error when loading a MOJO
- Fixed Python scoring package for TensorFlow
- Added OpenID configurations
- Documentation updates:
 - Updated the list of artifacts available in the Experiment Summary
 - Clarified language in the documentation for unsupported (but available) features
 - For the Terraform requirement in deployments, clarified that only Terraform versions in the 0.11.x release are supported, and specifically 0.11.10 or greater
 - Fixed link to the Miniconda installation instructions
- Various bug fixes

1.19 Version 1.6.2 LTS (May 10, 2019)

Available here

- This version provides PPC64le artifacts
- Improved stability of datatable
- Improved path filtering in the file browser
- Fixed units for RMSPE scorer to be a percentage (multiply by 100)
- Fixed segmentation fault on Ubuntu 18 with installed font package
- Fixed IBM Spectrum Conductor authentication
- Fixed handling of EC2 machine credentials
- Fixed of Lag transformer configuration
- Fixed KDB and Snowflake Error Reporting
- Gradually reduce number of used workers for column statistics computation in case of failure.
- Hide default Tornado header exposing used version of Tornado
- Documentation updates:
 - Added instructions for installing via AWS Marketplace
 - Improved documentation for installing via Google Cloud
 - Improved FAQ documentation
 - Added Data Sampling documentation topic
- Various bug fixes

1.20 Version 1.6.1.1 LTS (Apr 24, 2019)

Available here

- Fix in AWS role handling.

1.21 Version 1.6.1 LTS (Apr 18, 2019)

Available here

- Several fixes for MLI (partial dependence plots, Shapley values)
- Improved documentation for model deployment, time-series scoring, AutoVis and FAQs

1.22 Version 1.6.0 LTS (Apr 5, 2019)

Private build only.

- Fixed import of string columns larger than 2GB
- Fixed AutoViz crashes on Windows
- Fixed quantile binning in MLI
- Plot global absolute mean Shapley values instead of global mean Shapley values in MLI
- Improvements to PDP/ICE plots in MLI
- Validated Terraform version in AWS Lambda deployment
- Added support for NULL variable importance in AutoDoc
- Made Variable Importance table size configurable in AutoDoc
- Improved support for various combinations of data import options being enabled/disabled
- CUDA is now part of distribution for easier installation
- Security updates:
 - Enforced SSL settings to be honored for all h2oai_client calls
 - Added config option to prevent using LocalStorage in the browser to cache information
 - Upgraded Tornado server version to 5.1.1
 - Improved session expiration and autologout functionality
 - Disabled access to Driverless AI data folder in file browser
 - Provided an option to filter content that is shown in the file browser
 - Use login name for HDFS impersonation instead of predefined name
 - Disabled autocomplete in login form
- Various bug fixes

1.23 Version 1.5.4 (Feb 24, 2019)

Available here

- Speed up calculation of column statistics for date/datetime columns using certain formats (now uses ‘max_rows_col_stats’ parameter)
- Added computation of standard deviation for variable importances in experiment summary files
- Added computation of shift of variable importances between feature evolution and final pipeline
- Fix link to MLI Time-Series experiment
- Fix display bug for iteration scores for long experiments
- Fix display bug for early finish of experiment for GLM models
- Fix display bug for k-LIME when target is skewed
- Fix display bug for forecast horizon in MLI for Time-Series
- Fix MLI for Time-Series for single time group column

- Fix in-server scoring of time-series experiments created in 1.5.0 and 1.5.1
- Fix OpenBLAS dependency
- Detect disabled GPU persistence mode in Docker
- Reduce disk usage during TensorFlow NLP experiments
- Reduce disk usage of aborted experiments
- Refresh reported size of experiments during start of application
- Disable TensorFlow NLP transformers by default to speed up experiments (can enable in expert settings)
- Improved progress percentage shown during experiment
- Improved documentation (upgrade on Windows, how to create the simplest model, DTap connectors, etc.)
- Various bug fixes

1.24 Version 1.5.3 (Feb 8, 2019)

Available here

- Added support for splitting datasets by time via time column containing date, datetime or integer values
- Added option to disable file upload
- Require authentication to download experiment artifacts
- Automatically drop predictor columns from training frame if not found in validation or test frame and warn
- Improved performance by using physical CPU cores only (configurable in config.toml)
- Added option to not show inactive data connectors
- Various bug fixes

1.25 Version 1.5.2 (Feb 2, 2019)

Available here

- Added world-level bidirectional GRU Tensorflow models for NLP features
- Added character-level CNN Tensorflow models for NLP features
- Added support to import multiple individual datasets at once
- Added support for holdout predictions for time-series experiments
- Added support for regression and multinomial classification for FTRL (in addition to binomial classification)
- Improved scoring for time-series when test data contains actual target values (missing target values will be predicted)
- Reduced memory usage for LightGBM models
- Improved performance for feature engineering
- Improved speed for TensorFlow models
- Improved MLI GUI for time-series problems
- Fix final model fold splits when fold_column is provided

- Various bug fixes

1.26 Version 1.5.1 (Jan 22, 2019)

Available here

- Fix MOJO for GLM
- Add back .csv file of experiment summary
- Improve collection of pipeline timing artifacts
- Clean up Docker tag

1.27 Version 1.5.0 (Jan 18, 2019)

Available here

- Added model diagnostics (interactive model metrics on new test data incl. residual analysis for regression)
- Added FTRL model (Follow The Regularized Leader)
- Added Kolmogorov-Smirnov metric (degree of separation between positives and negatives)
- Added ability to retrain (only) the final model on new data
- Added one-hot encoding for low-cardinality categorical features, for GLM
- Added choice between 32-bit (now default) and 64-bit precision
- Added system information (CPU, GPU, disk, memory, experiments)
- Added support for time-series data with many more time gaps, and with weekday-only data
- Added one-click deployment to Amazon Lambda
- Added ability to split datasets randomly, with option to stratify by target column or group by fold column
- Added support for OpenID authentication
- Added connector for BlueData
- Improved responsiveness of the GUI under heavy load situations
- Improved speed and reduce memory footprint of feature engineering
- Improved performance for RuleFit models and enable GPU and multinomial support
- Improved auto-detection of temporal frequency for time-series problems
- Improved accuracy of final single model if external validation provided
- Improved final pipeline if external validation data is provided (add ensembling)
- Improved k-LIME in MLI by using original features deemed important by DAI instead of all original features
- Improved MLI by using 3-fold CV by default for all surrogate models
- Improved GUI for MLI time series (integrated help, better integration)
- Added ability to view MLI time series logs while MLI time series experiment is running
- PDF version of the Automatic Report (AutoDoc) is now replaced by a Word version
- Various bug fixes (GLM accuracy, UI slowness, MLI UI, AutoVis)

1.28 Version 1.4.2 (Dec 3, 2018)

Available here

- Support for IBM Power architecture
- Speed up training and reduce size of final pipeline
- Reduced resource utilization during training of final pipeline
- Display test set metrics (ROC, ROCPR, Gains, Lift) in GUI in addition to validation metrics (if test set provided)
- Show location of best threshold for Accuracy, MCC and F1 in ROC curves
- Add relative point sizing for scatter plots in AutoVis
- Fix file upload and add model checkpointing in python client API
- Various bug fixes

1.29 Version 1.4.1 (Nov 11, 2018)

Available here

- Improved integration of MLI for time-series
- Reduced disk and memory usage during final ensemble
- Allow scoring and transformations on previously imported datasets
- Enable checkpoint restart for unfinished models
- Add startup checks for OpenCL platforms for LightGBM on GPUs
- Improved feature importances for ensembles
- Faster dataset statistics for date/datetime columns
- Faster MOJO batch scoring
- Fix potential hangs
- Fix ‘not in list’ error in MOJO
- Fix NullPointerException in MLI
- Fix outlier detection in AutoVis
- Various bug fixes

1.30 Version 1.4.0 (Oct 27, 2018)

Available here

- Enable LightGBM by default (now with MOJO)
- LightGBM tuned for GBM decision trees, Random Forest (rf), and Dropouts meet Multiple Additive Regression Trees (dart)
- Add ‘isHoliday’ feature for time columns
- Add ‘time’ column type for date/datetime columns in data preview

- Add support for binary datatable file ingest in .jay format
- Improved final ensemble (each model has its own feature pipeline)
- Automatic smart checkpointing (feature brain) from prior experiments
- Add kdb+ connector
- Feature selection of original columns for data with many columns to handle >>100 columns
- Improved time-series recipe (multiple validation splits, better logic)
- Improved performance of AutoVis
- Improved date detection logic (now detects %Y%m%d and %Y-%m date formats)
- Automatic fallback to CPU mode if GPU runs out of memory (for XGBoost, GLM and LightGBM)
- No longer require header for validation and testing datasets if data types match
- No longer include text columns for data shift detection
- Add support for time-series models in MLI (including ability to select time-series groups)
- Add ability to download MLI logs from MLI experiment page (includes both Python and Java logs)
- Add ability to view MLI logs while MLI experiment is running (Python and Java logs)
- Add ability to download LIME and Shapley reason codes from MLI page
- Add ability to run MLI on transformed features
- Display all variables for MLI variable importance for both DAI and surrogate models in MLI summary
- Include variable definitions for DAI variable importance list in MLI summary
- Fix Gains/Lift charts when observations weights are given
- Various bug fixes

1.31 Version 1.3.1 (Sep 12, 2018)

Available here

- Fix ‘Broken pipe’ failures for TensorFlow models
- Fix time-series problems with categorical features and interpretability >= 8
- Various bug fixes

1.32 Version 1.3.0 (Sep 4, 2018)

Available here

- Added LightGBM models - now have [XGBoost, LightGBM, GLM, TensorFlow, RuleFit]
- Added TensorFlow NLP recipe based on CNN Deeplearning models (sentiment analysis, document classification, etc.)
- Added MOJO for GLM
- Added detailed confusion matrix statistics
- Added more expert settings

- Improved data exploration (columnar statistics and row-based data preview)
- Improved speed of feature evolution stage
- Improved speed of GLM
- Report single-pass score on external validation and test data (instead of bootstrap mean)
- Reduced memory overhead for data processing
- Reduced number of open files - fixes ‘Bad file descriptor’ error on Mac/Docker
- Simplified Python client API
- Query any data point in the MLI UI from the original dataset due to “on-demand” reason code generation
- Enhanced k-means clustering in k-LIME by only using a subset of features. See klime_technique for more information.
- Report k-means centers for k-LIME in MLI summary for better cluster interpretation
- Improved MLI experiment listing details
- Various bug fixes

1.33 Version 1.2.2 (July 5, 2018)

Available here

- MOJO Java scoring pipeline for time-series problems
- Multi-class confusion matrices
- AUCMACRO Scorer: Multi-class AUC via macro-averaging (in addition to the default micro-averaging)
- Expert settings (configuration override) for each experiment from GUI and client APIs.
- Support for HTTPS
- Improved downsampling logic for time-series problems (if enabled through accuracy knob settings)
- LDAP readonly access to Active Directory
- Snowflake data connector
- Various bug fixes

1.34 Version 1.2.1 (June 26, 2018)

- Added LIME-SUP (alpha) to MLI as alternative to k-LIME (local regions are defined by decision tree instead of k-means)
- Added RuleFit model (alpha), now have [GBM, GLM, TensorFlow, RuleFit] - TensorFlow and RuleFit are disabled by default
- Added Minio (private cloud storage) connector
- Added support for importing folders from S3
- Added ‘Upload File’ option to ‘Add Dataset’ (in addition to drag & drop)
- Predictions for binary classification problems now have 2 columns (probabilities per class), for consistency with multi-class

- Improved model parameter tuning
- Improved feature engineering for time-series problems
- Improved speed of MOJO generation and loading
- Improved speed of time-series related automatic calculations in the GUI
- Fixed potential rare hangs at end of experiment
- No longer require internet to run MLI
- Various bug fixes

1.35 Version 1.2.0 (June 11, 2018)

- Time-Series recipe
- Low-latency standalone MOJO Java scoring pipelines (now beta)
- Enable Elastic Net Generalized Linear Modeling (GLM) with lambda search (and GPU support), for interpretability ≥ 6 and accuracy ≤ 5 by default (alpha)
- Enable TensorFlow (TF) Deep Learning models (with GPU support) for interpretability=1 and/or multi-class models (alpha, enable via config.toml)
- Support for pre-tuning of [GBM, GLM, TF] models for picking best feature evolution model parameters
- Support for final ensemble consisting of mix of [GBM, GLM, TF] models
- Automatic Report (AutoDoc) in PDF and Markdown format as part of summary zip file
- Interactive tour (assistant) for first-time users
- MLI now runs on experiments from previous releases
- Surrogate models in MLI now use 3 folds by default
- Improved small data recipe with up to 10 cross-validation folds
- Improved accuracy for binary classification with imbalanced data
- Additional time-series transformers for interactions and aggregations between lags and lagging of non-target columns
- Faster creation of MOJOS
- Progress report during data ingest
- Normalize binarized multi-class confusion matrices by class count (global scaling factor)
- Improved parsing of boolean environment variables for configuration
- Various bug fixes

1.36 Version 1.1.6 (May 29, 2018)

- Improved performance for large datasets
- Improved speed and user interface for MLI
- Improved accuracy for binary classification with imbalanced data
- Improved generalization estimate for experiments with given validation data
- Reduced size of experiment directories
- Support for Parquet files
- Support for bzip2 compressed files
- Added Data preview in UI: ‘Describe’
- No longer add ID column to holdout and test set predictions for simplicity
- Various bug fixes

1.37 Version 1.1.4 (May 17, 2018)

- Native builds (RPM/DEB) for 1.1.3

1.38 Version 1.1.3 (May 16, 2018)

- Faster speed for systems with large CPU core counts
- Faster and more robust handling of user-specified missing values for training and scoring
- Same validation scheme for feature engineering and final ensemble for high enough accuracy
- MOJO scoring pipeline for text transformers
- Fixed single-row scoring in Python scoring pipeline (broken in 1.1.2)
- Fixed default scorer when experiment is started too quickly
- Improved responsiveness for time-series GUI
- Improved responsiveness after experiment abort
- Improved load balancing of memory usage for multi-GPU XGBoost
- Improved UI for selection of columns to drop
- Various bug fixes

1.39 Version 1.1.2 (May 8, 2018)

- Support for automatic time-series recipe (alpha)
- Now using Generalized Linear Model (GLM) instead of XGBoost (GBM) for interpretability 10
- Added experiment preview with runtime and memory usage estimation
- Added MER scorer (Median Error Rate, Median Abs. Percentage Error)
- Added ability to use integer column as time column
- Speed up type enforcement during scoring
- Support for reading ARFF file format (alpha)
- Quantile Binning for MLI
- Various bug fixes

1.40 Version 1.1.1 (April 23, 2018)

- Support string columns larger than 2GB

1.41 Version 1.1.0 (April 19, 2018)

- AWS/Azure integration (hourly cloud usage)
- Bug fixes for MOJO pipeline scoring (now beta)
- Google Cloud storage and BigQuery (alpha)
- Speed up categorical column stats computation during data import
- Further improved memory management on GPUs
- Improved accuracy for MAE scorer
- Ability to build scoring pipelines on demand (if not enabled by default)
- Additional target transformer for regression problems $\text{sqrt}(\text{sqrt}(x))$
- Add GLM models as candidates for interpretability=10 (alpha, disabled by default)
- Improved performance of native builds (RPM/DEB)
- Improved estimation of error bars
- Various bug fixes

1.42 Version 1.0.30 (April 5, 2018)

- Speed up MOJO pipeline creation and disable MOJO by default (still alpha)
- Improved memory management on GPUs
- Support for optional 32-bit floating-point precision for reduced memory footprint
- Added logging of test set scoring and data transformations
- Various bug fixes

1.43 Version 1.0.29 (April 4, 2018)

- If MOJO fails to build, no MOJO will be available, but experiment can still succeed

1.44 Version 1.0.28 (April 3, 2018)

- (Non-docker) RPM installers for RHEL7/CentOS7/SLES 12 with systemd support

1.45 Version 1.0.27 (March 31, 2018)

- MOJO scoring pipeline for Java standalone cross-platform low-latency scoring (alpha)
- Various bug fixes

1.46 Version 1.0.26 (March 28, 2018)

- Improved performance and reduced memory usage for large datasets
- Improved performance for F0.5, F2 and accuracy
- Improved performance of MLI
- Distribution shift detection now also between validation and test data
- Batch scoring example using datatable
- Various enhancements for AutoVis (outliers, parallel coordinates, log file)
- Various bug fixes

1.47 Version 1.0.25 (March 22, 2018)

- New scorers for binary/multinomial classification: F0.5, F2 and accuracy
- Precision-recall curve for binary/multinomial classification models
- Plot of actual vs predicted values for regression problems
- Support for excluding feature transformations by operation type
- Support for reading binary file formats: datatable and Feather
- Improved multi-GPU memory load balancing
- Improved display of initial tuning results
- Reduced memory usage during creation of final model
- Fixed several bugs in creation of final scoring pipeline
- Various UI improvements (e.g., zooming on iteration scoreboard)
- Various bug fixes

1.48 Version 1.0.24 (March 8, 2018)

- Fix test set scoring bug for data with an ID column (introduced in 1.0.23)
- Allow renaming of MLI experiments
- Ability to limit maximum number of cores used for datatable
- Print validation scores and error bars across final ensemble model CV folds in logs
- Various UI improvements
- Various bug fixes

1.49 Version 1.0.23 (March 7, 2018)

- Support for Gains and Lift curves for binomial and multinomial classification
- Support for multi-GPU single-model training for large datasets
- Improved recipes for large datasets (faster and less memory/disk usage)
- Improved recipes for text features
- Increased sensitivity of interpretability setting for feature engineering complexity
- Disable automatic time column detection by default to avoid confusion
- Automatic column type conversion for test and validation data, and during scoring
- Improved speed of MLI
- Improved feature importances for MLI on transformed features
- Added ability to download each MLI plot as a PNG file
- Added support for dropped columns and weight column to MLI stand-alone page
- Fix serialization of bytes objects larger than 4 GiB

- Fix failure to build scoring pipeline with ‘command not found’ error
- Various UI improvements
- Various bug fixes

1.50 Version 1.0.22 (Feb 23, 2018)

- Fix CPU-only mode
- Improved robustness of datatable CSV parser

1.51 Version 1.0.21 (Feb 21, 2018)

- Fix MLi GUI scaling issue on Mac
- Work-around segfault in truncated SVD scipy backend
- Various bug fixes

1.52 Version 1.0.20 (Feb 17, 2018)

- HDFS/S3/Excel data connectors
- LDAP/PAM/Kerberos authentication
- Automatic setting of default values for accuracy / time / interpretability
- Interpretability: per-observation and per-feature (signed) contributions to predicted values in scoring pipeline
- Interpretability setting now affects feature engineering complexity and final model complexity
- Standalone MLi scoring pipeline for Python
- Time setting of 1 now runs for only 1 iteration
- Early stopping of experiments if convergence is detected
- ROC curve display for binomial and multinomial classification, with confusion matrices and threshold/F1/MCC display
- Training/Validation/Test data shift detectors
- Added AUCPR scorer for multinomial classification
- Improved handling of imbalanced binary classification problems
- Configuration file for runtime limits such as cores/memory/harddrive (for admins)
- Various GUI improvements (ability to rename experiments, re-run experiments, logs)
- Various bug fixes

1.53 Version 1.0.19 (Jan 28, 2018)

- Fix hang during final ensemble (accuracy ≥ 5) for larger datasets
- Allow scoring of all models built in older versions ($\geq 1.0.13$) in GUI
- More detailed progress messages in the GUI during experiments
- Fix scoring pipeline to only use relative paths
- Error bars in model summary are now $\pm 1\text{-stddev}$ (instead of 2-stddev)
- Added RMSPE scorer (RMS Percentage Error)
- Added SMAPE scorer (Symmetric Mean Abs. Percentage Error)
- Added AUCPR scorer (Area under Precision-Recall Curve)
- Gracefully handle inf/-inf in data
- Various UI improvements
- Various bug fixes

1.54 Version 1.0.18 (Jan 24, 2018)

- Fix migration from version 1.0.15 and earlier
- Confirmation dialog for experiment abort and data/experiment deletion
- Various UI improvements
- Various AutoVis improvements
- Various bug fixes

1.55 Version 1.0.17 (Jan 23, 2018)

- Fix migration from version 1.0.15 and earlier (partial, for experiments only)
- Added model summary download from GUI
- Restructured and renamed logs archive, and add model summary to it
- Fix regression in AutoVis in 1.0.16 that led to slowdown
- Various bug fixes

1.56 Version 1.0.16 (Jan 22, 2018)

- Added support for validation dataset (optional, instead of internal validation on training data)
- Standard deviation estimates for model scores ($\pm 1\text{-std.dev.}$)
- Computation of all applicable scores for final models (in logs only for now)
- Standard deviation estimates for MLI reason codes ($\pm 1\text{-std.dev.}$) when running in stand-alone mode
- Added ability to abort MLI job

- Improved final ensemble performance
- Improved outlier visualization
- Updated H2O-3 to version 3.16.0.4
- More readable experiment names
- Various speedups
- Various bug fixes

1.57 Version 1.0.15 (Jan 11, 2018)

- Fix truncated per-experiment log file
- Various bug fixes

1.58 Version 1.0.14 (Jan 11, 2018)

- Improved performance

1.59 Version 1.0.13 (Jan 10, 2018)

- Improved estimate of generalization performance for final ensemble by removing leakage from target encoding
- Added API for re-fitting and applying feature engineering on new (potentially larger) data
- Remove access to pre-transformed datasets to avoid unintended leakage issues downstream
- Added mean absolute percentage error (MAPE) scorer
- Enforce monotonicity constraints for binary classification and regression models if interpretability ≥ 6
- Use squared Pearson correlation for R² metric (instead of coefficient of determination) to avoid negative values
- Separated HTTP and TCP scoring pipeline examples
- Reduced size of h2oai_client wheel
- No longer require weight column for test data if it was provided for training data
- Improved accuracy of final modeling pipeline
- Include H2O-3 logs in downloadable logs.zip
- Updated H2O-3 to version 3.16.0.2
- Various bug fixes

1.60 Version 1.0.11 (Dec 12, 2017)

- Faster multi-GPU training, especially for small data
- Increase default amount of exploration of genetic algorithm for systems with fewer than 4 GPUs
- Improved accuracy of generalization performance estimate for models on small data (< 100k rows)
- Faster abort of experiment
- Improved final ensemble meta-learner
- More robust date parsing
- Various bug fixes

1.61 Version 1.0.10 (Dec 4, 2017)

- Tool tips and link to documentation in parameter settings screen
- Faster training for multi-class problems with > 5 classes
- Experiment summary displayed in GUI after experiment finishes
- Python Client Library downloadable from the GUI
- Speedup for Maxwell-based GPUs
- Support for multinomial AUC and Gini scorers
- Add MCC and F1 scorers for binomial and multinomial problems
- Faster abort of experiment
- Various bug fixes

1.62 Version 1.0.9 (Nov 29, 2017)

- Support for time column for causal train/validation splits in time-series datasets
- Automatic detection of the time column from temporal correlations in data
- MLi improvements, dedicated page, selection of datasets and models
- Improved final ensemble meta-learner
- Test set score now displayed in experiment listing
- Original response is preserved in exported datasets
- Various bug fixes

1.63 Version 1.0.8 (Nov 21, 2017)

- Various bug fixes

1.64 Version 1.0.7 (Nov 17, 2017)

- Sharing of GPUs between experiments - can run multiple experiments at the same time while sharing GPU resources
- Persistence of experiments and data - can stop and restart the application without loss of data
- Support for weight column for optional user-specified per-row observation weights
- Support for fold column for user-specified grouping of rows in train/validation splits
- Higher accuracy through model tuning
- Faster training - overall improvements and optimization in model training speed
- Separate log file for each experiment
- Ability to delete experiments and datasets from the GUI
- Improved accuracy for regression tasks with very large response values
- Faster test set scoring - Significant improvements in test set scoring in the GUI
- Various bug fixes

1.65 Version 1.0.5 (Oct 24, 2017)

- Only display scorers that are allowed
- Various bug fixes

1.66 Version 1.0.4 (Oct 19, 2017)

- Improved automatic type detection logic
- Improved final ensemble accuracy
- Various bug fixes

1.67 Version 1.0.3 (Oct 9, 2017)

- Various speedups
- Results are now reproducible
- Various bug fixes

1.68 Version 1.0.2 (Oct 5, 2017)

- Improved final ensemble accuracy
- Weight of Evidence features added
- Various bug fixes

1.69 Version 1.0.1 (Oct 4, 2017)

- Improved speed of final ensemble
- Various bug fixes

1.70 Version 1.0.0 (Sep 24, 2017)

- Initial stable release

INTRODUCTION TO H2O DRIVERLESS AI

H2O Driverless AI is a high-performance, GPU-enabled, client-server application for the rapid development and deployment of state-of-the-art predictive analytics models. It reads tabular data from various sources and automates data visualization, grand-master level automatic feature engineering, model validation (overfitting and leakage prevention), model parameter tuning, model interpretability and model deployment. H2O Driverless AI is currently targeting common regression, binomial classification, and multinomial classification applications including loss-given-default, probability of default, customer churn, campaign response, fraud detection, anti-money-laundering, and predictive asset maintenance models. It also handles time-series problems for individual or grouped time-series such as weekly sales predictions per store and department, with time-causal feature engineering and validation schemes. The ability to model unstructured data is coming soon.

High-level capabilities:

- Client/server application for rapid experimentation and deployment of state-of-the-art supervised machine learning models
- User-friendly GUI
- Python and R client API
- Automatically creates machine learning modeling pipelines for highest predictive accuracy
- Automates data cleaning, feature selection, feature engineering, model selection, model tuning, ensembling
- Automatically creates stand-alone batch scoring pipeline for in-process scoring or client/server scoring via HTTP or TCP protocols in Python
- Automatically creates stand-alone (MOJO) low latency scoring pipeline for in-process scoring or client/server scoring via HTTP or TCP protocols, in C++ (with R and Python runtimes) and Java (runs anywhere)
- Multi-GPU and multi-CPU support for powerful workstations and NVidia DGX supercomputers
- Machine Learning model interpretation module with global and local model interpretation
- Automatic Visualization module
- Multi-user support
- Backward compatibility

Problem types supported:

- Regression (continuous target variable like age, income, price or loss prediction, time-series forecasting)
- Binary classification (0/1 or “N”/“Y”, for fraud prediction, churn prediction, failure prediction, etc.)
- Multinomial classification (“negative”/“neutral”/“positive” or 0/1/2/3 or 0.5/1.0/2.0 for categorical target variables, for prediction of membership type, next-action, product recommendation, sentiment analysis, etc.)

Data types supported:

- Tabular structured data, rows are observations, columns are fields/features/variables
- Numeric, categorical and textual fields
- Images
- Missing values are allowed
- i.i.d. (identically and independently distributed) data
- Time-series data with a single time-series (time flows across the entire dataset, not per block of data)
- Grouped time-series (e.g., sales per store per department per week, all in one file, with 3 columns for store, dept, week)
- Time-series problems with a gap between training and testing (i.e., the time to deploy), and a known forecast horizon (after which model has to be retrained)

Data types supported via custom recipes:

- Video
- Audio
- Graphs

Data sources supported:

- Local file system or NFS
- File upload from browser or Python client
- S3 (Amazon)
- Hadoop (HDFS)
- Azure Blob storage
- Blue Data Tap
- Google BigQuery
- Google Cloud storage
- kdb+
- Minio
- Snowflake
- JDBC
- Custom Data Recipe BYOR (Python, bring your own recipe)

File formats supported:

- Plain text formats of columnar data (.csv, .tsv, .txt)
- Compressed archives (.zip, .gz, .bz2)
- Excel
- Parquet
- Feather
- Python datatable (.jay)

2.1 Architecture

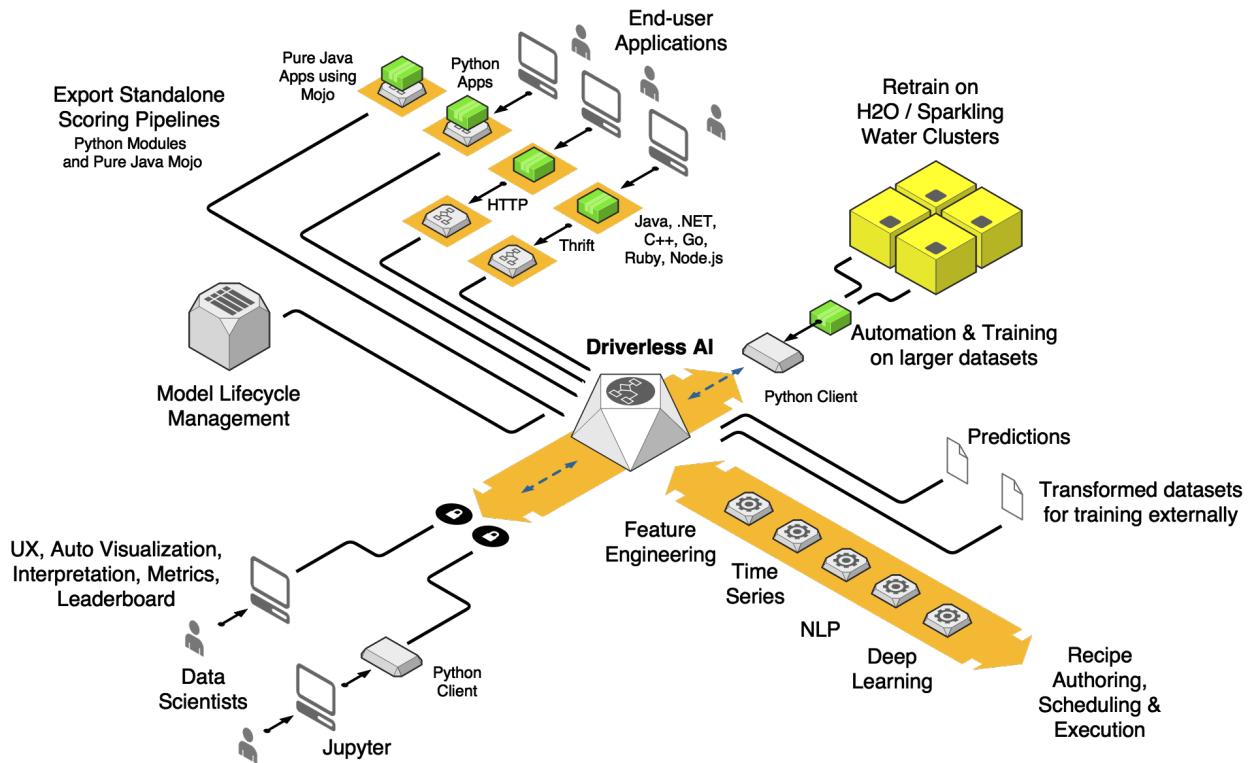


Fig. 1: DAI architecture

2.2 Roadmap

Driverless AI Roadmap			
	1.6 Apr 19	1.8 Oct 19	1.9 July 20
Driverless AI - Features	1.5 1.6 LTS	1.7 1.8 LTS	1.9 2.0 LTS
Kaggle Grandmaster Recipes for Automatic Machine Learning for Tabular Data, Time Series and Natural Language			
State of the Art Machine Learning			
Gradient Boosting (LightGBM, XGBoost), Random Forest, Linear Models (GLM), Deep Learning (MLP/CNN/LSTM), FTRL, RuleFit			
State of the Art Data Science			
Feature Selection, Feature Engineering, Feature Tuning, Model Selection, Model Tuning, Ensembles, Cross-Validation, Moving Windows, Back-Testing, Leak Detection, Data Shift Detection, Overfitting Prevention			
State of the Art Machine Learning Interpretability			
Explainable AI, Fairness, Shapley, Disparate Impact Analysis, Sensitivity Analysis (What-If), Partial Dependences			
Automatic Documentation (20+ page Word report, fully customizable customer templates)			
Automatic Visualization (by author of Grammar of Graphics)			
Data Connectors: NFS/HDFS/S3/Azure/GCS/BigQuery/kdb+, CSV/Excel/Parquet/Feather			
Hardware acceleration: Multi-Socket/Multi-Core CPUs (x86/PPC) and NVIDIA GPUs (Kepler/Tesla/Maxwell/Pascal/Volta/Ampere, DGX)			
Powered by Python/C++ datatabable backend for fast data manipulations https://h2oai.github.io/db-benchmark/			
Easy installation: On Premise: Native Installer (RPM/DEB/Tar-SH) or on any Cloud: AWS/Azure/GCP https://www.h2o.ai/download/			
Standalone Python Scoring Pipeline (Batch), Java Scoring Pipeline (MOJO, low latency)			
Authentication and Security (LDAP/Kerberos), Python client API			
BYOR - Bring Your Own Recipe https://github.com/h2oai/driverlessai-recipes/			
Custom Data Connectors, Custom Transformers, Custom Models, Custom Scorers - all in Python (150+ curated templates)			
Standalone C++ Scoring Pipeline (MOJO) with Python and R bindings, native support for TensorFlow/FTRL, Linux+Mac+PPC			
Pipeline Visualization, Project Workspace, Model Diagnostics, R client API, Custom Autoviz			
Authentication OpenID Connect (Single Sign-On), Secure Upload of Artifacts, Connectors: JDBC, Snowflake, MinIO, BlueData			
Kaggle Grandmaster Recipes for Computer Vision and NLP (BERT) based on TensorFlow and PyTorch Deep Learning, with MOJO support			GA
Job Queuing, Automatic Leaderboard Project, New MLI GUI/UX, Kernel Explainer, Custom Visualizations			GA
Prediction Intervals, Zeroinflated Models, Time Series De-Trending (+Epidemic SEIRD), Multi-Layer Feature Engineering, Data Prep Recipes			GA
Multi-Tenancy/Collaboration: Kubernetes (Enterprise Steam) + Authentication + Multi-Node Training + Model Storage + Model Ops			Preview
MLI Custom Recipes, Custom Problem Types (Image Segmentation, Unsupervised), Big Data Recipe for Training on Spark/H2O-3			WIP

Fig. 2: DAI roadmap

WHY DRIVERLESS AI?

Over the last several years, machine learning has become an integral part of many organizations' decision-making processes at various levels. With not enough data scientists to fill the increasing demand for data-driven business processes, H2O.ai offers Driverless AI, which automates several time consuming aspects of a typical data science workflow, including data visualization, feature engineering, predictive modeling, and model explanation.

H2O Driverless AI is a high-performance, GPU-enabled computing platform for automatic development and rapid deployment of state-of-the-art predictive analytics models. It reads tabular data from plain text sources and from a variety of external data sources, and it automates data visualization and the construction of predictive models.

Driverless AI also includes robust Machine Learning Interpretability (MLI), which incorporates a number of contemporary approaches to increase the transparency and accountability of complex models by providing model results in a human-readable format.

Driverless AI targets business applications such as loss-given-default, probability of default, customer churn, campaign response, fraud detection, anti-money-laundering, demand forecasting, and predictive asset maintenance models. (Or in machine learning parlance: common regression, binomial classification, and multinomial classification problems.)

Visit <https://www.h2o.ai/driverless-ai/> to download your free 21-day evaluation copy.

How do you frame business problems in a data set for Driverless AI?

The data that is read into Driverless AI must contain one entity per row, like a customer, patient, piece of equipment, or financial transaction. That row must also contain information about what you will be trying to predict using similar data in the future, like whether that customer in the row of data used a promotion, whether that patient was readmitted to the hospital within thirty days of being released, whether that piece of equipment required maintenance, or whether that financial transaction was fraudulent. (In data science speak, Driverless AI requires "labeled" data.) Driverless AI runs through your data many, many times looking for interactions, insights, and business drivers of the phenomenon described by the provided dataset.

How do you use Driverless AI results to create commercial value?

Commercial value is generated by Driverless AI in a few ways.

- Driverless AI empowers data scientists or data analysts to work on projects faster and more efficiently by using automation and state-of-the-art computing power to accomplish tasks in just minutes or hours instead of the weeks or months that it can take humans.
- Like in many other industries, automation leads to standardization of business processes, enforces best practices, and eventually drives down the cost of delivering the final product – in this case a predictive model.
- Driverless AI makes deploying predictive models easy – typically a difficult step in the data science process. In large organizations, value from predictive modeling is typically realized when a predictive model is moved from a data analyst's or data scientist's development environment into a production deployment setting. In this setting, the model is running on live data and making quick and automatic decisions that make or save money. Driverless AI provides both Java- and Python-based technologies to make production deployment simpler.

Moreover, the system was designed with interpretability and transparency in mind. Every prediction made by a Driverless AI model can be explained to business users, so the system is viable even for regulated industries.

KEY FEATURES

Below are some of the key features available in Driverless AI.

4.1 Flexibility of Data and Deployment

Driverless AI works across a variety of data sources including Hadoop HDFS, Amazon S3, and more. Driverless AI can be deployed everywhere including all clouds (Microsoft Azure, AWS, Google Cloud) and on premises on any system, but it is ideally suited for systems with GPUs, including IBM Power 9 with GPUs built in.

4.2 NVIDIA GPU Acceleration

Driverless AI is optimized to take advantage of GPU acceleration to achieve up to 40X speedups for automatic machine learning. It includes multi-GPU algorithms for XGBoost, GLM, K-Means, and more. GPUs allow for thousands of iterations of model features and optimizations.

4.3 Automatic Data Visualization (Autovis)

For datasets, Driverless AI automatically selects data plots based on the most relevant data statistics, generates visualizations, and creates data plots that are most relevant from a statistical perspective based on the most relevant data statistics. These visualizations help users get a quick understanding of their data prior to starting the model building process. They are also useful for understanding the composition of very large datasets and for seeing trends or even possible issues, such as large numbers of missing values or significant outliers that could impact modeling results. See [Visualizing Datasets](#) for more information.

4.4 Automatic Feature Engineering

Feature engineering is the secret weapon that advanced data scientists use to extract the most accurate results from algorithms. H2O Driverless AI employs a library of algorithms and feature transformations to automatically engineer new, high value features for a given dataset. (See [Driverless AI Transformations](#) for more information.) Included in the interface is an easy-to-read variable importance chart that shows the significance of original and newly engineered features.

4.5 Automatic Model Documentation

To explain models to business users and regulators, data scientists and data engineers must document the data, algorithms, and processes used to create machine learning models. Driverless AI provides an Autoreport (Autodoc) for each experiment, relieving the user from the time-consuming task of documenting and summarizing their workflow used when building machine learning models. The Autoreport includes details about the data used, the validation schema selected, model and feature tuning, and the final model created. With this capability in Driverless AI, practitioners can focus more on drawing actionable insights from the models and save weeks or even months in development, validation, and deployment process.

Driverless AI also provides a number of `autodoc_` configuration options, giving users even more control over output of the Autoreport. (Refer to the [Sample Config.toml File](#) topic for information about these configuration options.)

[Click here](#) to download and view a sample experiment report in Word format.

4.6 Time Series Forecasting

Time series forecasting is one of the biggest challenges for data scientists. These models address key use cases, including demand forecasting, infrastructure monitoring, and predictive maintenance. Driverless AI delivers superior time series capabilities to optimize for almost any prediction time window. Driverless AI incorporates data from numerous predictors, handles structured character data and high-cardinality categorical variables, and handles gaps in time series data and other missing values. See [Time Series in Driverless AI](#) for more information.

4.7 NLP with TensorFlow

Text data can contain critical information to inform better predictions. Driverless AI automatically converts short text strings into features using powerful techniques like TFIDF. With TensorFlow, Driverless AI can also process larger text blocks and build models using all available data to solve business problems like sentiment analysis, document classification, and content tagging. See [NLP in Driverless AI](#) for more information.

4.8 Automatic Scoring Pipelines

For completed experiments, Driverless AI automatically generates both Python scoring pipelines and new ultra-low latency automatic scoring pipelines. The new automatic scoring pipeline is a unique technology that deploys all feature engineering and the winning machine learning model in a highly optimized, low-latency, production-ready Java code that can be deployed anywhere. See [Scoring Pipelines Overview](#) for more information.

4.9 Machine Learning Interpretability (MLI)

Driverless AI provides robust interpretability of machine learning models to explain modeling results in a human-readable format. In the MLI view, Driverless AI employs a host of different techniques and methodologies for interpreting and explaining the results of its models. A number of charts are generated automatically (depending on experiment type), including K-LIME, Shapley, Variable Importance, Decision Tree Surrogate, Partial Dependence, Individual Conditional Expectation, Sensitivity Analysis, NLP Tokens, NLP LOCO, and more. Additionally, you can download a CSV of LIME and Shapley reasons codes from this view. See [MLI Overview](#) for more information.

4.10 Automatic Reason Codes

In regulated industries, an explanation is often required for significant decisions relating to customers (for example, credit denial). Reason codes show the key positive and negative factors in a model's scoring decision in a simple language. Reasons codes are also useful in other industries, such as healthcare, because they can provide insights into model decisions that can drive additional testing or investigation.

4.11 Custom Recipe Support

Driverless AI allows you to import custom recipes for ML algorithms, feature engineering (transformers), scorers, and configuration. You can use your custom recipes in combination with or instead of all built-in recipes. This allows you to have greater influence over the Driverless AI Automatic ML pipeline and gives you control over the optimization choices that Driverless AI makes. See [Appendix A: Custom Recipes](#) for more information.

SUPPORTED ALGORITHMS

5.1 Constant Model

A Constant Model predicts the same constant value for any input data. The constant value is computed by optimizing the given scorer. For example, for MSE/RMSE, the constant is the (weighted) mean of the target column. For MAE, it is the (weighted) median. For other scorers like MAPE or custom scorers, the constant is found with an optimization process. For classification problems, the constant probabilities are the observed priors.

A constant model is meant as a baseline reference model. If it ends up being used in the final pipeline, a warning will be issued because that would indicate a problem in the dataset or target column (e.g., when trying to predict a random outcome).

5.2 Decision Tree

A Decision Tree is a single (binary) tree model that splits the training data population into sub-groups (leaf nodes) with similar outcomes. No row or column sampling is performed, and the tree depth and method of growth (depth-wise or loss-guided) is controlled by hyper-parameters.

5.3 FTRL

Follow the Regularized Leader (FTRL) is a DataTable implementation [1] of the FTRL-Proximal online learning algorithm proposed in [4]. This implementation uses a hashing trick and Hogwild approach [3] for parallelization. FTRL supports binomial and multinomial classification for categorical targets, as well as regression for continuous targets.

5.4 GLM

Generalized Linear Models (GLM) estimate regression models for outcomes following exponential distributions. GLMs are an extension of traditional linear models. They have gained popularity in statistical data analysis due to:

- the flexibility of the model structure unifying the typical regression methods (such as linear regression and logistic regression for binary classification)
- the recent availability of model-fitting software
- the ability to scale well with large datasets

Notes:

- Driverless AI uses the XGBoost GLM implementation.
- GLM in Driverless AI is subject to early stopping.

5.5 Isolation Forest

Isolation Forest is useful for identifying anomalies or outliers in data. Isolation Forest isolates observations by randomly selecting a feature and then randomly selecting a split value between the maximum and minimum values of that selected feature. This split depends on how long it takes to separate the points. Random partitioning produces noticeably shorter paths for anomalies. When a forest of random trees collectively produces shorter path lengths for particular samples, they are highly likely to be anomalies.

5.6 LightGBM

LightGBM is a gradient boosting framework developed by Microsoft that uses tree based learning algorithms. It was specifically designed for lower memory usage and faster training speed and higher efficiency. Similar to XGBoost, it is one of the best gradient boosting implementations available. It is also used for fitting Random Forest, DART (experimental), and Decision Tree models inside of Driverless AI.

Note: LightGBM with GPUs is not currently supported on Power.

5.7 PyTorch Models

PyTorch is an open source library used for deep learning tasks such as natural language processing and computer vision. Driverless AI's NLP BERT models are implemented using PyTorch, and more models will be implemented in the future.

5.8 RuleFit

The RuleFit [2] algorithm creates an optimal set of decision rules by first fitting a tree model, and then fitting a Lasso (L1-regularized) GLM model to create a linear model consisting of the most important tree leaves (rules).

Note: MOJOs are not currently available for RuleFit models.

5.9 TensorFlow

TensorFlow is an open source software library for performing high performance numerical computation. Driverless AI includes a TensorFlow NLP recipe based on CNN Deeplearning models.

Note: Only C++ MOJOs are currently available for TensorFlow models.

5.10 XGBoost

XGBoost is a supervised learning algorithm that implements a process called boosting to yield accurate models. Boosting refers to the ensemble learning technique of building many models sequentially, with each new model attempting to correct for the deficiencies in the previous model. In tree boosting, each new model that is added to the ensemble is a decision tree. XGBoost provides parallel tree boosting (also known as GBDT, GBM) that solves many data science problems in a fast and accurate way. For many problems, XGBoost is one of the best gradient boosting machine (GBM) frameworks today. Driverless AI supports XGBoost GBM and XGBoost DART (experimental) models.

5.11 Zero-Inflated Models

Zero-inflated models fit the data with excess zero counts in the target variable for example in insurance claim use case. In Driverless AI, this model trains a classifier that attempts to classify zero and non-zero values. It then trains a regression model that attempts to predict the non-zero values. The classifier predictions are multiplied by the regression predictions to determine the final output.

Note: Driverless AI supports both LightGBM and XGBoost versions of zero-inflated models.

5.12 References

- [1] DataTable for Python, <https://github.com/h2oai/datatable>
- [2] J. Friedman, B. Popescu. “Predictive Learning via Rule Ensembles”. 2005. <http://statweb.stanford.edu/~jhf/ftp/RuleFit.pdf>
- [3] Niu, Feng, et al. “Hogwild: A lock-free approach to parallelizing stochastic gradient descent.” Advances in neural information processing systems. 2011. <https://people.eecs.berkeley.edu/~brecht/papers/hogwildTR.pdf>
- [4] McMahan, H. Brendan, et al. “Ad click prediction: a view from the trenches.” Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining. ACM, 2013. <https://research.google.com/pubs/archive/41159.pdf>

DRIVERLESS AI WORKFLOW

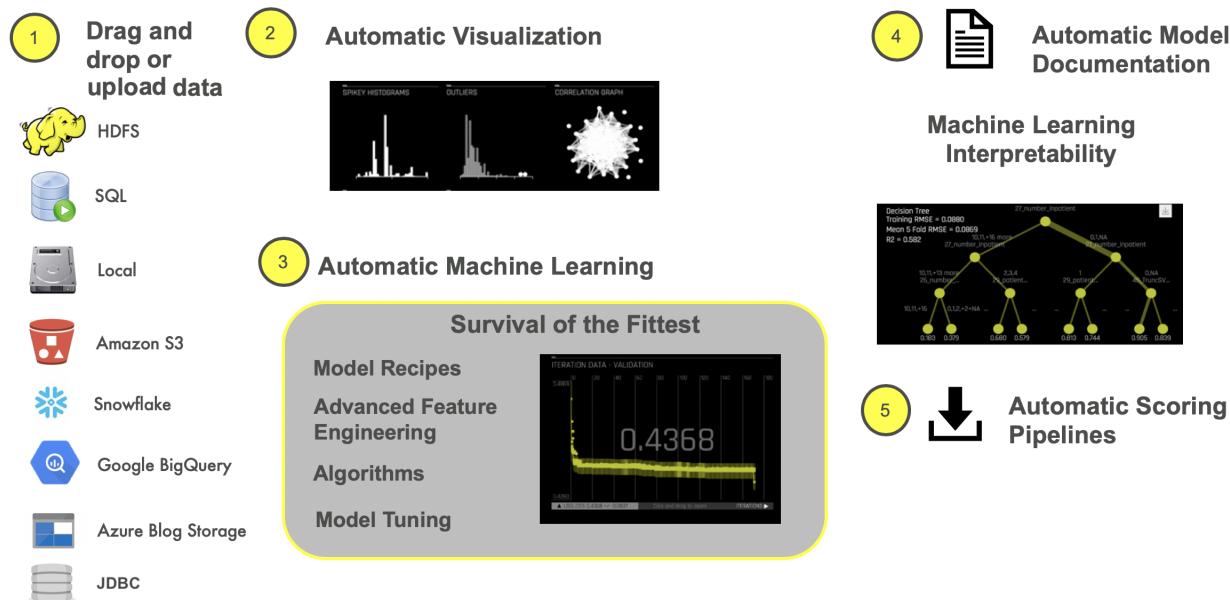
A typical Driverless AI workflow is to:

1. Load data
2. Visualize data
3. Run an experiment
4. Interpret the model
5. Deploy the scoring pipeline

In addition, you can diagnose a model, transform another dataset, score the model against another dataset, and manage your data in Projects.

The image below describes a typical workflow.

Typical Driverless AI Workflow



DRIVERLESS AI LICENSES

A valid license is required for running Driverless AI and for running the scoring pipelines.

7.1 About Licenses

Driverless AI is licensed per a single named user. Therefore, in order, to have different users run experiments simultaneously, they would each need a license. Driverless AI manages the GPU(s) that it is given and ensures that different experiments from different users can run safely simultaneously and don't interfere with each other. So when two licensed users log in with different credentials, neither of them will see the other's experiment. Similarly, if a licensed user logs in using a different set of credentials, that user will not see any previously run experiments.

7.2 Adding Licenses for the First Time

7.2.1 Specifying a License File for the Driverless AI Application

A license file to run Driverless AI can be added in one of three ways when starting Driverless AI.

- Specifying the license.sig file during launch in native installs
- Using the DRIVERLESS_AI_LICENSE_FILE and DRIVERLESS_AI_LICENSE_KEY environment variables when starting the Driverless AI Docker image
- Uploading your license in the Web UI

Specifying the license.sig File During Launch

By default, Driverless AI looks for a license key in `/opt/h2oai/dai/home/.driverlessai/license.sig`. If you are installing Driverless AI programmatically, you can copy a license key file to that location. If no license key is found, the application will prompt you to add one via the Web UI.

Specifying Environment Variables

You can use the DRIVERLESS_AI_LICENSE_FILE or DRIVERLESS_AI_LICENSE_KEY environment variable when starting the Driverless AI Docker image. For example:

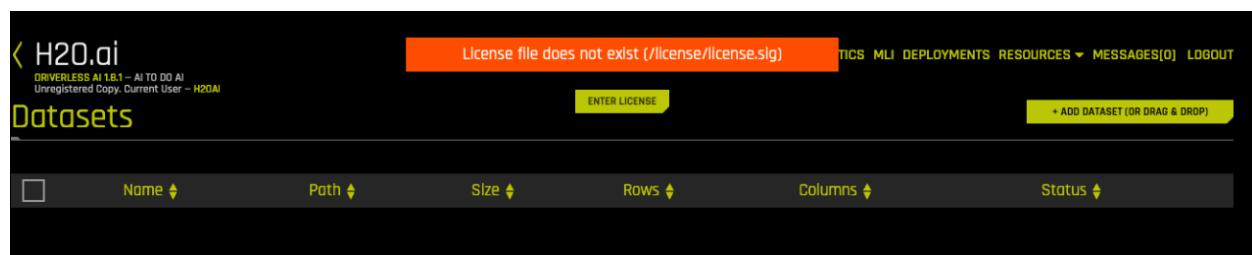
```
nvidia-docker run \
--pid=host \
--init \
--rm \
--shm-size=256m \
-u `id -u`:`id -g` \
-p 12345:12345 \
-e DRIVERLESS_AI_LICENSE_FILE="/license/license.sig" \
-v `pwd`/config:/config \
-v `pwd`/data:/data \
-v `pwd`/log:/log \
-v `pwd`/license:/license \
-v `pwd`/tmp:/tmp \
harbor.h2o.ai/h2oai/dai-centos7-x86_64:1.9.0-cuda10.0.23
```

or

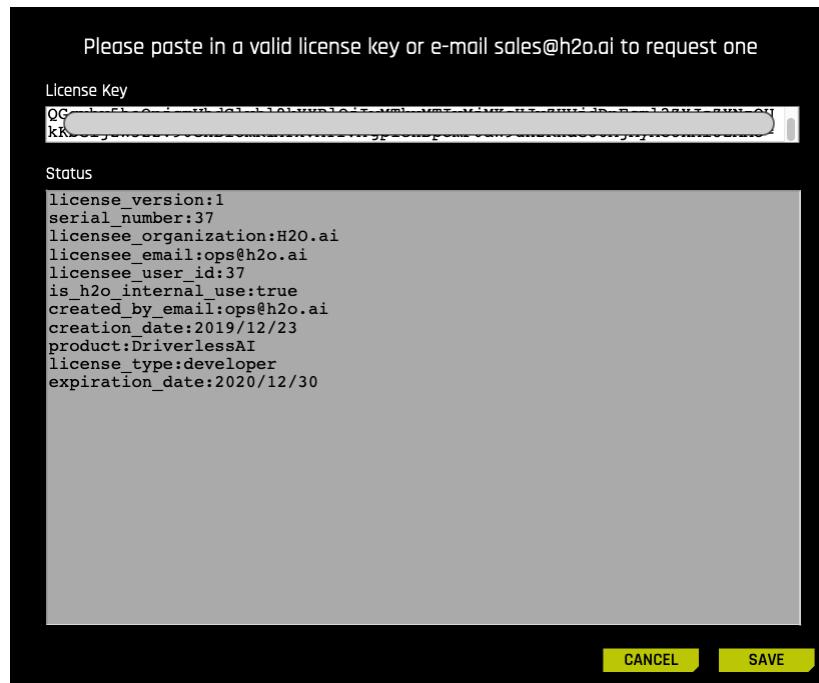
```
nvidia-docker run \
--pid=host \
--init \
--rm \
--shm-size=256m \
-u `id -u`:`id -g` \
-p 12345:12345 \
-e DRIVERLESS_AI_LICENSE_KEY="Y0uR11cens3KeyH3re" \
-v `pwd`/config:/config \
-v `pwd`/data:/data \
-v `pwd`/log:/log \
-v `pwd`/license:/license \
-v `pwd`/tmp:/tmp \
harbor.h2o.ai/h2oai/dai-centos7-x86_64:1.9.0-cuda10.0.23
```

Uploading Your License in the Web UI

If Driverless AI does not locate a license.sig file during launch, then the UI will prompt you to enter your license key after you log in the first time.



Click the **Enter License** button, and then paste the entire license into the provided **License Key** entry field. Click **Save** when you are done. Upon successful completion, you will be able to begin using Driverless AI.



7.2.2 Specifying a License for Scoring Pipelines

Driverless AI requires a license to be specified in order to run the Scoring Pipelines.

Python Scoring Pipeline

The license can be specified via an environment variable in Python:

```
# Set DRIVERLESS_AI_LICENSE_FILE, the path to the Driverless AI license file
%env DRIVERLESS_AI_LICENSE_FILE="/home/ubuntu/license/license.sig"

# Set DRIVERLESS_AI_LICENSE_KEY, the Driverless AI license key (Base64 encoded string)
%env DRIVERLESS_AI_LICENSE_KEY="oLqLZXMI0y..."
```

You can also export the license file when running the scoring pipeline:

```
export DRIVERLESS_AI_LICENSE_FILE="/path/to/license.sig"
bash run_example.sh
```

MOJO Scoring Pipeline

Driverless AI requires a license to be specified in order to run the MOJO Scoring Pipeline. The license can be specified in one of the following ways:

- Via an environment variable:
 - DRIVERLESS_AI_LICENSE_FILE: Path to the Driverless AI license file, or
 - DRIVERLESS_AI_LICENSE_KEY: The Driverless AI license key (Base64 encoded string)
- Via a system property of JVM (-D option):

- `ai.h2o.mojos.runtime.license.file`: Path to the Driverless AI license file, or
 - `ai.h2o.mojos.runtime.license.key`: The Driverless AI license key (Base64 encoded string)
- Via an application classpath:
 - The license is loaded from a resource called `/license.sig`.
 - The default resource name can be changed via the JVM system property `ai.h2o.mojos.runtime.license.filename`.

For example:

```
java -Dai.h2o.mojos.runtime.license.file=/etc/dai/license.sig -cp mojo2-runtime.jar  
ai.h2o.mojos.ExecuteMojo pipeline.mojo example.csv
```

7.2.3 Driverless AI Licenses in Production via AWS Lambdas

Driverless AI deployment pipelines to AWS Lambdas automatically set the license key as an environment variable based on the license key that was used in Driverless AI.

7.3 Updating Licenses

If your current Driverless AI license has expired, you will be required to update it in order to continue running Driverless AI, in order to run the scoring pipeline, in order to access deployed pipelines to AWS Lambdas, etc.

7.3.1 Updating the License for Driverless AI

Similar to adding a license for the first time, you can update your license for running Driverless AI either by replacing your current `license.sig` file or via the Web UI.

Updating the `license.sig` File

Update the license key in your `/opt/h2oai/dai/home/.driverlessai/license.sig` file by replacing the existing license with your new one.

Updating the License in the Web UI

If your license is expired, the Web UI will prompt you to enter a new one. The steps are the same as adding a license for the first time via the Driverless AI Web UI.

7.3.2 Updating the License for Scoring Pipelines

For the Python Scoring Pipeline, simply include the updated license file when setting the environment variable in Python. Refer to the above [Python Scoring Pipeline](#) section for adding licenses.

For the MOJO Scoring Pipeline, the updated license file can be specified using an environment variable, using a system property of JVM, or via an application classpath. This is the same as adding a license for the first time. Refer to the above [MOJO Scoring Pipeline](#) section for adding licenses.

7.3.3 Updating Driverless AI Licenses on AWS Lambda

Users can manually update each of their Driverless AI licenses deployed in production on AWS Lambda. For users with many MOJOs in production, though, H2O provides a script that will update Driverless AI licenses for all of your MOJOs currently deployed on AWS Lambda.

Manual Update

The Driverless AI deployment pipeline to AWS Lambdas explicitly sets the license key as an environment variable. Replace the expired license key with your updated one.

Environment variables

You can define environment variables as key-value pairs that are accessible from your function code. These are useful to store configuration set

DEPLOYMENT_S3_BUCKET_NAME

DRIVERLESS_AI_LICENSE_KEY

MOJO_FINGERPRINT

Automatic Update

H2O provides a script that can be used to update Driverless AI licenses for all of your MOJOs deployed on a specific AWS Lambda region. This script can be run for any machine.

Requirements

- New Driverless AI license
- The following Python packages are required for this script:
 - boto3
 - argparse
 - os

Update Steps

Perform the following steps to update your Driverless AI license for MOJOs on AWS Lambda.

1. Copy the following code and save it as **update_lambda.py**.

```
import boto3
import argparse
import os

def load_license(license_path):
    with open(license_path, 'r') as license_file:
        license_key = license_file.read()
```

(continues on next page)

(continued from previous page)

```

    return license_key

def update_function(client, license_key, function_names):

    for name in function_names:
        config = client.get_function_configuration(FunctionName=name)
        config['Environment']['Variables']['DRIVERLESS_AI_LICENSE_KEY'] = \
            license_key

        response = client.update_function_configuration(
            FunctionName= name,
            Environment= config['Environment']
        )
        print("{} License Key updated successfully! ".format(name))

def list_functions(client):

    response = client.list_functions()

    dai_functions = []
    for funct in response['Functions']:
        if 'h2oai' in funct['FunctionName']:
            dai_functions.append(funct['FunctionName'])

    return dai_functions

def main():

    parser = argparse.ArgumentParser()
    parser.add_argument("--list", action='store_true', required=False, help=
        "List all DAI lambda functions on the selected region. ")
    parser.add_argument("--update_all", action='store_true', required=False, \
        help="Update all DAI Lambda functions on the selected region. ")
    parser.add_argument("--region", type=str, required=True, help="Region \
        where the lambda function is deployed.")
    parser.add_argument("--name", type=str, required=False, help="The name \
        of the Lambda function. ")
    parser.add_argument("--license_path", type=str, required=False, help=
        "Location of the license.sig file")
    parser.add_argument("--aws_access_key_id", type=str, required=False, \
        help="AWS Credentials")
    parser.add_argument("--aws_secret_access_key", type=str, required=False, \
        help="AWS Credentials")
    args = parser.parse_args()

    if None not in (args.aws_access_key_id, args.aws_secret_access_key):
        os.environ['aws_access_key_id'] = args.aws_access_key_id
        os.environ['aws_secret_access_key'] = args.aws_secret_access_key

    client = boto3.client('lambda', region_name=args.region)

    if args.update_all or args.list:
        dai_functions = list_functions(client)
        print("H2O Driverless AI Lambda Functions in {}: {}".format(args.\
            region, dai_functions))

    _functions = []

```

(continues on next page)

(continued from previous page)

```

if args.update_all:
    _functions = dai_functions
elif args.name is not None:
    _functions.append(args.name)
else:
    print("Please set a name of a function to update")

if args.license_path is not None:
    license_key = load_license(args.license_path)
elif "DRIVERLESS_AI_LICENSE_FILE" in os.environ:
    license_key = load_license(os.environ['DRIVERLESS_AI_LICENSE_FILE'])
elif "DRIVERLESS_AI_LICENSE_KEY" in os.environ:
    license_key = os.environ['DRIVERLESS_AI_LICENSE_KEY']
else:
    print("No License Found")
    return 0

update_function(client, license_key, _functions)

if __name__ == '__main__':
    main()

```

2. To run the script, you will need to provide the following:

- The region where the MOJOs have been deployed. Note that this script must be run for each region that includes deployed MOJOs.
- The Lambda function name
- The Driverless AI license path
- Your AWS secret key ID
- Your AWS secret access key

```

# run the upgrade script
python update_lambda.py --update_all --region [deployed_region] --name_
↪[Lambda_function_name] --license_path [path_to_license_file] --aws_access_
↪key_id [AWS_ACCESS_KEY_ID] --aws_secret_access_key [AWS_SECRET_ACCESS_KEY]

# optionally view the help using either python or python3
python update_lambda.py --help

```


SUPPORTED ENVIRONMENTS

The following tables list the environments that support Driverless AI.

8.1 Linux

Package Type	OS	GPU	CPU
RPM	RHEL 7/CentOS 7/SLES 12	CUDA 10.0 and above/CPU only	x86_64
DEB	Ubuntu 16.04/Ubuntu 18.04	CUDA 10.0 and above/CPU only	x86_64
TAR SH	Most Linux	CUDA 10.0 and above/CPU only	x86_64
Docker	Docker CE	CUDA 10.0 and above/CPU only	x86_64

8.2 Windows 10 Pro, Enterprise, or Education

Caution: Windows computers (laptops in particular) should only be used with small datasets for the purpose of exploring the software. For serious use, server hardware is required. Consider spinning up a more powerful instance in the cloud instead of using a laptop. Avoid laptops with less than 16 GB of RAM. GPUs are not supported on Windows.

Package Type	OS	GPU Support?	CPU	Min Memory
DEB	Ubuntu 18.04 for WSL (not fully tested)	No	x86_64	16 GB
Docker	Docker Desktop for Win 2.2.0.3 (42716)	No	x86_64	16 GB

8.3 Mac OS X

Caution: Mac OS computers (laptops in particular) should only be used with small datasets for the purpose of exploring the software. For serious use, server hardware is required. Consider spinning up a more powerful instance in the cloud instead of using a laptop. Avoid laptops with less than 16 GB of RAM. GPUs are not supported on Mac OS.

Package Type	OS	GPU Support?	CPU	Min Memory
Docker	Mac OS X	No	x86_64	16 GB

8.4 IBM Power

Package Type	OS	GPU	CPU
RPM	RHEL 7	CUDA 10.0 and above/CPU only	ppc64le
DEB	Ubuntu 16.04 (not fully tested)	CUDA 10.0 and above/CPU only	ppc64le
TAR SH	Ubuntu 16.0.4 (not fully tested)/RHEL 7	CUDA 10.0 and above/CPU only	ppc64le
Docker	Docker CE	CUDA 10.0 and above/CPU only	ppc64le

BEFORE YOU BEGIN INSTALLING OR UPGRADING

Please review the following information before you begin installing Driverless AI. Be sure to also review the [Sizing Requirements](#) in the next section before beginning the installation.

9.1 Supported Browsers

Driverless AI is tested most extensively on Chrome and Firefox. For the best user experience, we recommend using the latest version of Chrome. You may encounter issues if you use other browsers or earlier versions of Chrome and/or Firefox.

9.2 To sudo or Not to sudo

Many of the installation steps show `sudo` prepending different commands. Note that `sudo` may not always be required, but the steps that are documented here are the steps that we followed in house.

9.3 Note about Docker Configuration (`ulimit`)

When running Driverless AI with Docker, it is recommended to configure `ulimit` options by using the `--ulimit` argument to `docker run`. The following is an example of how to configure these options:

```
--ulimit nproc=65535:65535 \
--ulimit nofile=4096:8192 \
```

Refer to <https://docs.docker.com/engine/reference/commandline/run/#set-ulimits-in-container---ulimit> for more information on these options.

9.4 Note about nvidia-docker 1.0

If you have nvidia-docker 1.0 installed, you need to remove it and all existing GPU containers. Refer to <https://github.com/NVIDIA/nvidia-docker/blob/master/README.md> for more information.

9.5 Deprecation of nvidia-smi

The nvidia-smi command has been deprecated by NVIDIA. Refer to <https://github.com/nvidia/nvidia-docker#upgrading-with-nvidia-docker2-deprecated> for more information. The installation steps have been updated for enabling persistence mode for GPUs.

9.6 New nvidia-container-runtime-hook Requirement for PowerPC Users

PowerPC users are now required to install the nvidia-container-runtime-hook when running in Docker. Refer to <https://github.com/nvidia/nvidia-docker#rhel-docker> for more information. The IBM Docker installation steps have been updated to reflect this information.

9.7 Note About CUDA Versions

Your host environment must have CUDA 10.0 or later with NVIDIA drivers >= 440.82 installed (GPU only). Driverless AI ships with its own CUDA libraries, but the driver must exist in the host environment. Go to <https://www.nvidia.com/Download/index.aspx> to get the latest NVIDIA Tesla V/P/K series driver.

9.8 Note About Authentication

The default authentication setting in Driverless AI is “unvalidated.” In this case, Driverless AI will accept any login and password combination, it will not validate whether the password is correct for the specified login ID, and it will connect to the system as the user specified in the login ID. This is true for all instances, including Cloud, Docker, and native instances.

We recommend that you configure authentication. Driverless AI provides a number of authentication options, including LDAP, PAM, Local, and None. Refer to *Configuring Authentication* for information on how to enable a different authentication method.

Note: Driverless AI is also integrated with IBM Spectrum Conductor and supports authentication from Conductor. Contact sales@h2o.ai for more information about using IBM Spectrum Conductor authentication.

9.9 Note About Shared File Systems

If your environment uses a shared file system, then you must set the following configuration option:

```
datatable_strategy='write'
```

The above can be specified in the `config.toml` file (for native installs) or specified as an environment variable (Docker image installs).

This configuration is required because, in some cases, Driverless AI can fail to read files during an experiment. The `write` option will allow Driverless AI to properly read and write data from shared file systems to disk.

9.10 Note About the Master Database File

If you are running two versions of Driverless AI, keep in mind that newer versions of the **master.db** file will not work with older versions of Driverless AI.

9.11 Backup Strategy

We recommend that you periodically stop Driverless AI and back up your Driverless AI tmp directory, even if you are not upgrading.

9.12 Upgrade Strategy

Keep in mind the following when upgrading Driverless AI:

- This release deprecates experiments and MLIs models from 1.7.0 and earlier.
- Experiments, MLIs, and MOJOS reside in the Driverless AI tmp directory and are not automatically upgraded when Driverless AI is upgraded. We recommend you take the following steps before upgrading.
 - Build MLIs models before upgrading.
 - Build MOJO pipelines before upgrading.
 - Stop Driverless AI and then make a backup of your Driverless AI tmp directory before upgrading.

If you did not build MLIs on a model before upgrading Driverless AI, then you will not be able to view MLIs on that model after upgrading. Before upgrading, be sure to run MLIs jobs on models that you want to continue to interpret in future releases. If that MLIs job appears in the list of Interpreted Models in your current version, then it will be retained after upgrading.

If you did not build a MOJO pipeline on a model before upgrading Driverless AI, then you will not be able to build a MOJO pipeline on that model after upgrading. Before upgrading, be sure to build MOJO pipelines on all desired models and then back up your Driverless AI tmp directory.

The upgrade process inherits the service user and group from /etc/dai/User.conf and /etc/dai/Group.conf. You do not need to manually specify the DAI_USER or DAI_GROUP environment variables during an upgrade.

SIZING REQUIREMENTS

10.1 Sizing Requirements for Native Installs

Driverless AI requires a minimum of 5 GB of system memory in order to start experiments and a minimum of 5 GB of disk space in order to run a small experiment. Note that these limits can be changed in the config.toml file. We recommend that you have lots of system CPU memory (64 GB or more) and 1 TB of free disk space available.

10.2 Sizing Requirements for Docker Installs

For Docker installs, we recommend 1 TB of free disk space. Driverless AI uses approximately 38 GB. In addition, the unpacking/temp files require space on the same Linux mount `/var` during installation. Once DAI runs, the mounts from the Docker container can point to other file system mount points.

10.3 GPU Sizing Requirements

If you are running Driverless AI with GPUs, be sure that your GPU has compute capability ≥ 3.5 and at least 4GB of RAM. If these requirements are not met, then Driverless AI will switch to CPU-only mode.

10.4 Sizing Requirements for Storing Experiments

We recommend that your tmp directory has at least 500 GB to 1 TB of space. The tmp directory holds all experiments and all datasets. We also recommend that you use SSDs (preferably NVMe).

10.5 Virtual Memory Settings in Linux

If you are running Driverless AI on a Linux machine, we recommend setting the overcommit memory to 0. The setting can be changed by the following command:

```
sudo echo 0 > /proc/sys/vm/overcommit_memory
```

This is the default value, and it indicates that the Linux kernel is free to overcommit memory. If this value is set to 2, then the Linux kernel will not overcommit memory. In this case, the memory requirements of Driverless AI may surpass the memory allocation limit, which would prevent the experiment from completing.

INSTALLING AND UPGRADING DRIVERLESS AI

For the best (and intended-as-designed) experience, install Driverless AI on modern data center hardware with GPUs and CUDA support. Use Pascal or Volta GPUs with maximum GPU memory for best results. (Note that the older K80 and M60 GPUs available in EC2 are supported and very convenient, but not as fast.)

Driverless AI supports local, LDAP, and PAM authentication. Authentication can be configured by setting environment variables or via a config.toml file. Refer to the [Configuring Authentication](#) section for more information. Note that the default authentication method is “unvalidated.”

Driverless AI also supports HDFS, S3, Google Cloud Storage, Google Big Query, KDB, Minio, and Snowflake access. Support for these data sources can be configured by setting environment variables for the data connectors or via a config.toml file. Refer to the [Enabling Data Connectors](#) section for more information.

11.1 Linux X86_64 Installs

This section provides installation steps for Linux 86_64 environments. This includes information for Docker image installs, RPMs, Deb, and Tar installs as well as Cloud installations.

11.1.1 Linux Docker Images

To simplify local installation, Driverless AI is provided as a Docker image for the following system combinations:

Host OS	Docker Version	Host Architecture	Min Mem
Ubuntu 16.04 or later	Docker CE	x86_64	64 GB
RHEL or CentOS 7.4 or later	Docker CE	x86_64	64 GB
NVIDIA DGX Registry		x86_64	

Note: CUDA 10.0 or later with NVIDIA drivers >= 440.82 is required (GPU only).

For the best performance, including GPU support, use nvidia-docker. For a lower-performance experience without GPUs, use regular docker (with the same docker image).

These installation steps assume that you have a license key for Driverless AI. For information on how to obtain a license key for Driverless AI, visit <https://www.h2o.ai/driverless-ai/>. Once obtained, you will be prompted to paste the license key into the Driverless AI UI when you first log in, or you can save it as a .sig file and place it in the license folder that you will create during the installation process.

Install on Ubuntu

This section describes how to install the Driverless AI Docker image on Ubuntu. The installation steps vary depending on whether your system has GPUs or if it is CPU only.

Environment

Operating System	GPUs?	Min Mem
Ubuntu with GPUs	Yes	64 GB
Ubuntu with CPUs	No	64 GB

Install on Ubuntu with GPUs

Note: Driverless AI is supported on Ubuntu 16.04 or later.

Open a Terminal and ssh to the machine that will run Driverless AI. Once you are logged in, perform the following steps.

1. Retrieve the Driverless AI Docker image from <https://www.h2o.ai/download/>. (Note that the contents of this Docker image include a CentOS kernel and CentOS packages.)
2. Install and run Docker on Ubuntu (if not already installed):

```
# Install and run Docker on Ubuntu
curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo apt-key add -
sudo apt-key fingerprint 0EBFCD88 sudo add-apt-repository \
    "deb [arch=amd64] https://download.docker.com/linux/ubuntu $(lsb_release -cs) stable"
sudo apt-get update
sudo apt-get install docker-ce
sudo systemctl start docker
```

3. Install nvidia-docker2 (if not already installed). More information is available at <https://github.com/NVIDIA/nvidia-docker/blob/master/README.md>.

```
curl -s -L https://nvidia.github.io/nvidia-docker/gpgkey | \
    sudo apt-key add -
distribution=$( . /etc/os-release;echo $ID$VERSION_ID )
curl -s -L https://nvidia.github.io/nvidia-docker/$distribution/nvidia-
docker.list | \
    sudo tee /etc/apt/sources.list.d/nvidia-docker.list
sudo apt-get update

# Install nvidia-docker2 and reload the Docker daemon configuration
sudo apt-get install -y nvidia-docker2
```

4. Verify that the NVIDIA driver is up and running. If the driver is not up and running, log on to <http://www.nvidia.com/Download/index.aspx?lang=en-us> to get the latest NVIDIA Tesla V/P/K series driver:

```
nvidia-smi
```

5. Set up a directory for the version of Driverless AI on the host machine:

```
# Set up directory with the version name
mkdir dai-1.9.0
```

6. Change directories to the new folder, then load the Driverless AI Docker image inside the new directory:

```
# cd into the new directory
cd dai-1.9.0

# Load the Driverless AI docker image
docker load < dai-docker-centos7-x86_64-1.9.0-10.0.tar.gz
```

7. Enable persistence of the GPU. Note that this needs to be run once every reboot. Refer to the following for more information: <http://docs.nvidia.com/deploy/driver-persistence/index.html>.

```
sudo nvidia-persistenced --persistence-mode
```

8. Set up the data, log, and license directories on the host machine:

```
# Set up the data, log, license, and tmp directories on the host machine
# (within the new directory)
mkdir data
mkdir log
mkdir license
mkdir tmp
```

9. At this point, you can copy data into the data directory on the host machine. The data will be visible inside the Docker container.

10. Run docker images to find the image tag.

11. Start the Driverless AI Docker image and replace TAG below with the image tag. Depending on your install version, use the docker run --runtime=nvidia (>= Docker 19.03) or nvidia-docker (< Docker 19.03) command.

Note: Use docker version to check which version of Docker you are using.

>= Docker 19.03

< Docker 19.03

```
# Start the Driverless AI Docker image
docker run --runtime=nvidia \
    --pid=host \
    --init \
    --rm \
    --shm-size=256m \
    -u `id -u`:`id -g` \
    -p 12345:12345 \
    -v `pwd`/data:/data \
    -v `pwd`/log:/log \
    -v `pwd`/license:/license \
    -v `pwd`/tmp:/tmp \
    h2oai/dai-centos7-x86_64:TAG
```

```
# Start the Driverless AI Docker image
nvidia-docker run \
    --pid=host \
    --init \
    --rm \
    --shm-size=256m \
    -u `id -u`:`id -g` \
    -p 12345:12345 \
```

(continues on next page)

(continued from previous page)

```
-v `pwd`/data:/data \
-v `pwd`/log:/log \
-v `pwd`/license:/license \
-v `pwd`/tmp:/tmp \
h2oai/dai-centos7-x86_64:TAG
```

Driverless AI will begin running:

```
-----
Welcome to H2O.ai's Driverless AI
-----
- Put data in the volume mounted at /data
- Logs are written to the volume mounted at /log/20180606-044258
- Connect to Driverless AI on port 12345 inside the container
- Connect to Jupyter notebook on port 8888 inside the container
```

12. Connect to Driverless AI with your browser:

```
http://Your-Driverless-AI-Host-Machine:12345
```

Install on Ubuntu with CPUs

Note: Driverless AI is supported on Ubuntu 16.04 or later.

This section describes how to install and start the Driverless AI Docker image on Ubuntu. Note that this uses `docker` and not `nvidia-docker`. GPU support will not be available.

Watch the installation video [here](#). Note that some of the images in this video may change between releases, but the installation steps remain the same.

Open a Terminal and ssh to the machine that will run Driverless AI. Once you are logged in, perform the following steps.

1. Retrieve the Driverless AI Docker image from <https://www.h2o.ai/download/>.
2. Install and run Docker on Ubuntu (if not already installed):

```
# Install and run Docker on Ubuntu
curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo apt-key add -
sudo apt-key fingerprint 0EBFCD88 sudo add-apt-repository \
"deb [arch=amd64] https://download.docker.com/linux/ubuntu \$ (lsb_release -c\$) stable"
sudo apt-get update
sudo apt-get install docker-ce
sudo systemctl start docker
```

3. Set up a directory for the version of Driverless AI on the host machine:

```
# Set up directory with the version name
mkdir dai-1.9.0
```

4. Change directories to the new folder, then load the Driverless AI Docker image inside the new directory:

```
# cd into the new directory
cd dai-1.9.0
```

(continues on next page)

(continued from previous page)

```
# Load the Driverless AI docker image
docker load < dai-docker-centos7-x86_64-1.9.0-10.0.tar.gz
```

5. Set up the data, log, license, and tmp directories on the host machine (within the new directory):

```
# Set up the data, log, license, and tmp directories
mkdir data
mkdir log
mkdir license
mkdir tmp
```

6. At this point, you can copy data into the data directory on the host machine. The data will be visible inside the Docker container.
7. Run docker images to find the new image tag.
8. Start the Driverless AI Docker image. Note that GPU support will not be available.

```
# Start the Driverless AI Docker image
docker run \
    --pid=host \
    --init \
    --rm \
    --shm-size=256m \
    -u `id -u`:`id -g` \
    -p 12345:12345 \
    -v `pwd`/data:/data \
    -v `pwd`/log:/log \
    -v `pwd`/license:/license \
    -v `pwd`/tmp:/tmp \
    -v /etc/passwd:/etc/passwd:ro \
    -v /etc/group:/etc/group:ro \
h2oai/dai-centos7-x86_64:1.9.0-cuda10.0.23
```

Driverless AI will begin running:

```
-----
Welcome to H2O.ai's Driverless AI
-----
- Put data in the volume mounted at /data
- Logs are written to the volume mounted at /log/20180606-044258
- Connect to Driverless AI on port 12345 inside the container
- Connect to Jupyter notebook on port 8888 inside the container
```

9. Connect to Driverless AI with your browser:

```
http://Your-Driverless-AI-Host-Machine:12345
```

Stopping the Docker Image

To stop the Driverless AI Docker image, type **Ctrl + C** in the Terminal (Mac OS X) or PowerShell (Windows 10) window that is running the Driverless AI Docker image.

Upgrading the Docker Image

This section provides instructions for upgrading Driverless AI versions that were installed in a Docker container. These steps ensure that existing experiments are saved.

WARNING: Experiments, MLIs, and MOJOS reside in the Driverless AI tmp directory and are not automatically upgraded when Driverless AI is upgraded.

- Build MLIs models before upgrading.
- Build MOJO pipelines before upgrading.
- Stop Driverless AI and make a backup of your Driverless AI tmp directory before upgrading.

If you did not build MLIs on a model before upgrading Driverless AI, then you will not be able to view MLIs on that model after upgrading. Before upgrading, be sure to run MLIs jobs on models that you want to continue to interpret in future releases. If that MLIs job appears in the list of Interpreted Models in your current version, then it will be retained after upgrading.

If you did not build a MOJO pipeline on a model before upgrading Driverless AI, then you will not be able to build a MOJO pipeline on that model after upgrading. Before upgrading, be sure to build MOJO pipelines on all desired models and then back up your Driverless AI tmp directory.

Note: Stop Driverless AI if it is still running.

Requirements

As of 1.7.0, CUDA 9 is no longer supported. Your host environment must have CUDA 10.0 or later with NVIDIA drivers >= 440.82 installed (GPU only). Driverless AI ships with its own CUDA libraries, but the driver must exist in the host environment. Go to <https://www.nvidia.com/Download/index.aspx> to get the latest NVIDIA Tesla V/P/K series driver.

Upgrade Steps

1. SSH into the IP address of the machine that is running Driverless AI.
2. Set up a directory for the version of Driverless AI on the host machine:

```
# Set up directory with the version name
mkdir dai-1.9.0

# cd into the new directory
cd dai-1.9.0
```

3. Retrieve the Driverless AI package from <https://www.h2o.ai/download/> and add it to the new directory.
4. Load the Driverless AI Docker image inside the new directory:

```
# Load the Driverless AI docker image
docker load < dai-docker-centos7-x86_64-1.9.0-10.0.tar.gz
```

- Copy the data, log, license, and tmp directories from the previous Driverless AI directory to the new Driverless AI directory:

```
# Copy the data, log, license, and tmp directories on the host machine
cp -a dai_rel_1.4.2/data dai-1.9.0/data
cp -a dai_rel_1.4.2/log dai-1.9.0/log
cp -a dai_rel_1.4.2/license dai-1.9.0/license
cp -a dai_rel_1.4.2/tmp dai-1.9.0/tmp
```

At this point, your experiments from the previous versions will be visible inside the Docker container.

- Use docker images to find the new image tag.
- Start the Driverless AI Docker image.
- Connect to Driverless AI with your browser at <http://Your-Driverless-AI-Host-Machine:12345>.

Install on RHEL

This section describes how to install the Driverless AI Docker image on RHEL. The installation steps vary depending on whether your system has GPUs or if it is CPU only.

Environment

Operating System	GPUs?	Min Mem
RHEL with GPUs	Yes	64 GB
RHEL with CPUs	No	64 GB

Install on RHEL with GPUs

Note: Refer to the following links for more information about using RHEL with GPUs. These links describe how to disable automatic updates and specific package updates. This is necessary in order to prevent a mismatch between the NVIDIA driver and the kernel, which can lead to the GPUs failures.

- <https://access.redhat.com/solutions/2372971>
- <https://www.rootusers.com/how-to-disable-specific-package-updates-in-rhel-centos/>

Watch the installation video [here](#). Note that some of the images in this video may change between releases, but the installation steps remain the same.

Note: As of this writing, Driverless AI has only been tested on RHEL version 7.4.

Open a Terminal and ssh to the machine that will run Driverless AI. Once you are logged in, perform the following steps.

- Retrieve the Driverless AI Docker image from <https://www.h2o.ai/download/>.
- Install and start Docker EE on RHEL (if not already installed). Follow the instructions on <https://docs.docker.com/engine/installation/linux/docker-ee/rhel/>.

Alternatively, you can run on Docker CE.

```
sudo yum install -y yum-utils
sudo yum-config-manager --add-repo https://download.docker.com/linux/centos/
  docker-ce.repo
sudo yum makecache fast
sudo yum -y install docker-ce
sudo systemctl start docker
```

3. Install nvidia-docker2 (if not already installed). More information is available at <https://github.com/NVIDIA/nvidia-docker/blob/master/README.md>.

```
curl -s -L https://nvidia.github.io/nvidia-docker/gpgkey | \
  sudo apt-key add -
distribution=$( . /etc/os-release;echo $ID$VERSION_ID )
curl -s -L https://nvidia.github.io/nvidia-docker/$distribution/nvidia-
  docker.list | \
  sudo tee /etc/apt/sources.list.d/nvidia-docker.list
sudo apt-get update

# Install nvidia-docker2 and reload the Docker daemon configuration
sudo apt-get install -y nvidia-docker2
```

Note: If you would like the nvidia-docker service to automatically start when the server is rebooted then run the following command. If you do not run this command, you will have to remember to start the nvidia-docker service manually; otherwise the GPUs will not appear as available.

```
sudo systemctl enable nvidia-docker
```

Alternatively, if you have installed Docker CE above you can install nvidia-docker with:

```
curl -s -L https://nvidia.github.io/nvidia-docker/centos7/x86_64/nvidia-
  docker.repo | \
  sudo tee /etc/yum.repos.d/nvidia-docker.repo
sudo yum install nvidia-docker2
```

4. Verify that the NVIDIA driver is up and running. If the driver is not up and running, log on to <http://www.nvidia.com/Download/index.aspx?lang=en-us> to get the latest NVIDIA Tesla V/P/K series driver.

```
nvidia-docker run --rm nvidia/cuda nvidia-smi
```

5. Set up a directory for the version of Driverless AI on the host machine:

```
# Set up directory with the version name
mkdir dai-1.9.0
```

6. Change directories to the new folder, then load the Driverless AI Docker image inside the new directory:

```
# cd into the new directory
cd dai-1.9.0

# Load the Driverless AI docker image
docker load < dai-docker-centos7-x86_64-1.9.0-10.0.tar.gz
```

7. Enable persistence of the GPU. Note that this needs to be run once every reboot. Refer to the following for more information: <http://docs.nvidia.com/deploy/Driver-Persistence/index.html>.

```
sudo nvidia-persistenced --persistence-mode
```

8. Set up the data, log, and license directories on the host machine (within the new directory):

```
# Set up the data, log, license, and tmp directories on the host machine
mkdir data
mkdir log
mkdir license
mkdir tmp
```

9. At this point, you can copy data into the data directory on the host machine. The data will be visible inside the Docker container.

10. Run docker images to find the image tag.

11. Start the Driverless AI Docker image and replace TAG below with the image tag. Depending on your install version, use the docker run --runtime=nvidia (>= Docker 19.03) or nvidia-docker (< Docker 19.03) command.

Note: Use docker version to check which version of Docker you are using.

>= Docker 19.03

< Docker 19.03

```
# Start the Driverless AI Docker image
docker run --runtime=nvidia \
--pid=host \
--init \
--rm \
--shm-size=256m \
-u `id -u`:`id -g` \
-p 12345:12345 \
-v `pwd`/data:/data \
-v `pwd`/log:/log \
-v `pwd`/license:/license \
-v `pwd`/tmp:/tmp \
h2oai/dai-centos7-x86_64:TAG
```

```
# Start the Driverless AI Docker image
nvidia-docker run \
--pid=host \
--init \
--rm \
--shm-size=256m \
-u `id -u`:`id -g` \
-p 12345:12345 \
-v `pwd`/data:/data \
-v `pwd`/log:/log \
-v `pwd`/license:/license \
-v `pwd`/tmp:/tmp \
h2oai/dai-centos7-x86_64:TAG
```

Driverless AI will begin running:

```
-----
Welcome to H2O.ai's Driverless AI
-----
- Put data in the volume mounted at /data
- Logs are written to the volume mounted at /log/20180606-044258
```

(continues on next page)

(continued from previous page)

- Connect to Driverless AI on port 12345 inside the container
- Connect to Jupyter notebook on port 8888 inside the container

12. Connect to Driverless AI with your browser at <http://Your-Driverless-AI-Host-Machine:12345>.

Install on RHEL with CPUs

This section describes how to install and start the Driverless AI Docker image on RHEL. Note that this uses `docker` and not `nvidia-docker`.

Watch the installation video [here](#). Note that some of the images in this video may change between releases, but the installation steps remain the same.

Note: As of this writing, Driverless AI has only been tested on RHEL version 7.4.

Open a Terminal and ssh to the machine that will run Driverless AI. Once you are logged in, perform the following steps.

1. Install and start Docker EE on RHEL (if not already installed). Follow the instructions on <https://docs.docker.com/engine/installation/linux/docker-ee/rhel/>.

Alternatively, you can run on Docker CE.

```
sudo yum install -y yum-utils
sudo yum-config-manager --add-repo https://download.docker.com/linux/centos/
  docker-ce.repo
sudo yum makecache fast
sudo yum -y install docker-ce
sudo systemctl start docker
```

2. On the machine that is running Docker EE, retrieve the Driverless AI Docker image from <https://www.h2o.ai/download/>.

3. Set up a directory for the version of Driverless AI on the host machine:

```
# Set up directory with the version name
mkdir dai-1.9.0
```

4. Load the Driverless AI Docker image inside the new directory:

```
# Load the Driverless AI Docker image
docker load < dai-docker-centos7-x86_64-1.9.0-10.0.tar.gz
```

5. Set up the data, log, license, and tmp directories (within the new directory):

```
# cd into the directory associated with your version of Driverless AI
cd dai-1.9.0

# Set up the data, log, license, and tmp directories on the host machine
mkdir data
mkdir log
mkdir license
mkdir tmp
```

6. Copy data into the **data** directory on the host. The data will be visible inside the Docker container at `/<user-home>/data`.

7. Run `docker images` to find the image tag.

- Start the Driverless AI Docker image. Note that GPU support will not be available.

```
$ docker run \
--pid=host \
--init \
--rm \
--shm-size=256m \
-u `id -u`:`id -g` \
-p 12345:12345 \
-v `pwd`/data:/data \
-v `pwd`/log:/log \
-v `pwd`/license:/license \
-v `pwd`/tmp:/tmp \
-v /etc/passwd:/etc/passwd:ro \
-v /etc/group:/etc/group:ro \
h2oai/dai-centos7-x86_64:1.9.0-cuda10.0.23
```

Driverless AI will begin running:

```
-----
Welcome to H2O.ai's Driverless AI
-----

- Put data in the volume mounted at /data
- Logs are written to the volume mounted at /log/20180606-044258
- Connect to Driverless AI on port 12345 inside the container
- Connect to Jupyter notebook on port 8888 inside the container
```

- Connect to Driverless AI with your browser at <http://Your-Driverless-AI-Host-Machine:12345>.

Stopping the Docker Image

To stop the Driverless AI Docker image, type **Ctrl + C** in the Terminal (Mac OS X) or PowerShell (Windows 10) window that is running the Driverless AI Docker image.

Upgrading the Docker Image

This section provides instructions for upgrading Driverless AI versions that were installed in a Docker container. These steps ensure that existing experiments are saved.

WARNING: Experiments, MLIs, and MOJOS reside in the Driverless AI tmp directory and are not automatically upgraded when Driverless AI is upgraded.

- Build ML models before upgrading.
- Build MOJO pipelines before upgrading.
- Stop Driverless AI and make a backup of your Driverless AI tmp directory before upgrading.

If you did not build ML on a model before upgrading Driverless AI, then you will not be able to view ML on that model after upgrading. Before upgrading, be sure to run ML jobs on models that you want to continue to interpret in future releases. If that ML job appears in the list of Interpreted Models in your current version, then it will be retained after upgrading.

If you did not build a MOJO pipeline on a model before upgrading Driverless AI, then you will not be able to build a MOJO pipeline on that model after upgrading. Before upgrading, be sure to build MOJO pipelines on all desired models and then back up your Driverless AI tmp directory.

Note: Stop Driverless AI if it is still running.

Requirements

As of 1.7.0, CUDA 9 is no longer supported. Your host environment must have CUDA 10.0 or later with NVIDIA drivers >= 440.82 installed (GPU only). Driverless AI ships with its own CUDA libraries, but the driver must exist in the host environment. Go to <https://www.nvidia.com/Download/index.aspx> to get the latest NVIDIA Tesla V/P/K series driver.

Upgrade Steps

1. SSH into the IP address of the machine that is running Driverless AI.
2. Set up a directory for the version of Driverless AI on the host machine:

```
# Set up directory with the version name
mkdir dai-1.9.0

# cd into the new directory
cd dai-1.9.0
```

3. Retrieve the Driverless AI package from <https://www.h2o.ai/download/> and add it to the new directory.
4. Load the Driverless AI Docker image inside the new directory:

```
# Load the Driverless AI docker image
docker load < dai-docker-centos7-x86_64-1.9.0-10.0.tar.gz
```

5. Copy the data, log, license, and tmp directories from the previous Driverless AI directory to the new Driverless AI directory:

```
# Copy the data, log, license, and tmp directories on the host machine
cp -a dai_rel_1.4.2/data dai-1.9.0/data
cp -a dai_rel_1.4.2/log dai-1.9.0/log
cp -a dai_rel_1.4.2/license dai-1.9.0/license
cp -a dai_rel_1.4.2/tmp dai-1.9.0/tmp
```

At this point, your experiments from the previous versions will be visible inside the Docker container.

6. Use `docker images` to find the new image tag.
7. Start the Driverless AI Docker image.
8. Connect to Driverless AI with your browser at <http://Your-Driverless-AI-Host-Machine:12345>.

Install on NVIDIA GPU Cloud/NGC Registry

Driverless AI is supported on the following NVIDIA DGX products, and the installation steps for each platform are the same.

- NVIDIA GPU Cloud
- NVIDIA DGX-1
- NVIDIA DGX-2
- NVIDIA DGX Station

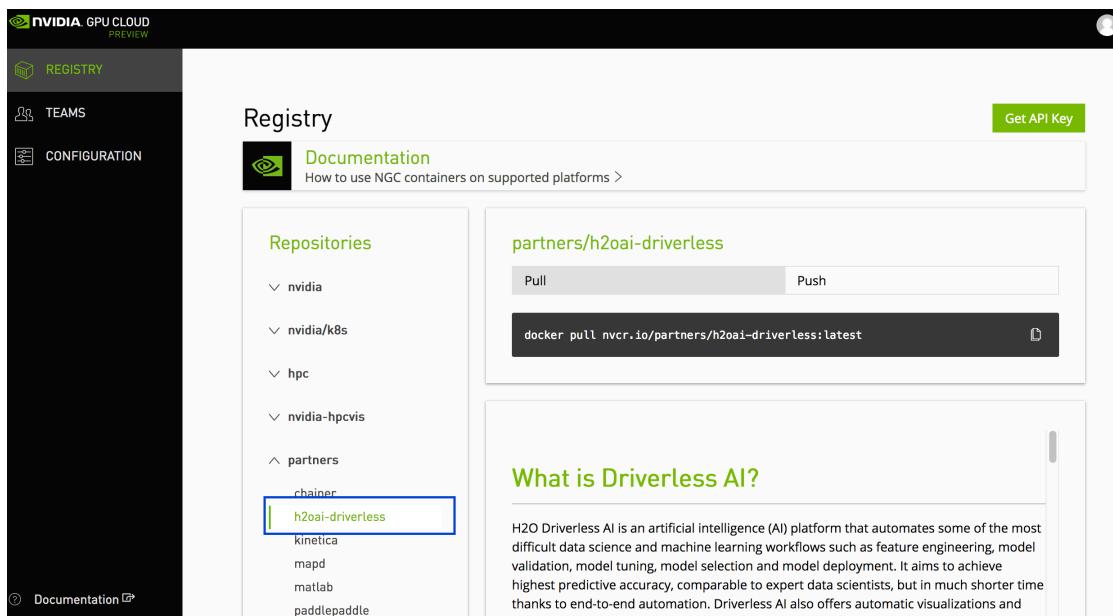
Environment

Provider	GPUs	Min Memory	Suitable for
NVIDIA GPU Cloud	Yes		Serious use
NVIDIA DGX-1/DGX-2	Yes	128 GB	Serious use
NVIDIA DGX Station	Yes	64 GB	Serious Use

Installing the NVIDIA NGC Registry

Note: These installation instructions assume that you are running on an NVIDIA DGX machine. Driverless AI is only available in the NGC registry for DGX machines.

1. Log in to your NVIDIA GPU Cloud account at <https://ngc.nvidia.com/registry>. (Note that NVIDIA Compute is no longer supported by NVIDIA.)
2. In the **Registry > Partners** menu, select **h2oai-driverless**.



3. At the bottom of the screen, select one of the H2O Driverless AI tags to retrieve the pull command.

The screenshot shows the nvcr.io Docker registry interface. On the left, there's a sidebar with a tree view of partners: partners, chainer, h2oai-driverless (which is selected and highlighted in green), kinetica, mapd, matlab, and paddlepaddle. The main content area has a title "What is Driverless AI?" followed by a detailed description of the H2O Driverless AI platform. Below the description is a table showing two Docker image tags: "latest" and "cuda9-1.0.19".

TAG	SIZE	LAST MODIFIED	PULL
latest	2 GB	March 9, 2018	
cuda9-1.0.19	2 GB	March 9, 2018	

4. On your NVIDIA DGX machine, open a command prompt and use the specified pull command to retrieve the Driverless AI image. For example:

```
docker pull nvcr.io/nvidia_partners/h2o-driverless-ai:latest
```

5. Set up a directory for the version of Driverless AI on the host machine:

```
# Set up directory with the version name  
mkdir dai-1.9.0
```

6. Set up the data, log, license, and tmp directories on the host machine:

```
# cd into the directory associated with the selected version of Driverless AI  
cd dai-1.9.0  
  
# Set up the data, log, license, and tmp directories on the host machine  
mkdir data  
mkdir log  
mkdir license  
mkdir tmp
```

7. At this point, you can copy data into the data directory on the host machine. The data will be visible inside the Docker container.
8. Enable persistence of the GPU. Note that this only needs to be run once. Refer to the following for more information: <http://docs.nvidia.com/deploy/driver-persistence/index.html>.

```
sudo nvidia-persistenced --persistence-mode
```

9. Run docker images to find the new image tag.
10. Start the Driverless AI Docker image and replace TAG below with the image tag. Depending on your install version, use the docker run --runtime=nvidia (>= Docker 19.03) or nvidia-docker (< Docker 19.03) command.

Note: Use docker version to check which version of Docker you are using.

>= Docker 19.03

< Docker 19.03

```
# Start the Driverless AI Docker image
docker run --runtime=nvidia \
    --pid=host \
    --init \
    --rm \
    --shm-size=256m \
    -u `id -u`:`id -g` \
    -p 12345:12345 \
    -v `pwd`/data:/data \
    -v `pwd`/log:/log \
    -v `pwd`/license:/license \
    -v `pwd`/tmp:/tmp \
    h2oai/dai-centos7-x86_64:TAG
```

```
# Start the Driverless AI Docker image
nvidia-docker run \
    --pid=host \
    --init \
    --rm \
    --shm-size=256m \
    -u `id -u`:`id -g` \
    -p 12345:12345 \
    -v `pwd`/data:/data \
    -v `pwd`/log:/log \
    -v `pwd`/license:/license \
    -v `pwd`/tmp:/tmp \
    h2oai/dai-centos7-x86_64:TAG
```

Driverless AI will begin running:

```
-----
Welcome to H2O.ai's Driverless AI
-----
- Put data in the volume mounted at /data
- Logs are written to the volume mounted at /log/20180606-044258
- Connect to Driverless AI on port 12345 inside the container
- Connect to Jupyter notebook on port 8888 inside the container
```

11. Connect to Driverless AI with your browser:

```
http://Your-Driverless-AI-Host-Machine:12345
```

Stopping Driverless AI

Use Ctrl+C to stop Driverless AI.

Upgrading Driverless AI

The steps for upgrading Driverless AI on an NVIDIA DGX system are similar to the installation steps.

WARNINGS:

- This release deprecates experiments and MLIs models from 1.7.0 and earlier.
- Experiments, MLIs, and MOJOs reside in the Driverless AI tmp directory and are not automatically upgraded when Driverless AI is upgraded. We recommend you take the following steps before upgrading.
 - Build MLIs models before upgrading.
 - Build MOJO pipelines before upgrading.
 - Stop Driverless AI and make a backup of your Driverless AI tmp directory before upgrading.

If you did not build MLIs on a model before upgrading Driverless AI, then you will not be able to view MLIs on that model after upgrading. Before upgrading, be sure to run MLIs jobs on models that you want to continue to interpret in future releases. If that MLIs job appears in the list of Interpreted Models in your current version, then it will be retained after upgrading.

If you did not build a MOJO pipeline on a model before upgrading Driverless AI, then you will not be able to build a MOJO pipeline on that model after upgrading. Before upgrading, be sure to build MOJO pipelines on all desired models and then back up your Driverless AI tmp directory.

The upgrade process inherits the service user and group from /etc/dai/User.conf and /etc/dai/Group.conf. You do not need to manually specify the DAI_USER or DAI_GROUP environment variables during an upgrade.

Note: Use Ctrl+C to stop Driverless AI if it is still running.

Requirements

As of 1.7.0, CUDA 9 is no longer supported. Your host environment must have CUDA 10.0 or later with NVIDIA drivers >= 440.82 installed (GPU only). Driverless AI ships with its own CUDA libraries, but the driver must exist in the host environment. Go to <https://www.nvidia.com/Download/index.aspx> to get the latest NVIDIA Tesla V/P/K series driver.

Upgrade Steps

1. On your NVIDIA DGX machine, create a directory for the new Driverless AI version.
2. Copy the data, log, license, and tmp directories from the previous Driverless AI directory into the new Driverless AI directory.
3. Run `docker pull nvcr.io/h2oai/h2oai-driverless-ai:latest` to retrieve the latest Driverless AI version.
4. Start the Driverless AI Docker image.
5. Connect to Driverless AI with your browser at <http://Your-Driverless-AI-Host-Machine:12345>.

11.1.2 Linux RPMs

For Linux machines that will not use the Docker image or DEB, an RPM installation is available for the following environments:

- x86_64 RHEL 7
- CentOS 7
- SLES 12

The installation steps assume that you have a license key for Driverless AI. For information on how to obtain a license key for Driverless AI, visit <https://www.h2o.ai/products/h2o-driverless-ai/>. Once obtained, you will be prompted to paste the license key into the Driverless AI UI when you first log in, or you can save it as a .sig file and place it in the license folder that you will create during the installation process.

Environment

Operating System	Min Mem
RHEL with GPUs	64 GB
RHEL with CPUs	64 GB
CentOS 7 with GPUS	64 GB
CentOS 7 with CPUs	64 GB
SLES 12 with GPUs	64 GB
SLES 12 with CPUs	64 GB

Requirements

- RedHat 7/CentOS 7/SLES 12
- CUDA 10 or later with NVIDIA drivers >= 440.82 (GPU only)
- cuDNN >=7.2.1 (Required for TensorFlow support on GPUs.)
- OpenCL (Required for LightGBM support on GPUs.)
- Driverless AI RPM, available from <https://www.h2o.ai/download/>

About the Install

- The ‘dai’ service user is created locally (in /etc/passwd) if it is not found by ‘getent passwd’. You can override the user by providing the DAI_USER environment variable during rpm or dpkg installation.
- The ‘dai’ service group is created locally (in /etc/group) if it is not found by ‘getent group’. You can override the group by providing the DAI_GROUP environment variable during rpm or dpkg installation.
- Configuration files are put in **/etc/dai** and owned by the ‘root’ user:
 - **/etc/dai/config.toml**: Driverless AI config file (See *Using the config.toml File* section for details)
 - **/etc/dai/User.conf**: Systemd config file specifying the service user
 - **/etc/dai/Group.conf**: Systemd config file specifying the service group
 - **/etc/dai/EnvironmentFile.conf**: Systemd config file specifying (optional) environment variable overrides
- Software files are put in **/opt/h2oai/dai** and owned by the ‘root’ user
- The following directories are owned by the service user so they can be updated by the running software:

- **/opt/h2oai/dai/home:** The application’s home directory (license key files are stored here)
 - **/opt/h2oai/dai/tmp:** Experiments and imported data are stored here
 - **/opt/h2oai/dai/log:** Log files go here if you are **not** using systemd (if you are using systemd, then the use the standard journalctl tool)
- By default, Driverless AI looks for a license key in **/opt/h2oai/dai/home/.driverlessai/license.sig**. If you are installing Driverless AI programmatically, you can copy a license key file to that location. If no license key is found, the application will interactively guide you to add one from the Web UI.
 - systemd unit files are put in **/usr/lib/systemd/system**
 - Symbolic links to the configuration files in **/etc/dai** files are put in **/etc/systemd/system**

If your environment is running an operational systemd, that is the preferred way to manage Driverless AI. The package installs the following systemd services and a wrapper service:

- **dai:** Wrapper service that starts/stops the other three services
- **dai-dai:** Main Driverless AI process
- **dai-h2o:** H2O-3 helper process used by Driverless AI
- **dai-procsy:** Procsy helper process used by Driverless AI
- **dai-vis-server:** Visualization server helper process used by Driverless AI

If you don’t have systemd, you can also use the provided run script to start Driverless AI.

Installing Driverless AI

Run the following commands to install the Driverless AI RPM.

```
# Install Driverless AI.  
sudo rpm -i dai-1.9.0-1.x86_64.rpm
```

Note: For RHEL 7.5, it is necessary to upgrade library glib2:

```
sudo yum upgrade glib2
```

By default, the Driverless AI processes are owned by the ‘dai’ user and ‘dai’ group. You can optionally specify a different service user and group as shown below. Replace <myuser> and <mygroup> as appropriate.

```
# Temporarily specify service user and group when installing Driverless AI.  
# rpm saves these for systemd in the /etc/dai/User.conf and /etc/dai/Group.conf files.  
sudo DAI_USER=myuser DAI_GROUP=mygroup rpm -i dai-1.9.0-1.x86_64.rpm
```

You may now optionally make changes to **/etc/dai/config.toml**.

Starting Driverless AI

If you have systemd (preferred):

```
# If you are running in multinode (worker_info="multinode"), you must start minio before starting Driverless AI
sudo systemctl start dai-minio

# Start Driverless AI.
sudo systemctl start dai
```

If you do not have systemd:

```
# Start Driverless AI.
sudo -H -u dai /opt/h2oai/dai/run-dai.sh

# No need to start minio separately. When starting Driverless AI manually, the command above takes care of that.
```

Starting NVIDIA Persistence Mode

If you have NVIDIA GPUs, you must run the following NVIDIA command. This command needs to be run every reboot. For more information: [http://docs.nvidia.com/deploy driver-persistence/index.html](http://docs.nvidia.com/deploy	driver-persistence/index.html).

```
sudo nvidia-persistenced --persistence-mode
```

Installing CUDA with NVIDIA drivers

Before installing CUDA you will need to make sure you have already installed wget, gcc, make, and elfutils-libelf-devel:

```
sudo yum -y install wget
sudo yum -y install gcc
sudo yum -y install make
sudo yum -y install elfutils-libelf-devel
```

Next, visit <https://docs.nvidia.com/cuda/cuda-installation-guide-linux/index.html> for instructions on installing CUDA. It is recommended that you use the runfile method of installation.

If prompted to select what tools you would like to install, select Drivers only.

Installing OpenCL

OpenCL is required in order to run LightGBM on GPUs. Run the following for Centos7/RH7 based systems using yum and x86.

```
sudo yum -y clean all
sudo yum -y makecache
sudo yum -y update
wget http://dl.fedoraproject.org/pub/epel/7/x86_64/Packages/c/clinfo-2.1.17.02.09-1.el7.x86_64.rpm
wget http://dl.fedoraproject.org/pub/epel/7/x86_64/Packages/o/ocl-icd-2.2.12-1.el7.x86_64.rpm
sudo rpm -if ocl-icd-2.2.12-1.el7.x86_64.rpm
```

(continues on next page)

(continued from previous page)

```
sudo rpm -if clinfo-2.1.17.02.09-1.el7.x86_64.rpm  
clinfo  
  
mkdir -p /etc/OpenCL/vendors && \  
    echo "libnvidia-opencl.so.1" > /etc/OpenCL/vendors/nvidia.icd
```

Installing cuDNN

cuDNN is required for TensorFlow support on GPUs. See <https://docs.nvidia.com/deeplearning/sdk/cudnn-install/index.html> and follow the instructions for installing the latest cuDNN.

Note: As of this writing, there is an error in the above instructions - the command `sudo yum install libcudnn7-dev` will fail because the `libcudnn7-dev` is a package only available on Debian based distros. You must instead run `sudo yum install libcudnn7-devel`.

Looking at Driverless AI log files

If you have systemd (preferred):

```
sudo systemctl status dai-dai  
sudo systemctl status dai-h2o  
sudo systemctl status dai-procsy  
sudo systemctl status dai-vis-server  
sudo journalctl -u dai-dai  
sudo journalctl -u dai-h2o  
sudo journalctl -u dai-procsy  
sudo journalctl -u dai-vis-server
```

If you do not have systemd:

```
sudo less /opt/h2oai/dai/log/dai.log  
sudo less /opt/h2oai/dai/log/h2o.log  
sudo less /opt/h2oai/dai/log/procsy.log  
sudo less /opt/h2oai/dai/log/vis-server.log
```

Stopping Driverless AI

If you have systemd (preferred):

```
# Stop Driverless AI.  
sudo systemctl stop dai  
  
# The processes should now be stopped. Verify.  
sudo ps -u dai
```

If you do not have systemd:

```
# Stop Driverless AI.  
sudo pkill -U dai  
  
# The processes should now be stopped. Verify.  
sudo ps -u dai
```

Upgrading Driverless AI

WARNINGS:

- This release deprecates experiments and MLIs models from 1.7.0 and earlier.
- Experiments, MLIs, and MOJOS reside in the Driverless AI tmp directory and are not automatically upgraded when Driverless AI is upgraded. We recommend you take the following steps before upgrading.
 - Build MLI models before upgrading.
 - Build MOJO pipelines before upgrading.
 - Stop Driverless AI and make a backup of your Driverless AI tmp directory before upgrading.

If you did not build MLI on a model before upgrading Driverless AI, then you will not be able to view MLI on that model after upgrading. Before upgrading, be sure to run MLI jobs on models that you want to continue to interpret in future releases. If that MLI job appears in the list of Interpreted Models in your current version, then it will be retained after upgrading.

If you did not build a MOJO pipeline on a model before upgrading Driverless AI, then you will not be able to build a MOJO pipeline on that model after upgrading. Before upgrading, be sure to build MOJO pipelines on all desired models and then back up your Driverless AI tmp directory.

The upgrade process inherits the service user and group from /etc/dai/User.conf and /etc/dai/Group.conf. You do not need to manually specify the DAI_USER or DAI_GROUP environment variables during an upgrade.

Requirements

As of 1.7.0, CUDA 9 is no longer supported. Your host environment must have CUDA 10.0 or later with NVIDIA drivers >= 440.82 installed (GPU only). Driverless AI ships with its own CUDA libraries, but the driver must exist in the host environment. Go to <https://www.nvidia.com/Download/index.aspx> to get the latest NVIDIA Tesla V/P/K series driver.

Upgrade Steps

If you have systemd (preferred):

```
# Stop Driverless AI.
sudo systemctl stop dai

# The processes should now be stopped. Verify.
sudo ps -u dai

# Make a backup of /opt/h2oai/dai/tmp directory at this time.

# Upgrade and restart.
sudo rpm -U dai-1.9.0-1.x86_64.rpm
sudo systemctl daemon-reload
sudo systemctl start dai
```

If you do not have systemd:

```
# Stop Driverless AI.
sudo pkill -U dai

# The processes should now be stopped. Verify.
```

(continues on next page)

(continued from previous page)

```
sudo ps -u dai

# Make a backup of /opt/h2oai/dai/tmp directory at this time.

# Upgrade and restart.
sudo rpm -U dai-1.9.0-1.x86_64.rpm
sudo -H -u dai /opt/h2oai/dai/run-dai.sh
```

Uninstalling Driverless AI

If you have systemd (preferred):

```
# Stop Driverless AI.
sudo systemctl stop dai

# The processes should now be stopped. Verify.
sudo ps -u dai

# Uninstall.
sudo rpm -e dai
```

If you do not have systemd:

```
# Stop Driverless AI.
sudo pkill -U dai

# The processes should now be stopped. Verify.
sudo ps -u dai

# Uninstall.
sudo rpm -e dai
```

CAUTION! At this point you can optionally completely remove all remaining files, including the database. (This cannot be undone.)

```
sudo rm -rf /opt/h2oai/dai
sudo rm -rf /etc/dai
```

Note: The UID and GID are not removed during the uninstall process. These can be removed with `userdel` and `usergroup`. However, we DO NOT recommend removing the UID and GID if you plan to re-install Driverless AI. If you remove the UID and GID and then reinstall Driverless AI, the UID and GID will likely be re-assigned to a different (unrelated) user/group in the future; this may cause confusion if there are any remaining files on the filesystem referring to the deleted user or group.

11.1.3 Linux DEBs

For Linux machines that will not use the Docker image or RPM, a DEB installation is available for x86_64 Ubuntu 16.04/18.04.

The installation steps assume that you have a license key for Driverless AI. For information on how to obtain a license key for Driverless AI, visit <https://www.h2o.ai/products/h2o-driverless-ai/>. Once obtained, you will be prompted to paste the license key into the Driverless AI UI when you first log in, or you can save it as a .sig file and place it in the license folder that you will create during the installation process.

Environment

Operating System	Min Mem
Ubuntu with GPUs	64 GB
Ubuntu with CPUs	64 GB

Requirements

- Ubuntu 16.04/Ubuntu 18.04
- CUDA 10 or later with CUDA drivers and NVIDIA drivers >= 440.82 (GPU only). CUDA 10 is included in the Driverless AI package. Refer to the following for more information on how to install the CUDA and NVIDIA drivers:
 - <https://developer.nvidia.com/cuda-downloads/>
 - <https://docs.nvidia.com/cuda/>
- cuDNN >=7.2.1 (Required for TensorFlow support on GPUs.)
- OpenCL (Required for LightGBM support on GPUs.)
- Driverless AI DEB, available from <https://www.h2o.ai/download/>

About the Install

- The ‘dai’ service user is created locally (in /etc/passwd) if it is not found by ‘getent passwd’. You can override the user by providing the DAI_USER environment variable during rpm or dpkg installation.
- The ‘dai’ service group is created locally (in /etc/group) if it is not found by ‘getent group’. You can override the group by providing the DAI_GROUP environment variable during rpm or dpkg installation.
- Configuration files are put in **/etc/dai** and owned by the ‘root’ user:
 - **/etc/dai/config.toml**: Driverless AI config file (See *Using the config.toml File* section for details)
 - **/etc/dai/User.conf**: Systemd config file specifying the service user
 - **/etc/dai/Group.conf**: Systemd config file specifying the service group
 - **/etc/dai/EnvironmentFile.conf**: Systemd config file specifying (optional) environment variable overrides
- Software files are put in **/opt/h2oai/dai** and owned by the ‘root’ user
- The following directories are owned by the service user so they can be updated by the running software:
 - **/opt/h2oai/dai/home**: The application’s home directory (license key files are stored here)
 - **/opt/h2oai/dai/tmp**: Experiments and imported data are stored here

- **/opt/h2oai/dai/log:** Log files go here if you are **not** using systemd (if you are using systemd, then the use the standard journalctl tool)
- By default, Driverless AI looks for a license key in **/opt/h2oai/dai/home/.driverlessai/license.sig**. If you are installing Driverless AI programmatically, you can copy a license key file to that location. If no license key is found, the application will interactively guide you to add one from the Web UI.
- systemd unit files are put in **/usr/lib/systemd/system**
- Symbolic links to the configuration files in **/etc/dai** files are put in **/etc/systemd/system**

If your environment is running an operational systemd, that is the preferred way to manage Driverless AI. The package installs the following systemd services and a wrapper service:

- **dai:** Wrapper service that starts/stops the other three services
- **dai-dai:** Main Driverless AI process
- **dai-h2o:** H2O-3 helper process used by Driverless AI
- **dai-procsy:** Procsy helper process used by Driverless AI
- **dai-vis-server:** Visualization server helper process used by Driverless AI

If you don't have systemd, you can also use the provided run script to start Driverless AI.

Starting NVIDIA Persistence Mode (GPU only)

If you have NVIDIA GPUs, you must run the following NVIDIA command. This command needs to be run every reboot. For more information: <http://docs.nvidia.com/Deploy/Driver-Persistence/index.html>.

```
sudo nvidia-persistenced --persistence-mode
```

Install OpenCL

OpenCL is required in order to run LightGBM on GPUs. Run the following for Ubuntu-based systems.

```
sudo apt-get install opencl-headers clinfo ocl-icd-opencl-dev  
mkdir -p /etc/OpenCL/vendors && \  
echo "libnvidia-opencl.so.1" > /etc/OpenCL/vendors/nvidia.icd
```

Installing the Driverless AI Linux DEB

Run the following commands to install the Driverless AI DEB.

```
# Install Driverless AI.  
sudo dpkg -i dai_1.9.0_amd64.deb
```

By default, the Driverless AI processes are owned by the ‘dai’ user and ‘dai’ group. You can optionally specify a different service user and group as shown below. Replace <myuser> and <mygroup> as appropriate.

```
# Temporarily specify service user and group when installing Driverless AI.  
# dpkg saves these for systemd in the /etc/dai/User.conf and /etc/dai/Group.conf  
# files.  
sudo DAI_USER=myuser DAI_GROUP=mygroup dpkg -i dai_1.9.0_amd64.deb
```

You may now optionally make changes to **/etc/dai/config.toml**.

Starting Driverless AI

If you have systemd (preferred):

```
# If you are running in multinode (worker_info="multinode"), you must start minio_
↪before starting Driverless AI
sudo systemctl start dai-minio

# Start Driverless AI.
sudo systemctl start dai
```

If you do not have systemd:

```
# Start Driverless AI.
sudo -H -u dai /opt/h2oai/dai/run-dai.sh

# No need to start minio separately. When starting Driverless AI manually, the_
↪command above takes care of that.
```

Looking at Driverless AI log files

If you have systemd (preferred):

```
sudo systemctl status dai-dai
sudo systemctl status dai-h2o
sudo systemctl status dai-procsy
sudo systemctl status dai-vis-server
sudo journalctl -u dai-dai
sudo journalctl -u dai-h2o
sudo journalctl -u dai-procsy
sudo journalctl -u dai-vis-server
```

If you do not have systemd:

```
sudo less /opt/h2oai/dai/log/dai.log
sudo less /opt/h2oai/dai/log/h2o.log
sudo less /opt/h2oai/dai/log/procsy.log
sudo less /opt/h2oai/dai/log/vis-server.log
```

Stopping Driverless AI

If you have systemd (preferred):

```
# Stop Driverless AI.
sudo systemctl stop dai

# The processes should now be stopped. Verify.
sudo ps -u dai
```

If you do not have systemd:

```
# Stop Driverless AI.
sudo pkill -U dai
```

(continues on next page)

(continued from previous page)

```
# The processes should now be stopped. Verify.  
sudo ps -u dai
```

Upgrading Driverless AI

WARNINGS:

- This release deprecates experiments and MLIs models from 1.7.0 and earlier.
- Experiments, MLIs, and MOJOS reside in the Driverless AI tmp directory and are not automatically upgraded when Driverless AI is upgraded. We recommend you take the following steps before upgrading.
 - Build MLI models before upgrading.
 - Build MOJO pipelines before upgrading.
 - Stop Driverless AI and make a backup of your Driverless AI tmp directory before upgrading.

If you did not build MLI on a model before upgrading Driverless AI, then you will not be able to view MLI on that model after upgrading. Before upgrading, be sure to run MLI jobs on models that you want to continue to interpret in future releases. If that MLI job appears in the list of Interpreted Models in your current version, then it will be retained after upgrading.

If you did not build a MOJO pipeline on a model before upgrading Driverless AI, then you will not be able to build a MOJO pipeline on that model after upgrading. Before upgrading, be sure to build MOJO pipelines on all desired models and then back up your Driverless AI tmp directory.

The upgrade process inherits the service user and group from /etc/dai/User.conf and /etc/dai/Group.conf. You do not need to manually specify the DAI_USER or DAI_GROUP environment variables during an upgrade.

Requirements

As of 1.7.0, CUDA 9 is no longer supported. Your host environment must have CUDA 10.0 or later with NVIDIA drivers >= 440.82 installed (GPU only). Driverless AI ships with its own CUDA libraries, but the driver must exist in the host environment. Go to <https://www.nvidia.com/Download/index.aspx> to get the latest NVIDIA Tesla V/P/K series driver.

Upgrade Steps

If you have systemd (preferred):

```
# Stop Driverless AI.  
sudo systemctl stop dai  
  
# Make a backup of /opt/h2oai/dai/tmp directory at this time.  
  
# Upgrade Driverless AI.  
sudo dpkg -i dai_1.9.0_amd64.deb  
sudo systemctl daemon-reload  
sudo systemctl start dai
```

If you do not have systemd:

```
# Stop Driverless AI.
sudo pkill -U dai

# The processes should now be stopped. Verify.
sudo ps -u dai

# Make a backup of /opt/h2oai/dai/tmp directory at this time. If you do not, all
# previous data will be lost.

# Upgrade and restart.
sudo dpkg -i dai_1.9.0_amd64.deb
sudo -H -u dai /opt/h2oai/dai/run-dai.sh
```

Uninstalling Driverless AI

If you have systemd (preferred):

```
# Stop Driverless AI.
sudo systemctl stop dai

# The processes should now be stopped. Verify.
sudo ps -u dai

# Uninstall Driverless AI.
sudo dpkg -r dai

# Purge Driverless AI.
sudo dpkg -P dai
```

If you do not have systemd:

```
# Stop Driverless AI.
sudo pkill -U dai

# The processes should now be stopped. Verify.
sudo ps -u dai

# Uninstall Driverless AI.
sudo dpkg -r dai

# Purge Driverless AI.
sudo dpkg -P dai
```

CAUTION! At this point you can optionally completely remove all remaining files, including the database (this cannot be undone):

```
sudo rm -rf /opt/h2oai/dai
sudo rm -rf /etc/dai
```

Note: The UID and GID are not removed during the uninstall process. These can be removed with `userdel` and `usergroup`. However, we DO NOT recommend removing the UID and GID if you plan to re-install Driverless AI. If you remove the UID and GID and then reinstall Driverless AI, the UID and GID will likely be re-assigned to a different (unrelated) user/group in the future; this may cause confusion if there are any remaining files on the filesystem referring to the deleted user or group.

Common Problems

Start of Driverless AI fails on the message ``Segmentation fault (core dumped)`` on Ubuntu 18.

This problem is caused by the font `NotoColorEmoji.ttf`, which cannot be processed by the Python matplotlib library. A workaround is to disable the font by renaming it. (Do not use `fontconfig` because it is ignored by matplotlib.) The following will print out the command that should be executed.

```
sudo find / -name "NotoColorEmoji.ttf" 2>/dev/null | xargs -I{} echo sudo mv {} {}.  
↳ backup
```

11.1.4 Linux TAR SH

The Driverless AI software is available for use in pure user-mode environments as a self-extracting TAR SH archive. This form of installation does not require a privileged user to install or to run.

This artifact has the same compatibility matrix as the RPM and DEB packages (combined), it just comes packaged slightly differently. See those sections for a full list of supported environments.

The installation steps assume that you have a license key for Driverless AI. For information on how to obtain a license key for Driverless AI, visit <https://www.h2o.ai/products/h2o-driverless-ai/>. Once obtained, you will be prompted to paste the license key into the Driverless AI UI when you first log in.

Requirements

- RedHat 7 or Ubuntu 16.04
- CUDA 10 or later with NVIDIA drivers >= 440.82 (GPU only)
- cuDNN >=7.2.1 (Required for TensorFlow support on GPUs.)
- OpenCL (Required for LightGBM support on GPUs.)
- Driverless AI TAR SH, available from <https://www.h2o.ai/download/>

Installing Driverless AI

Run the following commands to install the Driverless AI RPM.

```
# Install Driverless AI.  
chmod 755 dai-1.9.0-linux-ppc64le.sh  
./dai-1.9.0-linux-ppc64le.sh
```

You may now cd to the unpacked directory and optionally make changes to `config.toml`.

Starting Driverless AI

```
# Start Driverless AI.  
./run-dai.sh
```

Starting NVIDIA Persistence Mode

If you have NVIDIA GPUs, you must run the following NVIDIA command. This command needs to be run every reboot. For more information: <http://docs.nvidia.com/deploy/driver-persistence/index.html>.

```
sudo nvidia-persistenced --persistence-mode
```

Install OpenCL

OpenCL is required in order to run LightGBM on GPUs. Run the following for Centos7/RH7 based systems using yum and x86.

```
yum -y clean all
yum -y makecache
yum -y update
wget http://dl.fedoraproject.org/pub/epel/7/x86_64/Packages/c/clinfo-2.1.17.02.09-1.el7.x86_64.rpm
wget http://dl.fedoraproject.org/pub/epel/7/x86_64/Packages/o/ocl-icd-2.2.12-1.el7.x86_64.rpm
rpm -if clinfo-2.1.17.02.09-1.el7.x86_64.rpm
rpm -if ocl-icd-2.2.12-1.el7.x86_64.rpm
clinfo

mkdir -p /etc/OpenCL/vendors && \
echo "libnvidia-opencl.so.1" > /etc/OpenCL/vendors/nvidia.icd
```

Looking at Driverless AI log files

```
less log/dai.log
less log/h2o.log
less log/procsy.log
less log/vis-server.log
```

Stopping Driverless AI

```
# Stop Driverless AI.
./kill-dai.sh
```

Uninstalling Driverless AI

To uninstall Driverless AI, just remove the directory created by the unpacking process. By default, all files for Driverless AI are contained within this directory.

Upgrading Driverless AI

WARNINGS:

- This release deprecates experiments and MLIs models from 1.7.0 and earlier.
- Experiments, MLIs, and MOJOS reside in the Driverless AI tmp directory and are not automatically upgraded when Driverless AI is upgraded. We recommend you take the following steps before upgrading.
 - Build MLI models before upgrading.
 - Build MOJO pipelines before upgrading.
 - Stop Driverless AI and make a backup of your Driverless AI tmp directory before upgrading.

If you did not build MLI on a model before upgrading Driverless AI, then you will not be able to view MLI on that model after upgrading. Before upgrading, be sure to run MLI jobs on models that you want to continue to interpret in future releases. If that MLI job appears in the list of Interpreted Models in your current version, then it will be retained after upgrading.

If you did not build a MOJO pipeline on a model before upgrading Driverless AI, then you will not be able to build a MOJO pipeline on that model after upgrading. Before upgrading, be sure to build MOJO pipelines on all desired models and then back up your Driverless AI tmp directory.

The upgrade process inherits the service user and group from /etc/dai/User.conf and /etc/dai/Group.conf. You do not need to manually specify the DAI_USER or DAI_GROUP environment variables during an upgrade.

Requirements

As of 1.7.0, CUDA 9 is no longer supported. Your host environment must have CUDA 10.0 or later with NVIDIA drivers >= 440.82 installed (GPU only). Driverless AI ships with its own CUDA libraries, but the driver must exist in the host environment. Go to <https://www.nvidia.com/Download/index.aspx> to get the latest NVIDIA Tesla V/P/K series driver.

Upgrade Steps

1. Stop your previous version of Driverless AI.
2. Run the self-extracting archive for the new version of Driverless AI.
3. Port any previous changes you made to your config.toml file to the newly unpacked directory.
4. Copy the tmp directory (which contains all the Driverless AI working state) from your previous Driverless AI installation into the newly upacked directory.
5. Start your newly extracted version of Driverless AI.

11.1.5 Linux in the Cloud

To simplify cloud installation, Driverless AI is provided as an AMI for the following cloud platforms:

- AWS AMI
- Azure Image
- Google Cloud

The installation steps for AWS, Azure, and Google Cloud assume that you have a license key for Driverless AI. For information on how to obtain a license key for Driverless AI, visit <https://www.h2o.ai/developerless-ai/>. Once obtained, you will be prompted to paste the license key into the Driverless AI UI when you first log in, or you can save it as a .sig file and place it in the **license** folder that you will create during the installation process.

Install on AWS

Driverless AI can be installed on Amazon AWS using the AWS Marketplace AMI or the AWS Community AMI.

Choosing an Install Method

Consider the following when choosing between the AWS Marketplace and AWS Community AMIs:

Driverless AI AWS Marketplace AMI

- Native (Debian) install based
- Certified by AWS
- Will typically lag behind our standard releases, and may require updates to work with the latest versions of Driverless AI
- Features several default configurations like default password and HTTPS configuration, which are required by AWS

Driverless AI AWS Community AMI

- Docker based
- Not certified by AWS
- Will typically have an up-to-date version of Driverless AI for both LTS and latest stable releases
- Base Driverless AI installation on Docker does not feature preset configurations

Install the Driverless AI AWS Marketplace AMI

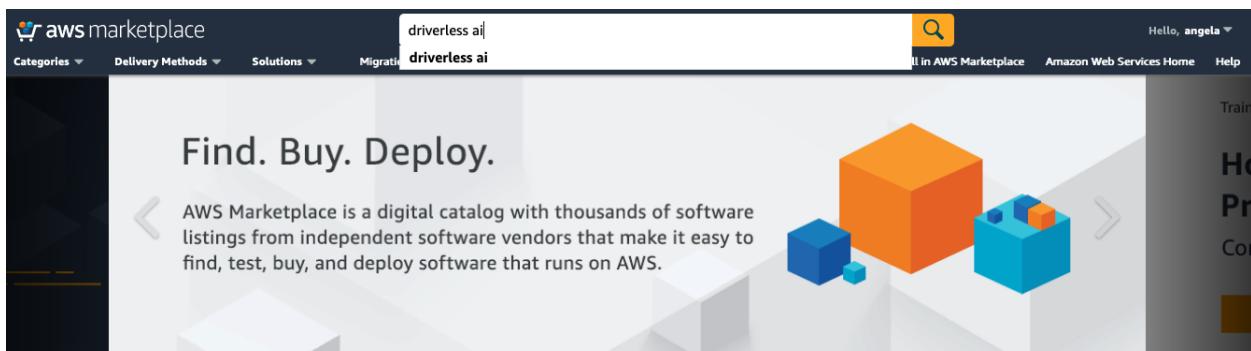
A Driverless AI AMI is available in the AWS Marketplace beginning with Driverless AI version 1.5.2. This section describes how to install and run Driverless AI through the AWS Marketplace.

Environment

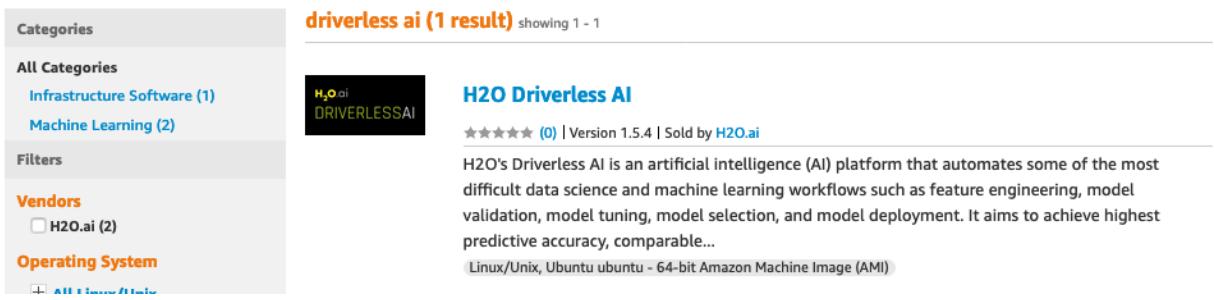
Provider	Instance Type	Num GPUs	Suitable for
AWS	p2.xlarge	1	Experimentation
	p2.8xlarge	8	Serious use
	p2.16xlarge	16	Serious use
	p3.2xlarge	1	Experimentation
	p3.8xlarge	4	Serious use
	p3.16xlarge	8	Serious use
	g3.4xlarge	1	Experimentation
	g3.8xlarge	2	Experimentation
	g3.16xlarge	4	Serious use

Installation Procedure

1. Log in to the AWS Marketplace.
2. Search for Driverless AI.



3. Select the version of Driverless AI that you want to install.



4. Scroll down to review/edit your region and the selected infrastructure and pricing.

Pricing Information

Use this tool to estimate the software and infrastructure costs based on your configuration choices. Your usage and costs might be different from this estimate. They will be reflected on your monthly AWS billing reports.

Estimating your costs

Choose your region and fulfillment option to see the pricing details. Then, modify the estimated price by choosing different instance types.

Region: US East (N. Virginia)

Fulfillment Option: 64-bit (x86) Amazon Machine Image (AMI)

Software Pricing Details

H2O Driverless AI	\$0 /hr >
running on p3.2xlarge	

Infrastructure Pricing Details

Estimated Infrastructure Cost	\$3.06 EC2/hr >
BYOL Available for customers with current licenses purchased via other channels.	

The table shows current software and infrastructure pricing for services hosted in US East (N. Virginia). Additional taxes or fees may apply.

H2O Driverless AI				
<input type="radio"/>	p2.16xlarge	\$0	\$14.4	\$14.4
<input checked="" type="radio"/>	p3.2xlarge ★Vendor Recommended	\$0	\$3.06	\$3.06
<input type="radio"/>	p3.8xlarge	\$0	\$12.24	\$12.24
<input type="radio"/>	p3.16xlarge	\$0	\$24.48	\$24.48
<input type="radio"/>	g2.2xlarge	\$0	\$0.65	\$0.65
<input type="radio"/>	g2.8xlarge	\$0	\$2.6	\$2.6

5. Return to the top and select **Continue to Subscribe**.

H2O Driverless AI

By: H2O.ai Latest Version: 1.5.4

H2O Driverless AI brings Automatic Machine Learning for the enterprise.

Linux/Unix ★★★★★ (0) BYOL

Continue to Subscribe

Save to List

Typical Total Price
\$3.060/hr

Total pricing per instance for services hosted on p3.2xlarge in US East (N. Virginia). [View Details](#)

Product Overview

H2O's Driverless AI is an artificial intelligence (AI) platform that automates some of the most difficult data science and machine learning workflows such as feature engineering, model validation, model tuning, model selection, and model deployment. It aims to achieve highest predictive accuracy, comparable to expert data scientists, but in much shorter time thanks to end-to-end automation. Driverless AI also offers automatic visualizations and machine learning interpretability (MLI). Especially in regulated industries, model transparency and explanation are just as important as predictive performance.

Highlights

- Automatic Feature Engineering
- Automatic Visualization
- Machine Learning Interpretability

6. Review the subscription, then click **Continue to Configure**.
7. If desired, change the Fulfillment Option, Software Version, and Region. Note that this page also includes the AMI ID for the selected software version. Click **Continue to Launch** when you are done.

Using Driverless AI, Release 1.9.0.1

The screenshot shows the configuration interface for H2O Driverless AI. At the top, there's a navigation bar with links for Product Detail, Subscribe, and Configure, and a prominent yellow "Continue to Launch" button. Below the navigation is a section titled "Configure this software" with a sub-section "Fulfillment Option" set to "64-bit (x86) Amazon Machine Image (AMI)". To the right, a "Pricing information" box provides an estimate of costs based on the selected configuration. Further down are sections for "Software Version" (set to "1.5.4 (Apr 03, 2019)"), "Region" (set to "US East (N. Virginia)"), and EC2 infrastructure details.

8. Review the configuration and choose a method for launching Driverless AI. **Be sure to also review the Usage Instructions. This button provides you with the login and password for launching Driverless AI.** Scroll down to the bottom of the page and click **Launch** when you are done.

The screenshot shows the launch interface for H2O Driverless AI. At the top, it displays the configuration details: Fulfillment Option (64-bit (x86) Amazon Machine Image (AMI)), Software Version (1.5.4), and Region (US East (N. Virginia)). Below this, a "Usage Instructions" button is visible. The main area is titled "Choose Action" and includes a dropdown menu set to "Launch from Website". A note below the dropdown says "Choose this action to launch from this website".

You will receive a “Success” message when the image launches successfully.

Congratulations! An instance of this software is successfully deployed on EC2!

AMI ID: ami-0cf5a8a9ab31c92f7 [\(View Launch Configuration Details\)](#)

You can view this instance on [EC2 Console](#). You can also view all instances on [Your Software](#). Software and AWS hourly usage fees apply when the instance is running and will appear on your monthly bill.

You can launch this configuration again below or go to the [configuration page](#) to start a new one.

Configuration Details

Fulfillment Option	64-bit (x86) Amazon Machine Image (AMI) H2O Driverless AI <i>running on p3.2xlarge</i>
Software Version	1.5.4
Region	US East (N. Virginia)

Usage Instructions

Starting Driverless AI

This section describes how to start Driverless AI after the Marketplace AMI has been successfully launched.

1. Navigate to the [EC2 Console](#).
2. Select your instance.
3. Open another browser and launch Driverless AI by navigating to <https://<public IP of the instance>:12345>.
4. Sign in to Driverless AI with the username and password provided in the **Usage Instructions**. You will be prompted to enter your Driverless AI license key the first time that you log in.

Stopping the EC2 Instance

The EC2 instance will continue to run even when you close the aws.amazon.com portal. To stop the instance:

1. On the EC2 Dashboard, click the **Running Instances** link under the Resources section.
2. Select the instance that you want to stop.
3. In the **Actions** drop down menu, select **Instance State > Stop**.
4. A confirmation page will display. Click **Yes, Stop** to stop the instance.

Upgrading the Driverless AI Marketplace Image

Note that the first offering of the Driverless AI Marketplace image was 1.5.2. As such, it is only possible to upgrade to versions greater than that.

Perform the following steps if you are upgrading to a Driverless AI Marketplace image version greater than 1.5.2. Replace `dai_NEVERVERSION.deb` below with the new Driverless AI version (for example, `dai_1.5.4_amd64.deb`). Note that this upgrade process inherits the service user and group from `/etc/dai/User.conf` and `/etc/dai/Group.conf`. You do not need to manually specify the `DAI_USER` or `DAI_GROUP` environment variables during an upgrade.

```
# Stop Driverless AI.  
sudo systemctl stop dai  
  
# Make a backup of /opt/h2oai/dai/tmp directory at this time.  
  
# Upgrade Driverless AI.  
sudo dpkg -i dai_NEVERVERSION.deb  
sudo systemctl daemon-reload  
sudo systemctl start dai
```

Install the Driverless AI AWS Community AMI

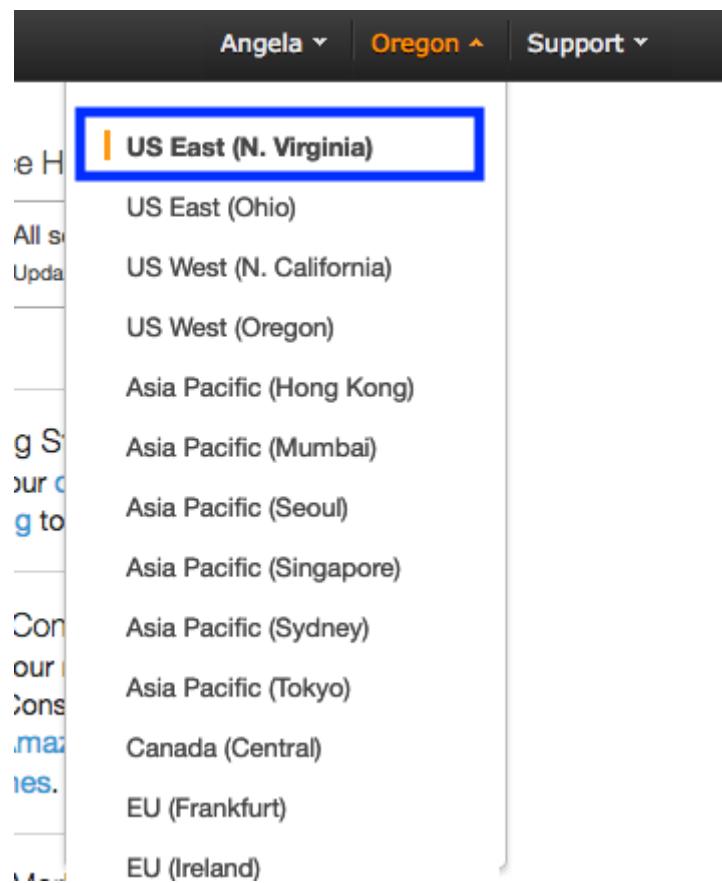
Watch the installation video [here](#). Note that some of the images in this video may change between releases, but the installation steps remain the same.

Environment

Provider	Instance Type	Num GPUs	Suitable for
AWS	p2.xlarge	1	Experimentation
	p2.8xlarge	8	Serious use
	p2.16xlarge	16	Serious use
	p3.2xlarge	1	Experimentation
	p3.8xlarge	4	Serious use
	p3.16xlarge	8	Serious use
	g3.4xlarge	1	Experimentation
	g3.8xlarge	2	Experimentation
	g3.16xlarge	4	Serious use

Installing the EC2 Instance

1. Log in to your AWS account at <https://aws.amazon.com>.
2. In the upper right corner of the Amazon Web Services page, set the location drop-down. (**Note:** We recommend selecting the US East region because H2O's resources are stored there. It also offers more instance types than other regions.)



3. Select the EC2 option under the Compute section to open the EC2 Dashboard.

AWS Management Console

AWS services

Find Services
You can enter names, keywords or acronyms.

▶ Recently visited services

▼ All services

<input checked="" type="checkbox"/> Compute <ul style="list-style-type: none"> <input checked="" type="checkbox"/> EC2 <input type="checkbox"/> Lightsail <input type="checkbox"/> ECR <input type="checkbox"/> ECS <input type="checkbox"/> EKS <input type="checkbox"/> Lambda <input type="checkbox"/> Batch 	<input type="checkbox"/> Management & Governance <ul style="list-style-type: none"> <input type="checkbox"/> AWS Organizations <input type="checkbox"/> CloudWatch <input type="checkbox"/> AWS Auto Scaling <input type="checkbox"/> CloudFormation <input type="checkbox"/> CloudTrail <input type="checkbox"/> Config <input type="checkbox"/> OpsWorks 	<input type="checkbox"/> Security, Identity, & Compliance <ul style="list-style-type: none"> <input type="checkbox"/> IAM <input type="checkbox"/> Resource Access Manager <input type="checkbox"/> Cognito <input type="checkbox"/> Secrets Manager <input type="checkbox"/> GuardDuty <input type="checkbox"/> Inspector <input type="checkbox"/> Amazon Macie
----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------	--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------	---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

4. Click the **Launch Instance** button under the Create Instance section.

The screenshot shows the AWS EC2 Dashboard. On the left, there's a sidebar with navigation links: EC2 Dashboard, Events, Tags, Reports, Limits, INSTANCES (with sub-links: Instances, Spot Requests, Reserved Instances, Scheduled Instances, Dedicated Hosts), IMAGES (with sub-links: AMIs, Bundle Tasks), and ELASTIC BLOCK STORE (with sub-link: Volumes). The main area is titled 'Resources' and displays the following statistics for the US East (N. Virginia) region:

- 0 Running Instances
- 0 Dedicated Hosts
- 0 Volumes
- 1 Key Pairs
- 0 Placement Groups
- 0 Elastic IPs
- 0 Snapshots
- 0 Load Balancers
- 2 Security Groups

Below this, a callout box says: "Just need a simple virtual private server? Get everything you need to jumpstart your project - compute, storage, and networking – for a low, predictable price. Try Amazon Lightsail for free." A large blue button labeled "Launch Instance" is centered in the "Create Instance" section. A note at the bottom states: "Note: Your instances will launch in the US East (N. Virginia) region".

- Under Community AMIs, search for **h2oai**, and then select the version that you want to launch.

The screenshot shows the "Choose an Amazon Machine Image (AMI)" step of the instance creation process. At the top, there are tabs: 1. Choose AMI, 2. Choose Instance Type, 3. Configure Instance, 4. Add Storage, 5. Add Tags, 6. Configure Security Group, and 7. Review. A "Cancel and Exit" button is on the right. Below the tabs, it says "Step 1: Choose an Amazon Machine Image (AMI)". A note states: "An AMI is a template that contains the software configuration (operating system, application server, and applications) required to launch your instance. You can select an AMI provided by AWS, our user community, or the AWS Marketplace; or you can select one of your own AMIs." A search bar contains the text "h2oai". The results list shows 16 AMIs found, with the first few listed:

AMI Name	AMI ID	Type	Action
h2oai-driverless-ai-1.5.0	ami-019f3fee8d51d71e4	64-bit (x86)	Select
h2oai-driverless-ai-1.5.0	ami-03d51c054e29b3a73	64-bit (x86)	Select
h2oai-driverless-ai-1.3.0	ami-0665b50182fbf1602	64-bit (x86)	Select
h2oai-driverless-ai-1.5.2	ami-066d51f8e1fb680ee	64-bit (x86)	Select
h2oai-driverless-ai-1.3.1	ami-07db97447bd5ed051	64-bit (x86)	Select

On the left, there's a sidebar with filters: Quick Start (0), My AMIs (0), AWS Marketplace (2), and Community AMIs (16). Under "Community AMIs", there are sections for Operating system (Amazon Linux, Cent OS, Debian, Fedora, Gentoo, openSUSE, Other Linux, Red Hat, SUSE Linux, Ubuntu, Windows) and Architecture.

- On the Choose an Instance Type page, select **GPU compute** in the **Filter by** dropdown. This will ensure that your Driverless AI instance will run on GPUs. Select a GPU compute instance from the available options. (We recommend at least 32 vCPUs.) Click the **Next: Configure Instance Details** button.

Step 2: Choose an Instance Type

Amazon EC2 provides a wide selection of instance types optimized to fit different use cases. Instances are virtual servers that can run applications. They have varying combinations of CPU, memory, storage, and networking capacity, and give you the flexibility to choose the appropriate mix of resources for your applications. [Learn more](#) about instance types and how they can meet your computing needs.

	Family	Type	vCPUs	Memory (GiB)	Instance Storage (GB)	EBS-Optimized Available	Network Performance	IPv6 Support
<input type="checkbox"/>	GPU instances	g2.2xlarge	8	15	1 x 60 (SSD)	Yes	Moderate	-
<input type="checkbox"/>	GPU instances	g2.8xlarge	32	60	2 x 120 (SSD)	-	High	-
<input type="checkbox"/>	GPU instances	g3s.xlarge	4	30.5	EBS only	Yes	Up to 10 Gigabit	Yes
<input type="checkbox"/>	GPU instances	g3.4xlarge	16	122	EBS only	Yes	Up to 10 Gigabit	Yes
<input checked="" type="checkbox"/>	GPU instances	g3.8xlarge	32	244	EBS only	Yes	10 Gigabit	Yes
<input type="checkbox"/>	GPU instances	g3.16xlarge	64	488	EBS only	Yes	25 Gigabit	Yes
<input type="checkbox"/>	GPU instances	p2.xlarge	4	61	EBS only	Yes	High	Yes
<input type="checkbox"/>	GPU instances	p2.8xlarge	32	488	EBS only	Yes	10 Gigabit	Yes
<input type="checkbox"/>	GPU instances	p2.16xlarge	64	732	EBS only	Yes	25 Gigabit	Yes
<input type="checkbox"/>	GPU instances	p3.2xlarge	8	61	EBS only	Yes	Up to 10 Gigabit	Yes

[Cancel](#) [Previous](#) [Review and Launch](#) [Next: Configure Instance Details](#)

- Specify the Instance Details that you want to configure. Create a VPC or use an existing one, and ensure that “Auto-Assign Public IP” is enabled and associated to your subnet. Click **Next: Add Storage**.

1. Choose AMI 2. Choose Instance Type 3. Configure Instance 4. Add Storage 5. Add Tags 6. Configure Security Group 7. Review

Step 3: Configure Instance Details
Configure the instance to suit your requirements. You can launch multiple instances from the same AMI, request Spot instances to take advantage of the lower pricing, assign an access management role to the instance, and more.

Number of instances	<input type="text" value="1"/> Launch into Auto Scaling Group
Purchasing option	<input type="checkbox"/> Request Spot instances
Network	vpc-e617b39c (default) Create new VPC
Subnet	subnet-0d900623 Default in us-east-1a Create new subnet
Auto-assign Public IP	Use subnet setting (Enable)
Placement group	<input type="checkbox"/> Add instance to placement group
Capacity Reservation	Open Create new Capacity Reservation
IAM role	None Create new IAM role
CPU options	<input type="checkbox"/> Specify CPU options
Shutdown behavior	Stop
Enable termination protection	<input type="checkbox"/> Protect against accidental termination

[Cancel](#) [Previous](#) [Review and Launch](#) [Next: Add Storage](#)

- Specify the Storage Device settings. Note again that Driverless AI requires 10 GB to run and will stop working if less than 10 GB is available. The machine should have a minimum of 30 GB of disk space. Click **Next: Add Tags**.

Using Driverless AI, Release 1.9.0.1

1. Choose AMI 2. Choose Instance Type 3. Configure Instance 4. Add Storage 5. Add Tags 6. Configure Security Group 7. Review

Step 4: Add Storage

Your instance will be launched with the following storage device settings. You can attach additional EBS volumes and instance store volumes to your instance, or edit the settings of the root volume. You can also attach additional EBS volumes after launching an instance, but not instance store volumes. [Learn more](#) about storage options in Amazon EC2.

Volume Type	Device	Snapshot	Size (GiB)	Volume Type	IOPS	Throughput (MB/s)	Delete on Termination	Encrypted
Root	/dev/sda1	snap-06b2151a8b2680d84	128	General Purpose SSD (GP2)	364 / 3000	N/A	<input checked="" type="checkbox"/>	Not Encrypted

Add New Volume

Free tier eligible customers can get up to 30 GB of EBS General Purpose (SSD) or Magnetic storage. [Learn more](#) about free usage tier eligibility and usage restrictions.

9. If desired, add unique Tag name to identify your instance. Click **Next: Configure Security Group**.
10. Add the following security rules to enable SSH access to Driverless AI, then click **Review and Launch**.

Type	Protocol	Port Range	Source	Description
SSH	TCP	22	Anywhere 0.0.0.0/0	
Custom TCP Rule	TCP	12345	Anywhere 0.0.0.0/0	Launch DAI

1. Choose AMI 2. Choose Instance Type 3. Configure Instance 4. Add Storage 5. Add Tags 6. Configure Security Group 7. Review

Step 6: Configure Security Group

A security group is a set of firewall rules that control the traffic for your instance. On this page, you can add rules to allow specific traffic to reach your instance. For example, if you want to set up a web server and allow Internet traffic to reach your instance, add rules that allow unrestricted access to the HTTP and HTTPS ports. You can create a new security group or select from an existing one below. [Learn more](#) about Amazon EC2 security groups.

Assign a security group: Create a new security group
 Select an existing security group

Security group name: launch-wizard-1
Description: launch-wizard-1 created 2019-04-30T10:57:32.006-07:00

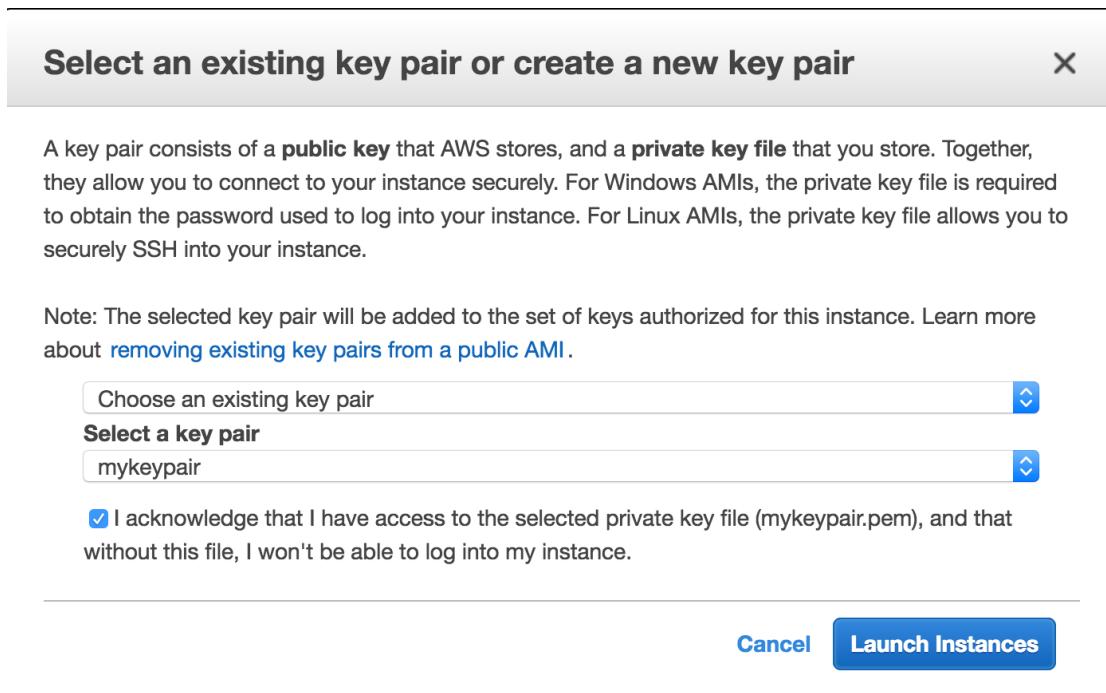
Type	Protocol	Port Range	Source	Description
SSH	TCP	22	Anywhere 0.0.0.0/0	e.g. SSH for Admin Desktop
Custom TCP	TCP	12345	Anywhere 0.0.0.0/0	Launch DAI

Add Rule

⚠ Warning
Rules with source of 0.0.0.0/0 allow all IP addresses to access your instance. We recommend setting security group rules to allow access from known IP addresses only.

Cancel **Previous** **Review and Launch**

11. Review the configuration, and then click **Launch**.
12. A popup will appear prompting you to select a key pair. This is required in order to SSH into the instance. You can select your existing key pair or create a new one. Be sure to accept the acknowledgement, then click **Launch Instances** to start the new instance.



13. Upon successful completion, a message will display informing you that your instance is launching. Click the **View Instances** button to see information about the instance including the IP address. The **Connect** button on this page provides information on how to SSH into your instance.
14. Open a Terminal window and SSH into the IP address of the AWS instance. Replace the DNS name below with your instance DNS.

```
ssh -i "mykeypair.pem" ubuntu@ec2-34-230-6-230.compute-1.amazonaws.com
```

Note: If you receive a “Permissions 0644 for ‘mykeypair.pem’ are too open” error, run the following command to give the user read permission and remove the other permissions.

```
chmod 400 mykeypair.pem
```

15. If you selected a GPU-compute instance, then you must enable persistence and optimizations of the GPU. The commands vary depending on the instance type. Note also that these commands need to be run once every reboot. Refer to the following for more information:

- <http://docs.nvidia.com/deploy/driver-persistence/index.html>
- https://docs.aws.amazon.com/AWSEC2/latest/WindowsGuide/optimize_gpu.html
- <https://www.migenius.com/articles/realityserver-on-aws>

```
# g3:
sudo nvidia-persistenced --persistence-mode
sudo nvidia-smi -acp 0
sudo nvidia-smi --auto-boost-permission=0
sudo nvidia-smi --auto-boost-default=0
sudo nvidia-smi -ac "2505,1177"

# p2:
sudo nvidia-persistenced --persistence-mode
sudo nvidia-smi -acp 0
```

(continues on next page)

(continued from previous page)

```
sudo nvidia-smi --auto-boost-permission=0  
sudo nvidia-smi --auto-boost-default=0  
sudo nvidia-smi -ac "2505,875"  
  
# p3:  
sudo nvidia-persistenced --persistence-mode  
sudo nvidia-smi -acp 0  
sudo nvidia-smi -ac "877,1530"
```

16. At this point, you can copy data into the data directory on the host machine using `scp`. For example:

```
scp -i /path/mykeypair.pem ubuntu@ec2-34-230-6-230.compute-1.amazonaws.com:/  
  ↪path/to/file/to/be/copied/example.csv /path/of/destination/on/local/machine
```

where:

- `i` is the identify file option
- `mykeypair` is the name of the private keypair file
- `ubuntu` is the name of the private keypair file
- `ec2-34-230-6-230.compute-1.amazonaws.com` is the public DNS name of the instance
- `example.csv` is the file to transfer

17. Connect to Driverless AI with your browser. You will be prompted to enter your Driverless AI license key the first time that you log in.

```
http://Your-Driverless-AI-Host-Machine:12345
```

Stopping the EC2 Instance

The EC2 instance will continue to run even when you close the `aws.amazon.com` portal. To stop the instance:

1. On the EC2 Dashboard, click the **Running Instances** link under the Resources section.
2. Select the instance that you want to stop.
3. In the **Actions** drop down menu, select **Instance State > Stop**.
4. A confirmation page will display. Click **Yes, Stop** to stop the instance.

Upgrading the Driverless AI Community Image

WARNINGS:

- This release deprecates experiments and MLIs models from 1.7.0 and earlier.
- Experiments, MLIs, and MOJOs reside in the Driverless AI tmp directory and are not automatically upgraded when Driverless AI is upgraded. We recommend you take the following steps before upgrading.
 - Build MLIs models before upgrading.
 - Build MOJO pipelines before upgrading.
 - Stop Driverless AI and make a backup of your Driverless AI tmp directory before upgrading.

If you did not build MLIs on a model before upgrading Driverless AI, then you will not be able to view MLIs on that model after upgrading. Before upgrading, be sure to run MLIs on models that you want to continue to interpret in future releases. If that MLI job appears in the list of Interpreted Models in your current version, then it will be retained after upgrading.

If you did not build a MOJO pipeline on a model before upgrading Driverless AI, then you will not be able to build a MOJO pipeline on that model after upgrading. Before upgrading, be sure to build MOJO pipelines on all desired models and then back up your Driverless AI tmp directory.

The upgrade process inherits the service user and group from /etc/dai/User.conf and /etc/dai/Group.conf. You do not need to manually specify the DAI_USER or DAI_GROUP environment variables during an upgrade.

Upgrading from Version 1.2.2 or Earlier

The following example shows how to upgrade from 1.2.2 or earlier to the current version. Upgrading from these earlier versions requires an edit to the start and h2oai scripts.

1. SSH into the IP address of the image instance and copy the existing experiments to a backup location:

```
# Set up a directory of the previous version name
mkdir dai_rel_1.2.2

# Copy the data, log, license, and tmp directories as backup
cp -a ./data dai_rel_1.2.2/data
cp -a ./log dai_rel_1.2.2/log
cp -a ./license dai_rel_1.2.2/license
cp -a ./tmp dai_rel_1.2.2/tmp
```

2. wget the newer image. The command below retrieves version 1.2.2:

```
wget https://s3.amazonaws.com/artifacts.h2o.ai/releases/ai/h2o/dai/rel-1.2.2-
˓→6/x86_64-centos7/dai-docker-centos7-x86_64-1.2.2-9.0.tar.gz
```

3. In the /home/ubuntu/scripts/ folder, edit both the start.sh and h2oai.sh scripts to use the newer image.

4. Use the docker load command to load the image:

```
docker load < ami-0c50db5e1999408a7
```

5. Optionally run docker images to ensure that the new image is in the registry.

6. Connect to Driverless AI with your browser at <http://Your-Driverless-AI-Host-Machine:12345>.

Upgrading from Version 1.3.0 or Later

The following example shows how to upgrade from version 1.3.0.

1. SSH into the IP address of the image instance and copy the existing experiments to a backup location:

```
# Set up a directory of the previous version name
mkdir dai_rel_1.3.0

# Copy the data, log, license, and tmp directories as backup
cp -a ./data dai_rel_1.3.0/data
cp -a ./log dai_rel_1.3.0/log
cp -a ./license dai_rel_1.3.0/license
cp -a ./tmp dai_rel_1.3.0/tmp
```

2. wget the newer image. Replace VERSION and BUILD below with the Driverless AI version.

```
wget https://s3.amazonaws.com/artifacts.h2o.ai/releases/ai/h2o/dai/VERSION-  
BUILD/x86_64-centos7/dai-docker-centos7-x86_64-VERSION.tar.gz
```

3. Use the docker load command to load the image:

```
docker load < dai-docker-centos7-x86_64-VERSION.tar.gz
```

4. In the new AMI, locate the DAI_RELEASE file, and edit that file to match the new image tag.

5. Stop and then start Driverless AI.

```
h2oai stop  
h2oai start
```

When installing via AWS, you can also enable role-based authentication.

AWS Role-Based Authentication

In Driverless AI, it is possible to enable role-based authentication via the [IAM role](#). This is a two-step process that involves setting up AWS IAM and then starting Driverless AI by specifying the role in the config.toml file or by setting the AWS_USE_EC2_ROLE_CREDENTIALS environment variable to True.

AWS IAM Setup

1. Create an IAM role. This IAM role should have a Trust Relationship with Principal Trust Entity set to your Account ID. For example: trust relationship for Account ID 524466471676 would look like:

```
{  
    "Version": "2012-10-17",  
    "Statement": [  
        {  
            "Effect": "Allow",  
            "Principal": {  
                "AWS": "arn:aws:iam::524466471676:root"  
            },  
            "Action": "sts:AssumeRole"  
        }  
    ]  
}
```

Role ARN: arn:aws:iam::524466471676:role/testingRoles

Role description: Allows EC2 instances to call AWS services on your behalf. [Edit](#)

Instance Profile ARNs: arn:aws:iam::524466471676:instance-profile/testingRoles

Path: /

Creation time: 2018-10-08 07:55 PDT

Maximum CLI/API session duration: 1 hour [Edit](#)

Give this link to users who can switch roles in the console: <https://signin.aws.amazon.com/switchrole?roleName=testingRoles&account=0xdata>

Permissions **Trust relationships** **Access Advisor** **Revoke sessions**

You can view the trusted entities that can assume the role and the access conditions for the role. [Show policy document](#)

Edit trust relationship

Trusted entities
The following trusted entities can assume this role.

Conditions
The following conditions define how and when trusted entities can assume the role.

Trusted entities
The account 524466471676
There are no conditions associated with this role.

2. Create a new policy that allows users to assume the role:

Policy ARN: arn:aws:iam::524466471676:policy/testRolesPolicy

Description: testing AssumeRole functionality

Permissions **Policy usage** **Policy versions** **Access Advisor**

Policy summary [JSON](#) [Edit policy](#)

```

1  {
2   "Version": "2012-10-17",
3   "Statement": [
4     {
5       "Sid": "VisualEditor0",
6       "Effect": "Allow",
7       "Action": [
8         "sts:DecodeAuthorizationMessage",
9         "sts:GetCallerIdentity"
10      ],
11      "Resource": "*"
12    },
13    {
14      "Sid": "VisualEditor1",
15      "Effect": "Allow",
16      "Action": "sts:*",
17      "Resource": "arn:aws:iam::524466471676:role/testingRoles"
18    }
19  ]
  
```

3. Assign the policy to the user.

The screenshot shows the AWS IAM User Summary page for a user named 'dauren'. The left sidebar has a 'Users' tab selected. The main summary section displays the User ARN (arn:aws:iam::524466471676:user/dauren), Path (/), and Creation time (2018-01-09 13:07 PDT). Below this, there are tabs for 'Permissions', 'Groups (1)', 'Security credentials', and 'Access Advisor'. The 'Permissions' tab is active, showing a list of applied policies: 'IAMFullAccess' (AWS managed policy) and 'testRolesPolicy' (Managed policy). There are buttons for 'Add permissions' and 'Add inline policy'. A link to 'Show 13 more' policies is also present. The 'Encryption keys' section is visible on the left.

4. Test role switching here: <https://signin.aws.amazon.com/switchrole>. (Refer to https://docs.aws.amazon.com/IAM/latest/UserGuide/troubleshoot_roles.html#troubleshoot_roles_cant-assume-role.)

Driverless AI Setup

Update the `aws_use_ec2_role_credentials` config variable in the `config.toml` file or start Driverless AI using the `AWS_USE_EC2_ROLE_CREDENTIALS` environment variable.

Resources

1. Granting a User Permissions to Switch Roles: https://docs.aws.amazon.com/IAM/latest/UserGuide/id_roles_use_permissions-to-switch.html
2. Creating a Role to Delegate Permissions to an IAM User: https://docs.aws.amazon.com/IAM/latest/UserGuide/id_roles_create_for-user.html
3. Assuming an IAM Role in the AWS CLI: <https://docs.aws.amazon.com/cli/latest/userguide/cli-configure-role.html>

Install on Azure

This section describes how to install the Driverless AI image from Azure.

Note: Prior versions of the Driverless AI installation and upgrade on Azure were done via Docker. This is no longer the case as of version 1.5.2.

Watch the installation video [here](#). Note that some of the images in this video may change between releases, but the installation steps remain the same.

Environment

Provider	Instance Type	Num GPUs	Suitable for
Azure	Standard_NV6	1	Experimentation
	Standard_NV12	2	Experimentation
	Standard_NV24	4	Serious use
	Standard_NC6	1	Experimentation
	Standard_NC12	2	Experimentation
	Standard_NC24	4	Serious use

About the Install

- The ‘dai’ service user is created locally (in /etc/passwd) if it is not found by ‘getent passwd’. You can override the user by providing the DAI_USER environment variable during rpm or dpkg installation.
- The ‘dai’ service group is created locally (in /etc/group) if it is not found by ‘getent group’. You can override the group by providing the DAI_GROUP environment variable during rpm or dpkg installation.
- Configuration files are put in **/etc/dai** and owned by the ‘root’ user:
 - **/etc/dai/config.toml**: Driverless AI config file (See [Using the config.toml File](#) section for details)
 - **/etc/dai/User.conf**: Systemd config file specifying the service user
 - **/etc/dai/Group.conf**: Systemd config file specifying the service group
 - **/etc/dai/EnvironmentFile.conf**: Systemd config file specifying (optional) environment variable overrides
- Software files are put in **/opt/h2oai/dai** and owned by the ‘root’ user
- The following directories are owned by the service user so they can be updated by the running software:
 - **/opt/h2oai/dai/home**: The application’s home directory (license key files are stored here)
 - **/opt/h2oai/dai/tmp**: Experiments and imported data are stored here
 - **/opt/h2oai/dai/log**: Log files go here if you are **not** using systemd (if you are using systemd, then the use the standard journalctl tool)
- By default, Driverless AI looks for a license key in **/opt/h2oai/dai/home/.driverlessai/license.sig**. If you are installing Driverless AI programmatically, you can copy a license key file to that location. If no license key is found, the application will interactively guide you to add one from the Web UI.
- systemd unit files are put in **/usr/lib/systemd/system**
- Symbolic links to the configuration files in /etc/dai files are put in **/etc/systemd/system**

If your environment is running an operational systemd, that is the preferred way to manage Driverless AI. The package installs the following systemd services and a wrapper service:

- **dai**: Wrapper service that starts/stops the other three services
- **dai-dai**: Main Driverless AI process
- **dai-h2o**: H2O-3 helper process used by Driverless AI
- **dai-procsy**: Procsy helper process used by Driverless AI
- **dai-vis-server**: Visualization server helper process used by Driverless AI

If you don’t have systemd, you can also use the provided run script to start Driverless AI.

Installing the Azure Instance

1. Log in to your Azure portal at <https://portal.azure.com>, and click the **Create a Resource** button.
2. Search for and select **H2O DriverlessAI** in the Marketplace.

The screenshot shows the Azure Marketplace search results for "H2O Driverless AI". The search bar at the top contains the query. Below it, a table lists the found item:

NAME	PUBLISHER	CATEGORY
H2O.ai H2O Driverless AI (BYOL)	H2O.ai	Compute

3. Click **Create**. This launches the H2O DriverlessAI Virtual Machine creation process.

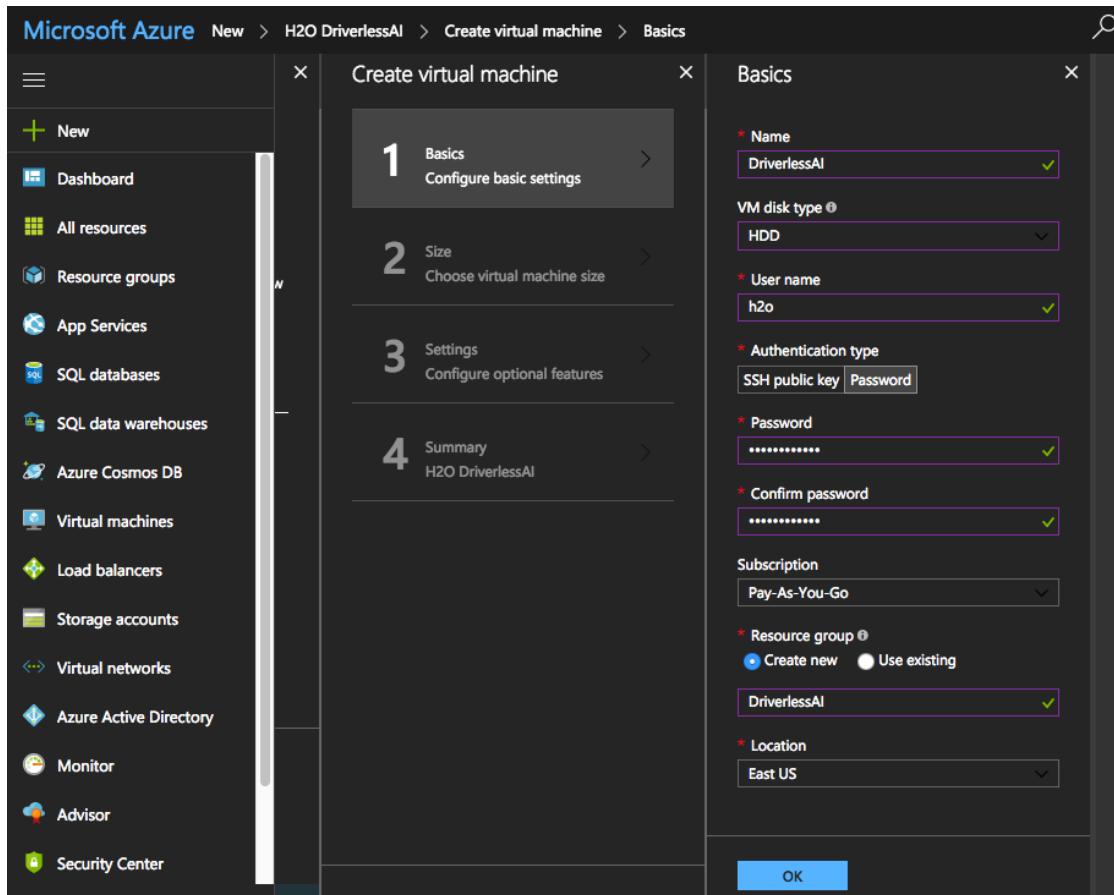
The screenshot shows the Microsoft Azure "Create" dialog for the H2O DriverlessAI resource. The left sidebar shows various Azure services like Dashboard, All resources, Resource groups, etc. The main pane displays the following details for the H2O DriverlessAI resource:

- H2O.ai H2O DriverlessAI**
- Description:** H2O Driverless AI is an artificial intelligence (AI) platform that automates some of the most difficult data science and machine learning workflows such as feature engineering, model validation, model tuning, model selection and model deployment. It aims to achieve highest predictive accuracy, comparable to expert data scientists, but in much shorter time thanks to end-to-end automation. Driverless AI also offers automatic visualizations and machine learning interpretability (MLI). Especially in regulated industries, model transparency and explanation are just as important as predictive performance.
- Publisher:** H2O.ai
- Useful Links:** H2O Driverless AI, User Guide
- Support:** <https://support.h2o.ai>
- Select a deployment model:** Resource Manager (dropdown menu)
- Create** button

4. On the **Basics** tab:
 - a. Enter a name for the VM.
 - b. Select the Disk Type for the VM. Use HDD for GPU instances.
 - c. Enter the name that you will use when connecting to the machine through SSH.

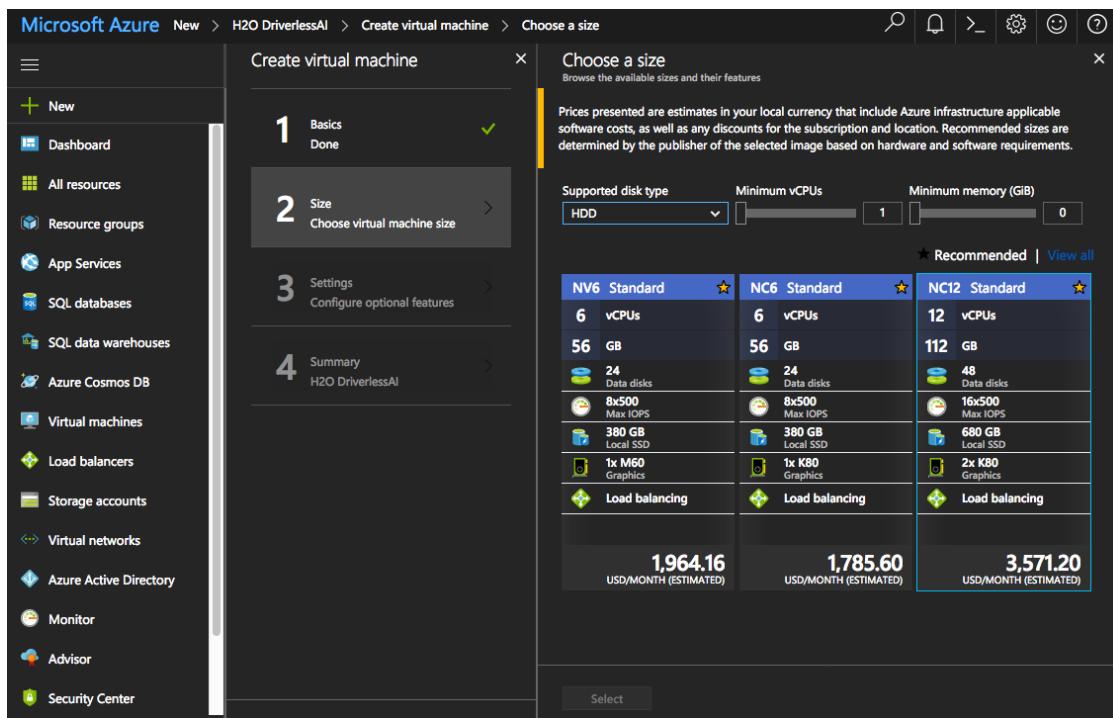
- d. Enter and confirm a password that will be used when connecting to the machine through SSH.
- e. Specify the Subscription option. (This should be Pay-As-You-Go.)
- f. Enter a name unique name for the resource group.
- g. Specify the VM region.

Click **OK** when you are done.

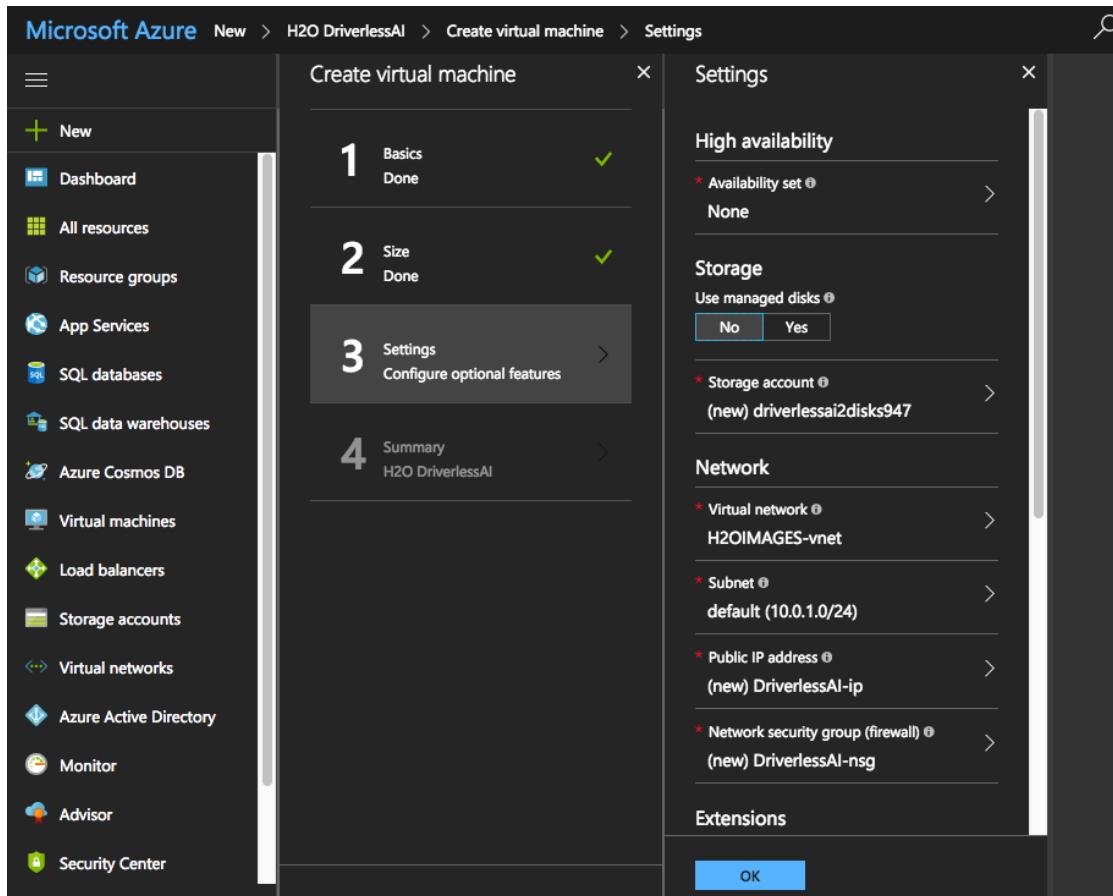


5. On the **Size** tab, select your virtual machine size. Specify the HDD disk type and select a configuration. We recommend using an N-Series type, which comes with a GPU. Also note that Driverless AI requires 10 GB of free space in order to run and will stop working if less than 10 GB is available. We recommend a minimum of 30 GB of disk space. Click **OK** when you are done.

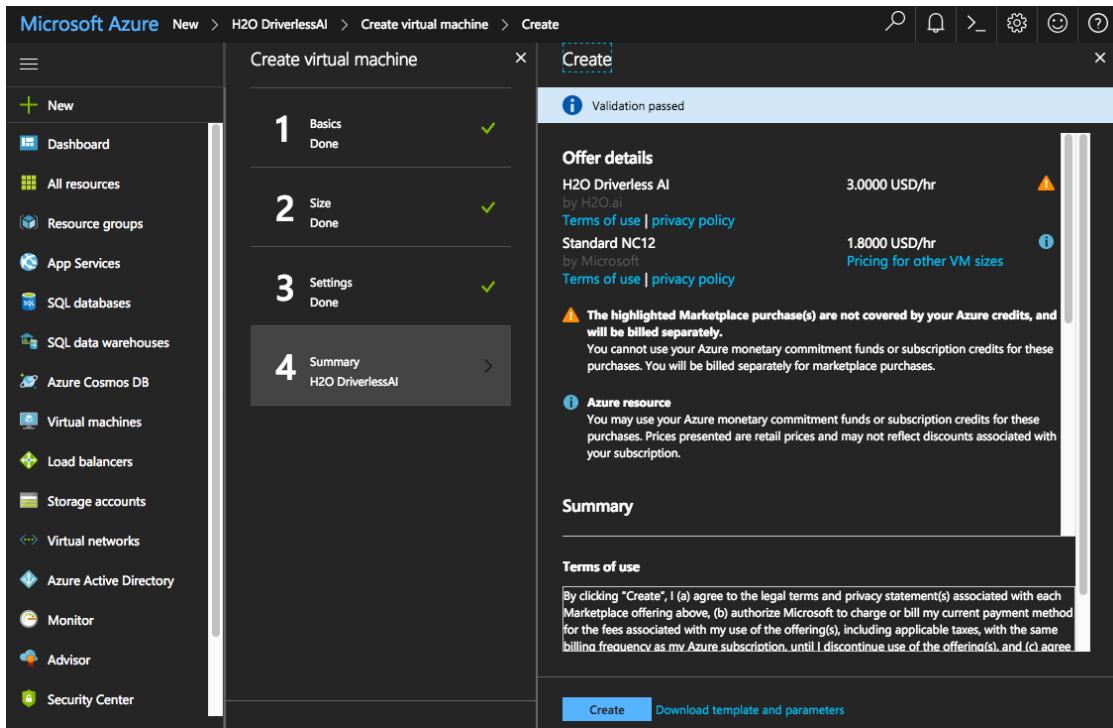
Using Driverless AI, Release 1.9.0.1



6. On the **Settings** tab, select or create the Virtual Network and Subnet where the VM is going to be located and then click **OK**.



7. The **Summary** tab performs a validation on the specified settings and will report back any errors. When the validation passes successfully, click **Create** to create the VM.



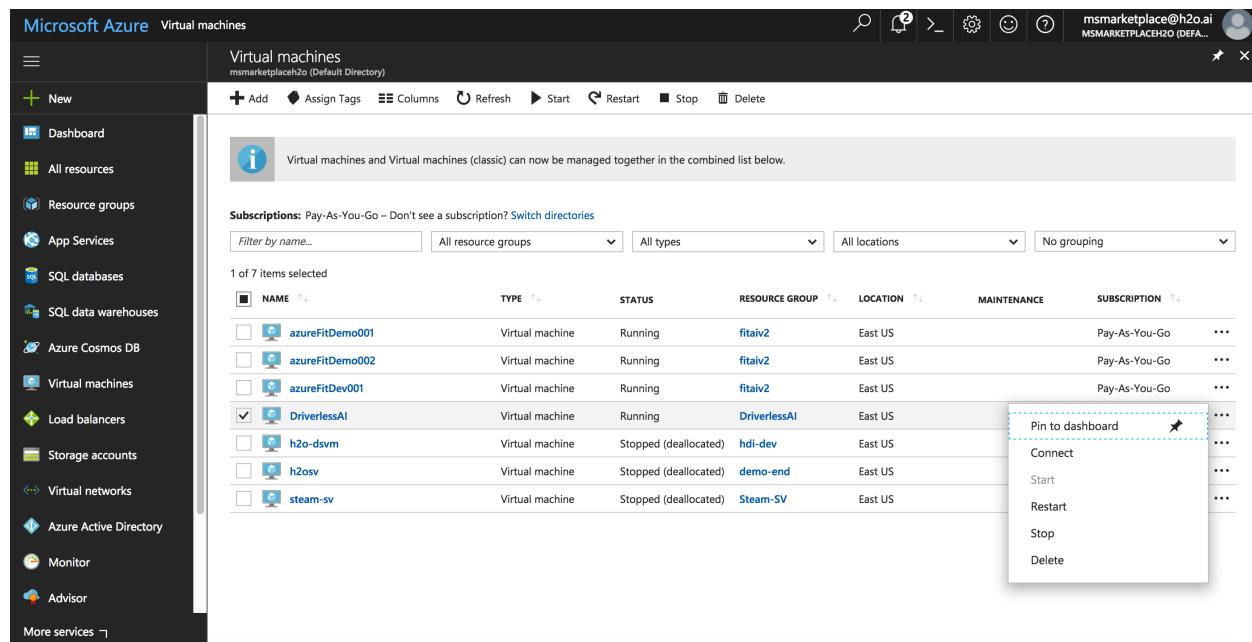
8. After the VM is created, it will be available under the list of Virtual Machines. Select this Driverless AI VM to view the IP address of your newly created machine.
 9. Connect to Driverless AI with your browser using the IP address retrieved in the previous step.

`http://Your-Driverless-AI-Host-Machine:12345`

Stopping the Azure Instance

The Azure instance will continue to run even when you close the Azure portal. To stop the instance:

1. Click the **Virtual Machines** left menu item.
2. Select the checkbox beside your DriverlessAI virtual machine.
3. On the right side of the row, click the ... button, then select **Stop**. (Note that you can then restart this by selecting **Start**.)



Upgrading the Driverless AI Image

WARNINGS:

- This release deprecates experiments and MLi models from 1.7.0 and earlier.
- Experiments, MLIs, and MOJOs reside in the Driverless AI tmp directory and are not automatically upgraded when Driverless AI is upgraded. We recommend you take the following steps before upgrading.
 - Build MLi models before upgrading.
 - Build MOJO pipelines before upgrading.
 - Stop Driverless AI and make a backup of your Driverless AI tmp directory before upgrading.

If you did not build MLi on a model before upgrading Driverless AI, then you will not be able to view MLi on that model after upgrading. Before upgrading, be sure to run MLi jobs on models that you want to continue to interpret in future releases. If that MLi job appears in the list of Interpreted Models in your current version, then it will be retained after upgrading.

If you did not build a MOJO pipeline on a model before upgrading Driverless AI, then you will not be able to build a MOJO pipeline on that model after upgrading. Before upgrading, be sure to build MOJO pipelines on all desired models and then back up your Driverless AI tmp directory.

The upgrade process inherits the service user and group from /etc/dai/User.conf and /etc/dai/Group.conf. You do not need to manually specify the DAI_USER or DAI_GROUP environment variables during an upgrade.

Upgrading from Version 1.2.2 or Earlier

It is not possible to upgrade from version 1.2.2 or earlier to the latest version. You have to manually remove the 1.2.2 container and then reinstall the latest Driverless AI version. Be sure to backup your data before doing this.

Upgrading from Version 1.3.0 to 1.5.1

1. SSH into the IP address of the image instance and copy the existing experiments to a backup location:

```
# Set up a directory of the previous version name
mkdir dai_rel_1.3.0

# Copy the data, log, license, and tmp directories as backup
cp -a ./data dai_rel_1.3.0/data
cp -a ./log dai_rel_1.3.0/log
cp -a ./license dai_rel_1.3.0/license
cp -a ./tmp dai_rel_1.3.0/tmp
```

2. wget the newer image. Replace VERSION and BUILD below with the Driverless AI version.

```
wget https://s3.amazonaws.com/artifacts.h2o.ai/releases/ai/h2o/dai/VERSION-
→BUILD/x86_64-centos7/dai-docker-centos7-x86_64-VERSION.tar.gz
```

3. Use the docker load command to load the image:

```
docker load < dai-docker-centos7-x86_64-VERSION.tar.gz
```

4. Run docker images to find the new image tag.

5. Start the Driverless AI Docker image and replace TAG below with the image tag. Depending on your install version, use the docker run --runtime=nvidia (>= Docker 19.03) or nvidia-docker (< Docker 19.03) command.

Note: Use docker version to check which version of Docker you are using.

>= Docker 19.03

< Docker 19.03

```
# Start the Driverless AI Docker image
docker run --runtime=nvidia \
    --pid=host \
    --init \
    --rm \
    --shm-size=256m \
    -u `id -u`:`id -g` \
    -p 12345:12345 \
    -v `pwd`/data:/data \
    -v `pwd`/log:/log \
    -v `pwd`/license:/license \
    -v `pwd`/tmp:/tmp \
    h2oai/dai-centos7-x86_64:TAG
```

```
# Start the Driverless AI Docker image
nvidia-docker run \
    --pid=host \
    --init \
```

(continues on next page)

(continued from previous page)

```
--rm \
--shm-size=256m \
-u `id -u`:`id -g` \
-p 12345:12345 \
-v `pwd`/data:/data \
-v `pwd`/log:/log \
-v `pwd`/license:/license \
-v `pwd`/tmp:/tmp \
h2oai/dai-centos7-x86_64:TAG
```

Upgrading from version 1.5.2 or Later

Upgrading to versions 1.5.2 and later is no longer done via Docker. Instead, perform the following steps if you are upgrading to version 1.5.2 or later. Replace `dai_NEVERSION.deb` below with the new Driverless AI version (for example, `dai_1.8.4.1_amd64.deb`). Note that this upgrade process inherits the service user and group from `/etc/dai/User.conf` and `/etc/dai/Group.conf`. You do not need to manually specify the `DAI_USER` or `DAI_GROUP` environment variables during an upgrade.

Note about upgrading to 1.7.x: As of 1.7.0, CUDA 9 is no longer supported. Your host environment must have CUDA 10.0 or later with NVIDIA drivers ≥ 440.82 installed (GPU only). Driverless AI ships with its own CUDA libraries, but the driver must exist in the host environment. Go to <https://www.nvidia.com/Download/index.aspx> to get the latest NVIDIA Tesla V/P/K series driver.

```
# Stop Driverless AI.
sudo systemctl stop dai

# Backup your /opt/h2oai/dai/tmp directory at this time.

# Upgrade Driverless AI.
sudo dpkg -i dai_NEVERSION.deb
sudo systemctl daemon-reload
sudo systemctl start dai
```

Install on Google Compute

Driverless AI can be installed on Google Compute using one of two methods:

- Install the Google Cloud Platform offering. This installs Driverless AI via the available GCP Marketplace offering.
- Install and Run in a Docker Container on Google Compute Engine. This installs and runs Driverless AI from scratch in a Docker container on Google Compute Engine.

Select your desired installation procedure below:

Install the Google Cloud Platform Offering

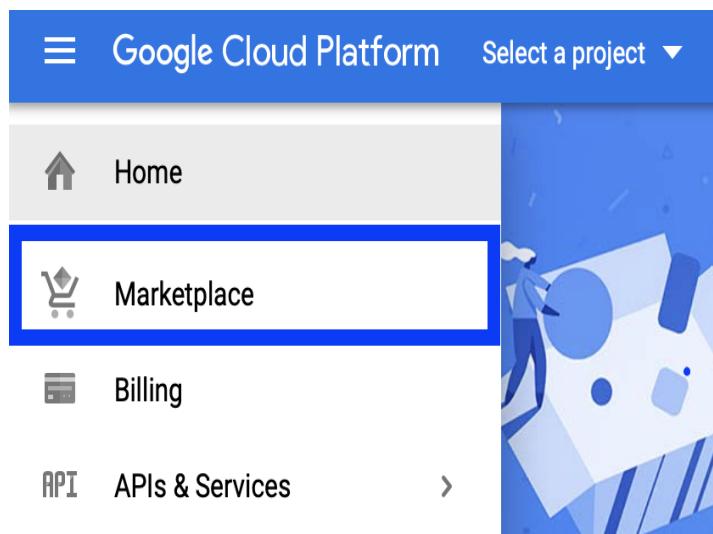
This section describes how to install and start Driverless AI in a Google Compute environment using the GCP Marketplace. This assumes that you already have a Google Cloud Platform account. If you don't have an account, go to <https://console.cloud.google.com/getting-started> to create one.

Before You Begin

If you are trying GCP for the first time and have just created an account, please check your Google Compute Engine (GCE) resource quota limits. By default, GCP allocates a maximum of 8 CPUs and no GPUs. Our default recommendation for launching Driverless AI is 32 CPUs, 120 GB RAM, and 2 P100 NVIDIA GPUs. You can change these settings to match your quota limit, or you can request more resources from GCP. Refer to <https://cloud.google.com/compute/quotas> for more information, including information on how to check your quota and request additional quota.

Installation Procedure

1. In your browser, log in to the Google Compute Engine Console at <https://console.cloud.google.com/>.
2. In the left navigation panel, select **Marketplace**.



3. On the Marketplace page, search for **Driverless** and select the H2O.ai Driverless AI offering. The following page will display.

The screenshot shows the Google Cloud Platform Marketplace interface. At the top, there's a blue header bar with the text "Google Cloud Platform" and "H2O-GCE". Below the header, the main content area displays the "H2O.ai Driverless AI" product. On the left, there's a circular icon containing the "H2O.ai" logo. To the right of the icon, the product name "H2O.ai Driverless AI" is displayed, along with the URL "H2O.ai" and the estimated cost "Estimated costs: \$2,279.08/month + BYOL license fee". A brief description follows: "Automatic Machine Learning for the Enterprise". A prominent blue button labeled "LAUNCH ON COMPUTE ENGINE" is centered below the description. On the left side of the main content area, there's a sidebar with various metadata fields:

Runs on	Google Compute Engine
Type	Single VM
BYOL	
Last updated	5/9/18, 3:55 PM
Category	Analytics
Version	1.1.0
Operating system	Ubuntu 16.04

Below the sidebar, there are three sections with links: "About H2O.ai", "About BYOL", and "Pricing". The "Pricing" section includes a note: "This is a BYOL solution which requires a valid license to use. You are" followed by two columns: "Item" and "Estimated costs".

4. Click **Launch on Compute Engine**. (If necessary, refer to [Google Compute Instance Types](#) for information about machine and GPU types.)

- Select a zone that has p100s or k80s (such as us-east1-)
- Optionally change the number of cores and amount of memory. (This defaults to 32 CPUs and 120 GB RAM.)
- Specify a GPU type. (This defaults to a p100 GPU.)
- Optionally change the number of GPUs. (Default is 2.)
- Specify the boot disk type and size.
- Optionally change the network name and subnetwork names. Be sure that whichever network you specify has port 12345 exposed.
- Click **Deploy** when you are done. Driverless AI will begin deploying. Note that this can take several minutes.

Machine type

- Cores: 32 vCPU (4 - 96)
- Memory: 120 GB (28.8 - 208)
- Extend memory:
- Choosing a machine type: [Learn more](#)

GPU

GPU Type: nvidia-tesla-p100

Number of GPUs: Instances with GPUs have specific restrictions that make them behave differently than other instance types. GPUs are only available in certain zones, and the number of GPU dies is linked to the number of CPU cores selected for this instance. [Learn more](#)

2

Boot Disk

Boot disk type: Standard Persistent Disk

Boot disk size in GB: 256

Networking

Network name: default

Subnetwork name: default

More

Deploy

Details

Software

Operating System: Ubuntu (16.04)

Launching a BYOL solution

H2O.ai Driverless AI is a BYOL (Bring Your Own License) solution. Cloud Launcher will deploy this solution, but you are responsible for purchasing and managing the license directly from the provider.

Terms of Service

The software or service you are about to use is not a Google product. By deploying the software or accessing the service you are agreeing to comply with the Google Cloud Launcher terms of service and the terms of any third party software licenses related to the software or service. Please review these licenses carefully for details about any obligations you may have related to the software or services. To the limited extent an open source software license related to the software or service expressly supersedes the Google Cloud Launcher Terms of Service, that open source software license governs your use of that software or service.

Google is providing this software or service "as-is" and will not perform any ongoing maintenance. Ongoing upgrades and maintenance are your responsibility.

5. A summary page displays when the compute engine is successfully deployed. This page includes the instance ID and the username (always h2oai) and password that will be required when starting Driverless AI. Click on the Instance link to retrieve the external IP address for starting Driverless AI.

The screenshot shows the H2O.ai Driverless AI deployment interface. On the left, there's a sidebar with icons for Overview, Instances, and Help. The main area shows a deployment window for 'h2oai-driverless-ai-byol-1'. A message box says 'h2oai-driverless-ai-byol-1 has been deployed'. Below it is a tree view of files: Overview - h2oai-driverless-ai-byol-1, h2oai-driverless-ai-byol h2oai-driverless-ai-byol.jinja, h2oai-driverless-ai-byol-vm-tmpl vm_instance.py, h2oai-driverless-ai-byol-1-vm vm instance, and h2oai-driverless-ai-byol-vm-tmpl-dai-fw firewall. To the right is the 'H2O.ai Driverless AI' card, which includes fields for Instance (h2oai-driverless-ai-byol-1-vm), Connect At (https://35.233.172.54:443), Username (h2oai), Password (redacted), Instance zone (us-west1-b), and Instance machine type (n1-standard-4). Below the card is a 'Get started with H2O.ai Driverless AI' section with an SSH dropdown and a 'Suggested next steps' section containing a 'NOTE' about SSL certificates and a 'Connect to Driverless AI' step. At the bottom is a terminal window with the command 'gcloud compute scp /path/to/config.toml ubuntu@<running->:service_account.json'.

6. In your browser, go to https://{{External_IP}}:12345 to start Driverless AI.
7. Agree to the Terms and Conditions.
8. Log in to Driverless AI using your user name and password.
9. Optionally enable GCS and Big Query access.
 - a. In order to enable GCS and Google BigQuery access, you must pass the running instance a service account json file configured with GCS and GBQ access. The Driverless AI image comes with a blank file called **service_account.json**. Obtain a functioning service account json file from [GCP](#), rename it to “service_account.json”, and copy it to the Ubuntu user on the running instance.

```
gcloud compute scp /path/to/service_account.json ubuntu@<running->:service_account.json
```

- b. SSH into the machine running Driverless AI, and verify that the service_account.json file is in the /etc/dai/ folder.
- c. Restart the machine for the changes to take effect.

```
sudo systemctl stop dai
# Wait for the system to stop
# Verify that the system is no longer running
sudo systemctl status dai
# Restart the system
sudo systemctl start dai
```

Upgrading the Google Cloud Platform Offering

Perform the following steps to upgrade the Driverless AI Google Platform offering. Replace `dai_NEWVERSION.deb` below with the new Driverless AI version (for example, `dai_1.6.1_amd64.deb`). Note that this upgrade process inherits the service user and group from `/etc/dai/User.conf` and `/etc/dai/Group.conf`. You do not need to manually specify the `DAI_USER` or `DAI_GROUP` environment variables during an upgrade.

```
# Stop Driverless AI.
sudo systemctl stop dai

# Make a backup of /opt/h2oai/dai/tmp directory at this time.

# Upgrade Driverless AI.
sudo dpkg -i dai_NEWVERSION.deb
sudo systemctl daemon-reload
sudo systemctl start dai
```

Install and Run in a Docker Container on Google Compute Engine

This section describes how to install and start Driverless AI **from scratch** using a Docker container in a Google Compute environment.

This installation assumes that you already have a Google Cloud Platform account. If you don't have an account, go to <https://console.cloud.google.com/getting-started> to create one. In addition, refer to Google's [Machine Types documentation](#) for information on Google Compute machine types.

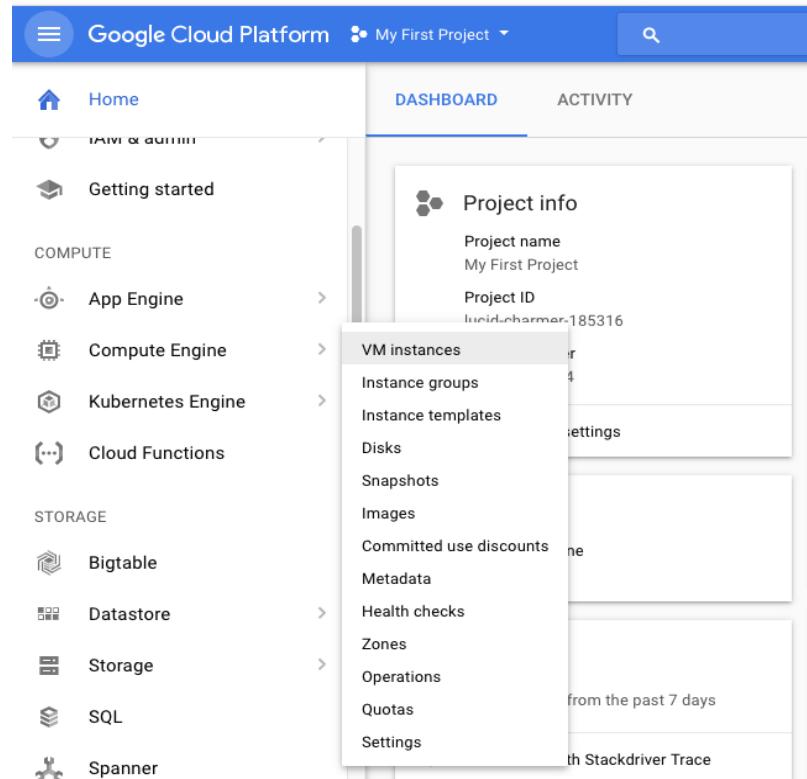
Watch the installation video [here](#). Note that some of the images in this video may change between releases, but the installation steps remain the same.

Before You Begin

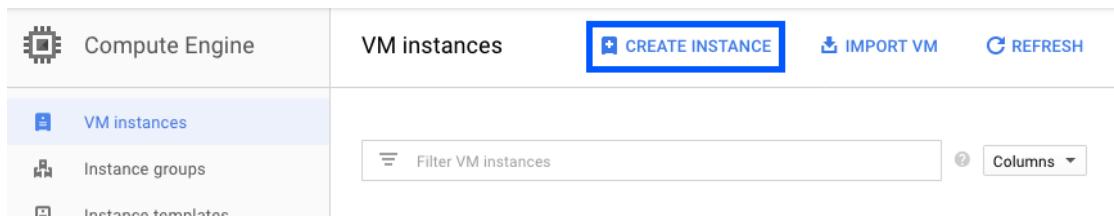
If you are trying GCP for the first time and have just created an account, please check your Google Compute Engine (GCE) resource quota limits. By default, GCP allocates a maximum of 8 CPUs and no GPUs. You can change these settings to match your quota limit, or you can request more resources from GCP. Refer to <https://cloud.google.com/compute/quotas> for more information, including information on how to check your quota and request additional quota.

Installation Procedure

1. In your browser, log in to the Google Compute Engine Console at <https://console.cloud.google.com/>.
2. In the left navigation panel, select **Compute Engine > VM Instances**.



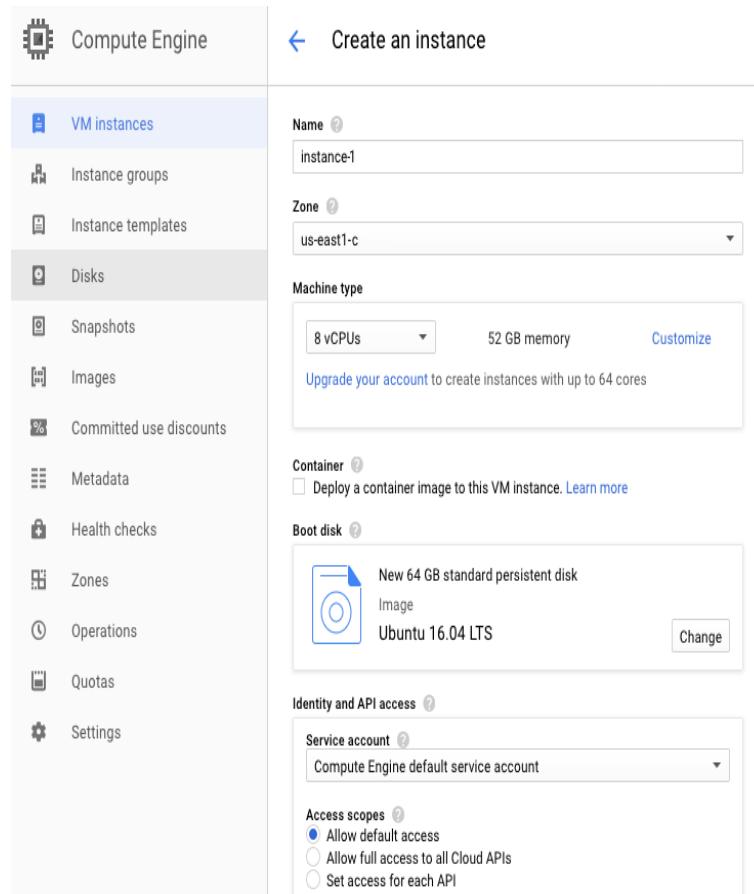
3. Click **Create Instance**.



4. Specify the following at a minimum:

- A unique name for this instance.
- The desired **zone**. Note that not all zones and user accounts can select zones with GPU instances. Refer to the following for information on how to add GPUs: <https://cloud.google.com/compute/docs/gpus/>.
- A supported OS, for example Ubuntu 16.04. Be sure to also increase the disk size of the OS image to be 64 GB.

Click **Create** at the bottom of the form when you are done. This creates the new VM instance.



5. Create a Firewall rule for Driverless AI. On the Google Cloud Platform left navigation panel, select **VPC network > Firewall rules**. Specify the following settings:
- Specify a unique name and Description for this instance.
 - Change the **Targets** dropdown to **All instances in the network**.
 - Specify the **Source IP ranges** to be **0 . 0 . 0 . 0 / 0**.
 - Under **Protocols and Ports**, select **Specified protocols and ports** and enter the following: `tcp:12345`.

Click **Create** at the bottom of the form when you are done.

Create a firewall rule

Description (Optional)
h2oai

Network
default

Priority
Priority can be 0 - 65535 [Check priority of other firewall rules](#)
1000

Direction of traffic
 Ingress
 Egress

Action on match
 Allow
 Deny

Targets
All instances in the network

Source filter
IP ranges

Source IP ranges
0.0.0.0/0

Second source filter
None

Protocols and ports
 Allow all
 Specified protocols and ports
tcp:12345

Create **Cancel**

Equivalent [REST](#) or [command line](#)

- On the VM Instances page, SSH to the new VM Instance by selecting **Open in Browser Window** from the SSH dropdown.

Name	Zone	Recommendation	Internal IP	External IP	Connect
instance-1	us-west1-c		10.138.0.2	35.185.245.105	<input type="button" value="SSH"/> <div style="margin-left: 10px;"> <input type="button" value="Open in browser window"/> <input type="button" value="Open in browser window on custom port"/> <input type="button" value="View gcloud command"/> <input type="button" value="Use another SSH client"/> </div>

- H2O provides a script for you to run in your VM instance. Open an editor in the VM instance (for example, vi). Copy one of the scripts below (depending on whether you are running GPUs or CPUs). Save the script as **install.sh**.

```
# SCRIPT FOR GPUs ONLY
apt-get -y update
apt-get -y --no-install-recommends install \
curl \
apt-utils \
```

(continues on next page)

(continued from previous page)

```

python-software-properties \
software-properties-common

add-apt-repository -y ppa:graphics-drivers/ppa
add-apt-repository -y "deb [arch=amd64] https://download.docker.com/linux/
↪ubuntu $(lsb_release -cs) stable"
curl -fsSL https://download.docker.com/linux/ubuntu/gpg | apt-key add -

apt-get update
apt-get install -y \
    nvidia-384 \
    nvidia-modprobe \
    docker-ce

curl -s -L https://nvidia.github.io/nvidia-docker/gpgkey | \
    sudo apt-key add -
distribution=$( . /etc/os-release;echo $ID$VERSION_ID)
curl -s -L https://nvidia.github.io/nvidia-docker/$distribution/nvidia-
↪docker.list | \
    sudo tee /etc/apt/sources.list.d/nvidia-docker.list
sudo apt-get update

# Install nvidia-docker2 and reload the Docker daemon configuration
sudo apt-get install -y nvidia-docker2

```

```

# SCRIPT FOR CPUs ONLY
apt-get -y update
apt-get -y --no-install-recommends install \
    curl \
    apt-utils \
    python-software-properties \
    software-properties-common

add-apt-repository -y "deb [arch=amd64] https://download.docker.com/linux/
↪ubuntu $(lsb_release -cs) stable"
curl -fsSL https://download.docker.com/linux/ubuntu/gpg | apt-key add -

apt-get update
apt-get install -y docker-ce

```

- Type the following commands to run the install script.

```

chmod +x install.sh
sudo ./install.sh

```

- In your user folder, create the following directories as your user.

```

mkdir ~/tmp
mkdir ~/log
mkdir ~/data
mkdir ~/scripts
mkdir ~/license
mkdir ~/demo
mkdir -p ~/jupyter/notebooks

```

- Add your Google Compute user name to the Docker container.

```
sudo usermod -aG docker <username>
```

11. Reboot the system to enable NVIDIA drivers.

```
sudo reboot
```

12. Retrieve the Driverless AI Docker image from <https://www.h2o.ai/download/>.

13. Load the Driverless AI Docker image. The following example shows how to load Driverless AI. Replace VERSION with your image.

```
sudo docker load < dai-docker-centos7-x86_64-1.9.0-10.0.tar.gz
```

14. If you are running CPUs, you can skip this step. Otherwise, you must enable persistence of the GPU. Note that this needs to be run once every reboot. Refer to the following for more information: <http://docs.nvidia.com/deploy/driver-persistence/index.html>.

```
sudo nvidia-persistenced --persistence-mode
```

15. Start the Driverless AI Docker image and replace TAG below with the image tag. Depending on your install version, use the `docker run --runtime=nvidia` (>= Docker 19.03) or `nvidia-docker` (< Docker 19.03) command. Refer to [Enabling Data Connectors](#) for information on how to add the GCS and GBQ data connectors to your Driverless AI instance.

Note: Use `docker version` to check which version of Docker you are using.

>= Docker 19.03

< Docker 19.03

```
# Start the Driverless AI Docker image
docker run --runtime=nvidia \
    --pid=host \
    --init \
    --rm \
    --shm-size=256m \
    -u `id -u`:`id -g` \
    -p 12345:12345 \
    -v `pwd`/data:/data \
    -v `pwd`/log:/log \
    -v `pwd`/license:/license \
    -v `pwd`/tmp:/tmp \
    h2oai/dai-centos7-x86_64:TAG
```

```
# Start the Driverless AI Docker image
nvidia-docker run \
    --pid=host \
    --init \
    --rm \
    --shm-size=256m \
    -u `id -u`:`id -g` \
    -p 12345:12345 \
    -v `pwd`/data:/data \
    -v `pwd`/log:/log \
    -v `pwd`/license:/license \
    -v `pwd`/tmp:/tmp \
    h2oai/dai-centos7-x86_64:TAG
```

Driverless AI will begin running:

```
-----
Welcome to H2O.ai's Driverless AI
-----
- Put data in the volume mounted at /data
- Logs are written to the volume mounted at /log/20180606-044258
- Connect to Driverless AI on port 12345 inside the container
- Connect to Jupyter notebook on port 8888 inside the container
```

16. Connect to Driverless AI with your browser:

```
http://Your-Driverless-AI-Host-Machine:12345
```

Stopping the GCE Instance

The Google Compute Engine instance will continue to run even when you close the portal. You can stop the instance using one of the following methods:

Stopping in the browser

1. On the VM Instances page, click on the VM instance that you want to stop.
2. Click **Stop** at the top of the page.
3. A confirmation page will display. Click **Stop** to stop the instance.

Stopping in Terminal

SSH into the machine that is running Driverless AI, and then run the following:

```
h2oai stop
```

Upgrading Driverless AI

This section provides instructions for upgrading Driverless AI versions that were installed in a Docker container. These steps ensure that existing experiments are saved.

WARNING: Experiments, MLIs, and MOJOS reside in the Driverless AI tmp directory and are not automatically upgraded when Driverless AI is upgraded.

- Build MLi models before upgrading.
- Build MOJO pipelines before upgrading.
- Stop Driverless AI and make a backup of your Driverless AI tmp directory before upgrading.

If you did not build MLi on a model before upgrading Driverless AI, then you will not be able to view MLi on that model after upgrading. Before upgrading, be sure to run MLi jobs on models that you want to continue to interpret in future releases. If that MLi job appears in the list of Interpreted Models in your current version, then it will be retained after upgrading.

If you did not build a MOJO pipeline on a model before upgrading Driverless AI, then you will not be able to build a MOJO pipeline on that model after upgrading. Before upgrading, be sure to build MOJO pipelines on all desired models and then back up your Driverless AI tmp directory.

Note: Stop Driverless AI if it is still running.

Requirements

As of 1.7.0, CUDA 9 is no longer supported. Your host environment must have CUDA 10.0 or later with NVIDIA drivers >= 440.82 installed (GPU only). Driverless AI ships with its own CUDA libraries, but the driver must exist in the host environment. Go to <https://www.nvidia.com/Download/index.aspx> to get the latest NVIDIA Tesla V/P/K series driver.

Upgrade Steps

1. SSH into the IP address of the machine that is running Driverless AI.
2. Set up a directory for the version of Driverless AI on the host machine:

```
# Set up directory with the version name
mkdir dai-1.9.0

# cd into the new directory
cd dai-1.9.0
```

3. Retrieve the Driverless AI package from <https://www.h2o.ai/download/> and add it to the new directory.
4. Load the Driverless AI Docker image inside the new directory:

```
# Load the Driverless AI docker image
docker load < dai-docker-centos7-x86_64-1.9.0-10.0.tar.gz
```

5. Copy the data, log, license, and tmp directories from the previous Driverless AI directory to the new Driverless AI directory:

```
# Copy the data, log, license, and tmp directories on the host machine
cp -a dai_rel_1.4.2/data dai-1.9.0/data
cp -a dai_rel_1.4.2/log dai-1.9.0/log
cp -a dai_rel_1.4.2/license dai-1.9.0/license
cp -a dai_rel_1.4.2/tmp dai-1.9.0/tmp
```

At this point, your experiments from the previous versions will be visible inside the Docker container.

6. Use `docker images` to find the new image tag.
7. Start the Driverless AI Docker image.
8. Connect to Driverless AI with your browser at <http://Your-Driverless-AI-Host-Machine:12345>.

11.2 IBM Power Installs

This section provides installation steps for IBM Power environments. This includes information for Docker image installs, RPMs, Deb, and Tar installs.

Note: OpenCL and LightGBM with GPUs are not supported on Power currently

11.2.1 IBM Docker Images

To simplify local installation, Driverless AI is provided as a Docker image for the following system combination:

Host OS	Docker Version	Host Architecture	Min Mem
RHEL or CentOS 7.4 or later	Docker CE	ppc64le	64 GB

Notes:

- CUDA 10 or later with NVIDIA drivers >= 440.82 (GPU only)
- OpenCL and LightGBM with GPUs are not supported on Power currently.

For the best performance, including GPU support, use nvidia-docker2. For a lower-performance experience without GPUs, use regular docker (with the same docker image).

These installation steps assume that you have a license key for Driverless AI. For information on how to obtain a license key for Driverless AI, visit <https://www.h2o.ai/products/h2o-driverless-ai/>. Once obtained, you will be prompted to paste the license key into the Driverless AI UI when you first log in, or you can save it as a .sig file and place it in the license folder that you will create during the installation process.

Install on IBM with GPUs

This section describes how to install and start the Driverless AI Docker image on RHEL for IBM Power LE systems with GPUs. Note that nvidia-docker has limited support for ppc64le machines. More information about nvidia-docker support for ppc64le machines is available [here](#).

Open a Terminal and ssh to the machine that will run Driverless AI. Once you are logged in, perform the following steps.

1. Retrieve the Driverless AI Docker image from <https://www.h2o.ai/download/>.
2. Add the following to cuda-rhel7.repo in /etc/yum.repos.d/:

```
[cuda]
name=cuda
baseurl=http://developer.download.nvidia.com/compute/cuda/repos/rhel7/ppc64le
enabled=1
gpgcheck=1
gpgkey=http://developer.download.nvidia.com/compute/cuda/repos/rhel7/ppc64le/
→7fa2af80.pub
```

3. Add the following to nvidia-container-runtime.repo in /etc/yum.repos.d/:

```
[libnvidia-container]
name=libnvidia-container
baseurl=https://nvidia.github.io/libnvidia-container/centos7/$basearch
repo_gpgcheck=1
gpgcheck=0
enabled=1
gpgkey=https://nvidia.github.io/libnvidia-container/gpgkey
sslverify=1
sslcacert=/etc/pki/tls/certs/ca-bundle.crt

[nvidia-container-runtime]
name=nvidia-container-runtime
baseurl=https://nvidia.github.io/nvidia-container-runtime/centos7/$basearch
```

(continues on next page)

(continued from previous page)

```
repo_gpgcheck=1
gpgcheck=0
enabled=1
gpgkey=https://nvidia.github.io/nvidia-container-runtime/gpgkey
sslverify=1
sslcacert=/etc/pki/tls/certs/ca-bundle.crt
```

4. Install the latest drivers and the latest version of CUDA:

```
yum -y install nvidia-driver-latest-dkms cuda --nogpgcheck
```

5. Install Docker on RedHat:

```
yum -y install docker
```

6. Install NVIDIA hook. (See <https://github.com/NVIDIA/nvidia-docker#rhel-docker> for more information.) This automatically switches Docker's runtime to nvidia-runtime:

```
yum -y install nvidia-container-runtime-hook
```

7. Set up a directory for the version of Driverless AI on the host machine:

```
# Set up directory with the version name
mkdir dai-1.9.0
```

8. Change directories to the new folder, then load the Driverless AI Docker image inside the new directory:

```
# cd into the new directory
cd dai-1.9.0

# Load the Driverless AI docker image
docker load < dai-docker-centos7-ppc64le-1.9.0-10.0.tar.gz
```

9. Enable persistence of the GPU. Note that this needs to be run once every reboot. Refer to the following for more information: <http://docs.nvidia.com/deploy/driver-persistence/index.html>.

```
sudo nvidia-persistenced --persistence-mode
```

10. Set up the data, log, and license directories on the host machine (within the new directory):

```
# Set up the data, log, license, and tmp directories on the host machine
mkdir data
mkdir log
mkdir license
mkdir tmp
```

11. At this point, you can copy data into the data directory on the host machine. The data will be visible inside the Docker container.

12. Run docker images to find the image tag.

13. Start the Driverless AI Docker image and replace TAG below with the image tag. Depending on your install version, use the docker run --runtime=nvidia (>= Docker 19.03) or nvidia-docker (< Docker 19.03) command.

Note: Use docker version to check which version of Docker you are using.

>= Docker 19.03

< Docker 19.03

```
# Start the Driverless AI Docker image
docker run --runtime=nvidia \
    --pid=host \
    --init \
    --rm \
    --shm-size=256m \
    -u `id -u`:`id -g` \
    -p 12345:12345 \
    -v `pwd`/data:/data \
    -v `pwd`/log:/log \
    -v `pwd`/license:/license \
    -v `pwd`/tmp:/tmp \
    h2oai/dai-centos7-x86_64:TAG
```

```
# Start the Driverless AI Docker image
nvidia-docker run \
    --pid=host \
    --init \
    --rm \
    --shm-size=256m \
    -u `id -u`:`id -g` \
    -p 12345:12345 \
    -v `pwd`/data:/data \
    -v `pwd`/log:/log \
    -v `pwd`/license:/license \
    -v `pwd`/tmp:/tmp \
    h2oai/dai-centos7-x86_64:TAG
```

Driverless AI will begin running:

```
-----
Welcome to H2O.ai's Driverless AI
-----
- Put data in the volume mounted at /data
- Logs are written to the volume mounted at /log/20180606-044258
- Connect to Driverless AI on port 12345 inside the container
- Connect to Jupyter notebook on port 8888 inside the container
```

14. Connect to Driverless AI with your browser at <http://Your-Driverless-AI-Host-Machine:12345>.

Install on IBM with CPUs

This section describes how to install and start the Driverless AI Docker image on RHEL for IBM Power LE systems with CPUs. Note that this uses `docker` and not `nvidia-docker`. GPU support will not be available.

Watch the installation video [here](#). Note that some of the images in this video may change between releases, but the installation steps remain the same.

Note: As of this writing, Driverless AI has only been tested on RHEL version 7.4.

Open a Terminal and ssh to the machine that will run Driverless AI. Once you are logged in, perform the following steps.

1. Install and start Docker CE.

```
sudo yum install -y yum-utils
sudo yum-config-manager --add-repo https://download.docker.com/linux/centos/
  docker-ce.repo
sudo yum makecache fast
sudo yum -y install docker-ce
sudo systemctl start docker
```

2. On the machine that is running Docker CE, retrieve the Driverless AI Docker image from <https://www.h2o.ai/driverless-ai-download/>.
3. Set up a directory for the version of Driverless AI on the host machine:

```
# Set up directory with the version name
mkdir dai-1.9.0
```

4. Load the Driverless AI Docker image inside the new directory:

```
# Load the Driverless AI Docker image
docker load < dai-docker-centos7-ppc64le-1.9.0-10.0.tar.gz
```

5. Set up the data, log, license, and tmp directories (within the new directory):

```
# cd into the directory associated with your version of Driverless AI
cd dai-1.9.0

# Set up the data, log, license, and tmp directories on the host machine
mkdir data
mkdir log
mkdir license
mkdir tmp
```

6. Copy data into the **data** directory on the host. The data will be visible inside the Docker container at **/<user-home>/data**.
7. Run `docker images` to find the image tag.

8. Start the Driverless AI Docker image. Note that GPU support will not be available.

```
$ docker run \
  --pid=host \
  --init \
  --rm \
  -u `id -u`:`id -g` \
  -p 12345:12345 \
  -v `pwd`/data:/data \
  -v `pwd`/log:/log \
  -v `pwd`/license:/license \
  -v `pwd`/tmp:/tmp \
  -v /etc/passwd:/etc/passwd:ro \
  -v /etc/group:/etc/group:ro \
  harbor.h2o.ai/h2oai/dai-centos7-x86_64:1.9.0-cuda10.0.23
```

Driverless AI will begin running:

```
-----
Welcome to H2O.ai's Driverless AI
-----
```

(continues on next page)

(continued from previous page)

- Put data in the volume mounted at /data
- Logs are written to the volume mounted at /log/20180606-044258
- Connect to Driverless AI on port 12345 inside the container
- Connect to Jupyter notebook on port 8888 inside the container

9. Connect to Driverless AI with your browser at <http://Your-Driverless-AI-Host-Machine:12345>.

Stopping the Docker Image

To stop the Driverless AI Docker image, type **Ctrl + C** in the Terminal (Mac OS X) or PowerShell (Windows 10) window that is running the Driverless AI Docker image.

Upgrading the Docker Image

This section provides instructions for upgrading Driverless AI versions that were installed in a Docker container. These steps ensure that existing experiments are saved.

WARNING: Experiments, MLIs, and MOJOS reside in the Driverless AI tmp directory and are not automatically upgraded when Driverless AI is upgraded.

- Build ML models before upgrading.
- Build MOJO pipelines before upgrading.
- Stop Driverless AI and make a backup of your Driverless AI tmp directory before upgrading.

If you did not build ML on a model before upgrading Driverless AI, then you will not be able to view ML on that model after upgrading. Before upgrading, be sure to run ML jobs on models that you want to continue to interpret in future releases. If that ML job appears in the list of Interpreted Models in your current version, then it will be retained after upgrading.

If you did not build a MOJO pipeline on a model before upgrading Driverless AI, then you will not be able to build a MOJO pipeline on that model after upgrading. Before upgrading, be sure to build MOJO pipelines on all desired models and then back up your Driverless AI tmp directory.

Note: Stop Driverless AI if it is still running.

Requirements

As of 1.7.0, CUDA 9 is no longer supported. Your host environment must have CUDA 10.0 or later with NVIDIA drivers >= 440.82 installed (GPU only). Driverless AI ships with its own CUDA libraries, but the driver must exist in the host environment. Go to <https://www.nvidia.com/Download/index.aspx> to get the latest NVIDIA Tesla V/P/K series driver.

Upgrade Steps

1. SSH into the IP address of the machine that is running Driverless AI.
2. Set up a directory for the version of Driverless AI on the host machine:

```
# Set up directory with the version name
mkdir dai-1.9.0

# cd into the new directory
cd dai-1.9.0
```

3. Retrieve the Driverless AI package from <https://www.h2o.ai/download/> and add it to the new directory.
4. Load the Driverless AI Docker image inside the new directory:

```
# Load the Driverless AI docker image
docker load < dai-docker-centos7-x86_64-1.9.0-10.0.tar.gz
```

5. Copy the data, log, license, and tmp directories from the previous Driverless AI directory to the new Driverless AI directory:

```
# Copy the data, log, license, and tmp directories on the host machine
cp -a dai_rel_1.4.2/data dai-1.9.0/data
cp -a dai_rel_1.4.2/log dai-1.9.0/log
cp -a dai_rel_1.4.2/license dai-1.9.0/license
cp -a dai_rel_1.4.2/tmp dai-1.9.0/tmp
```

At this point, your experiments from the previous versions will be visible inside the Docker container.

6. Use `docker images` to find the new image tag.
7. Start the Driverless AI Docker image.
8. Connect to Driverless AI with your browser at <http://Your-Driverless-AI-Host-Machine:12345>.

11.2.2 IBM RPMs

For IBM machines that will not use the Docker image or DEB, an RPM installation is available for ppc64le RHEL 7.

The installation steps assume that you have a license key for Driverless AI. For information on how to obtain a license key for Driverless AI, visit <https://www.h2o.ai/products/h2o-driverless-ai/>. Once obtained, you will be prompted to paste the license key into the Driverless AI UI when you first log in, or you can save it as a .sig file and place it in the license folder that you will create during the installation process.

Note: OpenCL and LightGBM with GPUs are not supported on Power currently.

Requirements

- RedHat 7
- CUDA 10 or later with NVIDIA drivers \geq 440.82 (GPU only)
- cuDNN \geq 7.2.1 (Required for TensorFlow support on GPUs.)
- Driverless AI RPM, available from <https://www.h2o.ai/download/>

About the Install

- The ‘dai’ service user is created locally (in /etc/passwd) if it is not found by ‘getent passwd’. You can override the user by providing the DAI_USER environment variable during rpm or dpkg installation.
- The ‘dai’ service group is created locally (in /etc/group) if it is not found by ‘getent group’. You can override the group by providing the DAI_GROUP environment variable during rpm or dpkg installation.
- Configuration files are put in **/etc/dai** and owned by the ‘root’ user:
 - /etc/dai/config.toml**: Driverless AI config file (See [Using the config.toml File](#) section for details)
 - /etc/dai/User.conf**: Systemd config file specifying the service user
 - /etc/dai/Group.conf**: Systemd config file specifying the service group
 - /etc/dai/EnvironmentFile.conf**: Systemd config file specifying (optional) environment variable overrides
- Software files are put in **/opt/h2oai/dai** and owned by the ‘root’ user
- The following directories are owned by the service user so they can be updated by the running software:
 - /opt/h2oai/dai/home**: The application’s home directory (license key files are stored here)
 - /opt/h2oai/dai/tmp**: Experiments and imported data are stored here
 - /opt/h2oai/dai/log**: Log files go here if you are **not** using systemd (if you are using systemd, then the use the standard journalctl tool)
- By default, Driverless AI looks for a license key in **/opt/h2oai/dai/home/.driverlessai/license.sig**. If you are installing Driverless AI programmatically, you can copy a license key file to that location. If no license key is found, the application will interactively guide you to add one from the Web UI.
- systemd unit files are put in **/usr/lib/systemd/system**
- Symbolic links to the configuration files in /etc/dai files are put in **/etc/systemd/system**

If your environment is running an operational systemd, that is the preferred way to manage Driverless AI. The package installs the following systemd services and a wrapper service:

- dai**: Wrapper service that starts/stops the other three services
- dai-dai**: Main Driverless AI process
- dai-h2o**: H2O-3 helper process used by Driverless AI
- dai-procsy**: Procsy helper process used by Driverless AI
- dai-vis-server**: Visualization server helper process used by Driverless AI

If you don’t have systemd, you can also use the provided run script to start Driverless AI.

Installing Driverless AI

Run the following commands to install the Driverless AI RPM.

```
# Install Driverless AI.
sudo rpm -i dai-1.9.0-ppc64le.rpm
```

By default, the Driverless AI processes are owned by the ‘dai’ user and ‘dai’ group. You can optionally specify a different service user and group as shown below. Replace <myuser> and <mygroup> as appropriate.

```
# Temporarily specify service user and group when installing Driverless AI.  
# rpm saves these for systemd in the /etc/dai/User.conf and /etc/dai/Group.conf files.  
sudo DAI_USER=myuser DAI_GROUP=mygroup rpm -i dai-1.9.0-ppc64le.rpm
```

You may now optionally make changes to **/etc/dai/config.toml**.

Starting Driverless AI

If you have systemd (preferred):

```
# If you are running in multinode (worker_info="multinode"), you must start minio  
↪before starting Driverless AI  
sudo systemctl start dai-minio  
  
# Start Driverless AI.  
sudo systemctl start dai
```

If you do not have systemd:

```
# Start Driverless AI.  
sudo -H -u dai /opt/h2oai/dai/run-dai.sh  
  
# No need to start minio separately. When starting Driverless AI manually, the  
↪command above takes care of that.
```

Starting NVIDIA Persistence Mode

If you have NVIDIA GPUs, you must run the following NVIDIA command. This command needs to be run every reboot. For more information: [http://docs.nvidia.com/deploy driver-persistence/index.html](http://docs.nvidia.com/deploy	driver-persistence/index.html).

```
sudo nvidia-persistenced --persistence-mode
```

Looking at Driverless AI log files

If you have systemd (preferred):

```
sudo systemctl status dai-dai  
sudo systemctl status dai-h2o  
sudo systemctl status dai-procsy  
sudo systemctl status dai-vis-server  
sudo journalctl -u dai-dai  
sudo journalctl -u dai-h2o  
sudo journalctl -u dai-procsy  
sudo journalctl -u dai-vis-server
```

If you do not have systemd:

```
sudo less /opt/h2oai/dai/log/dai.log  
sudo less /opt/h2oai/dai/log/h2o.log  
sudo less /opt/h2oai/dai/log/procsy.log  
sudo less /opt/h2oai/dai/log/vis-server.log
```

Stopping Driverless AI

If you have systemd (preferred):

```
# Stop Driverless AI.
sudo systemctl stop dai

# The processes should now be stopped. Verify.
sudo ps -u dai
```

If you do not have systemd:

```
# Stop Driverless AI.
sudo pkill -U dai

# The processes should now be stopped. Verify.
sudo ps -u dai
```

Upgrading Driverless AI

WARNINGS:

- This release deprecates experiments and Mali models from 1.7.0 and earlier.
- Experiments, MLIs, and MOJOs reside in the Driverless AI tmp directory and are not automatically upgraded when Driverless AI is upgraded. We recommend you take the following steps before upgrading.
 - Build Mali models before upgrading.
 - Build MOJO pipelines before upgrading.
 - Stop Driverless AI and make a backup of your Driverless AI tmp directory before upgrading.

If you did not build Mali on a model before upgrading Driverless AI, then you will not be able to view Mali on that model after upgrading. Before upgrading, be sure to run Mali jobs on models that you want to continue to interpret in future releases. If that Mali job appears in the list of Interpreted Models in your current version, then it will be retained after upgrading.

If you did not build a MOJO pipeline on a model before upgrading Driverless AI, then you will not be able to build a MOJO pipeline on that model after upgrading. Before upgrading, be sure to build MOJO pipelines on all desired models and then back up your Driverless AI tmp directory.

The upgrade process inherits the service user and group from /etc/dai/User.conf and /etc/dai/Group.conf. You do not need to manually specify the DAI_USER or DAI_GROUP environment variables during an upgrade.

Requirements

As of 1.7.0, CUDA 9 is no longer supported. Your host environment must have CUDA 10.0 or later with NVIDIA drivers >= 440.82 installed (GPU only). Driverless AI ships with its own CUDA libraries, but the driver must exist in the host environment. Go to <https://www.nvidia.com/Download/index.aspx> to get the latest NVIDIA Tesla V/P/K series driver.

Upgrade Steps

If you have systemd (preferred):

```
# Stop Driverless AI.  
sudo systemctl stop dai  
  
# The processes should now be stopped. Verify.  
sudo ps -u dai  
  
# Make a backup of /opt/h2oai/dai/tmp directory at this time.  
  
# Upgrade and restart.  
sudo rpm -U dai-1.9.0-ppc64le.rpm  
sudo systemctl daemon-reload  
sudo systemctl start dai
```

If you do not have systemd:

```
# Stop Driverless AI.  
sudo pkill -U dai  
  
# The processes should now be stopped. Verify.  
sudo ps -u dai  
  
# Make a backup of /opt/h2oai/dai/tmp directory at this time. If you do not, all  
→previous data will be lost.  
  
# Upgrade and restart.  
sudo rpm -U dai-1.9.0-ppc64le.rpm  
sudo -H -u dai /opt/h2oai/dai/run-dai.sh
```

Uninstalling Driverless AI

If you have systemd (preferred):

```
# Stop Driverless AI.  
sudo systemctl stop dai  
  
# The processes should now be stopped. Verify.  
sudo ps -u dai  
  
# Uninstall.  
sudo rpm -e dai
```

If you do not have systemd:

```
# Stop Driverless AI.  
sudo pkill -U dai  
  
# The processes should now be stopped. Verify.  
sudo ps -u dai  
  
# Uninstall.  
sudo rpm -e dai
```

CAUTION! At this point you can optionally completely remove all remaining files, including the database (this cannot be undone):

```
sudo rm -rf /opt/h2oai/dai
sudo rm -rf /etc/dai
```

Note: The UID and GID are not removed during the uninstall process. These can be removed with `userdel` and `usergroup`. However, we DO NOT recommend removing the UID and GID if you plan to re-install Driverless AI. If you remove the UID and GID and then reinstall Driverless AI, the UID and GID will likely be re-assigned to a different (unrelated) user/group in the future; this may cause confusion if there are any remaining files on the filesystem referring to the deleted user or group.

11.2.3 IBM DEB

For IBM machines that will not use the Docker image or RPM, a DEB installation is available for ppc64le Ubuntu 16.04.

Note: OpenCL and LightGBM with GPUs are not supported on Power currently.

Requirements

- Ubuntu 16.04
- CUDA 10 or later with NVIDIA drivers >= 440.82 (GPU only)
- cuDNN >=7.2.1 (Required for TensorFlow support on GPUs.)
- Driverless AI DEB, available from <https://www.h2o.ai/download/>

About the Install

- The ‘dai’ service user is created locally (in /etc/passwd) if it is not found by ‘getent passwd’. You can override the user by providing the DAI_USER environment variable during rpm or dpkg installation.
- The ‘dai’ service group is created locally (in /etc/group) if it is not found by ‘getent group’. You can override the group by providing the DAI_GROUP environment variable during rpm or dpkg installation.
- Configuration files are put in **/etc/dai** and owned by the ‘root’ user:
 - **/etc/dai/config.toml**: Driverless AI config file (See *Using the config.toml File* section for details)
 - **/etc/dai/User.conf**: Systemd config file specifying the service user
 - **/etc/dai/Group.conf**: Systemd config file specifying the service group
 - **/etc/dai/EnvironmentFile.conf**: Systemd config file specifying (optional) environment variable overrides
- Software files are put in **/opt/h2oai/dai** and owned by the ‘root’ user
- The following directories are owned by the service user so they can be updated by the running software:
 - **/opt/h2oai/dai/home**: The application’s home directory (license key files are stored here)
 - **/opt/h2oai/dai/tmp**: Experiments and imported data are stored here
 - **/opt/h2oai/dai/log**: Log files go here if you are **not** using systemd (if you are using systemd, then the use the standard journalctl tool)

- By default, Driverless AI looks for a license key in `/opt/h2oai/dai/home/.driverlessai/license.sig`. If you are installing Driverless AI programmatically, you can copy a license key file to that location. If no license key is found, the application will interactively guide you to add one from the Web UI.
- systemd unit files are put in `/usr/lib/systemd/system`
- Symbolic links to the configuration files in `/etc/dai` files are put in `/etc/systemd/system`

If your environment is running an operational systemd, that is the preferred way to manage Driverless AI. The package installs the following systemd services and a wrapper service:

- **dai**: Wrapper service that starts/stops the other three services
- **dai-dai**: Main Driverless AI process
- **dai-h2o**: H2O-3 helper process used by Driverless AI
- **dai-procsy**: Procsy helper process used by Driverless AI
- **dai-vis-server**: Visualization server helper process used by Driverless AI

If you don't have systemd, you can also use the provided run script to start Driverless AI.

Starting NVIDIA Persistence Mode (GPU only)

If you have NVIDIA GPUs, you must run the following NVIDIA command. This command needs to be run every reboot. For more information: <http://docs.nvidia.com/deploy/driver-persistence/index.html>.

```
sudo nvidia-persistenced --persistence-mode
```

Installing the Driverless AI DEB

Run the following commands to install the Driverless AI DEB.

```
# Install Driverless AI.  
sudo dpkg -i dai_1.9.0_ppc64el.deb
```

By default, the Driverless AI processes are owned by the ‘dai’ user and ‘dai’ group. You can optionally specify a different service user and group as shown below. Replace <myuser> and <mygroup> as appropriate.

```
# Temporarily specify service user and group when installing Driverless AI.  
# dpkg saves these for systemd in the /etc/dai/User.conf and /etc/dai/Group.conf  
# files.  
sudo DAI_USER=myuser DAI_GROUP=mygroup dpkg -i |VERSION-deb-ppc|
```

You may now optionally make changes to `/etc/dai/config.toml`.

Starting Driverless AI

If you have systemd (preferred):

```
# If you are running in multinode (worker_info="multinode"), you must start minio  
# before starting Driverless AI  
sudo systemctl start dai-minio  
  
# Start Driverless AI.  
sudo systemctl start dai
```

If you do not have systemd:

```
# Start Driverless AI.
sudo -H -u dai /opt/h2oai/dai/run-dai.sh

# No need to start minio separately. When starting Driverless AI manually, the
# command above takes care of that.
```

Looking at Driverless AI log files

If you have systemd (preferred):

```
sudo systemctl status dai-dai
sudo systemctl status dai-h2o
sudo systemctl status dai-procsy
sudo systemctl status dai-vis-server
sudo journalctl -u dai-dai
sudo journalctl -u dai-h2o
sudo journalctl -u dai-procsy
sudo journalctl -u dai-vis-server
```

If you do not have systemd:

```
sudo less /opt/h2oai/dai/log/dai.log
sudo less /opt/h2oai/dai/log/h2o.log
sudo less /opt/h2oai/dai/log/procsy.log
sudo less /opt/h2oai/dai/log/vis-server.log
```

Stopping Driverless AI

If you have systemd (preferred):

```
# Stop Driverless AI.
sudo systemctl stop dai

# The processes should now be stopped. Verify.
sudo ps -u dai
```

If you do not have systemd:

```
# Stop Driverless AI.
sudo pkill -U dai

# The processes should now be stopped. Verify.
sudo ps -u dai
```

Upgrading Driverless AI

WARNINGS:

- This release deprecates experiments and MLIs models from 1.7.0 and earlier.
- Experiments, MLIs, and MOJOS reside in the Driverless AI tmp directory and are not automatically upgraded when Driverless AI is upgraded. We recommend you take the following steps before upgrading.
 - Build MLI models before upgrading.
 - Build MOJO pipelines before upgrading.
 - Stop Driverless AI and make a backup of your Driverless AI tmp directory before upgrading.

If you did not build MLI on a model before upgrading Driverless AI, then you will not be able to view MLI on that model after upgrading. Before upgrading, be sure to run MLI jobs on models that you want to continue to interpret in future releases. If that MLI job appears in the list of Interpreted Models in your current version, then it will be retained after upgrading.

If you did not build a MOJO pipeline on a model before upgrading Driverless AI, then you will not be able to build a MOJO pipeline on that model after upgrading. Before upgrading, be sure to build MOJO pipelines on all desired models and then back up your Driverless AI tmp directory.

The upgrade process inherits the service user and group from /etc/dai/User.conf and /etc/dai/Group.conf. You do not need to manually specify the DAI_USER or DAI_GROUP environment variables during an upgrade.

Requirements

As of 1.7.0, CUDA 9 is no longer supported. Your host environment must have CUDA 10.0 or later with NVIDIA drivers >= 440.82 installed (GPU only). Driverless AI ships with its own CUDA libraries, but the driver must exist in the host environment. Go to <https://www.nvidia.com/Download/index.aspx> to get the latest NVIDIA Tesla V/P/K series driver.

Upgrade Steps

If you have systemd (preferred):

```
# Stop Driverless AI.  
sudo systemctl stop dai  
  
# Make a backup of /opt/h2oai/dai/tmp directory at this time. If you do not, all  
# previous data will be lost.  
  
# Upgrade Driverless AI.  
sudo dpkg -i dai_1.9.0_ppc64el.deb  
sudo systemctl daemon-reload  
sudo systemctl start dai
```

If you do not have systemd:

```
# Stop Driverless AI.  
sudo pkill -U dai  
  
# The processes should now be stopped. Verify.  
sudo ps -u dai
```

(continues on next page)

(continued from previous page)

```
# Make a backup of /opt/h2oai/dai/tmp directory at this time.

# Upgrade and restart.
sudo dpkg -i dai_1.9.0_ppc64el.deb
sudo -H -u dai /opt/h2oai/dai/run-dai.sh
```

Uninstalling Driverless AI

If you have systemd (preferred):

```
# Stop Driverless AI.
sudo systemctl stop dai

# The processes should now be stopped. Verify.
sudo ps -u dai

# Uninstall Driverless AI.
sudo dpkg -r dai

# Purge Driverless AI.
sudo dpkg -P dai
```

If you do not have systemd:

```
# Stop Driverless AI.
sudo pkill -U dai

# The processes should now be stopped. Verify.
sudo ps -u dai

# Uninstall Driverless AI.
sudo dpkg -r dai

# Purge Driverless AI.
sudo dpkg -P dai
```

CAUTION! At this point you can optionally completely remove all remaining files, including the database. (This cannot be undone.)

```
sudo rm -rf /opt/h2oai/dai
sudo rm -rf /etc/dai
```

11.2.4 IBM TAR SH

The Driverless AI software is available for use in pure user-mode environments as a self-extracting TAR SH archive. This form of installation does not require a privileged user to install or to run.

This artifact has the same compatibility matrix as the RPM and DEB packages (combined), it just comes packaged slightly differently. See those sections for a full list of supported environments.

The installation steps assume that you have a license key for Driverless AI. For information on how to obtain a license key for Driverless AI, visit <https://www.h2o.ai/products/h2o-driverless-ai/>. Once obtained, you will be prompted to paste the license key into the Driverless AI UI when you first log in.

Note: OpenCL and LightGBM with GPUs are not supported on Power currently.

Requirements

- RedHat 7 or Ubuntu 16.04
- CUDA 10 or later with NVIDIA drivers >= 440.82 (GPU only)
- cuDNN >=7.2.1 (Required for TensorFlow support on GPUs.)
- Driverless AI TAR SH, available from <https://www.h2o.ai/download/>

Installing Driverless AI

Run the following commands to install the Driverless AI RPM.

```
# Install Driverless AI.  
chmod 755 dai-1.9.0-linux-ppc64le.sh  
./dai-1.9.0-linux-ppc64le.sh
```

You may now cd to the unpacked directory and optionally make changes to **config.toml**.

Starting Driverless AI

```
# Start Driverless AI.  
./run-dai.sh
```

Starting NVIDIA Persistence Mode

If you have NVIDIA GPUs, you must run the following NVIDIA command. This command needs to be run every reboot. For more information: <http://docs.nvidia.com/deploy/driver-persistence/index.html>.

```
sudo nvidia-persistenced --persistence-mode
```

Looking at Driverless AI log files

```
less log/dai.log  
less log/h2o.log  
less log/procsy.log  
less log/vis-server.log
```

Stopping Driverless AI

```
# Stop Driverless AI.  
./kill-dai.sh
```

Uninstalling Driverless AI

To uninstall Driverless AI, just remove the directory created by the unpacking process. By default, all files for Driverless AI are contained within this directory.

Upgrading Driverless AI

WARNINGS:

- This release deprecates experiments and ML models from 1.7.0 and earlier.
- Experiments, MLIs, and MOJOs reside in the Driverless AI tmp directory and are not automatically upgraded when Driverless AI is upgraded. We recommend you take the following steps before upgrading.
 - Build ML models before upgrading.
 - Build MOJO pipelines before upgrading.
 - Stop Driverless AI and make a backup of your Driverless AI tmp directory before upgrading.

If you did not build ML on a model before upgrading Driverless AI, then you will not be able to view ML on that model after upgrading. Before upgrading, be sure to run ML jobs on models that you want to continue to interpret in future releases. If that ML job appears in the list of Interpreted Models in your current version, then it will be retained after upgrading.

If you did not build a MOJO pipeline on a model before upgrading Driverless AI, then you will not be able to build a MOJO pipeline on that model after upgrading. Before upgrading, be sure to build MOJO pipelines on all desired models and then back up your Driverless AI tmp directory.

The upgrade process inherits the service user and group from /etc/dai/User.conf and /etc/dai/Group.conf. You do not need to manually specify the DAI_USER or DAI_GROUP environment variables during an upgrade.

Requirements

As of 1.7.0, CUDA 9 is no longer supported. Your host environment must have CUDA 10.0 or later with NVIDIA drivers >= 440.82 installed (GPU only). Driverless AI ships with its own CUDA libraries, but the driver must exist in the host environment. Go to <https://www.nvidia.com/Download/index.aspx> to get the latest NVIDIA Tesla V/P/K series driver.

Upgrade Steps

1. Stop your previous version of Driverless AI.
2. Run the self-extracting archive for the new version of Driverless AI.
3. Port any previous changes you made to your config.toml file to the newly unpacked directory.
4. Copy the tmp directory (which contains all the Driverless AI working state) from your previous Driverless AI installation into the newly upacked directory.
5. Start your newly extracted version of Driverless AI.

11.2.5 Troubleshooting IBM Installations

Opening Port 12345

For default IBM Power9 systems with RHEL 7 installed, be sure to open port 12345 in the firewall. For example:

```
firewall-cmd --zone=public --add-port=12345/tcp --permanent  
firewall-cmd --reload
```

Growing the Disk

Some users may find it necessary to grow their disk. An example describing how to add disk space to a virtual machine is available at <https://www.geoffstratton.com/expand-hard-disk-ubuntu-lvm>. The steps for an IBM Power9 system with RHEL 7 would be similar.

11.3 Mac OS X

This section describes how to install, start, stop, and upgrade the Driverless AI Docker image on Mac OS X. Note that this uses regular Docker and not NVIDIA Docker.

Note: GPU support is not available on Mac OS X.

The installation steps assume that you have a license key for Driverless AI. For information on how to obtain a license key for Driverless AI, visit <https://www.h2o.ai/driverless-ai/>. Once obtained, you will be prompted to paste the license key into the Driverless AI UI when you first log in, or you can save it as a .sig file and place it in the license folder that you will create during the installation process.

Caution:

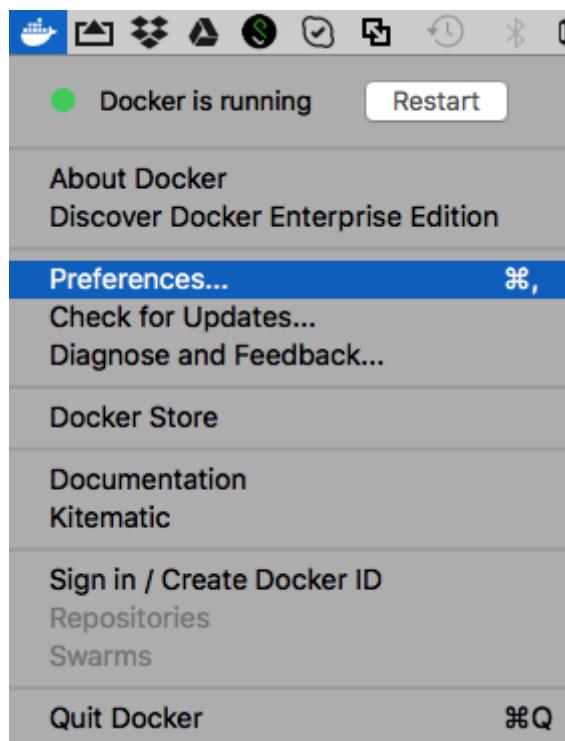
- This is an extremely memory-constrained environment for experimental purposes only. Stick to small datasets! For serious use, please use Linux.
- Be aware that there are known performance issues with Docker for Mac. More information is available here: <https://docs.docker.com/docker-for-mac/osxfs/#technology>.

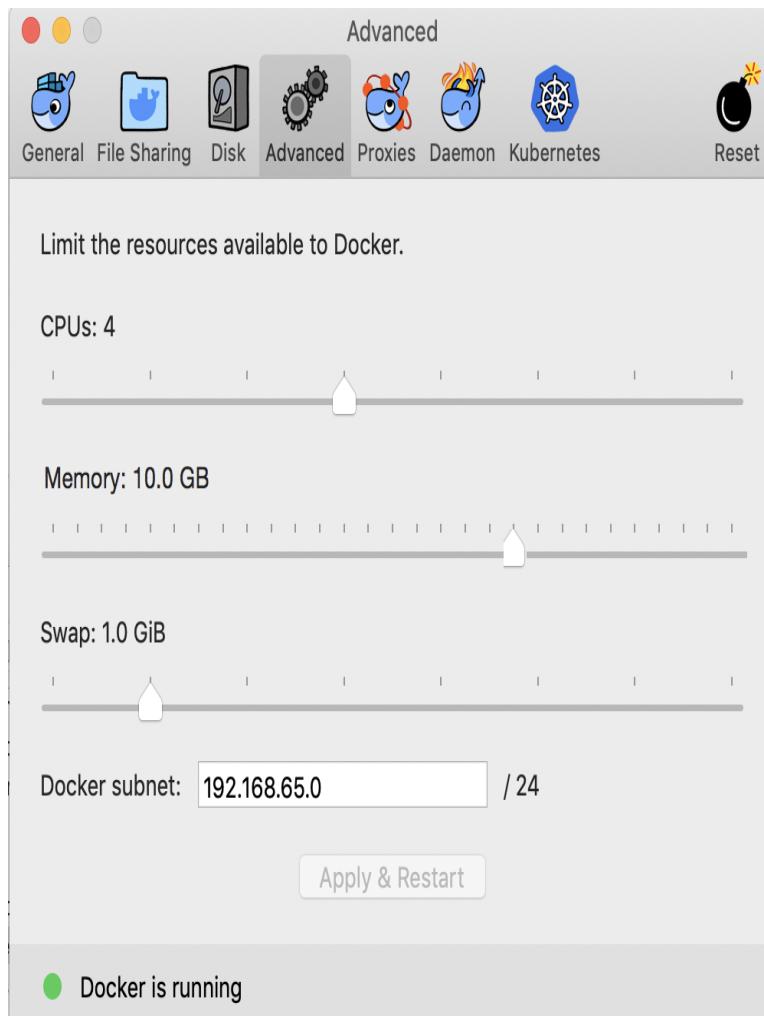
11.3.1 Environment

Operating System	GPU Support?	Min Mem	Suitable for
Mac OS X	No	16 GB	Experimentation

11.3.2 Installing Driverless AI

1. Retrieve the Driverless AI Docker image from <https://www.h2o.ai/download/>.
2. Download and run Docker for Mac from <https://docs.docker.com/docker-for-mac/install>.
3. Adjust the amount of memory given to Docker to be at least 10 GB. Driverless AI won't run at all with less than 10 GB of memory. You can optionally adjust the number of CPUs given to Docker. You will find the controls by clicking on (Docker Whale)->Preferences->Advanced as shown in the following screenshots. (Don't forget to Apply the changes after setting the desired memory value.)





4. On the File Sharing tab, verify that your macOS directories (and their subdirectories) can be bind mounted into Docker containers. More information is available here: <https://docs.docker.com/docker-for-mac/osxfs/#namespaces>.



5. Set up a directory for the version of Driverless AI within the Terminal:

```
mkdir dai-1.9.0
```

6. With Docker running, open a Terminal and move the downloaded Driverless AI image to your new directory.

7. Change directories to the new directory, then load the image using the following command:

```
cd dai-1.9.0
docker load < dai-docker-centos7-x86_64-1.9.0-10.0.tar.gz
```

8. Set up the data, log, license, and tmp directories (within the new Driverless AI directory):

```
mkdir data
mkdir log
mkdir license
mkdir tmp
```

9. Optionally copy data into the **data** directory on the host. The data will be visible inside the Docker container at **/data**. You can also upload data after starting Driverless AI.

10. Run `docker images` to find the image tag.

11. Start the Driverless AI Docker image and replace TAG below with the image tag. Depending on your install version, use the `docker run --runtime=nvidia` (\geq Docker 19.03) or `nvidia-docker` ($<$ Docker 19.03) command.

Note: Use `docker version` to check which version of Docker you are using.

\geq Docker 19.03

$<$ Docker 19.03

```
# Start the Driverless AI Docker image
docker run --runtime=nvidia \
    --pid=host \
    --init \
```

(continues on next page)

(continued from previous page)

```
--rm \
--shm-size=256m \
-u `id -u`:`id -g` \
-p 12345:12345 \
-v `pwd`/data:/data \
-v `pwd`/log:/log \
-v `pwd`/license:/license \
-v `pwd`/tmp:/tmp \
h2oai/dai-centos7-x86_64:TAG
```

```
# Start the Driverless AI Docker image
nvidia-docker run \
--pid=host \
--init \
--rm \
--shm-size=256m \
-u `id -u`:`id -g` \
-p 12345:12345 \
-v `pwd`/data:/data \
-v `pwd`/log:/log \
-v `pwd`/license:/license \
-v `pwd`/tmp:/tmp \
h2oai/dai-centos7-x86_64:TAG
```

12. Connect to Driverless AI with your browser at <http://localhost:12345>.

11.3.3 Stopping the Docker Image

To stop the Driverless AI Docker image, type **Ctrl + C** in the Terminal (Mac OS X) or PowerShell (Windows 10) window that is running the Driverless AI Docker image.

11.3.4 Upgrading the Docker Image

This section provides instructions for upgrading Driverless AI versions that were installed in a Docker container. These steps ensure that existing experiments are saved.

WARNING: Experiments, MLIs, and MOJOS reside in the Driverless AI tmp directory and are not automatically upgraded when Driverless AI is upgraded.

- Build MLIs before upgrading.
- Build MOJO pipelines before upgrading.
- Stop Driverless AI and make a backup of your Driverless AI tmp directory before upgrading.

If you did not build MLIs on a model before upgrading Driverless AI, then you will not be able to view MLIs on that model after upgrading. Before upgrading, be sure to run MLIs on models that you want to continue to interpret in future releases. If that MLI job appears in the list of Interpreted Models in your current version, then it will be retained after upgrading.

If you did not build a MOJO pipeline on a model before upgrading Driverless AI, then you will not be able to build a MOJO pipeline on that model after upgrading. Before upgrading, be sure to build MOJO pipelines on all desired models and then back up your Driverless AI tmp directory.

Note: Stop Driverless AI if it is still running.

Upgrade Steps

1. SSH into the IP address of the machine that is running Driverless AI.
2. Set up a directory for the version of Driverless AI on the host machine:

```
# Set up directory with the version name
mkdir dai-1.9.0

# cd into the new directory
cd dai-1.9.0
```

3. Retrieve the Driverless AI package from <https://www.h2o.ai/download/> and add it to the new directory.
4. Load the Driverless AI Docker image inside the new directory:

```
# Load the Driverless AI docker image
docker load < dai-docker-centos7-x86_64-1.9.0-10.0.tar.gz
```

5. Copy the data, log, license, and tmp directories from the previous Driverless AI directory to the new Driverless AI directory:

```
# Copy the data, log, license, and tmp directories on the host machine
cp -a dai_rel_1.8.4/data dai-1.9.0/data
cp -a dai_rel_1.8.4/log dai-1.9.0/log
cp -a dai_rel_1.8.4/license dai-1.9.0/license
cp -a dai_rel_1.8.4/tmp dai-1.9.0/tmp
```

At this point, your experiments from the previous versions will be visible inside the Docker container.

6. Use `docker images` to find the new image tag.
7. Start the Driverless AI Docker image.
8. Connect to Driverless AI with your browser at <http://Your-Driverless-AI-Host-Machine:12345>.

11.4 Windows 10

This section describes how to install, start, stop, and upgrade Driverless AI on a Windows 10 machine. The installation steps assume that you have a license key for Driverless AI. For information on how to obtain a license key for Driverless AI, visit <https://www.h2o.ai/driverless-ai/>. Once obtained, you will be prompted to paste the license key into the Driverless AI UI when you first log in, or you can save it as a .sig file and place it in the license folder that you will create during the installation process.

11.4.1 Overview of Installation on Windows

To install Driverless AI on Windows, use a Driverless AI Docker image.

Notes:

- GPU support is not available on Windows.
- Scoring is not available on Windows.

Caution: This should be used only for experimental purposes and only on small data. For serious use, please use Linux.

11.4.2 Environment

Operating System	GPU Support?	Min Mem	Suitable for
Windows 10 Pro	No	16 GB	Experimentation
Windows 10 Enterprise	No	16 GB	Experimentation
Windows 10 Education	No	16 GB	Experimentation

Note: Driverless AI cannot be installed on versions of Windows 10 that do not support Hyper-V. Refer to <https://docs.microsoft.com/en-us/virtualization/hyper-v-on-windows/reference/hyper-v-requirements> for more information.

11.4.3 Docker Image Installation

Notes:

- Be aware that there are known issues with Docker for Windows. More information is available here: <https://github.com/docker/for-win/issues/188>.
- Consult with your Windows System Admin if
 - Your corporate environment does not allow third-part software installs
 - You are running Windows Defender
 - Your machine is not running with Enable-WindowsOptionalFeature -Online -FeatureName Microsoft-Windows-Subsystem-Linux.

Watch the installation video [here](#). Note that some of the images in this video may change between releases, but the installation steps remain the same.

Requirements

- Windows 10 Pro / Enterprise / Education
- Docker Desktop for Windows 2.2.0.3 (42716)

Note: As of this writing, Driverless AI has only been tested on Docker Desktop for Windows version 2.2.0.3 (42716).

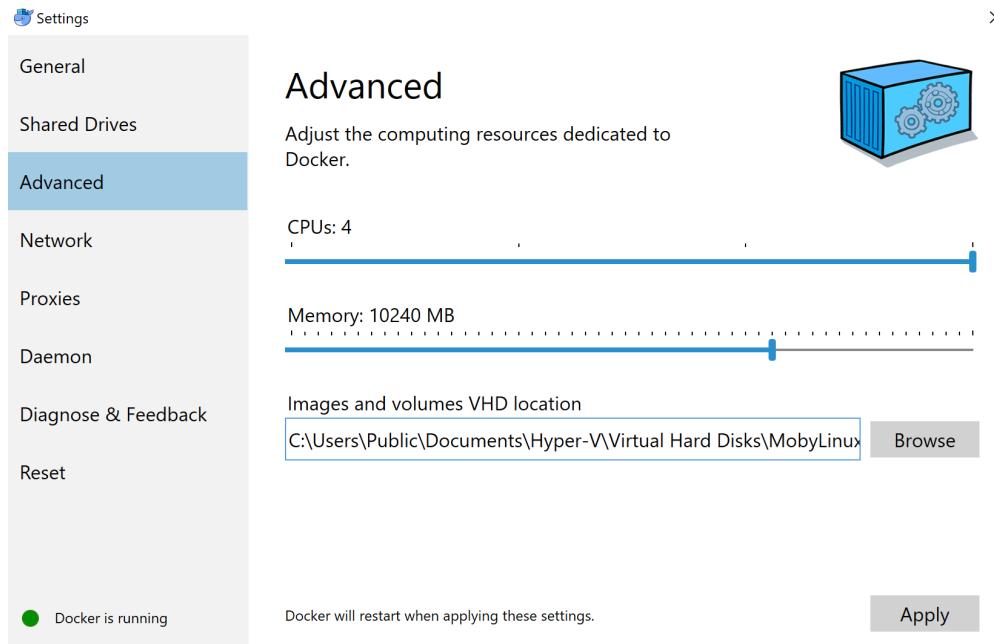
Installation Procedure

1. Retrieve the Driverless AI Docker image from <https://www.h2o.ai/download/>.
2. Download, install, and run Docker for Windows from <https://docs.docker.com/docker-for-windows/install/>. You can verify that Docker is running by typing `docker version` in a terminal (such as Windows PowerShell). Note that you may have to reboot after installation.
3. Before running Driverless AI, you must:
 - Enable shared access to the C drive. Driverless AI will not be able to see your local data if this is not set.
 - Adjust the amount of memory given to Docker to be at least 10 GB. Driverless AI won't run at all with less than 10 GB of memory.
 - Optionally adjust the number of CPUs given to Docker.

You can adjust these settings by clicking on the Docker whale in your taskbar (look for hidden tasks, if necessary), then selecting **Settings > Shared Drive** and **Settings > Advanced** as shown in the following screenshots. Don't forget to Apply the changes after setting the desired memory value. (Docker will

restart.) Note that if you cannot make changes, stop Docker and then start Docker again by right clicking on the Docker icon on your desktop and selecting **Run as Administrator**.

The image shows two screenshots related to Docker configuration. The top screenshot is a context menu from the Docker icon on a Windows desktop. The 'Settings...' option is highlighted in blue. The menu includes options like 'About Docker', 'Discover Docker Enterprise Edition', 'Check for Updates...', 'Diagnose and Feedback...', 'Switch to Windows containers...', 'Docker Store', 'Documentation', 'Kitematic', 'Sign in / Create Docker ID...', 'Swarms', 'Repositories', and 'Quit Docker'. The bottom screenshot shows the 'Shared Drives' section of the Docker Settings window. The left sidebar has 'General' selected, while 'Shared Drives' is the active tab. It displays a message: 'Select the local drives you want to be available to your containers.' A table lists a single drive: 'Shared' column has a checked checkbox and the letter 'C'; 'Drive' column has the letter 'C'. Below the table is a Microsoft PowerShell window showing the command: > docker run --rm -v c:/Users:/data alpine ls /data. At the bottom left is a green circle with a white dot labeled 'Docker is running'. At the bottom right are 'Reset credentials...' and 'Apply' buttons.



4. Open a PowerShell terminal and set up a directory for the version of Driverless AI on the host machine:

```
md dai-1.9.0
```

5. With Docker running, navigate to the location of your downloaded Driverless AI image. Move the downloaded Driverless AI image to your new directory.
6. Change directories to the new directory, then load the image using the following command:

```
cd dai-1.9.0  
docker load -i .\dai-docker-centos7-x86_64-1.9.0-10.0.tar.gz
```

7. Set up the data, log, license, and tmp directories (within the new directory).

```
md data  
md log  
md license  
md tmp
```

8. Copy data into the **/data** directory. The data will be visible inside the Docker container at **/data**.
9. Run `docker images` to find the image tag.
10. Start the Driverless AI Docker image. Be sure to replace `path_to_` below with the entire path to the location of the folders that you created (for example, “`c:/Users/user-name/driverlessai_folder/data`”). Note that this is regular Docker, not NVIDIA Docker. GPU support will not be available.

```
docker run --pid=host --init --rm --shm-size=256m -p 12345:12345 -v c:/path_to_data:/data -v c:/path_to_log:/log -v c:/path_to_license:/license -v c:/path_to_tmp:/tmp h2oai/dai-centos7-x86_64:1.9.0-cuda10.0.23
```

11. Connect to Driverless AI with your browser at <http://localhost:12345>.

Stopping the Docker Image

To stop the Driverless AI Docker image, type **Ctrl + C** in the Terminal (Mac OS X) or PowerShell (Windows 10) window that is running the Driverless AI Docker image.

Upgrading the Docker Image

This section provides instructions for upgrading Driverless AI versions that were installed in a Docker container. These steps ensure that existing experiments are saved.

WARNING: Experiments, MLIs, and MOJOS reside in the Driverless AI tmp directory and are not automatically upgraded when Driverless AI is upgraded.

- Build MLIs models before upgrading.
- Build MOJO pipelines before upgrading.
- Stop Driverless AI and make a backup of your Driverless AI tmp directory before upgrading.

If you did not build MLIs on a model before upgrading Driverless AI, then you will not be able to view MLIs on that model after upgrading. Before upgrading, be sure to run MLIs jobs on models that you want to continue to interpret in future releases. If that MLIs job appears in the list of Interpreted Models in your current version, then it will be retained after upgrading.

If you did not build a MOJO pipeline on a model before upgrading Driverless AI, then you will not be able to build a MOJO pipeline on that model after upgrading. Before upgrading, be sure to build MOJO pipelines on all desired models and then back up your Driverless AI tmp directory.

Note: Stop Driverless AI if it is still running.

Requirements

As of 1.7.0, CUDA 9 is no longer supported. Your host environment must have CUDA 10.0 or later with NVIDIA drivers >= 440.82 installed (GPU only). Driverless AI ships with its own CUDA libraries, but the driver must exist in the host environment. Go to <https://www.nvidia.com/Download/index.aspx> to get the latest NVIDIA Tesla V/P/K series driver.

Upgrade Steps

1. SSH into the IP address of the machine that is running Driverless AI.
2. Set up a directory for the version of Driverless AI on the host machine:

```
# Set up directory with the version name
mkdir dai-1.9.0

# cd into the new directory
cd dai-1.9.0
```

3. Retrieve the Driverless AI package from <https://www.h2o.ai/download/> and add it to the new directory.
4. Load the Driverless AI Docker image inside the new directory:

```
# Load the Driverless AI docker image
docker load < dai-docker-centos7-x86_64-1.9.0-10.0.tar.gz
```

5. Copy the data, log, license, and tmp directories from the previous Driverless AI directory to the new Driverless AI directory:

```
# Copy the data, log, license, and tmp directories on the host machine
cp -a dai_rel_1.4.2/data dai-1.9.0/data
cp -a dai_rel_1.4.2/log dai-1.9.0/log
cp -a dai_rel_1.4.2/license dai-1.9.0/license
cp -a dai_rel_1.4.2/tmp dai-1.9.0/tmp
```

At this point, your experiments from the previous versions will be visible inside the Docker container.

6. Use docker images to find the new image tag.
7. Start the Driverless AI Docker image.
8. Connect to Driverless AI with your browser at <http://Your-Driverless-AI-Host-Machine:12345>.

USING THE CONFIG.TOML FILE

Admins can edit a config.toml file when starting the Driverless AI Docker image. The config.toml file includes all possible configuration options that would otherwise be specified in the nvidia-docker run command. This file is located in a folder on the container. You can make updates to environment variables directly in this file. Driverless AI will use the updated config.toml file when starting from native installs. Docker users can specify that updated config.toml file when starting Driverless AI Docker image.

12.1 Configuration Override Chain

The configuration engine reads and overrides variables in the following order:

1. **h2oai/config/config.toml** - This is an internal file that is not visible.
2. **config.toml** - Place this file in a folder or mount it in a Docker container and specify the path in the “DRIVERLESS_AI_CONFIG_FILE” environment variable.
3. **Environment variable** - Configuration variables can also be provided as environment variables. They must have the prefix DRIVERLESS_AI_ followed by the variable name in all caps. For example, “authentication_method” can be provided as “DRIVERLESS_AI_AUTHENTICATION_METHOD”.

Docker Image Users

Native Install Users

1. Copy the config.toml file from inside the Docker image to your local filesystem.

```
# Make a config directory
mkdir config

# Copy the config.toml file to the new config directory.
docker run --runtime=nvidia \
    --pid=host \
    --rm \
    --init \
    -u `id -u`:`id -g` \
    -v `pwd`/config:/config \
    --entrypoint bash \
    harbor.h2o.ai/h2oai/dai-centos7-x86_64:1.9.0-cuda10.0.23
    -c "cp /etc/dai/config.toml /config"
```

2. Edit the desired variables in the config.toml file. Save your changes when you are done.
3. Start Driverless AI with the DRIVERLESS_AI_CONFIG_FILE environment variable. Make sure this points to the location of the edited config.toml file so that the software finds the configuration file.

```
1 docker run --runtime=nvidia \
2   --pid=host \
3   --init \
4   --rm \
5   --shm-size=256m \
6   -u `id -u`:`id -g` \
7   -p 12345:12345 \
8   -e DRIVERLESS_AI_CONFIG_FILE="/config/config.toml" \
9   -v `pwd`/config:/config \
10  -v `pwd`/data:/data \
11  -v `pwd`/log:/log \
12  -v `pwd`/license:/license \
13  -v `pwd`/tmp:/tmp \
14  harbor.h2o.ai/h2oai/dai-centos7-x86_64:1.9.0-cuda10.0.23
```

Native installs include DEBs, RPMs, and TAR SH installs.

1. Export the Driverless AI config.toml file or add it to `~/.bashrc`. For example:

```
export DRIVERLESS_AI_CONFIG_FILE="/config/config.toml"
```

2. Edit the desired variables in the config.toml file. Save your changes when you are done.
3. Start Driverless AI. Note that the command used to start Driverless AI varies depending on your install type.

12.2 Sample Config.toml File

For reference, below is a copy of the standard config.toml file included with this version of Driverless AI. The sections that follow describe some examples showing how to set different environment variables, data connectors, authentication, and notifications.

```
1 #####
2 #          DRIVERLESS AI CONFIGURATION FILE
3 #
4 #
5 # Comments:
6 # This file is authored in TOML (see https://github.com/toml-lang/toml)
7 #
8 # Config Override Chain
9 # Configuration variables for Driverless AI can be provided in several ways,
10 # the config engine reads and overrides variables in the following order
11 #
12 # 1. h2oai/config/config.toml
13 # [internal not visible to users]
14 #
15 # 2. config.toml
16 # [place file in a folder/mount file in docker container and provide path
17 # in "DRIVERLESS_AI_CONFIG_FILE" environment variable]
18 #
19 # 3. Environment variable
20 # [configuration variables can also be provided as environment variables
21 # they must have the prefix "DRIVERLESS_AI_" followed by
22 # variable name in caps e.g "authentication_method" can be provided as
23 # "DRIVERLESS_AI_AUTHENTICATION_METHOD"]
24 #####
25 #
26 # Whether to allow user to change non-server toml parameters per experiment in expert page.
27 #allow_config_overrides_in_expert_page = true
28 #
29 # Every *.toml file is read from this directory and process the same way as main config file.
30 #user_config_directory = ""
31 #
32 # IP address and port of autoviz process.
33 #vis_server_ip = "127.0.0.1"
34 #
35 # IP and port of autoviz process.
36 #vis_server_port = 12346
37 #
38 # IP address and port of procsy process.
39 #procsy_ip = "127.0.0.1"
40 #
41 # IP address and port of procsy process.
42 #procsy_port = 12347
43 #
44 # IP address and port of H2O instance.
45 #h2o_ip = "127.0.0.1"
```

(continues on next page)

(continued from previous page)

```

46 # IP address and port of H2O instance for use by MLi.
47 #h2o_port = 12348
48
49
50 # Enable h2o recipes server.
51 #enable_h2o_recipes = true
52
53 # URL of H2O instance for use by transformers, models, or scorers.
54 #h2o_recipes_url = "None"
55
56 # IP of H2O instance for use by transformers, models, or scorers.
57 #h2o_recipes_ip = "None"
58
59 # Port of H2O instance for use by transformers, models, or scorers. No other instances must be on that port or on next port.
60 #h2o_recipes_port = 50351
61
62 # Name of H2O instance for use by transformers, models, or scorers.
63 #h2o_recipes_name = "None"
64
65 # Number of threads for H2O instance for use by transformers, models, or scorers. -1 for all.
66 #h2o_recipes_nthreads = 8
67
68 # Log Level of H2O instance for use by transformers, models, or scorers.
69 #h2o_recipes_log_level = "None"
70
71 # Maximum memory size of H2O instance for use by transformers, models, or scorers.
72 #h2o_recipes_max_mem_size = "None"
73
74 # Minimum memory size of H2O instance for use by transformers, models, or scorers.
75 #h2o_recipes_min_mem_size = "None"
76
77 # General user overrides of kwargs dict to pass to h2o.init() for recipe server.
78 #h2o_recipes_kwargs = "{}"
79
80 # Number of trials to give h2o-3 recipe server to start.
81 #h2o_recipes_start_trials = 5
82
83 # Number of seconds to sleep before starting h2o-3 recipe server.
84 #h2o_recipes_start_sleep0 = 1
85
86 # Number of seconds to sleep between trials of starting h2o-3 recipe server.
87 #h2o_recipes_start_sleep = 5
88
89 # Lock source for recipes to a specific github repo.
90 # If True then all custom recipes must come from the repo specified in setting: custom_recipes_git_repo
91 #custom_recipes_lock_to_git_repo = false
92
93 # If custom_recipes_lock_to_git_repo is set to True, only this repo can be used to pull recipes from
94 #custom_recipes_git_repo = "https://github.com/h2oai/driverlessai-recipes"
95
96 # Branch constraint for recipe source repo. Any branch allowed if unset or None
97 #custom_recipes_git_branch = "None"
98
99 #custom_recipes_excluded_filenames_from_repo_download = []
100
101 #allow_old_recipes_use_datadir_as_data_directory = true
102
103 # IP address and port for Driverless AI HTTP server.
104 #ip = "127.0.0.1"
105
106 # IP address and port for Driverless AI HTTP server.
107 #port = 12345
108
109 # A list of two integers indicating the port range to search over, and dynamically find an open port to bind to (e.g., [11111,20000]).
110 #port_range = "[]"
111
112 # File upload limit (default 100GB)
113 #max_file_upload_size = 104857600000
114
115 # Verbosity of logging
116 # 0: quiet (CRITICAL, ERROR, WARNING)
117 # 1: default (CRITICAL, ERROR, WARNING, INFO, DATA)
118 # 2: verbose (CRITICAL, ERROR, WARNING, INFO, DATA, DEBUG)
119 # Affects server and all experiments
120 #log_level = 1
121
122 # Whether to collect relevant server logs (h2oai_server.log, dai.log from systemctl or docker, and h2o log)
123 # Useful for when sending logs to H2O.ai
124 #collect_server_logs_in_experiment_logs = false
125
126 # How the long running tasks are scheduled.
127 # multiprocessing: forks the current process immediately.
128 # singlenode: shares the task through redis and needs a worker running.
129 # multinode: same as singlenode and also shares the data through minio
130 # and allows worker to run on the different machine.
131 #
132 #worker_mode = "singlenode"
133
134 # Redis settings
135 #redis_ip = "127.0.0.1"
136
137 # Redis settings
138 #redis_port = 6379
139
140 # Redis database. Each DAI instance running on the redis server should have unique integer.
141 #redis_db = 0
142
143 # Redis settings
144 #main_server_redis_password = "PiWUjvEJSiWu9j0aopOyL5KwqnrKtyWVoZHunqxr"
145
146 # The port that Minio will listen on, this only takes effect if the current system is a multinode main server.
147 #local_minio_port = 9000
148
149 # Location of main server's minio server.

```

(continues on next page)

(continued from previous page)

```

150 #main_server_minio_address = "127.0.0.1:9000"
151
152 # Access key of main server's minio server.
153 #main_server_minio_access_key_id = "GMCSE2K2T3RV6YEHJUYW"
154
155 # Secret access key of main server's minio server.
156 #main_server_minio_secret_access_key = "JFxmXvE/WlAaqwgypxAUFSJZRnDWUaeQciZJUe9H"
157
158 # Name of minio bucket used for file synchronization.
159 #main_server_minio_bucket = "h2oai"
160
161 # Maximum number of local tasks processed at once
162 #worker_local_processors = 32
163
164 # Maximum number of remote tasks processed at once
165 #worker_remote_processors = 2
166
167 # If worker_remote_processors >= 2, factor by which each task reduces threads, used by various packages like datatable, lightgbm, xgboost, etc.
168 #worker_remote_processors_max_threads_reduction_factor = 0.7
169
170 # How often the server should extract results from redis queue in milliseconds.
171 #redis_result_queue_polling_interval = 100
172
173 # For how many seconds worker should wait for main server minio bucket before it fails
174 #main_server_minio_bucket_ping_timeout = 30
175
176 # For how many seconds the worker shouldn't respond to be marked unhealthy.
177 #worker_healthy_response_period = 300
178
179 # Exposes the DriverlessAI base version when enabled.
180 #expose_server_version = true
181
182 # https settings
183 # You can make a self-signed certificate for testing with the following commands:
184 # sudo openssl req -x509 -newkey rsa:4096 -keyout private_key.pem -out cert.pem -days 3650 -nodes -subj '/O=Driverless AI'
185 # sudo chmod dai:dai cert.pem private_key.pem
186 # sudo chmod 600 cert.pem private_key.pem
187 # sudo mv cert.pem private_key.pem /etc/dai
188 #enable_https = false
189
190 # https settings
191 # You can make a self-signed certificate for testing with the following commands:
192 # sudo openssl req -x509 -newkey rsa:4096 -keyout private_key.pem -out cert.pem -days 3650 -nodes -subj '/O=Driverless AI'
193 # sudo chmod dai:dai cert.pem private_key.pem
194 # sudo chmod 600 cert.pem private_key.pem
195 # sudo mv cert.pem private_key.pem /etc/dai
196 #ssl_key_file = "/etc/dai/private_key.pem"
197
198 # https settings
199 # You can make a self-signed certificate for testing with the following commands:
200 # sudo openssl req -x509 -newkey rsa:4096 -keyout private_key.pem -out cert.pem -days 3650 -nodes -subj '/O=Driverless AI'
201 # sudo chmod dai:dai cert.pem private_key.pem
202 # sudo chmod 600 cert.pem private_key.pem
203 # sudo mv cert.pem private_key.pem /etc/dai
204 #ssl_crt_file = "/etc/dai/cert.pem"
205
206 # SSL TLS
207 #ssl_no_ssllv2 = true
208
209 # SSL TLS
210 #ssl_no_ssllv3 = true
211
212 # SSL TLS
213 #ssl_no_tisv1 = true
214
215 # SSL TLS
216 #ssl_no_tisv1_1 = true
217
218 # SSL TLS
219 #ssl_no_tisv1_2 = false
220
221 # SSL TLS
222 #ssl_no_tisv1_3 = false
223
224 # https settings
225 # Sets the client verification mode.
226 # CERT_NONE: Client does not need to provide the certificate and if it does any
227 # verification errors are ignored.
228 # CERT_OPTIONAL: Client does not need to provide the certificate and if it does
229 # certificate is verified against set up CA chains.
230 # CERT_REQUIRED: Client needs to provide a certificate and certificate is
231 # verified.
232 # You'll need to set 'ssl_client_key_file' and 'ssl_client_crt_file'
233 # When this mode is selected for Driverless to be able to verify
234 # it's own callback requests.
235 #
236 #ssl_client_verify_mode = "CERT_NONE"
237
238 # https settings
239 # Path to the Certification Authority certificate file. This certificate will be
240 # used when to verify client certificate when client authentication is turned on.
241 # If this is not set, clients are verified using default system certificates.
242 #
243 #ssl_ca_file = ""
244
245 # https settings
246 # path to the private key that Driverless will use to authenticate itself when
247 # CERT_REQUIRED mode is set.
248 #
249 #ssl_client_key_file = ""
250
251 # https settings
252 # path to the client certificate that Driverless will use to authenticate itself
253 # when CERT_REQUIRED mode is set.

```

(continues on next page)

(continued from previous page)

```

254 #
255 #ssl_client_crt_file = ""
256
257 # Data directory. All application data and files related datasets and
258 # experiments are stored in this directory.
259 #data_directory = "./tmp"
260
261 # Whether to have all user content isolated into a directory for each user.
262 # If set to False, all users content is common to single directory,
263 # recipes are shared, and brain folder for restart/refit is shared.
264 # If set to True, each user has separate folder for all user tasks,
265 # recipes are isolated to each user, and brain folder for restart/refit is
266 # only for the specific user.
267 # Migration from False to True or back to False is allowed for
268 # all experiment content accessible by GUI or python client,
269 # all recipes, and starting experiment with same parameters, restart, or refit.
270 # However, if switch to per-user mode, the common brain folder is no longer used.
271 #
272 #per_user_directories = true
273
274 # Whether to run quick performance benchmark at start of application
275 #enable_quick_benchmark = true
276
277 # Whether to run extended performance benchmark at start of application
278 #enable_extended_benchmark = false
279
280 # Scaling factor for number of rows for extended performance benchmark. For rigorous performance benchmarking,
281 # values of 1 or larger are recommended.
282 #extended_benchmark_scale_num_rows = 0.1
283
284 # Whether to run quick startup checks at start of application
285 #enable_startup_checks = true
286
287 # Whether to opt in to usage statistics and bug reporting
288 #usage_stats_opt_in = false
289
290 # authentication_method
291 # unvalidated : Accepts user id and password. Does not validate password.
292 # none: Does not ask for user id or password. Authenticated as admin.
293 # openid: Users OpenID Connect provider for authentication. See additional OpenID settings below.
294 # pam: Accepts user id and password. Validates user with operating system.
295 # ldap: Accepts user id and password. Validates against an ldap server. Look
296 # for additional settings under LDAP settings.
297 # local: Accepts a user id and password. Validated against an htpasswd file provided in local_htpasswd_file.
298 # ibm_spectrum_conductor: Authenticate with IBM conductor auth api.
299 # tls_certificate: Authenticate with Driverless by providing a TLS certificate.
300 #
301 #authentication_method = "unvalidated"
302
303 # Additional authentication methods that will be enabled for the clients.Login forms for each method will be available on the' /login/<authentication_
304 #method>' path.Comma separated list.
305 #additional_authentication_methods = "[]"
306
307 # default amount of time in hours before we force user to login again (if not provided by authentication_method)
308 #authentication_default_timeout_hours = 72
309
310 # OpenID Connect Settings:
311 # Refer to OpenID Connect Basic Client Implementation Guide for details on how OpenID authentication flow works
312 # https://openid.net/specs/openid-connect-basic-1_0.html
313 # base server uri to the OpenID Provider server (ex: https://oidp.ourdomain.com
314 #auth_openid_provider_base_uri = ""
315
316 # uri to pull OpenID config data from (you can extract most of required OpenID config from this url)
317 # usually located at: /auth/realm/master/.well-known/openid-configuration
318 #auth_openid_configuration_uri = ""
319
320 # uri to start authentication flow
321 #auth_openid_auth_uri = ""
322
323 # uri to make request for token after callback from OpenID server was received
324 #auth_openid_token_uri = ""
325
326 # uri to get user information once access_token has been acquired (ex: list of groups user belongs to will be provided here)
327 #auth_openid_userinfo_uri = ""
328
329 # uri to logout user
330 #auth_openid_logout_uri = ""
331
332 # callback uri that OpenID provider will use to send 'authentication_code'
333 # This is OpenID callback endpoint in Driverless AI. Most OpenID providers need this to be HTTPS.
334 # (ex. https://driverless.ourdomain.com/openid/callback)
335 #auth_openid_redirect_uri = ""
336
337 # OAuth2 grant type (usually authorization_code for OpenID, can be access_token also)
338 #auth_openid_grant_type = ""
339
340 # OAuth2 response type (usually code)
341 #auth_openid_response_type = ""
342
343 # Client ID registered with OpenID provider
344 #auth_openid_client_id = ""
345
346 # Client secret provided by OpenID provider when registering Client ID
347 #auth_openid_client_secret = ""
348
349 # Scope of info (usually openid). Can be list of more than one, space delimited, possible
350 # values listed at https://openid.net/specs/openid-connect-basic-1_0.html#Scopes
351 #auth_openid_scope = ""
352
353 # What key in user_info json should we check to authorize user
354 #auth_openid_userinfo_auth_key = ""
355
356 # What value should the key have in user_info json in order to authorize user
357 #auth_openid_userinfo_auth_value = ""

```

(continues on next page)

(continued from previous page)

```

357 # Key that specifies username in user_info json (we will use the value of this key as username in Driverless AI)
358 #auth_openid_userinfo_username_key = ""
359
360 # Quote method from urllib.parse used to encode payload dict in Authentication Request
361 #auth_openid_urlencode_quote_via = "quote"
362
363 # Key in Token Response JSON that holds the value for access token expiry
364 #auth_openid_access_token_expiry_key = "expires_in"
365
366 # Key in Token Response JSON that holds the value for refresh token expiry
367 #auth_openid_refresh_token_expiry_key = "refresh_expires_in"
368
369 # Expiration time in seconds for access token
370 #auth_openid_token_expiration_secs = 3600
371
372 # Enables advanced matching for OpenID Connect authentication.
373 # When enabled ObjectPath (<http://objectpath.org/>) expression is used to
374 # evaluate the user identity.
375 #
376 #auth_openid_use_objectpath_match = false
377
378 # ObjectPath (<http://objectpath.org/>) expression that will be used
379 # to evaluate whether user is allowed to login into Driverless.
380 # Any expression that evaluates to True means user is allowed to log in.
381 # Examples:
382 # Simple claim equality: '$.our_claim is "our_value"'
383 # List of claims contains required value: 'expected_role' in $.roles'
384 #
385 #auth_openid_use_objectpath_expression = ""
386
387 # Sets token introspection URL for OpenID Connect authentication.(needs to be an absolute URL)
388 #auth_openid_token_introspection_url = ""
389
390 # Enables option to use Bearer token for authentication with the RPC endpoint.
391 #api_token_introspection_enabled = false
392
393 # Sets the method that is used to introspect the bearer token.
394 # OAUTH2_TOKEN_INTROSPECTION: Uses OAuth 2.0 Token Introspection (RPC 7662)
395 # endpoint to introspect the bearer token.
396 # This useful when 'openid' is used as the authentication method.
397 # Uses 'auth_openid_client_id' and 'auth_openid_client_secret' and to
398 # authenticate with the authorization server and
399 # 'auth_openid_token_introspection_url' to perform the introspection.
400 #
401 #api_token_introspection_method = "OAUTH2_TOKEN_INTROSPECTION"
402
403 # Sets the minimum of the scopes that the access token needs to have
404 # in order to pass the introspection. Space separated./
405 # This is passed to the introspection endpoint and also verified after response
406 # for the servers that don't enforce scopes.
407 # Keeping this empty turns any the verification off.
408 #
409 #api_token_oauth2_scopes = ""
410
411 # Which field of the response returned by the token introspection endpoint should be used as a username.
412 #api_token_oauth2_username_field_name = "username"
413
414 # Enables the option to initiate a PKCE flow from the UI in order to obtain tokens usable with Driverless clients
415 #oauth2_client_tokens_enabled = false
416
417 # Sets up client id that will be used in the OAuth 2.0 Authorization Code Flow to obtain the tokens. Client needs to be public and be able to use PKCE ↵
418 #with S256 code challenge.
419 #oauth2_client_tokens_client_id = ""
420
421 # Sets up the absolute url to the authorize endpoint.
422 #oauth2_client_tokens_authorize_url = ""
423
424 # Sets up the absolute url to the token endpoint.
425 #oauth2_client_tokens_token_url = ""
426
427 # Sets up the absolute url to the token introspection endpoint. It's displayed in the UI so that clients can inspect the token expiration.
428 #oauth2_client_tokens_introspection_url = ""
429
430 # Sets up the absolute to the redirect url where Driverless handles the redirect part of the Authorization Code Flow. this <Driverless base url>/oauth2/
431 ↵client_token
432 #oauth2_client_tokens_redirect_url = ""
433
434 # Sets up the scope for the requested tokens. Space separated list.
435 #oauth2_client_tokens_scope = "openid profile ai.h2o.storage"
436
437 # ldap server domain or ip
438 #ldap_server = ""
439
440 # ldap server port
441 #ldap_port = ""
442
443 # Complete DN of the LDAP bind user
444 #ldap_bind_dn = ""
445
446 # Password for the LDAP bind
447 #ldap_bind_password = ""
448
449 # Provide Cert file location
450 #ldap_tls_file = ""
451
452 # use true to use ssl or false
453 #ldap_use_ssl = false
454
455 # the location in the DIT where the search will start
456 #ldap_search_base = ""
457
458 # A string that describes what you are searching for. You can use Python substitution to have this constructed dynamically.(only {{DAI_USERNAME}} is ↵
459 supported)

```

(continues on next page)

(continued from previous page)

```

458 #ldap_search_filter = ""
459
460 # ldap attributes to return from search
461 #ldap_search_attributes = ""
462
463 # specify key to find user name
464 #ldap_user_name_attribute = ""
465
466 # When using this recipe, needs to be set to "1"
467 #ldap_recipe = "0"
468
469 # Deprecated do not use
470 #ldap_user_prefix = ""
471
472 # Deprecated, Use ldap_bind_dn
473 #ldap_search_user_id = ""
474
475 # Deprecated, ldap_bind_password
476 #ldap_search_password = ""
477
478 # Deprecated, use ldap_search_base instead
479 #ldap_ou_dn = ""
480
481 # Deprecated, use ldap_base_dn
482 #ldap_dc = ""
483
484 # Deprecated, use ldap_search_base
485 #ldap_base_dn = ""
486
487 # Deprecated, use ldap_search_filter
488 #ldap_base_filter = ""
489
490 # Path to the CRL file that will be used to verify client certificate.
491 #auth_tls_crl_file = ""
492
493 # What field of the subject would be used as source for username or other values used for further validation.
494 #auth_tls_subject_field = "CN"
495
496 # Regular expression that will be used to parse subject field to obtain the username or other values used for further validation.
497 #auth_tls_field_parse_regexp = "(?P<username>.*)"
498
499 # Sets up the way how user identity would be obtained
500 # REGEXP_ONLY: Will use 'auth_tls_subject_field' and 'auth_tls_field_parse_regexp'
501 # to extract the username from the client certificate.
502 # LDAP_LOOKUP: Will use LDAP server to lookup for the username.
503 # 'auth_tls_ldap_server', 'auth_tls_ldap_port',
504 # 'auth_tls_ldap_use_ssl', 'auth_tls_ldap_tls_file',
505 # 'auth_tls_ldap_bind_dn', 'auth_tls_ldap_bind_password'
506 # options are used to establish the connection with the LDAP server.
507 # 'auth_tls_subject_field' and 'auth_tls_field_parse_regexp'
508 # options are used to parse the certificate.
509 # 'auth_tls_ldap_search_base', 'auth_tls_ldap_search_filter', and
510 # 'auth_tls_ldap_username_attribute' options are used to do the
511 # lookup.
512 #
513 #auth_tls_user_lookup = "REGEXP_ONLY"
514
515 # Hostname or IP address of the LDAP server used with LDAP_LOOKUP with 'tls_certificate' authentication method.
516 #auth_tls_ldap_server = ""
517
518 # Port of the LDAP server used with LDAP_LOOKUP with 'tls_certificate' authentication method.
519 #auth_tls_ldap_port = ""
520
521 # Whether to SSL to when connecting to the LDAP server used with LDAP_LOOKUP with 'tls_certificate' authentication method.
522 #auth_tls_ldap_use_ssl = false
523
524 # Path to the SSL certificate used with LDAP_LOOKUP with 'tls_certificate' authentication method.
525 #auth_tls_ldap_tls_file = ""
526
527 # Complete DN of the LDAP bind user used with LDAP_LOOKUP with 'tls_certificate' authentication method.
528 #auth_tls_ldap_bind_dn = ""
529
530 # Password for the LDAP bind user used with LDAP_LOOKUP with 'tls_certificate' authentication method.
531 #auth_tls_ldap_bind_password = ""
532
533 # Location in the DIT where the search will start used with LDAP_LOOKUP with 'tls_certificate' authentication method.
534 #auth_tls_ldap_search_base = ""
535
536 # LDAP filter that will be used to lookup for the user
537 # with LDAP_LOOKUP with 'tls_certificate' authentication method.
538 # Can be built dynamically using the named capturing groups from the
539 # 'auth_tls_field_parse_regexp' for substitution.
540
541 # Example:
542 # auth_tls_field_parse_regexp = "w+ (?P<id>\d+)"
543 # auth_tls_ldap_search_filter = "(&(objectClass=person)(id={{id}}))"
544 #
545 #auth_tls_ldap_search_filter = ""
546
547 # Specified what LDAP record attribute will be used as username with LDAP_LOOKUP with 'tls_certificate' authentication method.
548 #auth_tls_ldap_username_attribute = ""
549
550 # Sets optional additional lookup filter that is performed after the
551 # user is found. This can be used for example to check whether the user is member of
552 # particular group.
553 # Filter can be built dynamically from the attributes returned by the lookup.
554 # Authorization fails when search does not return any entry. If one or more
555 # entries are returned authorization succeeds.
556 # Example:
557 # auth_tls_field_parse_regexp = "w+ (?P<id>\d+)"
558 # ldap_search_filter = "(&(objectClass=person)(id={{id}}))"
559 # auth_tls_ldap_authorization_lookup_filter = "(&(objectClass=group)(member=uid={{uid}}),dc=example,dc=com)"
560 # If this option is empty no additional lookup is done and just a successful user
561 # lookup is enough to authorize the user.
562 #

```

(continues on next page)

(continued from previous page)

```

562 #auth_tls_ldap_authorization_lookup_filter = ""
563
564 # Base DN where to start the Authorization lookup. Used when 'auth_tls_ldap_authorization_lookup_filter' is set.
565 #auth_tls_ldap_authorization_search_base = ""
566
567 # Local password file
568 # Generating a htpasswd file: see syntax below
569 # htpasswd -B '<location_to_place_htpasswd_file>' '<username>'
570 # note: -B forces use of bcrypt, a secure encryption method
571 #local_htpasswd_file = ""
572
573 # Supported file formats (file name endings must match for files to show up in file browser)
574 #supported_file_types = "csv, tsv, txt, dat, tgz, gz, bz2, zip, xz, xls, xlsx, jay, feather, bin, arff, parquet, pk1, orc"
575
576 # Supported file formats of data recipe files (file name endings must match for files to show up in file browser)
577 #recipe_supported_file_types = "py, pyc"
578
579 # By default, only supported file types (based on the file extensions listed above) will be listed for import into DAI
580 # Some data pipelines generate parquet files without any extensions. Enabling the below option will cause files
581 # without extensions to be listed in the file import dialog.
582 # DAI will import files without extensions as parquet files; if cannot be imported, an error is generated
583 #
584 #list_files_without_extensions = false
585
586 # File System Support
587 # upload : standard upload feature
588 # file : local file system/server file system
589 # hdfs : Hadoop file system, remember to configure the HDFS config folder path and keytab below
590 # dtap : Blue Data Tap file system, remember to configure the DTap section below
591 # s3 : Amazon S3, optionally configure secret and access key below
592 # gcs : Google Cloud Storage, remember to configure gcs_path_to_service_account_json below
593 # gbq : Google Big Query, remember to configure gcs_path_to_service_account_json below
594 # minio : Minio Cloud Storage, remember to configure secret and access key below
595 # snow : Snowflake Data Warehouse, remember to configure Snowflake credentials below (account name, username, password)
596 # kdb : KDB+ Time Series Database, remember to configure KDB credentials below (hostname and port, optionally: username, password, classpath, and jvm_args)
597 # azrs : Azure Blob Storage, remember to configure Azure credentials below (account name, account key)
598 # jdbc: JDBC Connector, remember to configure JDBC below. (jdbc_app_configs)
599 # hive: Hive Connector, remember to configure Hive below. (hive_app_configs)
600 # recipe_file: Custom recipe file upload
601 # recipe_url: Custom recipe upload via url
602 #
603 #enabled_file_systems = "upload, file, hdfs, s3, recipe_file, recipe_url"
604
605 #max_files_listed = 100
606
607 # do_not_log_list : add configurations that you do not wish to be recorded in logs here
608 #do_not_log_list = ["!artifacts_git_password", 'auth_openid_client_id', 'auth_openid_client_secret', 'auth_openid_userinfo_auth_key', 'auth_openid_userinfo_
609 #<auth_value>', 'auth_openid_userinfo_username_key', 'auth_tls_ldap_bind_password', 'aws_access_key_id', 'aws_secret_access_key', 'azure_blob_account_key',
610 #<azure_blob_account_name>', 'azure_connection_string', 'deployment_aws_access_key_id', 'deployment_aws_secret_access_key', 'gcs_path_to_service_account_
611 #<json>', 'kaggle_key', 'kaggle_username', 'kdb_password', 'kdb_user', 'ldap_bind_password', 'ldap_search_password', 'local_htpasswd_file', 'main_server_
612 #<minio_access_key_id>', 'main_server_minio_secret_access_key', 'main_server_redis_password', 'minio_access_key_id', 'minio_endpoint_url', 'minio_secret_
613 #<access_key>', 'snowflake_account', 'snowflake_password', 'snowflake_url', 'snowflake_user']"
614
615 # Allow using browser localstorage, to improve UX.
616 #allow_localstorage = true
617
618 # Allow original dataset columns to be present in downloaded predictions CSV
619 #allow_orig_cols_in_predictions = true
620
621 #mli_lime_method = "k-LIME"
622
623 #mli_use_raw_features = true
624
625 #mli_sample = true
626
627 #mli_dt_tree_depth = 3
628
629 #mli_vars_to_pdp = 10
630
631 #mli_nfolds = 3
632
633 #mli_qbin_count = 0
634
635 # If the experiment is not done after this many minutes, stop feature engineering and model tuning as soon as possible and proceed with building the final
636 #<modeling pipeline and deployment artifacts, independent of model score convergence or pre-determined number of iterations. Only active is not in
637 #<reproducible mode. Depending on the data and experiment settings, overall experiment runtime can differ significantly from this setting.
638 #max_runtime_minutes = 1440
639
640 # If the experiment is not done after this many minutes, push the abort button. Preserves experiment artifacts made so far for summary and log zip files,
641 #<but further artifacts are made.
642 #max_runtime_minutes_until_abort = 10080
643
644 # Recipe type
645 # Recipes override any GUI settings
646 # 'auto' : all models and features automatically determined by experiment settings, toml settings, and feature_engineering_effort
647 # 'compliant' : like 'auto' except:
648 # * interpretability=10 (to avoid complexity, overrides GUI or python client chose for interpretability)
649 # * enable_glm='on' (rest 'off', to avoid complexity and be compatible with algorithms supported by MLI)
650 # * fixed_ensemble_level=0: Don't use any ensemble (to avoid complexity)
651 # * feature_brain_level=0: No feature brain used (to ensure every restart is identical)
652 # * max_feature_interaction_depth=1: interaction depth is set to 1 (no multi-feature interactions to avoid complexity)
653 # * target_transformer='identity': for regression (to avoid complexity)
654 # * check_distribution_shift_drop='off': Don't use distribution shift between train, valid, and test to drop features (bit risky without fine-tuning)
655 # * 'kaggle' : like 'auto' except:
656 # * external_validation_set is concatenated with train set, with target marked as missing
657 # * test set is concatenated with train set, with target marked as missing
658 # * transformers that do not use the target are allowed to fit_transform across entire train + validation + test
659 # * several config toml expert options open-up limits (e.g. more numerics are treated as categoricals)
660 # Note: If plentiful memory, can:
661 # * choose kaggle mode and then change fixed_feature_interaction_depth to large negative number,
662 # otherwise default number of features given to transformer is limited to 50 by default
663 # * choose mutation_mode = "full", so even more types are transformations are done at once per transformer
664 # 'nlp_model' : Only enables NLP models that process pure text
665 # 'nlp_transformer' : Only enables NLP transformers that process pure text, while any model type is allowed

```

(continues on next page)

(continued from previous page)

```

658 # 'image_model' : Only enables Image models that process pure images
659 # 'image_transformer' : Only enables Image transformers that process pure images, while any model type is allowed
660 #
661 #recipe = "auto"
662
663 # Internal helper to allow memory of if changed recipe
664 #last_recipe = ""
665
666 # Whether to enable genetic algorithm for selection and hyper-parameter tuning of features and models.
667 # If disabled ('off'), will go directly to final pipeline training (using default feature engineering and
668 # feature selection). 'auto' is same as 'on' unless pure NLP or Image experiment.
669 #
670 #enable_genetic_algorithm = "auto"
671
672 # How much effort to spend on feature engineering (0...10)
673 # Heuristic combination of various developer-level toml parameters
674 # 0 : keep only numeric features, only model tuning during evolution
675 # 1 : keep only numeric features and frequency-encoded categorical, only model tuning during evolution
676 # 2 : Like #1 but instead just no Text features. Some feature tuning before evolution.
677 # 3 : Like #5 but only tuning during evolution. Mixed tuning of features and model parameters.
678 # 4 : Like #5, but slightly more focused on model tuning
679 # 5 : Default. Balanced feature-model tuning
680 # 6-7 : Like #5, but slightly more focused on feature engineering
681 # 8 : Like #6-7, but even more focused on feature engineering with high feature generation rate, no feature dropping even if high interpretability
682 # 9-10: Like #8, but no model tuning during feature evolution
683 #feature_engineering_effort = 5
684
685 # Whether to enable train/valid and train/test distribution shift detection ('auto'/'on'/'off')
686 #check_distribution_shift = "auto"
687
688 # Whether to drop high-shift features ('auto'/'on'/'off'). Auto disables for time series.
689 #check_distribution_shift_drop = "auto"
690
691 # If distribution shift detection is enabled, drop features (except ID, text, date/datetime, time, weight) for
692 # which shift AUC, GINI, or Spearman correlation is above this value
693 # (e.g. AUC of a binary classifier that predicts whether given feature value
694 # belongs to train or test data)
695 #drop_features_distribution_shift_threshold_auc = 0.999
696
697 # Whether to check leakage for each feature (True/False).
698 # If fold column, this checks leakage without fold column used.
699 #check_leakage = "auto"
700
701 # If leakage detection is enabled,
702 # drop features for which AUC (R2 for regression), GINI,
703 # or Spearman correlation is above this value.
704 # If fold column present, features are not dropped,
705 # because leakage test applies without fold column used.
706 #
707 #drop_features_leakage_threshold_auc = 0.999
708
709 # Max number of rows x number of columns to trigger (stratified) sampling for leakage checks
710 #leakage_max_data_size = 10000000
711
712 # Whether to create the Python scoring pipeline at the end of each experiment.
713 #make_python_scoring_pipeline = "auto"
714
715 # Whether to create the MOJO scoring pipeline at the end of each experiment. If set to "auto", will attempt to
716 # create it if possible (without dropping capabilities). If set to "on", might need to drop some models,
717 # transformers or custom recipes.
718 #make_mojo_scoring_pipeline = "auto"
719
720 # Whether to attempt to reduce the size of the MOJO scoring pipeline. A smaller MOJO will also lead to
721 # less memory footprint during scoring. It is achieved by reducing some other settings like interaction depth, and
722 # hence can affect the predictive accuracy of the model.
723 #
724 #reduce_mojo_size = false
725
726 # Whether to create the pipeline visualization at the end of each experiment.
727 #make_pipeline_visualization = "auto"
728
729 # Whether to create the experiment Autoreport after end of experiment.
730 #
731 #make_autoreport = true
732
733 # If config.toml overrides are set by env vars, and they differ from what the experiment's env looked like when it was trained, then unexpected
734 # consequences can occur. Enable this only to override certain well-controlled settings like the port for H2O-3 custom recipe server.
735 #pass_env_to_deprecated_python_scoring = false
736
737 # Whether to measure the MOJO scoring latency at the time of MOJO creation.
738 #benchmark_mojo_latency = "auto"
739
740 # Max size of pipeline.mojo file (in MB) for automatic mode of MOJO scoring latency measurement
741 #benchmark_mojo_latency_auto_size_limit = 100
742
743 # If MOJO creation times out at end of experiment, can still make MOJO from the GUI or from the R/Py clients (timeout doesn't apply there).
744 #mojo_building_timeout = 1800.0
745
746 # If MOJO creation is too slow, increase this value. Higher values can finish faster, but use more memory.
747 # If MOJO creation fails due to an out-of-memory error, reduce this value to 1.
748 # Set to -1 for all physical cores.
749 #
750 #mojo_building_parallelism = -1
751
752 # Max number of CPU cores to use per experiment. Set to <= 0 to use all cores.
753 # One can also set environment variable 'OMP_NUM_THREADS' to number of cores to use for OpenMP
754 # (e.g., in bash: 'export OMP_NUM_THREADS=32' and 'export OPENBLAS_NUM_THREADS=32').
755 #max_cores = 0
756
757 # Max number of CPU cores to use across all of DAI experiments and tasks.
758 # -1 is all available, with stall_subprocess_submission_dai_fork_threshold_count=0 means restricted to core count.
759 #
760 #max_cores_dai = -1

```

(continues on next page)

(continued from previous page)

```

761 # Stall submission of tasks if total DAI fork count exceeds count (-1 to disable, 0 for automatic of max_cores_dai)
762 #stall_subprocess_submission_dai_fork_threshold_count = 0
763
764 # Stall submission of tasks if system memory available is less than this threshold in percent (set to 0 to disable).
765 # Above this threshold, the number of workers in any pool of workers is linearly reduced down to 1 once hitting this threshold.
766 #
767 #stall_subprocess_submission_mem_threshold_pct = 2
768
769 # Whether to set automatic number of cores by physical (True) or logical (False) count.
770 # Using all logical cores can lead to poor performance due to cache thrashing.
771 #max_cores_by_physical = true
772
773 # Absolute limit to core count
774 #max_cores_limit = 100
775
776 # Control maximum number of cores to use for a model's fit call (0 = all physical cores >= 1 that count)
777 #max_fit_cores = 10
778
779 # Control maximum number of cores to use for a model's predict call (0 = all physical cores >= 1 that count)
780 #max_predict_cores = 0
781
782 # Control maximum number of cores to use for a model's transform and predict call when doing operations inside DAI-MLI GUI and R/Py client (0 = all physical cores >= 1 that count)
783 #max_predict_cores_in_dai = 4
784
785 # Control number of workers used in CPU mode for tuning (0 = socket count -1 = all physical cores >= 1 that count). More workers will be more parallel, but models learn less from each other.
786 #batch_cpu_tuning_max_workers = 0
787
788 # Control number of workers used in CPU mode for training (0 = socket count -1 = all physical cores >= 1 that count)
789 #cpu_max_workers = 0
790
791 # Number of GPUs to use per experiment for training task. Set to -1 for all GPUs.
792 # An experiment will generate many different models.
793 # Currently num_gpus_per_experiment!=1 disables GPU locking, so is only recommended for single experiments and single users.
794 # Ignored if GPUs disabled or no GPUs on system.
795 # More info at: https://github.com/NVIDIA/nvidia-docker/wiki/nvidia-docker#gpu-isolation
796 #num_gpus_per_experiment = -1
797
798 # Number of CPU cores per GPU. Limits number of GPUs in order to have sufficient cores per GPU.
799 # Set to -1 to disable.
800 #min_num_cores_per_gpu = 2
801
802 # Number of GPUs to use per model training task. Set to -1 for all GPUs.
803 # For example, when this is set to -1 and there are 4 GPUs available, all of them can be used for the training of a single model.
804 # Currently num_gpus_per_model!=1 disables GPU locking, so is only recommended for single experiments and single users.
805 # Ignored if GPUs disabled or no GPUs on system.
806 # More info at: https://github.com/NVIDIA/nvidia-docker/wiki/nvidia-docker#gpu-isolation
807 #num_gpus_per_model = 1
808
809 # Number of GPUs to use for predict for models and transform for transformers when running outside of fit/fit_transform.
810 # If predict/transform are called in same process as fit/fit_transform, number of GPUs will match,
811 # while new processes will use this count for number of GPUs for applicable models/transformers.
812 # Exception: Tensorflow and pytorch models/transformers predict on GPU always if GPUs exist.
813 #
814 #num_gpus_for_prediction = 0
815
816 # Kaggle username for automatic submission and scoring of test set predictions.
817 # See https://github.com/Kaggle/kaggle-api#api-credentials for details on how to obtain Kaggle API credentials",
818 #
819 #kaggle_username = ""
820
821 # Kaggle key for automatic submission and scoring of test set predictions.
822 # See https://github.com/Kaggle/kaggle-api#api-credentials for details on how to obtain Kaggle API credentials",
823 #
824 #kaggle_key = ""
825
826 # Max. number of seconds to wait for Kaggle API call to return scores for given predictions
827 #kaggle_timeout = 120
828
829 #kaggle_keep_submission = false
830
831
832 # If provided, can extend the list to arbitrary and potentially future Kaggle competitions to make submissions for. Only used if kaggle_key and kaggle_username are provided.
833 # Provide a quoted comma-separated list of tuples (target column name, number of test rows, competition, metric) like this:
834 # kaggle_competitions='("target", 200000, "santander-customer-transaction-prediction", "AUC"), ("TARGET", 75818, "santander-customer-satisfaction", "AUC")'
835 #
836 #kaggle_competitions = ""
837
838 # Expected maximum number of forks, used to ensure dt doesn't overload system. For actual use beyond this value, system will start to have slow-down issues
839 #assumed_simultaneous_dt_forks_munging = 3
840
841 # Expected maximum number of forks by computing statistics during ingestion, used to ensure dt doesn't overload system
842 #assumed_simultaneous_dt_forks_stats_openblas = 1
843
844 # Expected maximum of threads for dt no matter if many more cores
845 #max_max_dt_threads_munging = 4
846
847 # Expected maximum of threads for dt no matter if many more cores
848 #max_max_dt_threads_stats_openblas = 8
849
850 # Expected maximum of threads for dt no matter if many more cores
851 #max_max_dt_threads_readwrite = 4
852
853 # Minimum number of threads for datatable (and OpenMP) during data munging (per process).
854 # datatable is the main data munging tool used within Driverless ai (source :
855 # https://github.com/h2oai/datatable)
856 #min_dt_threads_munging = 1
857
858 # Like min_datatable (and OpenMP)_threads_munging but for final pipeline munging
859 #min_dt_threads_final_munging = 1
860
861
862

```

(continues on next page)

(continued from previous page)

```

863 # Maximum number of threads for datatable during data munging (per process) (0 = all, -1 = auto).
864 #max_dt_threads_munging = -1
865
866 # Maximum number of threads for datatable during data reading and writing (per process) (0 = all, -1 = auto).
867 #max_dt_threads_readwrite = -1
868
869 # Maximum number of threads for datatable stats and openblas (per process) (0 = all, -1 = auto).
870 #max_dt_threads_stats_openblas = -1
871
872 # Maximum number of threads for datatable during TS properties preview panel computations.
873 #max_dt_threads_do_timeseries_split_suggestion = 1
874
875 # Delimiter/Separator to use when parsing tabular text files like CSV. Automatic if empty. Must be provided at system start.
876 #datatable_separator = ""
877
878 # Which gpu_id to start with
879 # If using CUDA_VISIBLE_DEVICES=... to control GPUs (preferred method), gpu_id=0 is the
880 # first in that restricted list of devices.
881 # E.g. if CUDA_VISIBLE_DEVICES='4,5' then gpu_id_start=0 will refer to the
882 # device #4.
883 # E.g. from expert mode, to run 2 experiments, each on a distinct GPU out of 2 GPUs:
884 # Experiment#1: num_gpus_per_model=1, num_gpus_per_experiment=1, gpu_id_start=0
885 # Experiment#2: num_gpus_per_model=1, num_gpus_per_experiment=1, gpu_id_start=1
886 # E.g. from expert mode, to run 2 experiments, each on a distinct GPU out of 8 GPUs:
887 # Experiment#1: num_gpus_per_model=1, num_gpus_per_experiment=4, gpu_id_start=0
888 # Experiment#2: num_gpus_per_model=1, num_gpus_per_experiment=4, gpu_id_start=4
889 # E.g. Like just above, but now run on all 4 GPUs/model
890 # Experiment#1: num_gpus_per_model=4, num_gpus_per_experiment=1, gpu_id_start=0
891 # Experiment#2: num_gpus_per_model=4, num_gpus_per_experiment=4, gpu_id_start=4
892 # If num_gpus_per_model!=1, global GPU locking is disabled
893 # (because underlying algorithms don't support arbitrary gpu ids, only sequential ids),
894 # so must setup above correctly to avoid overlap across all experiments by all users
895 # More info at: https://github.com/NVIDIA/nvidia-docker/wiki/nvidia-docker#gpu-isolation
896 # Note that gpu selection does not wrap, so gpu_id_start + num_gpus_per_model must be less than number of visible gpus
897 #gpu_id_start = 0
898
899 # Maximum number of workers for Driverless AI server pool (only 1 needed currently)
900 #max_workers = 1
901
902 # Period (in seconds) of ping by Driverless AI server to each experiment
903 # (in order to get logger info like disk space and memory usage).
904 # 0 means don't print anything.
905 #ping_period = 60
906
907 # Whether to enable ping of system status during DAI data ingestion.
908 #ping_load_data_file = false
909
910 # Period between checking DAI status.
911 #ping_sleep_period = 1
912
913 # Minimum amount of disk space in GB needed to run experiments.
914 # Experiments will fail if this limit is crossed.
915 # This limit exists because Driverless AI needs to generate data for model training
916 # feature engineering, documentation and other such processes.
917 #disk_limit_gb = 5
918
919 # Minimum amount of disk space in GB needed to before stall forking of new processes during an experiment.
920 #stall_disk_limit_gb = 1
921
922 # Minimum amount of system memory in GB needed to start experiments.
923 # Similarly with disk space, a certain amount of system memory is needed to run some basic
924 # operations.
925 #memory_limit_gb = 5
926
927 # Minimum number of rows needed to run experiments (values lower than 100 might not work).
928 # A minimum threshold is set to ensure there is enough data to create a statistically
929 # reliable model and avoid other small-data related failures.
930 #min_num_rows = 100
931
932 # Minimum required number of rows (in the training data) for each class label for classification problems.
933 #min_rows_per_class = 5
934
935 # Minimum required number of rows for each split when generating validation samples.
936 #min_rows_per_split = 5
937
938 # Precision of how data is stored
939 # 'float32' best for speed, 'float64' best for accuracy or very large input values
940 # 'float32' allows numbers up to about +-3E38 with relative error of about 1E-7
941 # 'float64' allows numbers up to about +-1E308 with relative error of about 1E-16
942 # Some calculations, like the GLM standardization, can only handle up to sqrt() of these maximums for data values,
943 # So GLM with 32-bit precision can only handle up to about a value of 1E19 before standardization generates inf values.
944 # If you see "Best individual has invalid score" you may require higher precision.
945 #data_precision = "float32"
946
947 # Precision of most data transformers (same options and notes as data_precision).
948 # Useful for higher precision in transformers with numerous operations that can accumulate error.
949 # Also useful if want faster performance for transformers but otherwise want data stored in high precision.
950 #transformer_precision = "float32"
951
952 # Whether to change ulimit soft limits up to hard limits (for DAI server app, which is not a generic user app).
953 # Prevents resource limit problems in some cases.
954 # Restricted to no more than limit_nofile and limit_nproc for those resources.
955 #ulimit_up_to_hard_limit = true
956
957 #disable_core_files = false
958
959 # number of file limit
960 # Below should be consistent with start-dai.sh
961 #limit_nofile = 65535
962
963 # number of threads limit
964 # Below should be consistent with start-dai.sh
965 #limit_nproc = 16384
966

```

(continues on next page)

(continued from previous page)

```

967 # Level of reproducibility desired (for same data and same inputs).
968 # Only active if 'reproducible' mode is enabled (GUI button enabled or a seed is set from the client API).
969 # Supported levels are:
970 # reproducibility_level = 1 for same experiment results as long as same O/S, same CPU(s) and same GPU(s)
971 # reproducibility_level = 2 for same experiment results as long as same O/S, same CPU architecture and same GPU architecture
972 # reproducibility_level = 3 for same experiment results as long as same O/S, same CPU architecture, not using GPUs
973 # reproducibility_level = 4 for same experiment results as long as same O/S, (best effort)
974 #reproducibility_level = 1
975
976 # Seed for random number generator to make experiments reproducible, to a certain reproducibility level (see above).
977 # Only active if 'reproducible' mode is enabled (GUI button enabled or a seed is set from the client API).
978 #seed = 1234
979
980 # The list of values that should be interpreted as missing values during data import.
981 # This applies to both numeric and string columns. Note that the dataset must be reloaded after applying changes to this config via the expert settings.
982 # Also note that 'nan' is always interpreted as a missing value for numeric columns.
983 #missing_values = ["'", '?', 'None', 'nan', 'NA', 'N/A', 'unknown', 'inf', '-inf', '1.7976931348623157e+308', '-1.7976931348623157e+308']"
984
985 # For tensorflow, what numerical value to give to missing values, where numeric values are standardized.
986 # So 0 is center of distribution, and if Normal distribution then +-5 is 5 standard deviations away from the center.
987 # In many cases, an out of bounds value is a good way to represent missings, but in some cases the mean (0) may be better.
988 #tf_nan_impute_value = -5
989
990 # Internal threshold for number of rows x number of columns to trigger certain statistical
991 # techniques (small data recipe like including one hot encoding for all model types, and smaller learning rate)
992 # to increase model accuracy
993 #statistical_threshold_data_size_small = 100000
994
995 # Internal threshold for number of rows x number of columns to trigger certain statistical
996 # techniques (fewer genes created, removal of high max_depth for tree models, etc.) that can speed up modeling.
997 # Also controls maximum rows used in training final model,
998 # by sampling statistical_threshold_data_size_large / columns number of rows
999 #statistical_threshold_data_size_large = 500000000
1000
1001 # Internal threshold for number of rows x number of columns to trigger sampling for auxiliary data uses,
1002 # like imbalanced data set detection and bootstrap scoring sample size and iterations
1003 #aux_threshold_data_size_large = 10000000
1004
1005 # Internal threshold for number of rows x number of columns to trigger certain changes in performance
1006 # (fewer threads if beyond large value) to help avoid OOM or unnecessary slowdowns
1007 # (fewer threads if lower than small value) to avoid excess forking of tasks
1008 #performance_threshold_data_size_small = 100000
1009
1010 # Internal threshold for number of rows x number of columns to trigger certain changes in performance
1011 # (fewer threads if beyond large value) to help avoid OOM or unnecessary slowdowns
1012 # (fewer threads if lower than small value) to avoid excess forking of tasks
1013 #performance_threshold_data_size_large = 100000000
1014
1015 # Maximum number of columns to start an experiment. This threshold exists to constraint the # complexity and the length of the Driverless AI's processes.
1016 #max_cols = 10000
1017
1018 # Largest number of rows to use for column stats, otherwise sample randomly
1019 #max_rows_col_stats = 1000000
1020
1021 # Whether to obtain permutation feature importance on original features for reporting in logs and file.
1022 #
1023 #orig_features_fs_report = false
1024
1025 # Maximum number of rows when doing permutation feature importance, reduced by (stratified) random sampling.
1026 #
1027 #max_rows_fs = 500000
1028
1029 # How many workers to use for feature selection by permutation for predict phase.
1030 # (0 = auto, > 0: min of DAI value and this value, < 0: exactly negative of this value)
1031 #
1032 #max_workers_fs = 0
1033
1034 # How many workers to use for shift and leakage checks if using LightGBM on CPU.
1035 # (0 = auto, > 0: min of DAI value and this value, < 0: exactly negative of this value)
1036 #
1037 #max_workers_shift_leak = 0
1038
1039 # Maximum number of columns selected out of original set of original columns, using feature selection
1040 # The selection is based upon how well target encoding (or frequency encoding if not available) on categoricals and numerics treated as categoricals
1041 # This is useful to reduce the final model complexity. First the best
1042 # [max_orig_cols_selected] are found through feature selection methods and then
1043 # these features are used in feature evolution (to derive other features) and in modelling.
1044 #max_orig_cols_selected = 10000
1045
1046 # Maximum number of numeric columns selected, above which will do feature selection
1047 # same as above (max_orig_cols_selected) but for numeric columns.
1048 #max_orig_numeric_cols_selected = 10000
1049
1050 # Maximum number of non-numeric columns selected, above which will do feature selection on all features and avoid treating numerical as categorical
1051 # same as above (max_orig_numeric_cols_selected) but for categorical columns.
1052 #max_orig_nonnumeric_cols_selected = 300
1053
1054 # The factor times max_orig_cols_selected, by which column selection is based upon no target encoding and no treating numerical as categorical
1055 # in order to limit performance cost of feature engineering
1056 #max_orig_cols_selected_simple_factor = 2
1057
1058 # Like max_orig_cols_selected, but columns above which add special individual with original columns reduced.
1059 #
1060 #fs_orig_cols_selected = 500
1061
1062 # Like max_orig_numeric_cols_selected, but applicable to special individual with original columns reduced.
1063 # A separate individual in the genetic algorithm is created by doing feature selection by permutation importance on original features.
1064 #
1065 #fs_orig_numeric_cols_selected = 500
1066
1067 # Like max_orig_nonnumeric_cols_selected, but applicable to special individual with original columns reduced.
1068 # A separate individual in the genetic algorithm is created by doing feature selection by permutation importance on original features.
1069 #
1070 #fs_orig_nonnumeric_cols_selected = 200

```

(continues on next page)

(continued from previous page)

```

1071
1072 # Like max_orig_cols_selected_simple_factor, but applicable to special individual with original columns reduced.
1073 #fs_orig_cols_selected_simple_factor = 2
1074
1075 # Maximum allowed fraction of unique values for integer and categorical columns (otherwise will treat column as ID and drop)
1076 #max_relative_cardinality = 0.95
1077
1078 # Maximum allowed number of unique values for integer and categorical columns (otherwise will treat column as ID and drop)
1079 #max_absolute_cardinality = 1000000
1080
1081 # Whether to treat some numerical features as categorical.
1082 # For instance, sometimes an integer column may not represent a numerical feature but
1083 # represent different numerical codes instead.
1084 #num_as_cat = true
1085
1086 # Max number of unique values for integer/real columns to be treated as categoricals (test applies to first statistical_threshold_data_size_small rows only)
1087 #max_int_as_cat_uniques = 50
1088
1089 # Number of folds for models used during the feature engineering process.
1090 # Increasing this will put a lower fraction of data into validation and more into training
1091 # (e.g., num_folds=3 means 67%/33% training/validation splits).
1092 # Actual value will vary for small or big data cases.
1093 #num_folds = 3
1094
1095 # For multiclass problems only. Whether to allow different sets of target classes across (cross-)validation
1096 # fold splits. Especially important when passing a fold column that isn't balanced w.r.t class distribution.
1097 #
1098 #allow_different_classes_across_fold_splits = true
1099
1100 # Accuracy setting equal and above which enables full cross-validation (multiple folds) during feature evolution
1101 # as opposed to only a single holdout split (e.g. 2/3 train and 1/3 validation holdout)
1102 #full_cv_accuracy_switch = 8
1103
1104 # Accuracy setting equal and above which enables stacked ensemble as final model.
1105 # Stacking commences at the end of the feature evolution process..
1106 # It quite often leads to better model performance, but it does increase the complexity
1107 # and execution time of the final model.
1108 #ensemble_accuracy_switch = 5
1109
1110 # Number of fold splits to use for ensemble_level >= 2.
1111 # The ensemble modelling may require predictions to be made on out-of-fold samples
1112 # hence the data needs to be split on different folds to generate these predictions.
1113 # Less folds (like 2 or 3) normally create more stable models, but may be less accurate
1114 # More folds can get to higher accuracy at the expense of more time, but the performance
1115 # may be less stable when the training data is not enough (i.e. higher chance of overfitting).
1116 # Actual value will vary for small or big data cases.
1117 #num_ensemble_folds = 5
1118
1119 # Number of repeats for each fold for all validation
1120 # (modified slightly for small or big data cases)
1121 #fold_reps = 1
1122
1123 #max_num_classes_hard_limit = 10000
1124
1125 # Maximum number of classes to allow for a classification problem.
1126 # High number of classes may make certain processes of Driverless AI time-consuming.
1127 # Memory requirements also increase with higher number of classes
1128 #max_num_classes = 200
1129
1130 # Maximum number of classes to compute ROC and CM for,
1131 # beyond which roc_reduce_type choice for reduction is applied.
1132 # Too many classes can take much longer than model building time.
1133 #max_num_classes_compute_roc = 200
1134
1135 # Maximum number of classes to show in GUI for confusion matrix, showing first max_num_classes_client_and_gui labels.
1136 # The number 20 is default, but beyond 6 classes the diagnostics launched from GUI are visually truncated.
1137 # This will only modify client-GUI launched diagnostics if changed in config.toml and server is restarted,
1138 # while this value can be changed in expert settings to control experiment plots.
1139 #max_num_classes_client_and_gui = 10
1140
1141 # If too many classes when computing roc,
1142 # reduce by "rows" by randomly sampling rows,
1143 # or reduce by truncating classes to no more than max_num_classes_compute_roc.
1144 # If have sufficient rows for class count, can reduce by rows.
1145 #roc_reduce_type = "rows"
1146
1147 #min_roc_sample_size = 1
1148
1149 # Number of actuals vs. predicted data points to use in order to generate in the relevant
1150 # plot/graph which is shown at the right part of the screen within an experiment.
1151 #num_actuals_vs_predicted = 100
1152
1153 # Whether to use H2O.ai brain: the local caching and smart re-use of prior experiments,
1154 # in order to generate more useful features and models for new experiments.
1155 # It can also be used to control checkpointing for experiments that have been paused or interrupted.
1156 # DAI will use H2O.ai brain cache if cache file has
1157 # a) any matching column names and types for a similar experiment type
1158 # b) exactly matches classes
1159 # c) exactly matches class labels
1160 # d) matches basic time series choices
1161 # e) interpretability of cache is equal or lower
1162 # f) main model (booster) is allowed by new experiment.
1163 # Level of brain to use (for chosen level, where higher levels will also do all lower level operations automatically)
1164 # -1 = Don't use any brain cache and don't write any cache
1165 # 0 = Don't use any brain cache but still write cache
1166 # Use case: Want to save model for later use, but want current model to be built without any brain models
1167 # 1 = smart checkpoint from latest best individual model
1168 # Use case: Want to use latest matching model, but match can be loose, so needs caution
1169 # 2 = smart checkpoint from H2O.ai brain cache of individual best models
1170 # Use case: DAI scans through H2O.ai brain cache for best models to restart from
1171 # 3 = smart checkpoint like level #1, but for entire population. Tune only if brain population insufficient size
1172 # (will re-score entire population in single iteration, so appears to take longer to complete first iteration)
1173 # 4 = smart checkpoint like level #2, but for entire population. Tune only if brain population insufficient size
1174 # (will re-score entire population in single iteration, so appears to take longer to complete first iteration)

```

(continues on next page)

(continued from previous page)

```

1175 # 5 = like #4, but will scan over entire brain cache of populations to get best scored individuals
1176 # (can be slower due to brain cache scanning if big cache)
1177 # 1000 + feature_brain_level (above positive values) = use resumed_experiment_id and actual feature_brain_level,
1178 # to use other specific experiment as base for individuals or population,
1179 # instead of sampling from any old experiments
1180 # GUI has 3 options and corresponding settings:
1181 # 1) New Experiment: Uses feature brain level default of 2
1182 # 2) New Model With Same Parameters: Re-uses the same feature brain level as parent experiment
1183 # 3) Restart From Last Checkpoint: Resets feature brain level to 1003 and sets experiment ID to resume from
1184 # (continued genetic algorithm iterations)
1185 # 4) Retrain Final Pipeline: Like Restart but also time=0 so skips any tuning and heads straight to final model
1186 # (assumes had at least one tuning iteration in parent experiment)
1187 # Other use cases:
1188 # a) Restart on different data: Use same column names and fewer or more rows (applicable to 1 - 5)
1189 # b) Re-fit only final pipeline: Like (a), but choose time=1 and feature_brain_level=3 - 5
1190 # c) Restart with more columns: Add columns, so model builds upon old model built from old column names (1 - 5)
1191 # d) Restart with focus on model tuning: Restart, then select feature_engineering_effort = 3 in expert settings
1192 # e) can retrain final model but ignore any original features except those in final pipeline (normal retrain but set brain_add_features_for_new_
1193 #      ↪columns=false)
1194 # Notes:
1195 # 1) In all cases, we first check the resumed experiment id if given, and then the brain cache
1196 # 2) For Restart cases, may want to set min_dai_iterations to non-zero to force delayed early stopping, else may not be enough iterations to find better ↪
1197 #      ↪model.
1198 # 3) A "New model with Same Params" of a Restart will use feature_brain_level=1003 for default Restart mode (revert to 2, or even 0 if want to start a ↪
1199 #      ↪fresh experiment otherwise)
1200 #feature_brain_level = 2
1201 #enable_strict_conflict_key_check_for_brain = true
1202 # Relative number of columns that must match between current reference individual and brain individual.
1203 # 0.0: perfect match
1204 # 1.0: All columns are different, worst match
1205 # e.g. 0.1 implies no more than 10% of columns mismatch between reference set of columns and brain individual.
1206 #
1207 #brain_maximum_diff_score = 0.1
1208 # Maximum number of brain individuals pulled from H2O.ai brain cache for feature_brain_level=1, 2
1209 #max_num_brain_inds = 3
1210 #
1211 # Save feature brain iterations every iter_num % feature_brain_iterations_save_every_iteration == 0, to be able to restart/refit with which_iteration_
1212 #      ↪brain >= 0
1213 # 0 means disable
1214 #feature_brain_save_every_iteration = 0
1215 #
1216 # When doing restart or re-fit type feature_brain_level with resumed_experiment_id, choose which iteration to start from, instead of only last best
1217 # -1 means just use last best
1218 # Usage:
1219 # 1) Run one experiment with feature_brain_iterations_save_every_iteration=1 or some other number
1220 # 2) Identify which iteration brain dump one wants to restart/refit from
1221 # 3) Restart/Refit from original experiment, setting which_iteration_brain to that number in expert settings
1222 # Note: If restart from a tuning iteration, this will pull in entire scored tuning population and use that for feature evolution
1223 #which_iteration_brain = -1
1224 #
1225 # When doing re-fit, if change columns or features, population of individuals used to refit from may change order of which was best,
1226 # leading to better result chosen (False case). But sometimes want to see exact same model/features with only one feature added,
1227 # and then would need to set this to True case.
1228 # E.g. if refit with just 1 extra column and have interpretability=1, then final model will be same features,
1229 # with one more engineered feature applied to that new original feature.
1230 #
1231 #refit_same_best_individual = false
1232 #
1233 # Directory, relative to data_directory, to store H2O.ai brain meta model files
1234 #brain_rel_dir = "H2O.ai_brain"
1235 #
1236 # Maximum size in bytes the brain will store
1237 # We reserve this memory to save data in order to ensure we can retrieve an experiment if
1238 # for any reason it gets interrupted.
1239 # -1: unlimited
1240 # >=0 number of GB to limit brain to
1241 #brain_max_size_GB = 20
1242 #
1243 # Whether to take any new columns and add additional features to pipeline, even if doing retrain final model.
1244 # In some cases, one might have a new dataset but only want to keep same pipeline regardless of new columns,
1245 # in which case one sets this to False. For example, new data might lead to new dropped features,
1246 # due to shift or leak detection. To avoid change of feature set, one can disable all dropping of columns,
1247 # but set this to False to avoid adding any columns as new features,
1248 # so pipeline is perfectly preserved when changing data.
1249 #brain_add_features_for_new_columns = true
1250 #
1251 # If restart/refit and no longer have the original model class available, be conservative
1252 # and go back to defaults for that model class. If False, then try to keep original hyperparameters,
1253 # which can fail to work in general.
1254 #force_model_restart_to_defaults = true
1255 #
1256 # Whether to enable early stopping
1257 # Early stopping refers to stopping the feature evolution/engineering process
1258 # when there is no performance uplift after a certain number of iterations.
1259 # After early stopping has been triggered, Driverless AI will initiate the ensemble
1260 # process if selected.
1261 #early_stopping = true
1262 #
1263 # Whether to enable early stopping per individual
1264 # Each individual in the generic algorithm will stop early if no improvement,
1265 # and it will no longer be mutated.
1266 # Instead, the best individual will be additionally mutated.
1267 #early_stopping_per_individual = true
1268 #
1269 # Minimum number of Driverless AI iterations to stop the feature evolution/engineering
1270 # process even if score is not improving. Driverless AI needs to run for at least that many
1271 # iterations before deciding to stop. It can be seen a safeguard against suboptimal (early)

```

```

1271 # convergence.
1272 #min_dai_iterations = 0
1273
1274 # Maximum features per model (and each model within the final model if ensemble) kept.
1275 # Keeps top variable importance features, prunes rest away, after each scoring.
1276 # Final ensemble will exclude any pruned-away features and only train on kept features,
1277 # but may contain a few new features due to fitting on different data view (e.g. new clusters)
1278 # Final scoring pipeline will exclude any pruned-away features,
1279 # but may contain a few new features due to fitting on different data view (e.g. new clusters)
1280 # -1 means no restrictions except internally-determined memory and interpretability restrictions.
1281 # Notes:
1282 # * If interpretability > remove_scored_0gain_genes_in_postprocessing_above_interpretability, then
1283 # every GA iteration post-processed features down to this value just after scoring them. Otherwise,
1284 # only mutations of scored individuals will be pruned (until the final model where limits are strictly applied).
1285 # * If ngenes_max is not also limited, then some individuals will have more genes and features until
1286 # pruned by mutation or by preparation for final model.
1287 # * E.g. to generally limit every iteration to exactly 1 features, one must set nfeatures_max=ngenes_max=1
1288 # and remove_scored_0gain_genes_in_postprocessing_above_interpretability=0, but the genetic algorithm
1289 # will have a harder time finding good features.
1290 #
1291 #nfeatures_max = -1
1292
1293 # Maximum genes (transformer instances) per model (and each model within the final model if ensemble) kept.
1294 # Controls number of genes before features are scored, so just randomly samples genes if pruning occurs.
1295 # If restriction occurs after scoring features, then aggregated gene importances are used for pruning genes.
1296 # Instances includes all possible transformers, including original transformer for numeric features.
1297 # -1 means no restrictions except internally-determined memory and interpretability restrictions
1298 #ngenes_max = -1
1299
1300 # Whether to limit feature counts by interpretability setting via features_allowed_by_interpretability
1301 #limit_features_by_interpretability = true
1302
1303 # Whether to use Word-based CNN TensorFlow models for NLP if TensorFlow enabled
1304 #enable_tensorflow_textcnn = "auto"
1305
1306 # Whether to use Word-based Bi-GRU TensorFlow models for NLP if TensorFlow enabled
1307 #enable_tensorflow_textbigru = "auto"
1308
1309 # Whether to use Character-level CNN TensorFlow models for NLP if TensorFlow enabled
1310 #enable_tensorflow_charcnn = "auto"
1311
1312 # Whether to use pretrained PyTorch models and fine-tune them for NLP tasks. Requires internet connection. To enable, set to 'on'. Some PyTorch NLP models ↳
1313 # might only use one text column. Requires enable_pytorch be auto or on.
1314 #enable_pytorch_nlp = "auto"
1315
1316 # Select which pretrained PyTorch NLP model(s) to use. Non-default ones might have no MOJO support. Requires internet connection. Only if BERT PyTorch ↳
1317 #models for NLP are set to 'on'.
1318 #pytorch_nlp_pretrained_models = ["bert-base-uncased", "distilbert-base-uncased"]
1319
1320 # Max. number of epochs for TensorFlow models for making NLP features
1321 #tensorflow_max_epochs_nlp = 2
1322
1323 # Accuracy setting EQUAL and above which will add all enabled TensorFlow NLP models below at start of experiment for text dominated problems
1324 # when TensorFlow NLP transformers are set to auto. If set to on, this parameter is ignored.
1325 # Otherwise, at lower accuracy, TensorFlow NLP transformations will only be created as a mutation.
1326 #enable_tensorflow_nlp_accuracy_switch = 5
1327
1328 # Path to pretrained embeddings for TensorFlow NLP models
1329 # For example, download and unzip https://nlp.stanford.edu/data/glove.6B.zip
1330 # tensorflow_nlp_pretrained_embeddings_file_path = /path/on/server/to/glove.6B.300d.txt
1331 #tensorflow_nlp_pretrained_embeddings_file_path = ""
1332
1333 # Allow training of all weights of the neural network graph, including the pretrained embedding layer weights. If disabled, then the embedding layer is ↳
1334 #frozen, but all other weights are still fine-tuned.
1335 #tensorflow_nlp_pretrained_embeddings_trainable = false
1336
1337 #tensorflow_nlp_have_gpus_in_production = false
1338
1339 #enable_bert_transformer_acceptance_test = false
1340
1341 #enable_bert_model_acceptance_test = false
1342
1343 # Number of epochs for fine-tuning of PyTorch NLP models.
1344 #pytorch_nlp_fine_tuning_num_epochs = 2
1345
1346 # Batch size for PyTorch NLP models. Larger models and larger batch sizes will use more memory.
1347 #pytorch_nlp_fine_tuning_batch_size = 10
1348
1349 # Maximum sequence length (padding length) for PyTorch NLP models. Larger models and larger padding lengths will use more memory.
1350 #pytorch_nlp_fine_tuning_padding_length = 100
1351
1352 # Path to pretrained PyTorch NLP models.
1353 # To get all models, download http://s3.amazonaws.com/artifacts.h2o.ai/releases/ai/h2o/pretrained/bert_models.zip
1354 # and unzip and store it in a directory on the instance where DAI is installed.
1355 # pytorch_nlp_pretrained_models_dir = /path/on/server/to/bert_models_folder
1356 #pytorch_nlp_pretrained_models_dir = ""
1357
1358 # Fraction of text columns out of all features to be considered a text-dominated problem
1359 #text_fraction_for_text_dominated_problem = 0.3
1360
1361 # Fraction of text transformers to all transformers above which to trigger that text dominated problem
1362 #text_transformer_fraction_for_text_dominated_problem = 0.3
1363
1364 # Threshold for average string-is-text score as determined by internal heuristics
1365 # It decides when a string column will be treated as text (for an NLP problem) or just as
1366 # a standard categorical variable.
1367 # Higher values will favor string columns as categoricals, lower values will favor string columns as text
1368 #string_col_as_text_threshold = 0.3
1369
1370 # Minimum fraction of unique values for string columns to be considered as possible text (otherwise categorical)
1371 #string_col_as_text_min_relative_cardinality = 0.1
1372
1373 # Minimum number of uniques for string columns to be considered as possible text (otherwise categorical)
1374 #string_col_as_text_min_absolute_cardinality = 100

```

(continues on next page)

(continued from previous page)

```

1373 # Supported image types. URIs with these endings will be considered as image paths (local or remote).
1374 #supported_image_types = ["'jpg', 'jpeg', 'png', 'bmp', 'ppm', 'tif', 'tiff', 'JPG', 'JPEG', 'PNG', 'BMP', 'PPM', 'TIF', 'TIFF']"
1375
1376 # Whether to use pretrained TensorFlow deep learning models for processing of image data as part of the feature engineering pipeline. A column of URIs to
1377 # images (jpg, png) will be converted to a numeric representation using pre-trained deeplearning models. If no GPUs are found, then must be set to 'on'
1378 # to enable.
1379 #enable_tensorflow_image = "on"
1380
1381 # Supported pretrained image recognition models. Non-default ones will require internet access to download pretrained models from H2O S3 buckets (To get
1382 # all models, download http://s3.amazonaws.com/artifacts.h2o.ai/releases/ai/h2o/pretrained/image_models.zip and unzip inside ./tmp or in tensorflow_image_
1383 #pretrained_models_dir specified in config.toml).
1384 #tensorflow_image_pretrained_models = ["'exception'"]
1385
1386 # Dimensionality of feature space created by TensorFlow image models. If more than one is selected, multiple transformers can be active at the same time.
1387 #tensorflow_image_vectorization_output_dimension = "[100]"
1388
1389 # Enable fine-tuning of the pretrained image models used for the Image Vectorizer transformers. Enabling this will slow down training, but should increase
1390 #accuracy.
1391 #tensorflow_image_fine_tune = false
1392
1393 # Number of epochs for fine-tuning of TensorFlow image models.
1394 #tensorflow_image_fine_tuning_num_epochs = 2
1395
1396 # Batch size for TensorFlow image models. Larger models and larger batch sizes will use more memory.
1397 #tensorflow_image_batch_size = -1
1398
1399 #tensorflow_image_pretrained_models_dir = "./pretrained/image/"
1400
1401 # Max. number of seconds to wait for image download if images are provided by URL
1402 #image_download_timeout = 60
1403
1404 # Maximum fraction of missing elements in a string column for it to be considered as possible image paths (URIs)
1405 #string_col_as_image_max_missing_fraction = 0.1
1406
1407 # Fraction of (unique) image URIs that need to have valid endings (as defined by string_col_as_image_valid_types) for a string column to be considered as
1408 # image data
1409 #string_col_as_image_min_valid_types_fraction = 0.8
1410
1411 # Whether to use GPU(s), if available, to make predictions with pretrained TensorFlow models. Can lead to significant speedups.
1412 #tensorflow_image_use_gpu = true
1413
1414 # Percentile value cutoff of input text token lengths for nlp deep learning models
1415 #text_dl_token_pad_percentile = 99
1416
1417 # Maximum token length of input text to be used in nlp deep learning models
1418 #text_dl_token_pad_max = 512
1419
1420 # Interpretability setting equal and above which will use automatic monotonicity constraints in
1421 # XGBoostGBM/LightGBM/DecisionTree models.
1422 #
1423 #monotonicity_constraints_interpretability_switch = 7
1424
1425 # Threshold, of Pearson product-moment correlation coefficient between numerical or encoded transformed
1426 # feature and target, above (below negative for) which will enforce positive (negative) monotonicity
1427 # for XGBoostGBM, LightGBM and DecisionTree models.
1428 # Enabled when interpretability >= monotonicity_constraints_interpretability_switch config toml value.
1429 # Only if monotonicity_constraints_dict is not provided.
1430 #
1431 #monotonicity_constraints_correlation_threshold = 0.1
1432
1433 # Manual override for monotonicity constraints. Mapping of original numeric features to desired constraint
1434 # (1 for pos, -1 for neg, or 0 to disable). Features that are not listed here will automatically get no
1435 # constraint (i.e., 0). Example: {'PAY_0': -1, 'PAY_2': -1, 'AGE': -1, 'BILL_AMT1': 1, 'PAY_AMT1': -1}
1436 # If not provided, then the automatic correlation based method will be in effect if monotonicity constraints are
1437 # enabled at high enough interpretability settings.
1438 #
1439 #monotonicity_constraints_dict = "{}"
1440
1441 # Exploring feature interactions can be important in gaining better predictive performance.
1442 # The interaction can take multiple forms (i.e. feature1 + feature2 or feature1 * feature2 + ... featureN)
1443 # Although certain machine learning algorithms (like tree-based methods) can do well in
1444 # capturing these interactions as part of their training process, still generating them may
1445 # help them (or other algorithms) yield better performance.
1446 # The depth of the interaction level (as in "up to" how many features may be combined at
1447 # once to create one single feature) can be specified to control the complexity of the
1448 # feature engineering process. For transformers that use both numeric and categorical features, this constrains
1449 # the number of each type, not the total number. Higher values might be able to make more predictive models
1450 # at the expense of time (-1 means automatic).
1451 #max_feature_interaction_depth = -1
1452
1453 # Instead of sampling from min to max (up to max_feature_interaction_depth unless all specified)
1454 # columns allowed for each transformer (0), choose fixed non-zero number of columns to use.
1455 # Can make same as number of columns to use all columns for each transformers if allowed by each transformer.
1456 # -n can be chosen to do 50/50 sample and fixed of n features.
1457 #
1458 #fixed_feature_interaction_depth = 0
1459
1460 # Accuracy setting equal and above which enables tuning of model parameters
1461 # Only applicable if parameter_tuning_num_models=-1 (auto)
1462 #tune_parameters_accuracy_switch = 3
1463
1464 # Accuracy setting equal and above which enables tuning of target transform for regression.
1465 # This is useful for time series when instead of predicting the actual target value, it
1466 # might be better to predict a transformed target variable like sqrt(target) or log(target)
1467 # as a means to control for outliers.
1468 #tune_target_transform_accuracy_switch = 3
1469
1470 # Select a target transformation for regression problems. Must be one of: ['auto',
1471 # 'identity', 'identity_noclip', 'unit_box', 'log', 'square', 'sqrt', 'double_sqrt', 'inverse', 'anscombe', 'logit', 'sigmoid'].
1472 # If set to 'auto', will automatically pick the best target transformer (if accuracy is set to
1473 # tune_target_transform_accuracy_switch or larger). All transformers except for 'identity_noclip' perform clipping
1474 # to constrain the predictions to the domain of the target in the training data. Use 'identity_noclip' to
1475 # effectively disable target transformations and to allow predictions outside of the target domain observed in
1476 # the training data (for parametric models or custom models that support extrapolation).

```

(continues on next page)

(continued from previous page)

```

1471 #
1472 #target_transformer = "auto"
1473
1474 # Tournament style (method to decide which models are best at each iteration)
1475 # 'auto' : Choose based upon accuracy, etc.
1476 # 'fullstack' : Choose among optimal model and feature types
1477 # 'model' : individuals with same model type compete
1478 # 'feature' : individuals with similar feature types compete
1479 # 'uniform' : all individuals in population compete to win as best
1480 # 'model' and 'feature' styles preserve at least one winner for each type (and so 2 total indivs of each type after mutation)
1481 # For each case, a round robin approach is used to choose best scores among type of models to choose from
1482 #tournament_style = "auto"
1483
1484 # Interpretability above which will use 'uniform' tournament style
1485 #tournament_uniform_style_interpretability_switch = 6
1486
1487 # Accuracy below which will use uniform style if tournament_style = 'auto' (regardless of other accuracy tournament style switch values)
1488 #tournament_uniform_style_accuracy_switch = 6
1489
1490 # Accuracy equal and above which uses model style if tournament_style = 'auto'
1491 #tournament_model_style_accuracy_switch = 6
1492
1493 # Accuracy equal and above which uses feature style if tournament_style = 'auto'
1494 #tournament_feature_style_accuracy_switch = 7
1495
1496 # Accuracy equal and above which uses fullstack style if tournament_style = 'auto'
1497 #tournament_fullstack_style_accuracy_switch = 8
1498
1499 # Whether to use penalized score for GA tournament or actual score
1500 #tournament_use_feature_penalized_score = true
1501
1502 # Driverless AI uses a genetic algorithm (GA) to find the best features, best models and
1503 # best hyper parameters for these models. The GA facilitates getting good results while not
1504 # requiring torun/try every possible model/feature/parameter. This version of GA has
1505 # reinforcement learning elements - it uses a form of exploration-exploitation to reach
1506 # optimum solutions. This means it will capitalise on models/features/parameters that seem # to be working well and continue to exploit them even more, ↵
1507 # while allowing some room for
1508 # trying new (and semi-random) models/features/parameters to avoid settling on a local
1509 # minimum.
1510 # These models/features/parameters tried are what-we-call individuals of a population. More # individuals connote more models/features/parameters to be ↵
1511 # tried and compete to find the best # ones.
1512 #num_individuals = 2
1513
1514 # set fixed number of individuals (if > 0) - useful to compare different hardware configurations
1515 #fixed_num_individuals = 0
1516
1517 # set fixed number of fold reps (if > 0) - useful for quick runs regardless of data
1518 #fixed_fold_reps = 0
1519
1520 # number of unique targets or folds counts after which switch to faster/simpler non-natural sorting and print outs
1521 #sanitize_natural_sort_limit = 1000
1522
1523 # Whether target encoding (CV target encoding, weight of evidence, etc.) could be enabled
1524 # Target encoding refers to several different feature transformations (primarily focused on
1525 # categorical data) that aim to represent the feature using information of the actual
1526 # target variable. A simple example can be to use the mean of the target to replace each
1527 # unique category of a categorical feature. This type of features can be very predictive,
1528 # but are prone to overfitting and require more memory as they need to store mappings of
1529 # the unique categories and the target values.
1530 #enable_target_encoding = "auto"
1531
1532 # For target encoding,
1533 # whether an outer level of cross-fold validation is performed,
1534 # in cases when GINI is detected to flip sign (or have inconsistent sign for weight of evidence)
1535 # between fit_transform on training, transform on training, and transform on validation data.
1536 # The degree to which GINI is poor is also used to perform fold-averaging of look-up tables instead
1537 # of using global look-up tables.
1538 #cvte_cv_in_cv = true
1539
1540 #enable_lexilabel_encoding = "off"
1541
1542 #enable_isolation_forest = "off"
1543
1544 # Whether one hot encoding could be enabled. If auto, then only applied for small data and GLM.
1545 #enable_one_hot_encoding = "auto"
1546
1547 #isolation_forest_nestimators = 200
1548
1549 # Driverless AI categorises all data (feature engineering) transformers
1550 # More information for these transformers can be viewed here:
1551 # http://docs.h2o.ai/driverless-ai/latest-stable/docs/userguide/transformations.html
1552 # This section allows including/excluding these transformations and may be useful when
1553 # simpler (more interpretable) models are sought at the expense of accuracy.
1554 # the interpretability setting)
1555 # for multi-class: ['NumCatTETransformer', 'TextLinModelTransformer',
1556 # 'FrequentTransformer', 'CVTargetEncodeTransformer', 'ClusterDistTransformer',
1557 # 'WeightOfEvidenceTransformer', 'TruncSVDNumTransformer', 'CVCatNumEncodeTransformer',
1558 # 'DatesTransformer', 'TextTransformer', 'OriginalTransformer',
1559 # 'NumToCatWoETransformer', 'NumToCatTETransformer', 'ClusterTETransformer',
1560 # 'InteractionsTransformer']
1561 # for regression/binary: ['TextTransformer', 'ClusterDistTransformer',
1562 # 'OriginalTransformer', 'TextLinModelTransformer', 'NumToCatTETransformer',
1563 # 'DatesTransformer', 'WeightOfEvidenceTransformer', 'InteractionsTransformer',
1564 # 'FrequentTransformer', 'CVTargetEncodeTransformer', 'NumCatTETransformer',
1565 # 'NumToCatWoETransformer', 'TruncSVDNumTransformer', 'ClusterTETransformer',
1566 # 'CVCatNumEncodeTransformer']
1567 # This list appears in the experiment logs (search for 'Transformers used')
1568
1569 #included_transformers = "[]"
1570
1571 # Auxiliary to included_transformers
1572 # e.g. to disable all Target Encoding: excluded_transformers =
1573 # ['NumCatTETransformer', 'CVTargetEncodeF', 'NumToCatTETransformer',
1574 # 'ClusterTETransformer'].

```

(continues on next page)

(continued from previous page)

```

1573 # Does not affect transformers used for preprocessing with included_pretransformers.
1574 #
1575 #excluded_transformers = "[]"
1576
1577 # Exclude list of genes (i.e. genes (built on top of transformers) to not use,
1578 # independent of the interpretability setting)
1579 # Some transformers are used by multiple genes, so this allows different control over feature engineering
1580 # for multi-class: ['InteractionsGene', 'WeightOfEvidenceGene',
1581 # 'NumToCatTargetEncodeSingleGene', 'OriginalGene', 'TextGene', 'FrequentGene',
1582 # 'NumToCatWeightOfEvidenceGene', 'NumToCatWeightOfEvidenceMonotonicGene', '
1583 # CvTargetEncodeSingleGene', 'DateGene', 'NumToCatTargetEncodeMultiGene', '
1584 # DateTimeGene', 'TextLinRegressorGene', 'ClusterIDTargetEncodeSingleGene',
1585 # 'CvCatNumEncodeGene', 'TruncSvdNumGene', 'ClusterIDTargetEncodeMultiGene',
1586 # 'NumCatTargetEncodeMultiGene', 'CvTargetEncodeMultiGene', 'TextLinClassifierGene',
1587 # 'NumCatTargetEncodeSingleGene', 'ClusterDistGene']
1588 # for regression/binary: ['CvTargetEncodeSingleGene', 'NumToCatTargetEncodeSingleGene',
1589 # 'CvCatNumEncodeGene', 'ClusterIDTargetEncodeSingleGene', 'TextLinRegressorGene',
1590 # 'CvTargetEncodeMultiGene', 'ClusterDistGene', 'OriginalGene', 'DateGene',
1591 # 'ClusterIDTargetEncodeMultiGene', 'NumToCatTargetEncodeMultiGene',
1592 # 'NumCatTargetEncodeMultiGene', 'TextLinClassifierGene', 'WeightOfEvidenceGene',
1593 # 'FrequentGene', 'TruncSvdNumGene', 'InteractionsGene', 'TextGene',
1594 # 'DateTimeGene', 'NumToCatWeightOfEvidenceGene',
1595 # 'NumToCatWeightOfEvidenceMonotonicGene', 'NumCatTargetEncodeSingleGene']
1596 # This list appears in the experiment logs (search for 'Genes used')
1597 # e.g. to disable interaction gene, use: excluded_genes =
1598 # ['InteractionsGene'].
1599 # Does not affect transformers used for preprocessing with included_pretransformers.
1600 #
1601 #excluded_genes = "[]"
1602
1603 #included_models = "[]"
1604
1605 # Auxiliary to included_models
1606 #excluded_models = "[]"
1607
1608 #included_scorers = "[]"
1609
1610 # Select transformers to be used for preprocessing before other transformers operate.
1611 # Pre-processing transformers can potentially take any original features and output
1612 # arbitrary features, which will then be used by the normal layer of transformers
1613 # whose selection is controlled by toml included_transformers or via the GUI
1614 # "Include specific transformers".
1615 # Notes:
1616 # 1) preprocessing transformers (and all other layers of transformers) are part of the python and (if applicable) mojo scoring packages.
1617 # 2) any BYOR transformer recipe or native DAI transformer can be used as a preprocessing transformer.
1618 # So, e.g., a preprocessing transformer can do interactions, string concatenations, date extractions as a preprocessing step,
1619 # and next layer of Date and DateTime transformers will use that as input data.
1620 # Caveats:
1621 # 1) one cannot currently do a time-series experiment on a time_column that hasn't yet been made (setup of experiment only knows about original data, not ↵transformed)
1622 # However, one can use a run-time data recipe to (e.g.) convert a float date-time into string date-time, and this will
1623 # be used by DAIs Date and DateTime transformers as well as auto-detection of time series.
1624 # 2) in order to do a time series experiment with the GUI/client auto-selecting groups, periods, etc. the dataset
1625 # must have time column and groups prepared ahead of experiment by user or via a one-time data recipe.
1626 #
1627 #included_pretransformers = "[]"
1628
1629 # Number of full pipeline layers
1630 # (not including preprocessing layer when included_pretransformers is not empty).
1631 #
1632 #num_pipeline_layers = 1
1633
1634 # There are 2 data recipes:
1635 # 1) that adds new dataset or modifies dataset outside experiment by file/url (pre-experiment data recipe)
1636 # 2) that modifies dataset during experiment and python scoring (run-time data recipe)
1637 # This list applies to the 2nd case. One can use the same data recipe code for either case, but note:
1638 # A) the 1st case can make any new data, but is not part of scoring package.
1639 # B) the 2nd case modifies data during the experiment, so needs some original dataset.
1640 # The recipe can still create all new features, as long as it has same *name* for:
1641 # target, weight_column, fold_column, time_column, time group columns.
1642 #
1643 #included_datas = "[]"
1644
1645 # Auxiliary to included_datas
1646 #excluded_datas = "[]"
1647
1648 # Select the scorer to optimize the binary probability threshold that is being used in related Confusion Matrix based scorers such as: Precision, Recall, ↵FalsePositiveRate, FalseDiscoveryRate, FalseOmissionRate, TrueNegativeRate, FalseNegativeRate, NegativePredictiveValue. Use F1 if the target class ↵matters more, and MCC if all classes are equally important. AUTO will try to sync the threshold scorer with the scorer used for the experiment, ↵otherwise falls back to F1.
1649 #threshold_scorer = "AUTO"
1650
1651 # Auxiliary to included_scorers
1652 #excluded_scorers = "[]"
1653
1654 # Whether to enable constant models ('auto'/'on'/'off')
1655 #enable_constant_model = "auto"
1656
1657 # Whether to enable Decision Tree models ('auto'/'on'/'off')
1658 #enable_decision_tree = "auto"
1659
1660 # Whether to enable GLM models ('auto'/'on'/'off')
1661 #enable_glm = "auto"
1662
1663 # Whether to enable XGBoost GBM models ('auto'/'on'/'off')
1664 #enable_xgboost_gbm = "auto"
1665
1666 # Whether to enable XGBoost Dart models ('auto'/'on'/'off')
1667 #enable_xgboost_dart = "auto"
1668
1669 # Internal threshold for number of rows x number of columns to trigger no xgboost models due to high memory use
1670 # Overridden if enable_xgboost_gbm = "on" or enable_xgboost_dart = "on", in which case always allow each model type to be used
1671 #xgboost_threshold_data_size_large = 100000000
1672

```

(continues on next page)

(continued from previous page)

```

1673 # Internal threshold for number of rows x number of columns to trigger no xgboost models due to limits on GPU memory capability
1674 # Overridden if enable_xgboost_gbm = "on" or enable_xgboost_dart = "on", in which case always allow each model type to be used
1675 #xgboost_gpu_threshold_data_size_large = 30000000
1676
1677 # Whether to enable LightGBM models ('auto'/'on'/'off')
1678 #enable_lightgbm = "auto"
1679
1680 # Whether to enable TensorFlow models ('auto'/'on'/'off')
1681 #enable_tensorflow = "auto"
1682
1683 # Whether to enable PyTorch models ('auto'/'on'/'off') like NLP Bert Models.
1684 #enable_pytorch = "auto"
1685
1686 # Whether to enable FTRL support (follow the regularized leader) model ('auto'/'on'/'off')
1687 #enable_ftrl = "auto"
1688
1689 # Whether to enable RuleFit support (beta version, no mojo) ('auto'/'on'/'off')
1690 #enable_rulefit = "auto"
1691
1692 # Whether to enable automatic addition of zero-inflated models for regression problems with zero-inflated target values that meet certain conditions: y >= ↵
1693 #             ↵0, y.std() > y.mean()
1694 #enable_zero_inflated_models = "auto"
1695
1696 # Which boosting types to enable for LightGBM (gbdt = boosted trees, rf_early_stopping = random forest with early stopping rf = random forest (no early ↵
1697 #             ↵stopping), dart = drop-out boosted trees with no early stopping
1698 #enable_lightgbm_boosting_types = ["gbdt"]
1699
1700 # Whether to enable LightGBM categorical feature support (only CPU mode currently, and no MOJO built)
1701 #enable_lightgbm_cat_support = false
1702
1703 # Whether to show constant models in iteration panel
1704 #show_constant_model = false
1705
1706 # Parameters for LightGBM to override DAI parameters
1707 # parameters should be given as XGBoost equivalent unless unique LightGBM parameter
1708 # e.g. 'eval_metric' instead of 'metric' should be used
1709 # e.g. params_lightgbm = {"objective": "binary:logistic", "n_estimators": 100, "max_leaves": 64, "random_state": 1234}
1710 # e.g. params_lightgbm = {"n_estimators": 600, "learning_rate": 0.1, "reg_alpha": 0.0, "reg_lambda": 0.5, "gamma": 0, "max_depth": 0, "max_bin": 128, "max_
1711 #             ↵leaves": 256, "scale_pos_weight": 1.0, "max_delta_step": 1, "min_child_weight": 1, "subsample": 0.9, "colsample_bytree": 0.3, "tree_
1712 #             ↵method": "gpu_hist", "grow_policy": "lossguide", "min_data_in_bin": 3, "min_child_samples": 5, "early_stopping_rounds": 20, "num_classes": 2, "objective_
1713 #             ↵": "binary:logistic", "eval_metric": "logloss", "random_state": 987654, "early_stopping_threshold": 0.01, "monotonicity_constraints": False, "silent": ↵
1714 #             ↵True, "debug_verbose": 0, "subsample_freq": 1}
1715 # avoid including "system"-level parameters like 'n_gpus': 1, 'gpu_id': 0, 'n_jobs': 1, 'booster': 'lightgbm'
1716 # also likely should avoid parameters like: 'objective': 'binary:logistic', unless one really knows what one is doing (e.g. alternative objectives)
1717 # See: https://xgboost.readthedocs.io/en/latest/parameter.html
1718 # And see: https://github.com/Microsoft/LightGBM/blob/master/docs/Parameters.rst
1719 # Can also pass objective parameters if choose (or in case automatically chosen) certain objectives
1720 # https://lightgbm.readthedocs.io/en/latest/Parameters.html#metric-parameters
1721 #params_lightgbm = "{}"
1722
1723 # Parameters for XGBoost to override DAI parameters
1724 # similar parameters as lightgbm since lightgbm parameters are transcribed from xgboost equivalent versions
1725 # e.g. params_xgboost = {"n_estimators": 100, "max_leaves": 64, "max_depth": 0, "random_state": 1234}
1726 # See: https://xgboost.readthedocs.io/en/latest/parameter.html
1727 #params_xgboost = "{}"
1728
1729 # Like params_xgboost but for XGBoost's dart method
1730 #params_dart = "{}"
1731
1732 # Parameters for TensorFlow to override DAI parameters
1733 # e.g. params_tensorflow = {"lr": 0.01, "add_wide": False, "add_attention": True, "epochs": 30, "layers": [100, 100], "activation": "selu", "batch_size": ↵
1734 #             ↵64, "chunk_size": 1000, "dropout": 0.3, "strategy": "one_shot", "l1": 0.0, "l2": 0.0, "ort_loss": 0.5, "ort_loss_tau": 0.01, "normalize_type": ↵
1735 #             ↵"streaming"}
1736 # See: https://keras.io/, e.g. for activations: https://keras.io/activations/
1737 # Example layers: [500, 500, 500], [100, 100, 100], [100, 100], [50, 50]
1738 # Strategies: 'lcycle' or 'one_shot', See: https://github.com/fastai/fastai
1739 # normalize_type: 'streaming' or 'global' (using sklearn StandardScaler)
1740 #params_tensorflow = "{}"
1741
1742 # Parameters for XGBoost's gblinear to override DAI parameters
1743 # e.g. params_gblinear = {"n_estimators": 100}
1744 # See: https://xgboost.readthedocs.io/en/latest/parameter.html
1745 #params_gblinear = "{}"
1746
1747 # Parameters for Decision Tree to override DAI parameters
1748 # parameters should be given as XGBoost equivalent unless unique LightGBM parameter
1749 # e.g. 'eval_metric' instead of 'metric' should be used
1750 # e.g. params_decision_tree = {"objective": "binary:logistic", "n_estimators": 100, "max_leaves": 64, "random_state": 1234}
1751 # e.g. params_decision_tree = {"n_estimators": 1, "learning_rate": 1, "reg_alpha": 0.0, "reg_lambda": 0.5, "gamma": 0, "max_depth": 0, "max_
1752 #             ↵leaves": 256, "scale_pos_weight": 1.0, "max_delta_step": 3.469919910597877, "min_child_weight": 1, "subsample": 0.9, "colsample_bytree": 0.3, "tree_
1753 #             ↵method": "gpu_hist", "grow_policy": "lossguide", "min_data_in_bin": 3, "min_child_samples": 5, "early_stopping_rounds": 20, "num_classes": 2, "objective_
1754 #             ↵": "binary:logistic", "eval_metric": "logloss", "random_state": 987654, "early_stopping_threshold": 0.01, "monotonicity_constraints": False, "silent": ↵
1755 #             ↵True, "debug_verbose": 0, "subsample_freq": 1}
1756 # avoid including "system"-level parameters like 'n_gpus': 1, 'gpu_id': 0, 'n_jobs': 1, 'booster': 'lightgbm'
1757 # also likely should avoid parameters like: 'objective': 'binary:logistic', unless one really knows what one is doing (e.g. alternative objectives)
1758 # See: https://xgboost.readthedocs.io/en/latest/parameter.html
1759 # And see: https://github.com/Microsoft/LightGBM/blob/master/docs/Parameters.rst
1760 # Can also pass objective parameters if choose (or in case automatically chosen) certain objectives
1761 # https://lightgbm.readthedocs.io/en/latest/Parameters.html#metric-parameters
1762 #params_decision_tree = "{}"
1763
1764 # Parameters for Rulefit to override DAI parameters
1765 # e.g. params_rulefit = {"max_leaves": 64}
1766 # See: https://xgboost.readthedocs.io/en/latest/parameter.html
1767 #params_rulefit = "{}"
1768
1769 # Parameters for FTRL to override DAI parameters
1770 #params_ftrl = "{}"
1771
1772 # Dictionary of key:lists of values to use for LightGBM tuning, overrides DAI's choice per key
1773 # e.g. params_tune_lightgbm = {"min_child_samples": [1,2,5,100,1000], "min_data_in_bin": [1,2,3,10,100,1000]}
1774 #params_tune_lightgbm = "{}"

```

(continues on next page)

(continued from previous page)

```

1765
1766 # Like params_tune_lightgbm but for XGBoost
1767 # e.g. params_tune_xgboost = {'max_leaves': [8, 16, 32, 64]}
1768 #params_tune_xgboost = "{}"
1769
1770 # Like params_tune_lightgbm but for XGBoost's Dart
1771 # e.g. params_tune_dart = {'max_leaves': [8, 16, 32, 64]}
1772 #params_tune_dart = "{}"
1773
1774 # Like params_tune_lightgbm but for TensorFlow
1775 # e.g. params_tune_tensorflow = {'layers': [[10,10,10], [10, 10, 10, 10]]}
1776 #params_tune_tensorflow = "{}"
1777
1778 # Like params_tune_lightgbm but for gblinear
1779 # e.g. params_tune_gblinear = {'reg_lambda': [.01, .001, .0001, .0002]}
1780 #params_tune_gblinear = "{}"
1781
1782 # Like params_tune_lightgbm but for rulefit
1783 # e.g. params_tune_rulefit = {'max_depth': [4, 5, 6]}
1784 #params_tune_rulefit = "{}"
1785
1786 # Like params_tune_lightgbm but for ftrl
1787 #params_tune_ftrl = "{}"
1788
1789 # Whether to force max_leaves and max_depth to be 0 if grow_policy is depthwise and lossguide, respectively.
1790 #params_tune_grow_policy_simple_trees = true
1791
1792 # Maximum number of GBM trees or GLM iterations
1793 # Early-stopping usually chooses less
1794 #max_nestimators = 3000
1795
1796 # LightGBM dart mode and normal rf mode do not use early stopping and will sample from these values for n_estimators.
1797 #n_estimators_list_no_early_stopping = [50, 100, 200, 300]
1798
1799 # Lower limit on learning rate for final ensemble GBM models.
1800 # In some cases, the maximum number of trees/iterations is insufficient for the final learning rate,
1801 # which can lead to no early stopping triggered and poor final model performance.
1802 # Then, one can try increasing the learning rate by raising this minimum,
1803 # or one can try increasing the maximum number of trees/iterations.
1804 #
1805 #min_learning_rate_final = 0.01
1806
1807 # Upper limit on learning rate for final ensemble GBM models
1808 #max_learning_rate_final = 0.05
1809
1810 # factor by which max_nestimators is reduced for tuning and feature evolution
1811 #max_nestimators_feature_evolution_factor = 0.2
1812
1813 # Lower limit on learning rate for feature engineering GBM models
1814 #min_learning_rate = 0.05
1815
1816 # Upper limit on learning rate for GBM models
1817 # If want to override min_learning_rate and min_learning_rate_final, set this to smaller value
1818 #max_learning_rate = 0.5
1819
1820 # Max. number of epochs for TensorFlow and FTRL models
1821 #max_epochs = 10
1822
1823 # Maximum tree depth (and corresponding max max_leaves as 2**max_max_depth)
1824 #max_max_depth = 12
1825
1826 # Default max_bin for tree methods
1827 #default_max_bin = 256
1828
1829 # Default max_bin for lightgbm (recommended for GPU lightgbm)
1830 #default_lightgbm_max_bin = 64
1831
1832 # Maximum max_bin for tree features
1833 #max_max_bin = 256
1834
1835 # Minimum max_bin for any tree
1836 #min_max_bin = 32
1837
1838 # Amount of memory which can handle max_bin = 256 can handle 125 columns and max_bin = 32 for 1000 columns
1839 # As available memory on system goes higher than this scale, can handle proportionally more columns at higher max_bin
1840 # Currently set to 10GB
1841 #scale_mem_for_max_bin = 10737418240
1842
1843 # Factor by which rf gets more depth than gbdt
1844 #factor_rf = 1.25
1845
1846 # Whether TensorFlow will use all CPU cores, or if it will split among all transformers
1847 #tensorflow_use_all_cores = true
1848
1849 # Whether TensorFlow will use all CPU cores if reproducible is set, or if it will split among all transformers
1850 #tensorflow_use_all_cores_even_if_reproducible_true = false
1851
1852 # How many cores to use for each TensorFlow model, regardless if GPU or CPU based (0 = auto mode)
1853 #tensorflow_cores = 0
1854
1855 # Max number of rules to be used for RuleFit models (-1 for all)
1856 #rulefit_max_num_rules = -1
1857
1858 # Max tree depth for RuleFit models
1859 #rulefit_max_tree_depth = 6
1860
1861 # Max number of trees for RuleFit models
1862 #rulefit_max_num_trees = 100
1863
1864 # Internal threshold for number of rows x number of columns to trigger no rulefit models due to being too slow currently
1865 #rulefit_threshold_data_size_large = 100000000
1866
1867 # Enable One-Hot-Encoding (which does binning to limit to number of bins to no more than 100 anyway) for categorical columns with fewer than this many
1868 # unique values

```

(continues on next page)

(continued from previous page)

```

1868 # Set to 0 to disable
1869 #one_hot_encoding_cardinality_threshold = 50
1870
1871 # Fixed ensemble_level
1872 # -1 = auto, based upon ensemble_accuracy_switch, accuracy, size of data, etc.
1873 # 0 = No ensemble, only final single model on validated iteration/tree count
1874 # 1 = 1 model, multiple ensemble folds (cross-validation)
1875 # 2 = 2 models, multiple ensemble folds (cross-validation)
1876 # 3 = 3 models, multiple ensemble folds (cross-validation)
1877 # 4 = 4 models, multiple ensemble folds (cross-validation)
1878 #fixed_ensemble_level = -1
1879
1880 # If enabled, use cross-validation to determine optimal parameters for single final model,
1881 # and to be able to create training holdout predictions.
1882 #cross_validate_single_final_model = true
1883
1884 # Number of models to tune during pre-evolution phase
1885 # Can make this lower to avoid excessive tuning, or make higher to do enhanced tuning.
1886 # -1 : auto
1887 #
1888 #parameter_tuning_num_models = -1
1889
1890 #validate_meta_learner = true
1891
1892 #validate_meta_learner_extra = false
1893
1894 # Specify the fixed number of cross-validation folds (if >= 2) for feature evolution. (The actual number of splits allowed can be less and is determined at
1895 #experiment run-time).
1896 #fixed_num_folds_evolution = -1
1897
1898 # Specify the fixed number of cross-validation folds (if >= 2) for the final model. (The actual number of splits allowed can be less and is determined at
1899 #experiment run-time).
1900 #fixed_num_folds = -1
1901
1902 # set "on" to force only first fold for models - useful for quick runs regardless of data
1903 #fixed_only_first_fold_model = "auto"
1904
1905 #num_fold_ids_show = 10
1906
1907 #fold_scores_instability_warning_threshold = 0.25
1908
1909 # Upper limit on the number of rows x number of columns for feature evolution (applies to both training and validation/holdout splits)
1910 # feature evolution is the process that determines which features will be derived.
1911 # Depending on accuracy settings, a fraction of this value will be used
1912 #feature_evolution_data_size = 100000000
1913
1914 # Upper limit on the number of rows x number of columns for training final pipeline.
1915 #
1916 #final_pipeline_data_size = 500000000
1917
1918 # Smaller values can speed up final pipeline model training, as validation data is only used for early stopping.
1919 # Note that final model predictions and scores will always be provided on the full dataset provided.
1920 #
1921 #max_validation_to_training_size_ratio_for_final_ensemble = 2.0
1922
1923 # Ratio of minority to majority class of the target column beyond which stratified sampling is done for binary classification. Otherwise perform random
1924 #sampling. Set to 0 to always do random sampling. Set to 1 to always do stratified sampling.
1925 #force_stratified_splits_for_imbalanced_threshold_binary = 0.01
1926
1927 # Sampling method for imbalanced binary classification problems. Choices are:
1928 # "auto": sample both classes as needed, depending on data
1929 # "over_under_sampling": over-sample the minority class and under-sample the majority class, depending on data
1930 # "under_sampling": under-sample the majority class to reach class balance
1931 # "off": do not perform any sampling
1932 #
1933 #imbalance_sampling_method = "off"
1934
1935 # For smaller data, there's no generally no benefit in using imbalanced sampling methods.
1936 #imbalance_sampling_threshold_min_rows_original = 100000
1937
1938 # For imbalanced binary classification: ratio of majority to minority class equal and above which to enable
1939 # special imbalanced models with sampling techniques (specified by imbalance_sampling_method) to attempt to improve model performance.
1940 #imbalance_ratio_sampling_threshold = 5
1941
1942 # For heavily imbalanced binary classification: ratio of majority to minority class equal and above which to enable only
1943 # special imbalanced models on full original data, without upfront sampling.
1944 #heavy_imbalance_ratio_sampling_threshold = 25
1945
1946 # -1: automatic
1947 #imbalance_sampling_number_of_bags = -1
1948
1949 # Only for shift/leakage/tuning/feature evolution models. Not used for final models. Final models can
1950 # be limited by imbalance_sampling_max_number_of_bags.
1951 #imbalance_sampling_max_number_of_bags_feature_evolution = 3
1952
1953 # Max. size of data sampled during imbalanced sampling (in terms of dataset size),
1954 # controls number of bags (approximately). Only for imbalance_sampling_number_of_bags == -1.
1955 #imbalance_sampling_max_multiple_data_size = 1.0
1956
1957 # Rank averaging can be helpful when ensembling diverse models when ranking metrics like AUC/Gini
1958 # metrics are optimized. No MOJO support yet.
1959 #imbalance_sampling_rank_averaging = "auto"
1960
1961 # A value of 0.5 means that models/algorithms will be presented a balanced target class distribution
1962 # after applying under/over-sampling techniques on the training data. Sometimes it makes sense to
1963 # choose a smaller value like 0.1 or 0.01 when starting from an extremely imbalanced original target
1964 # distribution. -1.0: automatic
1965 #imbalance_sampling_target_minority_fraction = -1.0
1966
1967 # For binary classification: ratio of majority to minority class equal and above which to notify
1968 # of imbalance in GUI to say slightly imbalanced.

```

(continues on next page)

(continued from previous page)

```

1969 # More than imbalance_ratio_sampling_threshold will say problem is imbalanced.
1970 #
1971 #imbalance_ratio_notification_threshold = 2.0
1972
1973 # list of possible bins for FTRL (largest is default best value)
1974 #nbins_ftrl_list = "[1000000, 10000000, 100000000]"
1975
1976 # Samples the number of automatic FTRL interactions terms to no more than this value (for each of 2nd, 3rd, 4th order terms)
1977 #ftrl_max_interaction_terms_per_degree = 10000
1978
1979 # list of possible bins for target encoding (first is default value)
1980 #te_bin_list = "[25, 10, 100, 250]"
1981
1982 # list of possible bins for weight of evidence encoding (first is default value)
1983 # If only want one value: woe_bin_list = [2]
1984 #woe_bin_list = "[25, 10, 100, 250]"
1985
1986 # list of possible bins for ohe hot encoding (first is default value)
1987 #ohe_bin_list = "[10, 25, 50, 75, 100]"
1988
1989 # Whether to drop columns with constant values
1990 #drop_constant_columns = true
1991
1992 # Whether to drop columns that appear to be an ID
1993 #drop_id_columns = true
1994
1995 # Whether to avoid dropping any columns (original or derived)
1996 #no_drop_features = false
1997
1998 # Direct control over columns to drop in bulk so can copy-paste large lists instead of selecting each one separately in GUI
1999 #cols_to_drop = ""
2000
2001 # Control over columns to group by, default is empty list that means DAI automatically searches all columns,
2002 # selected randomly or by which have top variable importance.
2003 #cols_to_group_by = ""
2004
2005 # Whether to sample from given features to group by (True) or to always group by all features (False).
2006 #sample_cols_to_group_by = false
2007
2008 # Aggregation functions to use for groupby operations.
2009 #agg_funcs_for_group_by = ["'mean', 'sd', 'min', 'max', 'count']"
2010
2011 # Out of fold aggregations ensure less overfitting, but see less data in each fold.
2012 #folds_for_group_by = 5
2013
2014 # Strategy to apply when doing mutations on transformers.
2015 # Sample mode is default, with tendency to sample transformer parameters.
2016 # Batched mode tends to do multiple types of the same transformation together.
2017 # Full mode does even more types of the same transformation together.
2018 #
2019 #mutation_mode = "sample"
2020
2021 # Whether to enable checking text for shift, currently only via label encoding.
2022 #shift_check_text = false
2023
2024 # Whether to use LightGBM random forest mode without early stopping for shift detection.
2025 #use_rf_for_shift_if_have_lgbm = true
2026
2027 # Normalized training variable importance above which to check the feature for shift
2028 # Useful to avoid checking likely unimportant features
2029 #shift_key_features_varimp = 0.01
2030
2031 # Whether to only check certain features based upon the value of shift_key_features_varimp
2032 #shift_check_reduced_features = true
2033
2034 # Number of trees to use to train model to check shift in distribution
2035 # No larger than max_nestimators
2036 #shift_trees = 100
2037
2038 # The value of max_bin to use for trees to use to train model to check shift in distribution
2039 #shift_max_bin = 256
2040
2041 # The min. value of max_depth to use for trees to use to train model to check shift in distribution
2042 #shift_min_max_depth = 4
2043
2044 # The max. value of max_depth to use for trees to use to train model to check shift in distribution
2045 #shift_max_max_depth = 8
2046
2047 # If distribution shift detection is enabled, show features for which shift AUC is above this value
2048 # (AUC of a binary classifier that predicts whether given feature value belongs to train or test data)
2049 #detect_features_distribution_shift_threshold_auc = 0.55
2050
2051 # Minimum number of features to keep, keeping least shifted feature at least if 1
2052 #drop_features_distribution_shift_min_features = 1
2053
2054 # Whether to enable checking text for leakage, currently only via label encoding.
2055 #leakage_check_text = true
2056
2057 # Normalized training variable importance (per 1 minus AUC/R2 to control for leaky varimp dominance) above which to check the feature for leakage
2058 # Useful to avoid checking likely unimportant features
2059 #leakage_key_features_varimp = 0.001
2060
2061 # Like leakage_key_features_varimp, but applies if early stopping disabled when can trust multiple leaks to get uniform varimp.
2062 #leakage_key_features_varimp_if_no_early_stopping = 0.05
2063
2064 # Whether to only check certain features based upon the value of leakage_key_features_varimp. If any feature has AUC near 1, will consume all variable_importance, even if another feature is also leaky. So False is safest option, but True generally good if many columns.
2065 #leakage_check_reduced_features = true
2066
2067 # Whether to use LightGBM random forest mode without early stopping for leakage detection.
2068 #use_rf_for_leakage_if_have_lgbm = true
2069
2070 # Number of trees to use to train model to check for leakage
2071 # No larger than max_nestimators

```

(continues on next page)

(continued from previous page)

```

2072 #leakage_trees = 100
2073
2074 # The value of max_bin to use for trees to use to train model to check for leakage
2075 #leakage_max_bin = 256
2076
2077 # The value of max_depth to use for trees to use to train model to check for leakage
2078 #leakage_min_max_depth = 4
2079
2080 # The value of max_depth to use for trees to use to train model to check for leakage
2081 #leakage_max_max_depth = 8
2082
2083 # When leakage detection is enabled, if AUC (R2 for regression) on original data (label-encoded)
2084 # is above or equal to this value, then trigger per-feature leakage detection
2085 #detect_features_leakage_threshold_auc = 0.95
2086
2087 # When leakage detection is enabled, show features for which AUC (R2 for regression,
2088 # for whether that predictor/feature alone predicts the target) is above or equal to this value.
2089 # Feature is dropped if AUC/R2 is above or equal to drop_features_leakage_threshold_auc
2090 #detect_features_per_feature_leakage_threshold_auc = 0.8
2091
2092 # Minimum number of features to keep, keeping least leakage feature at least if 1
2093 #drop_features_leakage_min_features = 1
2094
2095 # Ratio of train to validation holdout when testing for leakage
2096 #leakage_train_test_split = 0.25
2097
2098 # Whether to enable detailed traces (in GUI Trace)
2099 #detailed_traces = false
2100
2101 # Whether to enable debug log level (in log files)
2102 #debug_log = false
2103
2104 # Whether to add logging of system information such as CPU, GPU, disk space at the start of each experiment log. Same information is already logged in ↵
2105 #log_system_info_per_experiment = true
2106
2107 # How close to the optimal value (usually 1 or 0) does the validation score need to be to be considered perfect (to stop the experiment)?
2108 #abs_tol_for_perfect_score = 0.0001
2109
2110 # Timeout in seconds to wait for data ingestion.
2111 #data_ingest_timeout = 86400.0
2112
2113 # Enable time series recipe
2114 #time_series_recipe = true
2115
2116 # Provide date or datetime timestamps (in same format as the time column) for custom training and validation splits like this: "tr_start1, tr_end1, va_
2117 # ↵start1, va_end1, ..., tr_startN, tr_endN, va_startN, va_endN"
2118 #time_series_validation_fold_split_datetime_boundaries = ""
2119
2120 # Timeout in seconds for time-series properties detection in UI.
2121 #timeseries_split_suggestion_timeout = 30.0
2122
2123 # Whether to use lag transformers when using causal-split for validation (as occurs when not using time-based lag recipe).
2124 # If no time groups columns, lag transformers will still use time-column as sole time group column.
2125 #
2126 #use_lags_if_not_time_series_recipe = false
2127
2128 # earliest datetime for automatic conversion of integers in %Y%m%d format to a time column during parsing
2129 #min_ymd_timestamp = 19000101
2130
2131 # lastet datetime for automatic conversion of integers in %Y%m%d format to a time column during parsing
2132 #max_ymd_timestamp = 21000101
2133
2134 # maximum number of data samples (randomly selected rows) for date/datetime format detection
2135 #max_rows_datetime_format_detection = 100000
2136
2137 # Whether to use datetime cache
2138 #use_datetime_cache = true
2139
2140 # Minimum amount of rows required to utilize datetime cache
2141 #datetime_cache_min_rows = 10000
2142
2143 # Automatically generate is-holiday features from date columns
2144 #holiday_features = true
2145
2146 #holiday_country = ""
2147
2148 # Country code(s) to use to look up holiday calendar (Python package 'holidays')
2149 #holiday_countries = ["UnitedStates", "UnitedKingdom", "EuropeanCentralBank", "Germany", "Mexico", "Japan"]"
2150
2151 # Max. sample size for automatic determination of time series train/valid split properties, only if time column is selected
2152 #max_time_series_properties_sample_size = 250000
2153
2154 # Maximum number of lag sizes to use for lags-based time-series experiments. are sampled from if sample_lag_sizes==True, else all are taken (-1 == ↵
2155 #automatic)
2156 #max_lag_sizes = 30
2157
2158 # Minimum required autocorrelation threshold for a lag to be considered for feature engineering
2159 #min_lag_autocorrelation = 0.1
2160
2161 # How many samples of lag sizes to use for a single time group (single time series signal)
2162 #max_signal_lag_sizes = 100
2163
2164 # Whether to sample lag sizes
2165 #sample_lag_sizes = false
2166
2167 # How many samples of lag sizes to use, chosen randomly out of original set of lag sizes
2168 #max_sampled_lag_sizes = 10
2169
2170 # Override lags to be used
2171 # e.g. [7, 14, 21] # this exact list
2172 # e.g. 21 # produce from 1 to 21
2173 # e.g. 21:3 produce from 1 to 21 in step of 3
2174 # e.g. 5-21 produce from 5 to 21

```

(continues on next page)

(continued from previous page)

```

2173 # e.g. 5-21:3 produce from 5 to 21 in step of 3
2174 #override_lag_sizes = []
2175
2176 # Smallest considered lag size
2177 #min_lag_size = -1
2178
2179 # Whether to enable feature engineering based on selected time column, e.g. Date~weekday.
2180 #allow_time_column_as_feature = true
2181
2182 # Whether to enable integer time column to be used as a numeric feature.
2183 # If using time series recipe, using time column (numeric time stamps) as input features can lead to model that
2184 # memorizes the actual time stamps instead of features that generalize to the future.
2185 #
2186 #allow_time_column_as_numeric_feature = false
2187
2188 # Allowed date or date-time transformations.
2189 # Date transformers include: year, quarter, month, week, weekday, day, dayofyear, num.
2190 # Date transformers also includes: hour, minute, second.
2191 # Features in DAI will show up as get_+ transformation name.
2192 # E.g. num is a direct numeric value representing the floating point value of time,
2193 # which can lead to over-fitting if used on IID problems. So this is turned off by default.
2194 #datetime_funcs = ['year', 'quarter', 'month', 'week', 'weekday', 'day', 'dayofyear', 'hour', 'minute', 'second']
2195
2196 # Whether to consider time groups columns (tgc) as standalone features.
2197 # Note that 'time_column' is treated separately via 'Allow to engineer features from time column'.
2198 # Use allowed_coltypes_for_tgc_as_features for control per feature type.
2199 #
2200 #allow_tgc_as_features = false
2201
2202 # Which time groups columns (tgc) feature types to consider as standalone features,
2203 # if the corresponding flag "Consider time groups columns as standalone features" is set to true.
2204 # E.g. all column types would be ["numeric", "categorical", "ohe_categorical", "datetime", "date", "text"]
2205 # Note that 'time_column' is treated separately via 'Allow to engineer features from time column'.
2206 # Note that if lag-based time series recipe is disabled, then all tgc are allowed features.
2207 #
2208 #allowed_coltypes_for_tgc_as_features = ['numeric', 'categorical', 'ohe_categorical', 'datetime', 'date', 'text']
2209
2210 # Whether various transformers (clustering, truncated SVD) are enabled,
2211 # that otherwise would be disabled for time series due to
2212 # potential to overfit by leaking across time within the fit of each fold.
2213 #enable_time_unaware_transformers = "auto"
2214
2215 # Whether to group by all time groups columns for creating lag features, instead of sampling from them
2216 #tgc_only_use_all_groups = true
2217
2218 # Enable creation of holdout predictions on training data
2219 # using moving windows (useful for MLI, but can be slow)
2220 #time_series_holdout_preds = true
2221
2222 # Set fixed number of time-based splits for internal model validation (actual number of splits allowed can be less and is determined at experiment run- ↴time).
2223 #time_series_validation_splits = -1
2224
2225 # Maximum overlap between two time-based splits. Higher values increase the amount of possible splits.
2226 #time_series_splits_max_overlap = 0.5
2227
2228 # Max number of splits used for creating final time-series model's holdout/backtesting predictions. With the default value '-1' the same amount of splits ↴as during model validation will be used. Use 'time_series_validation_splits' to control amount of time-based splits used for model validation.
2229 #time_series_max_holdout_splits = -1
2230
2231 #single_model_vs_cv_score_reldiff = 0.05
2232
2233 #single_model_vs_cv_score_reldiff2 = 0.0
2234
2235 # Whether to speed up time-series holdout predictions for back-testing on training data (used for MLI and metrics calculation). Can be slightly less ↴accurate.
2236 #mli_ts_fast_approx = false
2237
2238 # Whether to speed up Shapley values for time-series holdout predictions for back-testing on training data (used for MLI). Can be slightly less accurate.
2239 #mli_ts_fast_approx_contribs = true
2240
2241 # Enable creation of Shapley values for holdout predictions on training data
2242 # using moving windows (useful for MLI, but can be slow), at the time of the experiment. If disabled, MLI will
2243 # generate Shapley values on demand.
2244 #mli_ts_holdout_contribs = true
2245
2246 # Values of 5 or more can improve generalization by more aggressive dropping of least important features. Set to 1 to disable.
2247 #time_series_min_interpretability = 5
2248
2249 # Dropout mode for lag features in order to achieve an equal n.a.-ratio between train and validation/test. The independent mode performs a simple feature- ↴wise dropout, whereas the dependent one takes lag-size dependencies per sample/row into account.
2250 #lags_dropout = "dependent"
2251
2252 # Normalized probability of choosing to lag non-targets relative to targets (-1.0 = auto)
2253 #prob_lag_non_targets = -1.0
2254
2255 # Method to create rolling test set predictions, if the forecast horizon is shorter than the time span of the test set. One can choose between test time ↴augmentation (TTA) and a successive refitting of the final pipeline.
2256 #rolling_test_method = "ttta"
2257
2258 # Rolling test method setting of rolling_test_method used for validation set during genetic algorithm (True).
2259 #use_rolling_test_method_for_ga = true
2260
2261 # Probability for new Lags/EWMA gene to use default lags (determined by frequency/gap/horizon, independent of data) (-1.0 = auto)
2262 #prob_default_lags = -1.0
2263
2264 # Unnormalized probability of choosing other lag time-series transformers based on interactions (-1.0 = auto)
2265 #prob_lagsinteraction = -1.0
2266
2267 # Unnormalized probability of choosing other lag time-series transformers based on aggregations (-1.0 = auto)
2268 #prob_lagsaggregates = -1.0
2269
2270 # Time series centering or detrending transformation. The free parameter(s) of the trend model are fitted and the trend is removed from the target signal, ↴and the pipeline is fitted on the residuals. Predictions are made by adding back the trend. The robust centering or linear detrending variants use ↴PANSAC to achieve a higher tolerance w.r.t. outliers. The Epidemic target transformer uses the SEIR model: https://en.wikipedia.org/wiki/Compartmental ↴models_in_epidemiology#The_SEIR_model

```

(continues on next page)

(continued from previous page)

```

2271 #ts_target_trafo = "none"
2272
2273 # Dictionary to control Epidemic SEIRD model for de-trending of target per time series group.
2274 # Note: The target column must correspond to I(t), the infected cases as a function of time.
2275 # For each training split and time series group, the SEIRD model is fitted to the target signal (by optimizing
2276 # the free parameters shown below for each time series group).
2277 # Then, the SEIRD model's value is subtracted from the training response, and the residuals are passed to
2278 # the feature engineering and modeling pipeline. For predictions, the SEIRD model's value is added to the residual
2279 # predictions from the pipeline, for each time series group.
2280 # Note: Careful selection of the bounds for the free parameters N, beta, gamma, delta, alpha, rho, lockdown,
2281 # beta_decay, beta_decay_rate is extremely important for good results.
2282 # # S(t) : susceptible/healthy/not immune
2283 # # E(t) : exposed/not yet infectious
2284 # # I(t) : infectious/active <= target column
2285 # # R(t) : recovered/immune
2286 # # D(t) : deceased
2287 # # Free parameters:
2288 # # N : total population, N=S+E+I+R+D
2289 # # beta : rate of exposure (S -> E)
2290 # # gamma : rate of recovering (I -> R)
2291 # # delta : incubation period
2292 # # alpha : fatality rate
2293 # # rho : rate at which people die
2294 # # lockdown : day of lockdown (-1 => no lockdown)
2295 # # beta_decay : beta decay due to lockdown
2296 # # beta_decay_rate : speed of beta decay
2297 # # Dynamics:
2298 # if lockdown >= 0:
2299 # beta_min = beta * (1 - beta_decay)
2300 # beta = (beta - beta_min) / (1 + np.exp(-beta_decay_rate * (-t + lockdown))) + beta_min
2301 # dSdt = -beta * S * I / N
2302 # dEdt = beta * S * I / N - delta * E
2303 # dIdt = delta * E - (1 - alpha) * gamma * I - alpha * rho * I
2304 # dRdt = (1 - alpha) * gamma * I
2305 # dDdt = alpha * rho * I
2306 # Provide lower/upper bounds for each parameter you want to control the bounds for. Valid parameters are:
2307 # N_min, N_max, beta_min, beta_max, gamma_min, gamma_max, delta_min, delta_max, alpha_min, alpha_max,
2308 # rho_min, rho_max, lockdown_min, lockdown_max, beta_decay_min, beta_decay_max,
2309 # beta_decay_rate_min, beta_decay_rate_max. You can change any subset of parameters, e.g.,
2310 # ts_target_trafo_epidemic_params_dict={"N_min": 1000, "beta_max": 0.2}"
2311 # To get SEIR model (in cases where death rates are very low, can speed up calculations significantly):
2312 # set alpha_min=alpha_max=rho_min=rho_max=beta_decay_rate_min=beta_decay_rate_max=0, lockdown_min=lockdown_max=-1.
2313 #
2314 #ts_target_trafo_epidemic_params_dict = "{}"
2315
2316 #ts_target_trafo_epidemic_target = "I"
2317
2318 # Time series lag-based target transformation. One can choose between difference and ratio of the current and a lagged target target. The corresponding lag_size can be set via 'Target transformation lag size'.
2319 #ts_lag_target_trafo = "none"
2320
2321 # Lag size used for time series target transformation. See setting 'Time series target transformation'.
2322 #ts_target_trafo_lag_size = -1
2323
2324 # Maximum amount of columns send from UI to backend in order to auto-detect TGC
2325 #tgc_via_ui_max_ncols = 10
2326
2327 # Maximum frequency of duplicated timestamps for TGC detection
2328 #tgc_dup_tolerance = 0.01
2329
2330 # Build (if missing but available) the MOJO pipeline to be used for predictions in MLI. For certain models this can improve performance.
2331 #mli_use_mojo_pipeline = true
2332
2333 # When number of rows are above this limit sample for MLI for scoring UI data
2334 #mli_sample_above_for_scoring = 1000000
2335
2336 # When number of rows are above this limit sample for MLI for training surrogate models
2337 #mli_sample_above_for_training = 100000
2338
2339 # When sample for MLI how many rows to sample
2340 #mli_sample_size = 100000
2341
2342 # how many bins to do quantile binning
2343 #mli_num_quantiles = 10
2344
2345 # mli random forest number of trees
2346 #mli_drf_num_trees = 100
2347
2348 # Whether to speed up predictions used inside MLI with a fast approximation
2349 #mli_fast_approx = true
2350
2351 # mli number of trees for fast_approx during predict for Shapley
2352 #fast_approx_num_trees = 50
2353
2354 # whether to do only 1 fold and 1 model of all folds and models if ensemble
2355 #fast_approx_do_one_fold_one_model = true
2356
2357 # Maximum number of interpreters status cache entries.
2358 #mli_interpreter_status_cache_size = 1000
2359
2360 # mli random forest max depth
2361 #mli_drf_max_depth = 20
2362
2363 # not only sample training, but also sample scoring
2364 #mli_sample_training = true
2365
2366 # regularization strength for k-LIME GLM's
2367 #klime_lambda = "[1e-06, 1e-08]"
2368
2369 # regularization strength for k-LIME GLM's
2370 #klime_alpha = 0.0
2371
2372 # mli converts numeric columns to enum when cardinality is <= this value
2373 #mli_max_numeric_enum_cardinality = 25

```

(continues on next page)

(continued from previous page)

```

2374
2375 # Maximum number of features allowed for k-LIME k-means clustering
2376 #mli_max_number_cluster_vars = 6
2377
2378 # Use all columns for k-LIME k-means clustering (this will override `mli_max_number_cluster_vars` if set to 'True')
2379 #use_all_columns_klime_kmeans = false
2380
2381 # Strict version check for MLI
2382 #mli_strict_version_check = true
2383
2384 # MLI cloud name
2385 #mli_cloud_name = "H2O-MLI-DAI"
2386
2387 # Compute original model ICE using per feature's bin predictions (true) or use "one frame" strategy (false)
2388 #mli_ice_per_bin_strategy = false
2389
2390 # By default DIA will run for categorical columns with cardinality <= mli_dia_default_max_cardinality
2391 #mli_dia_default_max_cardinality = 10
2392
2393 # When number of rows are above this limit, then sample for MLI Shapley calculation
2394 #mli_shapley_sample_size = 100000
2395
2396 # Enable MLI Sensitivity Analysis
2397 #enable_mli_sa = true
2398
2399 # Enable MLI sequential task execution (interpretation tasks are run in parallel otherwise)
2400 #mli_sequential_task_execution = true
2401
2402 # When number of rows are above this limit, then sample for DIA
2403 #mli_dia_sample_size = 100000
2404
2405 # When number of rows are above this limit, then sample for DAI PD/ICE
2406 #mli_pd_sample_size = 100000
2407
2408 # Use dynamic switching between PD numeric and categorical binning and UI chart selection in case of features which were used both as numeric and categorical by auto ML
2409 #mli_pd_numcat_num_chart = false
2410
2411 # If 'mli_pd_numcat_num_chart' is enabled, then use numeric binning and chart if feature unique values count is bigger than threshold, else use categorical binning and chart
2412 #mli_pd_numcat_threshold = 10
2413
2414 # In New Interpretation screen show only datasets which can be used to explain a selected model. This can slow down the server significantly.
2415 #new_mli_list_only_explainable_datasets = false
2416
2417 # Enable async/await-based non-blocking MLI API
2418 #enable_mli_async_api = true
2419
2420 # Enable main chart aggregator in Sensitivity Analysis
2421 #enable_mli_sa_main_chart_aggregator = true
2422
2423 # When to sample for Sensitivity Analysis (number of rows after sampling)
2424 #mli_sa_sampling_limit = 500000
2425
2426 # Run main chart aggregator in Sensitivity Analysis when the number of dataset instances is bigger than given limit
2427 #mli_sa_main_chart_aggregator_limit = 1000
2428
2429 # Use predict_safe() (true) or predict_base() (false) in MLI (PD, ICE, SA, ...)
2430 #mli_predict_safe = false
2431
2432 # Allow predict method with fast approximation in MLI (PD, ICE, SA, ...)
2433 #enable_mli_predict_fast_approx = false
2434
2435 # Number of max retries should the surrogate model fail to build.
2436 #mli_max_surrogate_retries = 5
2437
2438 # Number of rows per batch when scoring using MOJO.
2439 #mli_mojo_batch_size = 50
2440
2441 # Number of parallel workers when scoring using MOJO in Kernel Explainer.
2442 #mli_kernel_explainer_workers = 4
2443
2444 # Use Kernel Explainer to obtain Shapley values for original features.
2445 #mli_run_kernel_explainer = false
2446
2447 # Sample input dataset for Kernel Explainer.
2448 #mli_kernel_explainer_sample = true
2449
2450 # Sample size for input dataset passed to Kernel Explainer.
2451 #mli_kernel_explainer_sample_size = 1000
2452
2453 # 'auto' or int. Number of times to re-evaluate the model when explaining each prediction. More samples lead to lower variance estimates of the SHAP values.
2454 # The 'auto' setting uses nsamples = 2 * X.shape[1] + 2048. This setting is disabled by default and DAI determines the right number internally.
2455 #mli_kernel_explainer_nsamples = ""
2456
2457 # 'num_features(int)', 'auto' (default for now, but deprecated), 'aic', 'bic', or float. The l1 regularization to use for feature selection (the estimation procedure is based on a debiased lasso). The 'auto' option currently uses aic when less than 20% of the possible sample space is enumerated, otherwise it uses no regularization. THE BEHAVIOR OF 'auto' WILL CHANGE in a future version to be based on 'num_features' instead of AIC. The aic and bic options use the AIC and BIC rules for regularization. Using 'num_features(int)' selects a fix number of top features. Passing a float directly sets the alpha parameter of the sklearn.linear_model.Lasso model used for feature selection.
2458 #mli_kernel_explainer_l1_reg = "auto"
2459
2460 # Tokenizer used to extract tokens from text columns for MLI.
2461 #mli_nlp_tokenizer = "tfidf"
2462
2463 # Number of tokens used for MLI NLP explanations. -1 means all.
2464 #mli_nlp_top_n = 20
2465
2466 # Maximum number of records on which we'll perform MLI NLP
2467 #mli_nlp_sample_limit = 10000
2468
2469 # Number of parallel workers when scoring using MOJO in MLI NLP.
2470 #mli_nlp_workers = 4

```

(continues on next page)

(continued from previous page)

```

2471 # Minimum number of documents in which token has to appear. Integer mean absolute count, float means percentage.
2472 #mli_nlp_min_df = 3
2473
2474 # Maximum number of documents in which token has to appear. Integer mean absolute count, float means percentage.
2475 #mli_nlp_max_df = 0.9
2476
2477 # The minimum value in the ngram range. The tokenizer will generate all possible tokens in the (mli_nlp_min_ngram, mli_nlp_max_ngram) range.
2478 #mli_nlp_min_ngram = 1
2479
2480 # The maximum value in the ngram range. The tokenizer will generate all possible tokens in the (mli_nlp_min_ngram, mli_nlp_max_ngram) range.
2481 #mli_nlp_max_ngram = 1
2482
2483 # Mode used to choose N tokens for MLI NLP.
2484 # "top" chooses N top tokens.
2485 # "bottom" chooses N bottom tokens.
2486 # "top-bottom" chooses math.floor(N/2) top and math.ceil(N/2) bottom tokens.
2487 # "linspace" chooses N evenly spaced out tokens.
2488 #mli_nlp_min_token_mode = "top"
2489
2490 # The number of top tokens to be used as features when building token based feature importance.
2491 #mli_nlp_tokenizer_max_features = -1
2492
2493 # The number of top tokens to be used as features when computing text LOCO.
2494 #mli_nlp_loco_max_features = -1
2495
2496 # The number of top tokens to be used as features when building surrogate models.
2497 #mli_nlp_surrogate_tokens = 100
2498
2499 # Whether to dump every scored individual's variable importance (both derived and original) to csv/tabulated/json file
2500 # produces files like: individual_scored_id%d.iter%d*features*
2501 #dump_varimp_every_scored_indiv = false
2502
2503 # Whether to dump every scored individual's model parameters to csv/tabulated/json file
2504 # produces files like: individual_scored.params.[txt, csv, json]
2505 #dump_modelparams_every_scored_indiv = true
2506
2507 # Number of features to show in model dump every scored individual
2508 #dump_modelparams_every_scored_indiv_feature_count = 3
2509
2510 # Whether to append (false) or have separate files, files like: individual_scored_id%d.iter%d*params*, (true) for modelparams every scored indiv
2511 #dump_modelparams_separate_files = false
2512
2513 # Whether to dump every scored fold's timing and feature info to a *timings*.txt file
2514 #dump_trans_timings = false
2515
2516 # whether to delete preview timings if wrote transformer timings
2517 #delete_preview_trans_timings = true
2518
2519 # Location of the AutoDoc template
2520 #autodoc_template = "report_template.docx"
2521
2522 # Specify the output of the report.
2523 # Options are 'docx' or 'md'.
2524 #autodoc_output_type = "docx"
2525
2526 # Specify the type of sub-templates to use. Set to '' to
2527 # copy *autodoc_output_type*. Options are 'docx', 'md' or ''.
2528 #autodoc_subtemplate_type = ""
2529
2530 # Specify the name of the report.
2531 #autodoc_report_name = "report"
2532
2533 # Specify the maximum number of classes in the confusion
2534 # matrix.
2535 #autodoc_max_cm_size = 10
2536
2537 # Set the number of models for which a glm coefficients
2538 # table is shown in the Autoreport. coef_table_num_models must
2539 # be -1 or an integer >= 1 (-1 shows all models).
2540 #autodoc_coef_table_num_models = 1
2541
2542 # Set the number of folds per model for which a glm
2543 # coefficients table is shown in the Autoreport. coef_table_num_folds
2544 # must be -1 or an integer >= 1 (-1 shows all folds per model).
2545 #autodoc_coef_table_num_folds = -1
2546
2547 # Set the number of coefficients to show within a glm
2548 # coefficients table in the Autoreport. coef_table_num_coeff, controls
2549 # the number of rows shown in a glm table and must be -1 or
2550 # an integer >= 1 (-1 shows all coefficients).
2551 #autodoc_coef_table_num_coeff = 50
2552
2553 # Set the number of classes to show within a glm
2554 # coefficients table in the Autoreport. coef_table_num_classes controls
2555 # the number of class-columns shown in a glm table and must be -1 or
2556 # an integer >= 4 (-1 shows all classes).
2557 #autodoc_coef_table_num_classes = 9
2558
2559 # Specify whether to show the full glm coefficient
2560 # table(s) in the appendix. coef_table_appendix_results_table must be
2561 # a boolean: True to show tables in appendix, False to not show them.
2562 #autodoc_coef_table_appendix_results_table = false
2563
2564 # Specify the minimum relative importance in order
2565 # for a feature to be displayed. autodoc_min_relative_importance
2566 # must be a float >= 0 and < 1.
2567 #autodoc_min_relative_importance = 0.003
2568
2569 # Specify the number of top features to display in
2570 # the document. setting to -1 disables this restriction
2571 #autodoc_num_features = 50
2572
2573 # Specify the number of rows to include in PDP and ICE plot
2574 # if individual rows are not specified.

```

(continues on next page)

(continued from previous page)

```

2575 #autodoc_num_rows = 0
2576
2577 # Maximum number of seconds Partial Dependency computation
2578 # can take when generating report. Set to -1 for no time limit.
2579 #autodoc_pd_max_runtime = 20
2580
2581 # Number of standard deviations outside of the range of
2582 # a column to include in partial dependence plots. This shows how the
2583 # model will react to data it has not seen before.
2584 #autodoc_out_of_range = 3
2585
2586 # Number of columns to be show in data summary. Value
2587 # must be an integer. Values lower than 1, f.e. 0 or -1, indicate that
2588 # all columns should be shown.
2589 #autodoc_data_summary_col_num = -1
2590
2591 # Whether to include prediction statistics information if
2592 # experiment is binary classification/regression.
2593 #autodoc_prediction_stats = false
2594
2595 # Number of quantiles to use for prediction statistics
2596 # computation.
2597 #autodoc_prediction_stats_n_quantiles = 20
2598
2599 # Whether to include population stability index if
2600 # experiment is binary classification/regression.
2601 #autodoc_population_stability_index = false
2602
2603 # Number of quantiles to use for population stability index
2604 # computation.
2605 #autodoc_population_stability_index_n_quantiles = 10
2606
2607 # Whether to compute permutation based feature importance.
2608 #autodoc_include_permutation_feature_importance = false
2609
2610 # Name of the scorer to be used to calculate feature
2611 # importance. Leave blank to use experiments default scorer
2612 #autodoc_feature_importance_scorer = ""
2613
2614 # Number of permutations to make per feature when computing
2615 # feature importance.
2616 #autodoc_feature_importance_num_perm = 1
2617
2618 # Whether to include response rates information if
2619 # experiment is binary classification.
2620 #autodoc_response_rate = false
2621
2622 # Number of quantiles to use for response rates information
2623 # computation.
2624 #autodoc_response_rate_n_quantiles = 10
2625
2626 # The number feature in a KLIME global GLM coefficients
2627 # table. Must be an integer greater than 0 or -1. To
2628 # show all features set to -1.
2629 #autodoc_global_klime_num_features = 10
2630
2631 # Set the number of KLIME global GLM coefficients tables. Set
2632 # to 1 to show one table with coefficients sorted by absolute
2633 # value. Set to 2 to two tables one with the top positive
2634 # coefficients and one with the top negative coefficients. Must
2635 # be set to the integer 1 or 2.
2636 #autodoc_global_klime_num_tables = 1
2637
2638 # Whether to show the Gini Plot.
2639 #autodoc_gini_plot = false
2640
2641 # Whether to show all config settings. If False, only
2642 # the changed settings (config overrides) are listed, otherwise all
2643 # settings are listed.
2644 #autodoc_list_all_config_settings = false
2645
2646 # Line length of the keras model architecture summary. Must
2647 # be an integer greater than 0 or -1. To use the default line length set
2648 # value -1.
2649 #autodoc_keras_summary_line_length = -1
2650
2651 # Maximum number of lines shown for advanced transformer
2652 # architecture in the Feature section. Note that the full architecture
2653 # can be found in the Appendix.
2654 #autodoc_transformer_architecture_max_lines = 30
2655
2656 # Show full NLP/Image transformer architecture in
2657 # the Appendix.
2658 #autodoc_full_architecture_in_appendix = false
2659
2660 # Maximum number of columns autoviz will work with.
2661 # If dataset has more columns than this number,
2662 # autoviz will pick columns randomly, prioritizing numerical columns
2663 #
2664 #autoviz_max_num_columns = 50
2665
2666 #
2667 # Whether to compute training, validation, and test correlation matrix (table and heatmap pdf) and save to disk
2668 # alpha: currently single threaded and slow for many columns
2669 #compute_correlation = false
2670
2671 # Whether to dump to disk a correlation heatmap
2672 #produce_correlation_heatmap = false
2673
2674 # Value to report high correlation between original features
2675 #high_correlation_value_to_report = 0.95
2676
2677 # Whether to delete preview cache on server exit
2678 #preview_cache_upon_server_exit = true

```

(continues on next page)

(continued from previous page)

```

2679 # Configurations for a HDFS data source
2680 # Path of hdfs coresite.xml
2681 # core_site_xml_path is deprecated, please use hdfs_config_path
2682 #core_site_xml_path = ""
2683
2684 # (Required) HDFS config folder path. Can contain multiple config files.
2685 #hdfs_config_path = ""
2686
2687 # Path of the principal key tab file. Required when hdfs_auth_type='principal'.
2688 # key_tab_path is deprecated, please use hdfs_keytab_path
2689 #
2690 #key_tab_path = ""
2691
2692 # Path of the principal key tab file. Required when hdfs_auth_type='principal'.
2693 #
2694 #hdfs_keytab_path = ""
2695
2696 # The option disable access to DAI data_directory from file browser
2697 #file_hide_data_directory = true
2698
2699 # Enable usage of path filters
2700 #file_path_filtering_enabled = false
2701
2702 # List of absolute path prefixes to restrict access to in file system browser.
2703 # First add the following environment variable to your command line to enable this feature:
2704 # file_path_filtering_enabled=true
2705 # This feature can be used in the following ways (using specific path or using logged user's directory):
2706 # file_path_filter_include=['/data/stage']
2707 # file_path_filter_include=['/data/stage','/data/prod']
2708 # file_path_filter_include=/home/{{DAI_USERNAME}}/
2709 # file_path_filter_include=['/home/{{DAI_USERNAME}}/','/data/stage','/data/prod']
2710 #
2711 #
2712 #file_path_filter_include = []
2713
2714 # (Required) HDFS connector
2715 # Specify HDFS Auth Type, allowed options are:
2716 # noauth : (default) No authentication needed
2717 # principal : Authenticate with HDFS with a principal user (DEPRECATED - use 'keytab' auth type)
2718 # keytab : Authenticate with a Key tab (recommended). If running
2719 # DAI as a service, then the Kerberos keytab needs to
2720 # be owned by the DAI user.
2721 # keytabimpersonation : Login with impersonation using a keytab
2722 #hdfs_auth_type = "noauth"
2723
2724 # Kerberos app principal user. Required when hdfs_auth_type='keytab'; recommended otherwise.
2725 #hdfs_app_principal_user = ""
2726
2727 # Deprecated - Do Not Use, login user is taken from the user name from login
2728 #hdfs_app_login_user = ""
2729
2730 # JVM args for HDFS distributions, provide args seperate by space
2731 # -Djava.security.krb5.conf=cpath>/krb5.conf
2732 # -Dsun.security.krb5.debug=True
2733 # -Dlog4j.configuration=file:///<path>log4j.properties
2734 #hdfs_app_jvm_args = ""
2735
2736 # hdfs class path
2737 #hdfs_app_classpath = ""
2738
2739 # List of supported DFS schemas. Ex. "[hdfs://, maprfs://, swift://]"
2740 # Supported schemas list is used as an initial check to ensure valid input to connector
2741 #
2742 #hdfs_app_supported_schemas = "[hdfs://, maprfs://, swift://]"
2743
2744 # Maximum number of files viewable in connector ui. Set to larger number to view more files
2745 #hdfs_max_files_listed = 100
2746
2747 # Blue Data DTap connector settings are similar to HDFS connector settings.
2748 # Specify DTap Auth Type, allowed options are:
2749 # noauth : No authentication needed
2750 # principal : Authenticate with DTab with a principal user
2751 # keytab : Authenticate with a Key tab (recommended). If running
2752 # DAI as a service, then the Kerberos keytab needs to
2753 # be owned by the DAI user.
2754 # keytabimpersonation : Login with impersonation using a keytab
2755 # NOTE: "hdfs_app_classpath" and "core_site_xml_path" are both required to be set for DTap connector
2756 #dtap_auth_type = "noauth"
2757
2758 # Dtap (HDFS) config folder path , can contain multiple config files
2759 #dtap_config_path = ""
2760
2761 # Path of the principal key tab file, dtap_key_tab_path is deprecated. Please use dtap_keytab_path
2762 #dtap_key_tab_path = ""
2763
2764 # Path of the principal key tab file
2765 #dtap_keytab_path = ""
2766
2767 # Kerberos app principal user (recommended)
2768 #dtap_app_principal_user = ""
2769
2770 # Specify the user id of the current user here as user@realm
2771 #dtap_app_login_user = ""
2772
2773 # JVM args for DTap distributions, provide args seperate by space
2774 #dtap_app_jvm_args = ""
2775
2776 # DTap (HDFS) class path. NOTE: set 'hdfs_app_classpath' also
2777 #dtap_app_classpath = ""
2778
2779 # S3 Connector credentials
2780 #aws_access_key_id = ""
2781
2782 # S3 Connector credentials

```

(continues on next page)

(continued from previous page)

```

2783 #aws_secret_access_key = ""
2784
2785 # S3 Connector credentials
2786 #aws_role_arn = ""
2787
2788 # What region to use when none is specified in the s3 url.
2789 # Ignored when aws_s3_endpoint_url is set.
2790 #
2791 #aws_default_region = ""
2792
2793 # Sets endpoint URL that will be used to access S3.
2794 #aws_s3_endpoint_url = ""
2795
2796 # If set to true S3 Connector will try to obtain credentials associated with
2797 # the role attached to the EC2 instance.
2798 #aws_use_ec2_role_credentials = false
2799
2800 # Starting S3 path displayed in UI S3 browser
2801 #s3_init_path = "s3://"
2802
2803 # GCS Connector credentials
2804 # example (recommended) -- '/licenses/my_service_account_json'
2805 #gcs_path_to_service_account_json = ""
2806
2807 # Minio Connector credentials
2808 #minio_endpoint_url = ""
2809
2810 # Minio Connector credentials
2811 #minio_access_key_id = ""
2812
2813 # Minio Connector credentials
2814 #minio_secret_access_key = ""
2815
2816 # Recommended Provide: url, user, password
2817 # Optionally Provide: account, user, password
2818 # Example URL: https://<snowflake_account>.<region>.snowflakecomputing.com
2819 # Snowflake Connector credentials
2820 #snowflake_url = ""
2821
2822 # Snowflake Connector credentials
2823 #snowflake_user = ""
2824
2825 # Snowflake Connector credentials
2826 #snowflake_password = ""
2827
2828 # Snowflake Connector credentials
2829 #snowflake_account = ""
2830
2831 # KDB Connector credentials
2832 #kdb_user = ""
2833
2834 # KDB Connector credentials
2835 #kdb_password = ""
2836
2837 # KDB Connector credentials
2838 #kdb_hostname = ""
2839
2840 # KDB Connector credentials
2841 #kdb_port = ""
2842
2843 # KDB Connector credentials
2844 #kdb_app_classpath = ""
2845
2846 # KDB Connector credentials
2847 #kdb_app_jvm_args = ""
2848
2849 # Azure Blob Store Connector credentials
2850 #azure_blob_account_name = ""
2851
2852 # Azure Blob Store Connector credentials
2853 #azure_blob_account_key = ""
2854
2855 # Azure Blob Store Connector credentials
2856 #azure_connection_string = ""
2857
2858 # Configuration for JDBC Connector.
2859 # JSON/Dictionary String with multiple keys.
2860 # Format as a single line without using carriage returns (the following example is formatted for readability).
2861 # Use triple quotations to ensure that the text is read as a single string.
2862 # Example:
2863 # '''
2864 # "postgres": {
2865 #   "url": "jdbc:postgresql://ip address:port/postgres",
2866 #   "jarpath": "/path/to/postgres_driver.jar",
2867 #   "classpath": "org.postgresql.Driver"
2868 # },
2869 # "mysql": {
2870 #   "url": "mysql connection string",
2871 #   "jarpath": "/path/to/mysql_driver.jar",
2872 #   "classpath": "my.sql.classpath.Driver"
2873 # }
2874 # }'
2875 #
2876 #jdbc_app_configs = "{}"
2877
2878 # extra jvm args for jdbc connector
2879 #jdbc_app_jvm_args = "-Xmx4g"
2880
2881 # alternative classpath for jdbc connector
2882 #jdbc_app_classpath = ""
2883
2884 # Configuration for Hive Connector.
2885 # Note that inputs are similar to configuring HDFS connectivity.
2886 # important keys:

```

(continues on next page)

(continued from previous page)

```

2887 # * hive_conf_path - path to hive configuration, may have multiple files. typically: hive-site.xml, hdfs-site.xml, etc
2888 # * auth_type - one of 'noauth', 'keytab', 'keytabimpersonation' for kerberos authentication
2889 # * keytab_path - path to the kerberos keytab to use for authentication, can be "" if using 'noauth' auth_type
2890 # * principal_user - Kerberos app principal user. Required when using auth_type 'keytab' or 'keytabimpersonation'
2891 # JSON/Dictionary String with multiple keys. Example:
2892 # {
2893 #   "hive_connection_1": {
2894 #     "hive_conf_path": "/path/to/hive/conf",
2895 #     "auth_type": "one of ['noauth', 'keytab', 'keytabimpersonation']",
2896 #     "keytab_path": "/path/to/<filename>.keytab",
2897 #     "principal_user": "hive/LOCALHOST@H2O.AI",
2898 #   },
2899 #   "hive_connection_2": {
2900 #     "hive_conf_path": "/path/to/hive/conf_2",
2901 #     "auth_type": "one of ['noauth', 'keytab', 'keytabimpersonation']",
2902 #     "keytab_path": "/path/to/<filename_2>.keytab",
2903 #     "principal_user": "my_user/LOCALHOST@H2O.AI",
2904 #   }
2905 # }
2906 #
2907 #hive_app_configs = "{}"
2908 #
2909 # Extra jvm args for hive connector. Provide args separated by a space.
2910 # Notes regarding default jvm args:
2911 # * -Djavax.security.auth.useSubjectCredsOnly=false -- setting required for kerberos authentication + impersonation
2912 # * -Djava.security.auth.login.config=/etc/dai/jaas.conf -- setting required to allow underlying connector process
2913 # to adopt kerberos login properties as defined in the file /etc/dai/jaas.conf
2914 # -- Note: user must create the 'jaas.conf' and place it in the specified directory
2915 # * Example /etc/dai/jaas.conf (result of `cat /etc/dai/jaas.conf`):
2916 # com.sun.security.jgss.initiate {
2917 #   com.sun.security.auth.module.Krb5LoginModule required
2918 #   useKeyTab=true
2919 #   useTicketCache=false
2920 #   principal="hive/LOCALHOST@H2O.AI"
2921 #   doNotPrompt=true
2922 #   keyTab="/path/to/<filename>.keytab"
2923 #   debug=true;
2924 # };
2925 # -- Note: principal and keytab settings should be the same as configurations above for hive_app_configs.
2926 # and are the ONLY settings that need to be changed for DAI Hive connector to function.
2927 #
2928 #hive_app_jvm_args = "-Xmx4g -Djavax.security.auth.useSubjectCredsOnly=false -Djava.security.auth.login.config=/etc/dai/jaas.conf"
2929 #
2930 # Alternative classpath for hive connector. Can be used to add additional jar files to classpath.
2931 #hive_app_classpath = ""
2932 #
2933 # Notification scripts
2934 # - the variable points to a location of script which is executed at given event in experiment lifecycle
2935 # - the script should have executable flag enabled
2936 # - use of absolute path is suggested
2937 # The on experiment start notification script location
#listeners_experiment_start = ""
2938 #
2939 # The on experiment finished notification script location
#listeners_experiment_done = ""
2940 #
2941 # Address of the H2O Storage endpoint. Keep empty to use the local storage only.
#h2o_storage_address = ""
2942 #
2943 # Whether to use remote projects stored in H2O Storage instead of local projects.
#h2o_storage_projects_enabled = false
2944 #
2945 # Whether the channel to the storage should be encrypted.
#h2o_storage_tls_enabled = true
2946 #
2947 # Path to the certification authority certificate that H2O Storage server identity will be checked against.
#h2o_storage_tls_ca_path = ""
2948 #
2949 # Path to the client certificate to authenticate with H2O Storage server
#h2o_storage_tls_cert_path = ""
2950 #
2951 # Path to the client key to authenticate with H2O Storage server
#h2o_storage_tls_key_path = ""
2952 #
2953 # UUID of a Storage project to use instead of the remote HOME folder.
#h2o_storage_internal_default_project_id = ""
2954 #
2955 # Default AWS credentials to be used for scorer deployments.
#deployment_aws_access_key_id = ""
2956 #
2957 # Default AWS credentials to be used for scorer deployments.
#deployment_aws_secret_access_key = ""
2958 #
2959 # AWS S3 bucket to be used for scorer deployments.
#deployment_aws_bucket_name = ""
2960 #
2961 # Allow the browser to store e.g. login credentials in login form (set to false for higher security)
#allow_form_autocomplete = true
2962 #
2963 # Enable Projects workspace (alpha version, for evaluation)
#enable_projects = true
2964 #
2965 # Enable custom recipes.
#enable_custom_recipes = true
2966 #
2967 # Enable uploading of custom recipes.
#enable_custom_recipes_upload = true
2968 #
2969 # Enable downloading of custom recipes from external URL.
#enable_custom_recipes_from_url = true
2970 #
2971 # Include custom recipes in default inclusion lists (warning: enables all custom recipes)
#include_custom_recipes_by_default = false
2972 
```

(continues on next page)

(continued from previous page)

```
2991 # Default application language - options are 'en', 'ja', 'cn', 'ko'  
2992 app_language = "en"  
2993  
2994 # If true, Logout button is not visible in the GUI.  
2995 disablelogout = false  
2996  
2997 # Include byte order mark (BOM) when writing CSV files. Required to support UTF-8 encoding in Excel.  
2998 datatable_bom_csv = false  
2999  
3000 # Whether to check if config.toml keys are valid and fail if not valid  
3001 check_invalid_config_toml_keys = true  
3002  
3003 enable_funnel = true  
3004  
3005 clean_funnel = true  
3006  
3007 quiet_funnel = false  
3008  
3009 debug_daimodel_level = 0  
3010  
3011 # Amount of time to stall (in seconds) before killing the job (assumes it hung). Reference time is scaled by train data shape of rows * cols to get used  
3012 ↪ stalled_time_kill  
3013 stalled_time_kill_ref = 440.0  
3014  
3015 predict_safe_trials = 2  
3016  
3017 fit_safe_trials = 2  
3018  
3019 interaction_finder_max_rows_x_cols = 200000.0  
3020  
3021 interaction_finder_corr_threshold = 0.95  
3022  
3023 # Required GINI relative improvement for InteractionTransformer.  
3024 # If GINI is not better than this relative improvement compared to original features considered  
3025 # in the interaction, then the interaction is not returned. If noisy data, and no clear signal  
3026 # in interactions but still want interactions, then can decrease this number.  
3027 interaction_finder_gini_rel_improvement_threshold = 0.5  
3028  
3029 # Number of transformed Interactions to make as best out of many generated trial interactions.  
3030 interaction_finder_return_limit = 5  
3031  
3032 # Whether to enable bootstrap sampling. Provides error bars to validation and test scores based on the standard error of the bootstrap mean.  
3033 enable_bootstrap = true  
3034  
3035 # Minimum number of bootstrap samples to use for estimating score and its standard deviation  
3036 # Actual number of bootstrap samples will vary between the min and max,  
3037 # depending upon row count (more rows, fewer samples) and accuracy settings (higher accuracy, more samples)  
3038  
3039 min_bootstrap_samples = 1  
3040  
3041 # Maximum number of bootstrap samples to use for estimating score and its standard deviation  
3042 # Actual number of bootstrap samples will vary between the min and max,  
3043 # depending upon row count (more rows, fewer samples) and accuracy settings (higher accuracy, more samples)  
3044  
3045 max_bootstrap_samples = 100  
3046  
3047 # Minimum fraction of row size to take as sample size for bootstrap estimator  
3048 # Actual sample size used for bootstrap estimate will vary between the min and max,  
3049 # depending upon row count (more rows, smaller sample size) and accuracy settings (higher accuracy, larger sample size)  
3050  
3051 min_bootstrap_sample_size_factor = 1.0  
3052  
3053 # Maximum fraction of row size to take as sample size for bootstrap estimator  
3054 # Actual sample size used for bootstrap estimate will vary between the min and max,  
3055 # depending upon row count (more rows, smaller sample size) and accuracy settings (higher accuracy, larger sample size)  
3056  
3057 max_bootstrap_sample_size_factor = 10.0  
3058  
3059 # Seed to use for final model bootstrap sampling, -1 means use experiment-derived seed.  
3060 # E.g. one can retrain final model with different seed to get different final model error bars for scores.  
3061  
3062 bootstrap_final_seed = -1  
3063  
3064 varimp_threshold_at_interpretability_10 = 0.05  
3065  
3066 features_allowed_by_interpretability = "(1: 10000000, 2: 10000, 3: 1000, 4: 500, 5: 300, 6: 200, 7: 150, 8: 100, 9: 80, 10: 50, 11: 50, 12: 50, 13: 50)"  
3067  
3068 nfeatures_max_threshold = 200  
3069  
3070 rdelta_percent_score_penalty_per_feature_by_interpretability = "(1: 0.0, 2: 0.1, 3: 1.0, 4: 2.0, 5: 5.0, 6: 10.0, 7: 20.0, 8: 30.0, 9: 50.0, 10: 100.0, 11:  
↪ 100.0, 12: 100.0, 13: 100.0)"  
3071  
3072 drop_low_meta_weights = true  
3073  
3074 meta_weight_allowed_by_interpretability = "(1: 1E-7, 2: 1E-5, 3: 1E-4, 4: 1E-3, 5: 1E-2, 6: 0.03, 7: 0.05, 8: 0.08, 9: 0.10, 10: 0.15, 11: 0.15, 12: 0.15,  
↪ 13: 0.15)"  
3075  
3076 meta_weight_allowed_for_reference = 1.0  
3077  
3078 feature_cost_mean_interp_for_penalty = 5  
3079  
3080 features_cost_per_interp = 0.25  
3081  
3082 varimp_threshold_shift_report = 0.3  
3083  
3084 apply_featuregene_limits_after_tuning = true  
3085  
3086 remove_scored_0gain_genes_in_postprocessing_above_interpretability = 13  
3087  
3088 remove_scored_0gain_genes_in_postprocessing_above_interpretability_final_population = 2  
3089  
3090 remove_scored_by_threshold_genes_in_postprocessing_above_interpretability_final_population = 7  
3091  
3092 # Graphviz is an optional requirement for native installations (RPM/DEB/Tar-SH, outside of Docker) to convert .dot files into .png files for pipeline  
↪ visualizations as part of experiment artifacts
```

(continues on next page)

(continued from previous page)

```

3092 #require_graphviz = true
3093
3094 # Unnormalized probability to add genes or instances of transformers with specific attributes.
3095 # If no genes can be added, other mutations
3096 # (mutating models hyper parameters, pruning genes, pruning features, etc.) are attempted.
3097 #
3098 #prob_add_genes = 0.5
3099
3100 # Unnormalized probability, conditioned on prob_add_genes,
3101 # to add genes or instances of transformers with specific attributes
3102 # that have shown to be beneficial to other individuals within the population.
3103 #
3104 #prob_addbest_genes = 0.5
3105
3106 # Unnormalized probability to prune genes or instances of transformers with specific attributes.
3107 # If a variety of transformers with many attributes exists, default value is reasonable.
3108 # However, if one has fixed set of transformers that should not change or no new transformer attributes
3109 # can be added, then setting this to 0.0 is reasonable to avoid undesired loss of transformations.
3110 #
3111 #prob_prune_genes = 0.5
3112
3113 # Unnormalized probability change model hyper parameters.
3114 #
3115 #prob_perturb_xgb = 0.25
3116
3117 # Unnormalized probability to prune features that have low variable importance,
3118 # as opposed to pruning entire instances of genes/transfomers.
3119 #
3120 #prob_prune_by_features = 0.25
3121
3122 # Whether to use exploit-explore logic like DAI 1.8.x. False will explore more.
3123 #use_187_prob_logic = true
3124
3125 #max_absolute_feature_expansion = 1000
3126
3127 #booster_for_fs_permute = "auto"
3128
3129 #model_class_name_for_fs_permute = "auto"
3130
3131 #switch_from_tree_to_lgbm_if_can = true
3132
3133 # Number of classes above which to always use TensorFlow (if TensorFlow is enabled),
3134 # instead of others models set on 'auto' (models set to 'on' are still used).
3135 #tensorflow_num_classes_switch = 10
3136
3137 # Compute empirical prediction intervals (based on holdout predictions).
3138 #prediction_intervals = true
3139
3140 # Confidence level for prediction intervals.
3141 #prediction_intervals_alpha = 0.9
3142
3143 # Class count above which do not use TextLin Transformer.
3144 #textlin_num_classes_switch = 5
3145
3146 #text_gene_dim_reduction_choices = "[50]"
3147
3148 #text_gene_max_ngram = "[1, 2, 3]"
3149
3150 #gbm_early_stopping_rounds_min = 1
3151
3152 #gbm_early_stopping_rounds_max = 10000000000
3153
3154 # Max. number of top variable importances to save per iteration (GUI can only display a max. of 14)
3155 #max_varimp_to_save = 100
3156
3157 # Max. number of top variable importances to show in logs during feature evolution
3158 #max_num_varimp_to_log = 10
3159
3160 # Max. number of top variable importance shifts to show in logs and GUI after final model built
3161 #max_num_varimp_shift_to_log = 10
3162
3163 # Stall submission of subprocesses if system CPU usage is higher than this threshold in percent (set to 100 to disable). A reasonable number is 90.0 if activated
3164 #stall_subprocess_submission_cpu_threshold_pct = 100
3165
3166 # Restrict/Stall submission of subprocesses if DAI fork count (across all experiments) per unit ulimit nproc soft limit is higher than this threshold in percent (set to -1 to disable, 0 for minimal forking. A reasonable number is 90.0 if activated
3167 #stall_subprocess_submission_dai_fork_threshold_pct = -1.0
3168
3169 # Restrict/Stall submission of subprocesses if experiment fork count (across all experiments) per unit ulimit nproc soft limit is higher than this threshold in percent (set to -1 to disable, 0 for minimal forking). A reasonable number is 90.0 if activated. For small data leads to overhead of about 0.1s per task submitted due to checks, so for scoring can slow things down for tests.
3170 #stall_subprocess_submission_experiment_fork_threshold_pct = -1.0
3171
3172 # Whether to restrict pool workers even if not used, by reducing number of pool workers available. Good if really huge number of experiments, but otherwise, best to have all pool workers ready and only stall submission of tasks so can be dynamic to multi-experiment environment
3173 #restrict_initpool_by_memory = true
3174
3175 #num_rows_acceptance_test_custom_transformer = 200
3176
3177 #num_rows_acceptance_test_custom_model = 100
3178
3179 # Dictionary to control recipes for each experiment and particular custom recipes.
3180 # E.g. if inserting into the GUI as any toml string, can use:
3181 # """recipe_dict={"key1": 2, "key2": "value2}"""
3182 # E.g. if putting into config.toml as a dict, can use:
3183 # recipe_dict={"key1": 2, "key2": "value2"}"
3184 #
3185 #recipe_dict = "{}"
3186
3187 #enable_custom_transformers = true
3188
3189 #enable_custom_pretransformers = true
3190

```

(continues on next page)

(continued from previous page)

```

3191 #enable_custom_models = true
3192
3193 #enable_custom_scorers = true
3194
3195 #enable_custom_datas = true
3196
3197 # location of custom recipes packages installed (relative to data_directory)
3198 # We will try to install packages dynamically, but can also do (before or after server started):
3199 # (inside docker running docker instance if running docker, or as user server is running as (e.g. dai user) if deb/tar native installation:
3200 # PYTHONPATH=<full tmp dir>/<contrib_env_relative_directory>/lib/python3.6/site-packages/ <path to dai>dai-env.sh python -m pip install --prefix=<full tmp_dir>/<contrib_env_relative_directory> <packagename> --upgrade --upgrade-strategy only-if-needed --log-file pip_log_file.log
3201 # where <path to dai> is /opt/h2oai/dai/ for native rpm/deb installation
3202 # Note can also install wheel files if <packagename> is name of wheel file or archive.
3203 #
3204 #contrib_env_relative_directory = "contrib/env"
3205
3206 # List of package versions to ignore. Useful when small version change but likely to function still.
3207 #
3208 #ignore_package_version = []
3209
3210 # pip install retry for call to pip. Sometimes need to try twice
3211 #pip_install_overall_retries = 2
3212
3213 # pip install verbosity level (number of -v's given to pip, up to 3
3214 #pip_install_verbosity = 2
3215
3216 # pip install timeout in seconds. Sometimes internet issues would mean want to fail faster
3217 #pip_install_timeout = 15
3218
3219 # pip install retry count
3220 #pip_install_retries = 5
3221
3222 # Whether to use DAI constraint file to help pip handle versions. pip can make mistakes and try to install updated packages for no reason.
3223 #pip_install_use_constraint = true
3224
3225 # pip install options: string of list of other options, e.g. "[--proxy, 'http://user:password@proxyserver:port']"
3226 #pip_install_options = ""
3227
3228 # Whether to enable basic acceptance testing. Tests if can pickle the state, etc.
3229 #enable_basic_acceptance_tests = true
3230
3231 # Whether acceptance tests should run for custom genes / models / scorers / etc.
3232 #enable_acceptance_tests = true
3233
3234 # Minutes to wait until a recipe's acceptance testing is aborted. A recipe is rejected if acceptance
3235 # testing is enabled and times out.
3236 # One may also set timeout for a specific recipe by setting the class's staticmethod function called
3237 # acceptance_test_timeout to return number of minutes to wait until timeout doing acceptance testing.
3238 # This timeout does not include the time to install required packages.
3239 #
3240 #acceptance_test_timeout = 20.0
3241
3242 # Skipping just avoids the failed transformer.
3243 # Sometimes python multiprocessing swallows exceptions,
3244 # so skipping and logging exceptions is also more reliable way to handle them.
3245 # Recipe can raise h2oaicore.systemutils.IgnoreError to ignore error and avoid logging error.
3246 #
3247 #skip_transformer_failures = true
3248
3249 # Skipping just avoids the failed model. Failures are logged depending upon detailed_skip_failure_messages_level.
3250 # Recipe can raise h2oaicore.systemutils.IgnoreError to ignore error and avoid logging error.
3251 #
3252 #skip_model_failures = true
3253
3254 # How much verbosity to log failure messages for failed and then skipped transformers or models.
3255 # Full failures always go to disk as *.stack files,
3256 # which upon completion of experiment goes into details folder within experiment log zip file.
3257 #
3258 #detailed_skip_failure_messages_level = 1
3259
3260 # Instructions for 'Add to config.toml via toml string' in GUI expert page
3261 # Self-referential toml parameter, for setting any other toml parameters as string of tomls separated by
3262 # (spaces around
3263 # are ok).
3264 # Useful when toml parameter is not in expert mode but want per-experiment control.
3265 # Setting this will override all other choices.
3266 # In expert page, each time expert options saved, the new state is set without memory of any prior settings.
3267 # The entered item is a fully compliant toml string that would be processed directly by toml.load().
3268 # One should include 2 double quotes around the entire setting, or double quotes need to be escaped.
3269 # One enters into the expert page text as follows:
3270 # e.g. enable_glm="off"
3271 # enable_xgboost_gbm="off"
3272 # enable_lightgbm="on"
3273 # e.g. ""enable_glm="off"
3274 # enable_xgboost_gbm="off"
3275 # enable_lightgbm="off"
3276 # enable_tensorflow="on"""
3277 # e.g. fixed_nu_individuals=4
3278 # e.g. params_lightgbm="{'objective':'poisson'}"
3279 # e.g. ""params_lightgbm="{'objective':'poisson'}"""
3280 # e.g. max_cores=10
3281 # data_precision="float32"
3282 # max_rows_feature_evolution=50000000000
3283 # ensemble_accuracy_switch=11
3284 # feature_engineering_effort=1
3285 # target_transformer="identity"
3286 # tournament_feature_style_accuracy_switch=5
3287 # params_tensorflow="['layers': [100, 100, 100, 100, 100, 100]]"
3288 # e.g. ""max_cores=10
3289 # data_precision="float32"
3290 # max_rows_feature_evolution=50000000000
3291 # ensemble_accuracy_switch=11
3292 # feature_engineering_effort=1
3293 # target_transformer="identity"

```

(continues on next page)

(continued from previous page)

```

3294 # tournament_feature_style_accuracy_switch=5
3295 # params_tensorflow="{'layers': [100, 100, 100, 100, 100, 100, 100]}"""
3296 # If you see: "toml.TomlDecodeError" then ensure toml is set correctly.
3297 # When set in the expert page of an experiment, these changes only affect experiments and not the server
3298 # Usually should keep this as empty string in this toml file.
3299 #config_overrides = ""
3300
3301 # Whether user can download dataset as csv file
3302 #enable_dataset_downloading = true
3303
3304 # Extra HTTP headers.
3305 #extra_http_headers = "{}"
3306
3307 # After how many days the audit log records are removed.
3308 # Set equal to 0 to disable removal of old records.
3309 #
3310 #audit_log_retention_period = 5
3311
3312 # Replace all the downloads on the experiment page to exports and allow users to push to the artifact store configured with artifacts_store
3313 #enable_artifacts_upload = false
3314
3315 # Artifacts store.
3316 # file_system: stores artifacts on a file system directory denoted by artifacts_file_system_directory.
3317 # s3: stores artifacts to S3 bucket.
3318 # bitbucket: stores data into Bitbucket repository.
3319 #
3320 #artifacts_store = "file_system"
3321
3322 # Decide whether to skip cert verification for Bitbucket when using a repo with HTTPS
3323 #bitbucket_skip_cert_verification = false
3324
3325 # Local temporary directory to clone artifacts to, relative to data_directory
3326 #bitbucket_tmp_relative_dir = "local_git_tmp"
3327
3328 # File system location where artifacts will be copied in case artifacts_store is set to file_system
3329 #artifacts_file_system_directory = "tmp"
3330
3331 # AWS S3 bucket to be used for storing artifacts.
3332 #artifacts_s3_bucket = ""
3333
3334 # Git auth user
3335 #artifacts_git_user = "git"
3336
3337 # Git auth password
3338 #artifacts_git_password = ""
3339
340 # Git repo where artifacts will be pushed upon and upload
341 #artifacts_git_repo = ""
342
343 # Git branch on the remote repo where artifacts are pushed
344 #artifacts_git_branch = "dev"
345
346 # File location for the ssh private key used for git authentication
347 #artifacts_git_ssh_private_key_file_location = ""
348
349 # Enable column imputation
350 #enable_imputation = false
351
352 #num_models_for_resume_graph = 1000
353

```


ENVIRONMENT VARIABLES AND CONFIGURATION OPTIONS

Driverless AI provides a number of environment variables that can be passed when starting Driverless AI or specified in a config.toml file. The complete list of variables is in the [Using the config.toml File](#) section. The steps for specifying variables vary depending on whether you installed a Driverless AI RPM, DEB, or TAR SH or whether you are running a Docker image.

13.1 Setting Environment Variables and Configurations Options

Docker Image Installs

Native Installs

Each property must be prepended with DRIVERLESS_AI_. The example below starts Driverless AI with environment variables that enable S3 and HDFS access (without authentication).

```
nvidia-docker run \
--pid=host \
--init \
--rm \
-u `id -u : id -g` \
-e DRIVERLESS_AI_ENABLED_FILE_SYSTEMS="file,s3,hdfs" \
-e DRIVERLESS_AI_AUTHENTICATION_METHOD="local" \
-e DRIVERLESS_AI_LOCAL_HTPASSWD_FILE="" \
-v /etc/passwd:/etc/passwd:ro \
-v /etc/group:/etc/group:ro \
-v `pwd`/data:/data \
-v `pwd`/log:/log \
-v `pwd`/license:/license \
-v `pwd`/tmp:/tmp \
harbor.h2o.ai/h2oai/dai-centos7-x86_64:1.9.0-cuda10.0.23
```

The config.toml file is available in the etc/dai folder after the RPM, DEB, or TAR SH is installed. Edit the desired variables in this file, and then restart Driverless AI.

The example below shows the configuration options in the config.toml file to set when enabling the S3 and HDFS access (without authentication)

1. Export the Driverless AI config.toml file or add it to ~/.bashrc. For example:

```
# DEB and RPM
export DRIVERLESS_AI_CONFIG_FILE="/etc/dai/config.toml"

# TAR SH
export DRIVERLESS_AI_CONFIG_FILE="/path/to/your/unpacked/dai/directory/config.toml"
```

2. Specify the desired configuration options to enable S3 and HDFS access (without authentication).

```
# File System Support
# upload : standard upload feature
# file : local file system/server file system
# hdfs : Hadoop file system, remember to configure the HDFS config folder path and keytab below
# dtap : Blue Data Tap file system, remember to configure the DTap section below
# s3 : Amazon S3, optionally configure secret and access key below
# gcs : Google Cloud Storage, remember to configure gcs_path_to_service_account_json below
# gbq : Google BigQuery, remember to configure gcs_path_to_service_account_json below
# minio : Minio Cloud Storage, remember to configure secret and access key below
# snow : Snowflake Data Warehouse, remember to configure Snowflake credentials below (account name, username, password)
```

(continues on next page)

(continued from previous page)

```
# kdb : KDB+ Time Series Database, remember to configure KDB credentials below (hostname and port, optionally: username, password, ↵
# classpath, and jvm_args)
# azrbs : Azure Blob Storage, remember to configure Azure credentials below (account name, account key)
# jdbc: JDBC Connector, remember to configure JDBC below. (jdbc_app_configs)
enabled_file_systems = "file,s3,hdfs"

# authentication_method
# unvalidated : Accepts user id and password, does not validate password
# none : Does not ask for user id or password, authenticated as admin
# pam : Accepts user id and password, Validates user with operating system
# ldap : Accepts user id and password, Validates against an ldap server, look
# local: Accepts a user id and password, Validated against a htpasswd file provided in local_htpasswd_file
# for additional settings under LDAP settings
authentication_method = "local"

# Local password file
# Generating a htpasswd file: see syntax below
# htpasswd -B "<location_to_place_htpasswd_file>" "<username>"
# note: -B forces use of bcrypt, a secure encryption method
local_htpasswd_file = "<htpasswd_file_location>"
```

3. Start Driverless AI. Note that the command used to start Driverless AI varies depending on your install type.

```
# Linux RPM or DEB with systemd
sudo systemctl start dai

# Linux RPM or DEB without systemd
sudo -H -u dai /opt/h2oai/dai/run-dai.sh

# Linux TAR SH
./run-dai.sh
```

CHAPTER FOURTEEN

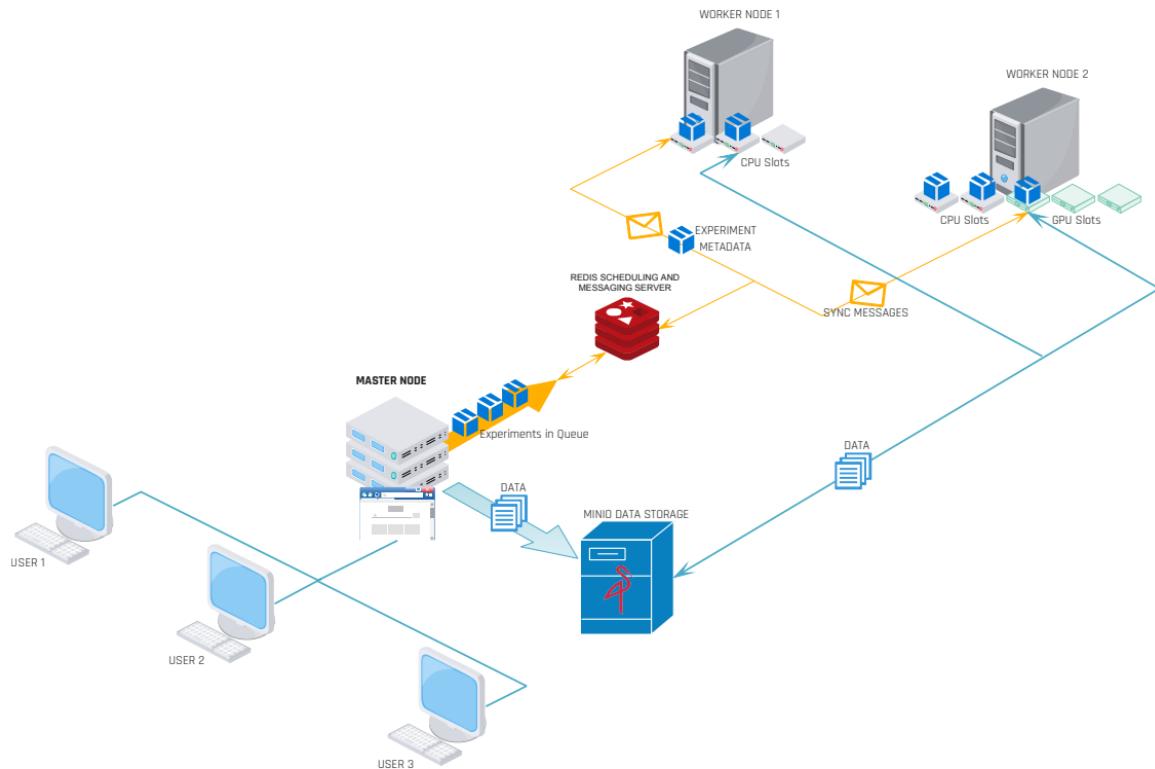
MULTINODE TRAINING (ALPHA)

Driverless AI can be configured to run in a multinode worker mode. This section describes the multinode training process and how to configure it.

14.1 Understanding Multinode Training

Multinode training in Driverless AI can be used to run multiple experiments at the same time. It is effective in situations where you need to run and complete many experiments simultaneously in a short amount of time without having to wait for each individual experiment to finish.

Multinode training uses a load distribution technique in which a set of machines (worker nodes) are used to help a main server node process experiments. These machines can be CPU only or CPU + GPU, with experiments being distributed accordingly.



Jobs (experiments) within the multinode setup are organized into a queue. Jobs remain in this queue when no processor is available. When a worker's processor becomes available, it asks the job queue service to assign it a new job. By default, each worker node processes two jobs at a time (configured with the `worker_remote_processors` option in the `config.toml` file). Each worker can process multiple jobs at the same time, but two workers cannot process the same experiment at the same time. Messaging and data exchange services are also implemented to allow the workers to effectively communicate with the main server node.

Notes:

- Multinode training in Driverless AI is currently in a preview stage. If you are interested in using multinode configurations, contact support@h2o.ai.
- Multinode training requires the transfer of data to several different workers. For example, if an experiment is scheduled to be on a remote worker node, the datasets it is using need to be copied to the worker machine by using the Minio service. The experiment can take longer to initialize depending on the size of the transferred objects.
- The number of jobs that each worker node processes is controlled by the `worker_remote_processors` option in the `config.toml` file.
- Tasks are not distributed to best fit workers. Workers consume tasks from the queue on a first in, first out (FIFO) basis.
- A single experiment runs entirely on one machine. For this reason, using a large number of commodity-grade hardware is not useful in the context of multinode.

14.2 Requirements

- Redis

14.3 Description of Configuration Attributes

- `worker_mode`: Specifies how the long-running tasks are scheduled. Available options include:
 - `multiprocessing`: Forks the current process immediately.
 - `singlenode`: Shares the task through Redis and needs a worker running.
 - `multinode`: Same as `singlenode`. Also shares the data through MinIO and allows the worker to run on the different machine.
- `redis_ip`: Redis IP address. Defaults to 127.0.0.1
- `redis_port`: Redis port. Defaults to 6379.
- `redis_db`: Redis database. Each DAI instance running on the Redis server should have unique integer. Defaults to 0.
- `main_server_redis_password`: Main Server Redis password. Defaults to empty string.
- `local_minio_port`: The port that MinIO will listen on. This only takes effect if the current system is a multinode main server.
- `main_server_minio_address`: The address of the main server's MinIO server. Defaults to 127.0.0.1:9000.
- `main_server_minio_access_key_id`: Access key of main server's MinIO server.
- `main_server_minio_secret_access_key`: The secret access key of main server MinIO server.

- `main_server_minio_bucket`: The name of MinIO bucket used for file synchronization.
- `worker_local_processors`: The maximum number of local tasks processed at once. Defaults to 32.
- `worker_remote_processors`: The maximum number of remote tasks processed at once. Defaults to 2.
- `redis_result_queue_polling_interval`: The frequency in milliseconds that the server should extract results from the Redis queue. Defaults to 100.
- `main_server_minio_bucket_ping_timeout`: The number of seconds the worker should wait for the main server MinIO bucket before it fails. Defaults to 30.
- `worker_healthy_response_period`: The number of seconds to wait for the worker to respond before being marked as unhealthy. Defaults to 300.

14.4 Multinode Setup Example

The following example configures a two-node Multinode Driverless AI cluster on AWS EC2 instances using bashtar distribution. This example can be expanded to multiple worker nodes. This example assumes that you have spun up two EC2 instances (Ubuntu 16.04) within the same VPC on AWS.

14.4.1 VPC Settings

In the VPC settings, enable inbound rules to listen to TCP connections on port 6379 for Redis and 9000 for MinIO.

14.4.2 Install the Driverless AI Natively

Install Driverless AI on the server node. Refer to one of the following topics for information on how to perform a native install on Linux systems.

- [Linux DEBs](#)
- [Linux RPMs](#)
- [Linux TAR SH](#)

14.4.3 Edit the Driverless AI config.toml

After Driverless AI is installed, edit the following config options in the config.toml file.

```
#set worker mode to multinode
worker_mode = "multinode"

# Redis settings -- set the ip address of redis server to aws instance ip
redis_ip = "<host_ip>"

# Redis settings
redis_port = 6379

# Redis settings
main_server_redis_password = "<main_server_redis_pwd>"

# Location of main server's minio server.
# Note that you can also use 'local_minio_port' to specify a different port.
main_server_minio_address = "<host_ip>:9000"
```

14.4.4 Start the Driverless AI Server Node

```
cd dai-1.9.0-linux-x86_64  
./run-dai.sh
```

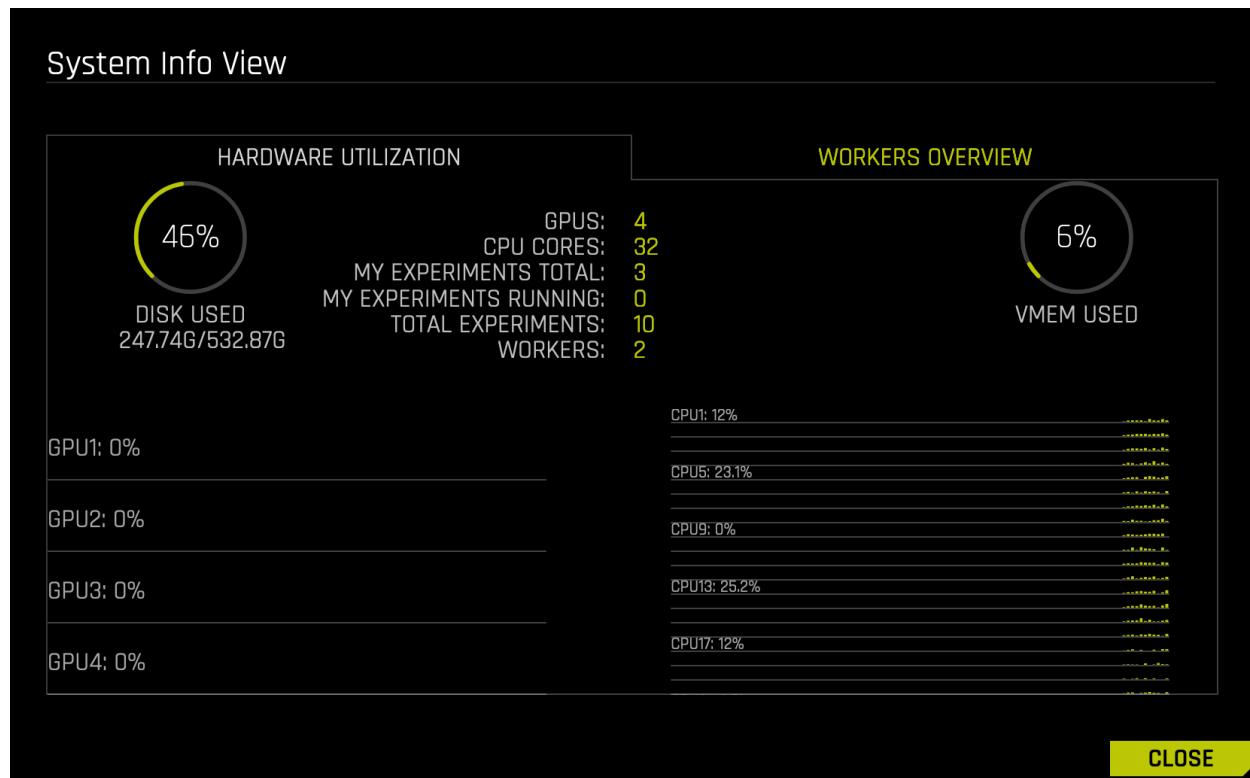
Install the Linux deb/rpm/tar package on the EC2 instance to create a Driverless AI worker node. After the installation is complete, edit the following in the config.toml.

```
# Redis settings, point to the dai main server's redis server ip address  
redis_ip = "<dai_main_server_host_ip>"  
  
# Redis settings  
redis_port = 6379  
  
# Redis settings, point to the dai main server's redis server password  
main_server_redis_password = "<dai_main_server_host_redis_pwd>"  
  
# Location of the dai main server's minio server.  
main_server_minio_address = "<dai_main_server_host>:9000"
```

14.4.5 Start the Driverless AI Worker Node

```
cd dai-1.9.0-linux-x86_64  
./run-dai.sh --worker  
  
# Note that when using rpm/deb you can run the following:  
sudo systemctl start dai-worker
```

Once the worker node starts, use the Driverless AI server IP to log into Driverless AI. Click on **Resources > System Info** to confirm that the number of workers is “2” if only one worker is used. (By default, each worker node processes two jobs at a time. This is configured with the `worker_remote_processors` option in the config.toml file.)



ENABLING DATA CONNECTORS

Driverless AI provides a number of data connectors for accessing external data sources. The following data connection types are enabled by default:

- `upload`: standard upload feature
- `file`: local file system/server file system
- `hdfs`: Hadoop file system, remember to configure the HDFS config folder path and keytab
- `s3`: Amazon S3, optionally configure secret and access key
- `recipe_file`: Custom recipe file upload
- `recipe_url`: Custom recipe upload via url

Additionally, the following connections types can be enabled by modifying the `enabled_file_systems` configuration option (Native installs) or environment variable (Docker image installs):

- `dtap`: Blue Data Tap file system, remember to configure the DTap section
- `gcs`: Google Cloud Storage, remember to configure `gcs_path_to_service_account_json`
- `gbq`: Google Big Query, remember to configure `gcs_path_to_service_account_json`
- `hive`: Hive Connector, remember to configure Hive
- `minio`: Minio Cloud Storage, remember to configure secret and access key
- `snow`: Snowflake Data Warehouse, remember to configure Snowflake credentials
- `kdb`: KDB+ Time Series Database, remember to configure KDB credentials
- `azrbs`: Azure Blob Storage, remember to configure Azure credentials
- `jdbc`: JDBC Connector, remember to configure JDBC

These data sources are exposed in the form of the file systems, and each file system is prefixed by a unique prefix. For example:

- To reference data on S3, use `s3://`.
- To reference data on HDFS, use the prefix `hdfs://`.
- To reference data on Azure Blob Store, use `https://<storage_name>.blob.core.windows.net.`
- To reference data on BlueData Datatap, use `dtap://`.
- To reference data on Google BigQuery, make sure you know the Google BigQuery dataset and the table that you want to query. Use a standard SQL query to ingest data.
- To reference data on Google Cloud Storage, use `gs://`
- To reference data on kdb+, use the hostname and the port `http://<kdb_server>:<port>`

- To reference data on Minio, use `http://<endpoint_url>`.
- To reference data on Snowflake, use a standard SQL query to ingest data.
- To access a SQL database via JDBC, use a SQL query with the syntax associated with your database.

Refer to the following sections for more information:

15.1 S3 Setup

Driverless AI allows you to explore S3 data sources from within the Driverless AI application. This section provides instructions for configuring Driverless AI to work with S3.

Note: Depending on your Docker install version, use either the `docker run --runtime=nvidia (>= Docker 19.03)` or `nvidia-docker (< Docker 19.03)` command when starting the Driverless AI Docker image. Use `docker version` to check which version of Docker you are using.

15.1.1 Description of Configuration Attributes

- `aws_access_key_id`: The S3 access key ID
- `aws_secret_access_key`: The S3 access key
- `aws_role_arn`: The Amazon Resource Name
- `aws_default_region`: The region to use when the `aws_s3_endpoint_url` option is not set. This is ignored when `aws_s3_endpoint_url` is set.
- `aws_s3_endpoint_url`: The endpoint URL that will be used to access S3.
- `aws_use_ec2_role_credentials`: If set to true, the S3 Connector will try to obtain credentials associated with the role attached to the EC2 instance.
- `s3_init_path`: The starting S3 path that will be displayed in UI S3 browser.
- `enabled_file_systems`: The file systems you want to enable. This must be configured in order for data connectors to function properly.

15.1.2 Example 1: Enable S3 with No Authentication

Docker Image Installs

Docker Image with the config.toml

Native Installs

This example enables the S3 data connector and disables authentication. It does not pass any S3 access key or secret; however it configures Docker DNS by passing the name and IP of the S3 name node. This allows users to reference data stored in S3 directly using the name node address, for example: `s3://name.node/datasets/iris.csv`.

```
nvidia-docker run \
--shm-size=256m \
--add-host name.node:172.16.2.186 \
-e DRIVERLESS_AI_ENABLED_FILE_SYSTEMS="file,s3" \
-p 12345:12345 \
--init -it --rm \
-v /tmp/dtmp:/tmp \
-v /tmp/dlog:/log \
-v /tmp/dlicense:/license \
-v /tmp/ddata:/data \
-u $(id -u):$(id -g) \
harbor.h2o.ai/h2oai/dai-centos7-x86_64:1.9.0-cuda10.0.23
```

This example shows how to configure S3 options in the config.toml file, and then specify that file when starting Driverless AI in Docker. Note that this example enables S3 with no authentication.

1. Configure the Driverless AI config.toml file. Set the following configuration options.

- enabled_file_systems = "file, upload, s3"

2. Mount the config.toml file into the Docker container.

```
nvidia-docker run \
--pid=host \
--init \
--rm \
--shm-size=256m \
--add-host name.node:172.16.2.186 \
-e DRIVERLESS_AI_CONFIG_FILE=/path/in/docker/config.toml \
-p 12345:12345 \
-v /local/path/to/config.toml:/path/in/docker/config.toml \
-v /etc/passwd:/etc/passwd:ro \
-v /etc/group:/etc/group:ro \
-v /tmp/dtmp/:/tmp \
-v /tmp/dlog/:/log \
-v /tmp/license/:/license \
-v /tmp/ddata/:/data \
-u $(id -u):$(id -g) \
harbor.h2o.ai/h2oai/dai-centos7-x86_64:1.9.0-cudal0.0.23
```

This example enables the S3 data connector and disables authentication. It does not pass any S3 access key or secret.

1. Export the Driverless AI config.toml file or add it to `~/.bashrc`. For example:

```
# DEB and RPM
export DRIVERLESS_AI_CONFIG_FILE="/etc/dai/config.toml"

# TAR SH
export DRIVERLESS_AI_CONFIG_FILE="/path/to/your/unpacked/dai/directory/config.toml"
```

2. Specify the following configuration options in the config.toml file.

```
# File System Support
# upload : standard upload feature
# file : local file system/server file system
# hdfs : Hadoop file system, remember to configure the HDFS config folder path and keytab below
# dtap : Blue Data Tap file system, remember to configure the DTap section below
# s3 : Amazon S3, optionally configure secret and access key below
# gcs : Google Cloud Storage, remember to configure gcs_path_to_service_account_json below
# gbq : Google Big Query, remember to configure gcs_path_to_service_account_json below
# minio : Minio Cloud Storage, remember to configure secret and access key below
# snow : Snowflake Data Warehouse, remember to configure Snowflake credentials below (account name, username, password)
# kdb : KDB+ Time Series Database, remember to configure KDB credentials below (hostname and port, optionally: username, password, ↵
#       classpath, and jvm_args)
# azrbs : Azure Blob Storage, remember to configure Azure credentials below (account name, account key)
# jdbc: JDBC Connector, remember to configure JDBC below. (jdbc_app_configs)
# hive: Hive Connector, remember to configure Hive below. (hive_app_configs)
# recipe_url: load custom recipe from URL
# recipe_file: load custom recipe from local file system
enabled_file_systems = "file, s3"
```

3. Save the changes when you are done, then stop/restart Driverless AI.

15.1.3 Example 2: Enable S3 with Authentication

Docker Image Installs

Docker Image with the config.toml

Native Installs

This example enables the S3 data connector with authentication by passing an S3 access key ID and an access key. It also configures Docker DNS by passing the name and IP of the S3 name node. This allows users to reference data stored in S3 directly using the name node address, for example: `s3://name.node/datasets/iris.csv`.

```
nvidia-docker run \
--shm-size=256m \
--add-host name.node:172.16.2.186 \
-e DRIVERLESS_AI_ENABLED_FILE_SYSTEMS="file,s3" \
-e DRIVERLESS_AI_AWS_ACCESS_KEY_ID="" \
-e DRIVERLESS_AI_AWS_SECRET_ACCESS_KEY="" \
-p 12345:12345 \
--init -it --rm \
-v /tmp/dtmp/:/tmp \
```

(continues on next page)

(continued from previous page)

```
-v /tmp/dlog/:/log \
-v /tmp/dlicense/:/license \
-v /tmp/ddata/:/data \
-u ${id}-u:${id}-g) \
harbor.h2o.ai/h2oai/dai-centos7-x86_64:1.9.0-cudal0.0.23
```

This example shows how to configure S3 options with authentication in the config.toml file, and then specify that file when starting Driverless AI in Docker.

1. Configure the Driverless AI config.toml file. Set the following configuration options.

- enabled_file_systems = "file, upload, s3"
- aws_access_key_id = "<access_key_id>"
- aws_secret_access_key = "<access_key>"

2. Mount the config.toml file into the Docker container.

```
nvidia-docker run \
--pid=host \
--init \
--rm \
--shm-size=256m \
--add-host name.node:172.16.2.186 \
-e DRIVERLESS_AI_CONFIG_FILE=/path/in/docker/config.toml \
-p 12345:12345 \
-v /local/path/to/config.toml:/path/in/docker/config.toml \
-v /etc/passwd:/etc/passwd:ro \
-v /etc/group:/etc/group:ro \
-v /tmp/dtmp/:/tmp \
-v /tmp/dlog/:/log \
-v /tmp/dlicense/:/license \
-v /tmp/ddata/:/data \
-u ${id}-u:${id}-g) \
harbor.h2o.ai/h2oai/dai-centos7-x86_64:1.9.0-cudal0.0.23
```

This example enables the S3 data connector with authentication by passing an S3 access key ID and an access key.

1. Export the Driverless AI config.toml file or add it to `~/.bashrc`. For example:

```
# DEB and RPM
export DRIVERLESS_AI_CONFIG_FILE="/etc/dai/config.toml"

# TAR SH
export DRIVERLESS_AI_CONFIG_FILE="/path/to/your/unpacked/dai/directory/config.toml"
```

2. Specify the following configuration options in the config.toml file.

```
# File System Support
# upload : standard upload feature
# file : local file system/server file system
# hdfs : Hadoop file system, remember to configure the HDFS config folder path and keytab below
# dtap : Blue Data Tap file system, remember to configure the DTap section below
# s3 : Amazon S3, optionally configure secret and access key below
# gcs : Google Cloud Storage, remember to configure gcs_path_to_service_account_json below
# gbig : Google Big Query, remember to configure gcs_path_to_service_account_json below
# minio : Minio Cloud Storage, remember to configure secret and access key below
# snow : Snowflake Data Warehouse, remember to configure Snowflake credentials below (account name, username, password)
# kdb : KDB+ Time Series Database, remember to configure KDB credentials below (hostname and port, optionally: username, password, classpath, and jvm_args)
# azrbs : Azure Blob Storage, remember to configure Azure credentials below (account name, account key)
# jdbc: JDBC Connector, remember to configure JDBC below. (jdbc_app_configs)
# hive: Hive Connector, remember to configure Hive below. (hive_app_configs)
# recipe_url: load custom recipe from URL
# recipe_file: load custom recipe from local file system
enabled_file_systems = "file, s3"

# S3 Connector credentials
aws_access_key_id = "<access_key_id>"
aws_secret_access_key = "<access_key>"
```

3. Save the changes when you are done, then stop/restart Driverless AI.

15.2 HDFS Setup

Driverless AI allows you to explore HDFS data sources from within the Driverless AI application. This section provides instructions for configuring Driverless AI to work with HDFS.

Note: Depending on your Docker install version, use either the `docker run --runtime=nvidia (>= Docker 19.03)` or `nvidia-docker (< Docker 19.03)` command when starting the Driverless AI Docker image. Use `docker version` to check which version of Docker you are using.

15.2.1 Description of Configuration Attributes

- `hdfs_config_path`: (Required) The location the HDFS config folder path. This folder can contain multiple config files.
- `hdfs_auth_type`: (Required) Specifies the HDFS authentication. Available values are:
 - `principal`: Authenticate with HDFS with a principal user.
 - `keytab`: Authenticate with a keytab (recommended). If running DAI as a service, then the Kerberos keytab needs to be owned by the DAI user.
 - `keytabimpersonation`: Login with impersonation using a keytab.
 - `noauth`: No authentication needed.
- `key_tab_path`: The path of the principal key tab file. This is required when `hdfs_auth_type='principal'`.
- `hdfs_app_principal_user`: The Kerberos application principal user. This is required when `hdfs_auth_type='keytab'`.
- `hdfs_app_jvm_args`: JVM args for HDFS distributions. Separate each argument with spaces.
 - `-Djava.security.krb5.conf`
 - `-Dsun.security.krb5.debug`
 - `-Dlog4j.configuration`
- `hdfs_app_classpath`: The HDFS classpath.
- `hdfs_app_supported_schemes`: The list of DFS schemas that is used to check whether a valid input to the connector has been established. For example:

```
hdfs_app_supported_schemes = ['hdfs://', 'maprfs://', 'custom://']
```

The following are the default values for this option. Additional schemas can be supported by adding values that are not selected by default to the list.

- `hdfs://`
- `maprfs://`
- `swift://`
- `enabled_file_systems`: The file systems you want to enable. This must be configured in order for data connectors to function properly.

15.2.2 Example 1: Enable HDFS with No Authentication

Docker Image Installs

Docker Image with the config.toml

Native Installs

This example enables the HDFS data connector and disables HDFS authentication. It does not pass any HDFS configuration file; however it configures Docker DNS by passing the name and IP of the HDFS name node. This allows users to reference data stored in HDFS directly using name node address, for example: `hdfs://name.node/datasets/iris.csv`.

```
nvidia-docker run \
--pid=host \
--init \
--rm \
--shm-size=256m \
--add-host name.node:172.16.2.186 \
-e DRIVERLESS_AI_ENABLED_FILE_SYSTEMS="file,hdfs" \
-e DRIVERLESS_AI_HDFS_AUTH_TYPE='noauth' \
-e DRIVERLESS_AI_PROCSY_PORT=8080 \
-p 12345:12345 \
-v /etc/passwd:/etc/passwd:ro \
-v /etc/group:/etc/group:ro \
-v /tmp/dtmp/:/tmp \
-v /tmp/dlog/:/log \
-v /tmp/dlicense/:/license \
-v /tmp/ddata/:/data \
-u ${id -u}:${id -g} \
harbor.h2o.ai/h2oai/dai-centos7-x86_64:1.9.0-cuda10.0.23
```

This example shows how to configure HDFS options in the config.toml file, and then specify that file when starting Driverless AI in Docker. Note that this example enables HDFS with no authentication.

1. Configure the Driverless AI config.toml file. Set the following configuration options. Note that the procsy port, which defaults to 12347, also has to be changed.
 - `enabled_file_systems = "file, upload, hdfs"`
 - `procsy_ip = "127.0.0.1"`
 - `procsy_port = 8080`
2. Mount the config.toml file into the Docker container.

```
nvidia-docker run \
--pid=host \
--init \
--rm \
--shm-size=256m \
--add-host name.node:172.16.2.186 \
-e DRIVERLESS_AI_CONFIG_FILE=/path/in/docker/config.toml \
-p 12345:12345 \
-v /local/path/to/config.toml:/path/in/docker/config.toml \
-v /etc/passwd:/etc/passwd:ro \
-v /etc/group:/etc/group:ro \
-v /tmp/dtmp/:/tmp \
-v /tmp/dlog/:/log \
-v /tmp/dlicense/:/license \
-v /tmp/ddata/:/data \
-u ${id -u}:${id -g} \
harbor.h2o.ai/h2oai/dai-centos7-x86_64:1.9.0-cuda10.0.23
```

This example enables the HDFS data connector and disables HDFS authentication in the config.toml file. This allows users to reference data stored in HDFS directly using the name node address, for example: `hdfs://name.node/datasets/iris.csv`.

1. Export the Driverless AI config.toml file or add it to `~/.bashrc`. For example:

```
# DEB and RPM
export DRIVERLESS_AI_CONFIG_FILE="/etc/dai/config.toml"

# TAR SH
export DRIVERLESS_AI_CONFIG_FILE="/path/to/your/unpacked/dai/directory/config.toml"
```

2. Specify the following configuration options in the config.toml file. Note that the procsy port, which defaults to 12347, also has to be changed.

```

# IP address and port of procsy process.
procsy_ip = "127.0.0.1"
procsy_port = 8080

# File System Support
# upload : standard upload feature
# file : local file system/server file system
# hdfs : Hadoop file system, remember to configure the HDFS config folder path and keytab below
# dtap : Blue Data Tap file system, remember to configure the DTap section below
# s3 : Amazon S3, optionally configure secret and access key below
# gcs : Google Cloud Storage, remember to configure gcs_path_to_service_account_json below
# gbq : Google Big Query, remember to configure gcs_path_to_service_account_json below
# minio : Minio Cloud Storage, remember to configure secret and access key below
# snow : Snowflake Data Warehouse, remember to configure Snowflake credentials below (account name, username, password)
# kdb : KDB+ Time Series Database, remember to configure KDB credentials below (hostname and port, optionally: username, password, ↵
#      classpath, and jvm_args)
# azrbs : Azure Blob Storage, remember to configure Azure credentials below (account name, account key)
# jdbc: JDBC Connector, remember to configure JDBC below. (jdbc_app_configs)
# hive: Hive Connector, remember to configure Hive below. (hive_app_configs)
# recipe_url: load custom recipe from URL
# recipe_file: load custom recipe from local file system
enabled_file_systems = "file, hdfs"

```

- Save the changes when you are done, then stop/restart Driverless AI.

15.2.3 Example 2: Enable HDFS with Keytab-Based Authentication

Notes:

- If using Kerberos Authentication, then the time on the Driverless AI server must be in sync with Kerberos server. If the time difference between clients and DCs are 5 minutes or higher, there will be Kerberos failures.
- If running Driverless AI as a service, then the Kerberos keytab needs to be owned by the Driverless AI user; otherwise Driverless AI will not be able to read/access the Keytab and will result in a fallback to simple authentication and, hence, fail.

Docker Image Installs

Docker Image with the config.toml

Native Installs

This example:

- Places keytabs in the /tmp/dtmp folder on your machine and provides the file path as described below.
- Configures the environment variable DRIVERLESS_AI_HDFS_APP_PRINCIPAL_USER to reference a user for whom the keytab was created (usually in the form of `user@realm`).

```

nvidia-docker run \
--pid=host \
--init \
--rm \
--shm-size=256m \
-e DRIVERLESS_AI_ENABLED_FILE_SYSTEMS="file,hdfs" \
-e DRIVERLESS_AI_HDFS_AUTH_TYPE='keytab' \
-e DRIVERLESS_AI_KEY_TAB_PATH='/tmp/<<keytabname>>' \
-e DRIVERLESS_AI_HDFS_APP_PRINCIPAL_USER='<<user@kerberosrealm>>' \
-e DRIVERLESS_AI_PROCSY_PORT=8080 \
-p 12345:12345 \
-v /etc/passwd:/etc/passwd:ro \
-v /etc/group:/etc/group:ro \
-v /tmp/dtmp:/tmp \
-v /tmp/dlog:/log \
-v /tmp/dlicense:/license \
-v /tmp/ddata:/data \
-u $(id -u):$(id -g) \
harbor.h2o.ai/h2oai/dai-centos7-x86_64:1.9.0-cuda10.0.23

```

This example:

- Places keytabs in the /tmp/dtmp folder on your machine and provides the file path as described below.
 - Configures the option `hdfs_app_principal_user` to reference a user for whom the keytab was created (usually in the form of `user@realm`).
- Configure the Driverless AI config.toml file. Set the following configuration options. Note that the procsy port, which defaults to 12347, also has to be changed.

- enabled_file_systems = "file, upload, hdfs"
- procsy_ip = "127.0.0.1"
- procsy_port = 8080
- hdfs_auth_type = "keytab"
- key_tab_path = "/tmp/<keytabname>"
- hdfs_app_principal_user = "<user@kerberosrealm>"

2. Mount the config.toml file into the Docker container.

```
nvidia-docker run \
--pid=host \
--init \
--rm \
--shm-size=256m \
--add-host name.node:172.16.2.186 \
-e DRIVERLESS_AI_CONFIG_FILE=/path/in/docker/config.toml \
-p 12345:12345 \
-v /local/path/to/config.toml:/path/in/docker/config.toml \
-v /etc/passwd:/etc/passwd:ro \
-v /etc/group:/etc/group:ro \
-v /tmp/dtmp:/tmp \
-v /tmp/dlog:/log \
-v /tmp/license:/license \
-v /tmp/ddata:/data \
-u ${id -u}:${id -g} \
harbor.h2o.ai/h2oai/dai-centos7-x86_64:1.9.0-cudal0.0.23
```

This example:

- Places keytabs in the /tmp/dtmp folder on your machine and provides the file path as described below.
- Configures the option hdfs_app_principal_user to reference a user for whom the keytab was created (usually in the form of user@realm).

1. Export the Driverless AI config.toml file or add it to ~/.bashrc. For example:

```
# DEB and RPM
export DRIVERLESS_AI_CONFIG_FILE="/etc/dai/config.toml"

# TAR SH
export DRIVERLESS_AI_CONFIG_FILE="/path/to/your/unpacked/dai/directory/config.toml"
```

2. Specify the following configuration options in the config.toml file.

```
# IP address and port of procsy process.
procsy_ip = "127.0.0.1"
procsy_port = 8080

# File System Support
# upload : standard upload feature
# file : local file system/server file system
# hdfs : Hadoop file system, remember to configure the HDFS config folder path and keytab below
# dtap : Blue Data Tap file system, remember to configure the DTap section below
# s3 : Amazon S3, optionally configure secret and access key below
# gcs : Google Cloud Storage, remember to configure gcs_path_to_service_account_json below
# gbig : Google Big Query, remember to configure gcs_path_to_service_account_json below
# minio : Minio Cloud Storage, remember to configure secret and access key below
# snow : Snowflake Data Warehouse, remember to configure Snowflake credentials below (account name, username, password)
# kdb : KDB+ Time Series Database, remember to configure KDB credentials below (hostname and port, optionally: username, password, classpath, and jvm_args)
# azrbs : Azure Blob Storage, remember to configure Azure credentials below (account name, account key)
# jdbc: JDBC Connector, remember to configure JDBC below. (jdbc_app_configs)
# hive: Hive Connector, remember to configure Hive below. (hive_app_configs)
# recipe_url: load custom recipe from URL
# recipe_file: load custom recipe from local file system
enabled_file_systems = "file, hdfs"

# HDFS connector
# Auth type can be Principal/keytab/keytabPrincipal
# Specify HDFS Auth Type, allowed options are:
#   noauth : No authentication needed
#   principal : Authenticate with HDFS with a principal user
#   keytab : Authenticate with a Key tab (recommended)
#   keytabimpersonation : Login with impersonation using a keytab
hdfs_auth_type = "keytab"

# Path of the principal key tab file
key_tab_path = "/tmp/<keytabname>"

# Kerberos app principal user (recommended)
hdfs_app_principal_user = "<user@kerberosrealm>"
```

3. Save the changes when you are done, then stop/restart Driverless AI.

15.2.4 Example 3: Enable HDFS with Keytab-Based Impersonation

Notes:

- If using Kerberos, be sure that the Driverless AI time is synched with the Kerberos server.
- If running Driverless AI as a service, then the Kerberos keytab needs to be owned by the Driverless AI user.
- Logins are case sensitive when keytab-based impersonation is configured.

Docker Image Installs

Docker Image with the config.toml

Native Installs

The example:

- Sets the authentication type to keytabimpersonation.
- Places keytabs in the /tmp/dtmp folder on your machine and provides the file path as described below.
- Configures the DRIVERLESS_AI_HDFS_APP_PRINCIPAL_USER variable, which references a user for whom the keytab was created (usually in the form of `user@realm`).

```
nvidia-docker run \
--pid=host \
--init \
--rm \
--shm-size=256m \
-e DRIVERLESS_AI_ENABLED_FILE_SYSTEMS="file,hdfs" \
-e DRIVERLESS_AI_HDFS_AUTH_TYPE='keytabimpersonation' \
-e DRIVERLESS_AI_KEY_TAB_PATH='/tmp/<<keytabname>>' \
-e DRIVERLESS_AI_HDFS_APP_PRINCIPAL_USER='<<appuser@kerberosrealm>>' \
-e DRIVERLESS_AI_PROCSY_PORT=8080 \
-p 12345:12345 \
-v /etc/passwd:/etc/passwd:ro \
-v /etc/group:/etc/group:ro \
-v /tmp/dlog/:/log \
-v /tmp/dlicense/:/license \
-v /tmp/ddata/:/data \
-u $(id -u):$(id -g) \
harbor.h2o.ai/h2oai/dai-centos7-x86_64:1.9.0-cuda10.0.23
```

This example:

- Sets the authentication type to keytabimpersonation.
- Places keytabs in the /tmp/dtmp folder on your machine and provides the file path as described below.
- Configures the hdfs_app_principal_user variable, which references a user for whom the keytab was created (usually in the form of `user@realm`).

1. Configure the Driverless AI config.toml file. Set the following configuration options. Note that the procsy port, which defaults to 12347, also has to be changed.

- `enabled_file_systems = "file, upload, hdfs"`
- `procsy_ip = "127.0.0.1"`
- `procsy_port = 8080`
- `hdfs_auth_type = "keytabimpersonation"`
- `key_tab_path = "/tmp/<keytabname>"`
- `hdfs_app_principal_user = "<user@kerberosrealm>"`

2. Mount the config.toml file into the Docker container.

```
nvidia-docker run \
--pid=host \
--init \
--rm \
--shm-size=256m \
--add-host name.node:172.16.2.186 \
```

(continues on next page)

(continued from previous page)

```
-e DRIVERLESS_AI_CONFIG_FILE=/path/in/docker/config.toml \
-p 12345:12345 \
-v /local/path/to/config.toml:/path/in/docker/config.toml \
-v /etc/passwd:/etc/passwd:ro \
-v /etc/group:/etc/group:ro \
-v /tmp/dtmp/:/tmp \
-v /tmp/dlog/:/log \
-v /tmp/dlicense:/license \
-v /tmp/ddata:/data \
-u ${id -u} ${id -g} \
harbor.h2o.ai/h2oai/dai-centos7-x86_64:1.9.0-cuda10.0.23
```

This example:

- Sets the authentication type to keytab impersonation.
- Places keytabs in the `/tmp/dtmp` folder on your machine and provides the file path as described below.
- Configures the `hdfs_app_principal_user` variable, which references a user for whom the keytab was created (usually in the form of `user@realm`).

1. Export the Driverless AI config.toml file or add it to `~/.bashrc`. For example:

```
# DEB and RPM
export DRIVERLESS_AI_CONFIG_FILE="/etc/dai/config.toml"

# TAR SH
export DRIVERLESS_AI_CONFIG_FILE="/path/to/your/unpacked/dai/directory/config.toml"
```

2. Specify the following configuration options in the config.toml file.

```
# IP address and port of procsy process.
procsy_ip = "127.0.0.1"
procsy_port = 8080

# File System Support
# upload : standard upload feature
# file : local file system/server file system
# hdfs : Hadoop file system, remember to configure the HDFS config folder path and keytab below
# dtap : Blue Data Tap file system, remember to configure the DTap section below
# s3 : Amazon S3, optionally configure secret and access key below
# gcs : Google Cloud Storage, remember to configure gcs_path_to_service_account_json below
# gbig : Google Big Query, remember to configure gcs_path_to_service_account_json below
# minio : Minio Cloud Storage, remember to configure secret and access key below
# snow : Snowflake Data Warehouse, remember to configure Snowflake credentials below (account name, username, password)
# kdb : KDB+ Time Series Database, remember to configure KDB credentials below (hostname and port, optionally: username, password, ↵
#       classpath, and jvm_args)
# azrbs : Azure Blob Storage, remember to configure Azure credentials below (account name, account key)
# jdbc: JDBC Connector, remember to configure JDBC below. (jdbc_app_configs)
# hive: Hive Connector, remember to configure Hive below. (hive_app_configs)
# recipe_url: load custom recipe from URL
# recipe_file: load custom recipe from local file system
enabled_file_systems = "file, hdfs"

# HDFS connector
# Auth type can be Principal/keytab/keytabPrincipal
# Specify HDFS Auth Type, allowed options are:
#   noauth : No authentication needed
#   principal : Authenticate with HDFS with a principal user
#   keytab : Authenticate with a Key tab (recommended)
#   keytabimpersonation : Login with impersonation using a keytab
hdfs_auth_type = "keytabimpersonation"

# Path of the principal key tab file
key_tab_path = "/tmp/<keytabname>"

# Kerberos app principal user (recommended)
hdfs_app_principal_user = "<user@kerberosrealm>"
```

3. Save the changes when you are done, then stop/restart Driverless AI.

15.2.5 Specifying a Hadoop Platform

The following example shows how to build an H2O-3 Hadoop image and run Driverless AI. This example uses CDH 6.0. Change the `H2O_TARGET` to specify a different platform.

1. Clone and then build H2O-3 for CDH 6.0.

```
git clone https://github.com/h2oai/h2o-3.git
cd h2o-3
./gradlew clean build -x test
export H2O_TARGET=cdh6.0
export BUILD_HADOOP=true
./gradlew clean build -x test
```

2. Start H2O.

```
docker run -it --rm \
-v `pwd` \
-w `pwd` \
--entrypoint bash \
--network=host \
-p 8020:8020 \
docker.h2o.ai/cdh-6-w-hive \
-c 'sudo -E startup.sh && \
source /envs/h2o_env_python3.6/bin/activate && \
hadoop jar h2o-hadoop-3/h2o-cdh6.0-assembly/build/libs/h2odriver.jar -libjars "$(cat /opt/hive-jars/hive-libjars)" -n 1 -mapperXmx 2g - \
baseport 54445 -notify h2o_one_node -ea -disown && \
export CLOUD_IP=localhost && \
export CLOUD_PORT=54445 && \
make -f scripts/jenkins/Makefile.jenkins test-hadoop-smoke; \
bash'
```

3. Run the Driverless AI HDFS connector.

```
java -cp h2oai-dai-connectors.jar ai.h2o.dai.connectors.HdfsConnector
```

4. Verify the commands for ls and cp, for example.

```
{"coreSiteXmlPath": "/etc/hadoop/conf", "keyTabPath": "", "authType: "noauth", "srcPath": "hdfs://localhost/user/jenkins/", "dstPath": "/
tmp/xxx", "command": "cp", "user": "", "appUser": ""}
```

15.3 Azure Blob Store Setup

Driverless AI allows you to explore Azure Blob Store data sources from within the Driverless AI application.

Note: Depending on your Docker install version, use either the `docker run --runtime=nvidia` (\geq Docker 19.03) or `nvidia-docker` (< Docker 19.03) command when starting the Driverless AI Docker image. Use `docker version` to check which version of Docker you are using.

15.3.1 Supported Data Sources Using the Azure Blob Store Connector

The following data sources can be used with the Azure Blob Store connector.

- *Azure Blob Storage (general purpose v1)*
- Blob Storage
- *Azure Files (File Storage)*
- *Azure Data Lake Storage Gen 2 (Storage V2)*

The following data sources can be used with the Azure Blob Store connector when also using the HDFS connector.

- *Azure Data Lake Gen 1 (HDFS connector required)*
- *Azure Data Lake Gen 2 (HDFS connector optional)*

15.3.2 Description of Configuration Attributes

The following configuration attributes are specific to enabling Azure Blob Storage.

- `azure_blob_account_name`: The Microsoft Azure Storage account name. This should be the dns prefix created when the account was created (for example, “mystorage”).
- `azure_blob_account_key`: Specify the account key that maps to your account name.
- `azure_connection_string`: Optionally specify a new connection string. With this option, you can include an override for a host, port, and/or account name. For example,

```
azure_connection_string = "DefaultEndpointsProtocol=http;AccountName=<account_name>;AccountKey=<account_key>;BlobEndpoint=http://<host>:<port>/<account_name>;"
```

- `enabled_file_systems`: The file systems you want to enable. This must be configured in order for data connectors to function properly.

The following additional configuration attributes can be used for enabling an HDFS Connector to connect to Azure Data Lake Gen 1 (and optionally with Azure Data Lake Gen 2).

- `hdfs_config_path`: The location the HDFS config folder path. This folder can contain multiple config files.
- `hdfs_app_classpath`: The HDFS classpath.
- `hdfs_app_supported_schemes`: Supported schemas list is used as an initial check to ensure valid input to connector.

15.3.3 Example 1: Enabling the Azure Blob Store Data Connector

Docker Image Installs

Docker Image with the config.toml

Native Installs

This example enables the Azure Blob Store data connector by specifying environment variables when starting the Driverless AI Docker image. This allows users to reference data stored on your Azure storage account using the account name, for example: <https://mystorage.blob.core.windows.net>.

```
nvidia-docker run \
--pid=host \
--init \
--rm \
--shm-size=256m \
-e DRIVERLESS_AI_ENABLED_FILE_SYSTEMS="file,azrbs" \
-e DRIVERLESS_AI_AZURE_BLOB_ACCOUNT_NAME="mystorage" \
-e DRIVERLESS_AI_AZURE_BLOB_ACCOUNT_KEY="" \
-p 12345:12345 \
-v /tmp/dtmp/:/tmp \
-v /tmp/dlog/:/log \
-v /tmp/dlicense/:/license \
-v /tmp/ddata/:/data \
-u $(id -u):$(id -g) \
harbor.h2o.ai/h2oai/dai-centos7-x86_64:1.9.0-cuda10.0.23
```

This example shows how to configure Azure Blob Store options in the config.toml file, and then specify that file when starting Driverless AI in Docker.

1. Configure the Driverless AI config.toml file. Set the following configuration options:
 - `enabled_file_systems = "file, upload, azrbs"`
 - `azure_blob_account_name = "mystorage"`
 - `azure_blob_account_key = "<account_key>"`
2. Mount the config.toml file into the Docker container.

```
nvidia-docker run \
--pid=host \
--init \
--rm \
--shm-size=256m \
--add-host name.node:172.16.2.186 \
-e DRIVERLESS_AI_CONFIG_FILE=/path/in/docker/config.toml \
-p 12345:12345 \
-v /local1/path/to/config.toml:/path/in/docker/config.toml \
-v /etc/passwd:/etc/passwd:ro \
-v /etc/group:/etc/group:ro \
-v /tmp/dtmp/:/tmp \
-v /tmp/dlog/:/log \
-v /tmp/dlicense/:/license \
-v /tmp/ddata/:/data \
-u $(id -u):$(id -g) \
harbor.h2o.ai/h2oai/dai-centos7-x86_64:1.9.0-cuda10.0.23
```

This example shows how to enable the Azure Blob Store data connector in the config.toml file when starting Driverless AI in native installs. This allows users to reference data stored on your Azure storage account using the account name, for example: <https://mystorage.blob.core.windows.net>.

1. Export the Driverless AI config.toml file or add it to `~/.bashrc`. For example:

```
# DEB and RPM
export DRIVERLESS_AI_CONFIG_FILE="/etc/dai/config.toml"

# TAR SH
export DRIVERLESS_AI_CONFIG_FILE="/path/to/your/unpacked/dai/directory/config.toml"
```

2. Specify the following configuration options in the config.toml file.

```
# File System Support
# upload : standard upload feature
# file : local file system/server file system
# hdfs : Hadoop file system, remember to configure the HDFS config folder path and keytab below
# dtap : Blue Data Tap file system, remember to configure the DTap section below
# s3 : Amazon S3, optionally configure secret and access key below
# gcs : Google Cloud Storage, remember to configure gcs_path_to_service_account_json below
# gbg : Google Big Query, remember to configure gcs_path_to_service_account_json below
# minio : Minio Cloud Storage, remember to configure secret and access key below
# snow : Snowflake Data Warehouse, remember to configure Snowflake credentials below (account name, username, password)
# kdb : KDB+ Time Series Database, remember to configure KDB credentials below (hostname and port, optionally: username, password, classpath, and jvm_args)
# azrbs : Azure Blob Storage, remember to configure Azure credentials below (account name, account key)
# jdbc: JDBC Connector, remember to configure JDBC below. (jdbc_app_configs)
# hive: Hive Connector, remember to configure Hive below. (hive_app_configs)
# recipe_url: load custom recipe from URL
# recipe_file: load custom recipe from local file system
enabled_file_systems = "file, azrbs"

# Azure Blob Store Connector credentials
azure_blob_account_name = "mystorage"
azure_blob_account_key = "<account_key>"
```

3. Save the changes when you are done, then stop/restart Driverless AI.

15.3.4 Example 2: Mount Azure File Shares to the Local File System

Supported Data Sources Using the Local File System

- Azure Files (File Storage)

Mounting Azure File Shares

Azure file shares can be mounted into the Local File system of Driverless AI. To mount the Azure file share, follow the steps listed on <https://docs.microsoft.com/en-us/azure/storage/files/storage-how-to-use-files-linux>.

15.3.5 Example 3: Enable HDFS Connector to Connect to Azure Data Lake Gen 1

This example enables the HDFS Connector to connect to Azure Data Lake Gen1. This allows users to reference data stored on your Azure Data Lake using the adl uri, for example: `adl://myadl.azuredatastorage.net`.

Docker Image with the config.toml

Native Installs

1. Create an Azure AD web application for service-to-service authentication: <https://docs.microsoft.com/en-us/azure/data-lake-store/data-lake-store-service-to-service-authenticate-using-active-directory>
2. Add the information from your web application to the Hadoop `core-site.xml` configuration file:

```
<configuration>
<property>
  <name>fs.adl.oauth2.access.token.provider.type</name>
  <value>ClientCredential</value>
</property>
<property>
```

(continues on next page)

(continued from previous page)

```
<name>fs.adl.oauth2.refresh.url</name>
<value>Token endpoint created in step 1.</value>
</property>
<property>
<name>fs.adl.oauth2.client.id</name>
<value>Client ID created in step 1</value>
</property>
<property>
<name>fs.adl.oauth2.credential</name>
<value>Client Secret created in step 1</value>
</property>
<property>
<name>fs.defaultFS</name>
<value>ADL URIt</value>
</property>
</configuration>
```

3. Take note of the Hadoop Classpath and add the `azure-datalake-store.jar` file. This file can be found on any Hadoop version in: `$HADOOP_HOME/share/hadoop/tools/lib/*`.

```
echo "$HADOOP_CLASSPATH:$HADOOP_HOME/share/hadoop/tools/lib/*"
```

4. Configure the Driverless AI config.toml file. Set the following configuration options:

```
enabled_file_systems = "upload, file, hdfs, azrbz, recipe_file, recipe_url"
hdfs_config_path = "/path/to/hadoop/conf"
hdfs_app_classpath = "/hadoop/classpath/"
hdfs_app_supported_schemes = ["adl://"]"
```

5. Mount the config.toml file into the Docker container.

```
nvidia-docker run \
--pid=host \
--init \
--rm \
--shm-size=256m \
--add-host name.node:172.16.2.186 \
-e DRIVERLESS_AI_CONFIG_FILE=/path/in/docker/config.toml \
-p 12345:12345 \
-v /local/path/to/config.toml:/path/in/docker/config.toml \
-v /etc/passwd:/etc/passwd:ro \
-v /etc/group:/etc/group:ro \
-v /tmp/dtmp/:/tmp \
-v /tmp/dlog/:/log \
-v /tmp/dlicense/:/license \
-v /tmp/ddata/:/data \
-u ${id}-u:${id}-g \
h2oai/dai-centos7-x86_64:1.9.0-cudal0.0.23
```

1. Create an Azure AD web application for service-to-service authentication. <https://docs.microsoft.com/en-us/azure/data-lake-store/data-lake-store-service-to-service-authenticate-using-active-directory>
2. Add the information from your web application to the hadoop core-site.xml configuration file:

```
<configuration>
<property>
<name>fs.adl.oauth2.access.token.provider.type</name>
<value>ClientCredential</value>
</property>
<property>
<name>fs.adl.oauth2.refresh.url</name>
<value>Token endpoint created in step 1.</value>
</property>
<property>
<name>fs.adl.oauth2.client.id</name>
<value>Client ID created in step 1</value>
</property>
<property>
<name>fs.adl.oauth2.credential</name>
<value>Client Secret created in step 1</value>
</property>
<property>
<name>fs.defaultFS</name>
<value>ADL URIt</value>
</property>
</configuration>
```

3. Take note of the Hadoop Classpath and add the `azure-datalake-store.jar` file. This file can be found on any hadoop version in: `$HADOOP_HOME/share/hadoop/tools/lib/*`

```
echo "$HADOOP_CLASSPATH:$HADOOP_HOME/share/hadoop/tools/lib/*"
```

4. Configure the Driverless AI config.toml file. Set the following configuration options:

```

enabled_file_systems = "upload, file, hdfs, azrbs, recipe_file, recipe_url"
hdfs_config_path = "/path/to/hadoop/conf"
hdfs_app_classpath = "/hadoop/classpath/"
hdfs_app_supported_schemes = "['adl://']"

```

- Save the changes when you are done, then stop/restart Driverless AI.

15.3.6 Example 4: Enable HDFS Connector to Connect to Azure Data Lake Gen 2

This example enables the HDFS Connector to connect to Azure Data Lake Gen2. This allows users to reference data stored on your Azure Data Lake using the Azure Blob Fyle System Driver, for example: abfs[s]://file_system@account_name.dfs.core.windows.net/<path>/<path>/<file_name>.

Docker Image with the config.toml

Native Installs

- Create an Azure Service Principal: <https://docs.microsoft.com/en-us/azure/active-directory/develop/howto-create-service-principal-portal>
- Grant permissions to the Service Principal created on step 1 to access blobs: <https://docs.microsoft.com/en-us/azure/storage/common/storage-auth-aad>
- Add the information from your web application to the Hadoop core-site.xml configuration file:

```

<configuration>
  <property>
    <name>fs.azure.account.auth.type</name>
    <value>OAuth</value>
  </property>
  <property>
    <name>fs.azure.account.oauth.provider.type</name>
    <value>org.apache.hadoop.fs.azurebfs.ClientCredsTokenProvider</value>
  </property>
  <property>
    <name>fs.azure.account.oauth2.client.endpoint</name>
    <value>Token endpoint created in step 1.</value>
  </property>
  <property>
    <name>fs.azure.account.oauth2.client.id</name>
    <value>Client ID created in step 1</value>
  </property>
  <property>
    <name>fs.azure.account.oauth2.client.secret</name>
    <value>Client Secret created in step 1</value>
  </property>
</configuration>

```

- Take note of the Hadoop Classpath and add the required jar files. These files can found on any Hadoop version 3.2 or higher at: \$HADOOP_HOME/share/hadoop/tools/lib/*

```
echo "$HADOOP_CLASSPATH:$HADOOP_HOME/share/hadoop/tools/lib/*"
```

Note: ABFS is only supported for Hadoop version 3.2 or higher.

- Configure the Driverless AI config.toml file. Set the following configuration options:

```

enabled_file_systems = "upload, file, hdfs, azrbs, recipe_file, recipe_url"
hdfs_config_path = "/path/to/hadoop/conf"
hdfs_app_classpath = "/hadoop/classpath/"
hdfs_app_supported_schemes = "['abfs://']"

```

- Mount the config.toml file into the Docker container.

```

nvidia-docker run \
--pid=host \
--init \
--rm \
--shm-size=256m \
--add-host name.node:172.16.2.186 \
-e DRIVERLESS_AI_CONFIG_FILE=/path/in/docker/config.toml \
-p 12345:12345 \
-v /local/path/to/config.toml:/path/in/docker/config.toml \
-v /etc/passwd:/etc/passwd:ro \
-v /etc/group:/etc/group:ro \
-v /tmp/dtmp/:/tmp \
-v /tmp/dlog/:/log \
-v /tmp/dlicense/:/license \

```

(continues on next page)

(continued from previous page)

```
-v /tmp/ddata/:/data \
-u ${id}-u:${id}-g \
h2oai/dai-centos7-x86_64:1.9.0-cudal0.0.23
```

1. Create an Azure Service Principal. <https://docs.microsoft.com/en-us/azure/active-directory/develop/howto-create-service-principal-portal>
2. Grant permissions to the Service Principal created on step 1 to access blobs: <https://docs.microsoft.com/en-us/azure/storage/common/storage-auth-aad>
3. Add the information from your web application to the hadoop core-site.xml configuration file:

```
<configuration>
<property>
<name>fs.azure.account.auth.type</name>
<value>OAuth</value>
</property>
<property>
<name>fs.azure.account.oauth.provider.type</name>
<value>org.apache.hadoop.fs.azurebfs.oauth2.ClientCredsTokenProvider</value>
</property>
<property>
<name>fs.azure.account.oauth2.client.endpoint</name>
<value>Token endpoint created in step 1.</value>
</property>
<property>
<name>fs.azure.account.oauth2.client.id</name>
<value>Client ID created in step 1</value>
</property>
<property>
<name>fs.azure.account.oauth2.client.secret</name>
<value>Client Secret created in step 1</value>
</property>
</configuration>
```

4. Take note of the Hadoop Classpath and add the required jar files. These files can found on any hadoop version 3.2 or higher at: \$HADOOP_HOME/share/hadoop/tools/lib/*

```
echo "$HADOOP_CLASSPATH:$HADOOP_HOME/share/hadoop/tools/lib/*"
```

Note: ABFS is only supported for hadoop version 3.2 or higher

5. Configure the Driverless AI config.toml file. Set the following configuration options:

```
enabled_file_systems = "upload, file, hdfs, azrbs, recipe_file, recipe_url"
hdfs_config_path = "/path/to/hadoop/conf"
hdfs_app_classpath = "/hadoop/classpath/"
hdfs_app_supported_schemes = "['abfs://']"
```

6. Save the changes when you are done, then stop/restart Driverless AI.

FAQ

Can I connect to my storage account using Private Endpoints?

Yes. Driverless AI can use privated endpoints if Driverless AI is located in the allowed VNET.

Does Driverless AI support secure transfer?

Yes. The Azure Blob Store Connector make all connections over HTTPS.

Does Driverless AI support hierarchical namespaces?

Yes.

Can I use Azure Managed Identities (MSI) to access the DataLake?

Yes. If Driverless AI is running on an Azure VM with managed identities. To enable the HDFS Connector to use MSI to authenticate, add to the core-site.xml:

For Gen1:

```
<property>
  <name>fs.adl.oauth2.access.token.provider.type</name>
  <value>MSI</value>
</property>
```

For Gen2:

```
<property>
  <name>fs.azure.account.auth.type</name>
  <value>OAuth</value>
</property>
<property>
  <name>fs.azure.account.oauth.provider.type</name>
  <value>org.apache.hadoop.fs.azurebfs.oauth2.MsiTokenProvider</value>
</property>
```

15.4 BlueData DataTap Setup

This section provides instructions for configuring Driverless AI to work with BlueData DataTap.

Note: Depending on your Docker install version, use either the `docker run --runtime=nvidia` (\geq Docker 19.03) or `nvidia-docker` (< Docker 19.03) command when starting the Driverless AI Docker image. Use `docker version` to check which version of Docker you are using.

15.4.1 Description of Configuration Attributes

- `dtap_auth_type`: Selects DTAP authentication. Available values are:
 - `noauth`: No authentication needed
 - `principal`: Authenticate with DataTap with a principal user
 - `keytab`: Authenticate with a Key tab (recommended). If running Driverless AI as a service, then the Kerberos keytab needs to be owned by the Driverless AI user.
 - `keytabimpersonation`: Login with impersonation using a keytab
- `dtap_config_path`: The location of the DTAP (HDFS) config folder path. This folder can contain multiple config files. **Note:** The DTAP config file `core-site.xml` needs to contain DTap FS configuration, for example:

```
<configuration>
  <property>
    <name>fs.dtap.impl</name>
    <value>com.bluedata.hadoop.bdfs.Bdfs</value>
    <description>The FileSystem for BlueData dtap: URIs.</description>
  </property>
</configuration>
```

- `dtap_key_tab_path`: The path of the principal key tab file. For use when `dtap_auth_type=principal`.
- `dtap_app_principal_user`: The Kerberos app principal user (recommended).
- `dtap_app_login_user`: The user ID of the current user (for example, `user@realm`).
- `dtap_app_jvm_args`: JVM args for DTap distributions. Separate each argument with spaces.
- `dtap_app_classpath`: The DTap classpath.
- `enabled_file_systems`: The file systems you want to enable. This must be configured in order for data connectors to function properly.

15.4.2 Example 1: Enable DataTap with No Authentication

Docker Image Installs

Docker Image with the config.toml

Native Installs

This example enables the DataTap data connector and disables authentication. It does not pass any configuration file; however it configures Docker DNS by passing the name and IP of the DTap name node. This allows users to reference data stored in DTap directly using the name node address, for example: `dtap://name.node/datasets/iris.csv` or `dtap://name.node/datasets/`. (**Note:** The trailing slash is currently required for directories.)

```
nvidia-docker run \
--pid=host \
--init \
--rm \
--shm-size=256m \
--add-host name.node:172.16.2.186 \
-e DRIVERLESS_AI_ENABLED_FILE_SYSTEMS="file,dtap" \
-e DRIVERLESS_AI_DTCP_AUTH_TYPE='noauth' \
-p 12345:12345 \
-v /etc/passwd:/etc/passwd \
-v /tmp/dtmp:/tmp \
-v /tmp/dlog:/log \
-v /tmp/dlicense:/license \
-v /tmp/ddata:/data \
-u $(id -u):$(id -g) \
harbor.h2o.ai/h2oai/dai-centos7-x86_64:1.9.0-cuda10.0.23
```

This example shows how to configure DataTap options in the config.toml file, and then specify that file when starting Driverless AI in Docker. Note that this example enables DataTap with no authentication.

1. Configure the Driverless AI config.toml file. Set the following configuration options:

- `enabled_file_systems = "file, upload, dtap"`

2. Mount the config.toml file into the Docker container.

```
nvidia-docker run \
--pid=host \
--init \
--rm \
--shm-size=256m \
--add-host name.node:172.16.2.186 \
-e DRIVERLESS_AI_CONFIG_FILE=/path/in/docker/config.toml \
-p 12345:12345 \
-v /local/path/to/config.toml:/path/in/docker/config.toml \
-v /etc/passwd:/etc/passwd:ro \
-v /etc/group:/etc/group:ro \
-v /tmp/dtmp:/tmp \
-v /tmp/dlog:/log \
-v /tmp/dlicense:/license \
-v /tmp/ddata:/data \
-u $(id -u):$(id -g) \
harbor.h2o.ai/h2oai/dai-centos7-x86_64:1.9.0-cuda10.0.23
```

This example enables the DataTap data connector and disables authentication in the config.toml file. This allows users to reference data stored in DataTap directly using the name node address, for example: `dtap://name.node/datasets/iris.csv` or `dtap://name.node/datasets/`. (**Note:** The trailing slash is currently required for directories.)

1. Export the Driverless AI config.toml file or add it to `~/.bashrc`. For example:

```
# DEB and RPM
export DRIVERLESS_AI_CONFIG_FILE="/etc/dai/config.toml"

# TAR SH
export DRIVERLESS_AI_CONFIG_FILE="/path/to/your/unpacked/dai/directory/config.toml"
```

2. Specify the following configuration options in the config.toml file.

```
# File System Support
# upload : standard upload feature
# file : local file system/server file system
# hdfs : Hadoop file system, remember to configure the HDFS config folder path and keytab below
# dtap : Blue Data Tap file system, remember to configure the DTap section below
# s3 : Amazon S3, optionally configure secret and access key below
# gcs : Google Cloud Storage, remember to configure gcs_path_to_service_account_json below
# gbigq : Google Big Query, remember to configure gbigq_path_to_service_account_json below
# minio : Minio Cloud Storage, remember to configure secret and access key below
```

(continues on next page)

(continued from previous page)

```
# snow : Snowflake Data Warehouse, remember to configure Snowflake credentials below (account name, username, password)
# kdb : KDB+ Time Series Database, remember to configure KDB credentials below (hostname and port, optionally: username, password, ↵
# classpath, and jvm_args)
# azrbs : Azure Blob Storage, remember to configure Azure credentials below (account name, account key)
# jdbc: JDBC Connector, remember to configure JDBC below. (jdbc_app_configs)
# hive: Hive Connector, remember to configure Hive below. (hive_app_configs)
# recipe_url: load custom recipe from URL
# recipe_file: load custom recipe from local file system
enabled_file_systems = "file, dtap"
```

- Save the changes when you are done, then stop/restart Driverless AI.

15.4.3 Example 2: Enable DataTap with Keytab-Based Authentication

Notes:

- If using Kerberos Authentication, the time on the Driverless AI server must be in sync with Kerberos server. If the time difference between clients and DCs are 5 minutes or higher, there will be Kerberos failures.
- If running Driverless AI as a service, then the Kerberos keytab needs to be owned by the Driverless AI user; otherwise Driverless AI will not be able to read/access the Keytab and will result in a fallback to simple authentication and, hence, fail.

Docker Image Installs

Docker Image with the config.toml

Native Installs

This example:

- Places keytabs in the /tmp/dtmp folder on your machine and provides the file path as described below.
- Configures the environment variable DRIVERLESS_AI_DTAP_APP_PRINCIPAL_USER to reference a user for whom the keytab was created (usually in the form of `user@realm`).

```
nvidia-docker run \
--pid=host \
--init \
--rm \
--shm-size=256m \
-e DRIVERLESS_AI_ENABLED_FILE_SYSTEMS="file,dtap" \
-e DRIVERLESS_AI_DTAP_AUTH_TYPE='keytab' \
-e DRIVERLESS_AI_DTAP_KEY_TAB_PATH='/tmp/<<keytabname>>' \
-e DRIVERLESS_AI_DTAP_APP_PRINCIPAL_USER='<<user@kerberosrealm>>' \
-p 12345:12345 \
-v /etc/passwd:/etc/passwd \
-v /tmp/dtmp/:/tmp \
-v /tmp/dlog/:/log \
-v /tmp/dlicense/:/license \
-v /tmp/ddata/:/data \
-u $(id -u):$(id -g) \
harbor.h2o.ai/h2oai/dai-centos7-x86_64:1.9.0-cuda10.0.23
```

This example:

- Places keytabs in the /tmp/dtmp folder on your machine and provides the file path as described below.
- Configures the option dtap_app_principal_user to reference a user for whom the keytab was created (usually in the form of `user@realm`).

- Configure the Driverless AI config.toml file. Set the following configuration options:

- `enabled_file_systems = "file, upload, dtap"`
- `dtap_auth_type = "keytab"`
- `dtap_key_tab_path = "/tmp/<keytabname>"`
- `dtap_app_principal_user = "<user@kerberosrealm>"`

- Mount the config.toml file into the Docker container.

```
nvidia-docker run \
--pid=host \
--init \
--rm \
--shm-size=256m \
--add-host name.node:172.16.2.186 \
-e DRIVERLESS_AI_CONFIG_FILE=/path/in/docker/config.toml \
-p 12345:12345 \
-v /local/path/to/config.toml:/path/in/docker/config.toml \
-v /etc/passwd:/etc/passwd:ro \
-v /etc/group:/etc/group:ro \
-v /tmp/dtmp:/tmp \
-v /tmp/diag/:/log \
-v /tmp/dlicense/:/license \
-v /tmp/ddata/:/data \
-u $(id -u):$(id -g) \
harbor.h2o.ai/h2oai/dai-centos7-x86_64:1.9.0-cuda10.0.23
```

This example:

- Places keytabs in the /tmp/dtmp folder on your machine and provides the file path as described below.
 - Configures the option dtap_app_principal_user to reference a user for whom the keytab was created (usually in the form of `user@realm`).
- Export the Driverless AI config.toml file or add it to `~/.bashrc`. For example:

```
# DEB and RPM
export DRIVERLESS_AI_CONFIG_FILE="/etc/dai/config.toml"

# TAR SH
export DRIVERLESS_AI_CONFIG_FILE="/path/to/your/unpacked/dai/directory/config.toml"
```

- Specify the following configuration options in the config.toml file.

```
# File System Support
# upload : standard upload feature
# file : local file system/server file system
# hdfs : Hadoop file system, remember to configure the HDFS config folder path and keytab below
# dtap : Blue Data Tap file system, remember to configure the DTap section below
# s3 : Amazon S3, optionally configure secret and access key below
# gcs : Google Cloud Storage, remember to configure gcs_path_to_service_account_json below
# gbq : Google Big Query, remember to configure gcs_path_to_service_account_json below
# minio : Minio Cloud Storage, remember to configure secret and access key below
# snow : Snowflake Data Warehouse, remember to configure Snowflake credentials below (account name, username, password)
# kdb : KDB+ Time Series Database, remember to configure KDB credentials below (hostname and port, optionally: username, password, classpath, and jvm_args)
# azrbs : Azure Blob Storage, remember to configure Azure credentials below (account name, account key)
# jdbc: JDBC Connector, remember to configure JDBC below. (jdbc_app_configs)
# hive: Hive Connector, remember to configure Hive below. (hive_app_configs)
# recipe_url: load custom recipe from URL
# recipe_file: load custom recipe from local file system
enabled_file_systems = "file, dtap"

# Blue Data DTap connector settings are similar to HDFS connector settings.
#
# Specify DTap Auth Type, allowed options are:
#   noauth : No authentication needed
#   principal : Authenticate with DTap with a principal user
#   keytab : Authenticate with a Key tab (recommended). If running
#           DAI as a service, then the Kerberos keytab needs to
#           be owned by the DAI user.
#   keytabimpersonation : Login with impersonation using a keytab
dtap_auth_type = "keytab"

# Path of the principal key tab file
dtap_key_tab_path = "/tmp/<keytabname>"

# Kerberos app principal user (recommended)
dtap_app_principal_user = "<user@kerberosrealm>"
```

- Save the changes when you are done, then stop/restart Driverless AI.

15.4.4 Example 3: Enable DataTap with Keytab-Based Impersonation

Notes:

- If using Kerberos, be sure that the Driverless AI time is synched with the Kerberos server.
- If running Driverless AI as a service, then the Kerberos keytab needs to be owned by the Driverless AI user.

Docker Image Installs

Docker Image with the config.toml

Native Installs

This example:

- Places keytabs in the `/tmp/dtmp` folder on your machine and provides the file path as described below.
- Configures the `DRIVERLESS_AI_DTAP_APP_PRINCIPAL_USER` variable, which references a user for whom the keytab was created (usually in the form of `user@realm`).
- Configures the `DRIVERLESS_AI_DTAP_APP_LOGIN_USER` variable, which references a user who is being impersonated (usually in the form of `user@realm`).

```
# Docker instructions
nvidia-docker run \
--pid=host \
--init \
--rm \
--shm-size=256m \
-e DRIVERLESS_AI_ENABLED_FILE_SYSTEMS="file,dtap" \
-e DRIVERLESS_AI_DTAP_AUTH_TYPE='keytabimpersonation' \
-e DRIVERLESS_AI_DTAP_KEY_TAB_PATH='/tmp/<keytabname>' \
-e DRIVERLESS_AI_DTAP_APP_PRINCIPAL_USER='<appuser@kerberosrealm>' \
-e DRIVERLESS_AI_DTAP_APP_LOGIN_USER='<thisuser@kerberosrealm>' \
-p 12345:12345 \
-v /etc/passwd:/etc/passwd \
-v /tmp/dtmp/:/tmp \
-v /tmp/dlog/:/log \
-v /tmp/dlicense/:/license \
-v /tmp/ddata/:/data \
-u $(id -u):$(id -g) \
harbor.h2o.ai/h2oai/dai-centos7-x86_64:1.9.0-cuda10.0.23
```

This example:

- Places keytabs in the `/tmp/dtmp` folder on your machine and provides the file path as described below.
- Configures the `dtap_app_principal_user` variable, which references a user for whom the keytab was created (usually in the form of `user@realm`).
- Configures the `dtap_app_login_user` variable, which references a user who is being impersonated (usually in the form of `user@realm`).

1. Configure the Driverless AI config.toml file. Set the following configuration options:

- `enabled_file_systems = "file, upload, dtap"`
- `dtap_auth_type = "keytabimpersonation"`
- `dtap_key_tab_path = "/tmp/<keytabname>"`
- `dtap_app_principal_user = "<user@kerberosrealm>"`
- `dtap_app_login_user = "<user@realm>"`

2. Mount the config.toml file into the Docker container.

```
nvidia-docker run \
--pid=host \
--init \
--rm \
--shm-size=256m \
--add-host name.node:172.16.2.186 \
-e DRIVERLESS_AI_CONFIG_FILE=/path/in/docker/config.toml \
-p 12345:12345 \
-v /local/path/to/config.toml:/path/in/docker/config.toml \
-v /etc/passwd:/etc/passwd:ro \
-v /etc/group:/etc/group:ro \
-v /tmp/dtmp/:/tmp \
-v /tmp/dlog/:/log \
-v /tmp/dlicense/:/license \
-v /tmp/ddata/:/data \
-u $(id -u):$(id -g) \
harbor.h2o.ai/h2oai/dai-centos7-x86_64:1.9.0-cuda10.0.23
```

This example:

- Places keytabs in the `/tmp/dtmp` folder on your machine and provides the file path as described below.
- Configures the `dtap_app_principal_user` variable, which references a user for whom the keytab was created (usually in the form of `user@realm`).

- Configures the `dtap_app_login_user` variable, which references a user who is being impersonated (usually in the form of `user@realm`).

1. Export the Driverless AI config.toml file or add it to `~/.bashrc`. For example:

```
# DEB and RPM
export DRIVERLESS_AI_CONFIG_FILE="/etc/dai/config.toml"

# TAR SH
export DRIVERLESS_AI_CONFIG_FILE="/path/to/your/unpacked/dai/directory/config.toml"
```

2. Specify the following configuration options in the config.toml file.

```
# File System Support
# upload : standard upload feature
# file : local file system/server file system
# hdfs : Hadoop file system, remember to configure the HDFS config folder path and keytab below
# dtap : Blue Data Tap file system, remember to configure the DTap section below
# s3 : Amazon S3, optionally configure secret and access key below
# gcs : Google Cloud Storage, remember to configure gcs_path_to_service_account_json below
# gbq : Google Big Query, remember to configure gcs_path_to_service_account_json below
# minio : Minio Cloud Storage, remember to configure secret and access key below
# snow : Snowflake Data Warehouse, remember to configure Snowflake credentials below (account name, username, password)
# kdb : KDB+ Time Series Database, remember to configure KDB credentials below (hostname and port, optionally: username, password, ↳
#       classpath, and jvm_args)
# azrbs : Azure Blob Storage, remember to configure Azure credentials below (account name, account key)
# jdbc: JDBC Connector, remember to configure JDBC below. (jdbc_app_configs)
# hive: Hive Connector, remember to configure Hive below. (hive_app_configs)
# recipe_uri: load custom recipe from URL
# recipe_file: load custom recipe from local file system
enabled_file_systems = "file, dtap"

# Blue Data DTap connector settings are similar to HDFS connector settings.
#
# Specify DTap Auth Type, allowed options are:
#   noauth : No authentication needed
#   principal : Authenticate with DTap with a principal user
#   keytab : Authenticate with a Key tab (recommended). If running
#             DAI as a service, then the Kerberos keytab needs to
#             be owned by the DAI user.
#   keytabimpersonation : Login with impersonation using a keytab
dtap_auth_type = "keytabimpersonation"

# Path of the principal key tab file
dtap_key_tab_path = "/tmp/<keytabname>"

# Kerberos app principal user (recommended)
dtap_app_principal_user = "<user@kerberosrealm>"

# Specify the user id of the current user here as user@realm
dtap_app_login_user = "<user@realm>"
```

3. Save the changes when you are done, then stop/restart Driverless AI.

15.5 Google BigQuery Setup

Driverless AI allows you to explore Google BigQuery data sources from within the Driverless AI application. This section provides instructions for configuring Driverless AI to work with Google BigQuery. This setup requires you to enable authentication. If you enable the GCS and/or GBQ connectors, those file systems will be available in the UI, but you will not be able to use those connectors without authentication.

In order to enable the GBQ data connector with authentication, you must:

1. Retrieve a JSON authentication file from GCP.
2. Mount the JSON file to the Docker instance.
3. Specify the path to the `/json_auth_file.json` with the `gcs_path_to_service_account_json` config option.

Notes:

- The account JSON includes authentications as provided by the system administrator. You can be provided a JSON file that contains both Google Cloud Storage and Google BigQuery authentications, just one or the other, or none at all.
- Depending on your Docker install version, use either the `docker run --runtime=nvidia (>= Docker 19.03)` or `nvidia-docker (< Docker 19.03)` command when starting the Driverless AI Docker image. Use `docker version` to check which version of Docker you are using.

15.5.1 Enable GBQ with Authentication

Docker Image Installs

Docker Image with the config.toml

Native Installs

This example enables the GBQ data connector with authentication by passing the JSON authentication file. This assumes that the JSON file contains Google BigQuery authentications.

```
nvidia-docker run \
--pid=host \
--init \
--rm \
--shm-size=256m \
-e DRIVERLESS_AI_ENABLED_FILE_SYSTEMS="file,gbq" \
-e DRIVERLESS_AI_GCS_PATH_TO_SERVICE_ACCOUNT_JSON="/service_account_json.json" \
-u id -u : id -g \
-p 12345:12345 \
-v `pwd`:/data \
-v `pwd`:/log \
-v `pwd`/license \
-v `pwd`/tmp \
-v `pwd`/service_account_json.json:/service_account_json.json \
harbor.h2o.ai/h2oai/dai-centos7-x86_64:1.9.0-cuda10.0.23
```

This example shows how to configure the GBQ data connector options in the config.toml file, and then specify that file when starting Driverless AI in Docker.

1. Configure the Driverless AI config.toml file. Set the following configuration options:
 - enabled_file_systems = "file, upload, gbq"
 - gcs_path_to_service_account_json = "/service_account_json.json"
2. Mount the config.toml file into the Docker container.

```
nvidia-docker run \
--pid=host \
--init \
--rm \
--shm-size=256m \
--add-host name.node:172.16.2.186 \
-e DRIVERLESS_AI_CONFIG_FILE=/path/in/docker/config.toml \
-p 12345:12345 \
-v /local/path/to/config.toml:/path/in/docker/config.toml \
-v /etc/passwd:/etc/passwd:ro \
-v /etc/group:/etc/group:ro \
-v /tmp/dtmp/:/tmp \
-v /tmp/dlog/:/log \
-v /tmp/dlicense/:/license \
-v /tmp/ddata/:/data \
-u $id -u:$id -g \
harbor.h2o.ai/h2oai/dai-centos7-x86_64:1.9.0-cuda10.0.23
```

This example enables the GBQ data connector with authentication by passing the JSON authentication file. This assumes that the JSON file contains Google BigQuery authentications.

1. Export the Driverless AI config.toml file or add it to `~/.bashrc`. For example:

```
# DEB and RPM
export DRIVERLESS_AI_CONFIG_FILE="/etc/dai/config.toml"

# TAR SH
export DRIVERLESS_AI_CONFIG_FILE="/path/to/your/unpacked/dai/directory/config.toml"
```

2. Specify the following configuration options in the config.toml file.

```
# File System Support
# upload : standard upload feature
# file : local file system/server file system
# hdfs : Hadoop file system, remember to configure the HDFS config folder path and keytab below
# dtap : Blue Data Tap file system, remember to configure the DTap section below
# s3 : Amazon S3, optionally configure secret and access key below
# gcs : Google Cloud Storage, remember to configure gcs_path_to_service_account_json below
# gbq : Google Big Query, remember to configure gcs_path_to_service_account_json below
# minio : Minio Cloud Storage, remember to configure secret and access key below
# snow : Snowflake Data Warehouse, remember to configure Snowflake credentials below (account name, username, password)
# kdb : KDB+ Time Series Database, remember to configure KDB credentials below (hostname and port, optionally: username, password, classpath, and jvm_args)
# azrbs : Azure Blob Storage, remember to configure Azure credentials below (account name, account key)
# jdbc: JDBC Connector, remember to configure JDBC below. (jdbc_app_configs)
# hive: Hive Connector, remember to configure Hive below. (hive_app_configs)
```

(continues on next page)

(continued from previous page)

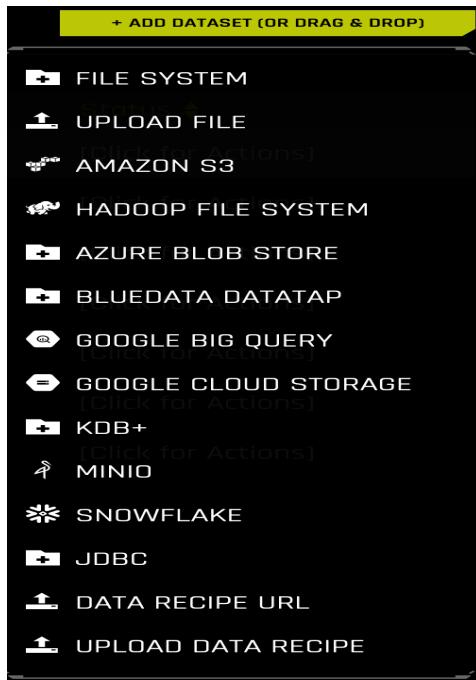
```
# recipe_url: load custom recipe from URL
# recipe_file: load custom recipe from local file system
enabled_file_systems = "file, gbq"

# GCS Connector credentials
# example (suggested) -- "/licenses/my_service_account.json"
gcs_path_to_service_account_json = "/service_account.json"
```

3. Save the changes when you are done, then stop/restart Driverless AI.

15.5.2 Adding Datasets Using GBQ

After Google BigQuery is enabled, you can add datasets by selecting **Google Big Query** from the **Add Dataset (or Drag and Drop)** drop-down menu.

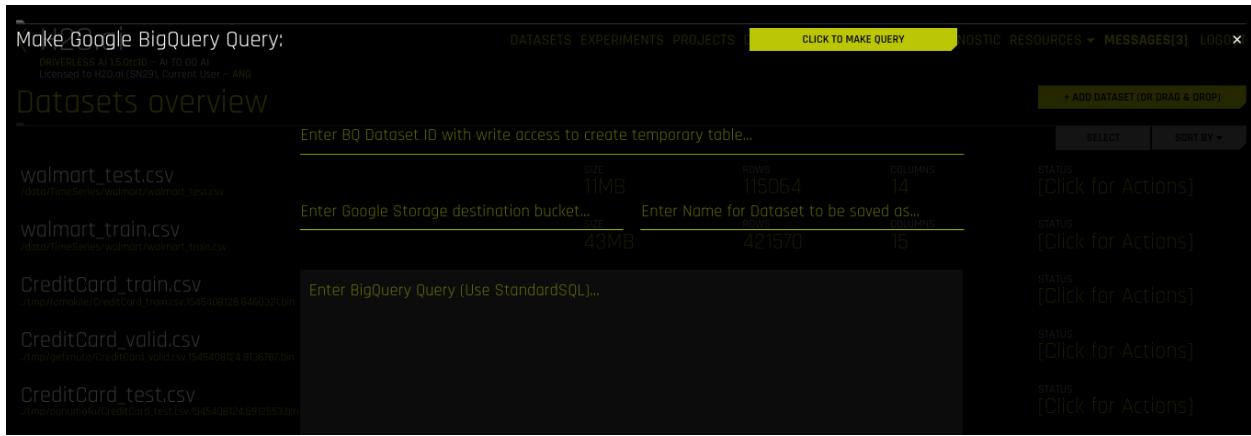


Specify the following information to add your dataset.

1. **Enter BQ Dataset ID with write access to create temporary table:** Enter a dataset ID in Google BigQuery that this user has read/write access to. BigQuery uses this dataset as the location for the new table generated by the query.

Note: Driverless AI's connection to GBQ will inherit the top-level directory from the service JSON file. So if a dataset named "my-dataset" is in a top-level directory named "dai-gbq", then the value for the dataset ID input field would be "my-dataset" and not "dai-gbq:my-dataset".

2. **Enter Google Storage destination bucket:** Specify the name of Google Cloud Storage destination bucket. Note that the user must have write access to this bucket.
3. **Enter Name for Dataset to be saved as:** Specify a name for the dataset, for example, `my_file`.
4. **Enter BigQuery Query (Use StandardSQL):** Enter a StandardSQL query that you want BigQuery to execute. For example: `SELECT * FROM <my_dataset>.<my_table>`.
5. When you are finished, select the **Click to Make Query** button to add the dataset.



15.6 Google Cloud Storage Setup

Driverless AI allows you to explore Google Cloud Storage data sources from within the Driverless AI application. This section provides instructions for configuring Driverless AI to work with Google Cloud Storage. This setup requires you to enable authentication. If you enable GCS or GBP connectors, those file systems will be available in the UI, but you will not be able to use those connectors without authentication.

In order to enable the GCS data connector with authentication, you must:

1. Obtain a JSON authentication file from [GCP](#).
2. Mount the JSON file to the Docker instance.
3. Specify the path to the `/json_auth_file.json` in the `gcs_path_to_service_account_json` config option.

Notes:

- The account JSON includes authentications as provided by the system administrator. You can be provided a JSON file that contains both Google Cloud Storage and Google BigQuery authentications, just one or the other, or none at all.
- Depending on your Docker install version, use either the `docker run --runtime=nvidia` (\geq Docker 19.03) or `nvidia-docker` (< Docker 19.03) command when starting the Driverless AI Docker image. Use `docker version` to check which version of Docker you are using.

15.6.1 Start GCS with Authentication

Docker Image Installs

Docker Image with the config.toml

Native Installs

This example enables the GCS data connector with authentication by passing the JSON authentication file. This assumes that the JSON file contains Google Cloud Storage authentications.

```
nvidia-docker run \
--pid=host \
--init \
--rm \
--shm-size=256m \
```

(continues on next page)

(continued from previous page)

```
-e DRIVERLESS_AI_ENABLED_FILE_SYSTEMS="file,gcs" \
-e DRIVERLESS_AI_GCS_PATH_TO_SERVICE_ACCOUNT_JSON="/service_account_json.json" \
-u $id -u : id -g \
-p 12345:12345 \
-v `pwd`:/data:/data \
-v `pwd`:/log:/log \
-v `pwd`:/license:/license \
-v `pwd`:/tmp:/tmp \
-v `/service_account_json.json:/service_account_json.json` \
harbor.h2o.ai/h2oai/dai-centos7-x86_64:1.9.0-cudal0.0.23
```

This example shows how to configure the GCS data connector options in the config.toml file, and then specify that file when starting Driverless AI in Docker.

1. Configure the Driverless AI config.toml file. Set the following configuration options:
 - enabled_file_systems = "file, upload, gcs"
 - gcs_path_to_service_account_json = "/service_account_json.json"
2. Mount the config.toml file into the Docker container.

```
nvidia-docker run \
--pid=host \
--init \
--rm \
--shm-size=256m \
--add-host name.node:172.16.2.186 \
-e DRIVERLESS_AI_CONFIG_FILE=/path/in/docker/config.toml \
-p 12345:12345 \
-v /local/path/to/config.toml:/path/in/docker/config.toml \
-v /etc/passwd:/etc/passwd:ro \
-v /etc/group:/etc/group:ro \
-v /tmp/dtmp:/tmp \
-v /tmp/dlog:/log \
-v /tmp/dlicense:/license \
-v /tmp/ddata:/data \
-u $id -u:$id -g \
harbor.h2o.ai/h2oai/dai-centos7-x86_64:1.9.0-cudal0.0.23
```

This example enables the GCS data connector with authentication by passing the JSON authentication file. This assumes that the JSON file contains Google Cloud Storage authentications.

1. Export the Driverless AI config.toml file or add it to `~/.bashrc`. For example:

```
# DEB and RPM
export DRIVERLESS_AI_CONFIG_FILE="/etc/dai/config.toml"

# TAR SH
export DRIVERLESS_AI_CONFIG_FILE="/path/to/your/unpacked/dai/directory/config.toml"
```

2. Specify the following configuration options in the config.toml file.

```
# File System Support
# upload : standard upload feature
# file : local file system/server file system
# hdfs : Hadoop file system, remember to configure the HDFS config folder path and keytab below
# dtap : Blue Data Tap file system, remember to configure the DTap section below
# s3 : Amazon S3, optionally configure secret and access key below
# gcs : Google Cloud Storage, remember to configure gcs_path_to_service_account_json below
# gbq : Google Big Query, remember to configure gcs_path_to_service_account_json below
# minio : Minio Cloud Storage, remember to configure secret and access key below
# snow : Snowflake Data Warehouse, remember to configure Snowflake credentials below (account name, username, password)
# kdb : KDB+ Time Series Database, remember to configure KDB credentials below (hostname and port, optionally: username, password, classpath, and jvm_args)
# azrbs : Azure Blob Storage, remember to configure Azure credentials below (account name, account key)
# jdbc: JDBC Connector, remember to configure JDBC below. (jdbc_app_configs)
# hive: Hive Connector, remember to configure Hive below. (hive_app_configs)
# recipe_url: load custom recipe from URL
# recipe_file: load custom recipe from local file system
enabled_file_systems = "file, gcs"

# GCS Connector credentials
# example (suggested) -- "/licenses/my_service_account_json.json"
gcs_path_to_service_account_json = "/service_account_json.json"
```

3. Save the changes when you are done, then stop/restart Driverless AI.

15.7 Hive Setup

Driverless AI allows you to explore Hive data sources from within the Driverless AI application. This section provides instructions for configuring Driverless AI to work with Hive.

Note: Depending on your Docker install version, use either the `docker run --runtime=nvidia (>= Docker 19.03)` or `nvidia-docker (< Docker 19.03)` command when starting the Driverless AI Docker image. Use `docker version` to check which version of Docker you are using.

15.7.1 Description of Configuration Attributes

- `enabled_file_systems`: The file systems you want to enable. This must be configured in order for data connectors to function properly.
- `hive_app_configs`: Configuration for Hive Connector. Inputs are similar to configuring the HDFS connector. Important keys include:
 - `hive_conf_path`: The path to Hive configuration. This can have multiple files (e.g. `hive-site.xml`, `hdfs-site.xml`, etc.)
 - `auth_type`: Specify one of `noauth`, `keytab`, or `keytabimpersonation` for Kerberos authentication
 - `keytab_path`: Specify the path to Kerberos keytab to use for authentication (this can be "" if using `auth_type='noauth'`)
 - `principal_user`: Specify the Kerberos app principal user (required when using `auth_type='keytab'` or `auth_type='keytabimpersonation'`)

The configuration should be JSON/Dictionary String with multiple keys. For example:

```
"""
  "hive_connection_1": {
    "hive_conf_path": "/path/to/hive/conf",
    "auth_type": "one of ['noauth', 'keytab',
    'keytabimpersonation']",
    "keytab_path": "/path/to/<filename>.keytab",
    "principal_user": "hive/LOCALHOST@H2O.AI",
  },
  "hive_connection_2": {
    "hive_conf_path": "/path/to/hive/conf_2",
    "auth_type": "one of ['noauth', 'keytab',
    'keytabimpersonation']",
    "keytab_path": "/path/to/<filename_2>.keytab",
    "principal_user": "my_user/LOCALHOST@H2O.AI",
  }
}"""
```

Note: The expected input of `hive_app_configs` is a [JSON string](#). Double quotation marks (" . . . ") must be used to denote keys and values *within* the JSON dictionary, and *outer* quotations must be formatted as either " " ", ' ' ', or ' . Depending on how the configuration value is applied, different forms of outer quotations may be required. The following examples show two unique methods for applying outer quotations.

- Configuration value applied with the config.toml file:

```
hive_app_configs = """{"my_json_string": "value", "json_key_2": "value2"}"""
```

- Configuration value applied with an environment variable:

```
DRIVERLESS_AI_HIVE_APP_CONFIGS="{"my_json_string": "value", "json_key_2": "value2"}"
```

- `hive_app_jvm_args`: In cases where JAAS is required, specify additional Java Virtual Machine (JVM) args for the Hive connector. Each arg must be separated by a space. The following is an example of how this config.toml option can be specified:

```
hive_app_jvm_args = "-Xmx20g -Djavax.security.auth.useSubjectCredsOnly=false -Djava.security.auth.login.config=/etc/dai/jaas.conf"
```

Notes:

- The `-Djavax.security.auth.useSubjectCredsOnly=false` default arg is required for Kerberos authentication and impersonation.
- The `-Djava.security.auth.login.config=/etc/dai/jaas.conf` default arg is required to allow the underlying connector process to adopt the Kerberos login properties defined in `/etc/dai/jaas.conf`. You must create the `jaas.conf` file and place it in the specified directory. The following is an example of how the `jaas.conf` file can be specified:

```
com.sun.security.jgss.initiate {  
    com.sun.security.auth.module.Krb5LoginModule required  
    useKeyTab=true  
    useTicketCache=false  
    principal="super-qa/mr-0xg9.0xdata.loc@H2OAI.LOC" [Replace this line]  
    doNotPrompt=true  
    keyTab="/etc/dai/super-qa.keytab" [Replace this line]  
    debug=true;  
};
```

15.7.2 Enable Hive with Authentication

This section describes how to enable Hive when starting Driverless AI in Docker. This can be done by specifying each environment variable in the `nvidia-docker run` command or by editing the configuration options in the `config.toml` file and then specifying that file in the `nvidia-docker run` command.

Docker Image Installs

Docker Image with the `config.toml`

Native Installs

1. Start the Driverless AI Docker Image.

```
nvidia-docker run \  
  --pid=host \  
  --init \  
  --rm \  
  --shm-size=256m \  
  --add-host name:node:172.16.2.186 \  
  -e DRIVERLESS_AI_ENABLED_FILE_SYSTEMS="file,hdfs,hive" \  
  -e DRIVERLESS_AI_HIVE_APP_CONFIGS='{"hive_1": {"auth_type": "keytab",  
    "keytab_path": "/path/in/docker/hive.keytab",  
    "hive_conf_path": "/path/to/hive/conf/in/docker",  
    "principal_user": "hive/localhost@H2O.AI"}}' \  
  -p 12345:12345 \  
  -v /etc/passwd:/etc/passwd:ro \  
  -v /etc/group:/etc/group:ro \  
  -v /tmp/dtmp/:/tmp \  
  -v /tmp/dlog/:/log \  
  -v /tmp/dlicense/:/license \  
  -v /tmp/ddata/:/data \  
  -v /path/to/hive/conf:/path/to/hive/conf/in/docker \  
  -v /path/to/hive.keytab:/path/in/docker/hive.keytab \  
  -u $id -u:$id -g) \  
h2oai/dai-centos7-x86_64:1.9.0-cudal0.0.23
```

This example shows how to configure Hive options in the `config.toml` file, and then specify that file when starting Driverless AI in Docker.

1. Enable and configure the Hive connector in the Driverless AI `config.toml` file. The Hive connector configuration must be a JSON/Dictionary string with multiple keys.

```
enabled_file_systems = "file, hdfs, s3, hive"  
hive_app_configs = """{"hive_1": {"auth_type": "keytab",  
    "key_tab_path": "/path/to/Downloads/hive.keytab",  
    "hive_conf_path": "/path/to/Downloads/hive-resources",  
    "principal_user": "hive/localhost@H2O.AI"}}"""
```

2. Mount the `config.toml` file into the Docker container.

```
nvidia-docker run \
--pid=host \
--init \
--rm \
--shm-size=256m \
--add-host name.node:172.16.2.186 \
-e DRIVERLESS_AI_CONFIG_FILE=/path/in/docker/config.toml \
-p 12345:12345 \
-v /local/path/to/config.toml:/path/in/docker/config.toml \
-v /etc/passwd:/etc/passwd:ro /
-v /tmp/dtmp:/tmp \
-v /tmp/dlog:/log \
-v /tmp/dlicense:/license \
-v /tmp/ddata:/data \
-v /path/to/hive/conf:/path/to/hive/conf/in/docker \
-v /path/to/hive.keytab:/path/in/docker/hive.keytab \
-u $@:$@ \
h2oai/dai-centos7-x86_64:1.9.0-cudal0.0.23
```

This enables the Hive connector.

1. Export the Driverless AI config.toml file or add it to `~/.bashrc`.

```
# DEB and RPM
export DRIVERLESS_AI_CONFIG_FILE="/etc/dai/config.toml"

# TAR SH
export DRIVERLESS_AI_CONFIG_FILE="/path/to/your/unpacked/dai/directory/config.toml"
```

2. Specify the following configuration options in the config.toml file.

```
# File System Support
# upload : standard upload feature
# file : local file system/server file system
# hdfs : Hadoop file system, remember to configure the HDFS config folder path and keytab below
# dtap : Blue Data Tap file system, remember to configure the DTap section below
# s3 : Amazon S3, optionally configure secret and access key below
# gcs : Google Cloud Storage, remember to configure gcs_path_to_service_account_json below
# gbq : Google Big Query, remember to configure gcs_path_to_service_account_json below
# minio : Minio Cloud Storage, remember to configure secret and access key below
# snow : Snowflake Data Warehouse, remember to configure Snowflake credentials below (account name, username, password)
# kdb : KDB+ Time Series Database, remember to configure KDB credentials below (hostname and port, optionally: username, password, classpath, and jvm_args)
# azrbs : Azure Blob Storage, remember to configure Azure credentials below (account name, account key)
# jdbc: JDBC Connector, remember to configure JDBC below. (jdbc_app_configs)
# hive: Hive Connector, remember to configure Hive below. (hive_app_configs)
# recipe_url: load custom recipe from URL
# recipe_file: load custom recipe from local file system
enabled_file_systems = "file, hdfs, s3, hive"

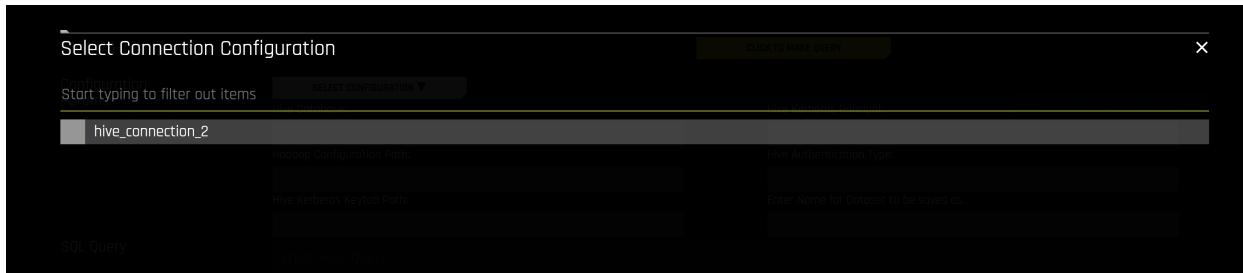
# Configuration for Hive Connector
# Note that inputs are similar to configuring HDFS connectivity
# Important keys:
# * hive_conf_path - path to hive configuration, may have multiple files. Typically: hive-site.xml, hdfs-site.xml, etc
# * auth_type - one of `noauth`, `keytab`, `keytabimpersonation` for kerberos authentication
# * keytab_path - path to the kerberos keytab to use for authentication, can be "" if using `noauth` auth_type
# * principal_user = Kerberos app principal user. Required when using auth_type `keytab` or `keytabimpersonation`
# JSON/Dictionary String with multiple keys. Example:
# """
# "hive_connection_1": {
#   "hive_conf_path": "/path/to/hive/conf",
#   "auth_type": "one of ['noauth', 'keytab', 'keytabimpersonation']",
#   "keytab_path": "/path/to/<filename>.keytab",
#   "principal_user": "hive/LOCALHOST@H2O.AI",
# }
# }"""
#
hive_app_configs = """("hive_1": {"auth_type": "keytab",
"key_tab_path": "/path/to/Downloads/hive.keytab",
"hive_conf_path": "/path/to/Downloads/hive-resources",
"principal_user": "hive/localhost@H2O.AI"})"""
```

3. Save the changes when you are done, then stop/restart Driverless AI.

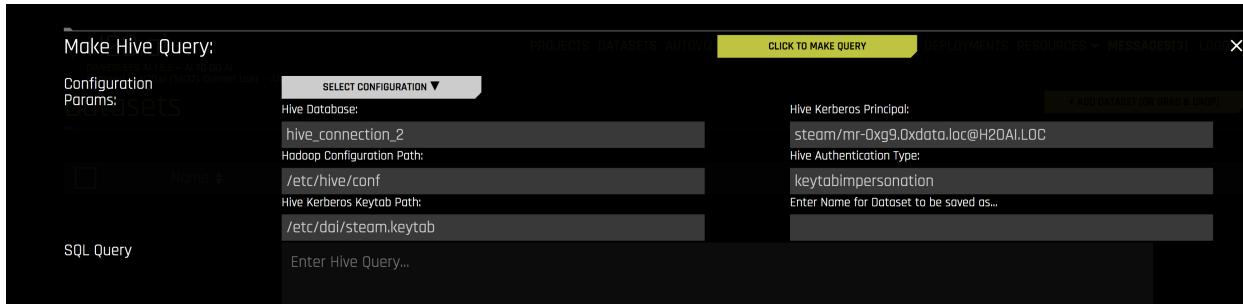
15.7.3 Adding Datasets Using Hive

After the Hive connector is enabled, you can add datasets by selecting **Hive** from the **Add Dataset (or Drag and Drop)** drop-down menu.

1. Select the Hive configuraton that you want to use.



2. Specify the following information to add your dataset.
 - **Hive Database:** Specify the name of the Hive database that you are querying.
 - **Hadoop Configuration Path:** Specify the path to your Hive configuration file.
 - **Hive Kerberos Keytab Path:** Specify the path for the Hive Kerberos keytab.
 - **Hive Kerberos Principal:** Specify the Hive Kerberos principal. This is required if the Hive Authentication Type is keytabimpersonation.
 - **Hive Authentication Type:** Specify the authentication type. This can be noauth, keytab, or keytabimpersonation.
 - **Enter Name for Dataset to be saved as:** Optionally specify a new name for the dataset that you are uploading.
 - **SQL Query:** Specify the Hive query that you want to execute.



15.8 JDBC Setup

Driverless AI allows you to explore Java Database Connectivity (JDBC) data sources from within the Driverless AI application. This section provides instructions for configuring Driverless AI to work with JDBC.

Note: Depending on your Docker install version, use either the `docker run --runtime=nvidia (>= Docker 19.03)` or `nvidia-docker (< Docker 19.03)` command when starting the Driverless AI Docker image. Use `docker version` to check which version of Docker you are using.

15.8.1 Tested Databases

The following databases have been tested for minimal functionality. Note that JDBC drivers that are not included in this list should work with Driverless AI. We recommend that you test out your JDBC driver even if you do not see it on list of tested databases. See the [Adding an Untested JDBC Driver](#) section at the end of this chapter for information on how to try out an untested JDBC driver.

- Oracle DB
- PostgreSQL
- Amazon Redshift
- Teradata

15.8.2 Description of Configuration Attributes

- `jdbc_app_configs`: Configuration for the JDBC connector. This is a JSON/Dictionary String with multiple keys. **Note:** This requires a JSON key (typically the name of the database being configured) to be associated with a nested JSON that contains the `url`, `jarpath`, and `classpath` fields. In addition, this should take the format:

```
"""{"my_jdbc_database": {"url": "jdbc:my_jdbc_database://hostname:port/database",
"jarpath": "/path/to/my/jdbc/database.jar", "classpath": "com.my.jdbc.Driver"}}""
```

For example:

```
"""
{
"postgres": {
"url": "jdbc:postgresql://ip address:port/postgres",
"jarpath": "/path/to/postgres_driver.jar",
"classpath": "org.postgresql.Driver"
},
"mysql": {
"url": "mysql connection string",
"jarpath": "/path/to/mysql_driver.jar",
"classpath": "my.sql.classpath.Driver"
}
}""
```

Note: The expected input of `jdbc_app_configs` is a JSON string. Double quotation marks (" . . . ") must be used to denote keys and values *within* the JSON dictionary, and *outer* quotations must be formatted as either " " ", ' ' ', or ' . Depending on how the configuration value is applied, different forms of outer quotations may be required. The following examples show two unique methods for applying outer quotations.

- Configuration value applied with the config.toml file:

```
jdbc_app_configs = """{"my_json_string": "value", "json_key_2": "value2"}"""
```

- Configuration value applied with an environment variable:

```
DRIVERLESS_AI_JDBC_APP_CONFIGS='{"my_json_string": "value", "json_key_2": "value2"}'
```

- `jdbc_app_jvm_args`: Extra jvm args for JDBC connector. For example, “`-Xmx4g`”.
- `jdbc_app_classpath`: Optionally specify an alternative classpath for the JDBC connector.
- `enabled_file_systems`: The file systems you want to enable. This must be configured in order for data connectors to function properly.

15.8.3 Retrieve the JDBC Driver

1. Download JDBC Driver JAR files:

- Oracle DB
- PostgreSQL
- Amazon Redshift
- Teradata

Note: Remember to take note of the driver classpath, as it is needed for the configuration steps (for example, org.postgresql.Driver).

2. Copy the driver JAR to a location that can be mounted into the Docker container.

Note: The folder storing the JDBC jar file must be visible/readable by the dai process user.

15.8.4 Enable the JDBC Connector

Docker Image Installs

Docker Image with the config.toml

Native Installs

This example enables the JDBC connector for PostgreSQL. Note that the JDBC connection strings will vary depending on the database that is used.

```
nvidia-docker run \
--pid=host \
--init \
--rm \
--shm-size=256m \
--add-host name.node:172.16.2.186 \
-e DRIVERLESS_AI_ENABLED_FILE_SYSTEMS="file,hdfs,jdbc" \
-e DRIVERLESS_AI_JDBC_APP_CONFIGS='["postgres": {
    "url": "jdbc:postgresql://localhost:5432/my_database",
    "jarpath": "/path/to/postgresql/jdbc/driver.jar",
    "classpath": "org.postgresql.Driver"}]' \
-e DRIVERLESS_AI_JDBC_APP_JVM_ARGS="-Xmx2g" \
-p 12345:12345 \
-v /path/to/local/postgresql/jdbc/driver.jar:/path/to/postgresql/jdbc/driver.jar \
-v /etc/passwd:/etc/passwd:ro \
-v /etc/group:/etc/group:ro \
-v /tmp/dtmp/:/tmp \
-v /tmp/dlog/:/log \
-v /tmp/dlicense/:/license \
-v /tmp/ddata/:/data \
-u $(id -u):$(id -g) \
harbor.h2o.ai/h2oai/dai-centos7-x86_64:1.9.0-cuda10.0.23
```

This example shows how to configure JDBC options in the config.toml file, and then specify that file when starting Driverless AI in Docker.

1. Configure the Driverless AI config.toml file. Set the following configuration options:

```
enabled_file_systems = "file, upload, jdbc"
jdbc_app_configs = """["postgres": { "url": "jdbc:postgresql://localhost:5432/my_database",
    "jarpath": "/path/to/postgresql/jdbc/driver.jar",
    "classpath": "org.postgresql.Driver"}]"""
```

2. Mount the config.toml file and requisite JAR files into the Docker container.

```
nvidia-docker run \
--pid=host \
--init \
--rm \
--shm-size=256m \
--add-host name.node:172.16.2.186 \
-e DRIVERLESS_AI_CONFIG_FILE=/path/in/docker/config.toml \
-p 12345:12345 \
-v /local/path/to/jdbc/driver.jar:/path/in/docker/jdbc/driver.jar \
-v /local/path/to/config.toml:/path/in/docker/config.toml \
-v /etc/passwd:/etc/passwd:ro \
-v /etc/group:/etc/group:ro \
-v /tmp/dtmp/:/tmp \

```

(continues on next page)

(continued from previous page)

```
-v /tmp/dlog/:/log \
-v /tmp/dlicense/:/license \
-v /tmp/ddata/:/data \
-u $(id -u):$(id -g) \
harbor.h2o.ai/h2oai/dai-centos7-x86_64:1.9.0-cuda10.0.23
```

This example enables the JDBC connector for PostgresQL.

Notes:

- The JDBC connection strings will vary depending on the database that is used.
- The configuration requires a JSON key (typically the name of the database being configured) to be associated with a nested JSON that contains the url, jarpath, and classpath fields. In addition, this should take the format:

```
"""{"my_jdbc_database": {"url": "jdbc:my_jdbc_database://hostname:port/database",
"jarpath": "/path/to/my/jdbc/database.jar", "classpath": "com.my.jdbc.Driver"}}"""
```

- Export the Driverless AI config.toml file or add it to `~/.bashrc`. For example:

```
# DEB and RPM
export DRIVERLESS_AI_CONFIG_FILE="/etc/dai/config.toml"

# TAR SH
export DRIVERLESS_AI_CONFIG_FILE="/path/to/your/unpacked/dai/directory/config.toml"
```

- Edit the following values in the config.toml file.

```
# File System Support
# upload : standard upload feature
# file : local file system/server file system
# hdfs : Hadoop file system, remember to configure the HDFS config folder path and keytab below
# dtap : Blue Data Tap file system, remember to configure the DTap section below
# s3 : Amazon S3, optionally configure secret and access key below
# gcs : Google Cloud Storage, remember to configure gcs_path_to_service_account_json below
# gbg : Google Big Query, remember to configure gcs_path_to_service_account_json below
# minio : Minio Cloud Storage, remember to configure secret and access key below
# snow : Snowflake Data Warehouse, remember to configure Snowflake credentials below (account name, username, password)
# kdb : KDB+ Time Series Database, remember to configure KDB credentials below (hostname and port, optionally: username, password, ↵
#       classpath, and jvm_args)
# azrbs : Azure Blob Storage, remember to configure Azure credentials below (account name, account key)
# jdbc: JDBC Connector, remember to configure JDBC below. (jdbc_app_configs)
# hive: Hive Connector, remember to configure Hive below. (hive_app_configs)
# recipe_url: load custom recipe from URL
# recipe_file: load custom recipe from local file system
enabled_file_systems = "upload, file, hdfs, jdbc"

# Configuration for JDBC Connector.
# JSON/Dictionary String with multiple keys.
# Format as a single line without using carriage returns (the following example is formatted for readability).
# Use triple quotations to ensure that the text is read as a single string.
# Example:
# """
#   """
#     "postgres": {
#       "url": "jdbc:postgresql://ip address:port/postgres",
#       "jarpath": "/path/to/postgres_driver.jar",
#       "classpath": "org.postgresql.Driver"
#     },
#     "mysql": {
#       "url": "mysql connection string",
#       "jarpath": "/path/to/mysql_driver.jar",
#       "classpath": "my.sql.classpath.Driver"
#     }
#   """
# }

jdbc_app_configs = """{"postgres": {"url": "jdbc:postgress://localhost:5432/my_database",
"jarpath": "/path/to/postgresql/jdbc/driver.jar",
"classpath": "org.postgresql.Driver"}}

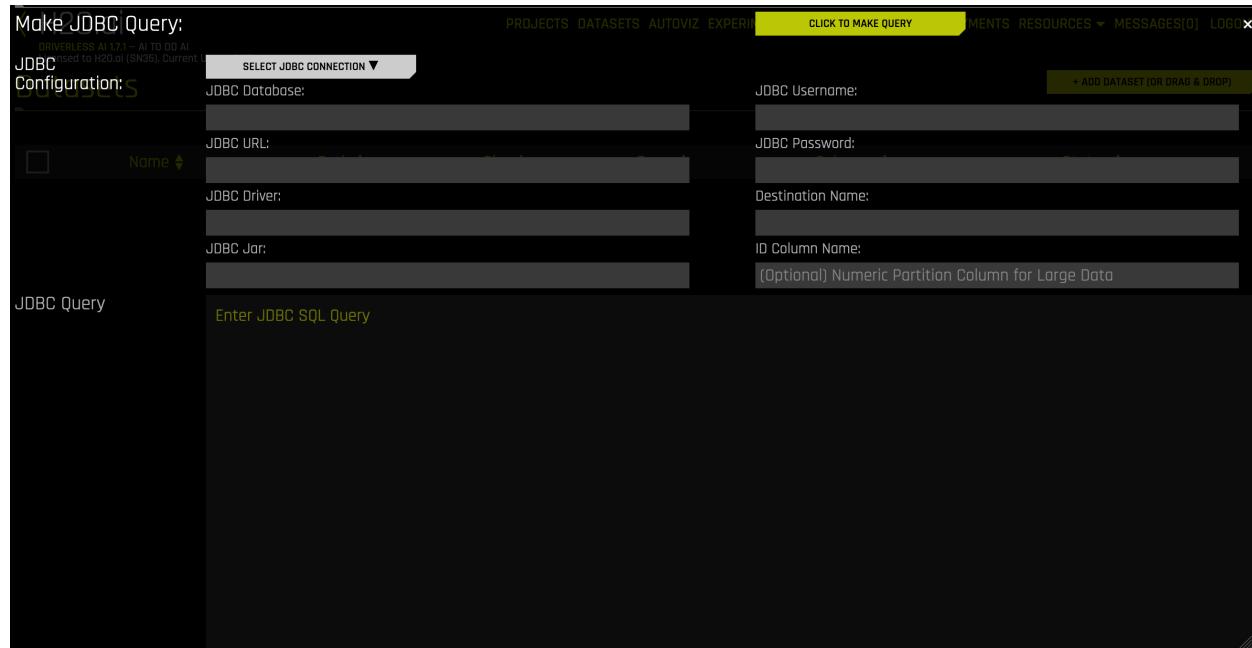
# extra jvm args for jdbc connector
jdbc_app_jvm_args = ""

# alternative classpath for jdbc connector
jdbc_app_classpath = """
```

- Save the changes when you are done, then stop/restart Driverless AI.

15.8.5 Adding Datasets Using JDBC

After the JDBC connector is enabled, you can add datasets by selecting **JDBC** from the **Add Dataset (or Drag and Drop)** drop-down menu.



1. Click on the **Add Dataset** button on the Datasets page.
2. Select **JDBC** from the list that appears.
3. Click on the **Select JDBC Connection** button to select a JDBC configuration.
4. The form will populate with the JDBC Database, URL, Driver, and Jar information. Complete the following remaining fields:
 - **JDBC Username:** Enter your JDBC username.
 - **JDBC Password:** Enter your JDBC password.
 - **Destination Name:** Enter a name for the new dataset.
 - (Optional) **ID Column Name:** Enter a name for the ID column. Specify this field when making large data queries.

Notes:

- Due to resource sharing within Driverless AI, the JDBC Connector is only allocated a relatively small amount of memory.
 - When making large queries, the ID column is used to partition the data into manageable portions. This ensures that the maximum memory allocation is not exceeded.
 - If a query that is larger than the maximum memory allocation is made without specifying an ID column, the query will not complete successfully.
5. Write a SQL Query in the format of the database that you want to query. (See the *Query Examples* section below.) The format will vary depending on the database that is used.

- Click the **Click to Make Query** button to execute the query. The time it takes to complete depends on the size of the data being queried and the network speeds to the database.

On a successful query, you will be returned to the datasets page, and the queried data will be available as a new dataset.

15.8.6 Query Examples

The following are sample configurations and queries for Oracle DB and PostgreSQL:

Oracle DB

PostgreSQL

- Configuration:

```
jdbc_app_configs = """("oracledb": {"url": "jdbc:oracle:thin:@localhost:1521/oracledatabase", "jarpath": "/home/ubuntu/jdbc-jars/ojdbc8.jar", "classpath": "oracle.jdbc.OracleDriver"})"""
```

- Sample Query:

- Select **oracledb** from the **Select JDBC Connection** dropdown menu.
- JDBC Username:** oracleuser
- JDBC Password:** oracleuserpassword
- ID Column Name:**
- Query:**

```
SELECT MIN(ID) AS NEW_ID, EDUCATION, COUNT(EDUCATION) FROM my_oracle_schema.creditcardtrain GROUP BY EDUCATION
```

Note: Because this query does not specify an **ID Column Name**, it will only work for small data. However, the **NEW_ID** column can be used as the ID Column if the query is for larger data.

- Click the **Click to Make Query** button to execute the query.

- Configuration:

```
jdbc_app_configs = """("postgres": {"url": "jdbc:postgresql://localhost:5432/postgresdatabase", "jarpath": "/home/ubuntu/postgres-artifacts/postgres/Driver.jar", "classpath": "org.postgresql.Driver"})"""
```

- Sample Query:

- Select **postgres** from the **Select JDBC Connection** dropdown menu.
- JDBC Username:** postgres_user
- JDBC Password:** pguserpassword
- ID Column Name:** id
- Query:**

```
SELECT * FROM loan_level WHERE LOAN_TYPE = 5 (selects all columns from table loan_level with column LOAN_TYPE containing value 5)
```

- Click the **Click to Make Query** button to execute the query.

15.8.7 Adding an Untested JDBC Driver

We encourage you to try out JDBC drivers that are not tested in house.

Docker Image Installs

Docker Image with the config.toml

Native Installs

1. Download the JDBC jar for your database.
2. Move your JDBC jar file to a location that DAI can access.
3. Start the Driverless AI Docker image using the JDBC-specific environment variables.

```
nvidia-docker run \
--pid=host \
--init \
--rm \
--shm-size=256m \
--add-host name.node:172.16.2.186 \
-e DRIVERLESS_AI_ENABLED_FILE_SYSTEMS="upload,file,hdfs,s3,recipe_file,jdbc" \
-e DRIVERLESS_AI_JDBC_APP_CONFIGS="" \
("my_jdbc_database": {"url": "jdbc:my_jdbc_database://hostname:port/database",
"jarpath": "/path/to/my/jdbc/database.jar",
"classpath": "com.my.jdbc.Driver"})"""\"
-e DRIVERLESS_AI_JDBC_APP_JVM_ARGS="-Xmx2g" \
-p 12345:12345 \
-v /path/to/local/postgresql/jdbc/driver.jar:/path/to/postgresql/jdbc/driver.jar \
-v /etc/passwd:/etc/passwd:ro \
-v /etc/group:/etc/group:ro \
-v /tmp/dtmp/:/tmp \
-v /tmp/dlog/:/log \
-v /tmp/dlicense/:/license \
-v /tmp/ddata/:/data \
-u ${id -u}:$(id -g) \
harbor.h2o.ai/h2oai/dai-centos7-x86_64:1.9.0-cudal0.0.23
```

1. Download the JDBC jar for your database.
2. Move your JDBC jar file to a location that DAI can access.
3. Configure the Driverless AI config.toml file. Set the following configuration options:

```
enabled_file_systems = "upload, file, hdfs, s3, recipe_file, jdbc"
jdbc_app_configs = """ \
("my_jdbc_database": {"url": "jdbc:my_jdbc_database://hostname:port/database",
"jarpath": "/path/to/my/jdbc/database.jar",
"classpath": "com.my.jdbc.Driver"})"""
#Optional arguments
jdbc_app_jvm_args = ""
jdbc_app_classpath = ""
```

4. Mount the config.toml file and requisite JAR files into the Docker container.

```
nvidia-docker run \
--pid=host \
--init \
--rm \
--shm-size=256m \
--add-host name.node:172.16.2.186 \
-e DRIVERLESS_AI_CONFIG_FILE=/path/in/docker/config.toml \
-p 12345:12345 \
-v /local/path/to/jdbc/driver.jar:/path/in/docker/jdbc/driver.jar \
-v /local/path/to/config.toml:/path/in/docker/config.toml \
-v /etc/passwd:/etc/passwd:ro \
-v /etc/group:/etc/group:ro \
-v /tmp/dtmp/:/tmp \
-v /tmp/dlog/:/log \
-v /tmp/dlicense/:/license \
-v /tmp/ddata/:/data \
-u ${id -u}:$(id -g) \
harbor.h2o.ai/h2oai/dai-centos7-x86_64:1.9.0-cudal0.0.23
```

1. Download the JDBC jar for your database.
2. Move your JDBC jar file to a location that DAI can access.
3. Modify the following config.toml settings. Note that these can also be specified as environment variables when starting Driverless AI in Docker:

```
# enable the JDBC file system
enabled_file_systems = "upload, file, hdfs, s3, recipe_file, jdbc"
# Configure the JDBC Connector.
```

(continues on next page)

(continued from previous page)

```

# JSON/Dictionary String with multiple keys.
# Format as a single line without using carriage returns (the following example is formatted for readability).
# Use triple quotations to ensure that the text is read as a single string.
# Example:
jdbc_app_configs = """{"my_jdbc_database": {"url": "jdbc:my_jdbc_database://hostname:port/database",
    "jarpath": "/path/to/my/jdbc/database.jar",
    "classpath": "com.my.jdbc.Driver"}}"""
# optional extra jvm args for jdbc connector
jdbc_app_jvm_args = ""

# optional alternative classpath for jdbc connector
jdbc_app_classpath = ""

```

- Save the changes when you are done, then stop/restart Driverless AI.

15.9 kdb+ Setup

Driverless AI allows you to explore [kdb+](#) data sources from within the Driverless AI application. This section provides instructions for configuring Driverless AI to work with kdb+.

Note: Depending on your Docker install version, use either the `docker run --runtime=nvidia` (\geq Docker 19.03) or `nvidia-docker` (< Docker 19.03) command when starting the Driverless AI Docker image. Use `docker version` to check which version of Docker you are using.

15.9.1 Description of Configuration Attributes

- `kdb_user`: (Optional) User name
- `kdb_password`: (Optional) User's password
- `kdb_hostname`: IP address or host of the KDB server
- `kdb_port`: Port on which the kdb+ server is listening
- `kdb_app_jvm_args`: (Optional) JVM args for kdb+ distributions (for example, `-Dlog4j.configuration`). Separate each argument with spaces.
- `kdb_app_classpath`: (Optional) The kdb+ classpath (or other if the jar file is stored elsewhere).
- `enabled_file_systems`: The file systems you want to enable. This must be configured in order for data connectors to function properly.

15.9.2 Example 1: Enable kdb+ with No Authentication

Docker Image Installs

Docker Image with the config.toml

Native Installs

This example enables the kdb+ connector without authentication. The only required flags are the hostname and the port.

```

nvidia-docker run \
--pid=host \
--init \
--rm \
--shm-size=256m \
--add-host name.node:172.16.2.186 \
-e DRIVERLESS_AI_ENABLED_FILE_SYSTEMS="file,kdb" \
-e DRIVERLESS_AI_KDB_HOSTNAME=<ip_or_host_of_kdb_server> \
-e DRIVERLESS_AI_KDB_PORT=<kdb_server_port> \
-p 12345:12345 \
-v /tmp/dtmp:/tmp \
-v /tmp/dlog:/log \

```

(continues on next page)

(continued from previous page)

```
-v /tmp/dlicense:/license \
-v /tmp/ddata:/data \
-u $(id -u):$(id -g) \
harbor.h2o.ai/h2oai/dai-centos7-x86_64:1.9.0-cuda10.0.23
```

This example shows how to configure kdb+ options in the config.toml file, and then specify that file when starting Driverless AI in Docker. Note that this example enables kdb+ with no authentication.

1. Configure the Driverless AI config.toml file. Set the following configuration options.
 - enabled_file_systems = "file, upload, kdb"
 - kdb_hostname = <ip_or_host_of_kdb_server>"
 - kdb_port = "<kdb_server_port>"
2. Mount the config.toml file into the Docker container.

```
nvidia-docker run \
--pid=host \
--init \
--rm \
--shm-size=256m \
--add-host name.node:172.16.2.186 \
-e DRIVERLESS_AI_CONFIG_FILE=/path/in/docker/config.toml \
-p 12345:12345 \
-v /local/path/to/config.toml:/path/in/docker/config.toml \
-v /etc/passwd:/etc/passwd:ro \
-v /etc/group:/etc/group:ro \
-v /tmp/dtmp/:/tmp \
-v /tmp/dlog/:/log \
-v /tmp/dlicense:/license \
-v /tmp/ddata:/data \
-u $(id -u):$(id -g) \
harbor.h2o.ai/h2oai/dai-centos7-x86_64:1.9.0-cuda10.0.23
```

This example enables the kdb+ connector without authentication. The only required flags are the hostname and the port.

1. Export the Driverless AI config.toml file or add it to ~/.bashrc. For example:

```
# DEB and RPM
export DRIVERLESS_AI_CONFIG_FILE="/etc/dai/config.toml"

# TAR SH
export DRIVERLESS_AI_CONFIG_FILE="/path/to/your/unpacked/dai/directory/config.toml"
```

2. Specify the following configuration options in the config.toml file.

```
# File System Support
# upload : standard upload feature
# file : local file system/server file system
# hdfs : Hadoop file system, remember to configure the HDFS config folder path and keytab below
# dtap : Blue Data Tap file system, remember to configure the DTap section below
# s3 : Amazon S3, optionally configure secret and access key below
# gcs : Google Cloud Storage, remember to configure gcs_path_to_service_account_json below
# gbq : Google Big Query, remember to configure gcs_path_to_service_account_json below
# minio : Minio Cloud Storage, remember to configure secret and access key below
# snow : Snowflake Data Warehouse, remember to configure Snowflake credentials below (account name, username, password)
# kdb : KDB+ Time Series Database, remember to configure KDB credentials below (hostname and port, optionally: username, password, classpath, and jvm_args)
# azblob : Azure Blob Storage, remember to configure Azure credentials below (account name, account key)
# jdbc : JDBC Connector, remember to configure JDBC below. (jdbc_app_configs)
# hive : Hive Connector, remember to configure Hive below. (hive_app_configs)
# recipe_url: load custom recipe from URL
# recipe_file: load custom recipe from local file system
enabled_file_systems = "file, kdb"

# KDB Connector credentials
kdb_hostname = <ip_or_host_of_kdb_server>
kdb_port = "<kdb_server_port>"
```

3. Save the changes when you are done, then stop/restart Driverless AI.

15.9.3 Example 2: Enable kdb+ with Authentication

Docker Image Installs

Docker Image with the config.toml

Native Installs

This example provides users credentials for accessing a kdb+ server from Driverless AI.

```
nvidia-docker run \
--pid=host \
--init \
--rm \
--shm-size=256m \
-e DRIVERLESS_AI_ENABLED_FILE_SYSTEMS="file,kdb" \
-e DRIVERLESS_AI_KDB_HOSTNAME=<ip_or_host_of_kdb_server> \
-e DRIVERLESS_AI_KDB_PORT=<kdb_server_port> \
-e DRIVERLESS_AI_KDB_USER=<username> \
-e DRIVERLESS_AI_KDB_PASSWORD=<password> \
-p 12345:12345 \
-v /tmp/dtmp:/tmp \
-v /tmp/dlog:/log \
-v /tmp/dlicense:/license \
-v /tmp/ddata:/data \
-u $(id -u):$(id -g) \
harbor.h2o.ai/h2oai/dai-centos7-x86_64:1.9.0-cudal0.0.23
```

This example shows how to configure kdb+ options in the config.toml file, and then specify that file when starting Driverless AI in Docker. Note that this example enables kdb+ with no authentication.

1. Configure the Driverless AI config.toml file. Set the following configuration options.

- enabled_file_systems = "file, upload, kdb"
- kdb_user = "<username>"
- kdb_password = "<password>"
- kdb_hostname = <ip_or_host_of_kdb_server>"
- kdb_port = "<kdb_server_port>"
- kdb_app_classpath = ""
- kdb_app_jvm_args = ""

2. Mount the config.toml file into the Docker container.

```
nvidia-docker run \
--pid=host \
--init \
--rm \
--shm-size=256m \
--add-host name.node:172.16.2.186 \
-e DRIVERLESS_AI_CONFIG_FILE=/path/in/docker/config.toml \
-p 12345:12345 \
-v /local/path/to/config.toml:/path/in/docker/config.toml \
-v /etc/passwd:/etc/passwd:ro \
-v /etc/group:/etc/group:ro \
-v /tmp/dtmp:/tmp \
-v /tmp/dlog:/log \
-v /tmp/dlicense:/license \
-v /tmp/ddata:/data \
-u $(id -u):$(id -g) \
harbor.h2o.ai/h2oai/dai-centos7-x86_64:1.9.0-cudal0.0.23
```

This example provides users credentials for accessing a kdb+ server from Driverless AI.

1. Export the Driverless AI config.toml file or add it to ~/.bashrc. For example:

```
# DEB and RPM
export DRIVERLESS_AI_CONFIG_FILE="/etc/dai/config.toml"

# TAR SH
export DRIVERLESS_AI_CONFIG_FILE="/path/to/your/unpacked/dai/directory/config.toml"
```

2. Specify the following configuration options in the config.toml file.

```
# File System Support
# upload : standard upload feature
# file : local file system/server file system
# hdfs : Hadoop file system, remember to configure the HDFS config folder path and keytab below
# dtap : Blue Data Tap file system, remember to configure the DTap section below
# s3 : Amazon S3, optionally configure secret and access key below
# gcs : Google Cloud Storage, remember to configure gcs_path_to_service_account_json below
# gbs : Google Big Query, remember to configure gcs_path_to_service_account_json below
# minio : Minio Cloud Storage, remember to configure secret and access key below
# snow : Snowflake Data Warehouse, remember to configure Snowflake credentials below (account name, username, password)
# kdb : KDB+ Time Series Database, remember to configure KDB credentials below (hostname and port, optionally: username, password, classpath, and jvm_args)
# azrbs : Azure Blob Storage, remember to configure Azure credentials below (account name, account key)
# jdbc: JDBC Connector, remember to configure JDBC below. (jdbc_app_configs)
# hive: Hive Connector, remember to configure Hive below. (hive_app_configs)
# recipe_url: load custom recipe from URL
# recipe_file: load custom recipe from local file system
enabled_file_systems = "file, kdb"

# kdb+ Connector credentials
kdb_user = "<username>"
kdb_password = "<password>"
kdb_hostname = "<ip_or_host_of_kdb_server>"
kdb_port = "<kdb_server_port>"
kdb_app_classpath = ""
kdb_app_jvm_args = ""
```

3. Save the changes when you are done, then stop/restart Driverless AI.

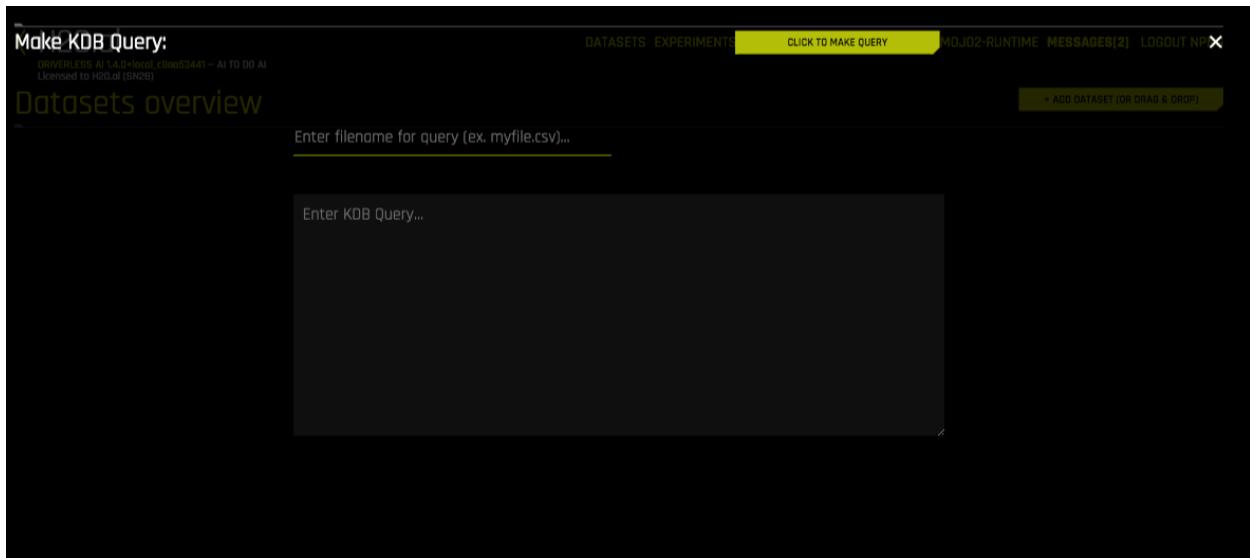
15.9.4 Adding Datasets Using kdb+

After the kdb+ connector is enabled, you can add datasets by selecting **kdb+** from the **Add Dataset (or Drag and Drop)** drop-down menu.



Specify the following information to add your dataset.

1. **Enter filepath to save query.** Enter the local file path for storing your dataset. For example, `/home/<user>/myfile.csv`. Note that this can only be a CSV file.
2. **Enter KDB Query:** Enter a kdb+ query that you want to execute. Note that the connector will accept any `q` ueries. For example: `select from <mytable> or <mytable> lj <myothertable>`
3. When you are finished, select the **Click to Make Query** button to add the dataset.



15.10 Minio Setup

This section provides instructions for configuring Driverless AI to work with [Minio](#). Note that unlike S3, authentication must also be configured when the Minio data connector is specified.

Note: Depending on your Docker install version, use either the `docker run --runtime=nvidia (>= Docker 19.03)` or `nvidia-docker (< Docker 19.03)` command when starting the Driverless AI Docker image. Use `docker version` to check which version of Docker you are using.

15.10.1 Description of Configuration Attributes

- `minio_endpoint_url`: The endpoint URL that will be used to access Minio.
- `minio_access_key_id`: The Minio access key
- `minio_secret_access_key`: The Minio secret access key
- `enabled_file_systems`: The file systems you want to enable. This must be configured in order for data connectors to function properly.

15.10.2 Enable Minio with Authentication

Docker Image Installs

Docker Image with the config.toml

Native Installs

This example enables the Minio data connector with authentication by passing an endpoint URL, access key ID, and an access key. It also configures Docker DNS by passing the name and IP of the name node. This allows users to reference data stored in Minio directly using the endpoint URL, for example: http://<endpoint_url>/<bucket>/datasets/iris.csv.

```
nvidia-docker run \
--shm-size=256m \
--add-host name.node:172.16.2.186 \
-e DRIVERLESS_AI_ENABLED_FILE_SYSTEMS="file,minio" \
-e DRIVERLESS_AI_MINIO_ENDPOINT_URL="" \
-e DRIVERLESS_AI_MINIO_ACCESS_KEY_ID="" \
-e DRIVERLESS_AI_MINIO_SECRET_ACCESS_KEY="" \
-p 12345:12345 \
--init -i --rm \
-v /tmp/dtmp:/tmp \
-v /tmp/dlog:/log \
-v /tmp/dlicense:/license \
-v /tmp/ddata:/data \
-u $(id -u):$(id -g) \
harbor.h2o.ai/h2oai/dai-centos7-x86_64:1.9.0-cudal0.0.23
```

This example shows how to configure Minio options in the config.toml file, and then specify that file when starting Driverless AI in Docker.

1. Configure the Driverless AI config.toml file. Set the following configuration options.

- enabled_file_systems = "file, upload, minio"
- minio_endpoint_url = "<endpoint_url>"
- minio_access_key_id = "<access_key_id>"
- minio_secret_access_key = "<access_key>"

2. Mount the config.toml file into the Docker container.

```
nvidia-docker run \
--pid=host \
--init \
--rm \
--shm-size=256m \
--add-host name.node:172.16.2.186 \
-e DRIVERLESS_AI_CONFIG_FILE=/path/in/docker/config.toml \
-p 12345:12345 \
-v /local/path/to/config.toml:/path/in/docker/config.toml \
-v /etc/passwd:/etc/passwd:ro \
-v /etc/group:/etc/group:ro \
-v /tmp/dtmp:/tmp \
-v /tmp/dlog:/log \
-v /tmp/dlicense:/license \
-v /tmp/ddata:/data \
-u $(id -u):$(id -g) \
harbor.h2o.ai/h2oai/dai-centos7-x86_64:1.9.0-cudal0.0.23
```

This example enables the Minio data connector with authentication by passing an endpoint URL, access key ID, and an access key. It also configures Docker DNS by passing the name and IP of the Minio endpoint. This allows users to reference data stored in Minio directly using the endpoint URL, for example: http://<endpoint_url>/<bucket>/datasets/iris.csv.

1. Export the Driverless AI config.toml file or add it to `~/.bashrc`. For example:

```
# DEB and RPM
export DRIVERLESS_AI_CONFIG_FILE="/etc/dai/config.toml"

# TAR SH
export DRIVERLESS_AI_CONFIG_FILE="/path/to/your/unpacked/dai/directory/config.toml"
```

2. Specify the following configuration options in the config.toml file.

```
# File System Support
# upload : standard upload feature
# file : local file system/server file system
# hdfs : Hadoop file system, remember to configure the HDFS config folder path and keytab below
# dtap : Blue Data Tap file system, remember to configure the DTap section below
# s3 : Amazon S3, optionally configure secret and access key below
# gcs : Google Cloud Storage, remember to configure gcs_path_to_service_account_json below
# gbig : Google Big Query, remember to configure gcs_path_to_service_account_json below
# minio : Minio Cloud Storage, remember to configure secret and access key below
# snow : Snowflake Data Warehouse, remember to configure Snowflake credentials below (account name, username, password)
# kdb : KDB+ Time Series Database, remember to configure KDB credentials below (hostname and port, optionally: username, password, ↳classpath, and jvm_args)
# azrbs : Azure Blob Storage, remember to configure Azure credentials below (account name, account key)
# jdbc : JDBC Connector, remember to configure JDBC below. (jdbc_app_configs)
# hive : Hive Connector, remember to configure Hive below. (hive_app_configs)
# recipe_url: load custom recipe from URL
# recipe_file: load custom recipe from local file system
enabled_file_systems = "file, minio"

# Minio Connector credentials
minio_endpoint_url = "<endpoint_url>"
minio_access_key_id = "<access_key_id>"
minio_secret_access_key = "<access_key>"
```

3. Save the changes when you are done, then stop/restart Driverless AI.

15.11 Snowflake Setup

Driverless AI allows you to explore Snowflake data sources from within the Driverless AI application. This section provides instructions for configuring Driverless AI to work with Snowflake. This setup requires you to enable authentication. If you enable Snowflake connectors, those file systems will be available in the UI, but you will not be able to use those connectors without authentication.

Note: Depending on your Docker install version, use either the `docker run --runtime=nvidia (>= Docker 19.03)` or `nvidia-docker (< Docker 19.03)` command when starting the Driverless AI Docker image. Use `docker version` to check which version of Docker you are using.

15.11.1 Description of Configuration Attributes

- `snowflake_account`: The Snowflake account ID
- `snowflake_user`: The username for accessing the Snowflake account
- `snowflake_password`: The password for accessing the Snowflake account
- `enabled_file_systems`: The file systems you want to enable. This must be configured in order for data connectors to function properly.

15.11.2 Enable Snowflake with Authentication

Docker Image Installs

Docker Image with the config.toml

Native Installs

This example enables the Snowflake data connector with authentication by passing the `account`, `user`, and `password` variables.

```
nvidia-docker run \
--rm \
--shm-size=256m \
-e DRIVERLESS_AI_ENABLED_FILE_SYSTEMS="file,snow" \
-e DRIVERLESS_AI_SNOWFLAKE_ACCOUNT = "<account_id>" \
-e DRIVERLESS_AI_SNOWFLAKE_USER = "<username>" \
-e DRIVERLESS_AI_SNOWFLAKE_PASSWORD = "<password>" \
-u `id -u : id -g` \
-p 12345:12345 \
-v `pwd`:/data:/data \
-v `pwd`:/log:/log \
-v `pwd`:/license:/license \
-v `pwd`:/tmp:/tmp \
-v `pwd`:/service_account_json.json:/service_account_json.json \
harbor.h2o.ai/h2oai/dai-centos7-x86_64:1.9.0-cuda10.0.23
```

This example shows how to configure Snowflake options in the config.toml file, and then specify that file when starting Driverless AI in Docker.

1. Configure the Driverless AI config.toml file. Set the following configuration options.
 - `enabled_file_systems = "file, snow"`
 - `snowflake_account = "<account_id>"`
 - `snowflake_user = "<username>"`
 - `snowflake_password = "<password>"`
2. Mount the config.toml file into the Docker container.

```
nvidia-docker run \
--pid=host \
--init \
--rm \
--shm-size=256m \
--add-host name.node:172.16.2.186 \
-e DRIVERLESS_AI_CONFIG_FILE=/path/in/docker/config.toml \
-p 12345:12345 \
-v /local/path/to/config.toml:/path/in/docker/config.toml \
-v /etc/passwd:/etc/passwd:ro \
-v /etc/group:/etc/group:ro \
-v /tmp/dtmp/:/tmp \
-v /tmp/diag/:/log \
-v /tmp/dlicense/:/license \
-v /tmp/ddata/:/data \
-u $(id -u):$(id -g) \
harbor.h2o.ai/h2oai/dai-centos7-x86_64:1.9.0-cuda10.0.23
```

This example enables the Snowflake data connector with authentication by passing the account, user, and password variables.

1. Export the Driverless AI config.toml file or add it to `~/.bashrc`. For example:

```
# DEB and RPM
export DRIVERLESS_AI_CONFIG_FILE="/etc/dai/config.toml"

# TAR SH
export DRIVERLESS_AI_CONFIG_FILE="/path/to/your/unpacked/dai/directory/config.toml"
```

2. Specify the following configuration options in the config.toml file.

```
# File System Support
# upload : standard upload feature
# file : local file system/server file system
# hdfs : Hadoop file system, remember to configure the HDFS config folder path and keytab below
# dtap : Blue Data Tap file system, remember to configure the DTap section below
# s3 : Amazon S3, optionally configure secret and access key below
# gcs : Google Cloud Storage, remember to configure gcs_path_to_service_account_json below
# g bq : Google Big Query, remember to configure gcs_path_to_service_account_json below
# minio : Minio Cloud Storage, remember to configure secret and access key below
# snow : Snowflake Data Warehouse, remember to configure Snowflake credentials below (account name, username, password)
# kdb : KDB+ Time Series Database, remember to configure KDB credentials below (hostname and port, optionally: username, password, ↵
#       classpath, and jvm_args)
# azrbs : Azure Blob Storage, remember to configure Azure credentials below (account name, account key)
# jdbc: JDBC Connector, remember to configure JDBC below. (jdbc_app_configs)
# hive: Hive Connector, remember to configure Hive below. (hive_app_configs)
# recipe_url: load custom recipe from URL
# recipe_file: load custom recipe from local file system
enabled_file_systems = "file, snow"

# Snowflake Connector credentials
snowflake_account = "<account_id>"
snowflake_user = "<username>"
snowflake_password = "<password>"
```

3. Save the changes when you are done, then stop/restart Driverless AI.

15.11.3 Adding Datasets Using Snowflake

After the Snowflake connector is enabled, you can add datasets by selecting **Snowflake** from the **Add Dataset (or Drag and Drop)** drop-down menu.



Specify the following information to add your dataset.

1. **Enter Database:** Specify the name of the Snowflake database that you are querying.
2. **Enter Warehouse:** Specify the name of the Snowflake warehouse that you are querying.
3. **Enter Schema:** Specify the schema of the dataset that you are querying.
4. **Enter Name for Dataset to Be Saved As:** Specify a name for the dataset to be saved as. Note that this can only be a CSV file (for example, **myfile.csv**).
5. **Enter Username:** (Optional) Specify the username associated with this Snowflake account. This can be left blank if `snowflake_user` was specified in the `config.toml` when starting Driverless AI; otherwise, this field is required.
6. **Enter Password:** (Optional) Specify the password associated with this Snowflake account. This can be left blank if `snowflake_password` was specified in the `config.toml` when starting Driverless AI; otherwise, this field is required.
7. **Enter Role:** (Optional) Specify your role as designated within Snowflake. See <https://docs.snowflake.net/manuals/user-guide/security-access-control-overview.html> for more information.
8. **Enter Region:** (Optional) Specify the region of the warehouse that you are querying. This can be found in the Snowflake-provided URL to access your database (as in `<optional-deployment-name>.<region>.<cloud-provider>.snowflakecomputing.com`). This is optional and can also be left blank if `snowflake_url` was specified with a `<region>` in the `config.toml` when starting Driverless AI.
9. **Enter File Formatting Parameters:** (Optional) Specify any additional parameters for formatting your datasets. Available parameters are listed in <https://docs.snowflake.com/en/sql-reference/sql/create-file-format.html#type-csv>. (Note: Use only parameters for `TYPE = CSV`.) For example, if your dataset includes a text column that contains commas, you can specify a different delimiter using `FIELD_DELIMITER='character'`. Multiple parameters must be separated with spaces:

```
FIELD_DELIMITER=',' FIELD_OPTIONALLY_ENCLOSED_BY="" SKIP_BLANK_LINES=TRUE
```

Note: Be sure that the specified delimiter is not also used as a character within a cell; otherwise an error will occur. For example, you might specify the following to load the “AMAZON_REVIEWS” dataset:

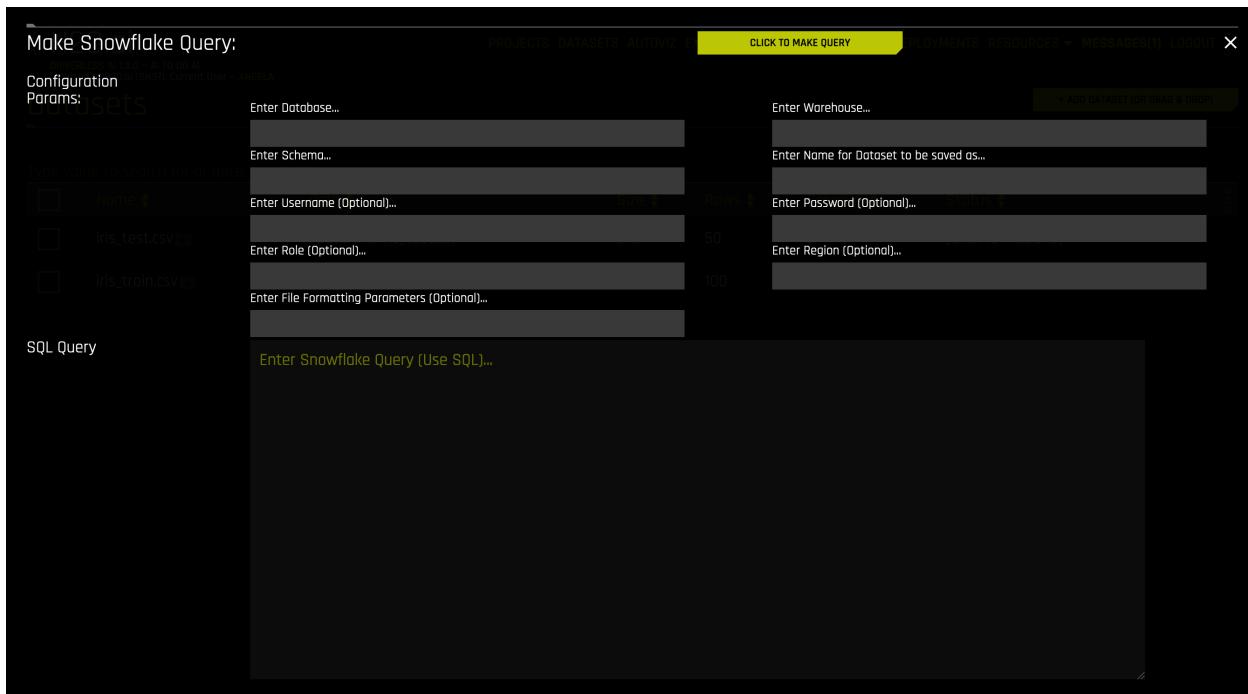
- Database: UTIL_DB
- Warehouse: DAI_SNOWFLAKE_TEST
- Schema: AMAZON_REVIEWS_SCHEMA
- Query: SELECT * FROM AMAZON_REVIEWS
- Enter File Formatting Parameters (Optional): FIELD_OPTIONALLY_ENCLOSED_BY = “”

In the above example, if the FIELD_OPTIONALLY_ENCLOSED_BY option is not set, the following row will result in a failure to import the dataset (as the dataset’s delimiter is , by default):

```
positive, 2012-05-03,Wonderful\, tasty taffy,0,0,3,5,2012,Thu,0
```

10. **Enter Snowflake Query:** Specify the Snowflake query that you want to execute.

11. When you are finished, select the **Click to Make Query** button to add the dataset.



15.12 Data Recipe URL Setup

Driverless AI allows you to explore data recipe URL data sources from within the Driverless AI application. This section provides instructions for configuring Driverless AI to work with data recipe URLs. When enabled (default), you will be able to modify datasets that have been added to Driverless AI. (Refer to `modify_by_recipe` for more information.)

Notes:

- This connector is enabled by default. These steps are provided in case this connector was previously disabled and you want to re-enable it.
- Depending on your Docker install version, use either the `docker run --runtime=nvidia (>= Docker 19.03)` or `nvidia-docker (< Docker 19.03)` command when starting the Driverless AI Docker image. Use `docker version` to check which version of Docker you are using.

15.12.1 Enable Data Recipe URL

Docker Image Installs

Docker Image with the config.toml

Native Installs

This example enables the data recipe URL data connector.

```
nvidia-docker run \
--shm-size=256m \
--add-host name.node:172.16.2.186 \
-e DRIVERLESS_AI_ENABLED_FILE_SYSTEMS="file, recipe_url" \
-p 12345:12345 \
--init -it --rm \
-v /tmp/dtmp/:/tmp \
-v /tmp/dlog/:/log \
-v /tmp/dlicense/:/license \
-v /tmp/ddata/:/data \
-u ${id -u};${id -g} \
h2oai/dai-centos7-x86_64:1.9.0-cuda10.0.23
```

This example shows how to enable the Data Recipe URL data connector in the config.toml file, and then specify that file when starting Driverless AI in Docker. Note that `recipe_url` is enabled in the config.toml file by default.

1. Configure the Driverless AI config.toml file. Set the following configuration options.
 - `enabled_file_systems = "file, upload, recipe_url"`
2. Mount the config.toml file into the Docker container.

```
nvidia-docker run \
--pid=host \
--init \
--rm \
--shm-size=256m \
--add-host name.node:172.16.2.186 \
-e DRIVERLESS_AI_CONFIG_FILE=/path/in/docker/config.toml \
-p 12345:12345 \
-v /local/path/to/config.toml:/path/in/docker/config.toml \
-v /etc/passwd:/etc/passwd:ro \
-v /etc/group:/etc/group:ro \
-v /tmp/dtmp/:/tmp \
-v /tmp/dlog/:/log \
-v /tmp/dlicense/:/license \
-v /tmp/ddata/:/data \
-u ${id -u};${id -g} \
h2oai/dai-centos7-x86_64:1.9.0-cuda10.0.23
```

This example enables the Data Recipe URL data connector. Note that `recipe_url` is enabled by default.

1. Export the Driverless AI config.toml file or add it to `~/.bashrc`. For example:

```
# DEB and RPM
export DRIVERLESS_AI_CONFIG_FILE="/etc/dai/config.toml"

# TAR SH
export DRIVERLESS_AI_CONFIG_FILE="/path/to/your/unpacked/dai/directory/config.toml"
```

2. Specify the following configuration options in the config.toml file.

```
# File System Support
# upload : standard upload feature
# file : local file system/server file system
# hdfs : Hadoop file system, remember to configure the HDFS config folder path and keytab below
# dtap : Blue Data Tap file system, remember to configure the DTap section below
# s3 : Amazon S3, optionally configure secret and access key below
# gcs : Google Cloud Storage, remember to configure gcs_path_to_service_account_json below
# gbq : Google Big Query, remember to configure gcs_path_to_service_account_json below
# minio : Minio Cloud Storage, remember to configure secret and access key below
# snow : Snowflake Data Warehouse, remember to configure Snowflake credentials below (account name, username, password)
```

(continues on next page)

(continued from previous page)

```
# kdb : KDB+ Time Series Database, remember to configure KDB credentials below (hostname and port, optionally: username, password, ↵
# azrbs : Azure Blob Storage, remember to configure Azure credentials below (account name, account key)
# jdbc: JDBC Connector, remember to configure JDBC below. (jdbc_app_configs)
# hive: Hive Connector, remember to configure Hive below. (hive_app_configs)
# recipe_url: load custom recipe from URL
# recipe_file: load custom recipe from local file system
enabled_file_systems = "file, recipe_url"
```

3. Save the changes when you are done, then stop/restart Driverless AI.

15.13 Data Recipe File Setup

Driverless AI allows you to explore data recipe file data sources from within the Driverless AI application. This section provides instructions for configuring Driverless AI to work with local data recipe files. When enabled (default), you will be able to modify datasets that have been added to Driverless AI. (Refer to `modify_by_recipe` for more information.)

Notes:

- This connector is enabled by default. These steps are provided in case this connector was previously disabled and you want to re-enable it.
- Depending on your Docker install version, use either the `docker run --runtime=nvidia` (\geq Docker 19.03) or `nvidia-docker` (< Docker 19.03) command when starting the Driverless AI Docker image. Use `docker version` to check which version of Docker you are using.

15.13.1 Enable Data Recipe File

Docker Image Installs

Docker Image with the config.toml

Native Installs

This example enables the data recipe file data connector.

```
nvidia-docker run \
--shm-size=256m \
--add-host name.node:172.16.2.186 \
-e DRIVERLESS_AI_ENABLED_FILE_SYSTEMS="file,upload,recipe_file" \
-p 12345:12345 \
--init -it --rm \
-v /tmp/dtmp:/tmp \
-v /tmp/dlog:/log \
-v /tmp/dlicense:/license \
-v /tmp/ddata:/data \
-u $(id -u):$(id -g) \
h2oai/dai-centos7-x86_64:1.9.0-cudal0.0.23
```

This example shows how to enable the Upload Data Recipe connector in the config.toml file, and then specify that file when starting Driverless AI in Docker. Note that `recipe_file` is enabled in the config.toml file by default.

1. Configure the Driverless AI config.toml file. Set the following configuration options.
 - `enabled_file_systems = "file, upload, recipe_file"`
2. Mount the config.toml file into the Docker container.

```
nvidia-docker run \
--pid=host \
--init \
--rm \
--shm-size=256m \
--add-host name.node:172.16.2.186 \
-e DRIVERLESS_AI_CONFIG_FILE=/path/in/docker/config.toml \
-p 12345:12345 \
-v /local/path/to/config.toml:/path/in/docker/config.toml \
-v /etc/passwd:/etc/passwd:ro \
-v /etc/group:/etc/group:ro \
```

(continues on next page)

(continued from previous page)

```
-v /tmp/dtmp:/tmp \
-v /tmp/dlog:/log \
-v /tmp/dlicense:/license \
-v /tmp/ddata:/data \
-u ${id -u} ${id -g} \
h2oai/dai-centos7-x86_64:1.9.0-cuda10.0.23
```

This example enables the Upload Data Recipe data connector. Note that `recipe_file` is enabled by default.

1. Export the Driverless AI config.toml file or add it to `~/.bashrc`. For example:

```
# DEB and RPM
export DRIVERLESS_AI_CONFIG_FILE="/etc/dai/config.toml"

# TAR SH
export DRIVERLESS_AI_CONFIG_FILE="/path/to/your/unpacked/dai/directory/config.toml"
```

2. Specify the following configuration options in the config.toml file.

```
# File System Support
# upload : standard upload feature
# file : local file system/server file system
# hdfs : Hadoop file system, remember to configure the HDFS config folder path and keytab below
# dtap : Blue Data Tap file system, remember to configure the DTap section below
# s3 : Amazon S3, optionally configure secret and access key below
# gcs : Google Cloud Storage, remember to configure gcs_path_to_service_account_json below
# gbq : Google Big Query, remember to configure gcs_path_to_service_account_json below
# minio : Minio Cloud Storage, remember to configure secret and access key below
# snow : Snowflake Data Warehouse, remember to configure Snowflake credentials below (account name, username, password)
# kdb : KDB+ Time Series Database, remember to configure KDB credentials below (hostname and port, optionally: username, password, ↵
#       classpath, and jvm_args)
# azrbs : Azure Blob Storage, remember to configure Azure credentials below (account name, account key)
# jdbc: JDBC Connector, remember to configure JDBC below. (jdbc_app_configs)
# hive: Hive Connector, remember to configure Hive below. (hive_app_configs)
# recipe_url: load custom recipe from URL
# recipe_file: load custom recipe from local file system
enabled_file_systems = "file, recipe_file"
```

3. Save the changes when you are done, then stop/restart Driverless AI.

CONFIGURING AUTHENTICATION

Driverless AI supports Client Certificate, LDAP, Local, mTLS, OpenID, PAM, none, and unvalidated (default) authentication. These can be configured by specifying the environment variables when starting the Driverless AI Docker image or by specifying the appropriate configuration options in the config.toml file.

Notes:

- You can enable multiple authentication methods with the `additional_authentication_methods` config.toml setting. These are enabled alongside the default method specified with the `authentication_method` config.toml setting. Login forms for each additional method are available on the `/login/<authentication_method>` path.
- If multiple authentication methods are enabled, each method must be set up so that it results in the same user-name to provide access to the same resources.
- Driverless AI is also integrated with IBM Spectrum Conductor and supports authentication from Conductor. Contact sales@h2o.ai for more information about using IBM Spectrum Conductor authentication.
- Driverless AI does not support LDAP client auth. If you have LDAP client auth enabled, then the Driverless AI LDAP connector will not work.

16.1 Client Certificate Authentication Example

This section describes how to configure client certificate authentication in Driverless AI.

16.1.1 Client Certificate and SSL Configuration Options

The following options can be specified when configuring client certificate authentication.

SSL Configuration Options

Mutual TLS authentication (mTLS) must be enabled in order to enable Client Certificate Authentication. Use the following configuration options to configure mTLS. Refer to the [mTLS Authentication topic](#) for more information on how to enable mTLS.

- `ssl_client_verify_mode`: Sets the client verification mode. Choose from the following verification modes:
 - `CERT_NONE`: The client will not need to provide a certificate. If it does provide a certificate, any resulting verification errors are ignored.
 - `CERT_OPTIONAL`: The client does not need to provide a certificate. If it does provide a certificate, it is verified against the configured CA chains.

- **CERT_REQUIRED**: The client needs to provide a certificate for verification. Note that you will need to configure the `ssl_client_key_file` and `ssl_client_crt_file` options when this mode is selected in order for Driverless to be able to verify its own callback requests.
- `ssl_ca_file`: Specifies the path to the [certification authority \(CA\)](#) certificate file. This certificate will be used to verify the client certificate when client authentication is enabled. If this is not specified, clients are verified using the default system certificates.
- `ssl_client_key_file`: Required if `ssl_client_verify_mode = "CERT_REQUIRED"`. Specifies the HTTPS settings path to the [private key](#) that Driverless AI uses to authenticate itself.
- `ssl_client_crt_file`: Required if `ssl_client_verify_mode = "CERT_REQUIRED"`. Specifies the HTTPS settings path to the [client certificate](#) that Driverless AI will use to authenticate itself.

Client Certificate Options

- `auth_tls_crl_file`: The path to the [certificate revocation list \(CRL\)](#) file that is used to verify the client certificate.
- `auth_tls_user_lookup`: Specifies how a user's identity is obtained. Choose from the following:
 - `REGEXP_ONLY`: Uses `auth_tls_subject_field` and `auth_tls_field_parse_regexp` to extract the username from the client certificate.
 - `LDAP_LOOKUP`: Uses the LDAP server to obtain the username. (Refer to the [LDAP Authentication Example](#) section for information about additional LDAP Authentication configuration options.)

Used with `LDAP_LOOKUP`:

- `auth_tls_ldap_server`: Specifies the LDAP server hostname or IP address.
- `auth_tls_ldap_port`: Specifies the LDAP server port number. This is 389 by default.
- `auth_tls_ldap_use_ssl`: Specifies whether to enable (True) or disable (False) SSL when connecting to the LDAP server.
- `auth_tls_ldap_tls_file`: Specifies the path to the SSL certificate.
- `auth_tls_ldap_bind_dn`: Specifies the complete DN of the LDAP bind user.
- `auth_tls_ldap_bind_password`: Specifies the password for the LDAP bind.
- `auth_tls_subject_field`: The subject field that is used as a source for a username or other values that provide further validation.
- `auth_tls_field_parse_regexp`: The regular expression that is used to parse the subject field in order to obtain the username or other values that provide further validation.
- `auth_tls_ldap_search_base`: Specifies the location in the Directory Information Tree (DIT) where the search will start.
- `auth_tls_ldap_search_filter`: Specifies an LDAP search filter that is used to find a specific user with `LDAP_LOOKUP` when using the `tls_certificate` authentication method. This can be dynamically built by using the named capturing groups from `auth_tls_field_parse_regexp` for substitution:

```
auth_tls_field_parse_regexp = "\w+ (?P<id>\d+)"
auth_tls_ldap_search_filter = "(&(objectClass=person) (id={{id}}))"
```

- `auth_tls_ldap_username_attribute`: Specifies the LDAP record attribute that is used as a username.
- `auth_tls_ldap_authorization_lookup_filter`: (Optional) Specifies an additional search filter that is performed after the user is found. This is useful for checking whether a user is a member of a specific group in LDAP schemas where group membership is defined within group entries as opposed to individual user

entries. (Refer to the *Lookup Filter Example* section that follows to see an example of how this option can be used.)

- auth_tls_ldap_authorization_search_base: Specifies the base distinguished name (DN) to start the authorization lookup from. Required when auth_tls_ldap_authorization_lookup_filter is specified.

Lookup Filter Example

The following example uses the auth_tls_ldap_authorization_lookup_filter option to determine whether individual users are members of the chemists group in an LDAP schema where group (organizational unit) membership is defined within group entries.

```
# Specify to use email as username
auth_tls_ldap_username_attribute = "mail"
# Specify search string
auth_tls_ldap_search_filter = "(&(objectClass=inetOrgPerson)(uid={{username}}))"
# Specify the base DN to start the search from
auth_tls_ldap_authorization_search_base="dc=example,dc=com"
# Filter the results of the search to determine which users are members of a specific group
auth_tls_ldap_authorization_lookup_filter = "(&(objectClass=groupOfUniqueNames)(uniqueMember=uid={{uid}},dc=example,dc=com)(ou=chemists))"
```

16.1.2 Enabling Client Certificate Authentication

Docker Image Installs

Native Installs

To enable Client Certificate authentication in Docker images, specify the authentication environment variable that you want to use. Each variable must be prepended with DRIVERLESS_AI_. The following example enables Client Certification authentication and uses LDAP_LOOKUP for the TLS user lookup method.

```
nvidia-docker run \
--pid=host \
--init \
--rm \
--shm-size=256m \
-p 12345:12345 \
-u `id -u`:`id -g` \
-e DRIVERLESS_AI_ENABLED_FILE_SYSTEMS="file,s3,hdfs" \
-e DRIVERLESS_AI_ENABLE_HTTPS="true" \
-e DRIVERLESS_AI_SSL_KEY_FILE="/etc/pki/dai-server.key" \
-e DRIVERLESS_AI_SSL_CRT_FILE="/etc/pki/dai-server.crt" \
-e DRIVERLESS_AI_SSL_CA_FILE="/etc/pki/ca.crt" \
-e DRIVERLESS_AI_SSL_CLIENT_VERIFY_MODE="CERT_REQUIRED" \
-e DRIVERLESS_AI_SSL_CLIENT_KEY_FILE="/etc/pki/dai-self.key" \
-e DRIVERLESS_AI_SSL_CLIENT_CRT_FILE="/etc/pki/dai-self.cert" \
-e DRIVERLESS_AI_AUTHENTICATION_METHOD="tls_certificate" \
-e DRIVERLESS_AI_AUTH_TLS_SUBJECT_FIELDS="CN" \
-e DRIVERLESS_AI_AUTH_TLS_CRL_FILE="/etc/pki/crl.pem" \
-e DRIVERLESS_AI_AUTH_TLS_FIELDS_PARS_REGEX="^(?P<id>.*)" \
-e DRIVERLESS_AI_AUTH_TLS_USER_LOOKUP="LDAP_LOOKUP" \
-e DRIVERLESS_AI_LDAP_SERVER="ldap.forumsys.com" \
-e DRIVERLESS_AI_LDAP_BIND_DN="cn=read-only-admin,dc=example,dc=com" \
-e DRIVERLESS_AI_LDAP_BIND_PASSWORD="password" \
-e DRIVERLESS_AI_LDAP_SEARCH_BASE="dc=example,dc=com" \
-e DRIVERLESS_AI_LDAP_USER_NAME_ATTRIBUTE="uid" \
-e DRIVERLESS_AI_LDAP_SEARCH_FILTER="(&(objectClass=inetOrgPerson)(uid={{id}}))" \
-e DRIVERLESS_AI_AUTH_TLS_LDAP_AUTHORIZATION_SEARCH_BASE="dc=example,dc=com" \
-e DRIVERLESS_AI_AUTH_TLS_LDAP_AUTHORIZATION_LOOKUP_FILTER="(&(objectClass=groupOfUniqueNames)(uniqueMember=uid={{uid}},dc=example,dc=com)(ou=chemists))"
-v `pwd`:/data:/data \
-v `pwd`:/log:/log \
-v `pwd`:/license:/license \
-v `pwd`:/tmp:/tmp \
harbor.h2o.ai/h2oai/dai-centos7-x86_64:1.9.0-cuda10.0.23
```

Native installs include DEBs, RPMs, and TAR SH installs. The following example shows how to edit the config.toml file to enable Client Certification authentication and uses the LDAP_LOOKUP for the TLS user lookup method.

1. Export the Driverless AI config.toml file or add it to ~/.bashrc. For example:

```
# DEB and RPM
export DRIVERLESS_AI_CONFIG_FILE="/etc/dai/config.toml"

# TAR SH
export DRIVERLESS_AI_CONFIG_FILE="/path/to/your/unpacked/dai/directory/config.toml"
```

2. Open the config.toml file and edit the following authentication variables. The config.toml file is available in the etc/dai folder after Driverless AI is installed.

```
# https settings
enable_https = true

# https settings
# Path to the SSL key file
#
ssl_key_file = "/etc/pki/dai-server.key"

# https settings
# Path to the SSL certificate file
#
ssl_crt_file = "/etc/pki/dai-server.crt"

# https settings
# Path to the Certification Authority certificate file. This certificate will be
# used when to verify client certificate when client authentication is turned on.
# If this is not set, clients are verified using default system certificates.
#
ssl_ca_file = "/etc/pki/ca.crt"

# https settings
# Sets the client verification mode.
# CERT_NONE: Client does not need to provide the certificate and if it does any
# verification errors are ignored.
# CERT_OPTIONAL: Client does not need to provide the certificate and if it does
# certificate is verified agains set up CA chains.
# CERT_REQUIRED: Client needs to provide a certificate and certificate is
# verified.
# You'll need to set 'ssl_client_key_file' and 'ssl_client_crt_file'
# When this mode is selected for Driverless to be able to verify
# it's own callback requests.
#
ssl_client_verify_mode = "CERT_REQUIRED"

# https settings
# Path to the private key that Driverless will use to authenticate itself when
# CERT_REQUIRED mode is set.
#
ssl_client_key_file = "/etc/pki/dai-self.key"

# https settings
# Path to the client certificate that Driverless will use to authenticate itself
# when CERT_REQUIRED mode is set.
#
ssl_client_crt_file = "/etc/pki/dai-self.crt"

# Enable client certificate authentication
authentication_method = "tls_certificate"

# Subject field that is used as a source for a username or other values that provide further validation
auth_tls_subject_field = "CN"

# Path to the CRL file that will be used to verify client certificate.
auth_tls_crl_file = "/etc/pki/crl.pem"

# Sets up the way how user identity would be obtained
# REGEXP_ONLY: Will use 'auth_tls_subject_field' and 'auth_tls_field_parse_regex'
# to extract the username from the client certificate.
# LDAP_LOOKUP: Will use LDAP server to lookup for the username.
# 'ldap_server', 'ldap_use_ssl', 'ldap_tis_file', 'ldap_bind_dn',
# 'ldap_bind_password' options are used to establish
# the connection with the LDAP server.
# 'auth_tls_subject_field' and 'auth_tls_field_parse_regex'
# options are used to parse the certificate.
# 'ldap_search_base', 'ldap_search_filter', and
# 'ldap_username_attribute' options are used to do the lookup.
# 'ldap_search_filter' can be built dynamically using the named
# capturing groups from the 'auth_tls_field_parse_regex' for
# substitution.
# Example:
# auth_tls_field_parse_regex = "\w+ (?P<id>\d+)"
# ldap_search_filter = "(|(objectClass=person) (id={{id}}))"
auth_tis_user_lookup = "LDAP_LOOKUP"

# Regular expression that is used to parse the subject field in order to
# obtain the username or other values that provide further validation
auth_tls_field_parse_regex = "\w+ (?P<id>\d+)"

# ldap server domain or ip
ldap_server = "ldap.forumsys.com"

# Complete DN of the LDAP bind user
ldap_bind_dn = "cn=read-only-admin,dc=example,dc=com"

# Password for the LDAP bind
ldap_bind_password = "password"

# the location in the DIT where the search will start
ldap_search_base = "dc=example,dc=com"

# specify key to find user name
ldap_user_name_attribute = "uid"

# A string that describes what you are searching for. You can use Python
# substitution to have this constructed dynamically.
# (only {{DAI_USERNAME}} is supported)
ldap_search_filter = "(|(objectClass=inetOrgPerson) (uid={{id}}))"

# Base DN where to start the Authorization lookup. Used when
# 'auth_tis_ldap_authorization_lookup_filter' is set.
# auth_tis_ldap_authorization_search_base="dc=example,dc=com"
```

(continues on next page)

(continued from previous page)

```

# Sets optional additional lookup filter that is performed after the
# user is found. This can be used for example to check whether the is member of
# particular group.
# Filter can be built dynamically from the attributes returned by the lookup.
# Authorization fails when search does not return any entry. If one ore more
# entries are returned authorization succeeds.
# Example:
# auth_tis_field_parse_regexp = "\w+ (?P<id>\d+)"
# ldap_search_filter = "(&(objectClass=person)(id={{id}}))"
# auth_tis_ldap_authorization_lookup_filter = "(&(objectClass=group)(member=uid={{uid}}),dc=example,dc=com)"
# If this option is empty no additional lookup is done and just a successful user
# lookup is enough to authorize the user.
# auth_tls_ldap_authorization_lookup_filter = "(&(objectClass=groupOfUniqueNames)(uniqueMember=uid={{uid}}),dc=example,dc=com)(ou=chemists)"

```

3. Start (or restart) Driverless AI.

16.2 LDAP Authentication Example

This section describes how to enable [Lightweight Directory Access Protocol](#) in Driverless AI. The available parameters can be specified as environment variables when starting the Driverless AI Docker image, or they can be set via the config.toml file for native installs. Upon completion, all the users in the configured LDAP should be able to log in to Driverless AI and run experiments, visualize datasets, interpret models, etc.

Note: Driverless AI does not support LDAP client auth. If you have LDAP client auth enabled, then the Driverless AI LDAP connector will not work.

16.2.1 Description of Configuration Attributes

The following options can be specified when enabling LDAP authentication.

- `ldap_server`: The LDAP server domain or IP.
- `ldap_port`: The LDAP server port.
- `ldap_bind_dn`: The complete distinguished name (DN) of the LDAP bind user.
- `ldap_bind_password`: The password for the LDAP bind.
- `ldap_tls_file`: The Transport Layer Security (TLS) certificate file location.
- `ldap_use_ssl`: Whether to enable (TRUE) or disable (FALSE) SSL.
- `ldap_search_base`: The location in the Directory Information Tree (DIT) where the search will start.
- `ldap_search_filter`: A string that describes what you are searching for. You can use Python substitution to have this constructed dynamically. (Only {{DAI_USERNAME}} is supported. For example, “(&(objectClass=person)(cn:dn:={{DAI_USERNAME}}))”.)
- `ldap_search_attributes`: LDAP attributes to return from search.
- `ldap_user_name_attribute="uid"`: Specify the key to find user name.

16.2.2 LDAP without SSL

The following examples describe how to enable LDAP without SSL when running Driverless AI in the Docker image or through native installs. If the configuration and authentication authentication are successful, the user can access Driverless AI and run experiments, visualize datasets, interpret models, etc.

Docker Image Installs

Native Installs

The following example shows how to configure LDAP without SSL when starting the Driverless AI Docker image.

```
nvidia-docker run \
--pid=host \
--init \
--rm \
--shm-size=256m \
-p 12345:12345 \
-u `id -u`:`id -g` \
-e DRIVERLESS_AI_ENABLED_FILE_SYSTEMS="file,s3,hdfs" \
-e DRIVERLESS_AI_AUTHENTICATION_METHOD="ldap" \
-e DRIVERLESS_AI_LDAP_USE_SSL="false" \
-e DRIVERLESS_AI_LDAP_SERVER="ldap.forumsys.com" \
-e DRIVERLESS_AI_LDAP_PORT="389" \
-e DRIVERLESS_AI_LDAP_SEARCH_BASE="dc=example,dc=com" \
-e DRIVERLESS_AI_LDAP_BIND_DN="cn=read-only-admin,dc=example,dc=com" \
-e DRIVERLESS_AI_LDAP_BIND_PASSWORD="password" \
-e DRIVERLESS_AI_LDAP_SEARCH_FILTER="(&(objectClass=person)(cn:dn:={DAI_USERNAME}))" \
-e DRIVERLESS_AI_LDAP_USER_NAME_ATTRIBUTE="uid" \
-v `pwd`/data:/data \
-v `pwd`/log:/log \
-v `pwd`/license:/license \
-v `pwd`/tmp:/tmp \
harbor.h2o.ai/h2oai/dai-centos7-x86_64:1.9.0-cuda10.0.23
```

The following example shows how to configure LDAP without SSL when starting Driverless AI from a native install. Native installs include DEBs, RPMs, and TAR SH installs.

1. Export the Driverless AI config.toml file or add it to `~/.bashrc`. For example:

```
# DEB and RPM
export DRIVERLESS_AI_CONFIG_FILE="/etc/dai/config.toml"

# TAR SH
export DRIVERLESS_AI_CONFIG_FILE="/path/to/your/unpacked/dai/directory/config.toml"
```

2. Enable LDAP authentication without SSL.

```
# Enable LDAP authentication
authentication_method = "ldap"

# Specify the LDAP server domain or IP to connect to
ldap_server = "ldap.forumsys.com"

# Specify the LDAP port to connect to
ldap_port = "389"

# Disable SSL
ldap_use_ssl="false"

# Specify the location in the DIT where the search will start
ldap_search_base = "dc=example,dc=com"

# Specify the LDAP search filter
# This is A string that describes what you are searching for. You
# can use Python substitution to have this constructed dynamically.
# (Only {{DAI_USERNAME}} is supported. For example, "(&(objectClass=person)(cn:dn:={{DAI_USERNAME}}))".
ldap_search_filter = "(&(objectClass=person)(cn:dn:={{DAI_USERNAME}}))"

# Specify the complete DN of the LDAP bind user
ldap_bind_dn = "cn=read-only-admin,dc=example,dc=com"

# Specify the LDAP password for the above user
ldap_bind_password = "password"

# Specify a key to find the user name
ldap_user_name_attribute = "uid"
```

3. Start (or restart) Driverless AI. Note that the command used to start Driverless AI varies depending on your install type.

```
# Linux RPM or DEB with systemd
sudo systemctl start dai

# Linux RPM or DEB without systemd
sudo -H -u dai /opt/h2oai/dai/run-dai.sh
```

(continues on next page)

(continued from previous page)

```
# Linux TAR SH
./run-dai.sh
```

If authentication is successful, the user can access Driverless AI and run experiments, visualize datasets, interpret models, etc.

16.2.3 LDAP with SSL

These examples show how to enable LDAP authentication with SSL and additional parameters that can be specified as environment variables when starting the Driverless AI Docker image, or they can be set via the config.toml file for native installs. Upon completion, all the users in the configured LDAP should be able to log in to Driverless AI and run experiments, visualize datasets, interpret models, etc.

Docker Image Installs

Native Installs

Specify the following LDAP environment variables when starting the Driverless AI Docker image. This example enables LDAP authentication and shows how to specify additional options enabling SSL.

```
nvidia-docker run \
--pid=host \
--init \
--rm \
--shm-size=256m \
-p 12345:12345 \
-u `id -u`:`id -g` \
-e DRIVERLESS_AI_ENABLED_FILE_SYSTEMS="file,s3,hdfs" \
-e DRIVERLESS_AI_AUTHENTICATION_METHOD="ldap" \
-e DRIVERLESS_AI_LDAP_SERVER="ldap.forumsys.com" \
-e DRIVERLESS_AI_LDAP_PORT="389" \
-e DRIVERLESS_AI_LDAP_SEARCH_BASE="dc/example,dc=com" \
-e DRIVERLESS_AI_LDAP_SEARCH_FILTER="(objectClass=person)(cn:dn:={{DAI_USERNAME}})" \
-e DRIVERLESS_AI_LDAP_USE_SSL="true" \
-e DRIVERLESS_AI_LDAP_TLS_FILE="/tmp/abc-def-root.cer" \
-e DRIVERLESS_AI_LDAP_BIND_DN="cn=read-only-admin,dc=example,dc=com" \
-e DRIVERLESS_AI_LDAP_BIND_PASSWORD="password" \
-e DRIVERLESS_AI_LDAP_USER_NAME_ATTRIBUTE="uid" \
-v `pwd`:/data \
-v `pwd`:/log \
-v `pwd`:/license \
-v `pwd`:/tmp \
harbor.h2o.ai/h2oai/dai-centos7-x86_64:1.9.0-cuda10.0.23
```

Native installs include DEBs, RPMs, and TAR SH installs.

1. Export the Driverless AI config.toml file or add it to `~/.bashrc`. For example:

```
# DEB and RPM
export DRIVERLESS_AI_CONFIG_FILE="/etc/dai/config.toml"

# TAR SH
export DRIVERLESS_AI_CONFIG_FILE="/path/to/your/unpacked/dai/directory/config.toml"
```

2. Enable LDAP authentication with SSL.

```
# Enable LDAP authentication
authentication_method = "ldap"

# Specify the LDAP server domain or IP to connect to
ldap_server = "ldap.forumsys.com"

# Specify the LDAP port to connect to
ldap_port = "389"

# Specify the location in the DIT where the search will start
ldap_search_base = "dc@example,dc=com"

# Specify the LDAP search filter
# This is a string that describes what you are searching for. You
# can use Python substitution to have this constructed dynamically.
# (Only {{DAI_USERNAME}} is supported.)
ldap_search_filter = "(objectClass=person)(cn:dn:={{DAI_USERNAME}})"

# If the LDAP connection to the LDAP server needs an SSL certificate,
# then this needs to be specified
ldap_use_ssl = "True"

# Specify the LDAP TLS file location if SSL is set to True
ldap_tls_file = "/tmp/abc-def-root.cer"
```

(continues on next page)

(continued from previous page)

```
# Complete DN of the LDAP bind user  
ldap_bind_dn = "cn=read-only-admin,dc=example,dc=com"  
  
# Specify the LDAP password for the above user  
ldap_bind_password = "password"  
  
# Specify a key to find the user name  
ldap_user_name_attribute = "uid"
```

3. Start (or restart) Driverless AI. Users can now launch Driverless AI using their LDAP credentials. Note that the command used to start Driverless AI varies depending on your install type.

```
# Linux RPM or DEB with systemd  
sudo systemctl start dai  
  
# Linux RPM or DEB without systemd  
sudo -H -u dai /opt/h2oai/dai/run-dai.sh  
  
# Linux TAR SH  
.run-dai.sh
```

If authentication is successful, the user can access Driverless AI and run experiments, visualize datasets, interpret models, etc.

16.3 Local Authentication Example

This section describes how to enable local authentication in Driverless AI.

Docker Image Installs

Native Installs

To enable authentication in Docker images, specify the authentication environment variable that you want to use. Each variable must be prepended with DRIVERLESS_AI_. The example below starts Driverless AI with environment variables the enable the following:

- Local authentication when starting Driverless AI
- S3 and HDFS access (without authentication)

```
nvidia-docker run \  
--pid=host \  
--init \  
--rm \  
--shm-size=256m \  
-p 12345:12345 \  
-u `id -u`:`id -g` \  
-e DRIVERLESS_AI_ENABLED_FILE_SYSTEMS="file,s3,hdfs" \  
-e DRIVERLESS_AI_AUTHENTICATION_METHOD="local" \  
-e DRIVERLESS_AI_LOCAL_HTPASSWD_FILE=<htpasswd_file_location> \  
-v `pwd`:/data:/data \  
-v `pwd`:/log:/log \  
-v `pwd`:/license:/license \  
-v `pwd`:/tmp:/tmp \  
harbor.h2o.ai/h2oai/dai-centos7-x86_64:1.9.0-cuda10.0.23
```

Native installs include DEBs, RPMs, and TAR SH installs. The example below shows the configuration options in the config.toml file to set when enabling the following:

- Local authentication when starting Driverless AI
- S3 and HDFS access (without authentication)

1. Export the Driverless AI config.toml file or add it to `~/.bashrc`. For example:

```
# DEB and RPM  
export DRIVERLESS_AI_CONFIG_FILE="/etc/dai/config.toml"  
  
# TAR SH  
export DRIVERLESS_AI_CONFIG_FILE="/path/to/your/unpacked/dai/directory/config.toml"
```

2. Open the config.toml file and edit the authentication variables. The config.toml file is available in the `etc/dai` folder after the RPM or DEB is installed.

```

# File System Support
# file : local file system/server file system
# hdfs : Hadoop file system, remember to configure the hadoop coresite and keytab below
# s3 : Amazon S3, optionally configure secret and access key below
# gcs : Google Cloud Storage, remember to configure gcs_path_to_service_account_json below
# gbq : Google Big Query, remember to configure gcs_path_to_service_account_json below
enabled_file_systems = "file,s3,hdfs"

# authentication_method
# unvalidated : Accepts user id and password, does not validate password
# none : Does not ask for user id or password, authenticated as admin
# pam : Accepts user id and password, Validates user with operating system
# ldap : Accepts user id and password, Validates against an ldap server, look
# local: Accepts a user id and password, Validated against a htpasswd file provided in local_htpasswd_file
# for additional settings under LDAP settings
authentication_method = "local"

# Local password file
# Generating a htpasswd file: see syntax below
# htpasswd -B "<location_to_place_htpasswd_file>" "<username>"
# note: -B forces use of bcrypt, a secure encryption method
local_htpasswd_file = "<htpasswd_file_location>"
```

- Start (or restart) Driverless AI. Note that the command used to start Driverless AI varies depending on your install type.

```

# Linux RPM or DEB with systemd
sudo systemctl start dai

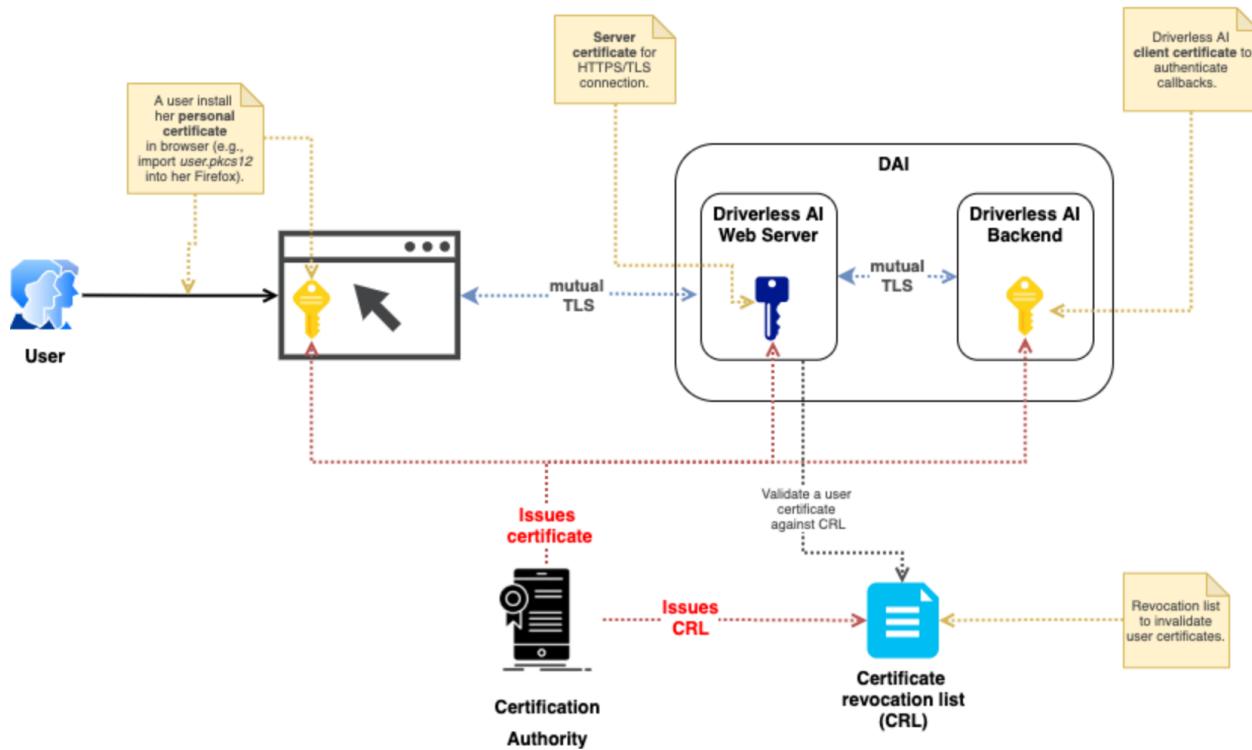
# Linux RPM or DEB without systemd
sudo -H -u dai /opt/h2oai/dai/run-dai.sh

# Linux TAR SH
./run-dai.sh
```

16.4 mTLS Authentication Example

Driverless AI supports [Mutual TLS authentication \(mTLS\)](#) by setting a specific verification mode along with a certificate authority file, an SSL private key, and an SSL certificate file. The diagram below is a visual representation of the mTLS authentication process.

Scenario: mTLS enabled + TLS authentication



16.4.1 Description of Configuration Attributes

Use the following configuration options to configure mTLS.

- `ssl_client_verify_mode`: Sets the client verification mode. Choose from the following verification modes:
 - `CERT_NONE`: The client will not need to provide a certificate. If it does provide a certificate, any resulting verification errors are ignored.
 - `CERT_OPTIONAL`: The client does not need to provide a certificate. If it does provide a certificate, it is verified against the configured CA chains.
 - `CERT_REQUIRED`: The client needs to provide a certificate for verification. Note that you will need to configure the `ssl_client_key_file` and `ssl_client_crt_file` options when this mode is selected in order for Driverless to be able to verify its own callback requests.
- `ssl_ca_file`: Specifies the path to the **certification authority (CA)** certificate file, provided by your organization. This certificate will be used to verify the client certificate when client authentication is enabled. If this is not specified, clients are verified using the default system certificates.
- `ssl_key_file`: Specifies your web server private key file. This is normally created by your organization's sys admin.
- `ssl_crt_file`: Specifies your web server public certificate file. This is normally created by your organization's sys admin.
- `ssl_client_key_file`: Required if `ssl_client_verify_mode = "CERT_REQUIRED"`. Specifies the **private key** file that Driverless AI uses to authenticate itself. This is normally created by your organization's sys admin.

tion's sys admin.

- `ssl_client_crt_file`: Required if `ssl_client_verify_mode = "CERT_REQUIRED"`. Specifies the private `client certificate` file that Driverless AI will use to authenticate itself. This is normally created by your organization's sys admin.
- `auth_tls_crl_file`: Specifies the path to the `certificate revocation list` file that will be used to verify the client certificate. This file contains a list of revoked user IDs.

16.4.2 Configuration Scenarios

The table below describes user certificate behavior for mTLS authentication based on combinations of the configuration options described above.

config.toml settings	User does not have a certificate	User has a correct and valid certificate	User has a revoked certificate
<code>ssl_client_verify_mode='CERT_NONE'</code>	Certs are ignored	User certs are ignored	User revoked certs are ignored
<code>ssl_client_verify_mode='CERT_OPTIONAL'</code>	Certs are ignored	User certs are set to Driverless AI but are not used for validating the certs	User revoked certs are not validated
<code>ssl_client_verify_mode='CERT_REQUIRED'</code> allowed	User provides a valid certificate used by Driverless AI but does not authenticate the user	User revoke lists are not validated	
<code>ssl_client_verify_mode='CERT_REQUIRED'</code> AND <code>authentication_method='tls_authentication'</code>	User provides a valid certificate. The certificate is used for connecting to the Driverless AI server as well as for authentication.	User revoked certs are validated and the revoked file is provided in <code>AUTH_TLS_CRL_FILE</code>	

16.4.3 Enabling mTLS Authentication

Docker Image Installs

Native Installs

To enable mTLS authentication in Docker images, specify the authentication environment variable that you want to use. Each variable must be prepended with `DRIVERLESS_AI_`.

```
nvidia-docker run \
--pid=host \
--init \
--rm \
--shm-size=256m \
-p 12345:12345 \
-u `id -u`:`id -g` \
-e DRIVERLESS_AI_ENABLE_HTTPS=true \
-e DRIVERLESS_AI_SSL_KEY_FILE=/etc/dai/private_key.pem \
-e DRIVERLESS_AI_SSL_CRT_FILE=/etc/dai/cert.pem \
-e DRIVERLESS_AI_AUTHENTICATION_METHOD=tls_certificate \
-e DRIVERLESS_AI_SSL_CLIENT_VERIFY_MODE=CERT_REQUIRED \
-e DRIVERLESS_AI_SSL_CA_FILE=/etc/dai/rootCA.pem \
-e DRIVERLESS_AI_SSL_CLIENT_KEY_FILE=/etc/dai/client_config_key.key \
-e DRIVERLESS_AI_SSL_CLIENT_CRT_FILE=/etc/dai/client_config_cert.pem \
-v /user/log:/log \
-v /user/tmp:/tmp \
-v /user/certificates/server_config_key.pem:/etc/dai/private_key.pem \
-v /user/certificates/server_config_cert.pem:/etc/dai/cert.pem \
-v /user/certificates/client_config_cert.pem:/etc/dai/client_config_cert.pem \
-v /user/certificates/client_config_key.key:/etc/dai/client_config_key.key \
-v /user/certificates/rootCA.pem:/etc/dai/rootCA.pem \
harbor.h2o.ai/h2oai/dai-centos7-x86_64:1.9.0-cudal0.0.23
```

Note: When certificate verification is required, use the Docker parameter `--hostname` to ensure that the certificate hostname is resolvable from within the Docker container to the container's IP address.

Native installs include DEBs, RPMs, and TAR SH installs. The example below shows how to edit the config.toml file to enable mTLS authentication.

1. Export the Driverless AI config.toml file or add it to `~/.bashrc`. For example:

```
# DEB and RPM
export DRIVERLESS_AI_CONFIG_FILE="/etc/dai/config.toml"

# TAR SH
export DRIVERLESS_AI_CONFIG_FILE="/path/to/your/unpacked/dai/directory/config.toml"
```

2. Open the config.toml file and edit the following authentication variables. The config.toml file is available in the `etc/dai` folder after Driverless AI is installed.

```
# Path to the Certification Authority certificate file. This certificate will be
# used when to verify client certificate when client authentication is turned on.
# If this is not set, clients are verified using default system certificates.
#
ssl_ca_file = "/etc/pki/ca.crt"

# Sets the client verification mode.
# CERT_NONE: Client does not need to provide the certificate and if it does any
# verification errors are ignored.
# CERT_OPTIONAL: Client does not need to provide the certificate and if it does
# certificate is verified againsts set up CA chains.
# CERT_REQUIRED: Client needs to provide a certificate and certificate is
# verified.
# You'll need to set 'ssl_client_key_file' and 'ssl_client_crt_file'
# When this mode is selected for Driverless to be able to verify
# it's own callback requests.
#
ssl_client_verify_mode = "CERT_REQUIRED"

# Path to the private key that Driverless will use to authenticate itself when
# CERT_REQUIRED mode is set.
#
ssl_client_key_file = "/etc/pki/dai-self.key"

# Path to the client certificate that Driverless will use to authenticate itself
# when CERT_REQUIRED mode is set.
#
ssl_client_crt_file = "/etc/pki/dai-self.crt"

# Enable client certificate authentication
authentication_method = "tls_certificate"
```

3. Start (or restart) Driverless AI.

16.5 OpenID Connect Authentication Examples

This section describes how to enable OpenID Connect authentication in Driverless AI. It provides two examples. The first describes how to enable OpenID connect and log in to the Driverless AI UI. The second describes additional token-based authentication settings, which allows you to run the Driverless AI Python client. (Note that token-based authentication is not yet supported on the Driverless AI R client.) This section assumes that you have an understanding of [OpenID Connect](#).

16.5.1 The OpenID Connect Protocol

OpenID Connect follows a distinct protocol during the authentication process:

1. A request is sent from the client (RP) to the OpenID provider (OP).
2. The OP authenticates the end user and obtains authorization.
3. The OP responds with an ID Token. (An Access Token is usually provided as well.)
4. The Relying Party (RP) can send a request with the Access Token to the UserInfo Endpoint.
5. The UserInfo Endpoint returns Claims about the End User.

Refer to the OpenID Connect Basic Client Implementer's Guide for more information: https://openid.net/specs/openid-connect-basic-1_0.html

16.5.2 Understanding the Well-Known Endpoint

In order to begin the process of configuring Driverless AI for OpenID-based authentication, the end user must retrieve OpenID Connect metadata about their authorization server by requesting information from the **well-known endpoint**. This information is subsequently used to configure further interactions with the provider.

The well-known endpoint is typically configured as follows:

```
https://yourOpenIDProviderHostname/.well-known/openid-configuration
```

16.5.3 Configuration Options

OpenID Configuration Options

The following options in the config.toml file are used for enabling OpenID-based authentication. Setting these options allow you to log in to the Driverless AI UI using OpenID.

```
# The OpenID server URL. (Ex: https://oidp.ourdomain.com) Do not end with a "/"
auth_openid_provider_base_url= "https://yourOpenIDProviderHostname"

# The uri to pull OpenID config data from. (You can extract most of required OpenID config from this URL.)
# Usually located at: /auth/realm/master/.well-known/openid-configuration

# Quote method from urlib.parse used to encode payload dict in Authentication Request
auth_openid_urllib_quote_via="quote"

# These endpoints are made available by the well-known endpoint of the OpenID provider
# All endpoints should start with a "/"
auth_openid_auth_uri=""
auth_openid_token_uri=""
auth_openid_userinfo_uri=""
auth_openid_logout_uri=""

# In most cases, these values are usually 'code' and 'authorization_code' (as shown below)
# Supported values for response_type and grant_type are listed in the response of well-known endpoint
auth_openid_response_type="code"
auth_openid_grant_type="authorization_code"

# Scope values--supported values are available in the response from the well-known endpoint
# 'openid' is required
# Additional scopes may be necessary if the response to the userinfo request
# does not include enough information to use for authentication
# Separate additional scopes with a blank space.
# See https://openid.net/specs/openid-connect-basic-1_0.html#Scopes for more info
auth_openid_scope="openid"

# The OpenID client details that are available from the provider
# A new client for Driverless AI in your OpenID provider must be created if one does not already exist
auth_openid_client_id=""
auth_openid_client_secret=""

# Sample redirect value: http[s]://driverlessai-server-address:port/openid/callback
# Ensure that the client configuration in the OpenID provider (see previous step) includes
# this exact URL as one of the possible redirect URLs for the client
# If these do not match, the OpenID connection will fail
auth_openid_redirect_uri=""

# Token endpoint response key configs
auth_openid_access_token_expires_in="expires_in"
auth_openid_refresh_token_expires_in="refresh_expires_in"

# UserInfo response key configs for all users who log in to Driverless AI
# The userinfo_auth_key and userinfo_auth_value are
# a key value combination in the userinfo response that remain static for everyone
# If this key value pair does not exist in the user_info response,
# then the Authentication is considered failed
auth_openid_userinfo_auth_key=""
auth_openid_userinfo_auth_value=""

# Key that specifies username in user_info json (we will use value of this key as username in Driverless AI)
auth_openid_userinfo_username_key=""

# Enable advanced matching for OpenID authentication
# When enabled, the ObjectPath expression is used to evaluate the user's identity
# Disabled by default
# For more information, refer to http://objectpath.org/
auth_openid_use_objectpath_match=false

# Set the ObjectPath expression
# Used to evaluate whether a user is allowed to login to Driverless AI
# The user is allowed to log in when the expression evaluates to True
```

(continues on next page)

(continued from previous page)

```
# Examples:  
# $our_claim is "our_value" (simple claim equality)  
# "expected_role" in @.roles (list of claims contains required value)  
auth_openid_use_objectpath_expression=""
```

Token-Based Authentication Configuration Options

The following additional options in the config.toml file are used for enabling token-based authentication. Token-based authentication allows clients to authenticate with the Driverless AI server by providing a token with each request. This is targeted for (but not limited to) the environments with OpenID Connect authentication. If these options are not set, then clients are not able to authenticate with the server when OpenID Connect is configured as the authentication method.

```
# Sets token introspection URL for OpenID Connect authentication.(needs to be an absolute URL)  
auth_openid_token_introspection_url = ""  
  
# Enables option to use Bearer token for authentication with the RPC endpoint.  
api_token_introspection_enabled = false  
  
# Sets the method that is used to introspect the bearer token.  
# OAUTH2_TOKEN_INTROSPECTION: Uses OAuth 2.0 Token Introspection (RPC 7662)  
# endpoint to introspect the bearer token.  
# This useful when 'openid' is used as the authentication method.  
# Uses 'auth_openid_client_id' and 'auth_openid_client_secret' and to  
# authenticate with the authorization server and  
# 'auth_openid_token_introspection_url' to perform the introspection.  
#  
api_token_introspection_method = "OAUTH2_TOKEN_INTROSPECTION"  
  
# Sets the minimum of the scopes that the access token needs to have  
# in order to pass the introspection. Space separated./  
# This is passed to the introspection endpoint and also verified after response  
# for the servers that don't enforce scopes.  
# Keeping this empty turns any the verification off.  
#  
api_token_oauth2_scopes = ""  
  
# Which field of the response returned by the token introspection endpoint should be used as a username.  
api_token_oauth2_username_field_name = "username"  
  
# Enables the option to initiate a PKCE flow from the UI in order to obtain tokens usable with Driverless clients  
oauth2_client_tokens_enabled = false  
  
# Sets up client id that will be used in the OAuth 2.0 Authorization Code Flow to obtain the tokens. Client needs to be public and be able to use PKCE  
# with S256 code challenge.  
oauth2_client_tokens_client_id = ""  
  
# Sets up the absolute url to the authorize endpoint.  
oauth2_client_tokens_authorize_url = ""  
  
# Sets up the absolute url to the token endpoint.  
oauth2_client_tokens_token_url = ""  
  
# Sets up the absolute url to the token introspection endpoint. It's displayed in the UI so that clients can inspect the token expiration.  
oauth2_client_tokens_introspection_url = ""  
  
# Sets up the absolute to the redirect url where Driverless handles the redirect part of the Authorization Code Flow. this <Driverless base url>/oauth2/  
# client_token  
oauth2_client_tokens_redirect_url = ""  
  
# Sets up the scope for the requested tokens. Space separated list.  
oauth2_client_tokens_scope = "openid profile ai.h2o.storage"
```

16.5.4 Example 1: Enabling OpenID Connect

This example describes how to start Driverless AI in the Docker image and with native installs after OpenID has been configured. Note that this example does not enable tokens, so the Driverless AI Python client will be incompatible with this installation.

Docker Image Installs

Native Installs

1. Edit the OpenID configuration options in your config.toml file as described in the [Configuration Options](#) section.
2. Mount the edited config.toml file into the Docker container.

```
nvidia-docker run \
--net=openid-network \
--name="dai-with-openid" \
--pid=host \
--init \
--rm \
--shm-size=256m \
-u `id -u`: `id -g` \
-p 12345:12345 \
-v "/pwd:/DAI_DATA/data":/data \
-v "/pwd:/DAI_DATA/log":/log \
-v "/pwd:/DAI_DATA/license":/license \
-v "/pwd:/DAI_DATA/tmp":/tmp \
-v "/pwd:/DAI_DATA/config":/config \
-e DRIVERLESS_AI_CONFIG_FILE="/config/config.toml" \
harbor.h2o.ai/h2oai/dai-centos7-x86_64:1.9.0-cuda10.0.23
```

The next step is to launch and log in to Driverless AI. Refer to [Logging in to Driverless AI](#).

1. Export the Driverless AI config.toml file or add it to `~/bashrc`. For example:

```
# DEB and RPM
export DRIVERLESS_AI_CONFIG_FILE="/etc/dai/config.toml"

# TAR SH
export DRIVERLESS_AI_CONFIG_FILE="/path/to/your/unpacked/dai/directory/config.toml"
```

2. Edit the OpenID configuration properties in the config.toml file as described in the [Configuration Options](#) section.

3. Start (or restart) Driverless AI.

The next step is to launch and log in to Driverless AI. Refer to [Logging in to Driverless AI](#).

Example 2: Enabling Token-based Authentication with OpenID Connect

Similar to Example 1, this example describes how to start Driverless AI in the Docker image and with native installs after OpenID has been configured. It also enables tokens for compatibility with the Driverless AI Python client.

Docker Image Installs

Native Installs

1. Edit the OpenID configuration options in your config.toml file as described in the [Configuration Options](#) section. Be sure to also enable the token-based authentication options described in the [Token-Based Authentication Configuration Options](#) options section.
2. Mount the edited config.toml file into the Docker container.

```
nvidia-docker run \
--net=openid-network \
--name="dai-with-openid" \
--pid=host \
--init \
--rm \
--shm-size=256m \
-u `id -u`: `id -g` \
-p 12345:12345 \
-v "/pwd:/DAI_DATA/data":/data \
-v "/pwd:/DAI_DATA/log":/log \
-v "/pwd:/DAI_DATA/license":/license \
-v "/pwd:/DAI_DATA/tmp":/tmp \
-v "/pwd:/DAI_DATA/config":/config \
-e DRIVERLESS_AI_CONFIG_FILE="/config/config.toml" \
harbor.h2o.ai/h2oai/dai-centos7-x86_64:1.9.0-cuda10.0.23
```

The next step is to launch and log in to Driverless AI. Refer to [Logging in to Driverless AI](#).

1. Export the Driverless AI config.toml file or add it to `~/bashrc`. For example:

```
# DEB and RPM
export DRIVERLESS_AI_CONFIG_FILE="/etc/dai/config.toml"

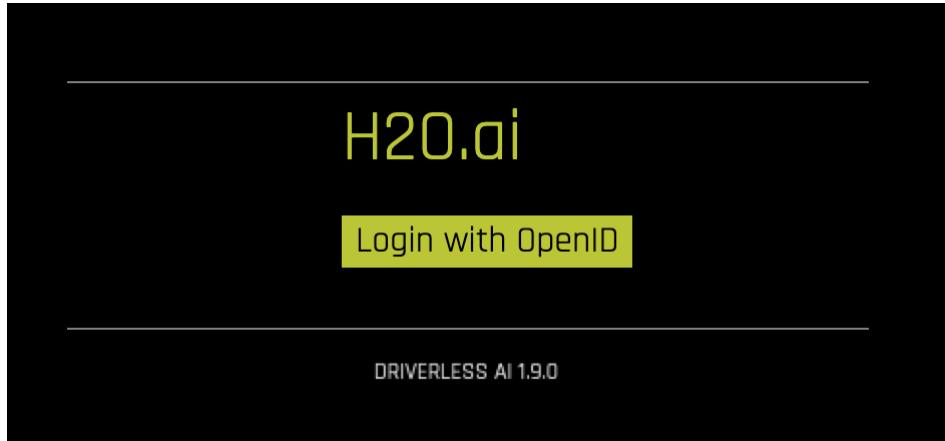
# TAR SH
export DRIVERLESS_AI_CONFIG_FILE="/path/to/your/unpacked/dai/directory/config.toml"
```

2. Edit the OpenID configuration properties in the config.toml file as described in the [Configuration Options](#) section. Be sure to also enable the token-based authentication options described in the [Token-Based Authentication Configuration Options](#) options section.
3. Start (or restart) Driverless AI.

The next step is to launch and log in to Driverless AI. Refer to [Logging in to Driverless AI](#).

16.5.5 Logging in to Driverless AI

Open a browser and launch Driverless AI. Notice the you will be prompted to log in with OpenID.



16.6 PAM Authentication Example

The following sections describe how to enable [Pluggable Authentication Modules \(PAM\)](#) in Driverless AI. You can do this by specifying environment variables in the Docker image or by updating the config.toml file.

Note: This assumes that the user has an understanding of how to grant permissions in their own environment in order for PAM to work. Specifically for Driverless AI, be sure that the Driverless AI processes owner has access to **/etc/shadow** (without root); otherwise authentication will fail.

Docker Image Installs

Native Installs

Note: The following instructions are only applicable with a CentOS 7 host.

In this example, the host Linux system has PAM enabled for authentication and Docker running on that Linux system. The goal is to enable PAM for Driverless AI authentication while the Linux system hosts the user information.

1. Verify that the username (“eric” in this case) is defined in the Linux system.

```
[root@Linux-Server]# cat /etc/shadow | grep eric
eric:$6$inOv3GsQuRanR1H4$kYgys3cc2dQ3u9it02WtvAYqiGiQgQ/yq0ios.g4F9DM1UJGpruUVoG15G6OD3MrX/3uy4gWf1YJnbJofaAni/:0:99999:7:::
```

2. Start Docker on the Linux Server and enable PAM in Driverless AI.

```
[root@Linux-Server]# docker run \
--rm \
--shm-size=256m \
-u `id -u:`id -g` \
-p 12345:12345 \
-v `pwd`:/config:/config \
-v `pwd`:/data:/data \
-v `pwd`:/log:/log \
-v `pwd`:/license:/license \
-v `pwd`:/tmp:/tmp \
-v /etc/passwd:/etc/passwd \
-v /etc/shadow:/etc/shadow \
-v /etc/pam.d/:/etc/pam.d/ \
-e DRIVERLESS_AI_AUTHENTICATION_METHOD="pam" \
harbor.h2o.ai/h2oai/dai-centos7-x86_64:1.9.0-cuda10.0.23
```

- Obtain the Driverless AI container ID. This ID is required for the next step and will be different every time Driverless AI is started.

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	PORTS
8e333475ffd8	opsh2oai/h2oai-runtime	"/run.sh"	36 seconds ago	Up 35 seconds	192.168.0.1:9090->9090/tcp, 192.168.0.1:12345->12345/tcp
168.0.1:12345->12345/tcp			clever_swirles		

- From the Linux Server, verify that the Docker Driverless AI instance can see the shadow file. The example below references 8e333475ffd8, which is the container ID obtained in the previous step.

```
[root@Linux-Server]# docker exec 8e333475ffd8 cat /etc/shadow|grep eric
eric:$6$in0v3GsQuRanR1H4$KYgys3oc2dQ3u9it02WTvAYqiGiQgQ/yq0ios.g4F9DM1UJGpruUVoG15G6OD3MrX/3uy4Wf1YJnbJofaAni:::0:99999:7:::
```

- Open a Web browser and navigate to port 12345 on the Linux system that is running the Driverless AI Docker Image. Log in with credentials known to the Linux system. The login information will now be validated using PAM.

In this example, the host Linux system has PAM enabled for authentication. The goal is to enable PAM for Driverless AI authentication while the Linux system hosts the user information.

This example shows how to edit the config.toml file to enable PAM. The config.toml file is available in the etc/dai folder after the RPM or DEB is installed. Edit the authentication_method variable in this file to enable PAM authentication, and then restart Driverless AI.

- Verify that the username ("eric" in this case) is defined in the Linux system.

```
[root@Linux-Server]# cat /etc/shadow | grep eric
eric:$6$in0v3GsQuRanR1H4$KYgys3oc2dQ3u9it02WTvAYqiGiQgQ/yq0ios.g4F9DM1UJGpruUVoG15G6OD3MrX/3uy4Wf1YJnbJofaAni:::0:99999:7:::
```

- Export the Driverless AI config.toml file or add it to ~/.bashrc. For example:

```
# DEB and RPM
export DRIVERLESS_AI_CONFIG_FILE="/etc/dai/config.toml"

# TAR SH
export DRIVERLESS_AI_CONFIG_FILE="/path/to/your/unpacked/dai/directory/config.toml"
```

- Edit the authentication_method variable in the config.toml file so that PAM is enabled.

```
# authentication_method
# unvalidated : Accepts user id and password, does not validate password
# none : Does not ask for user id or password, authenticated as admin
# pam : Accepts user id and password, Validates user with operating system
# ldap : Accepts user id and password, Validates against an ldap server, look
# local: Accepts a user id and password, Validated against a htpasswd file provided in local_htpasswd_file
# for additional settings under LDAP settings
authentication_method = "pam"
```

- Start Driverless AI. Note that the command used to start Driverless AI varies depending on your install type.

```
# Linux RPM or DEB with systemd
[root@Linux-Server]# sudo systemctl start dai

# Linux RPM or DEB without systemd
[root@Linux-Server]# sudo -H -u dai /opt/h2oai/dai/run-dai.sh

# Linux TAR SH
[root@Linux-Server]# ./run-dai.sh
```

5. Open a Web browser and navigate to port 12345 on the Linux system that is running Driverless AI. Log in with credentials known to the Linux system (as verified in the first step). The login information will now be validated using PAM.

ENABLING NOTIFICATIONS

Driverless AI can be configured to trigger a user-defined script at the beginning and end of an experiment. This functionality can be used to send notifications to services like Slack or to trigger a machine shutdown.

The **config.toml** file exposes the following variables:

- `listeners_experiment_start`: Registers an absolute location of a script that gets executed at the start of an experiment.
- `listeners_experiment_done`: Registers an absolute location of a script that gets executed when an experiment is finished successfully.

Driverless AI accepts any executable as a script. (For example, a script can be implemented in Bash or Python.) There are only two requirements:

- The specified script can be executed. (i.e., The file has executable flag.)
- The script should be able to accept command line parameters.

17.1 Script Interfaces

When Driverless AI executes a script, it passes the following parameters as a script command line:

- Application ID: A unique identifier of a running Driverless AI instance.
- User ID: The identification of the user who is running the experiment.
- Experiment ID: A unique identifier of the experiment.
- Experiment Path: The location of the experiment results.

17.2 Example

The following example demonstrates how to use notification scripts to shutdown an EC2 machine that is running Driverless AI after all launched experiments are finished. The example shows how to use a notification script in a Docker container and with native installations. The idea of a notification script is to create a simple counter (i.e., number of files in a directory) that counts the number of running experiments. If counter reaches 0-value, then the specified action is performed.

In this example, we use the AWS command line utility to shut down the actual machine; however, the same functionality can be achieved by executing `sudo poweroff` (if the actual user has password-less sudo capability configured) or `poweroff` (if the script `poweroff` has setuid bit set up together with executable bit. For more info, please visit: <https://unix.stackexchange.com/questions/85663/poweroff-or-reboot-as-normal-user>.)

17.2.1 The on_start Script

This script increases the counter of running experiments.

```
#!/usr/bin/env bash
app_id="${1}"
experiment_id="${3}"
tmp_dir="${TMDIR:-/tmp}/${app_id}"
exp_file="${tmp_dir}/${experiment_id}"

mkdir -p "${tmp_dir}"
touch "${exp_file}"
```

17.2.2 The on_done Script

This script decreases the counter and executes machine shutdown when the counter reaches 0-value.

```
#!/usr/bin/env bash
app_id="${1}"
experiment_id="${3}"
tmp_dir="${TMDIR:-/tmp}/${app_id}"
exp_file="${tmp_dir}/${experiment_id}"

if [ -f "${exp_file}" ]; then
    rm -f "${exp_file}"
fi

running_experiments=$(ls -l "${tmp_dir}" | wc -l)

if [ "${running_experiments}" -gt 0 ]; then
    echo "There is still ${running_experiments} running experiments!"
else
    echo "No experiments running! Machine is going to shutdown!"
# Use instance meta-data API to get instance ID and then use AWS CLI to shutdown the machine
# This expects, that AWS CLI is properly configured and has capability to shutdown instances enabled.
aws ec2 stop-instances --instance-ids $(curl http://169.254.169.254/latest/meta-data/instance-id)
fi
```

Docker Image Installs

Native Installs

1. Copy the config.toml file from inside the Docker image to your local filesystem. (Change nvidia-docker run to docker run for non-GPU environments.)

```
# In your Driverless AI folder (for example, dai_1.5.1),
# make config and scripts directories
mkdir config
mkdir scripts

# Copy the config.toml file to the new config directory.
nvidia-docker run \
    --pid=host \
    --rm \
    --init \
    -u `id -u`:`id -g` \
    -v `pwd`/config:/config \
    --entrypoint bash \
    h2oai/dai-centos7-x86_64:TAG
    -c "cp /etc/dai/config.toml /config"
```

2. Edit the **Notification scripts** section in the **config.toml** file and save your changes. Note that in this example, the scripts are saved to a **dai_VERSION/scripts** folder.

```
# Notification scripts
# - the variable points to a location of script which is executed at given event in experiment lifecycle
# - the script should have executable flag enabled
# - use of absolute path is suggested
# The on experiment start notification script location
listeners_experiment_start = "dai_VERSION/scripts/on_start.sh"
# The on experiment finished notification script location
listeners_experiment_done = "dai_VERSION/scripts/on_done.sh"
```

3. Start Driverless AI with the DRIVERLESS_AI_CONFIG_FILE environment variable. Make sure this points to the location of the edited config.toml file so that the software finds the configuration file. (Change nvidia-docker run to docker run for non-GPU environments.)

```
nvidia-docker run \
--pid=host \
--init \
--rm \
-u `id -u`:`id -g` \
-e DRIVERLESS_AI_CONFIG_FILE="/config/config.toml" \
-v `pwd`/config:/config \
-v `pwd`/data:/data \
-v `pwd`/log:/log \
-v `pwd`/license:/license \
-v `pwd`/tmp:/tmp \
-v `pwd`/scripts:/scripts \
harbor.h2o.ai/h2oai/dai-centos7-x86_64:1.9.0-cudal0.0.23
```

1. Export the Driverless AI **config.toml** file or add it to `~/.bashrc`. For example:

```
# DEB and RPM
export DRIVERLESS_AI_CONFIG_FILE="/etc/dai/config.toml"

# TAR SH
export DRIVERLESS_AI_CONFIG_FILE="/path/to/your/unpacked/dai/directory/config.toml"
```

2. Edit the **Notification scripts** section in the **config.toml** file to point to the new scripts. Save your changes when you are done.

```
# Notification scripts
# - the variable points to a location of script which is executed at given event in experiment lifecycle
# - the script should have executable flag enabled
# - use of absolute path is suggested
# The on experiment start notification script location
listeners_experiment_start = "/opt/h2oai/dai/scripts/on_start.sh"
# The on experiment finished notification script location
listeners_experiment_done = "/opt/h2oai/dai/scripts/on_done.sh"
```

3. Start Driverless AI. Note that the command used to start Driverless AI varies depending on your install type.

```
# Deb or RPM with systemd (preferred for Deb and RPM):
# Start Driverless AI.
sudo systemctl start dai

# Deb or RPM without systemd:
# Start Driverless AI.
sudo -H -u dai /opt/h2oai/dai/run-dai.sh

# Tar.sh
# Start Driverless AI
./run-dai.sh
```

CHAPTER
EIGHTEEN

EXPORT ARTIFACTS

In some cases, you might find that you do not want your users to download artifacts directly to their machines. Driverless AI provides several configuration options/environment variables that enable exporting of artifacts instead of downloading. Artifacts can be exported to a file system directory or an Amazon S3 bucket.

Note: The option to download artifacts is automatically disabled when exporting is enabled.

18.1 Enabling Artifact Exports

The config.toml file exposes the following variables:

- `enable_artifacts_upload`: Replace all the downloads on the experiment page to exports, and allow users to push to the artifact store configured with `artifacts_store`. This is disabled by default.
- `artifacts_store`: Specify one of the following storage methods:
 - `file_system`: Store artifacts in the file system directory specified by the `artifacts_file_system_directory` setting.
 - `S3`: Store artifacts in the S3 bucket specified by the `artifacts_s3_bucket` setting.

Specify the following for the storage method you selected:

File System Directory

- `artifacts_file_system_directory`: The file system location where artifacts will be copied. This is expected to be a directory on your server.

AWS S3

- `artifacts_s3_bucket`: The AWS S3 bucket where artifacts will be stored.

Note: The option to disable artifact downloads does not extend to datasets. Whether users can download datasets is controlled by the `enable_dataset_downloading` configuration option, which is set to `true` by default. Set this to `false` if you do not want users to download datasets to their local machine. There is currently no configuration option that enables exporting datasets to a file system.

Docker Image Installs

Native Installs

The following example shows how to enable artifact exporting to a file system when starting the Driverless AI Docker image.

```
docker run \
--pid=host \
--init \
--rm \
--shm-size=256m \
-e DRIVERLESS_AT_ENABLE_ARTIFACTS_UPLOAD="true" \
```

(continues on next page)

(continued from previous page)

```
-e DRIVERLESS_AI_ARTIFACTS_STORE="file_system" \
-e DRIVERLESS_AI_ARTIFACTS_FILE_SYSTEM_DIRECTORY="tmp" \
-e id -u: id -g` \
-p 12345:12345 \
-v `pwd`/data:/data \
-v `pwd`/log:/log \
-v `pwd`/license:/license \
-v `pwd`/tmp:/tmp \
harbor.h2o.ai/h2oai/dai-centos7-x86_64:1.9.0-cuda10.0.23
```

The following example shows how to enable artifact exporting to a file system on native installs.

1. Export the Driverless AI config.toml file or add it to `~/.bashrc`. For example:

```
# DEB and RPM
export DRIVERLESS_AI_CONFIG_FILE="/etc/dai/config.toml"

# TAR SH
export DRIVERLESS_AI_CONFIG_FILE="/path/to/your/unpacked/dai/directory/config.toml"
```

2. Edit the following configuration option in the config.toml file. Save your changes when you are done.

```
# Replace all the downloads on the experiment page to exports and allow users to push to the artifact store configured with artifacts_store
enable_artifacts_upload = true

# Artifacts store.
# file_system: stores artifacts on a file system directory denoted by artifacts_file_system_directory.
#
artifacts_store = "file_system"

# File system location where artifacts will be copied in case artifacts_store is set to file_system
artifacts_file_system_directory = "tmp"
```

3. Start Driverless AI. Note that the command used to start Driverless AI varies depending on your install type.

```
# Deb or RPM with systemd (preferred for Deb and RPM):
# Start Driverless AI.
sudo systemctl start dai

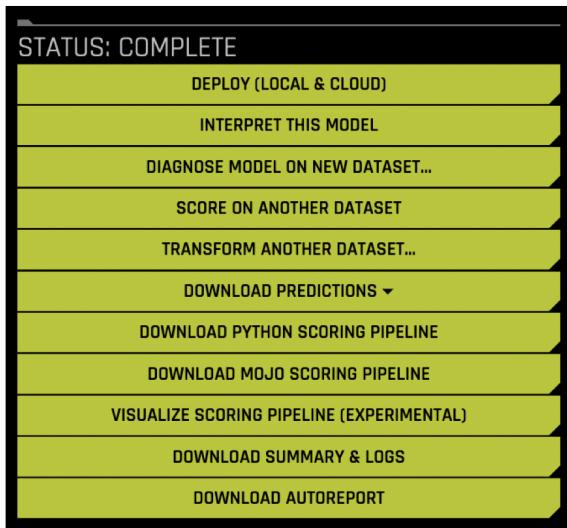
# Deb or RPM without systemd:
# Start Driverless AI.
sudo -H -u dai /opt/h2oai/dai/run-dai.sh

# Tar.sh
# Start Driverless AI
./run-dai.sh
```

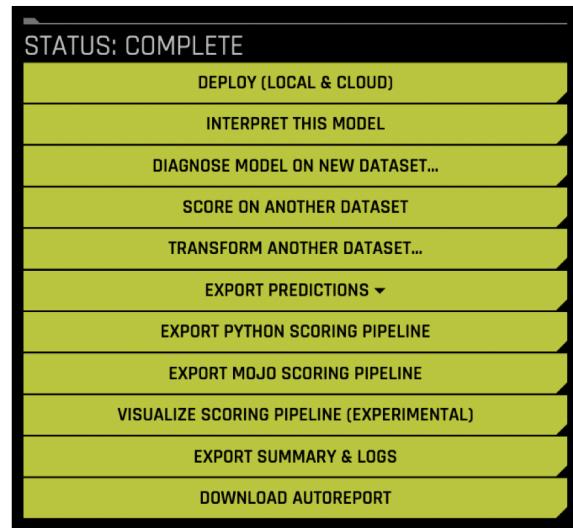
18.2 Exporting an Artifact

When the export artifacts options are enabled/configured, the menu options on the *Completed Experiment* page will change. Specifically, all “Download” options (with the exception of Autoreport) will change to “Export.”

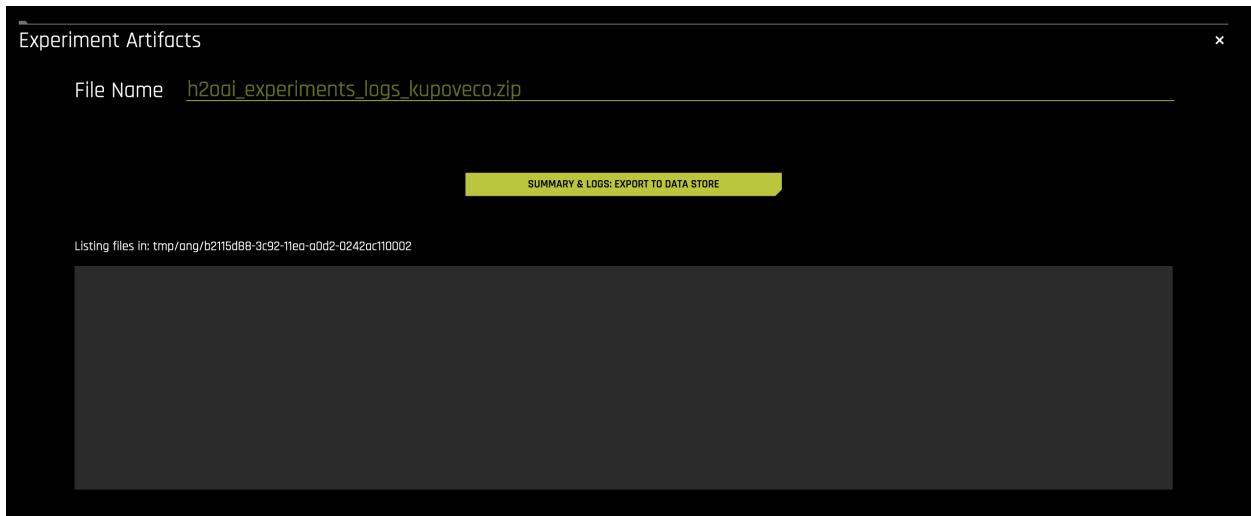
Driverless AI completed experiment menu with default configuration



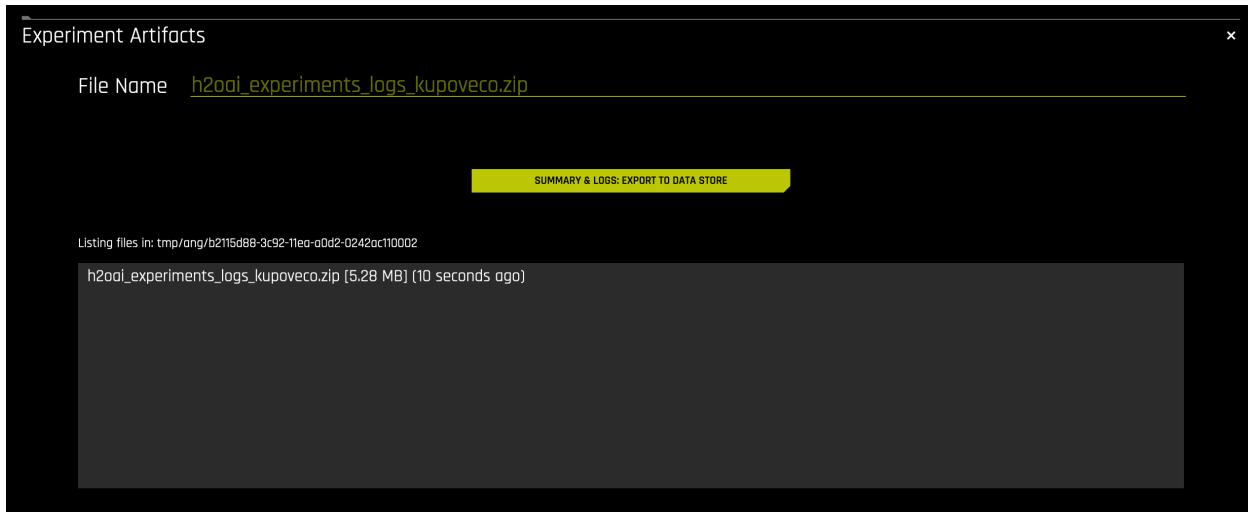
Driverless AI completed experiment menu with artifact exporting configured



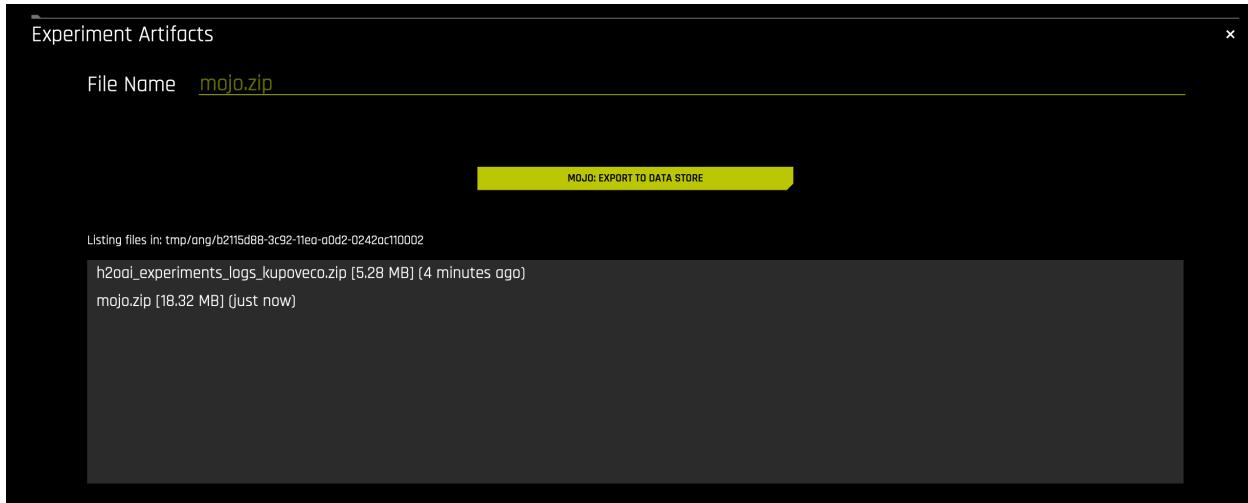
1. Click on an artifact to begin exporting. For example, click on **Export Summary and Logs**.



2. Specify a file name or use the default file name. This denotes the new name to be given to the exported artifact. By default, this name matches the selected export artifact name.
3. Now click the **Summary and Logs: Export to Data Store** button. (Note that this button name changes depending on the artifact that you select.) This begins the export action. Upon completion, the exported artifact will display in the list of artifacts. The directory structure is: <path_to_export_to>/<user>/<experiment_id>/



4. Continue exporting additional artifacts for this experiment.



CHANGING THE LANGUAGE IN THE UI

The Driverless AI UI is available in English (default), Japanese, Chinese (Simplified), and Korean. This section describes how you can use the `app_language` config setting/environment variable to change the language of the UI before starting Driverless AI.

When using `app_language`, the following options can be specified:

- en: English (default)
- ja: Japanese
- cn: Chinese (Simplified)
- ko: Korean

19.1 Examples

The following examples show how to change the app language from English to Japanese.

Docker Image Installs

Docker Image with the config.toml

Native Installs

To change the application language in Docker images, specify the `APP_LANGUAGE` environment variable. Note that this variable must be prepended with `DRIVERLESS_AI_`. Replace `nvidia-docker` with `docker` in the example below if necessary.

```
nvidia-docker run \
--pid=host \
--init \
--rm \
--shm-size=256m \
-p 12345:12345 \
-u `id -u`:`id -g` \
-e DRIVERLESS_AI_ENABLED_FILE_SYSTEMS="file,s3,hdfs" \
-e DRIVERLESS_AI_APP_LANGUAGE="ja" \
-v `pwd`/data:/data \
-v `pwd`/log:/log \
-v `pwd`/license:/license \
-v `pwd`/tmp:/tmp \
harbor.h2o.ai/h2cai/dai-centos7-x86_64:1.9.0-cudal0.0.23
```

This example shows how to configure Minio options in the config.toml file, and then specify that file when starting Driverless AI in Docker.

1. Configure the Driverless AI config.toml file. Set the following configuration option.
 - `app_language="ja"`
2. Mount the config.toml file into the Docker container. Replace `nvidia-docker` with `docker` if necessary.

```
nvidia-docker run \
--pid=host \
--init \
--rm \
--shm-size=256m \
--add-host name.node:172.16.2.186 \
-e DRIVERLESS_AI_CONFIG_FILE=/path/in/docker/config.toml \
-p 12345:12345 \
-v /local:/path/to/config.toml:/path/in/docker/config.toml \
-v /etc/passwd:/etc/passwd:ro \
-v /etc/group:/etc/group:ro \
-v /tmp/dtmp:/tmp \
-v /tmp/dlog/:/log \
-v /tmp/dlicense/:/license \
-v /tmp/ddata/:/data \
-u $(id -u):$(id -g) \
harbor.h2o.ai/h2oai/dai-centos7-x86_64:1.9.0-cuda10.0.23
```

Native installs include DEBs, RPMs, and TAR SH installs. The example below shows how to use the app_language configuration option in the config.toml file to change the language to Japanese.

1. Export the Driverless AI config.toml file or add it to ~/.bashrc. For example:

```
# DEB and RPM
export DRIVERLESS_AI_CONFIG_FILE="/etc/dai/config.toml"

# TAR SH
export DRIVERLESS_AI_CONFIG_FILE="/path/to/your/unpacked/dai/directory/config.toml"
```

2. Open the config.toml file and edit the app_language variable. The config.toml file is available in the etc/dai folder after the RPM or DEB is installed.

```
# Default application language - options are 'en', 'ja', 'cn', 'ko'
app_language = "ja"
```

3. Start (or restart) Driverless AI. Note that the command used to start Driverless AI varies depending on your install type.

```
# Linux RPM or DEB with systemctl
sudo systemctl start dai

# Linux RPM or DEB without systemctl
sudo -H -u dai /opt/h2oai/dai/run-dai.sh

# Linux TAR SH
./run-dai.sh
```

The screenshot shows the H2O.ai interface for managing datasets. At the top, there's a header with the H2O.ai logo and navigation links like 'Project', 'データセット', '視覚化', 'Experiment', '診断', 'MLI', 'デプロイ', 'リソース▼', 'メッセージ[2]', and 'ログアウト'. Below the header, a sub-header reads 'DRIVERLESS AI 1.9.0 - AI to do AI' and 'ライセンス先: H2O.ai (SN37), ユーザー名 - ANG'. The main area is titled 'データセット' and displays a table of datasets. The table has columns: '名前' (Name), 'パス' (Path), 'サイズ' (Size), '行' (Rows), '列' (Columns), and 'Status' (Status). The datasets listed are:

名前	パス	サイズ	行	列	Status
begebase	...egebase.1589916860.6277387.bin	3KB	45	5	[クリックして操作]
bimedapu	...imedapu.1589916860.6221528.bin	5KB	105	5	[クリックして操作]
iris-getting-started	...ris.csv.1589913686.6236985.bin	8KB	150	5	[クリックして操作]
iris-getting-started	...ris.csv.1589913344.3161323.bin	8KB	150	5	[クリックして操作]
iris-getting-started	...iris.csv.1589912711.133594.bin	8KB	150	5	[クリックして操作]
iris-getting-started	...ris.csv.1589912686.7015603.bin	8KB	150	5	[クリックして操作]
coronavirusdataset.zip	...oset.zip.1583498980.617256.bin	458KB	6K	14	[クリックして操作]
titanic_test	...ic_test.1583179223.6774807.bin	37KB	328	14	[クリックして操作]
titanic_train	...c_train.1583179223.6743522.bin	109KB	981	14	[クリックして操作]
titanic.csv	...nic.csv.1583179018.9252644.bin	145KB	1K	14	[クリックして操作]

At the bottom of the page, there are navigation links '1 2 次のページ' and a copyright notice '© 2017-2020 H2O.ai. All rights reserved.'

CHAPTER
TWENTY

LAUNCHING DRIVERLESS AI

Driverless AI is tested on Chrome and Firefox but is supported on all major browsers. For the best user experience, we recommend using Chrome.

1. After Driverless AI is installed and started, open a browser and navigate to <server>:12345.
2. The first time you log in to Driverless AI, you will be prompted to read and accept the Evaluation Agreement. You must accept the terms before continuing. Review the agreement, then click **I agree to these terms** to continue.
3. Log in by entering unique credentials. For example:

Username: h2oai **Password:** h2oai

Note that these credentials do not restrict access to Driverless AI; they are used to tie experiments to users. If you log in with different credentials, for example, then you will not see any previously run experiments.

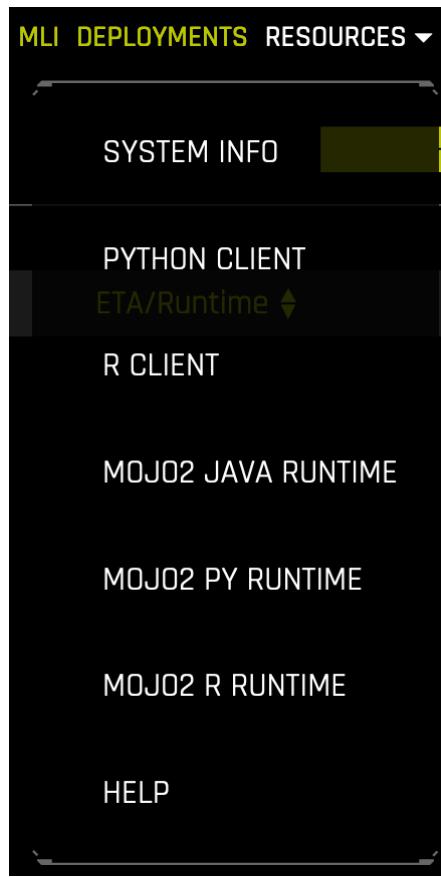
4. As with accepting the Evaluation Agreement, the first time you log in, you will be prompted to enter your License Key. Click the **Enter License** button, then paste the License Key into the **License Key** entry field. Click **Save** to continue. This license key will be saved in the host machine's **/license** folder.

Note: Contact sales@h2o.ai for information on how to purchase a Driverless AI license.

Upon successful completion, you will be ready to add datasets and run experiments.

The screenshot shows the H2O.ai Driverless AI interface. At the top, there is a navigation bar with links for PROJECTS, DATASETS, AUTOVIZ, EXPERIMENTS, DIAGNOSTICS, MLI, DEPLOYMENTS, and RESOURCES. To the right of these are links for MESSAGES [2], LOGOUT, and a button for ADD DATASET (OR DRAG & DROP). Below the navigation bar, the page title is "Datasets". A header row contains columns for Name, Path, Size, Rows, Columns, and Status, each with a downward arrow indicating they are sortable. A message below the header states: "No datasets available. Import or drag supported file type from your file browser (mouseover for list of supported types)". At the bottom of the page, there is a copyright notice: "© 2017-2020 H2O.ai. All rights reserved."

20.1 Resources



The **Resources** dropdown menu allows you to view System Information and access the Driverless AI User Guide.

- System Info (**Note:** The Workers Overview tab is only available if Driverless AI was started in multinode.)
- Help: A link to this Driverless AI User Guide



From the Resources dropdown menu, you can also download the following:

- Python Client (See [The Python Client](#))
- R Client (See [The R Client](#))
- MOJO2 Java Runtime (See [Driverless AI MOJO Scoring Pipeline - Java Runtime](#))
- MOJO2 Python Runtime (See [Driverless AI MOJO Scoring Pipeline - C++ Runtime with Python and R Wrappers](#))
- MOJO2 R Runtime (See [Driverless AI MOJO Scoring Pipeline - C++ Runtime with Python and R Wrappers](#))

20.2 Messages

A **Messages** menu option is available in the top menu when you launch Driverless AI. Click this to view news and upcoming events regarding Driverless AI.

Messages: X

SHOW ALL DISMISS ALL

Announcing Driverless AI Community

Join the Driverless AI Community

H2O.ai is excited to announce the formation of the inaugural community for H2O Driverless AI users. The [Driverless AI Community](#) is open to anyone looking to engage with other users as well as experts from H2O.ai's Driverless AI team. This community engages everyone, from experienced Driverless AI users to AI enthusiasts interested in learning more about how to use the platform. Community members can learn from each other and share ideas, best practices, and use cases of how they are leveraging AI in their organizations. Additionally, the H2O Driverless AI Community members can also engage with Expert Kaggle Grandmasters and the makers of H2O Driverless AI.

This new community provides members the opportunity to interact through an [H2O Community Slack](#) chat workspace, [Driverless AI User Group meetups](#), and events such as H2O AI World London.

The H2O Driverless AI meetup group alone already has over 3,900 members and was the first stepping stone to creating an H2O Driverless AI Community. Joining a Driverless Us Group is easy! Sign up on the Driverless AI Community meetup page and look for your city.

H2O's Driverless AI community on Slack is an open and positive space for data scientists and analysts to discuss the latest in AI and machine learning, help and learn from one another, connect and share. If this sounds like something you want to be part of, we'd love to invite you to join in!

In addition to Driverless AI meetups and community chat on Slack, there are many more resources within the H2O Driverless AI's website; we recommend that you check these out at your leisure:

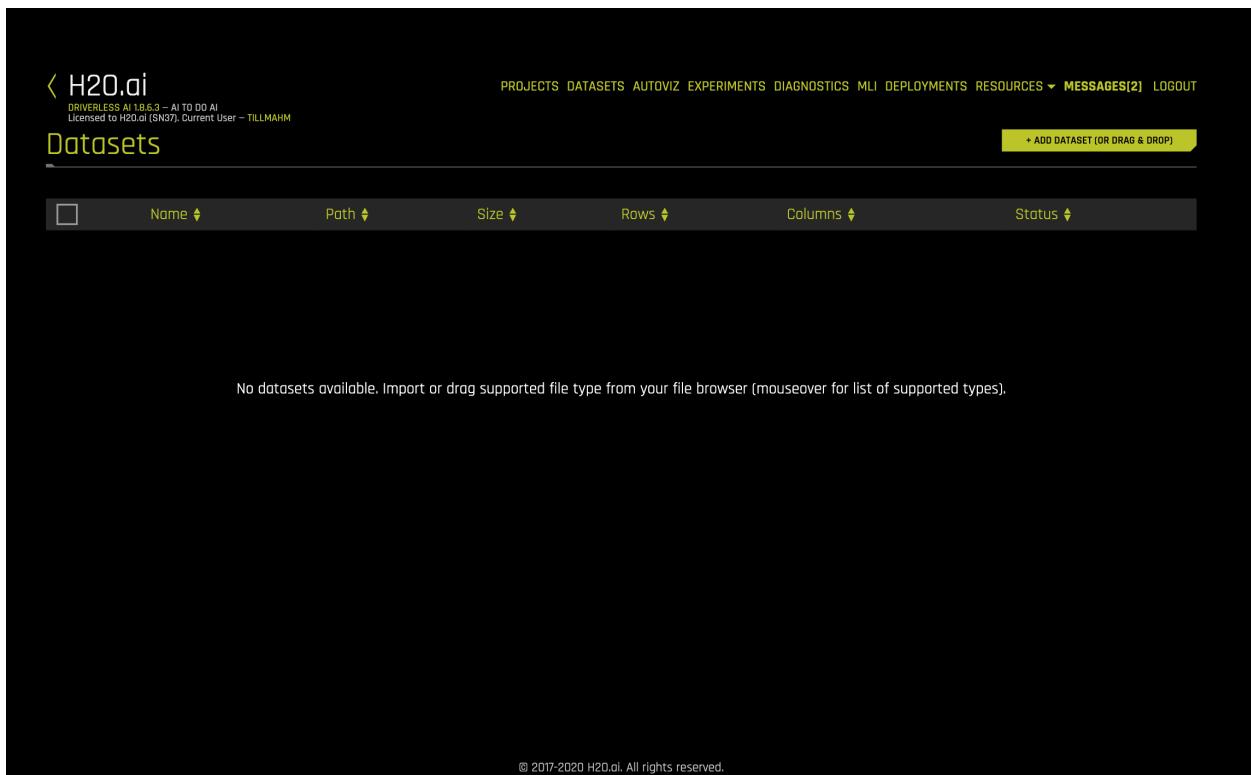
- Blogs: Driverless AI is continuously evolving. For the newest developments, see the [Driverless AI blogs](#).
- Social Media: To stay up to date on the latest improvements and tips from the team that created Driverless AI, follow H2O on [Twitter](#) and [LinkedIn](#).
- Newsletter: Subscribe to the [Driverless newsletter](#).
- Documentation: For in-depth documentation on Driverless AI, see the [H2O.ai docs site](#).

DISMISS X

CHAPTER TWENTYONE

THE DATASETS PAGE

The Datasets Overview page is the Driverless AI Home page. This shows all datasets that have been imported. Note that the first time you log in, this list will be empty.



21.1 Supported File Types

Driverless AI supports the following dataset file formats:

- arff
- bin
- bz2
- csv (See note below)
- dat

- feather
- gz
- jay (See note below)
- orc (See notes below)
- parquet (See notes below)
- pkl
- tgz
- tsv
- txt
- xls
- xlsx
- xz
- zip

Notes:

- CSV in UTF-16 encoding is only supported when implemented with a byte order mark (BOM). If a BOM is not present, the dataset is read as UTF-8.
- For ORC and Parquet file formats, if you select to import multiple files, those files will be imported as multiple datasets. If you select a folder of ORC or Parquet files, the folder will be imported as a single dataset. Tools like Spark/Hive export data as multiple ORC or Parquet files that are stored in a directory with a user-defined name. For example, if you export with `Spark DataFrame.write.parquet("/data/big_parquet_dataset")`, Spark creates a folder **/data/big_parquet_dataset**, which will contain multiple Parquet files (depending on the number of partitions in the input dataset) and metadata. Exporting ORC files produces a similar result.
- For ORC and Parquet file formats, you may receive a “Failed to ingest binary file with ORC / Parquet: lists with structs are not supported” error when ingesting an ORC or Parquet file that has a struct as an element of an array. This is because [PyArrow cannot handle a struct that's an element of an array](#).
- A workaround to flatten Parquet files is provided in Sparkling Water. Refer to [our Sparkling Water solution](#) for more information.
- You can create new datasets from Python script files (custom recipes) by selecting **Data Recipe URL** or **Upload Data Recipe** from the **Add Dataset (or Drag & Drop)** dropdown menu. If you select the **Data Recipe URL** option, the URL must point to either an HTML or raw version of the file, a GitHub repository or tree, or a local file. In addition, you can create a new dataset by modifying an existing dataset with a custom recipe. Refer to `modify_by_recipe` for more information. Datasets created or added from recipes will be saved as .jay files.

21.2 Adding Datasets

You can add datasets using one of the following methods:

Drag and drop files from your local machine directly onto this page. Note that this method currently works for files that are less than 10 GB.

or

Click the **Add Dataset (or Drag & Drop)** button to upload or add a dataset.

Notes:

- Upload File, File System, HDFS, S3, Data Recipe URL, and Upload Data Recipe are enabled by default. These can be disabled by removing them from the `enabled_file_systems` setting in the `config.toml` file. (Refer to [Using the config.toml file](#) section for more information.)
- If File System is disabled, Driverless AI will open a local filebrowser by default.
- If Driverless AI was started with data connectors enabled for Azure Blob Store, BlueData Datatap, Google Big Query, Google Cloud Storage, KDB+, Minio, Snowflake, or JDBC, then these options will appear in the **Add Dataset (or Drag & Drop)** dropdown menu. Refer to the [Enabling Data Connectors](#) section for more information.
- When specifying to add a dataset using **Data Recipe URL**, the URL must point to either an HTML or raw version of the file, a GitHub repository or tree, or a local file. When adding or uploading datasets via recipes, the dataset will be saved as a `.jay` file.
- Datasets must be in delimited text format.
- Driverless AI can detect the following separators: `,;|t`
- When importing a folder, the entire folder and all of its contents are read into Driverless AI as a single file.
- When importing a folder, all of the files in the folder must have the same columns.
- If you try to import a folder via a data connector on Windows, the import will fail if the folder contains files that do not have file extensions (the resulting error is usually related to the above note).

Upon completion, the datasets will appear in the Datasets Overview page. Click on a dataset to open a submenu. From this menu, you can specify to Rename, view Details of, Visualize, Split, Download, or Delete a dataset. **Note:** You cannot delete a dataset that was used in an active experiment. You have to delete the experiment first.

21.3 Renaming Datasets

In Driverless AI, you can rename datasets from the Datasets Overview page.

To rename a dataset, click on the dataset or select the **[Click for Actions]** button beside the dataset that you want to rename, and then select **Rename** from the submenu that appears.

Note: If the name of a dataset is changed, every instance of the dataset in Driverless AI will be changed to reflect the new name.

21.4 Dataset Details and Rows

To view a summary of a dataset or to preview the dataset, click on the dataset or select the **[Click for Actions]** button beside the dataset that you want to view, and then click **Details** from the submenu that appears. This opens the Dataset Details page, which provides a summary of the dataset. This summary lists each of the dataset's columns and displays accompanying rows for logical type, format, storage type (integer, string, real, boolean, or time), count, number of missing values, mean, minimum, maximum, standard deviation, frequency, and number of unique values.

Hover over the top of a column to view a summary of the first 20 rows of that column. To view information for a specific column, type the column name in the field above the graph.

To switch the view and preview the dataset, click the **Dataset Rows** button in the top right portion of the UI. Click the **Dataset Overview** button to return to the original view.

21.4.1 Changing a Column Type

Driverless AI also allows you to change a column type. If a column's data type or distribution does not match the manner in which you want the column to be handled during an experiment, changing the **Logical Type** can help to make the column fit better. For example, an integer zip code can be changed into a categorical so that it is only used with categorical-related feature engineering. For Date and Datetime columns, use the **Format** option. To change the Logical Type or Format of a column, click on the group of square icons located to the right of the words **Auto-detect**. (The squares light up when you hover over them with your cursor.) Then select the new column type for that column.

Modify By Recipe

The option to create a new dataset by modifying an existing dataset with custom recipes is also available from this page. Scoring pipelines can be created on the new dataset by building an experiment. This feature is useful when you want to make changes to the training data that you would not need to make on the new data you are predicting on. For example, you can change the target column from regression to classification, add a weight column to mark specific training rows as being more important, or remove outliers that you do not want to model on. Refer to the [Adding a Data Recipe](#) section for more information.

Click the **Modify by Recipe** button in the top right portion of the UI and select from the following options:

- **Data Recipe URL:** Load a custom recipe from a URL to use to modify the dataset. The URL must point to either an HTML or raw version of the file, a GitHub repository or tree, or a local file. Sample custom data recipes are available in the [driverlessai-recipes](#) repository.
- **Upload Data Recipe:** If you have a custom recipe available on your local system, click this button to upload that recipe.
- **Live Code:** Manually enter custom recipe code to use to modify the dataset. Click the **Get Preview** button to preview the code's effect on the dataset, then click **Save** to create a new dataset.

Notes:

- These options are enabled by default. You can disable them by removing `recipe_file` and `recipe_url` from the `enabled_file_systems` configuration option.
- Modifying a dataset with a recipe will not overwrite the original dataset. The dataset that is selected for modification will remain in the list of available datasets in its original form, and the modified dataset will appear in this list as a new dataset.
- Changes made to the original dataset through this feature will not be applied to new data that is scored.

21.5 Downloading Datasets

In Driverless AI, you can download datasets from the Datasets Overview page.

To download a dataset, click on the dataset or select the **[Click for Actions]** button beside the dataset that you want to download, and then select **Download** from the submenu that appears.

Note: The option to download datasets will not be available if the `enable_dataset_downloading` option is set to `false` when starting Driverless AI. This option can be specified in the `config.toml` file.

21.6 Splitting Datasets

In Driverless AI, you can split a training dataset into test and validation datasets.

Perform the following steps to split a dataset.

1. To split a dataset, click on the dataset or select the [Click for Actions] button beside the dataset that you want to split, and then select **Split** from the submenu that appears.
2. The Dataset Splitter form displays. Specify an Output Name 1 and an Output Name 2 for the first and second part of the split. (For example, you can name one test and one valid.)
3. Optionally specify a Target column (for stratified sampling), a Fold column (to keep rows belonging to the same group together), a Time column, and/or a Random Seed (defaults to 1234).
4. Use the slider to select a split ratio, or enter a value in the Train/Valid Split Ratio field.
5. Click **Save** when you are done.

Upon completion, the split datasets will be available on the Datasets page.

21.7 Visualizing Datasets

Perform one of the following steps to visualize a dataset:

- On the Datasets page, select the [Click for Actions] button beside the dataset that you want to view, and then click **Visualize** from the submenu that appears.
- Click the **Autoviz** top menu link to go to the Visualizations list page, click the **New Visualization** button, then select or import the dataset that you want to visualize.

21.7.1 The Visualization Page

The Visualization page shows all available graphs for the selected dataset. Note that the graphs on the Visualization page can vary based on the information in your dataset. You can also view and download logs that were generated during the visualization.

The following is a complete list of available graphs.

- **Correlated Scatterplots:** Correlated scatterplots are 2D plots with large values of the squared Pearson correlation coefficient. All possible scatterplots based on pairs of features (variables) are examined for correlations. The displayed plots are ranked according to the correlation. Some of these plots may not look like textbook examples of correlation. The only criterion is that they have a large value of squared Pearson's r (greater than .95). When modeling with these variables, you may want to leave out variables that are perfectly correlated with others.

Note that points in the scatterplot can have different sizes. Because Driverless AI aggregates the data and does not display all points, the bigger the point is, the bigger number of exemplars (aggregated points) the plot covers.

- **Spikey Histograms:** Spikey histograms are histograms with huge spikes. This often indicates an inordinate number of single values (usually zeros) or highly similar values. The measure of "spikeyness" is a bin frequency that is ten times the average frequency of all the bins. You should be careful when modeling (particularly regression models) with spiky variables.
- **Skewed Histograms:** Skewed histograms are ones with especially large skewness (asymmetry). The robust measure of skewness is derived from Groeneveld, R.A. and Meeden, G. (1984), "Measuring Skewness and

Kurtosis.” The Statistician, 33, 391-399. Highly skewed variables are often candidates for a transformation (e.g., logging) before use in modeling. The histograms in the output are sorted in descending order of skewness.

- **Varying Boxplots:** Varying boxplots reveal unusual variability in a feature across the categories of a categorical variable. The measure of variability is computed from a robust one-way analysis of variance (ANOVA). Sufficiently diverse variables are flagged in the ANOVA. A boxplot is a graphical display of the fractiles of a distribution. The center of the box denotes the median, the edges of a box denote the lower and upper quartiles, and the ends of the “whiskers” denote that range of values. Sometimes outliers occur, in which case the adjacent whisker is shortened to the next lower or upper value. For variables (features) having only a few values, the boxes can be compressed, sometimes into a single horizontal line at the median.
- **Heteroscedastic Boxplots:** Heteroscedastic boxplots reveal unusual variability in a feature across the categories of a categorical variable. Heteroscedasticity is calculated with a Brown-Forsythe test: Brown, M. B. and Forsythe, A. B. (1974), “Robust tests for equality of variances. Journal of the American Statistical Association, 69, 364-367. Plots are ranked according to their heteroscedasticity values. A boxplot is a graphical display of the fractiles of a distribution. The center of the box denotes the median, the edges of a box denote the lower and upper quartiles, and the ends of the “whiskers” denote that range of values. Sometimes outliers occur, in which case the adjacent whisker is shortened to the next lower or upper value. For variables (features) having only a few values, the boxes can be compressed, sometimes into a single horizontal line at the median.
- **Biplots:** A Biplot is an enhanced scatterplot that uses both points and vectors to represent structure simultaneously for rows and columns of a data matrix. Rows are represented as points (scores), and columns are represented as vectors (loadings). The plot is computed from the first two principal components of the correlation matrix of the variables (features). You should look for unusual (non-elliptical) shapes in the points that might reveal outliers or non-normal distributions. And you should look for purple vectors that are well-separated. Overlapping vectors can indicate a high degree of correlation between variables.
- **Outliers:** Variables with anomalous or outlying values are displayed as red points in a dot plot. Dot plots are constructed using an algorithm in Wilkinson, L. (1999). “Dot plots.” The American Statistician, 53, 276–281. Not all anomalous points are outliers. Sometimes the algorithm will flag points that lie in an empty region (i.e., they are not near any other points). You should inspect outliers to see if they are miscodings or if they are due to some other mistake. Outliers should ordinarily be eliminated from models only when there is a reasonable explanation for their occurrence.
- **Correlation Graph:** The correlation network graph is constructed from all pairwise squared correlations between variables (features). For continuous-continuous variable pairs, the statistic used is the squared Pearson correlation. For continuous-categorical variable pairs, the statistic is based on the squared intraclass correlation (ICC). This statistic is computed from the mean squares from a one-way analysis of variance (ANOVA). The formula is $(MS_{\text{between}} - MS_{\text{within}})/(MS_{\text{between}} + (k - 1)MS_{\text{within}})$, where k is the number of categories in the categorical variable. For categorical-categorical pairs, the statistic is computed from Cramer’s V squared. If the first variable has k₁ categories and the second variable has k₂ categories, then a k₁ x k₂ table is created from the joint frequencies of values. From this table, we compute a chi-square statistic. Cramer’s V squared statistic is then $(\chi^2 / n) / \min(k_1, k_2)$, where n is the total of the joint frequencies in the table. Variables with large values of these respective statistics appear near each other in the network diagram. The color scale used for the connecting edges runs from low (blue) to high (red). Variables connected by short red edges tend to be highly correlated.
- **Parallel Coordinates Plot:** A Parallel Coordinates Plot is a graph used for comparing multiple variables. Each variable has its own vertical axis in the plot. Each profile connects the values on the axes for a single observation. If the data contain clusters, these profiles will be colored by their cluster number.
- **Radar Plot:** A Radar Plot is a two-dimensional graph that is used for comparing multiple variables. Each variable has its own axis that starts from the center of the graph. The data are standardized on each variable between 0 and 1 so that values can be compared across variables. Each profile, which usually appears in the form of a star, connects the values on the axes for a single observation. Multivariate outliers are represented by red profiles. The Radar Plot is the polar version of the popular Parallel Coordinates plot. The polar layout enables us to represent more variables in a single plot.

- **Data Heatmap:** The heatmap graphic is constructed from the transposed data matrix. Rows of the heatmap represent variables, and columns represent cases (instances). The data are standardized before display so that small values are yellow and large values are red. The rows and columns are permuted via a singular value decomposition (SVD) of the data matrix so that similar rows and similar columns are near each other.
- **Recommendations:** The recommendations graphic implements the Tukey ladder of powers collection of log, square root, and inverse data transformations described in *Exploratory Data Analysis* (Tukey, 1977). Also implemented are extensions of these three transformers that handle negative values, which are derived from I.K. Yeo and R.A. Johnson, “A new family of power transformations to improve normality or symmetry.” *Biometrika*, 87(4), (2000). For each transformer, transformations are selected by comparing the robust skewness of the transformed column with the robust skewness of the original raw column. When a transformation leads to a relatively low value of skewness, it is recommended.
- **Missing Values Heatmap:** The missing values heatmap graphic is constructed from the transposed data matrix. Rows of the heatmap represent variables and columns represent cases (instances). The data are coded into the values 0 (missing) and 1 (nonmissing). Missing values are colored red and nonmissing values are left blank (white). The rows and columns are permuted via a singular value decomposition (SVD) of the data matrix so that similar rows and similar columns are near each other.
- **Gaps Histogram:** The gaps index is computed using an algorithm of Wainer and Schacht based on work by John Tukey. (Wainer, H. and Schacht, *Psychometrika*, 43, 2, 203-12.) Histograms with gaps can indicate a mixture of two or more distributions based on possible subgroups not necessarily characterized in the dataset.

The images on this page are thumbnails. You can click on any of the graphs to view and download a full-scale image. You can also view an explanation for each graph by clicking the **Help** button in the lower-left corner of each expanded graph.

21.7.2 Creating Custom Plots

To create a custom plot, click the **Add Graph** button in the upper-right corner and select one of the following plot types:

- **Bar chart:** This plot presents categorical data with rectangular bars that are proportional to the values they represent. The type of marker used to represent bars determines the bar chart type. The most common marker is the bar marker, which ranges from a lower value (usually zero) to an upper value. Also available are the Cleveland dot plot (replaces the bar with a dot located at the upper value) and the area chart (covers the bars with a solid area marker). Bars are always plotted against the categories of a categorical variable. They may represent counts (if no y variable is specified) or the average value of the y variable per category (if the y variable is specified).

When creating a bar chart, specify the following options:

- *x* variable name: Specify the name of the *x* variable
- *y* variable name: Specify the name of the *y* variable
- Transpose: Specify whether to switch the *X*-axis and *Y*-axis
- Sort: Specify whether to sort bars alphabetically by *x* values
- Mark: Specify a marker type. Select **point** to create a Cleveland dot plot

- **Boxplot:** This plot presents the fractiles of a distribution. The center of the box represents the median, the edges of a box represent the lower and upper quartiles, and the ends of the “whiskers” represent that range of values. When outliers occur, the adjacent whisker is shortened to the next lower or upper value. For variables having only a few values, the boxes can be compressed.

When creating a boxplot, specify the following options:

- Variable name: Specify the variable that you want the box to represent

- Transpose: Specify whether to switch the X -axis and Y -axis
- **Dotplot:** This plot represents individual data values with dots. When more than one value falls within a small neighborhood, the dots are stacked.

When creating a dotplot, specify the following options:

- Variable name: Specify the name of the variable on which dots are calculated
- Mark: Specify a marker type
- **Grouped Boxplot:** This plot is a boxplot where categories are organized into groups and subgroups.

When creating a grouped boxplot, specify the following options:

- Variable name: Specify the variable that you want the box to represent
- Group variable name: Specify the name of the grouping variable
- Transpose: Specify whether to switch the X -axis and Y -axis
- **Heatmap** - See [data heatmap](#). When creating a heatmap, specify the following options:
 - Variable names: Specify one or more variables to use. If none are specified, all the variables in the dataset are used
 - Permute: Specify whether to reorder variables using [singular value decomposition \(SVD\)](#)
 - Transpose: Specify whether to switch the X -axis and Y -axis
 - Matrix type: Specify a matrix type. Choose from **rectangular** and **symmetric**
- **Histogram:** This plot is a graphical display of data that uses bars of differing height. Each bar groups numbers into ranges by its width, and taller bars show that more data falls within a specific range. This plot is often used to display the shape and spread of a continuous variable.

When creating a histogram, specify the following options:

- Variable name: Specify the variable name
- Transformation: Specify whether to use a transformation. Choose from **log** and **square root**
- Number of bars: Specify the number of bars to use
- Mark: Specify a marker type. Use **area** to create a density polygon
- **Linear Regression:** This plot predicts a set of values on a variable y from values on a variable x by fitting a linear function ($ax + b$) so that for any value on the x variable, this function yields the most probable value on the y variable. The effectiveness of this prediction in a sample of values is represented by the discrepancies between the y values and their corresponding predicted values.

When creating a linear regression plot, specify the following options:

- x variable name: Specify the name of the x variable
- y variable name: Specify the name of the y variable
- Mark: Specify a marker type. Choose from **point** and **square**
- **LOESS Regression:** This plot predicts a set of values on a variable y from values on a variable x by fitting a locally linear function ($ax + b$) that determines the most probable y variable values based on the available x variable values. The effectiveness of this prediction in a sample of values is represented by the discrepancies between the y values and their corresponding predicted values.

When creating a LOESS regression plot, specify the following options:

- x variable name: Specify the name of the x variable

- *y* variable name: Specify the name of the *y* variable
 - Mark: Specify a marker type. Choose from **point** and **square**
 - Bandwidth: Specify the interval that represents the proportion of cases during the smoothing window. This is set to 0.5 by default
 - **Parallel Coordinates Plot:** This plot is used for comparing multiple variables. Each variable has its own vertical axis in the plot, and each profile connects the values on the axes for a single observation. If the data contains clusters, these profiles are color-coded by their cluster number.
- When creating a parallel coordinates plot, specify the following options:
- Variable names: Specify one or more variables to use. If none are specified, all the variables in the dataset are used
 - Permute: Specify whether to reorder variables using [singular value decomposition \(SVD\)](#)
 - Transpose: Specify whether to switch the *X*-axis and *Y*-axis
 - Cluster: Specify whether to include *k*-Means cluster variables. Unique colors are assigned for each cluster ID
- **Probability Plot:** This plot evaluates the skewness of a distribution by plotting two cumulative distribution functions against each other.

When creating a probability plot, specify the following options:

- *x* variable name: Specify the name of the *x* variable
- Distribution: Specify a distribution type. Choose from **normal** and **uniform**
- Mark: Specify a marker type. Choose from **point** and **square**
- Transpose: Specify whether to switch the *X*-axis and *Y*-axis

- **Quantile Plot:** This plot compares two probability distributions by plotting their quantiles against each other.

When creating a quantile plot, specify the following options:

- *x* variable name: Specify the name of the *x* variable
- *y* variable name: Specify the name of the *y* variable
- Distribution: Specify a distribution type. Choose from **normal** and **uniform**
- Mark: Specify a marker type. Choose from **point** and **square**
- Transpose: Specify whether to switch the *X*-axis and *Y*-axis

- **Scatterplot:** This plot represents the values of two variables (*y* and *x*) in a frame that contains one point for each row of the input sample data. They are useful for analyzing the joint distribution of two variables.

When creating a scatterplot, specify the following options:

- *x* variable name: Specify the name of the *x* variable
- *y* variable name: Specify the name of the *y* variable
- Mark: Specify a marker type. Choose from **point** and **square**

After selecting a plot, configure the available settings for that plot type and click **Save**. The custom plot appears on the Visualization page once it has been created.

The following example creates a custom histogram plot for the CreditCard-Train dataset:

CHAPTER
TWENTYTWO

EXPERIMENTS

22.1 Before You Begin

This section describes how to run an experiment using the Driverless AI UI. Before you begin, it is best that you understand the available options that you can specify. Note that only a dataset and a target column are required to be specified, but Driverless AI provides a variety of experiment, expert settings, and scorers that you can use to build your models. After you have a comfortable working knowledge of these options, proceed to the [New Experiments](#) section.

22.1.1 Experiment Settings

This section describes the settings that are available when running an experiment.

Display Name

Optional: Specify a display name for the new experiment. There are no character or length restrictions for naming. If this field is left blank, Driverless AI will automatically generate a name for the experiment.

Dropped Columns

Dropped columns are columns that you do not want to be used as predictors in the experiment. Note that Driverless AI will automatically drop ID columns and columns that contain a significant number of unique values (above `max_relative_cardinality` in the config.toml file or **Max. allowed fraction of uniques for integer and categorical cols** in Expert settings).

Validation Dataset

The validation dataset is used for tuning the modeling pipeline. If provided, the entire training data will be used for training, and validation of the modeling pipeline is performed with only this validation dataset. When you do not include a validation dataset, Driverless AI will do K-fold cross validation for I.I.D. experiments and multiple rolling window validation splits for time series experiments. For this reason it is not generally recommended to include a validation dataset as you are then validating on only a single dataset. Please note that time series experiments cannot be used with a validation dataset: including a validation dataset will disable the ability to select a time column and vice versa.

This dataset must have the same number of columns (and column types) as the training dataset. Also note that if provided, the validation set is not sampled down, so it can lead to large memory usage, even if accuracy=1 (which reduces the train size).

Test Dataset

The test dataset is used for testing the modeling pipeline and creating test predictions. The test set is never used during training of the modeling pipeline. (Results are the same whether a test set is provided or not.) If a test dataset is provided, then test set predictions will be available at the end of the experiment.

Weight Column

Optional: Column that indicates the observation weight (a.k.a. sample or row weight), if applicable. This column must be numeric with values ≥ 0 . Rows with higher weights have higher importance. The weight affects model training through a weighted loss function and affects model scoring through weighted metrics. The weight column is not used when making test set predictions, but a weight column (if specified) is used when computing the test score.

Fold Column

Optional: Rows with the same value in the fold column represent groups that should be kept together in the training, validation, or cross-validation datasets.

By default, Driverless AI assumes that the dataset is i.i.d. (identically and independently distributed) and creates validation datasets randomly for regression or with stratification of the target variable for classification.

The fold column is used to create the training and validation datasets so that all rows with the same Fold value will be in the same dataset. This can prevent data leakage and improve generalization. For example, when viewing data for a pneumonia dataset, `person_id` would be a good Fold Column. This is because the data may include multiple diagnostic snapshots per person, and we want to ensure that the same person's characteristics show up only in either the training or validation frames, but not in both to avoid data leakage.

This column must be an integer or categorical variable and cannot be specified if a validation set is used or if a Time Column is specified.

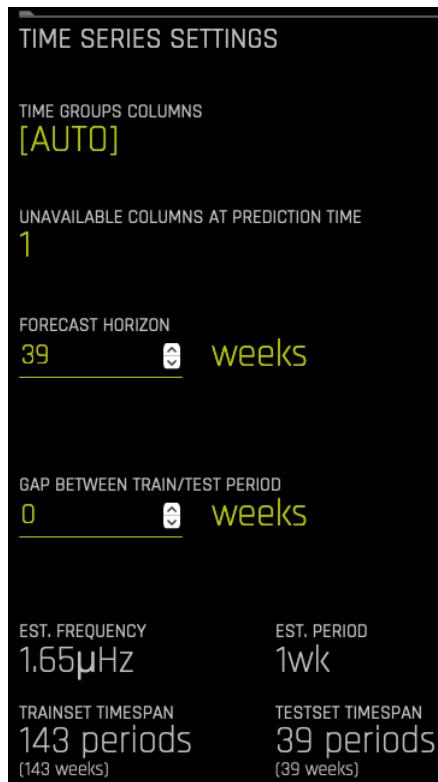
Time Column

Optional: Specify a column that provides a time order (time stamps for observations), if applicable. This can improve model performance and model validation accuracy for problems where the target values are auto-correlated with respect to the ordering (per time-series group).

The values in this column must be a datetime format understood by `pandas.to_datetime()`, like “2017-11-29 00:30:35” or “2017/11/29”, or integer values. If [AUTO] is selected, all string columns are tested for potential date/datetime content and considered as potential time columns. If a time column is found, feature engineering and model validation will respect the causality of time. If [OFF] is selected, no time order is used for modeling and data may be shuffled randomly (any potential temporal causality will be ignored).

When your data has a date column, then in most cases, specifying [AUTO] for the Time Column will be sufficient. However, if you select a specific date column, then Driverless AI will provide you with an additional side menu. From this side menu, you can specify Time Group columns or specify [Auto] to let Driverless AI determine the best time group columns. You can also specify the columns that will be unavailable at prediction time (see [More About Unavailable Columns at Time of Prediction](#) for more information), the Forecast Horizon (in a unit of time identified by Driverless AI), and the Gap between the train and test periods.

Refer to [Time Series in Driverless AI](#) for more information about time series experiments in Driverless AI and to see a time series example.

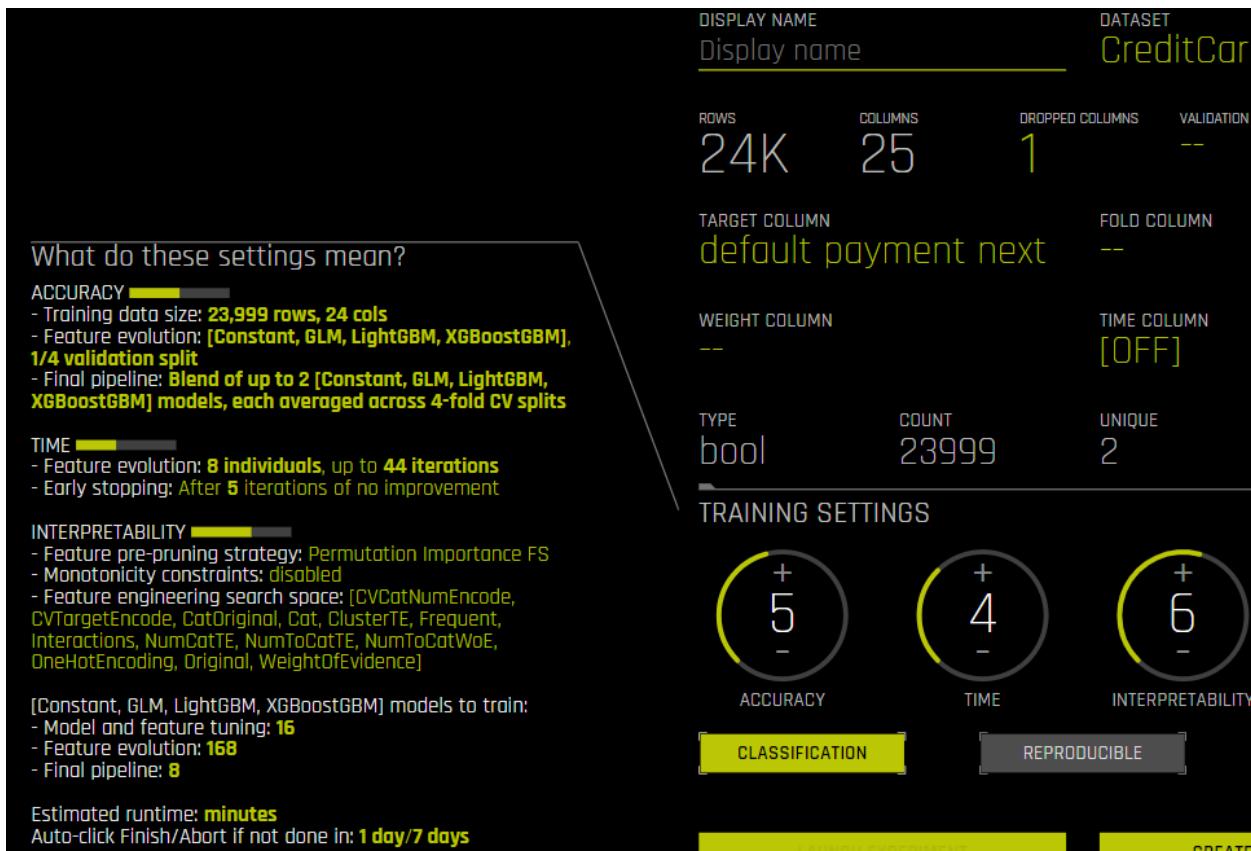


Notes:

- Engineered features will be used for MLI when a time series experiment is built. This is because munged time series features are more useful features for MLI compared to raw time series features.
- A Time Column cannot be specified if a Fold Column is specified. This is because both fold and time columns are only used to split training datasets into training/validation, so once you split by time, you cannot also split with the fold column. If a Time Column is specified, then the time group columns play the role of the fold column for time series.
- A Time Column cannot be specified if a validation dataset is used.
- A column that is specified as being unavailable at prediction time will only have lag-related features created for (or with) it.
- Unavailable Columns at Time of Prediction will only have lag-related features created for (or with) it, so this option is only used when *Time-Series Lag-Based Recipe* is enabled.

Accuracy, Time, and Interpretability Knobs

The experiment preview describes what the Accuracy, Time, and Interpretability settings mean for your specific experiment. This preview will automatically update if any of the knob values change. The following is more detailed information describing how these values affect an experiment.



Accuracy

As accuracy increases (as indicated by the tournament_* toml settings), Driverless AI gradually adjusts the method for performing the evolution and ensemble. At low accuracy, Driverless AI varies features and models, but they all compete evenly against each other. At higher accuracy, each independent main model will evolve independently and be part of the final ensemble as an ensemble over different main models. At higher accuracies, Driverless AI will evolve+ensemble feature types like Target Encoding on and off that evolve independently. Finally, at highest accuracies, Driverless AI performs both model and feature tracking and ensembles all those variations.

Changing this value affects the feature evolution and final pipeline.

Note: A check for a shift in the distribution between train and test is done for accuracy ≥ 5 .

Feature evolution: This represents the algorithms used to create the experiment. If a test set is provided without a validation set, then Driverless AI will perform a 1/3 validation split during the experiment. If a validation set is provided, then the experiment will perform external validation.

Final pipeline: This represents the number of models and the validation method used in the final pipeline. For ensemble modeling, information about how models are combined is also shown here.

Time

This specifies the relative time for completing the experiment (i.e., higher settings take longer).

Feature evolution: This shows the number of individuals and maximum number of iterations that will be run in this experiment. **Early stopping:** Early stopping will take place if the experiment doesn't improve the score for the specified amount of iterations.

Interpretability

Specify the relative interpretability for this experiment. Higher values favor more interpretable models. Changing the interpretability level affects the feature pre-pruning strategy, monotonicity constraints, and the feature engineering search space.

Feature pre-pruning strategy: This represents the feature selection strategy (to prune-away features that do not clearly give improvement to model score). Strategy = “Permutation Importance FS” if interpretability ≥ 6 ; otherwise strategy is None.

Monotonicity constraints: If **Monotonicity Constraints** are enabled, the model will satisfy knowledge about monotonicity in the data and monotone relationships between the predictors and the target variable. For example, in house price prediction, the house price should increase with lot size and number of rooms, and should decrease with crime rate in the area. If enabled, Driverless AI will automatically determine if monotonicity is present and enforce it in its modeling pipelines. Depending on the correlation, Driverless AI will assign positive, negative, or no monotonicity constraints. Monotonicity is enforced if the absolute correlation is greater than 0.1. All other predictors will not have monotonicity enforced.

Note: Monotonicity constraints are used in XGBoost GBM, XGBoost Dart, LightGBM, and Decision Tree models.

Feature engineering search space: This represents the transformers that will be used during the experiment.

Classification/Regression Button

Driverless AI automatically determines the problem type based on the response column. Though not recommended, you can override this setting by clicking this button.

Reproducible

The **Reproducible** button allows you to build an experiment with a random seed and get reproducible results. If this is disabled (default), then results will vary between runs, which can give a good sense of variance among experiment results.

Please keep in mind the following when enabling this option:

- Experiments can only be reproducible when run on the same hardware (same number and type of GPUs/CPUs, same architecture such as Linux or PPC, etc). For example, you will not get the same results if you try an experiment on a GPU machine, and then attempt to reproduce the results on a CPU-only machine or on a machine with a different number and type of GPUs.
- This option should be used with the *Reproducibility Level* expert setting option, which ensures different degrees of reproducibility based on the OS and environment architecture. Keep in mind that when **Reproducibility** is enabled, then `reproducibility_level=1` by default.
- Experiments run using TensorFlow with multiple cores cannot be reproduced.

- LightGBM is more reproducible with 64-bit floats, and Driverless AI will switch to 64-bit floats for LightGBM. (Refer to https://lightgbm.readthedocs.io/en/latest/Parameters.html#gpu_use_dp for more information.)
- Enabling this option automatically disables all of the Feature Brain expert settings options; specifically:
 - *Model/Feature Brain Level*
 - *Feature Brain Save Every Which Iteration*
 - *Feature Brain Restart from Which Iteration*
 - *Feature Brain Refit Uses Same Best Individual*
 - *Feature Brain Adds Features with New Columns Even During Retraining of Final Model*

Enable GPUs

Click the **Enable GPUs** button to enable/disable GPUs. Note that this option is ignored on CPU-only systems.

22.1.2 Expert Settings

This section describes the Expert Settings that are available when starting an experiment. Driverless AI provides a variety of options in the Expert Settings that allow you to customize your experiment. Use the search bar to refine the list of settings or locate a specific setting.

The default values for these options are derived from the configuration options in the config.toml file. Refer to the *Sample Config.toml File* section for more information about each of these options.

Note about Feature Brain Level: By default, the feature brain pulls in any better model regardless of the features even if the new model disabled those features. For full control over features pulled in via changes in these Expert Settings, users should set the **Feature Brain Level** option to 0.

Expert Experiment Settings

+ UPLOAD CUSTOM RECIPE + LOAD CUSTOM RECIPE FROM URL OFFICIAL RECIPES (EXTERNAL) TOML

Start typing to filter out items

EXPERIMENT	MODEL	FEATURES	TIMESERIES	NLP	IMAGE	RECIPES	SYSTEM
Max. runtime in minutes before triggering the 'Finish' button. Approximately enforced. (0 = disabled) 1440	Max. runtime in minutes before triggering the 'Abort' button.(0 = disabled) 10080	Kaggle key Kaggle username		Pipeline Building Recipe AUTO ▾	Kaggle submission timeout in seconds 120	Enable genetic algorithm for selection and tuning of features and models AUTO ON OFF	Make Python scoring pipeline AUTO ON OFF
Make MOJO scoring pipeline AUTO ON OFF	Attempt to reduce the size of the MOJO DISABLED			Measure MOJO scoring latency AUTO ON OFF		Timeout in seconds to wait for MOJO creation at end of experiment. 1800	
Number of parallel workers to use during MOJO creation (-1 = all cores) -1	Make pipeline visualization AUTO ON OFF			Make Autoreport ENABLED		Min. number of rows needed to run experiment 100	
Reproducibility Level 1	Random seed 1234			Allow different sets of classes across all train/validation fold splits ENABLED		Max. number of classes for classification problems 200	
Max. number of classes to compute ROC and confusion matrix for classification	Max. number of classes to show in GUI for ROC/CM reduction technique for large class						

SAVE CANCEL

Upload Custom Recipe

Driverless AI supports the use of custom recipes (optional). If you have a custom recipe available on your local system, click this button to upload that recipe. If you do not have a custom recipe, you can select from a number of recipes available in the <https://github.com/h2oai/driverlessai-recipes> repository. Clone this repository on your local machine and upload the desired recipe. Refer to the [Custom Recipes](#) appendix for examples.

Load Custom Recipe from URL

If you have a custom recipe available on an external system, specify the URL for that recipe here. Both HTML and raw versions of the Python file are supported. Refer to the [Custom Recipes](#) appendix for examples.

Official Recipes (External)

Click this button to access H2O's official recipes repository (<https://github.com/h2oai/driverlessai-recipes>).

Edit the TOML Configuration

This feature is a graphical representation of the [Add to config.toml via toml String](#) expert setting. Click this button to edit the config.toml file from within Driverless AI. Click **Save** to confirm your changes. The experiment preview changes to reflect the specified edits.

Experiment Settings

Max Runtime in Minutes Before Triggering the Finish Button

Specify the maximum runtime in minutes for an experiment. This is equivalent to pushing the **Finish** button once half of the specified time value has elapsed. Note that the overall enforced runtime is only an approximation.

This value defaults to 1440, which is the equivalent of a 24 hour approximate overall runtime. The **Finish** button will be automatically selected once 12 hours have elapsed, and Driverless AI will subsequently attempt to complete the overall experiment in the remaining 12 hours. Set this value to 0 to disable this setting.

Max Runtime in Minutes Before Triggering the Abort Button

Specify the maximum runtime in minutes for an experiment before triggering the abort button. This option preserves experiment artifacts that have been generated for the summary and log zip files while continuing to generate additional artifacts. This value defaults to 10080.

Pipeline Building Recipe

Specify the Pipeline Building recipe type (overrides GUI settings). Select from the following:

- **Auto:** Specifies that all models and features are automatically determined by experiment settings, config.toml settings, and the feature engineering effort. (Default)
- **Compliant:** Similar to **Auto** except for the following:
 - Interpretability is set to 10.
 - Only uses GLM.

- Fixed ensemble level is set to 0.
 - Feature brain level is set to 0.
 - Max feature interaction depth is set to 1.
 - Target transformers is set to ‘identity’ for regression.
 - Does not use distribution shift.
- **Kaggle:** Similar to **Auto** except for the following:
 - Any external validation set is concatenated with the train set, with the target marked as missing.
 - The test set is concatenated with the train set, with the target marked as missing
 - Transformers that do not use the target are allowed to `fit_transform` across the entirety of the train, validation, and test sets.
 - Has several config.toml expert options open-up limits.
 - **nlp_model:** Only enable NLP models that process pure text.
 - **nlp_transformer:** Only enable NLP transformers that process pure text.
 - **image_model:** Only enable image models that process pure images (ImageAutoModel). See *Automatic Image Model (GPU Only)* for more information.

Notes:

- This option disables the Genetic Algorithm (GA).
 - Image insights are only available when this option is selected.
- **image_transformer:** Only enable image transformers that process pure images.

Kaggle Username

Optionally specify your Kaggle username to enable automatic submission and scoring of test set predictions. If this option is specified, then you must also specify a value for the **Kaggle Key** option. If you don’t have a Kaggle account, you can sign up at <https://www.kaggle.com>.

Kaggle Key

Specify your Kaggle API key to enable automatic submission and scoring of test set predictions. If this option is specified, then you must also specify a value for the **Kaggle Username** option. For more information on obtaining Kaggle API credentials, see <https://github.com/Kaggle/kaggle-api#api-credentials>.

Kaggle Submission Timeout in Seconds

Specify the Kaggle submission timeout in seconds. This value defaults to 120.

Make Python Scoring Pipeline

Specify whether to automatically build a Python Scoring Pipeline for the experiment. Select **On** or **Auto** (default) to make the Python Scoring Pipeline immediately available for download when the experiment is finished. Select **Off** to disable the automatic creation of the Python Scoring Pipeline.

Make MOJO Scoring Pipeline

Specify whether to automatically build a MOJO (Java) Scoring Pipeline for the experiment. Select **On** to make the MOJO Scoring Pipeline immediately available for download when the experiment is finished. With this option, any capabilities that prevent the creation of the pipeline are dropped. Select **Off** to disable the automatic creation of the MOJO Scoring Pipeline. Select **Auto** (default) to attempt to create the MOJO Scoring Pipeline without dropping any capabilities.

Attempt to Reduce the Size of the MOJO

Specify whether to attempt to reduce the size of the MOJO scoring pipeline when it is being built. A smaller MOJO has a smaller memory footprint during scoring. This is disabled by default.

Note: Enabling this setting can affect the overall predictive accuracy of the model because it is implemented by reducing values like interaction depth.

Measure MOJO Scoring Latency

Specify whether to measure the MOJO scoring latency at the time of MOJO creation. This is set to **Auto** by default. In this case, MOJO scoring latency will be measured if the pipeline.mojo file size is less than 100 MB.

Timeout in Seconds to Wait for MOJO Creation at End of Experiment

Specify the amount of time in seconds to wait for MOJO creation at the end of an experiment. If the MOJO creation process times out, a MOJO can still be made from the GUI or the R and Python clients (the timeout constraint is not applied to these). This value defaults to 1800 (30 minutes).

Number of Parallel Workers to Use During MOJO Creation

Specify the number of parallel workers to use during MOJO creation. Higher values can speed up MOJO creation but use more memory. Set this value to -1 (default) to use all physical cores.

Make Pipeline Visualization

Specify whether to create a visualization of the scoring pipeline at the end of an experiment. This is set to **Auto** by default. Note that the **Visualize Scoring Pipeline** feature is experimental and is not available for deprecated models. Visualizations are available for all newly created experiments.

Make Autoreport

Specify whether to create the experiment Autoreport after the experiment is finished. This is enabled by default.

Min Number of Rows Needed to Run an Experiment

Specify the minimum number of rows that a dataset must contain in order to run an experiment. This value defaults to 100.

Reproducibility Level

Specify one of the following levels of reproducibility. Note that this setting is only used when the *Reproducible* option is enabled in the experiment:

- 1 = Same experiment results for same O/S, same CPU(s), and same GPU(s) (Default)
- 2 = Same experiment results for same O/S, same CPU architecture, and same GPU architecture
- 3 = Same experiment results for same O/S, same CPU architecture (excludes GPUs)
- 4 = Same experiment results for same O/S (best approximation)

This value defaults to 1.

Random Seed

Specify a random seed for the experiment. When a seed is defined and the reproducible button is enabled (not by default), the algorithm will behave deterministically.

Allow Different Sets of Classes Across All Train/Validation Fold Splits

(**Note:** Applicable for multiclass problems only.) Specify whether to enable full cross-validation (multiple folds) during feature evolution as opposed to a single holdout split. This is enabled by default.

Max Number of Classes for Classification Problems

Specify the maximum number of classes to allow for a classification problem. A higher number of classes may make certain processes more time-consuming. Memory requirements also increase with a higher number of classes. This value defaults to 200.

Max Number of Classes to Compute ROC and Confusion Matrix for Classification Problems

Specify the maximum number of classes to use when computing the ROC and CM. When this value is exceeded, the reduction type specified by `roc_reduce_type` is applied. This value defaults to 200 and cannot be lower than 2.

Max Number of Classes to Show in GUI for Confusion Matrix

Specify the maximum number of classes to show in the GUI for CM, showing first max_num_classes_client_and_gui labels. This value defaults to 10, but any value beyond 6 will result in visually truncated diagnostics. Note that if this value is changed in the config.toml and the server is restarted, then this setting will only modify client-GUI launched diagnostics. To control experiment plots, this value must be changed in the expert settings panel.

ROC/CM Reduction Technique for Large Class Counts

Specify the ROC/CM reduction technique used for large class counts:

- **Rows** (Default): Reduce by randomly sampling rows
- **Classes**: Reduce by truncating classes to no more than the value specified by max_num_classes_compute_roc

Model/Feature Brain Level

Specify whether to use H2O.ai brain, which enables local caching and smart re-use (checkpointing) of prior experiments to generate useful features and models for new experiments. It can also be used to control checkpointing for experiments that have been paused or interrupted.

When enabled, this will use the H2O.ai brain cache if the cache file:

- has any matching column names and types for a similar experiment type
- has classes that match exactly
- has class labels that match exactly
- has basic time series choices that match
- the interpretability of the cache is equal or lower
- the main model (booster) is allowed by the new experiment
- -1: Don't use any brain cache (default)
- 0: Don't use any brain cache but still write to cache. Use case: Want to save the model for later use, but we want the current model to be built without any brain models.
- 1: Smart checkpoint from the latest best individual model. Use case: Want to use the latest matching model. The match may not be precise, so use with caution.
- 2: Smart checkpoint if the experiment matches all column names, column types, classes, class labels, and time series options identically. Use case: Driverless AI scans through the H2O.ai brain cache for the best models to restart from.
- 3: Smart checkpoint like level #1 but for the entire population. Tune only if the brain population is of insufficient size. Note that this will re-score the entire population in a single iteration, so it appears to take longer to complete first iteration.
- 4: Smart checkpoint like level #2 but for the entire population. Tune only if the brain population is of insufficient size. Note that this will re-score the entire population in a single iteration, so it appears to take longer to complete first iteration.
- 5: Smart checkpoint like level #4 but will scan over the entire brain cache of populations to get the best scored individuals. Note that this can be slower due to brain cache scanning if the cache is large.

When enabled, the directory where the H2O.ai Brain meta model files are stored is H2O.ai_brain. In addition, the default maximum brain size is 20GB. Both the directory and the maximum size can be changed in the config.toml file. This value defaults to 2.

Feature Brain Save Every Which Iteration

Save feature brain iterations every iter_num % feature_brain_iterations_save_every_iteration == 0, to be able to restart/refit with which_iteration_brain >= 0. This is disabled (0) by default.

- -1: Don't use any brain cache.
- 0: Don't use any brain cache but still write to cache.
- 1: Smart checkpoint if an old experiment_id is passed in (for example, via running "resume one like this" in the GUI).
- 2: Smart checkpoint if the experiment matches all column names, column types, classes, class labels, and time series options identically. (default)
- 3: Smart checkpoint like level #1 but for the entire population. Tune only if the brain population is of insufficient size.
- 4: Smart checkpoint like level #2 but for the entire population. Tune only if the brain population is of insufficient size.
- 5: Smart checkpoint like level #4 but will scan over the entire brain cache of populations (starting from resumed experiment if chosen) in order to get the best scored individuals.

When enabled, the directory where the H2O.ai Brain meta model files are stored is H2O.ai_brain. In addition, the default maximum brain size is 20GB. Both the directory and the maximum size can be changed in the config.toml file.

Feature Brain Restart from Which Iteration

When performing restart or re-fit of type feature_brain_level with a resumed ID, specify which iteration to start from instead of only last best. Available options include:

- -1: Use the last best
- 1: Run one experiment with feature_brain_iterations_save_every_iteration=1 or some other number
- 2: Identify which iteration brain dump you wants to restart/refit from
- 3: Restart/Refit from the original experiment, setting which_iteration_brain to that number here in expert settings.

Note: If restarting from a tuning iteration, this will pull in the entire scored tuning population and use that for feature evolution. This value defaults to -1.

Feature Brain Refit Uses Same Best Individual

Specify whether to use the same best individual when performing a refit. Disabling this setting allows the order of best individuals to be rearranged, leading to a better final result. Enabling this setting allows you to view the exact same model or feature with only one new feature added. This is disabled by default.

Feature Brain Adds Features with New Columns Even During Retraining of Final Model

Specify whether to add additional features from new columns to the pipeline, even when performing a retrain of the final model. Use this option if you want to keep the same pipeline regardless of new columns from a new dataset. New data may lead to new dropped features due to shift or leak detection. Disable this to avoid adding any columns as new features so that the pipeline is perfectly preserved when changing data. This is enabled by default.

Restart-Refit Use Default Model Settings If Model Switches

When restarting or refitting, specify whether to use the model class's default settings if the original model class is no longer available. If this is disabled, the original hyperparameters will be used instead. (Note that this may result in errors.) This is enabled by default.

Min DAI Iterations

Specify the minimum number of Driverless AI iterations for an experiment. This can be used during restarting, when you want to continue for longer despite a score not improving. This value defaults to 0.

Select Target Transformation of the Target for Regression Problems

Specify whether to automatically select target transformation for regression problems. Available options include:

- auto
- identity
- identity_noclip
- unit_box
- log
- square
- sqrt
- double_sqrt
- inverse
- logit
- sigmoid

If set to **auto** (default), Driverless AI will automatically pick the best target transformer if the **Accuracy** is set to the value of the `tune_target_transform_accuracy_switch` configuration option (defaults to 3) or larger. Selecting **identity** automatically turns off any target transformations. All transformers except for **identity_noclip** perform clipping to constrain the predictions to the domain of the target in the training data. Use **identity_noclip** to disable target transformations and to allow predictions outside of the target domain observed in the training data for parametric models or custom models that support extrapolation.

Enable Genetic Algorithm for Selection and Tuning of Features and Models

Specify whether to enable genetic algorithm for selection and hyper-parameter tuning of features and models. If this is disabled, the final pipeline is trained using the default feature engineering and feature selection. This is set to **Auto** by default.

Tournament Model for Genetic Algorithm

Select a method to decide which models are best at each iteration. This is set to **Auto** by default. Choose from the following:

- **auto**: Choose based on scoring metric
- **fullstack**: Choose from optimal model and feature types
- **feature**: Individuals with similar feature types compete
- **model**: Individuals with same model type compete
- **uniform**: All individuals in population compete

Number of Cross-Validation Folds for Feature Evolution

Specify the fixed number of cross-validation folds (if ≥ 2) for feature evolution. Note that the actual number of allowed folds can be less than the specified value, and that the number of allowed folds is determined at the time an experiment is run. This value defaults to -1 (auto).

Number of Cross-Validation Folds for Final Model

Specify the fixed number of cross-validation folds (if ≥ 2) for the final model. Note that the actual number of allowed folds can be less than the specified value, and that the number of allowed folds is determined at the time an experiment is run. This value defaults to -1 (auto).

Force Only First Fold for Models

Specify whether to force only the first fold for models. Select from **Auto** (Default), **On**, or **Off**.

Max Number of Rows Times Number of Columns for Feature Evolution Data Splits

Specify the maximum number of rows allowed for feature evolution data splits (not for the final pipeline). This value defaults to 100,000,000.

Max Number of Rows Times Number of Columns for Reducing Training Dataset

Specify the upper limit on the number of rows times the number of columns for training the final pipeline. This value defaults to 500,000,000.

Maximum Size of Validation Data Relative to Training Data

Specify the maximum size of the validation data relative to the training data. Smaller values can make the final pipeline model training process quicker. Note that final model predictions and scores will always be provided on the full dataset provided. This value defaults to 2.0.

Perform Stratified Sampling for Binary Classification If the Target Is More Imbalanced Than This

For binary classification experiments, specify a threshold ratio of minority to majority class for the target column beyond which stratified sampling is performed. If the threshold is not exceeded, random sampling is performed. This value defaults to 0.1. You can choose to always perform random sampling by setting this value to 0, or to always perform stratified sampling by setting this value to 1.

Add to config.toml via toml String

Specify any additional configuration overrides from the config.toml file that you want to include in the experiment. (Refer to the [Sample Config.toml File](#) section to view options that can be overridden during an experiment.) Setting this will override all other settings. Separate multiple config overrides with \n. For example, the following enables Poisson distribution for LightGBM and disables Target Transformer Tuning. Note that in this example double quotes are escaped (\\" \").

```
params_lightgbm="{'objective':'poisson'}\n target_transformer=identity
```

Or you can specify config overrides similar to the following without having to escape double quotes:

```
"enable_glm="off"\n enable_xgboost_gbm="off"\n enable_lightgbm="off"\n enable_tensorflow="on"""\n\nmax_cores=10\n data_precision="float32"\n max_rows_feature_evolution=5000000000\n ensemble_accuracy_switch=11\n feature_engineering_effort=1\n\n←target_transformer="identity"\n tournament_feature_style_accuracy_switch=5\n params_tensorflow="{'layers': [100, 100, 100, 100, 100]}"""
```

When running the Python client, config overrides would be set as follows:

```
model = h2o.start_experiment_sync(\n    dataset_key=train.key,\n    target_col='target',\n    is_classification=True,\n    accuracy=7,\n    time=5,\n    interpretability=1,\n    config_overrides=""\n        feature_brain_level=0\n        enable_lightgbm="off"\n        enable_xgboost_gbm="off"\n        enable_ftree="off"\n    ""\n)
```

Model Settings

Constant Models

Specify whether to enable *constant models*. This is set to **Auto** (enabled) by default.

Decision Tree Models

Specify whether to build Decision Tree models as part of the experiment. This is set to **Auto** by default. In this case, Driverless AI will build Decision Tree models if interpretability is greater than or equal to the value of `decision_tree_interpretability_switch` (which defaults to 7) and accuracy is less than or equal to `decision_tree_accuracy_switch` (which defaults to 7).

GLM Models

Specify whether to build GLM models (generalized linear models) as part of the experiment (usually only for the final model unless it's used exclusively). GLMs are very interpretable models with one coefficient per feature, an intercept term and a link function. This is set to **Auto** by default (enabled if accuracy ≤ 5 and interpretability ≥ 6).

XGBoost GBM Models

Specify whether to build XGBoost models as part of the experiment (for both the feature engineering part and the final model). XGBoost is a type of gradient boosting method that has been widely successful in recent years due to its good regularization techniques and high accuracy. This is set to **Auto** by default. In this case, Driverless AI will use XGBoost unless the number of rows * columns is greater than a threshold. This threshold is a config setting that is 100M by default for CPU and 30M by default for GPU.

XGBoost Dart Models

Specify whether to use XGBoost's Dart method when building models for experiment (for both the feature engineering part and the final model). This is set to **Auto** (disabled) by default.

LightGBM Models

Specify whether to build LightGBM models as part of the experiment. LightGBM Models are the default models. This is set to **Auto** (enabled) by default.

TensorFlow Models

Specify whether to build TensorFlow models as part of the experiment (usually only for text features engineering and for the final model unless it's used exclusively). Enable this option for NLP experiments. This is set to **Auto** by default (not used unless the number of classes is greater than 10).

TensorFlow models are not yet supported by MOJOs (only Python scoring pipelines are supported).

PyTorch Models

Specify whether to build PyTorch models as part of the experiment. This is set to **Auto** by default.

FTRL Models

Specify whether to build Follow the Regularized Leader (FTRL) models as part of the experiment. Note that MOJOS are not yet supported (only Python scoring pipelines). FTRL supports binomial and multinomial classification for categorical targets, as well as regression for continuous targets. This is set to **Auto** (disabled) by default.

RuleFit Models

Specify whether to build RuleFit models as part of the experiment. Note that MOJOS are not yet supported (only Python scoring pipelines). Note that multiclass classification is not yet supported for RuleFit models. Rules are stored to text files in the experiment directory for now. This is set to **Auto** (disabled) by default.

Zero-Inflated Models

Specify whether to enable the automatic addition of *zero-inflated models* for regression problems with zero-inflated target values that meet certain conditions:

```
y >= 0, y.std() > y.mean()
```

This is set to **Auto** by default.

LightGBM Boosting Types

Specify which boosting types to enable for LightGBM. Select one or more of the following:

- **gbdt**: Boosted trees
- **rf_early_stopping**: Random Forest with early stopping
- **rf**: Random Forest
- **dart**: Dropout boosted trees with no early stopping

gbdt and **rf** are both enabled by default.

LightGBM Categorical Support

Specify whether to enable LightGBM categorical feature support. This is disabled by default.

Notes:

- Only supported for CPU.
- A MOJO is not built when this is enabled.

Whether to Show Constant Models in Iteration Panel

Specify whether to show constant models in the iteration panel. This is disabled by default.

Parameters for TensorFlow

Specify specific parameters for TensorFlow to override Driverless AI parameters. The following is an example of how the parameters can be configured:

```
params_tensorflow = {'lr': 0.01, 'add_wide': False, 'add_attention': True, 'epochs': 30,
'layers': [100, 100], 'activation': 'selu', 'batch_size': 64, 'chunk_size': 1000, 'dropout': 0.3,
'strategy': 'one_shot', 'l1': 0.0, 'l2': 0.0, 'ort_loss': 0.5, 'ort_loss_tau': 0.01, 'normalize_type': 'streaming'}
```

The following is an example of how layers can be configured:

```
[500, 500, 500], [100, 100, 100], [100, 100], [50, 50]
```

More information about TensorFlow parameters can be found in the [Keras documentation](#). Different strategies for using TensorFlow parameters can be viewed [here](#).

Max Number of Trees/Iterations

Specify the upper limit on the number of trees (GBM) or iterations (GLM). This defaults to 3000. Depending on accuracy settings, a fraction of this limit will be used.

n_estimators List to Sample From for Model Mutations for Models That Do Not Use Early Stopping

For LightGBM, the dart and normal random forest modes do not use early stopping. This setting allows you to specify the n_estimators (number of trees in the forest) list to sample from for model mutations for these types of models.

Minimum Learning Rate for Final Ensemble GBM Models

Specify the minimum learning rate for final ensemble GBM models. This value defaults to 0.01.

Maximum Learning Rate for Final Ensemble GBM Models

Specify the maximum learning rate for final ensemble GBM models. This value defaults to 0.05.

Reduction Factor for Max Number of Trees/Iterations During Feature Evolution

Specify the factor by which the value specified by the *Max Number of Trees/Iterations* setting is reduced for tuning and feature evolution. This option defaults to 0.2. So by default, Driverless AI will produce no more than $0.2 * 3000$ trees/iterations during feature evolution.

Minimum Learning Rate for Feature Engineering GBM Models

Specify the minimum learning rate for feature engineering GBM models. This value defaults to 0.05.

Max Learning Rate for Tree Models

Specify the maximum learning rate for tree models during feature engineering. Higher values can speed up feature engineering but can hurt accuracy. This value defaults to 0.5.

Max Number of Epochs for TensorFlow/FTRL

When building TensorFlow or FTRL models, specify the maximum number of epochs to train models with (it might stop earlier). This value defaults to 10. This option is ignored if **TensorFlow models** and/or **FTRL models** is disabled.

Max Tree Depth

Specify the maximum tree depth. The corresponding maximum value for `max_leaves` is double the specified value. This value defaults to 12.

Max `max_bin` for Tree Features

Specify the maximum `max_bin` for tree features. This value defaults to 256.

Max Number of Rules for RuleFit

Specify the maximum number of rules to be used for RuleFit models. This defaults to -1, which specifies to use all rules.

Ensemble Level for Final Modeling Pipeline

Specify one of the following ensemble levels:

- -1 = auto, based upon `ensemble_accuracy_switch`, `accuracy`, size of data, etc. (Default)
- 0 = No ensemble, only final single model on validated iteration/tree count. Note that holdout predicted probabilities will not be available. (Refer to the following [FAQ](#).)
- 1 = 1 model, multiple ensemble folds (cross-validation)
- 2 = 2 models, multiple ensemble folds (cross-validation)
- 3 = 3 models, multiple ensemble folds (cross-validation)
- 4 = 4 models, multiple ensemble folds (cross-validation)

Cross-Validate Single Final Model

Driverless AI normally produces a single final model for low accuracy settings (typically, less than 5). When the **Cross-validate single final model** option is enabled (default for regular experiments), Driverless AI will perform cross-validation to determine optimal parameters and early stopping before training the final single modeling pipeline on the entire training data. The final pipeline will build $N + 1$ models, with N-fold cross validation for the single final model. This also creates holdout predictions for all non-time-series experiments with a single final model.

Note that the setting for this option is ignored for time-series experiments or when a validation dataset is provided.

Number of Models During Tuning Phase

Specify the number of models to tune during pre-evolution phase. Specify a lower value to avoid excessive tuning, or specify a higher to perform enhanced tuning. This option defaults to -1 (auto).

Sampling Method for Imbalanced Binary Classification Problems

Specify the sampling method for imbalanced binary classification problems. This is set to **off** by default. Choose from the following options:

- **auto**: sample both classes as needed, depending on data
- **over_under_sampling**: over-sample the minority class and under-sample the majority class, depending on data
- **under_sampling**: under-sample the majority class to reach class balance
- **off**: do not perform any sampling

This option is closely tied with the Imbalanced Light GBM and Imbalanced XGBoost GBM models, which can be enabled/disabled on the Recipes tab under *Include Specific Models*. Specifically:

- If this option is ENABLED (set to a value other than **off**) and the ImbalancedLightGBM and/or ImbalancedXGBoostGBM models are ENABLED, then Driverless AI will check your target imbalance fraction. If the target fraction proves to be above the allowed imbalance threshold, then sampling will be triggered.
- If this option is ENABLED and the ImbalancedLightGBM and/or ImbalancedXGBoostGBM models are DISABLED, then no special sampling technique will be performed. The setting here will be ignored.
- If this option is DISABLED and the ImbalancedLightGBM and/or ImbalancedXGBoostGBM models are ENABLED, sampling will not be used, and the imbalanced models will be disabled.

This means that to enable sampling, you should enable this option (either as **auto**, **over_under_sampling**, or **under_sampling**) AND enable the imbalanced models.

Threshold for Minimum Number of Rows in Original Training Data to Allow Imbalanced Sampling

Specify a threshold for the minimum number of rows in the original training data that allow imbalanced sampling. This value defaults to 100,000.

Ratio of Majority to Minority Class for Imbalanced Binary Classification to Trigger Special Sampling Techniques (if Enabled)

For imbalanced binary classification problems, specify the ratio of majority to minority class. Special imbalanced models with sampling techniques are enabled when the ratio is equal to or greater than the specified ratio. This value defaults to 5.

Ratio of Majority to Minority Class for Heavily Imbalanced Binary Classification to Only Enable Special Sampling Techniques (if Enabled)

For heavily imbalanced binary classification, specify the ratio of the majority to minority class equal and above which to enable only special imbalanced models on the full original data without upfront sampling. This value defaults to 25.

Number of Bags for Sampling Methods for Imbalanced Binary Classification (if Enabled)

Specify the number of bags for sampling methods for imbalanced binary classification. This value defaults to -1.

Hard Limit on Number of Bags for Sampling Methods for Imbalanced Binary Classification

Specify the limit on the number of bags for sampling methods for imbalanced binary classification. This value defaults to 10.

Hard Limit on Number of Bags for Sampling Methods for Imbalanced Binary Classification During Feature Evolution Phase

Specify the limit on the number of bags for sampling methods for imbalanced binary classification. This value defaults to 3. Note that this setting only applies to shift, leakage, tuning, and feature evolution models. To limit final models, use the **Hard Limit on Number of Bags for Sampling Methods for Imbalanced Binary Classification** setting.

Max Size of Data Sampled During Imbalanced Sampling

Specify the maximum size of the data sampled during imbalanced sampling in terms of the dataset's size. This setting controls the approximate number of bags and is only active when the "Hard limit on number of bags for sampling methods for imbalanced binary classification during feature evolution phase" option is set to -1. This value defaults to 1.

Target Fraction of Minority Class After Applying Under/Over-Sampling Techniques

Specify the target fraction of a minority class after applying under/over-sampling techniques. A value of 0.5 means that models/algorithms will be given a balanced target class distribution. When starting from an extremely imbalanced original target, it can be advantageous to specify a smaller value such as 0.1 or 0.01. This value defaults to -1.

Max Number of Automatic FTRL Interactions Terms for 2nd, 3rd, 4th order interactions terms (Each)

Specify a limit for the number of FTRL interactions terms sampled for each of second, third, and fourth order terms. This value defaults to 10,000.

Enable Detailed Scored Model Info

Specify whether to dump every scored individual's model parameters to a csv/tabulated file. If enabled (default), Driverless AI produces files such as "individual_scored_id%d.iter%d*params*". This is enabled by default.

Whether to Enable Bootstrap Sampling for Validation and Test Scores

Specify whether to enable bootstrap sampling. When enabled, this setting provides error bars to validation and test scores based on the standard error of the bootstrap mean. This is enabled by default.

For Classification Problems with This Many Classes, Default to TensorFlow

Specify the number of classes above which to use TensorFlow when it is enabled. Others model that are set to **Auto** will not be used above this number. (Models set to **On**, however, are still used.) This value defaults to 10.

Compute Prediction Intervals

Specify whether to compute empirical prediction intervals based on holdout predictions. This is enabled by default.

Confidence Level for Prediction Intervals

Specify a confidence level for prediction intervals. This value defaults to 0.9.

Features Settings

Feature Engineering Effort

Specify a value from 0 to 10 for the Driverless AI feature engineering effort. Higher values generally lead to more time (and memory) spent in feature engineering. This value defaults to 5.

- 0: Keep only numeric features. Only model tuning during evolution.
- 1: Keep only numeric features and frequency-encoded categoricals. Only model tuning during evolution.
- 2: Similar to 1 but instead just no Text features. Some feature tuning before evolution.
- 3: Similar to 5 but only tuning during evolution. Mixed tuning of features and model parameters.
- 4: Similar to 5 but slightly more focused on model tuning.
- 5: Balanced feature-model tuning. (Default)
- 6-7: Similar to 5 but slightly more focused on feature engineering.
- 8: Similar to 6-7 but even more focused on feature engineering with high feature generation rate and no feature dropping even if high interpretability.

- 9-10: Similar to 8 but no model tuning during feature evolution.

Data Distribution Shift Detection

Specify whether Driverless AI should detect data distribution shifts between train/valid/test datasets (if provided). Currently, this information is only presented to the user and not acted upon.

Data Distribution Shift Detection Drop of Features

Specify whether to drop high-shift features. This defaults to **Auto**. Note that Auto for time series experiments turns this feature off.

Max Allowed Feature Shift (AUC) Before Dropping Feature

Specify the maximum allowed AUC value for a feature before dropping the feature.

When train and test differ (or train/valid or valid/test) in terms of distribution of data, then there can be a model built that tells you for each row whether the row is in train or test. That model includes an AUC value. If the AUC, GINI, or Spearman correlation is above this specified threshold, then Driverless AI will consider it a strong enough shift to drop those features.

This value defaults to 0.999.

Leakage Detection

Specify whether to check leakage for each feature. Note that this is always disabled if a fold column is specified and if the experiment is a time series experiment. This is set to **Auto** by default.

Leakage Detection Dropping AUC/R2 Threshold

If Leakage Detection is enabled, specify the threshold for dropping features. When the AUC (or R2 for regression), GINI, or Spearman correlation is above this value, the feature will be dropped. This value defaults to 0.999.

Max Rows Times Columns for Leakage

Specify the maximum number of rows times the number of columns to trigger sampling for leakage checks. This value defaults to 10,000,000.

Report Permutation Importance on Original Features

Specify whether Driverless AI reports permutation importance on original features. This is disabled by default.

Maximum Number of Rows to Perform Permutation-Based Feature Selection

Specify the maximum number of rows to when performing permutation feature importance. This value defaults to 500,000.

Max Number of Original Features Used

Specify the maximum number of columns to be selected from an existing set of columns using feature selection. This value defaults to 10,000.

Max Number of Original Non-Numeric Features

Specify the maximum number of non-numeric columns to be selected. Feature selection is performed on all features when this value is exceeded. This value defaults to 300.

Max Number of Original Features Used for FS Individual

Specify the maximum number of features you want to be selected in an experiment. Additional columns above the specified value add special individual with original columns reduced. This value defaults to 500.

Number of Original Numeric Features to Trigger Feature Selection Model Type

The maximum number of original numeric columns, above which Driverless AI will do feature selection. Note that this is applicable only to special individuals with original columns reduced. A separate individual in the genetic algorithm is created by doing feature selection by permutation importance on original features. This value defaults to 500.

Number of Original Non-Numeric Features to Trigger Feature Selection Model Type

The maximum number of original non-numeric columns, above which Driverless AI will do feature selection on all features. Note that this is applicable only to special individuals with original columns reduced. A separate individual in the genetic algorithm is created by doing feature selection by permutation importance on original features. This value defaults to 200.

Max Allowed Fraction of Uniques for Integer and Categorical Columns

Specify the maximum fraction of unique values for integer and categorical columns. If the column has a larger fraction of unique values than that, it will be considered an ID column and ignored. This value defaults to 0.95.

Allow Treating Numerical as Categorical

Specify whether to allow some numerical features to be treated as categorical features. This is enabled by default.

Max Number of Unique Values for Int/Float to be Categoricals

Specify the number of unique values for integer or real columns to be treated as categoricals. This value defaults to 50.

Max Number of Engineered Features

Specify the maximum number of features to include in the final model's feature engineering pipeline. If -1 is specified (default), then Driverless AI will automatically determine the number of features.

Max Number of Genes

Specify the maximum number of genes (transformer instances) kept per model (and per each model within the final model for ensembles). This controls the number of genes before features are scored, so Driverless AI will just randomly samples genes if pruning occurs. If restriction occurs after scoring features, then aggregated gene importances are used for pruning genes. Instances includes all possible transformers, including original transformer for numeric features. A value of -1 means no restrictions except internally-determined memory and interpretability restriction.

Limit Features by Interpretability

Specify whether to limit feature counts with the **Interpretability** training setting as specified by the `features_allowed_by_interpretability config.toml` setting.

Threshold for Interpretability Above Which to Enable Automatic Monotonicity Constraints for Tree Models

Specify an Interpretability setting value equal and above which to use automatic monotonicity constraints in XGBoost-GBM, LightGBM, or Decision Tree models. This value defaults to 7.

Correlation Beyond Which Triggers Monotonicity Constraints (if Enabled)

Specify the threshold of Pearson product-moment correlation coefficient between numerical or encoded transformed feature and target above (below negative for) which to use positive (negative) monotonicity for XGBoostGBM, LightGBM and Decision Tree models. This value defaults to 0.1.

Note: This setting is only enabled when Interpretability is greater than or equal to the value specified by the [*Threshold for Interpretability Above Which to Enable Automatic Monotonicity Constraints for Tree Models*](#) setting and when the [*Manual Override for Monotonicity Constraints*](#) setting is not specified.

Manual Override for Monotonicity Constraints

Specify a list of features for which monotonicity constraints are applied. Original numeric features are mapped to the desired constraint:

- 1: Positive constraint
- -1: Negative constraint
- 0: Constraint disabled

Constraint is automatically disabled (set to 0) for features that are not in this list.

The following is an example of how this list can be specified:

```
{'PAY_0': -1, 'PAY_2': -1, 'AGE': -1, 'BILL_AMT1': 1, 'PAY_AMT1': -1}
```

Note: If a list is not provided, then the automatic correlation-based method is used when monotonicity constraints are enabled at high enough interpretability settings.

Max Feature Interaction Depth

Specify the maximum number of features to use for interaction features like grouping for target encoding, weight of evidence, and other likelihood estimates.

Exploring feature interactions can be important in gaining better predictive performance. The interaction can take multiple forms (i.e. feature1 + feature2 or feature1 * feature2 + ... featureN). Although certain machine learning algorithms (like tree-based methods) can do well in capturing these interactions as part of their training process, still generating them may help them (or other algorithms) yield better performance.

The depth of the interaction level (as in “up to” how many features may be combined at once to create one single feature) can be specified to control the complexity of the feature engineering process. Higher values might be able to make more predictive models at the expense of time. This value defaults to 8.

Fixed Feature Interaction Depth

Specify a fixed non-zero number of features to use for interaction features like grouping for target encoding, weight of evidence, and other likelihood estimates. To use all features for each transformer, set this to be equal to the number of columns. To do a 50/50 sample and a fixed feature interaction depth of n features, set this to $-n$.

Enable Target Encoding

Specify whether to use Target Encoding when building the model. Target encoding refers to several different feature transformations (primarily focused on categorical data) that aim to represent the feature using information of the actual target variable. A simple example can be to use the mean of the target to replace each unique category of a categorical feature. These type of features can be very predictive but are prone to overfitting and require more memory as they need to store mappings of the unique categories and the target values.

Enable Outer CV for Target Encoding

For target encoding, specify whether an outer level of cross-fold validation is performed in cases where GINI is detected to flip sign or have an inconsistent sign for weight of evidence between `fit_transform` (on training data) and `transform` (on training and validation data). The degree to which GINI is inaccurate is also used to perform fold-averaging of look-up tables instead of using global look-up tables. This is enabled by default.

Enable Lexicographical Label Encoding

Specify whether to enable lexicographical label encoding. This is disabled by default.

Enable Isolation Forest Anomaly Score Encoding

[Isolation Forest](#) is useful for identifying anomalies or outliers in data. Isolation Forest isolates observations by randomly selecting a feature and then randomly selecting a split value between the maximum and minimum values of that selected feature. This split depends on how long it takes to separate the points. Random partitioning produces noticeably shorter paths for anomalies. When a forest of random trees collectively produces shorter path lengths for particular samples, they are highly likely to be anomalies.

This option allows you to specify whether to return the anomaly score of each sample. This is disabled by default.

Enable One HotEncoding

Specify whether one-hot encoding is enabled. The default **Auto** setting is only applicable for small datasets and GLMs.

Number of Estimators for Isolation Forest Encoding

Specify the number of estimators for [Isolation Forest](#) encoding. This value defaults to 200.

Drop Constant Columns

Specify whether to drop columns with constant values. This is enabled by default.

Drop ID Columns

Specify whether to drop columns that appear to be an ID. This is enabled by default.

Don't Drop Any Columns

Specify whether to avoid dropping any columns (original or derived). This is disabled by default.

Features to Drop

Specify which features to drop. This setting allows you to select many features at once by copying and pasting a list of column names (in quotes) separated by commas.

Features to Group By

Specify which features to group columns by. When this field is left empty (default), Driverless AI automatically searches all columns (either at random or based on which columns have high variable importance).

Sample from Features to Group By

Specify whether to sample from given features to group by or to always group all features. This is disabled by default.

Aggregation Functions (Non-Time-Series) for Group By Operations

Specify whether to enable aggregation functions to use for group by operations. Choose from the following (all are selected by default):

- mean
- sd
- min
- max
- count

Number of Folds to Obtain Aggregation When Grouping

Specify the number of folds to obtain aggregation when grouping. Out-of-fold aggregations will result in less overfitting, but they analyze less data in each fold.

Type of Mutation Strategy

Specify which strategy to apply when performing mutations on transformers. Select from the following:

- **sample**: Sample transformer parameters (Default)
- **batched**: Perform multiple types of the same transformation together
- **full**: Perform more types of the same transformation together than the above strategy

Enable Detailed Scored Features Info

Specify whether to dump every scored individual's variable importance (both derived and original) to a csv/tabulated/json file. If enabled, Driverless AI produces files such as "individual_scored_id%d.ter%d*features*". This is disabled by default.

Enable Detailed Logs for Timing and Types of Features Produced

Specify whether to dump every scored fold's timing and feature info to a **timings.txt** file. This is disabled by default.

Compute Correlation Matrix

Specify whether to compute training, validation, and test correlation matrixes. When enabled, this setting creates table and heatmap PDF files that are saved to disk. Note that this setting is currently a single threaded process that may be slow for experiments with many columns. This is disabled by default.

Required GINI Relative Improvement for Interactions

Specify the required GINI relative improvement value for the InteractionTransformer. If the GINI coefficient is not better than the specified relative improvement value in comparison to the original features considered in the interaction, then the interaction is not returned. If the data is noisy and there is no clear signal in interactions, this value can be decreased to return interactions. This value defaults to 0.5.

Number of Transformed Interactions to Make

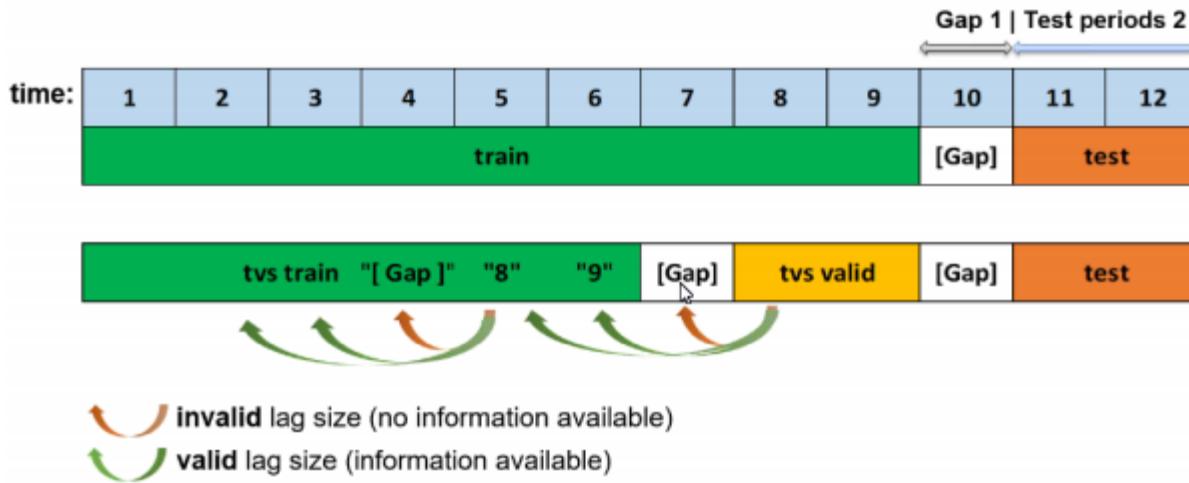
Specify the number of transformed interactions to make from generated trial interactions. (The best transformed interactions are selected from the group of generated trial interactions.) This value defaults to 5.

Time Series Settings

Time-Series Lag-Based Recipe

This recipe specifies whether to include Time Series lag features when training a model with a provided (or autodetected) time column. This is enabled by default. Lag features are the primary automatically generated time series features and represent a variable's past values. At a given sample with time stamp t , features at some time difference T (lag) in the past are considered. For example, if the sales today are 300, and sales of yesterday are 250, then the lag of one day for sales is 250. Lags can be created on any feature as well as on the target. Lagging variables are important in time series because knowing what happened in different time periods in the past can greatly facilitate predictions for the future. **Note:** Ensembling is disabled when the lag-based recipe with time columns is activated because it only supports a single final model. Ensembling is also disabled if a time column is selected or if time column is set to **[Auto]** on the experiment setup screen.

More information about time series lag is available in the *Time Series Use Case: Sales Forecasting* section.



Custom Validation Splits for Time-Series Experiments

Specify date or datetime timestamps (in the same format as the time column) to use for custom training and validation splits.

Timeout in Seconds for Time-Series Properties Detection in UI

Specify the timeout in seconds for time-series properties detection in Driverless AI's user interface. This value defaults to 30.

Generate Holiday Features

For time-series experiments, specify whether to generate holiday features for the experiment. This is enabled by default.

List of Countries for Which to Look up Holiday Calendar and to Generate Is-Holiday Features For

Specify country codes in the form of a list that is used to look up holidays.

Note: This setting is for migration purposes only.

Time-Series Lags Override

Specify the override lags to be used. These can be used to give more importance to the lags that are still considered after the override is applied. The following examples show the variety of different methods that can be used to specify override lags:

- “[7, 14, 21]” specifies this exact list
- “21” specifies every value from 1 to 21
- “21:3” specifies every value from 1 to 21 in steps of 3
- “5-21” specifies every value from 5 to 21

- “5-21:3” specifies every value from 5 to 21 in steps of 3

Smallest Considered Lag Size

Specify a minimum considered lag size. This value defaults to -1.

Enable Feature Engineering from Time Column

Specify whether to enable feature engineering based on the selected time column, e.g. Date~weekday. This is enabled by default.

Allow Integer Time Column as Numeric Feature

Specify whether to enable feature engineering from an integer time column. Note that if you are using a time series recipe, using a time column (numeric time stamps) as an input feature can lead to a model that memorizes the actual timestamps instead of features that generalize to the future. This is disabled by default.

Allowed Date and Date-Time Transformations

Specify the date or date-time transformations to allow Driverless AI to use. Choose from the following transformers:

- year
- quarter
- month
- week
- weekday
- day
- dayofyear
- num (direct numeric value representing the floating point value of time, disabled by default)
- hour
- minute
- second

Features in Driverless AI will appear as `get_` followed by the name of the transformation. Note that `get_num` can lead to overfitting if used on IID problems and is disabled by default.

Consider Time Groups Columns as Standalone Features

Specify whether to consider time groups columns as standalone features. This is disabled by default.

Which TGC Feature Types to Consider as Standalone Features

Specify whether to consider time groups columns (TGC) as standalone features. If “Consider time groups columns as standalone features” is enabled, then specify which TGC feature types to consider as standalone features. Available types are **numeric**, **categorical**, **ohe_categorical**, **datetime**, **date**, and **text**. All types are selected by default. Note that “**time_column**” is treated separately via the “Enable Feature Engineering from Time Column” option. Also note that if “Time Series Lag-Based Recipe” is disabled, then all time group columns are allowed features.

Enable Time Unaware Transformers

Specify whether various transformers (clustering, truncated SVD) are enabled, which otherwise would be disabled for time series experiments due to the potential to overfit by leaking across time within the fit of each fold. This is set to **Auto** by default.

Always Group by All Time Groups Columns for Creating Lag Features

Specify whether to group by all time groups columns for creating lag features. This is enabled by default.

Generate Time-Series Holdout Predictions

Specify whether to create diagnostic holdout predictions on training data using moving windows. This is enabled by default. This can be useful for MLi, but it will slow down the experiment considerably when enabled. Note that the model itself remains unchanged when this setting is enabled.

Number of Time-Based Splits for Internal Model Validation

Specify a fixed number of time-based splits for internal model validation. Note that the actual number of allowed splits can be less than the specified value, and that the number of allowed splits is determined at the time an experiment is run. This value defaults to -1 (auto).

Maximum Overlap Between Two Time-Based Splits

Specify the maximum overlap between two time-based splits. The amount of possible splits increases with higher values. This value defaults to 0.5.

Maximum Number of Splits Used for Creating Final Time-Series Model's Holdout Predictions

Specify the maximum number of splits used for creating the final time-series Model's holdout predictions. The default value (-1) will use the same number of splits that are used during model validation.

Whether to Speed up Calculation of Time-Series Holdout Predictions

Specify whether to speed up time-series holdout predictions for back-testing on training data. This setting is used for MLI and calculating metrics. Note that predictions can be slightly less accurate when this setting is enabled. This is disabled by default.

Whether to Speed up Calculation of Shapley Values for Time-Series Holdout Predictions

Specify whether to speed up Shapley values for time-series holdout predictions for back-testing on training data. This setting is used for MLI. Note that predictions can be slightly less accurate when this setting is enabled. This is enabled by default.

Generate Shapley Values for Time-Series Holdout Predictions at the Time of Experiment

Specify whether to enable the creation of Shapley values for holdout predictions on training data using moving windows at the time of the experiment. This can be useful for MLI, but it can slow down the experiment when enabled. If this setting is disabled, MLI will generate Shapley values on demand. This is enabled by default.

Lower Limit on Interpretability Setting for Time-Series Experiments (Implicitly Enforced)

Specify the lower limit on interpretability setting for time-series experiments. Values of 5 (default) or more can improve generalization by more aggressively dropping the least important features. To disable this setting, set this value to 1.

Dropout Mode for Lag Features

Specify the dropout mode for lag features in order to achieve an equal n.a. ratio between train and validation/tests. **Independent** mode performs a simple feature-wise dropout. **Dependent** mode takes the lag-size dependencies per sample/row into account. **Dependent** is enabled by default.

Probability to Create Non-Target Lag Features

Lags can be created on any feature as well as on the target. Specify a probability value for creating non-target lag features. This value defaults to 0.1.

Method to Create Rolling Test Set Predictions

Specify the method used to create rolling test set predictions. Choose between test time augmentation (TTA) and a successive refitting of the final pipeline. TTA is enabled by default.

Rolling Method Used for GA Validation

Specify whether the rolling test method specified by the *Method to Create Rolling Test Set Predictions* setting is used for the validation set when the genetic algorithm is applied. This is enabled by default.

Probability for New Time-Series Transformers to Use Default Lags

Specify the probability for new lags or the EWMA gene to use default lags. This is determined independently of the data by frequency, gap, and horizon. This value defaults to 0.2.

Probability of Exploring Interaction-Based Lag Transformers

Specify the unnormalized probability of choosing other lag time-series transformers based on interactions. This value defaults to 0.2.

Probability of Exploring Aggregation-Based Lag Transformers

Specify the unnormalized probability of choosing other lag time-series transformers based on aggregations. This value defaults to 0.2.

Time Series Centering or Detrending Transformation

Specify whether to use centering or detrending transformation for time series experiments. Select from the following:

- None (Default)
- Centering (Fast)
- Centering (Robust)
- Linear (Fast)
- Linear (Robust)
- Logistic
- Epidemic (Uses the **SEIRD** model)

The trend is removed from the target signal once the free parameters of the trend model are fitted. Predictions are made by adding the previously removed trend once the pipeline is fitted on the residuals.

Notes:

- MOJO support is currently disabled when this setting is enabled.
- The **Robust** centering and linear detrending options use random sample consensus (RANSAC) to achieve higher tolerance w.r.t. outliers.

Custom Bounds for SEIRD Epidemic Model Parameters

Specify the custom bounds for controlling Susceptible-Infected-Exposed-Recovered-Dead (SEIRD) epidemic model parameters for detrending of the target for each time series group. The target column must correspond to $I(t)$, which represents infection cases as a function of time.

For each training split and time series group, the SEIRD model is fit to the target signal by optimizing a set of free parameters for each time series group. The model's value is then subtracted from the training response, and the residuals are passed to the feature engineering and modeling pipeline. For predictions, the SEIRD model's value is added to the residual predictions from the pipeline for each time series group.

The following is a list of free parameters:

- **N**: Total population, $N = S+E+I+R+D$
- **beta**: Rate of exposure ($S \rightarrow E$)
- **gamma**: Rate of recovering ($I \rightarrow R$)
- **delta**: Incubation period
- **alpha**: Fatality rate
- **rho**: Rate at which individuals expire
- **lockdown**: Day of lockdown (-1 => no lockdown)
- **beta_decay**: Beta decay due to lockdown
- **beta_decay_rate**: Speed of beta decay

Provide upper or lower bounds for each parameter you want to control. The following is a list of valid parameters:

- N_min
- N_max
- beta_min
- beta_max
- gamma_min
- gamma_max
- delta_min
- delta_max
- alpha_min
- alpha_max
- rho_min
- rho_max
- lockdown_min
- lockdown_max
- beta_decay_min
- beta_decay_max
- beta_decay_rate_min
- beta_decay_rate_max

You can change any subset of parameters. For example:

```
ts_target_trafo_epidemic_params_dict = {"N_min": 1000, "beta_max": 0.2}
```

Refer to https://en.wikipedia.org/wiki/Compartmental_models_in_epidemiology and <https://arxiv.org/abs/1411.3435> for more information on the SEIRD model.

Note: In cases where death rates are very low, the **SEIR** model can speed up calculations significantly. To get the SEIR model, set `alpha_min=alpha_max=rho_min=rho_max=beta_decay_rate_min=beta_decay_rate_max=0` and `lockdown_min=lockdown_max=-1`.

Which SEIRD Model Component the Target Column Corresponds To

Specify a SEIRD model component for the target column to correspond to. Select from the following:

- **I** (Default): Infected
- **R**: Recovered
- **D**: Deceased

Time Series Lag-Based Target Transformation

Specify whether to use either the difference between or ratio of the current target and a lagged target. Select from **None** (default), **Difference**, and **Ratio**.

Notes:

- MOJO support is currently disabled when this setting is enabled.
- The corresponding lag size is specified with the *Lag Size Used for Time Series Target Transformation* expert setting.

Lag Size Used for Time Series Target Transformation

Specify the lag size used for time series target transformation. Specify this setting when using the *Time Series Lag-Based Target Transformation* setting. This value defaults to -1.

NLP Settings

Max TensorFlow Epochs for NLP

When building TensorFlow NLP features (for text data), specify the maximum number of epochs to train feature engineering models with (it might stop earlier). The higher the number of epochs, the higher the run time. This value defaults to 2 and is ignored if **TensorFlow models** is disabled.

Accuracy Above Enable TensorFlow NLP by Default for All Models

Specify the accuracy threshold. Values equal and above will add all enabled TensorFlow NLP models at the start of the experiment for text-dominated problems when the following NLP expert settings are set to **Auto**:

- Enable word-based CNN TensorFlow models for NLP
- Enable word-based BiGRU TensorFlow models for NLP
- Enable character-based CNN TensorFlow models for NLP

If the above transformations are set to **ON**, this parameter is ignored.

At lower accuracy, TensorFlow NLP transformations will only be created as a mutation. This value defaults to 5.

Enable Word-Based CNN TensorFlow Models for NLP

Specify whether to use Word-based CNN TensorFlow models for NLP. This option is ignored if TensorFlow is disabled. We recommend that you disable this option on systems that do not use GPUs.

Enable Word-Based BiGRU TensorFlow Models for NLP

Specify whether to use Word-based BiG-RU TensorFlow models for NLP. This option is ignored if TensorFlow is disabled. We recommend that you disable this option on systems that do not use GPUs.

Enable Character-Based CNN TensorFlow Models for NLP

Specify whether to use Character-level CNN TensorFlow models for NLP. This option is ignored if TensorFlow is disabled. We recommend that you disable this option on systems that do not use GPUs.

Enable PyTorch Models for NLP (Experimental)

Specify whether to enable pretrained PyTorch models and fine-tune them for NLP tasks. This is set to **Auto** by default. You need to set this to **On** if you want to use the PyTorch models like BERT for feature engineering or for modeling. We recommend that you use GPUs to speed up execution when this option is used.

Notes:

- This setting requires an Internet connection.
- Some PyTorch NLP models may only use one text column.

Select Which Pretrained PyTorch NLP Models to Use

Specify one or more pretrained PyTorch NLP models to use. Select from the following:

- bert-base-uncased (Default)
- distilbert-base-uncased (Default)
- xlnet-base-cased
- xlm-mlm-enfr-1024
- roberta-base

- albert-base-v2
- camembert-base
- xlm-roberta-base

Notes:

- This setting requires an Internet connection.
- Models that are not selected by default may not have MOJO support.
- Using BERT-like models may result in a longer experiment completion time.

Number of Epochs for Fine-Tuning of PyTorch NLP Models

Specify the number of epochs used when fine-tuning PyTorch NLP models. This value defaults to 2.

Batch Size for PyTorch NLP Models

Specify the batch size for PyTorch NLP models. This value defaults to 10.

Note: Large models and batch sizes require more memory.

Maximum Sequence Length for PyTorch NLP Models

Specify the maximum sequence length (padding length) for PyTorch NLP models. This value defaults to 100.

Note: Large models and padding lengths require more memory.

Path to Pretrained PyTorch NLP Models

Specify a path to pretrained PyTorch NLP models. To get all available models, download http://s3.amazonaws.com/artifacts.h2o.ai/releases/ai/h2o/pretrained/bert_models.zip, then extract the folder and store it in a directory on the instance where Driverless AI is installed:

```
pytorch_nlp_pretrained_models_dir = /path/on/server/to/bert_models_folder
```

Path to Pretrained Embeddings for TensorFlow NLP Models

Specify a path to pretrained embeddings that will be used for the TensorFlow NLP models. For example, /path/on/server/to/file.txt

- You can download the Glove embeddings from [here](#) and specify the local path in this box.
- You can download the fasttext embeddings from [here](#) and specify the local path in this box.
- You can also train your own custom embeddings. Please refer to [this code sample](#) for creating custom embeddings that can be passed on to this option.
- If this field is left empty, embeddings will be trained from scratch.

Allow Training of Unfrozen Pretrained Embeddings

Specify whether to allow training of all weights of the neural network graph, including the pretrained embedding layer weights. If this is disabled, the embedding layer will be frozen. All other weights, however, will still be fine-tuned. This is disabled by default.

Fraction of Text Columns Out of All Features to be Considered a Text-Dominated Problem

Specify the fraction of text columns out of all features to be considered as a text-dominated problem. This value defaults to 0.3.

Specify when a string column will be treated as text (for an NLP problem) or just as a standard categorical variable. Higher values will favor string columns as categoricals, while lower values will favor string columns as text. This value defaults to 0.3.

Fraction of Text per All Transformers to Trigger That Text Dominated

Specify the fraction of text columns out of all features to be considered a text-dominated problem. This value defaults to 0.3.

Threshold for String Columns to be Treated as Text

Specify the threshold value (from 0 to 1) for string columns to be treated as text (0.0 - text; 1.0 - string). This value defaults to 0.3.

Image Settings

Image Download Timeout in Seconds

When providing images through URLs, specify the maximum number of seconds to wait for an image to download. This value defaults to 60.

Batch Size for TensorFlow Image Models

Specify the batch size for TensorFlow image models. By default, the batch size is set to -1. **Note:** Larger models and batch sizes use more memory.

Enable Fine-Tuning of the Pretrained Image Models Used for the Image Vectorizer Transformers

Specify whether to enable fine-tuning of the pretrained image models used for the Image Vectorizer transformers. This is disabled by default.

Number of Epochs for Fine-Tuning of TensorFlow Image Models

Specify the number of epochs used for fine-tuning TensorFlow image models. This value defaults to 2.

Supported Pretrained Image Recognition Models

Specify the supported pretrained image recognition models. Select from the following:

- xception (Selected by default)
- resnet50
- inception_v3
- inception_resnet_v2
- resnext50
- seresnext50
- efficientnetb1
- efficientnetb3

Note: If an internet connection is available, non-default models are downloaded automatically. If an internet connection is not available, non-default models must be downloaded from http://s3.amazonaws.com/artifacts.h2o.ai/releases/ai/h2o/pretrained/image_models.zip and extracted into `./tmp` or `tensorflow_image_pretrained_models_dir` (specified in the config.toml file).

Dimensionality of Feature Space Created by Pretrained TensorFlow Image Models

Specify the dimensionality of the feature space created by TensorFlow image models. Select from the following:

- 10
- 25
- 50
- 100 (Default)
- 200
- 300

Note: Multiple transformers can be activated at the same time to allow the selection of multiple options.

Enable TensorFlow Models for Processing of Image Data

Specify whether to use pretrained TensorFlow deep learning models for processing image data as part of the feature engineering pipeline. When this is enabled, a column of Uniform Resource Identifiers (URIs) to images is converted to a numeric representation using pretrained deep learning models. This is enabled by default.

Maximum Allowed Fraction of Missing Values for Image Column

Specify the maximum allowed fraction of missing elements in a string column for it to be considered as a potential image path. This value defaults to 0.1.

Minimum Fraction of Images That Need to Be of Valid Types for Image Column to Be Used

Specify the fraction of unique image URIs that need to have valid endings (as defined by `string_col_as_image_valid_types`) for a string column to be considered as image data. This value defaults to 0.8.

Enable GPU(s) for Faster Predictions With Pretrained Tensorflow Models

Specify whether to use any available GPUs to make predictions with pretrained TensorFlow models. Enabling this setting can lead to significantly faster prediction speeds. This is enabled by default.

Recipes Settings

Include Specific Transformers

Select the transformer(s) that you want to use in the experiment. Use the **Check All/Uncheck All** button to quickly add or remove all transformers at once. **Note:** If you uncheck all transformers so that none is selected, Driverless AI will ignore this and will use the default list of transformers for that experiment. This list of transformers will vary for each experiment.

Include Specific Preprocessing Transformers

Specify which *transformers* to use for preprocessing before other transformers are activated. Preprocessing transformers can take any original features and output arbitrary features that are used by the normal layer of transformers.

Notes:

- Preprocessing transformers and all other layers of transformers are part of the Python and (if applicable) MOJO scoring packages.
- Any BYOR transformer recipe or native DAI transformer can be used as a preprocessing transformer. For example, a preprocessing transformer can perform interactions, string concatenations, or date extractions as a preprocessing step before the next layer of Date and DateTime transformations are performed.

Number of Pipeline Layers

Specify the number of pipeline layers. This value defaults to 1.

Note: This does not include the preprocessing layer specified by the *Include Specific Preprocessing Transformers* expert setting.

Include Specific Models

Specify the types of models that you want Driverless AI to build in the experiment. This list includes natively supported algorithms and models added with custom recipes.

Note: The ImbalancedLightGBM and ImbalancedXGBoostGBM models are closely tied with the *Sampling Method for Imbalanced Binary Classification Problems* option. Specifically:

- If the ImbalancedLightGBM and/or ImbalancedXGBoostGBM models are ENABLED and the *Sampling Method for Imbalanced Binary Classification Problems* is ENABLED (set to a value other than **off**), then Driverless AI will check your target imbalance fraction. If the target fraction proves to be above the allowed imbalance threshold, then sampling will be triggered.
- If the ImbalancedLightGBM and/or ImbalancedXGBoostGBM models are DISABLED and the *Sampling Method for Imbalanced Binary Classification Problems* option is ENABLED, then no special sampling technique will be performed.
- If the ImbalancedLightGBM and/or ImbalancedXGBoostGBM models are ENABLED and the *Sampling Method for Imbalanced Binary Classification Problems* is DISABLED, sampling will not be used, and these imbalanced models will be disabled.

Include Specific Scorers

Specify the scorer(s) that you want Driverless AI to include when running the experiment.

Scorer to Optimize Threshold to Be Used in Other Confusion-Matrix Based Scorers (For Binary Classification)

Specify the scorer used to optimize the binary probability threshold that is being used in related Confusion Matrix based scorers such as Precision, Recall, FalsePositiveRate, FalseDiscoveryRate, FalseOmissionRate, TrueNegativeRate, FalseNegativeRate, and NegativePredictiveValue. Select from the following:

- **Auto** (Default): Use this option to sync the threshold scorer with the scorer used for the experiment. If this is not possible, F1 is used.
- **F05**: More weight on precision, less weight on recall.
- **F1**: Equal weight on precision and recall.
- **F2**: Less weight on precision, more weight on recall.
- **MCC**: Use this option when all classes are equally important.

Include Specific Data Recipes During Experiment

Specify whether to include specific data recipes during the experiment.

Probability to Add Transformers

Specify the unnormalized probability to add genes or instances of transformers with specific attributes. If no genes can be added, other mutations are attempted. This value defaults to 0.5.

Probability to Add Best Shared Transformers

Specify the unnormalized probability to add genes or instances of transformers with specific attributes that have shown to be beneficial to other individuals within the population. This value defaults to 0.5.

Probability to Prune Transformers

Specify the unnormalized probability to prune genes or instances of transformers with specific attributes. This value defaults to 0.5.

Probability to Mutate Model Parameters

Specify the unnormalized probability to change model hyper parameters. This value defaults to 0.25.

Probability to Prune Weak Features

Specify the unnormalized probability to prune features that have low variable importance instead of pruning entire instances of genes/transfomers. This value defaults to 0.25.

Timeout in Minutes for Testing Acceptance of Each Recipe

Specify the number of minutes to wait until a recipe's acceptance testing is aborted. A recipe is rejected if acceptance testing is enabled and it times out. This value defaults to 20.0.

Whether to Skip Failures of Transformers

Specify whether to avoid failed transformers. This is enabled by default.

Whether to Skip Failures of Models

Specify whether to avoid failed models. Failures are logged according to the specified level for logging skipped failures. This is enabled by default.

Level to Log for Skipped Failures

Specify one of the following levels for the verbosity of log failure messages for skipped transformers or models:

- 0 = Log simple message
- 1 = Log code line plus message (Default)
- 2 = Log detailed stack traces

System Settings

Number of Cores to Use

Specify the number of cores to use for the experiment. Note that if you specify 0, all available cores will be used. Lower values can reduce memory usage but might slow down the experiment. This value defaults to 0.

Maximum Number of Cores to Use for Model Fit

Specify the maximum number of cores to use for a model's fit call. Note that if you specify 0, all available cores will be used. This value defaults to 10.

Maximum Number of Cores to Use for Model Predict

Specify the maximum number of cores to use for a model's predict call. Note that if you specify 0, all available cores will be used. This value defaults to 0.

Maximum Number of Cores to Use for Model Transform and Predict When Doing MLI, Autoreport

Specify the maximum number of cores to use for a model's transform and predict call when doing operations in the Driverless AI MLI GUI and the Driverless AI R and Python clients. Note that if you specify 0, all available cores will be used. This value defaults to 4.

Tuning Workers per Batch for CPU

Specify the number of workers used in CPU mode for tuning. A value of 0 uses the socket count, while a value of -1 uses all physical cores greater than or equal to 1 that count. This value defaults to 0.

Number of Workers for CPU Training

Specify the number of workers used in CPU mode for training:

- 0: Use socket count (Default)
- -1: Use all physical cores ≥ 1 that count

#GPUs/Experiment

Specify the number of GPUs to user per experiment. A value of -1 (default) specifies to use all available GPUs. Must be at least as large as the number of GPUs to use per model (or -1).

Num Cores/GPU

Specify the number of CPU cores per GPU. In order to have a sufficient number of cores per GPU, this setting limits the number of GPUs used. This value defaults to 4.

#GPUs/Model

Specify the number of GPUs to user per model, with -1 meaning all GPUs per model. In all cases, XGBoost tree and linear models use the number of GPUs specified per model, while LightGBM and Tensorflow revert to using 1 GPU/model and run multiple models on multiple GPUs. This value defaults to 1.

Note: FTRL does not use GPUs. Rulefit uses GPUs for parts involving obtaining the tree using LightGBM.

Num. of GPUs for Isolated Prediction/Transform

Specify the number of GPUs to use for predict for models and transform for transformers when running outside of fit/fit_transform. If predict or transform are called in the same process as fit/fit_transform, the number of GPUs will match. New processes will use this count for applicable models and transformers. Note that enabling tensorflow_nlp_have_gpus_in_production will override this setting for relevant TensorFlow NLP transformers. This value defaults to 0.

Note: When GPUs are used, TensorFlow and PyTorch models and transformers always predict on GPU.

Max Number of Threads to Use for datatable and OpenBLAS for Munging and Model Training

Specify the maximum number of threads to use for datatable and OpenBLAS during data munging (applied on a per process basis):

- 0 = Use all threads
- -1 = Automatically select number of threads (Default)

Max Number of Threads to Use for datatable Read and Write of Files

Specify the maximum number of threads to use for datatable during data reading and writing (applied on a per process basis):

- 0 = Use all threads
- -1 = Automatically select number of threads (Default)

Max Number of Threads to Use for datatable Stats and OpenBLAS

Specify the maximum number of threads to use for datatable stats and OpenBLAS (applied on a per process basis):

- 0 = Use all threads
- -1 = Automatically select number of threads (Default)

GPU Starting ID

Specify Which gpu_id to start with. If using CUDA_VISIBLE_DEVICES=... to control GPUs (preferred method), gpu_id=0 is the first in that restricted list of devices. For example, if CUDA_VISIBLE_DEVICES='4, 5' then gpu_id_start=0 will refer to device #4.

From expert mode, to run 2 experiments, each on a distinct GPU out of 2 GPUs, then:

- Experiment#1: num_gpus_per_model=1, num_gpus_per_experiment=1, gpu_id_start=0
- Experiment#2: num_gpus_per_model=1, num_gpus_per_experiment=1, gpu_id_start=1

From expert mode, to run 2 experiments, each on a distinct GPU out of 8 GPUs, then:

- Experiment#1: num_gpus_per_model=1, num_gpus_per_experiment=4, gpu_id_start=0
- Experiment#2: num_gpus_per_model=1, num_gpus_per_experiment=4, gpu_id_start=4

To run on all 4 GPUs/model, then

- Experiment#1: num_gpus_per_model=4, num_gpus_per_experiment=4, gpu_id_start=0
- Experiment#2: num_gpus_per_model=4, num_gpus_per_experiment=4, gpu_id_start=4

If num_gpus_per_model!=1, global GPU locking is disabled. This is because the underlying algorithms do not support arbitrary gpu ids, only sequential ids, so be sure to set this value correctly to avoid overlap across all experiments by all users.

More information is available at: <https://github.com/NVIDIA/nvidia-docker/wiki/nvidia-docker#gpu-isolation> Note that gpu selection does not wrap, so gpu_id_start + num_gpus_per_model must be less than the number of visible GPUs.

Enable Detailed Traces

Specify whether to enable detailed tracing in Driverless AI trace when running an experiment. This is disabled by default.

Enable Debug Log Level

If enabled, the log files will also include debug logs. This is disabled by default.

Enable Logging of System Information for Each Experiment

Specify whether to include system information such as CPU, GPU, and disk space at the start of each experiment log. Note that this information is already included in system logs. This is enabled by default.

22.1.3 Scorers

Classification or Regression

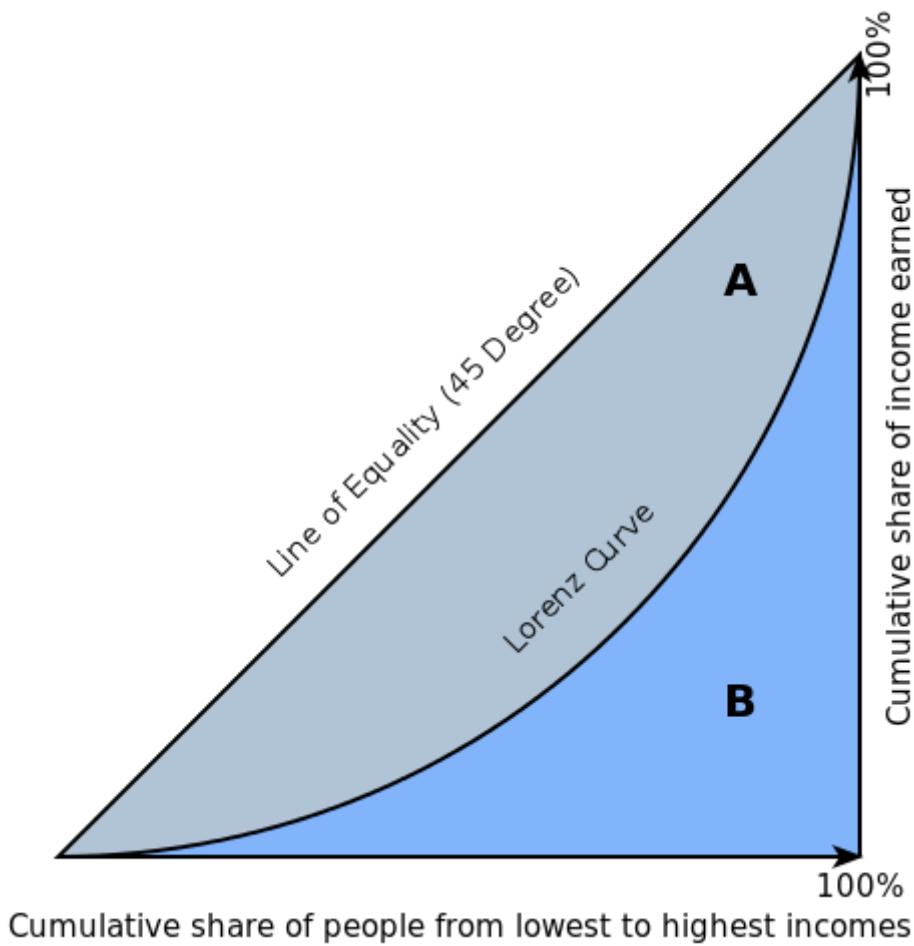
GINI (Gini Coefficient)

The Gini index is a well-established method to quantify the inequality among values of a frequency distribution, and can be used to measure the quality of a binary classifier. A Gini index of zero expresses perfect equality (or a totally useless classifier), while a Gini index of one expresses maximal inequality (or a perfect classifier).

The Gini index is based on the Lorenz curve. The Lorenz curve plots the true positive rate (y-axis) as a function of percentiles of the population (x-axis).

The Lorenz curve represents a collective of models represented by the classifier. The location on the curve is given by the probability threshold of a particular model. (i.e., Lower probability thresholds for classification typically lead to more true positives, but also to more false positives.)

The Gini index itself is independent of the model and only depends on the Lorenz curve determined by the distribution of the scores (or probabilities) obtained from the classifier.



Regression

R2 (R Squared)

The R2 value represents the degree that the predicted value and the actual value move in unison. The R2 value varies between 0 and 1 where 0 represents no correlation between the predicted and actual value and 1 represents complete correlation.

Calculating the R2 value for linear models is mathematically equivalent to $1 - SSE/SST$ (or $1 - \text{residual sum of squares/total sum of squares}$). For all other models, this equivalence does not hold, so the $1 - SSE/SST$ formula cannot be used. In some cases, this formula can produce negative R2 values, which is mathematically impossible for a real number. Because Driverless AI does not necessarily use linear models, the R2 value is calculated using the squared Pearson correlation coefficient.

R2 equation:

$$R2 = \frac{\sum_{i=1}^n (x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^n (x_i - \bar{x})^2 \sum_{i=1}^n (y_i - \bar{y})^2}}$$

Where:

- x is the predicted target value

- y is the actual target value

MSE (Mean Squared Error)

The MSE metric measures the average of the squares of the errors or deviations. MSE takes the distances from the points to the regression line (these distances are the “errors”) and squaring them to remove any negative signs. MSE incorporates both the variance and the bias of the predictor.

MSE also gives more weight to larger differences. The bigger the error, the more it is penalized. For example, if your correct answers are 2,3,4 and the algorithm guesses 1,4,3, then the absolute error on each one is exactly 1, so squared error is also 1, and the MSE is 1. But if the algorithm guesses 2,3,6, then the errors are 0,0,2, the squared errors are 0,0,4, and the MSE is a higher 1.333. The smaller the MSE, the better the model’s performance. (**Tip:** MSE is sensitive to outliers. If you want a more robust metric, try mean absolute error (MAE).)

MSE equation:

$$MSE = \frac{1}{N} \sum_{i=1}^N (y_i - \hat{y}_i)^2$$

RMSE (Root Mean Squared Error)

The RMSE metric evaluates how well a model can predict a continuous value. The RMSE units are the same as the predicted target, which is useful for understanding if the size of the error is of concern or not. The smaller the RMSE, the better the model’s performance. (**Tip:** RMSE is sensitive to outliers. If you want a more robust metric, try mean absolute error (MAE).)

RMSE equation:

$$RMSE = \sqrt{\frac{1}{N} \sum_{i=1}^N (y_i - \hat{y}_i)^2}$$

Where:

- N is the total number of rows (observations) of your corresponding dataframe.
- y is the actual target value.
- \hat{y} is the predicted target value.

RMSLE (Root Mean Squared Logarithmic Error)

This metric measures the ratio between actual values and predicted values and takes the log of the predictions and actual values. Use this instead of RMSE if an under-prediction is worse than an over-prediction. You can also use this when you don’t want to penalize large differences when both of the values are large numbers.

RMSLE equation:

$$RMSLE = \sqrt{\frac{1}{N} \sum_{i=1}^N \left(\ln\left(\frac{y_i + 1}{\hat{y}_i + 1}\right) \right)^2}$$

Where:

- N is the total number of rows (observations) of your corresponding dataframe.

- y is the actual target value.
- \hat{y} is the predicted target value.

RMSPE (Root Mean Square Percentage Error)

This metric is the RMSE expressed as a percentage. The smaller the RMSPE, the better the model performance.

RMSPE equation:

$$RMSPE = \sqrt{\frac{1}{N} \sum_{i=1}^N \frac{(y_i - \hat{y}_i)^2}{(y_i)^2}}$$

MAE (Mean Absolute Error)

The mean absolute error is an average of the absolute errors. The MAE units are the same as the predicted target, which is useful for understanding whether the size of the error is of concern or not. The smaller the MAE the better the model's performance. (**Tip:** MAE is robust to outliers. If you want a metric that is sensitive to outliers, try root mean squared error (RMSE).)

MAE equation:

$$MAE = \frac{1}{N} \sum_{i=1}^N |x_i - x|$$

Where:

- N is the total number of errors
- $|x_i - x|$ equals the absolute errors.

MAPE (Mean Absolute Percentage Error)

MAPE measures the size of the error in percentage terms. It is calculated as the average of the unsigned percentage error.

MAPE equation:

$$MAPE = \left(\frac{1}{N} \sum \frac{|Actual - Forecast|}{|Actual|} \right) * 100$$

Because the MAPE measure is in percentage terms, it gives an indication of how large the error is across different scales. Consider the following example:

Actual	Predicted	Absolute Error	Absolute Percentage Error
5	1	4	80%
15,000	15,004	4	0.03%

Both records have an absolute error of 4, but this error could be considered “small” or “big” when you compare it to the actual value.

SMAPE (Symmetric Mean Absolute Percentage Error)

Unlike the MAPE, which divides the absolute errors by the absolute actual values, the SMAPE divides by the mean of the absolute actual and the absolute predicted values. This is important when the actual values can be 0 or near 0. Actual values near 0 cause the MAPE value to become infinitely high. Because SMAPE includes both the actual and the predicted values, the SMAPE value can never be greater than 200%.

Consider the following example:

Actual	Predicted
0.01	0.05
0.03	0.04

The MAPE for this data is 216.67% but the SMAPE is only 80.95%.

Both records have an absolute error of 4, but this error could be considered “small” or “big” when you compare it to the actual value.

MER (Median Error Rate or Median Absolute Percentage Error)

MER measures the median size of the error in percentage terms. It is calculated as the median of the unsigned percentage error.

MER equation:

$$MER = \left(\text{median} \frac{|Actual - Forecast|}{|Actual|} \right) * 100$$

Because the MER is the median, half the scored population has a lower absolute percentage error than the MER, and half the population has a larger absolute percentage error than the MER.

Classification

MCC (Matthews Correlation Coefficient)

The goal of the MCC metric is to represent the confusion matrix of a model as a single number. The MCC metric combines the true positives, false positives, true negatives, and false negatives using the equation described below.

A Driverless AI model will return probabilities, not predicted classes. To convert probabilities to predicted classes, a threshold needs to be defined. Driverless AI iterates over possible thresholds to calculate a confusion matrix for each threshold. It does this to find the maximum MCC value. Driverless AI’s goal is to continue increasing this maximum MCC.

Unlike metrics like Accuracy, MCC is a good scorer to use when the target variable is imbalanced. In the case of imbalanced data, high Accuracy can be found by simply predicting the majority class. Metrics like Accuracy and F1 can be misleading, especially in the case of imbalanced data, because they do not consider the relative size of the four confusion matrix categories. MCC, on the other hand, takes the proportion of each class into account. The MCC value ranges from -1 to 1 where -1 indicates a classifier that predicts the opposite class from the actual value, 0 means the classifier does no better than random guessing, and 1 indicates a perfect classifier.

MCC equation:

$$MCC = \frac{TP \times TN - FP \times FN}{\sqrt{(TP + FP)(TP + FN)(TN + FP)(TN + FN)}}$$

F05, F1, and F2

A Driverless AI model will return probabilities, not predicted classes. To convert probabilities to predicted classes, a threshold needs to be defined. Driverless AI iterates over possible thresholds to calculate a confusion matrix for each threshold. It does this to find the maximum F metric value. Driverless AI's goal is to continue increasing this maximum F metric.

The **F1** score provides a measure for how well a binary classifier can classify positive cases (given a threshold value). The F1 score is calculated from the harmonic mean of the precision and recall. An F1 score of 1 means both precision and recall are perfect and the model correctly identified all the positive cases and didn't mark a negative case as a positive case. If either precision or recall are very low it will be reflected with a F1 score closer to 0.

F1 equation:

$$F1 = 2 \left(\frac{(precision) (recall)}{precision + recall} \right)$$

Where:

- *precision* is the positive observations (true positives) the model correctly identified from all the observations it labeled as positive (the true positives + the false positives).
- *recall* is the positive observations (true positives) the model correctly identified from all the actual positive cases (the true positives + the false negatives).

The **F0.5** score is the weighted harmonic mean of the precision and recall (given a threshold value). Unlike the F1 score, which gives equal weight to precision and recall, the F0.5 score gives more weight to precision than to recall. More weight should be given to precision for cases where False Positives are considered worse than False Negatives. For example, if your use case is to predict which products you will run out of, you may consider False Positives worse than False Negatives. In this case, you want your predictions to be very precise and only capture the products that will definitely run out. If you predict a product will need to be restocked when it actually doesn't, you incur cost by having purchased more inventory than you actually need.

F05 equation:

$$F0.5 = 1.25 \left(\frac{(precision) (recall)}{0.25 precision + recall} \right)$$

Where:

- *precision* is the positive observations (true positives) the model correctly identified from all the observations it labeled as positive (the true positives + the false positives).
- *recall* is the positive observations (true positives) the model correctly identified from all the actual positive cases (the true positives + the false negatives).

The **F2** score is the weighted harmonic mean of the precision and recall (given a threshold value). Unlike the F1 score, which gives equal weight to precision and recall, the F2 score gives more weight to recall than to precision. More weight should be given to recall for cases where False Negatives are considered worse than False Positives. For example, if your use case is to predict which customers will churn, you may consider False Negatives worse than False Positives. In this case, you want your predictions to capture all of the customers that will churn. Some of these customers may not be at risk for churning, but the extra attention they receive is not harmful. More importantly, no customers actually at risk of churning have been missed.

F2 equation:

$$F2 = 5 \left(\frac{(precision) (recall)}{4 precision + recall} \right)$$

Where:

- *precision* is the positive observations (true positives) the model correctly identified from all the observations it labeled as positive (the true positives + the false positives).
- *recall* is the positive observations (true positives) the model correctly identified from all the actual positive cases (the true positives + the false negatives).

Accuracy

In binary classification, Accuracy is the number of correct predictions made as a ratio of all predictions made. In multiclass classification, the set of labels predicted for a sample must exactly match the corresponding set of labels in `y_true`.

A Driverless AI model will return probabilities, not predicted classes. To convert probabilities to predicted classes, a threshold needs to be defined. Driverless AI iterates over possible thresholds to calculate a confusion matrix for each threshold. It does this to find the maximum Accuracy value. Driverless AI's goal is to continue increasing this maximum Accuracy.

Accuracy equation:

$$\text{Accuracy} = \left(\frac{\text{number correctly predicted}}{\text{number of observations}} \right)$$

Logloss

The logarithmic loss metric can be used to evaluate the performance of a binomial or multinomial classifier. Unlike AUC which looks at how well a model can classify a binary target, logloss evaluates how close a model's predicted values (uncalibrated probability estimates) are to the actual target value. For example, does a model tend to assign a high predicted value like .80 for the positive class, or does it show a poor ability to recognize the positive class and assign a lower predicted value like .50? Logloss can be any value greater than or equal to 0, with 0 meaning that the model correctly assigns a probability of 0% or 100%.

Binary classification equation:

$$\text{Logloss} = - \frac{1}{N} \sum_{i=1}^N w_i (y_i \ln(p_i) + (1 - y_i) \ln(1 - p_i))$$

Multiclass classification equation:

$$\text{Logloss} = - \frac{1}{N} \sum_{i=1}^N \sum_{j=1}^C w_i (y_{i,j} \ln(p_{i,j}))$$

Where:

- N is the total number of rows (observations) of your corresponding dataframe.
- w is the per row user-defined weight (defaults is 1).
- C is the total number of classes ($C=2$ for binary classification).
- p is the predicted value (uncalibrated probability) assigned to a given row (observation).
- y is the actual target value.

AUC (Area Under the Receiver Operating Characteristic Curve)

This model metric is used to evaluate how well a binary classification model is able to distinguish between true positives and false positives. For multi-class problems, this score is computed by micro-averaging the ROC curves for each class. Use MACROAUC if you prefer the macro average.

An AUC of 1 indicates a perfect classifier, while an AUC of .5 indicates a poor classifier whose performance is no better than random guessing.

AUCPR (Area Under the Precision-Recall Curve)

This model metric is used to evaluate how well a binary classification model is able to distinguish between precision recall pairs or points. These values are obtained using different thresholds on a probabilistic or other continuous-output classifier. AUCPR is an average of the precision-recall weighted by the probability of a given threshold.

The main difference between AUC and AUCPR is that AUC calculates the area under the ROC curve and AUCPR calculates the area under the Precision Recall curve. The Precision Recall curve does not care about True Negatives. For imbalanced data, a large quantity of True Negatives usually overshadows the effects of changes in other metrics like False Positives. The AUCPR will be much more sensitive to True Positives, False Positives, and False Negatives than AUC. As such, AUCPR is recommended over AUC for highly imbalanced data.

MACROAUC (Macro Average of Areas Under the Receiver Operating Characteristic Curves)

For multiclass classification problems, this score is computed by macro-averaging the ROC curves for each class (one per class). The area under the curve is a constant. A MACROAUC of 1 indicates a perfect classifier, while a MACROAUC of .5 indicates a poor classifier whose performance is no better than random guessing. This option is not available for binary classification problems.

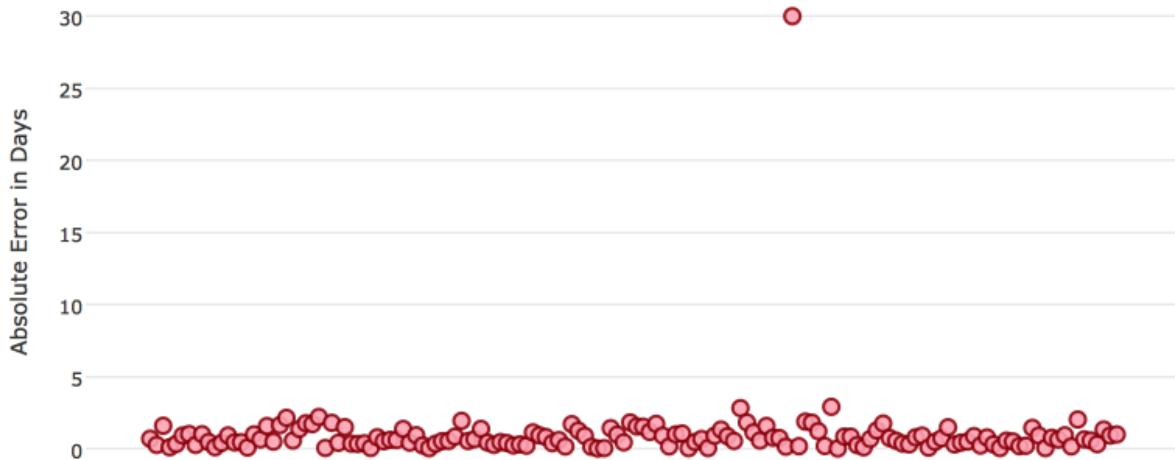
Scorer Best Practices - Regression

When deciding which scorer to use in a regression problem, some main questions to ask are:

- Do you want your scorer sensitive to outliers?
- What unit should the scorer be in?

Sensitive to Outliers

Certain scorers are more sensitive to outliers. When a scorer is sensitive to outliers, it means that it is important that the model predictions are never “very” wrong. For example, let’s say we have an experiment predicting number of days until an event. The graph below shows the absolute error in our predictions.



Usually our model is very good. We have an absolute error less than 1 day about 70% of the time. There is one instance, however, where our model did very poorly. We have one prediction that was 30 days off.

Instances like this will more heavily penalize scorers that are sensitive to outliers. If we do not care about these outliers in poor performance as long as we typically have a very accurate prediction, then we would want to select a scorer that is robust to outliers. We can see this reflected in the behavior of the scorers: MSE and RMSE.

	MSE	RMSE
Outlier	0.99	2.64
No Outlier	0.80	1.0

Calculating the RMSE and MSE on our error data, the RMSE is more than twice as large as the MSE because RMSE is sensitive to outliers. If we remove the one outlier record from our calculation, RMSE drops down significantly.

Performance Units

Different scorers will show the performance of the Driverless AI experiment in different units. Let's continue with our example where our target is to predict the number of days until an event. Some possible performance units are:

- Same as target: The unit of the scorer is in days
 - ex: MAE = 5 means the model predictions are off by 5 days on average
- Percent of target: The unit of the scorer is the percent of days
 - ex: MAPE = 10% means the model predictions are off by 10 percent on average
- Square of target: The unit of the scorer is in days squared
 - ex: MSE = 25 means the model predictions are off by 5 days on average (square root of 25 = 5)

Comparison

Metric	Units	Sensitive to Outliers	Tip
R2	scaled between 0 and 1	No	use when you want performance scaled between 0 and 1
MSE	square of target	Yes	
RMSE	same as target	Yes	
RM-SLE	log of target	Yes	
RM-SPE	percent of target	Yes	use when target values are across different scales
MAE	same as target	No	
MAPE	percent of target	No	use when target values are across different scales
SMAPE	percent of target divided by 2	No	use when target values are close to 0

Scorer Best Practices - Classification

When deciding which scorer to use in a classification problem some main questions to ask are:

- Do you want the scorer to evaluate the predicted probabilities or the classes that those probabilities can be converted to?
- Is your data imbalanced?

Scorer Evaluates Probabilities or Classes

The final output of a Driverless AI model is a predicted probability that a record is in a particular class. The scorer you choose will either evaluate how accurate the probability is or how accurate the assigned class is from that probability.

Choosing this depends on the use of the Driverless AI model. Do we want to use the probabilities or do we want to convert those probabilities into classes? For example, if we are predicting whether a customer will churn, we may take the predicted probabilities and turn them into classes - customers who will churn vs customers who won't churn. If we are predicting the expected loss of revenue, we will instead use the predicted probabilities (predicted probability of churn * value of customer).

If your use case requires a class assigned to each record, you will want to select a scorer that evaluates the model's performance based on how well it classifies the records. If your use case will use the probabilities, you will want to select a scorer that evaluates the model's performance based on the predicted probability.

Robust to Imbalanced Data

For certain use cases, positive classes may be very rare. In these instances, some scorers can be misleading. For example, if I have a use case where 99% of the records have Class = No, then a model which always predicts No will have 99% accuracy.

For these use cases, it is best to select a metric that does not include True Negatives or considers relative size of the True Negatives like AUCPR or MCC.

Comparison

Metric	Evaluation Based On	Tip
MCC	Class	all classes are equally weighted
F1	Class	equal weight on precision and recall
F0.5	Class	more weight on precision, less weight on recall
F2	Class	more weight on recall, less weight on precision
Accuracy	Class	highly interpretable
Logloss	Probability	optimizes probabilities
AUC	Class	optimizes sort order of predictions
AUCPR	Class	good for imbalanced data

22.2 New Experiments

- Run an experiment by selecting [Click for Actions] button beside the dataset that you want to use. Click Predict to begin an experiment.

The screenshot shows the H2O.ai interface for managing datasets. At the top, there's a navigation bar with links like PROJECTS, DATASETS, AUTOVIZ, EXPERIMENTS, DIAGNOSTICS, MLI, DEPLOYMENTS, and RESOURCES. Below the navigation is a search bar labeled '+ ADD DATASET (OR DRAG & DROP)'. The main area is titled 'Datasets' and lists three datasets:

Name	Path	Size	Rows	Columns	Status
CreditCard_train.csv	...ain.csv.1574699349.6031156.bin	2MB	17K	25	[Click for Actions]
CreditCard_test.csv	...est.csv.1574699347.1410177.bin	459KB	5K	25	DETAILS [Click for Actions]
CreditCard_valid.csv	...lid.csv.1574699346.0325682.bin	228KB	2K	25	VISUALIZE [Click for Actions]

A context menu is open over the first dataset row, showing options: SPLIT, PREDICT, RENAME, DOWNLOAD, and DELETE.

- The Experiment Settings form displays and auto-fills with the selected dataset. Optionally enter a custom name for this experiment. If you do not add a name, Driverless AI will create one for you.
- Optionally specify a validation dataset and/or a test dataset.
 - The validation set is used to tune parameters (models, features, etc.). If a validation dataset is not provided, the training data is used (with holdout splits). If a validation dataset is provided, training data is not used for parameter tuning - only for training. A validation dataset can help to improve the generalization performance on shifting data distributions.
 - The test dataset is used for the final stage scoring and is the dataset for which model metrics will be computed against. Test set predictions will be available at the end of the experiment. This dataset is not used during training of the modeling pipeline.

Keep in mind that these datasets must have the same number of columns as the training dataset. Also note that if provided, the validation set is not sampled down, so it can lead to large memory usage, even if accuracy=1 (which reduces the train size).

4. Specify the target (response) column. Note that not all explanatory functionality will be available for multiclass classification scenarios (scenarios with more than two outcomes). When the target column is selected, Driverless AI automatically provides the target column type and the number of rows. If this is a classification problem, then the UI shows unique and frequency statistics (Target Freq/Most Freq) for numerical columns. If this is a regression problem, then the UI shows the dataset mean and standard deviation values.

Notes Regarding Frequency:

- For data imported in versions <= 1.0.19, TARGET FREQ and MOST FREQ both represent the count of the least frequent class for numeric target columns and the count of the most frequent class for categorical target columns.
 - For data imported in versions 1.0.20-1.0.22, TARGET FREQ and MOST FREQ both represent the frequency of the target class (second class in lexicographic order) for binomial target columns; the count of the most frequent class for categorical multinomial target columns; and the count of the least frequent class for numeric multinomial target columns.
 - For data imported in version 1.0.23 (and later), TARGET FREQ is the frequency of the target class for binomial target columns, and MOST FREQ is the most frequent class for multinomial target columns.
5. The next step is to set the parameters and settings for the experiment. (Refer to the [Experiment Settings](#) section for more information about these settings.) You can set the parameters individually, or you can let Driverless AI infer the parameters and then override any that you disagree with. Available parameters and settings include the following:
 - Dropped Columns: The columns we do not want to use as predictors such as ID columns, columns with data leakage, etc.
 - **Weight Column**: The column that indicates the per row observation weights. If “None” is specified, each row will have an observation weight of 1.
 - **Fold Column**: The column that indicates the fold. If “None” is specified, the folds will be determined by Driverless AI. This is set to “Disabled” if a validation set is used.
 - **Time Column**: The column that provides a time order, if applicable. If “AUTO” is specified, Driverless AI will auto-detect a potential time order. If “OFF” is specified, auto-detection is disabled. This is set to “Disabled” if a validation set is used.
 - Specify the **Scorer** to use for this experiment. The available scorers vary based on whether this is a classification or regression experiment. Scorers include:
 - Regression: GINI, MAE, MAPE, MER, MSE, R2, RMSE (default), RMSLE, RMSPE, SMAPE, TOPDECILE
 - Classification: ACCURACY, AUC (default), AUCPR, F05, F1, F2, GINI, LOGLOSS, MACROAUC, MCC
 - Specify a desired relative Accuracy from 1 to 10
 - Specify a desired relative Time from 1 to 10
 - Specify a desired relative Interpretability from 1 to 10

Driverless AI will automatically infer the best settings for Accuracy, Time, and Interpretability and provide you with an experiment preview based on those suggestions. If you adjust these knobs, the experiment preview will automatically update based on the new settings.

Expert Settings (optional):

- Optionally specify additional expert settings for the experiment. Refer to the [Expert Settings](#) section for more information about these settings. The default values for these options are derived from the environment variables in the config.toml file. Refer to the [Setting Environment Variables](#) section for more information.

Expert Experiment Settings

+ UPLOAD CUSTOM RECIPE + LOAD CUSTOM RECIPE FROM URL OFFICIAL RECIPES (EXTERNAL) TOML

Start typing to filter out items

EXPERIMENT	MODEL	FEATURES	TIMESERIES	NLP	IMAGE	RECIPES	SYSTEM
Max. runtime in minutes before triggering the 'Finish' button. Approximately enforced. (0 = disabled). 1440	Max. runtime in minutes before triggering the 'Abort' button.(0 = disabled) 10080	Kaggle key		Pipeline Building Recipe AUTO ▾	Kaggle submission timeout in seconds 120	Enable genetic algorithm for selection and tuning of features and models AUTO ON OFF	
Make MOJO scoring pipeline AUTO ON OFF	Attempt to reduce the size of the MOJO DISABLED			Measure MOJO scoring latency AUTO ON OFF		Make Python scoring pipeline AUTO ON OFF	Timeout in seconds to wait for MOJO creation at end of experiment. 1800
Number of parallel workers to use during MOJO creation (-1 = all cores) -1	Make pipeline visualization AUTO ON OFF	Reproducibility Level 1	Random seed 1234	Make Autoreport ENABLED	Allow different sets of classes across all train/validation fold splits ENABLED	Min. number of rows needed to run experiment 100	Max. number of classes for classification problems 200
Max. number of classes to compute ROC and confusion matrix for classification	Max. number of classes to show in GUI for ROC/CM reduction technique for large class						

SAVE **CANCEL**

Additional settings (optional):

- **Classification or Regression** button. Driverless AI automatically determines the problem type based on the response column. Though not recommended, you can override this setting by clicking this button.
 - **Reproducible:** This button allows you to build an experiment with a random seed and get reproducible results. If this is disabled (default), then results will vary between runs.
 - **Enable GPUs:** Specify whether to enable GPUs. (Note that this option is ignored on CPU-only systems.)
6. After your settings are made, review the Experiment Preview to learn what each of the settings means. **Note:** When changing the algorithms used via [Expert Settings](#), you may notice that those changes are not applied. Driverless AI determines whether to include models and/or recipes based on a hierarchy of those expert settings. Refer to the [Why do my selected algorithms not show up in the Experiment Preview?](#) FAQ for more information.

The screenshot shows the H2O.ai Experiment interface. At the top, it says "DRIVERLESS AI 1.8.2 – AI TO DO AI Licensed to H2O.ai (SN37). Current User – ANG". The main area is titled "Experiment" and shows the following details:

- EXPERIMENT SETUP**
 - DISPLAY NAME: Display name
 - ROWS: 17K
 - COLUMNS: 25
 - DROPPED COLUMNS: 1
 - VALIDATION DATASET: Yes (CreditCard_valid.csv)
 - TEST DATASET: Yes (CreditCard_test.csv)
 - TARGET COLUMN: default payment next
 - FOLD COLUMN: Disabled
 - WEIGHT COLUMN: --
 - TIME COLUMN: Disabled
 - TYPE: bool
 - COUNT: 16784
 - UNIQUE: 2
 - TARGET FREQ: 3740
- TRAINING SETTINGS**
 - ACCURACY: 6
 - TIME: 3
 - INTERPRETABILITY: 7
 - SCORER: AUC
- EXPERT SETTINGS**
 - CLASSIFICATION
 - REPRODUCIBLE
 - GPUS ENABLED

A callout box on the left side provides a summary of the settings:

- ACCURACY**
 - Training data size: **16,784 rows, 24 cols**
 - Feature evolution: **[DecisionTree, LightGBM, XGBoostGBM], external validation, 2 reps**
 - Final pipeline: **Ensemble (2 models), external validation**
- TIME**
 - Feature evolution: **4 Individuals, up to 36 Iterations**
 - Early stopping: **After 5 iterations of no improvement**
- INTERPRETABILITY**
 - Feature pre-pruning strategy: Permutation importance FS
 - Monotonicity constraints: **enabled**
 - Feature engineering search space: **[CVCatNumEncode, CVTargetEncode, Frequent, Interactions, NumCatTE, NumToCatTE, NumToCatWoEMonotonic, NumToCatWoE, Original, WeightOfEvidence]**
- (DecisionTree, LightGBM, XGBoostGBM) models to train:**
 - Model & feature tuning: **144**
 - Feature evolution: **384**
 - Final pipeline: **2**
- Estimated runtime: minutes**
Auto-click Finish/Abort if not done in: **1 day/7 days**

At the bottom right is a large green button labeled "LAUNCH EXPERIMENT".

7. Click **Launch Experiment** to start the experiment.

The experiment launches with a randomly generated experiment name. You can change this name at anytime during or after the experiment. Mouse over the name of the experiment to view an edit icon, then type in the desired name.

As the experiment runs, a running status displays in the upper middle portion of the UI. First Driverless AI figures out the backend and determines whether GPUs are running. Then it starts parameter tuning, followed by feature engineering. Finally, Driverless AI builds the scoring pipeline.

22.2.1 Understanding the Experiment Page

In addition to the status, as an experiment is running, the UI also displays the following:

- Details about the dataset.
- The iteration data (internal validation) for each cross validation fold along with the specified scorer value. Click on a specific iteration or drag to view a range of iterations. Double click in the graph to reset the view. In this graph, each “column” represents one iteration of the experiment. During the iteration, Driverless AI will train n models. (This is called individuals in the experiment preview.) So for any column, you may see the score value for those n models for each iteration on the graph.
- The variable importance values. To view variable importance for a specific iteration, just select that iteration in the Iteration Data graph. The Variable Importance list will automatically update to show variable importance information for that iteration. Hover over an entry to view more info.

Notes:

- Transformed feature names are encoded as follows:

<transformation/gene_details_id>_<transformation_name>:<orig>:<...>:<orig>.<extra>

So in 32_NumToCatTE:BILL_AMT1:EDUCATION:MARRIAGE:SEX.0, for example:

- 32_ is the transformation index for specific transformation parameters.
- NumToCatTE is the transformation type.
- BILL_AMT1:EDUCATION:MARRIAGE:SEX represent original features used.
- 0 represents the likelihood encoding for target[0] after grouping by switch and making out-of-fold estimates. For multiclass experiments this value will be > 0. For binary experiments, this value will always be 0.
- When hovering over an entry, you may notice the term “Internal[...] specification.” This label is used for features that do not need to be translated/explained and ensures that all features are uniquely identified.

The values that display are specific to the variable importance of the model class:

- XGBoost and LightGBM: Gains Variable importance. Gain-based importance is calculated from the gains a specific variable brings to the model. In the case of a decision tree, the gain-based importance will sum up the gains that occurred whenever the data was split by the given variable. The gain-based importance is normalized between 0 and 1. If a variable is never used in the model, the gain-based importance will be 0.
- GLM: The variable importance is the absolute value of the coefficient for each predictor. The variable importance is normalized between 0 and 1. If a variable is never used in the model, the importance will be 0.
- TensorFlow: TensorFlow follows the Gedeon method described in this paper: <https://www.ncbi.nlm.nih.gov/pubmed/9327276>.
- RuleFit: Sums over a feature’s contribution to each rule. Specifically, Driverless AI:
 - Assigns all features to have zero importance.
 - Scans through all the rules. If a feature is in that rule, Driverless AI adds its contribution (i.e., the absolute values of a rule’s coefficient) to its overall feature importance.
 - Normalizes the importance.

The calculation for the shift of variable importance is determined by the ensemble level:

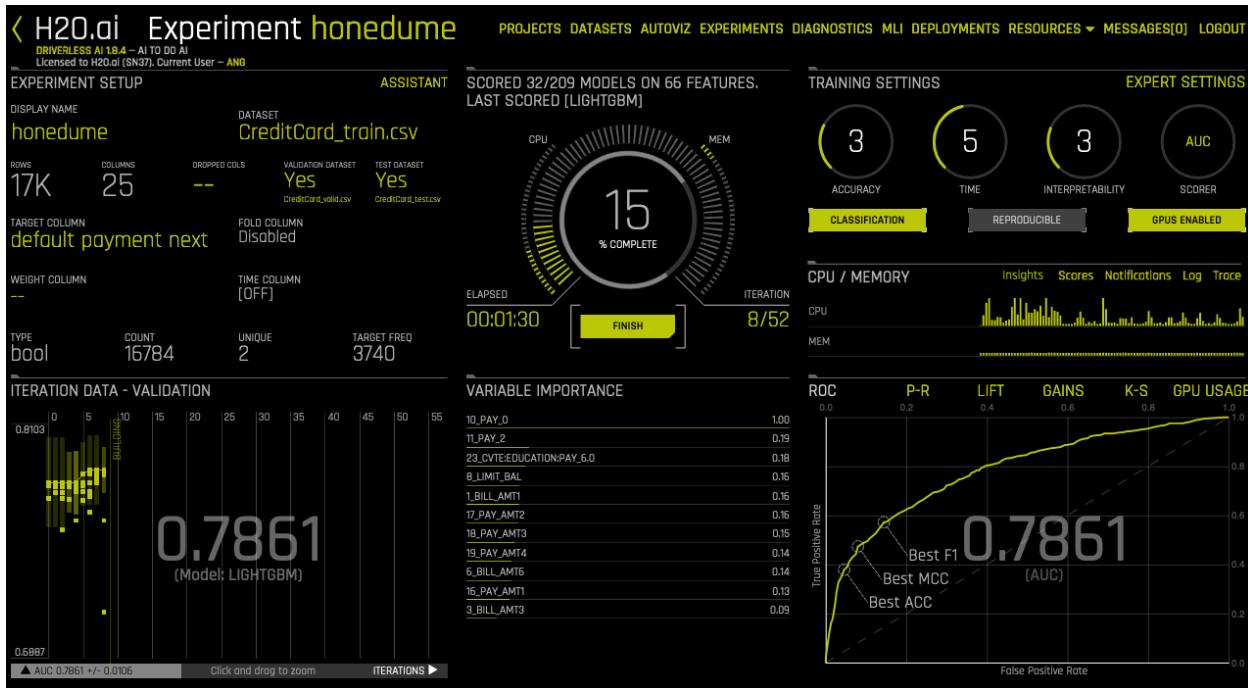
- Ensemble Level = 0: The shift is determined between the last best genetic algorithm (GA) and the single final model.
- Ensemble Level >=1: GA individuals used for the final model have variable importance blended with the model’s meta learner weights, and the final model itself has variable importance blended with its final weights. The shift of variable importance is determined between these two final variable importance blends.

This information is reported in the logs or in the GUI if the shift is beyond the absolute magnitude specified by the `max_num_varimp_shift_to_log` configuration option. The Experiment Summary also includes **experiment_features_shift** files that contain information about shift.

- CPU/Memory information along with *Insights* (for time-series experiments), *Scores*, Notifications, Logs, and Trace info. (Note that Trace is used for development/debugging and to show what the system is doing at that moment.)
- For classification problems, the lower right section includes a toggle between an ROC curve, Precision-Recall graph, Lift chart, Gains chart, and GPU Usage information (if GPUs are available). For regression problems,

the lower right section includes a toggle between a Residuals chart, an Actual vs. Predicted chart, and GPU Usage information (if GPUs are available). (Refer to the [Experiment Graphs](#) section for more information.) Upon completion, an Experiment Summary section will populate in the lower right section.

- The bottom portion of the experiment screen will show any warnings that Driverless AI encounters. You can hide this pane by clicking the x icon.



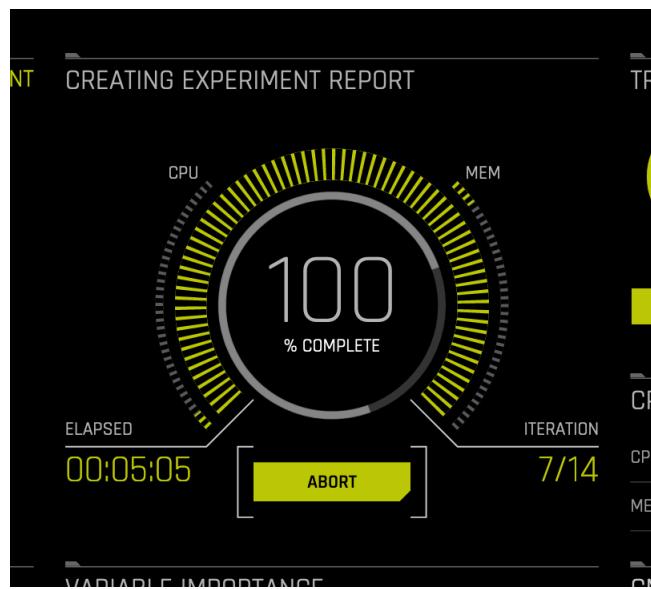
22.2.2 Finishing/Aborting Experiments

You can finish and/or abort experiments that are currently running.

- Finish** Click the **Finish** button to stop a running experiment. Driverless AI will end the experiment and then complete the ensembling and the deployment package.
- Abort:** After clicking **Finish**, you have the option to click **Abort**, which terminates the experiment. (You will be prompted to confirm the abort.) Aborted experiments will display on the Experiments page as Failed. You can restart aborted experiments by clicking the right side of the experiment, then selecting **Restart from Last Checkpoint**. This will start a new experiment based on the aborted one. Alternatively, you can start a new experiment based on the aborted one by selecting **New Model with Same Params**. Refer to [Checkpointing, Rerunning, and Retraining](#) for more information.

22.2.3 Aborting Experiment Report

The final step that Driverless AI performs during an experiment is to complete the experiment report. During this step, you can click **Abort** to skip this report.



22.2.4 “Pausing” an Experiment

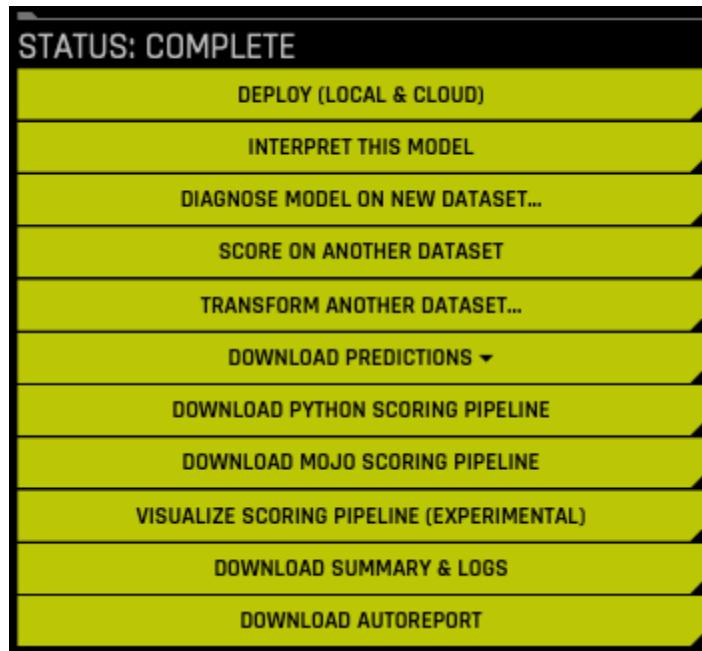
A trick for “pausing” an experiment is to:

1. Abort the experiment.
2. On the Experiments page, select **Restart from Last Checkpoint** for the aborted experiment.
3. On the Expert Settings page, specify 0 for the **Ensemble level for final modeling pipeline** option in the new experiment’s [Expert Settings](#).

22.3 Completed Experiment

22.3.1 Completed Experiment Actions

After an experiment status changes from RUNNING to COMPLETE, the UI provides you with several actions that you can perform:



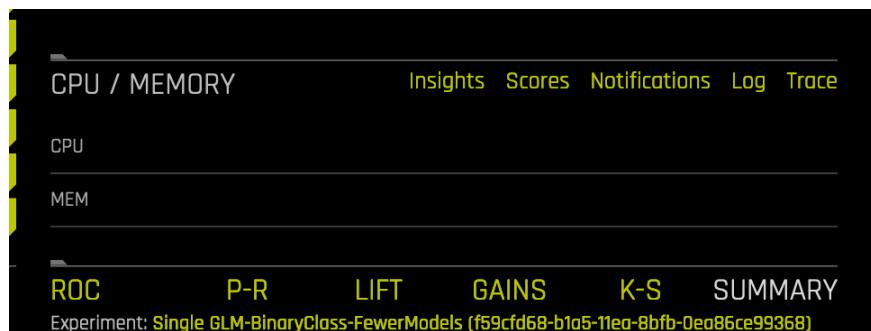
- **Deploy (Local and Cloud):** Refer to [Deploying the MOJO Pipeline](#).
- **Interpret this Model:** Refer to [Interpreting a Model](#). (Not supported for Image or multiclass Time Series experiments.)
- **Diagnose Model on New Dataset:** Refer to [Diagnosing a Model](#).
- **Score on Another Dataset:** Refer to [Score on Another Dataset](#).
- **Transform Another Dataset:** Refer to [Transform Another Dataset](#). (Not available for Time Series experiments.)
- **Download Predictions** dropdown:
 - **Training (Holdout) Predictions:** In csv format, available if a validation set was NOT provided.
 - **Validation Set Predictions:** In csv format, available if a validation set was provided.
 - **Test Set Predictions:** In csv format, available if a test dataset is used.
- **Download Python Scoring Pipeline:** A standalone Python scoring pipeline for H2O Driverless AI. Refer to [Driverless AI Standalone Python Scoring Pipeline](#).
- **Download MOJO Scoring Pipeline:** A standalone Model Object, Optimized scoring pipeline. Refer to [MOJO Scoring Pipelines](#). (Not available for TensorFlow or RuleFit models.)
- **Visualize Scoring Pipeline (Experimental):** Opens an experiment pipeline visualization page. Refer to [Visualizing the Scoring Pipeline](#).
- **Download Summary & Logs:** A zip file containing the following files. Refer to the [Experiment Summary](#) section for more information.
 - Experiment logs (regular and anonymized)
 - A summary of the experiment
 - The experiment features along with their relative importance

- Ensemble information
- An experiment preview
- Word version of an auto-generated report for the experiment
- A target transformations tuning leaderboard
- A tuning leaderboard
- **Download Autoreport:** A Word version of an auto-generated report for the experiment. This file is also available in the Experiment Summary zip file. Note that this option is not available for deprecated models. Refer to the [Experiment Autoreport](#) section for more information.

Note: The “Download” options above (with the exception of Autoreport) will appear as “Export” options if artifacts were enabled when Driverless AI was started. Refer to [Export Artifacts](#) for more information.

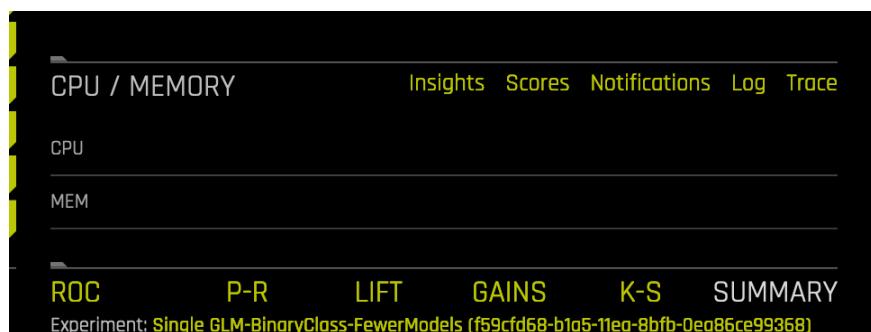
Experiment Insight and Scores

While an experiment is running, the UI provides you with options for viewing model insights (for time-series experiments) and model scores. Refer to [Model Insights](#) and [Model Scores](#) for more information.



22.4 Model Insights

For time series and Automatic Image Model experiments, you can view detailed insights while an experiment is running or after an experiment is complete by clicking on the **Insights** option.

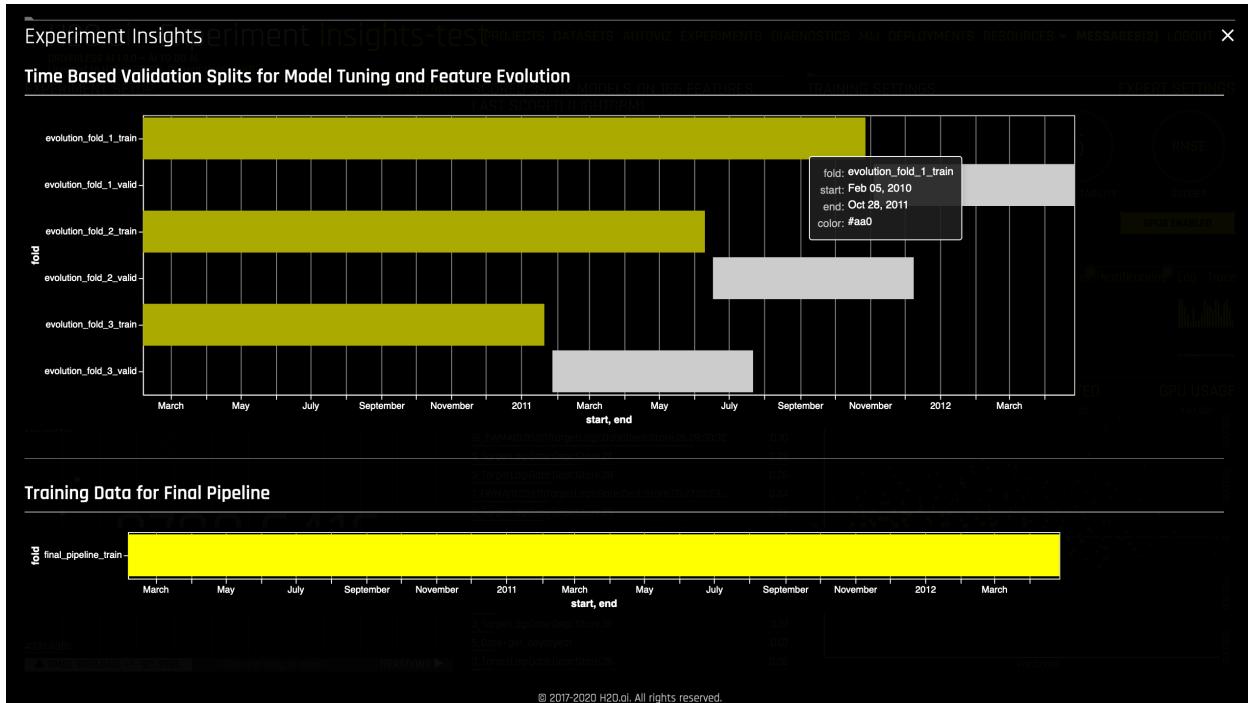


22.4.1 Time Series Insights

The Insights page for time series experiments includes the following graphs:

- Time Based Validation Splits for Model Tuning and Feature Evolution
- Training Data for Final Pipeline
- Time Based Back Testing of Final Pipeline on Training Data

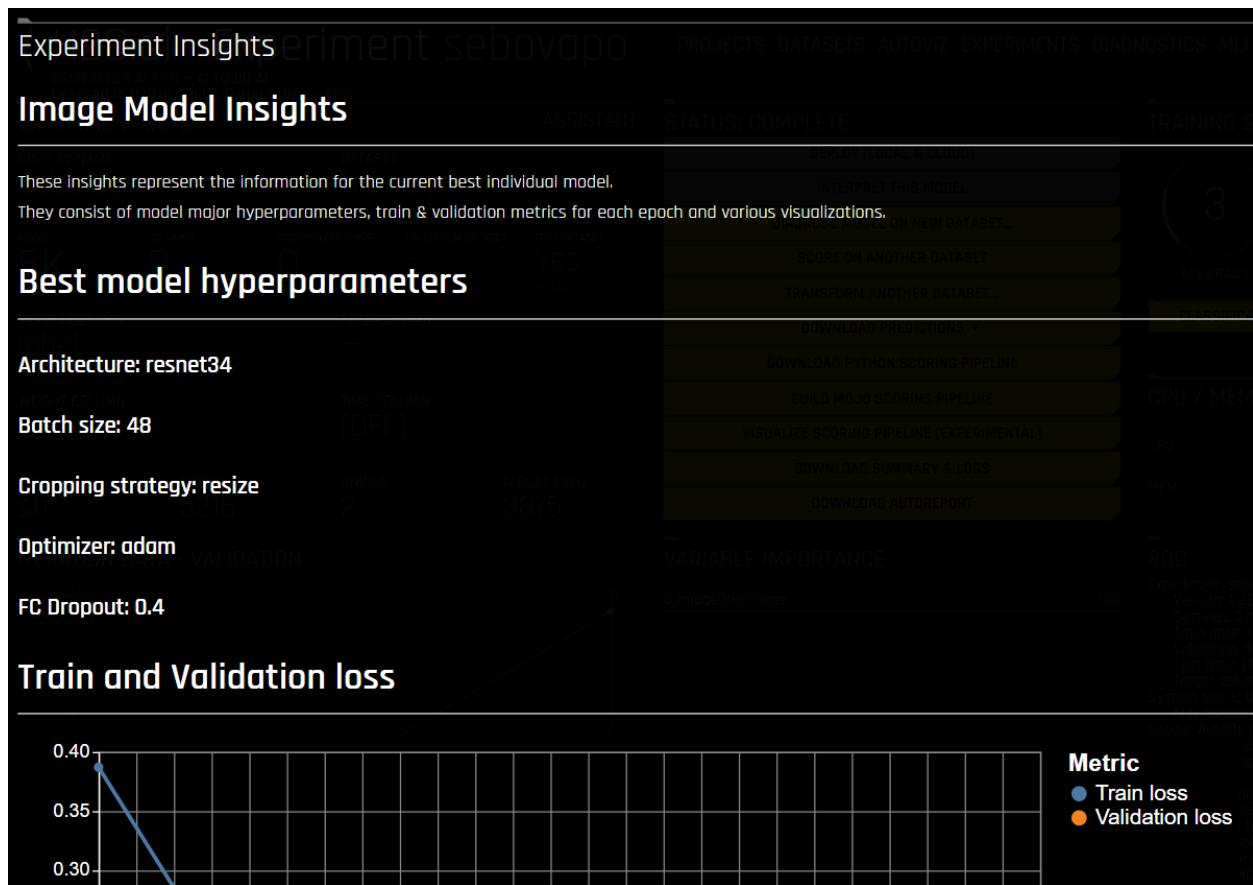
You can hover over a point in the graphs to view the dates in the graphs.



22.4.2 Automatic Image Model Insights

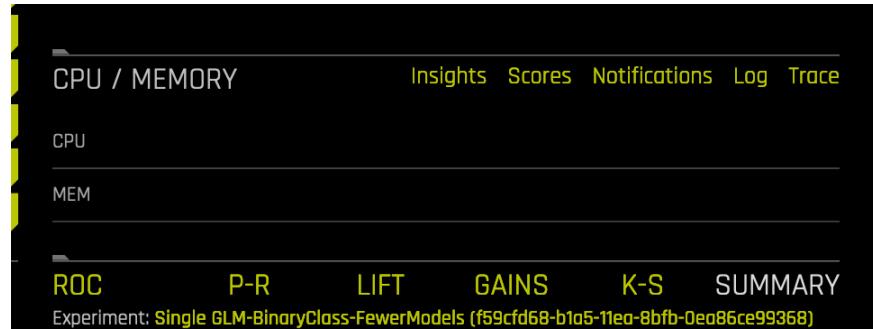
The Insights page for Image Model experiments contains the following information about the current best individual model:

- Best model hyperparameters
- Train and validation loss graph (by epoch)
- Validation Scorer graph
- Sample train and augmented train images
- Sample validation error images
- Sample Grad-CAM visualizations (only available for classification problems)



22.5 Model Scores

You can view detailed information about model scores after an experiment is complete by clicking on the **Scores** option.



The Model Scores page that opens includes the following tables:

- **Model and feature tuning leaderboard:** This leaderboard shows scoring information based on the scorer that was selected in the experiment. This information is also available in the **tuning_leaderboard.json** file of the *Experiment Summary*. You can download that file directly from the bottom of this table.

Model Scores											
Model and feature tuning leaderboard (scorer=AUC):											
Job Order	Job Status	Accuracy	Booster	Boosting Type	Colsample_Bytree	Debug_Verbose	Disable_Gpus	Dummy	Early_Stopping_Rounds	Early_Stopping_Threshold	Enable_Early_Stopping
9	PASSED	6	lightgbm	gbdt	0.8		False	False	20		True
5	PASSED	6	lightgbm	gbdt	0.8		False	False	20		True
6	PASSED	6	lightgbm	gbdt	0.8		False	False	20		True
10	PASSED	6	lightgbm	gbdt	0.8		False	False	20		True
19	PASSED	6	gbtree		0.8	0	False	False	20	0.0001	
2	PASSED	6	lightgbm	gbdt	0.4		False	False	20		True
14	PASSED	6	lightgbm	gbdt	0.55		False	False	20		True
0	PASSED	6	lightgbm	gbdt	0.3		False	False	20		True
20	PASSED	6	lightgbm	gbdt	0.8		False	False	20		True
3	PASSED	6	lightgbm	gbdt	0.3		False	False	20		True

- **Final pipeline scores across cross-validation folds and models:** This table shows the final pipeline scores across cross-validation folds and models. Note that if Constant Model was enabled (default), then that model is added in this table as a baseline (reference) only and will be dropped in most cases. This information is also included in the `ensemble_base_learner_fold_scores.json` file of the *Experiment Summary*. You can download that file directly from a link at the bottom of this table.

Model Scores												
Final pipeline scores across cross-validation folds and models (available in <code>ensemble_base_learner_fold_scores.json</code>) (ConstantModel added for reference only, will be dropped in most cases):												
Model Name	Fold ID	Model ID	Accuracy	AUC	AUCPR	F05	F1	F2	Gini	LogLoss	MacroAUC	MCC
LightGBMModel	0	0	0.8278173	0.779897	0.5622561	0.6031469	0.5396825	0.6433409	0.5597941	0.4268015	0.779897	0.432115
LightGBMModel	0	1	0.830331	0.7903428	0.5655594	0.6102362	0.5451092	0.6442228	0.5605855	0.4262702	0.7903428	0.432184
ConstantModel	0	2	0.2211982	0.5	0	0.2620087	0.3622642	0.5867971	0	0.5284286	0.5	0

[download json](#) (also in summary.zip)

- **Pipeline Description:** This shows how the final Stacked Ensemble pipeline was calculated. This information is also included in the `ensemble_model_description.json` file of the *Experiment Summary*. You can download that file directly from a link at the bottom of this table.

Model Scores													
Pipeline Description: Final StackedEnsemble pipeline with ensemble_level=2 transforming 23 original features -> 25 features in each of 3 models each fit on external validation set then linearly blended:													
Type	Split Type	Model Weight	Booster	Model Class Name	Max Depth	Max Leaves	Subsample	Colsample_Bytree	Tree Method	Grow Policy	Random State	Learning Rate	Target
LightGBMModel	External	0.4	lightgbm	LightGBMModel	10	1024	0.7	0.8	gpu_hist	depthwise	360245737	0.03	Label
LightGBMModel	External	0.6	lightgbm	LightGBMModel	10	1024	0.7	0.8	gpu_hist	depthwise	360245737	0.03	Label
ConstantModel	External	0	constant										Label

[download json](#) (also in summary.zip)

- **Final Ensemble Scores:** This shows the final scores for each scorer in DAI. If a *custom scorer* was used in the experiment, that scorer will also appear here. This information is also included in the **ensemble_scores.json** file of the *Experiment Summary*. You can download that file directly from a link at the bottom of this table.

Final Ensemble Scores:				
SCORER	OPTIMIZED	BETTER SCORE IS	FINAL ENSEMBLE SCORES +/- STANDARD DEVIATION ON VALIDATION (INTERNAL OR EXTERNAL HOLDOUT(S)) DATA	FINAL TEST SCORES +/- STANDARD DEVIATION
MCC		higher	0.4292432 +/- 0.02331793	0.4144627 +/- 0.01925442
MACROAUC		higher	0.7814441 +/- 0.01177874	0.7656547 +/- 0.0090673
LOGLOSS		lower	0.4256566 +/- 0.01251221	0.4442352 +/- 0.009443205
GINI		higher	0.5626882 +/- 0.02355748	0.5313093 +/- 0.0181346
F2		higher	0.6457563 +/- 0.01242227	0.6379061 +/- 0.009465365
F1		higher	0.5400763 +/- 0.01682045	0.5431426 +/- 0.01436734
F05		higher	0.6058673 +/- 0.02485451	0.5807893 +/- 0.01697348
AUCPR		higher	0.5647043 +/- 0.02656562	0.5492047 +/- 0.01797327
ACCURACY		higher	0.8290742 +/- 0.008120806	0.8160729 +/- 0.005864942
AUC	*	higher	0.7814441 +/- 0.01177874	0.7656547 +/- 0.0090673

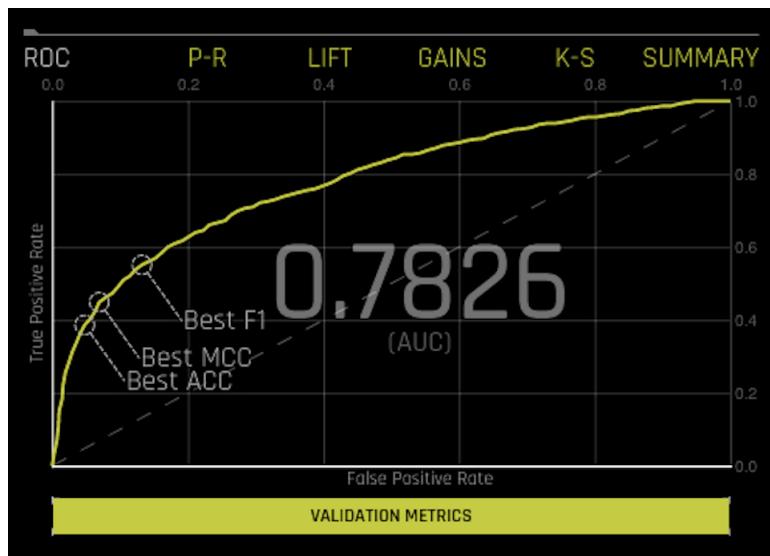
[download Json \(also in summary zip\)](#)

22.6 Experiment Graphs

This section describes the dashboard graphs that display for running and completed experiments. These graphs are interactive. Hover over a point on the graph for more details about the point.

22.6.1 Binary Classification Experiments

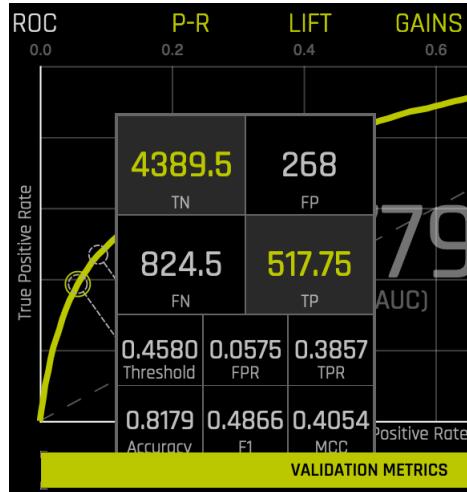
For Binary Classification experiments, Driverless AI shows a ROC Curve, a Precision-Recall graph, a Lift chart, a Kolmogorov-Smirnov chart, and a Gains chart.



- **ROC:** This shows Receiver-Operator Characteristics curve stats on validation data along with the best Accuracy, MCC, and F1 values. An ROC curve is a useful tool because it only focuses on how well the model was able to distinguish between classes. Keep in mind, though, that for models where one of the classes happens rarely, a high AUC could provide a false sense that the model is correctly predicting the results. This is where the notion of precision and recall become important.

The area under this curve is called AUC. The True Positive Rate (TPR) is the relative fraction of correct positive predictions, and the False Positive Rate (FPR) is the relative fraction of incorrect positive corrections. Each point corresponds to a classification threshold (e.g., YES if probability ≥ 0.3 else NO). For each threshold, there is a unique confusion matrix that represents the balance between TPR and FPR. Most useful operating points are in the top left corner in general.

Hover over a point in the ROC curve to see the True Negative, False Positive, False Negative, True Positive, Threshold, FPR, TPR, Accuracy, F1, and MCC values for that point in the form of a confusion matrix.



If a test set was provided for the experiment, then click on the **Validation Metrics** button below the graph to view these stats on test data.

- **Precision-Recall:** This shows the Precision-Recall curve on validation data along with the best Accuracy, MCC,

and F1 values. The area under this curve is called AUCPR. Prec-Recall is a complementary tool to ROC curves, especially when the dataset has a significant skew. The Prec-Recall curve plots the precision or positive predictive value (y-axis) versus sensitivity or true positive rate (x-axis) for every possible classification threshold. At a high level, you can think of precision as a measure of exactness or quality of the results and recall as a measure of completeness or quantity of the results obtained by the model. Prec-Recall measures the relevance of the results obtained by the model.

- Precision: correct positive predictions (TP) / all positives (TP + FP).
- Recall: correct positive predictions (TP) / positive predictions (TP + FN).

Each point corresponds to a classification threshold (e.g., YES if probability ≥ 0.3 else NO). For each threshold, there is a unique confusion matrix that represents the balance between Recall and Precision. This ROCPR curve can be more insightful than the ROC curve for highly imbalanced datasets.

Hover over a point in this graph to see the True Positive, True Negative, False Positive, False Negative, Threshold, Recall, Precision, Accuracy, F1, and MCC value for that point.

If a test set was provided for the experiment, then click on the **Validation Metrics** button below the graph to view these stats on test data.

- **Lift:** This chart shows lift stats on validation data. For example, “How many times more observations of the positive target class are in the top predicted 1%, 2%, 10%, etc. (cumulative) compared to selecting observations randomly?” By definition, the Lift at 100% is 1.0. Lift can help answer the question of how much better you can expect to do with the predictive model compared to a random model (or no model). Lift is a measure of the effectiveness of a predictive model calculated as the ratio between the results obtained with a model and with a random model(or no model). In other words, the ratio of gain % to the random expectation % at a given quantile. The random expectation of the x th quantile is $x\%$.

Hover over a point in the Lift chart to view the quantile percentage and cumulative lift value for that point.

If a test set was provided for the experiment, then click on the **Validation Metrics** button below the graph to view these stats on test data.

- **Kolmogorov-Smirnov:** This chart measures the degree of separation between positives and negatives for validation or test data.

Hover over a point in the chart to view the quantile percentage and Kolmogorov-Smirnov value for that point.

If a test set was provided for the experiment, then click on the **Validation Metrics** button below the graph to view these stats on test data.

- **Gains:** This shows Gains stats on validation data. For example, “What fraction of all observations of the positive target class are in the top predicted 1%, 2%, 10%, etc. (cumulative)?” By definition, the Gains at 100% are 1.0.

Hover over a point in the Gains chart to view the quantile percentage and cumulative gain value for that point.

If a test set was provided for the experiment, then click on the **Validation Metrics** button below the graph to view these stats on test data.

22.6.2 Multiclass Classification Experiments

For multiclass classification experiments, a Confusion Matrix is available in addition to the ROC Curve, Precision-Recall graph, Lift chart, Kolmogorov-Smirnov chart, and Gains chart. Driverless AI generates these graphs by considering the multiclass problem as multiple one-vs-all problems. These graphs and charts (Confusion Matrix excepted) are based on a method known as micro-averaging (reference: http://scikit-learn.org/stable/auto_examples/model_selection/plot_roc.html#multiclass-settings).

For example, you may want to predict the species in the iris data. The predictions would look something like this:

class.Iris-setosa	class.Iris-versicolor	class.Iris-virginica
0.9628	0.021	0.0158
0.0182	0.3172	0.6646
0.0191	0.9534	0.0276

To create these charts, Driverless AI converts the results to 3 one-vs-all problems:

prob-setosa	actual-setosa	prob-versicolor	actual-versicolor	prob-virginica	actual-virginica
0.9628	1	0.021	0	0.0158	0
0.0182	0	0.3172	1	0.6646	0
0.0191	0	0.9534	1	0.0276	0

The result is 3 vectors of predicted and actual values for binomial problems. Driverless AI concatenates these 3 vectors together to compute the charts.

```
predicted = [0.9628, 0.0182, 0.0191, 0.021, 0.3172, 0.9534, 0.0158, 0.6646, 0.0276]
actual = [1, 0, 0, 0, 1, 1, 0, 0, 0]
```

Multiclass Confusion Matrix

A confusion matrix shows experiment performance in terms of false positives, false negatives, true positives, and true negatives. For each threshold, the confusion matrix represents the balance between TPR and FPR (ROC) or Precision and Recall (Prec-Recall). In general, most useful operating points are in the top left corner.

In this graph, the actual results display in the columns and the predictions display in the rows; correct predictions are highlighted. In the example below, *Iris-setosa* was predicted correctly 30 times, while *Iris-virginica* was predicted correctly 32 times, and *Iris-versicolor* was predicted as *Iris-virginica* 2 times (against the validation set).

If a test set was provided for the experiment, then click on the **Validation Metrics** button below the graph to view these stats on test data.

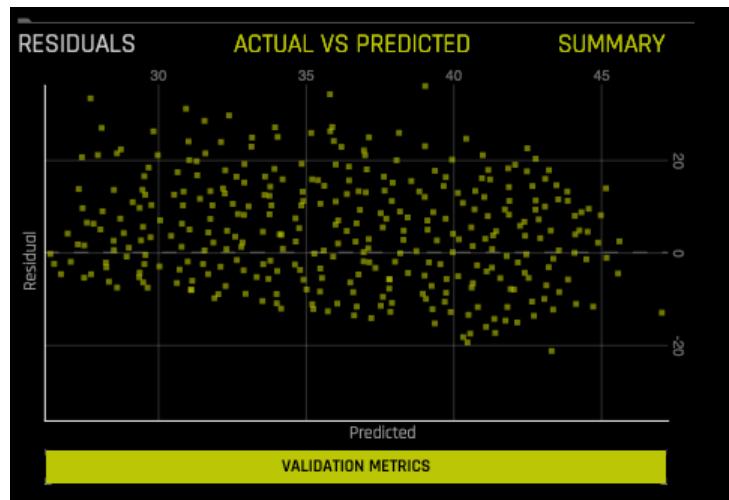
CM	ROC	P-R	LIFT	GAINS	K-S	SUMMARY
Actual label	Predicted label					
		Iris-setosa	Iris-versicolor	Iris-virginica	Total	Error
	Iris-setosa	30	0	0	30	0.0000
	Iris-versicolor	0	32	2	34	0.0588
	Iris-virginica	0	4	32	36	0.1111
	Total	30	36	34	100	
Error	0	0.1111	0.0588			0.0600

22.6.3 Regression Experiments

- **Residuals:** Residuals are the differences between observed responses and those predicted by a model. Any pattern in the residuals is evidence of an inadequate model or of irregularities in the data, such as outliers, and suggests how the model may be improved. This chart shows Residuals (Actual-Predicted) vs Predicted values on validation or test data. Note that this plot preserves all outliers. For a perfect model, residuals are zero.

Hover over a point on the graph to view the Predicted and Residual values for that point.

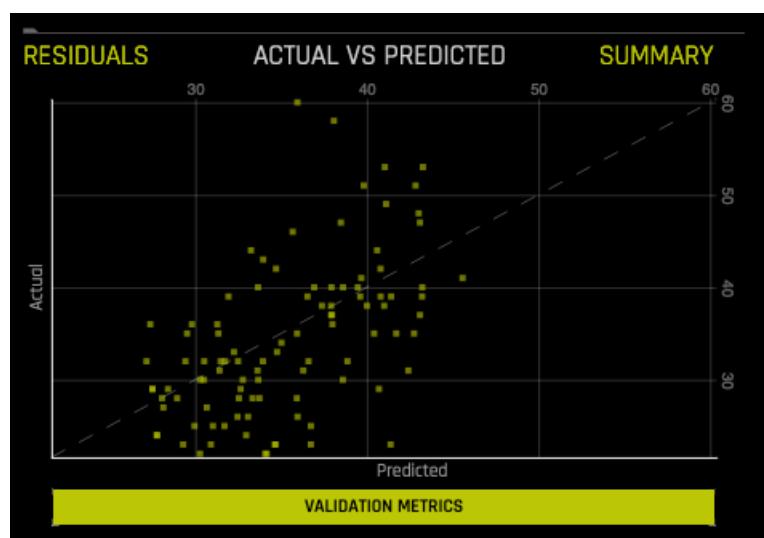
If a test set was provided for the experiment, then click on the **Validation Metrics** button below the graph to view these stats on test data.



- **Actual vs. Predicted:** This chart shows Actual vs Predicted values on validation data. A small sample of values are displayed. A perfect model has a diagonal line.

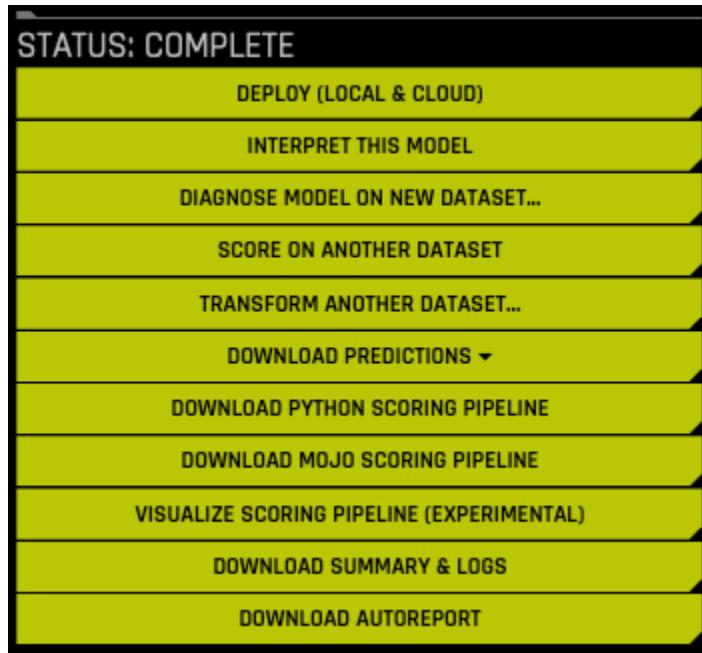
Hover over a point on the graph to view the Actual and Predicted values for that point.

If a test set was provided for the experiment, then click on the **Validation Metrics** button below the graph to view these stats on test data.



22.7 Experiment Summary

An experiment summary is available for each completed experiment. Click the **Download Summary & Logs** button to download the **h2oai_experiment_summary_<experiment>.zip** file.



The files within the experiment summary zip provide textual explanations of the graphical representations that are shown on the Driverless AI UI. Details of each artifact are described below.

22.7.1 Experiment Autoreport

A **report** file (AutoDoc) is included in the experiment summary. This report provides insight into the training data and any detected shifts in distribution, the validation schema selected, model parameter tuning, feature evolution and the final set of features chosen during the experiment.

- **report.docx:** the report available in Word format

[Click here](#) to download and view a sample experiment report in Word format.

BYOR Recipes with Autoreport

The experiment Autoreport supports experiments that use custom scorers, transformers, or models. Custom scorers and transformers are documented the same as Driverless AI scorers and transformers. If Driverless AI used a custom transformer, it is included in the Feature Transformations table under its display name; otherwise it is only included in the Feature Evolution section. (Note: custom-transformer descriptions are currently shown as “None” in this section.) For custom models, the standard performance metrics and plots are included; however, information that Driverless AI cannot access is not included, or is shown as “custom”, “unavailable”, or “auto.” For example, in the Model Tuning table, the booster is listed as “custom”, and in the Alternative Models section, the model package documentation is listed as “unavailable.”

22.7.2 Autoreport Support

Autoreport only supports resumed experiments for certain Driverless AI versions. See the following table to check the types of resumed experiments that are supported for your version:

Autoreport Support for Resumed Experiments Via	LTS	1.7.0 and older	1.7.1	1.9.0
New model with same parameters	yes	yes	yes	yes
Restart from last checkpoint	no	no	yes	yes
Retrain final pipeline	no	no	no	yes

Notes:

- Autoreport does not support experiments that were built off of previously aborted or failed experiments.
- Reports for unsupported resumed experiments will still build, but they will only include the following text: “AutoDoc not yet supported for resumed experiments.”

22.7.3 Experiment Artifacts Overview

The Experiment Summary contains artifacts that provide overviews of the experiment.

- **preview.txt:** Provides a preview of the experiment. (This is the same information that was included on the UI before starting the experiment.)
- **summary:** Provides the same summary that appears in the lower-right portion of the UI for the experiment. (Available in txt or json.)
- **config.json:** Provides a list of the settings used in the experiment.
- **config_overrides_toml_string.txt:** Provides any overrides for this experiment that were made to the config.toml file.
- **args_do_auto_dl.json:** The internal arguments used in the Driverless AI experiment based on the dataset and accuracy, time and interpretability settings.
- **experiment_column_types.json:** Provides the column types for each column included in the experiment.
- **experiment_original_column.json:** A list of all columns available in the dataset that was used in the experiment.
- **experiment_pipeline_original_required_columns.json:** For columns used in the experiment, this includes the column name and type.
- **experiment_sampling_description.json:** A description of the sampling performed on the dataset.
- **timing.json:** The timing and number of models generated in each part of the Driverless AI pipeline.

22.7.4 Tuning Artifacts

During the Driverless AI experiment, model tuning is performed to determine the optimal algorithm and parameter settings for the provided dataset. For regression problems, target tuning is also performed to determine the best way to represent the target column (i.e. does taking the log of the target column improve results). The results from these tuning steps are available in the Experiment Summary.

- **tuning_leaderboard:** A table of the model tuning performed along with the score generated from the model and training time. (Available in txt or json.)
- **target_transform_tuning_leaderboard.txt:** A table of the transforms applied to the target column along with the score generated from the model and training time. (This will be empty for binary and multiclass use cases.)

22.7.5 Features Artifacts

Driverless AI performs feature engineering on the dataset to determine the optimal representation of the data. The top features used in the final model can be seen in the GUI. The complete list of features used in the final model is available in the Experiment Summary artifacts.

The Experiment Summary also provides a list of the original features and their estimated feature importance. For example, given the features in the final Driverless AI model, we can estimate the feature importance of the original features.

Feature	Feature Importance
NumToCatWoE:PAY_AMT2	1
PAY_3	0.92
ClusterDist9:BILL_AMT1:LIMIT_BAL:PAY_3	0.90

To calculate the feature importance of PAY_3, we can aggregate the feature importance for all variables that used PAY_3:

- NumToCatWoE:PAY_AMT2: 1 * 0 (PAY_3 not used.)
- PAY_3: 0.92 * 1 (PAY_3 is the only variable used.)
- ClusterDist9:BILL_AMT1:LIMIT_BAL:PAY_3: 0.90 * 1/3 (PAY_3 is one of three variables used.)

$$\text{Estimated Feature Importance} = (1*0) + (0.92*1) + (0.9*(1/3)) = 1.22$$

Note: The feature importance is converted to relative feature importance. (The feature with the highest estimated feature importance will have a relative feature importance of 1).

- **ensemble_features:** A list of features used in the final model, a description of the feature, and the relative feature importance. Feature importances for multiple models are linearly blended with same weights as the final ensemble of models. (Available in txt, table, or json.)
- **ensemble_features_orig:** A complete list of all original features used in the final model, a description of the feature, the relative feature importance, and the standard deviation of relative importance. (Available in txt or json.)
- **ensemble_features_orig_shift:** A list of original user features used in the final model and the difference in relative feature importance between the final model and the corresponding feature importance of the final population. (Available in txt or json.)
- **ensemble_features_prefit:** A list of features used by the best individuals in the final population, each model blended with same weights as ensemble if ensemble used blending. (Available in txt, table, or json.)
- **ensemble_features_shift:** A list of features used in the final model and the difference in relative feature importance between the final model and the corresponding feature importance of the final population. (Available in txt, table, or json.)
- **features:** A list of features used by the best individual pipeline (identified by the genetic algorithm) and each feature's relative importance. (Available in txt, table, or json.)
- **features_orig:** A list of original user features used by the best individual pipeline (identified by the genetic algorithm) and each feature's estimated relative importance. (Available in txt or json.)
- **leaked_features.json:** A list of all leaked features provided along with the relative importance and the standard deviation of relative importance. (Available in txt, table, or json.)
- **leakage_features_orig.json:** A list of leaked original features provided and an estimate of the relative feature importance of that leaked original feature in the final model.

- **shift_features.json**: A list of all features provided along with the relative importance and the shift in standard deviation of relative importance of that feature.
- **shift_features_orig.json**: A list of original features provided and an estimate of the shift in relative feature importance of that original feature in the final model.

22.7.6 Final Model Artifacts

The Experiment Summary includes artifacts that describe the final model. This is the model that is used to score new datasets and create the MOJO scoring pipeline. The final model may be an ensemble of models depending on the Accuracy setting.

- **coefs**: A list of coefficients and standard deviation of coefficients for features. (Available in txt or json.)
- **ensemble.txt**: A summary of the final model which includes a description of the model(s), gains/lifts table, confusion matrix, and scores of the final model for our list of scorers.
- **ensemble_base_learner_fold_scores**: The internal validation scorer metrics for each base learner when the final model is an ensemble. (Available in table or json.) Note that this is not available for Time Series experiments.
- **ensemble_description.txt**: A sentence describing the final model. (For example: “Final TensorFlowModel pipeline with ensemble_level=0 transforming 21 original features -> 54 features in each of 1 models each fit on full training data (i.e. no hold-out).”)
- **ensemble_coefs**: The coefficient and standard deviation coefficient for each feature in the ensemble. (Available as txt or json.)
- **ensemble_coefs_shift**: The coefficient and shift of coefficient for each feature in the ensemble. (Available as txt or json.)
- **ensemble_model_description.json/ensemble_model_extra_description**: A json file describing the model(s) and for ensembles how the model predictions are weighted.
- **ensemble_model_params.json**: A json file describing the parameters of the model(s).
- **ensemble_folds_data.json**: A json file describing the folds used for the final model(s). This includes the size of each fold of data and the performance of the final model on each fold. (Available if a fold column was specified.)
- **ensemble_features_orig**: A list of the original features provided and an estimate of the relative feature importance of that original feature in the ensemble of models. (Available in txt or json.)
- **ensemble_features**: A complete list of all features used in the final ensemble of models, a description of the feature, and the relative feature importance. (Available in txt, table, or json.)
- **leakage_coefs.json**: A list of coefficients and standard deviation of coefficients for leaked features.
- **pipeline**: A visual representation of the experiment pipeline.
- **shift_coefs.json**: A list of coefficients and the shift in standard deviation for those coefficients used in the experiment.

The Experiment Summary also includes artifacts about the final model performance.

- **ensemble_scores.json**: The scores of the final model for our list of scorers.
- **ensemble_confusion_matrix_test**: The confusion matrix for the test data if test data is provided. Note that this is not available for Time Series experiments.
- **ensemble_confusion_matrix_with_validation**: The confusion matrix for the internal validation data. Note that this is not available for Time Series experiments.

- **ensemble_confusion_matrix_stats_validation:** The confusion matrix statistics on internal validation data. Note that this is not available for Time Series experiments.
- **ensemble_confusion_matrix_stats_test.json:** Confusion matrix statistics on the test data. This is only available if test data is provided. Note that this is not available for Time Series experiments.
- **ensemble_gains_test:** The lift and gains table for test data if test data is provided. (Visualization of lift and gains can be seen in the UI.) Note that this is not available for Time Series experiments.
- **ensemble_gains_with_validation:** The lift and gains table for the internal validation data. (Visualization of lift and gains can be seen in the UI.) Note that this is not available for Time Series experiments.
- **ensemble_roc_test:** The ROC and Precision Recall table for test data if test data is provided. (Visualization of ROC and Precision Recall curve can be seen in the UI.) Note that this is not available for Time Series experiments.
- **ensemble_roc_with_validation:** The ROC and Precision Recall table for the internal validation data. (Visualization of ROC and Precision Recall curve can be seen in the UI.) Note that this is not available for Time Series experiments.
- **fs_normalized_varimp:** The normalized frequency variable importance values. (Available in table or json.) Note that this is not available for Time Series experiments.
- **fs_unnormalized_varimp:** The unnormalized frequency variable importance values. (Available in table or json.) Note that this is not available for Time Series experiments.
- **individual_scored.params_base:** Detailed information about each iteration run in the experiment. (Available in csv, table, or json.)

22.8 Viewing Experiments

The upper-right corner of the Driverless AI UI includes an **Experiments** link.



Click this link to open the Experiments page. From this page, you can rename an experiment, view previous experiments, begin a new experiment, rerun an experiment, and delete an experiment.

A screenshot of the Driverless AI Experiments page. The page has a dark header with the H2O.ai logo and navigation links. Below the header is a sub-header "Experiments" with a dropdown arrow. To the right is a yellow button labeled "+ NEW EXPERIMENT". The main content is a table listing two experiments: "wodasaga" and "honedume".

Name	Target	Dataset	Acc	Time	Int	Size	Scorer	Vl. score	Test score	Status	ETA/Runtime
wodasaga	default payment...	CreditCard_train...	5	3	5	1GB	AUCPR	0.5486	NA	Done	00:05:22
honedume	default payment...	CreditCard_train...	7	3	7	1GB	AUC	0.7832	0.76657	Done	00:07:36

22.8.1 Checkpointing, Rerunning, and Retraining

In Driverless AI, you can retry an experiment from the last checkpoint, you can run a new experiment using an existing experiment's settings, and you can retrain an experiment's final pipeline.

	Name	Target	Dataset	Acc	Time	Int	Size	Scorer	Val. score	Test score	Status	ETA/Runtime	
<input type="checkbox"/>	wodasaga	default payment...	CreditCard_train...	5	3	5	1GB	AUCPR	0.5486	NA	Done	00:05:22	
<input type="checkbox"/>	hone dumme	default payment...	CreditCard_train...	7	3	7	1GB	AUC	0.7832	0.76657	OPEN	Done	00:07:36

RENAME
 NEW MODEL WITH SAME PARAMS
 RESTART FROM LAST CHECKPOINT
 RETRAIN FINAL PIPELINE
 DELETE

Checkpointing Experiments

In real-world scenarios, data can change. For example, you may have a model currently in production that was built using 1 million records. At a later date, you may receive several hundred thousand more records. Rather than building a new model from scratch, Driverless AI includes H2O.ai Brain, which enables caching and smart re-use of prior models to generate features for new models.

You can configure one of the following Brain levels in the experiment's *Expert Settings*.

- -1: Don't use any brain cache
- 0: Don't use any brain cache but still write to cache
- 1: Smart checkpoint if an old experiment_id is passed in (for example, via running "resume one like this" in the GUI)
- 2: Smart checkpoint if the experiment matches all column names, column types, classes, class labels, and time series options identically. (default)
- 3: Smart checkpoint like level #1, but for the entire population. Tune only if the brain population is of insufficient size.
- 4: Smart checkpoint like level #2, but for the entire population. Tune only if the brain population is of insufficient size.
- 5: Smart checkpoint like level #4, but will scan over the entire brain cache of populations (starting from resumed experiment if chosen) in order to get the best scored individuals.

If you chooses Level 2 (default), then Level 1 is also done when appropriate.

To make use of smart checkpointing, be sure that the new data has:

- The same data column names as the old experiment

- The same data types for each column as the old experiment. (This won't match if, e.g., a column was all int and then had one string row.)
- The same target as the old experiment
- The same target classes (if classification) as the old experiment
- For time series, all choices for intervals and gaps must be the same

When the above conditions are met, then you can:

- Start the same kind of experiment, just rerun for longer.
- Use a smaller or larger data set (i.e. fewer or more rows).
- Effectively do a final ensemble re-fit by varying the data rows and starting an experiment with a new accuracy, time=1, and interpretability. Check the experiment preview for what the ensemble will be.
- Restart/Resume a cancelled, aborted, or completed experiment

To run smart checkpointing on an existing experiment, click the right side of the experiment that you want to retry, then select **Restart from Last Checkpoint**. The experiment settings page opens. Specify the new dataset. If desired, you can also change experiment settings, though the target column must be the same. Click **Launch Experiment** to resume the experiment from the last checkpoint and build a new experiment.

The smart checkpointing continues by adding a prior model as another model used during tuning. If that prior model is better (which is likely if it was run for more iterations), then that smart checkpoint model will be used during feature evolution iterations and final ensemble.

Notes:

- Driverless AI does not guarantee exact continuation, only smart continuation from any last point.
- The directory where the H2O.ai Brain meta model files are stored is **tmp/H2O.ai_brain**. In addition, the default maximum brain size is 20GB. Both the directory and the maximum size can be changed in the config.toml file.

Rerunning Experiments

To run a new experiment using an existing experiment's settings, click the right side of the experiment that you want to use as the basis for the new experiment, then select **New Model with Same Params**. This opens the experiment settings page. From this page, you can rerun the experiment using the original settings, or you can specify to use new data and/or specify different experiment settings. Click **Launch Experiment** to create a new experiment with the same options.

Retrain Final Pipeline

To retrain an experiment's final pipeline, click the right side of the experiment that you want to use as the basis for the new experiment, then select **Retrain Final Pipeline**. This opens the experiment settings page with the same settings as the original experiment except that Time is set to 0.

Note: When retraining an experiment's final pipeline, Driverless AI also refits the experiment's final model(s). This may include the addition of new features, the exclusion of previously used features, a change in the hyperparameter search space, or finding new parameters for the existing model architecture.

To retrain the final pipeline without adding new features, specify the following config.toml overrides with the *Add to config.toml via toml String* expert setting:

```
refit_same_best_individual=True  
brain_add_features_for_new_columns=False
```

For more information, refer to the `feature_brain_level` setting in the config.toml file.

22.8.2 “Pausing” an Experiment

A trick for “pausing” an experiment is to:

1. Abort the experiment.
2. On the Experiments page, select **Restart from Last Checkpoint** for the aborted experiment.
3. On the Expert Settings page, specify 0 for the *Ensemble Level for Final Modeling Pipeline* option.

22.8.3 Deleting Experiments

To delete an experiment, click the right side of the experiment that you want to remove, then select **Delete**. A confirmation message will display asking you to confirm the delete. Click **OK** to delete the experiment or **Cancel** to return to the experiments page without deleting.

The screenshot shows the H2O.ai interface with the following details:

- Header:** PROJECTS DATASETS AUTOVIZ EXPERIMENTS DIAGNOSTICS MLI DEPLOYMENTS RESOURCES ▾ MESSAGES [2] LOGOUT
- Sub-Header:** DRIVERLESS AI 1.7.0 – AI TO DO AI
Licensed to H2O.ai (SN29). Current User – ANG
- Section:** Experiments
- Table:** A list of experiments with columns: Name, Target, Dataset, Acc, Time, Int, Size, Scorer, Val. score, Test score, Status, ETA/Runtime. The rows are:
 - wodosaga: default payment..., CreditCard_train..., 5, 3, 5, 1GB, AUCPR, 0.5496, NA, Done, 00:05:22
 - honedume: default payment..., CreditCard_train..., 7, 3, 7, 1GB, AUC, 0.7832, 0.76657, OPEN, Done, 00:07:36
- Context Menu (for 'honedume'):**
 - RENAME
 - NEW MODEL WITH SAME PARAMS
 - RESTART FROM LAST CHECKPOINT
 - RETRAIN FINAL PIPELINE
 - DELETE

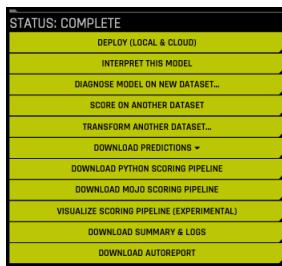
CHAPTER
TWENTYTHREE

DIAGNOSING A MODEL

The **Diagnosing Model on New Dataset** option allows you to view model performance for multiple scorers based on existing model and dataset.

On the completed experiment page, click the **Diagnose Model on New Dataset** button.

Note: You can also diagnose a model by selecting **Diagnostic** from the top menu, then selecting an experiment and test dataset.



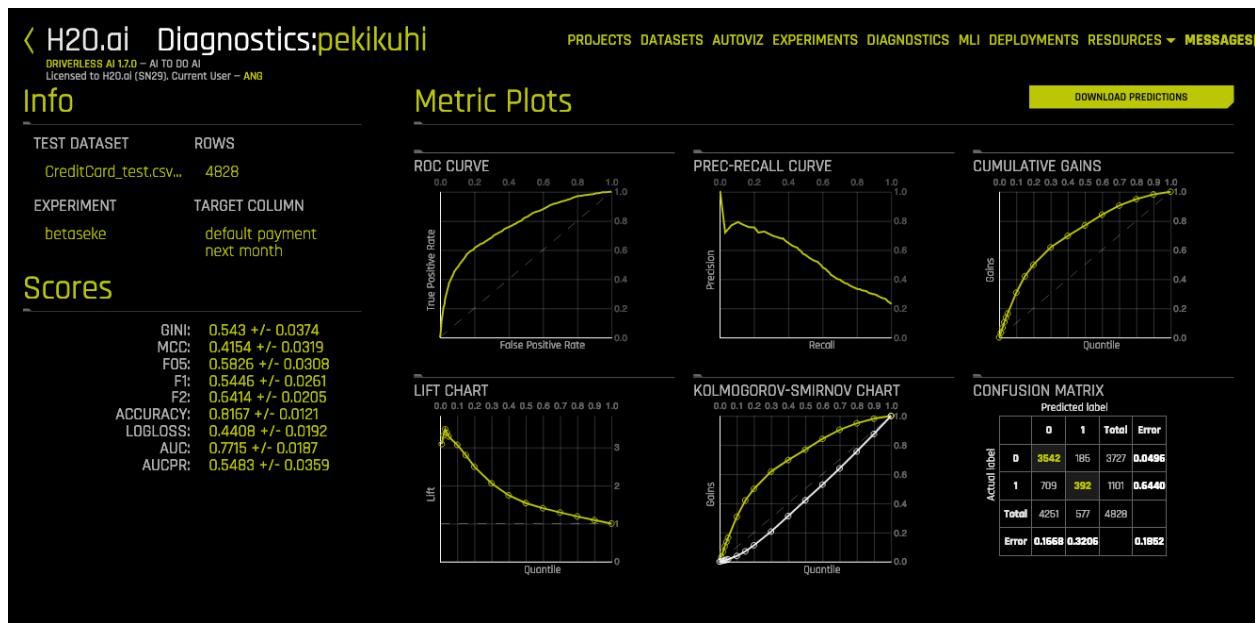
Select a dataset to use when diagnosing this experiment. Note that the dataset must include the target column that is in the original dataset. At this point, Driverless AI will begin calculating all available scores for the experiment.

When the diagnosis is complete, it will be available on the **Model Diagnostics** page. Click on the new diagnosis. From this page, you can download predictions. You can also view scores and metric plots. The plots are interactive. Click a graph to enlarge. In the enlarged view, you can hover over the graph to view details for a specific point. You can also download the graph in the enlarged view.

23.1 Classification Metric Plots

Classification metric plots include the following graphs:

- ROC Curve
- Precision-Recall Curve
- Cumulative Gains
- Lift Chart
- Kolmogorov-Smirnov Chart
- Confusion Matrix



Note: In the Confusion Matrix graph, the threshold value defaults to 0.5. For binary classification experiments, users can specify a different threshold value. The threshold selector is available after clicking on the Confusion Matrix and opening the enlarged view. When you specify a value or change the slider value, Driverless AI automatically computes a diagnostic Confusion Matrix for that given threshold value.



23.2 Regression Metric Plots

Regression metric plots include the following graphs:

- Actual vs Predicted
- Residual Plot with LOESS curve
- Residual Histogram



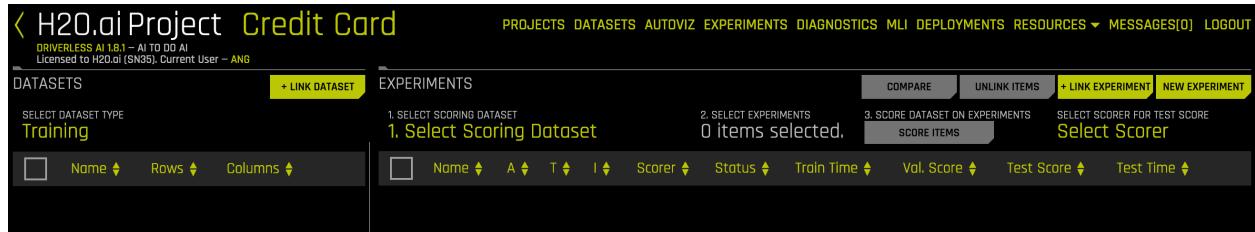
CHAPTER TWENTYFOUR

PROJECT WORKSPACE

Driverless AI provides a Project Workspace for managing datasets and experiments related to a specific business problem or use case. Whether you are trying to detect fraud or predict user retention, datasets and experiments can be stored and saved in the individual projects. A Leaderboard on the Projects page allows you to easily compare performance and results and identify the best solution for your problem.

To create a Project Workspace:

1. Click the **Projects** option on the top menu.
2. Click **New Project**.
3. Specify a name for the project and provide a description.
4. Click **Create Project**. This creates an empty Project page.



The screenshot shows the H2O.ai Project Credit Card interface. At the top, there's a navigation bar with links for PROJECTS, DATASETS, AUTOVIZ, EXPERIMENTS, DIAGNOSTICS, MLI, DEPLOYMENTS, RESOURCES, and MESSAGES. Below the navigation bar, there are two main sections: 'DATASETS' on the left and 'EXPERIMENTS' on the right. Under 'DATASETS', there's a dropdown menu set to 'Training'. Under 'EXPERIMENTS', there are three numbered steps: 1. SELECT SCORING DATASET (with a sub-step '1. Select Scoring Dataset'), 2. SELECT EXPERIMENTS (showing '0 items selected'), and 3. SCORE DATASET ON EXPERIMENTS (with a sub-step '3. SCORE DATASET ON EXPERIMENTS SCORE ITEMS'). There are also buttons for 'COMPARE', 'UNLINK ITEMS', '+ LINK EXPERIMENT', and 'NEW EXPERIMENT'. At the bottom, there are dropdown menus for 'Name', 'A', 'T', 'I', 'Scorer', 'Status', 'Train Time', 'Val. Score', 'Test Score', and 'Test Time'.

From the Projects page, you can link datasets and/or experiments, and you can run new experiments. When you link an existing experiment to a Project, the datasets used for the experiment will automatically be linked to this project (if not already linked).

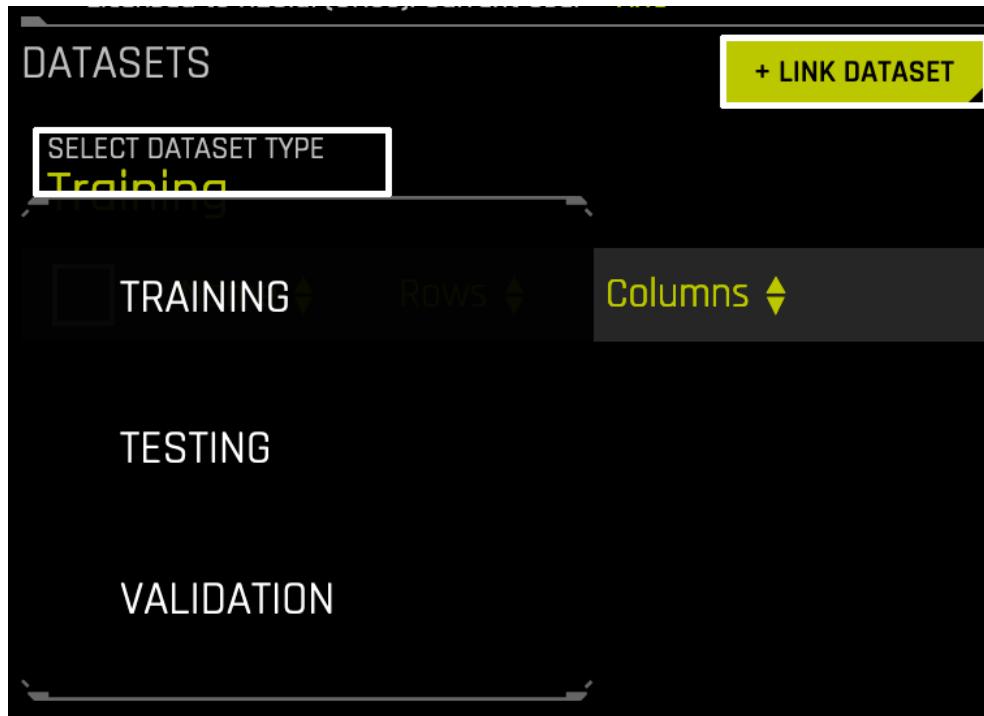
24.1 Linking Datasets

Any dataset that has been added to Driverless AI can be linked to a project. In addition, when you link an experiment, the datasets used for that experiment are also automatically linked to the project.

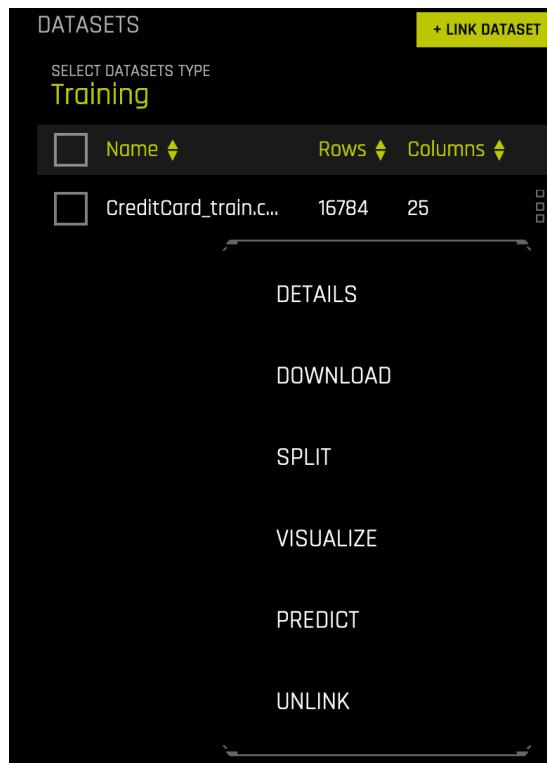
To link a dataset:

1. Select **Training**, **Validation**, or **Test** from the dropdown menu.
2. Click the **Link Dataset** button.
3. Select the dataset(s) that you want to link.

The list available datasets link include those that were added on [The Datasets Page](#), or you can browse datasets in your file system. Be sure to select the correct dropdown option before linking a training, validation, or test dataset. This is because, when you run a new experiment in the project, the training data, validation data, and test data options for that experiment come from list of datasets linked here. You will not be able to, for example, select any datasets from within the Training tab when specifying a test dataset on the experiment.



When datasets are linked, the same menu options are available here as on the Datasets page. Refer to [The Datasets Page](#) section for more information.



24.2 Linking Experiments

Existing experiments can be selected and linked to a Project. Additionally, you can run a new experiment or check-pointing an existing experiment from this page, and those experiments will automatically be linked to this Project.

Link an existing experiment to the project by clicking **Link Experiment** and then selecting the experiment(s) to include. When you link experiments, the datasets used to create the experiments are also automatically linked.

The screenshot shows the 'EXPERIMENTS' section of the H2O.ai Project interface. At the top, there's a yellow button labeled '+ LINK DATASET'. Below it, a dropdown menu says 'SELECT DATASET TYPE' with 'Training' selected. A table lists datasets: 'CreditCard-train.csv' (24K rows, 25 columns) and 'iris_train.csv' (100 rows, 5 columns). To the right of the table are sorting options for 'Name', 'Rows', and 'Columns'. On the right side of the interface, there are three main sections: '1. SELECT SCORING DATASET' (labeled '1. Select Scoring Dataset'), '2. SELECT EXPERIMENTS' (labeled '0 items selected.'), and '3. SCORE DATASET ON EXPERIMENTS' (labeled 'SCORE ITEMS'). A 'Select Scorer' dropdown is open, showing options like 'AUC', 'LOGLOSS', and 'N/A'. There are also buttons for 'COMPARE', 'UNLINK ITEMS', '+ LINK EXPERIMENT', and 'NEW EXPERIMENT'.

24.2.1 Selecting Datasets

In the Datasets section, you can select a training, validation, or testing dataset. The Experiments section will show experiments in the Project that use the selected dataset.

The screenshot shows the H2O.ai Project interface for a project named "Credit Card". The top navigation bar includes links for PROJECTS, DATASETS, AUTOVIZ, EXPERIMENTS, DIAGNOSTICS, MLI, DEPLOYMENTS, RESOURCES, and LOGOUT. The DATASETS section on the left lists two datasets: "CreditCard-train.csv" (selected) and "iris_train.csv". The EXPERIMENTS section in the center shows three completed experiments: "wikikubi", "honedume", and "CreditCard-train.csv". A sidebar on the right provides options for scoring datasets and selecting scorers.

24.2.2 New Experiments

When experiments are run from within a Project, only linked datasets or datasets available on the file system can be used.

1. Click the **New Experiment** link to begin a new experiment.
2. Select your training data and optionally your validation and/or testing data.
3. Specify your desired experiment settings (refer to *Experiment Settings* and *Expert Settings*), and then click **Launch Experiment**.

As the experiment is running, it will be listed at the top of the Experiments Leaderboard until it is completed. It will also be available on the **Experiments** page.

24.2.3 Checkpointing Experiments

When experiments are linked to a Project, the same checkpointing options for experiments are available here as on the Experiments page. Refer to *Checkpointing, Rerunning, and Retraining* for more information.

Name	A	T	I	Scorer	Status	Train Time	Val. Score	Test Score	Test Time
wikikubi	5	4	6	AUC	Completed	00:07:26	0.7790	NA	N/A
honedume	3	5	3	AUC	Completed	00:03:17	0.7797	NA	N/A
mabuviwi	3	3	3	LOGLOSS	Completed	00:05:45	0.1251	NA	N/A

NEW MODEL WITH SAME PARAMETERS
RESTART FROM LAST CHECKPOINT
RETRAIN FINAL PIPELINE
UNLINK

24.3 Experiments List

When attempting to solve a business problem, a normal workflow will include running multiple experiments, either with different/new data or with a variety of settings, and the optimal solution can vary for different users and/or business problems. For some users, the model with the highest accuracy for validation and test data could be the most optimal one. Other users might be willing to make an acceptable compromise on the accuracy of the model for a model with greater performance (faster prediction). For some, it could also mean how quickly the model could be trained with acceptable levels of accuracy. The Experiments list makes it easy for you to find the best solution for your business problem.

The list is organized based on experiment name. You can change the sorting of experiments by selecting the up/down arrows beside a column heading in the experiment menu.

Hover over the right menu of an experiment to view additional information about the experiment, including the problem type, datasets used, and the target column.

24.3.1 Experiment Scoring

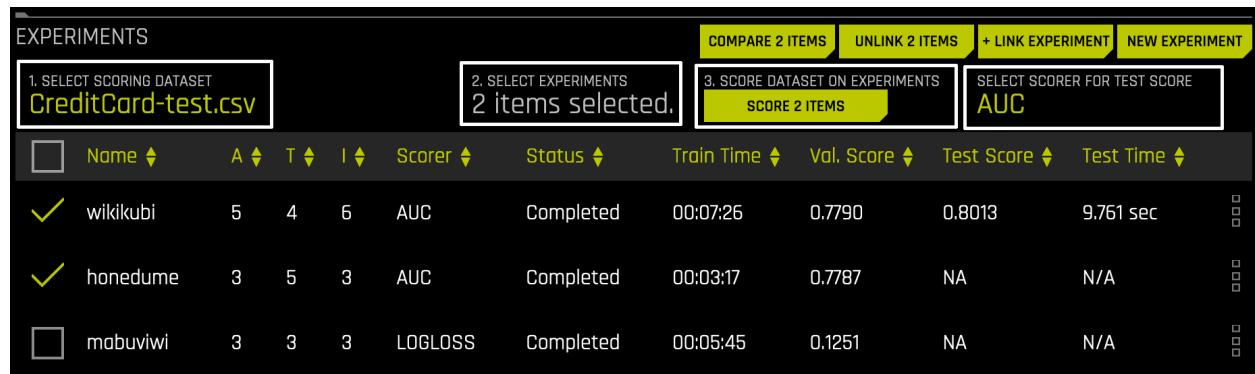
Experiments linked to projects do not automatically include a test score. To view Test Scores in the Leaderboard, you must first complete the scoring step for a particular dataset and experiment combination. Without the scoring step, no scoring data is available to populate in the Test Score and Score Time columns. Experiments that do not include a test score or that have an invalid scorer (for example, if the R2 scorer is selected for classification experiments) show N/A in the Leaderboard. Also, if **None** is selected for the scorer, then all experiments will show N/A.

To score the experiment:

1. Click **Select Scoring Dataset** at the top of the Experiments list and select a linked Test Dataset or a test dataset available on the file system.
2. Select the model or models that you want to score.

3. Click the **Select Scorer** button at the top of the Experiments list and select a scorer.
4. Click the **Score n Items** button.

This starts the Model Diagnostic process and scores the selected experiment(s) against the selected scorer and dataset. (Refer to *Diagnosing a Model* for more information.) Upon completion, the experiment(s) will be populated with a test score, and the performance information will also be available on the Model Diagnostics page.



The screenshot shows the Driverless AI interface for managing experiments. At the top, there are four main steps: 1. SELECT SCORING DATASET (CreditCard-test.csv), 2. SELECT EXPERIMENTS (2 items selected), 3. SCORE DATASET ON EXPERIMENTS (SCORE 2 ITEMS), and 4. SELECT SCORER FOR TEST SCORE (AUC). Below these steps is a table titled 'EXPERIMENTS' containing three rows of data:

	Name	A	T	I	Scorer	Status	Train Time	Val. Score	Test Score	Test Time
<input checked="" type="checkbox"/>	wikikubi	5	4	6	AUC	Completed	00:07:26	0.7790	0.8013	9.761 sec
<input checked="" type="checkbox"/>	honedume	3	5	3	AUC	Completed	00:03:17	0.7787	NA	N/A
<input type="checkbox"/>	mabuvuwi	3	3	3	LOGLOSS	Completed	00:05:45	0.1251	NA	N/A

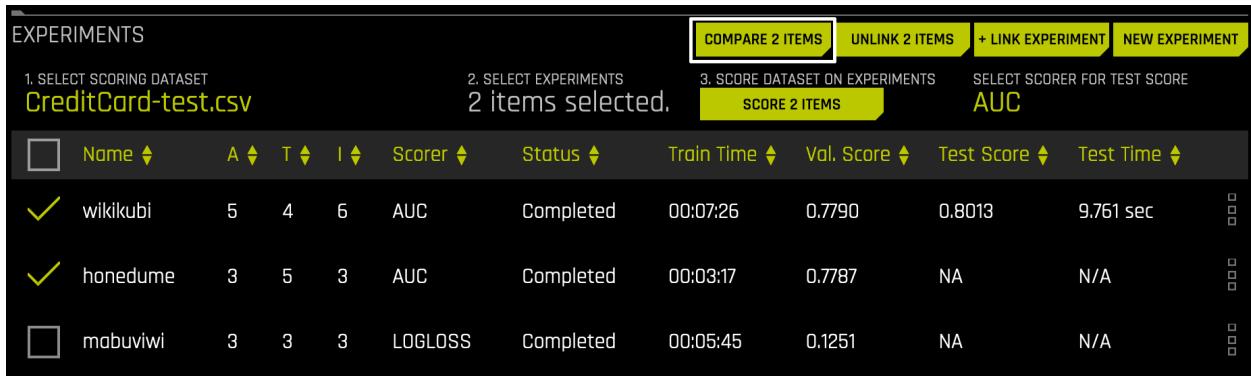
Notes:

- If an experiment has already scored a dataset, Driverless AI will not score it again. The scoring step is deterministic, so for a particular test dataset and experiment combination, the score will be same regardless of how many times you repeat it.
- The test dataset absolutely needs to have all the columns that are expected by the various experiments you are scoring it on. However, the columns of the test dataset need not be exactly the same as input features expected by the experiment. There can be additional columns in the test dataset. If these columns were not used for training, they will be ignored. This feature gives you the ability to train experiments on different training datasets (i.e., having different features), and if you have an “uber test dataset” that includes all these feature columns, then you can use the same dataset to score these experiments.
- You will notice a Score Time in the Experiments Leaderboard. This values shows the total time (in seconds) that it took for calculating the experiment scores for all applicable scorers for the experiment type. This is valuable to users who need to estimate the runtime performance of an experiment.

24.3.2 Comparing Experiments

You can compare two or three experiments and view side-by-side detailed information about each.

1. Click the **Select** button at the top of the Leaderboard and select either two or three experiments that you want to compare. You cannot compare more than three experiments.
2. Click the **Compare n Items** button.



The screenshot shows the Driverless AI Experiment Leaderboard interface. At the top, there are four tabs: 'EXPERIMENTS', 'COMPARE 2 ITEMS' (which is highlighted in yellow), 'UNLINK 2 ITEMS', '+ LINK EXPERIMENT', and 'NEW EXPERIMENT'. Below the tabs, there are three sections labeled 1. SELECT SCORING DATASET, 2. SELECT EXPERIMENTS, and 3. SCORE DATASET ON EXPERIMENTS. Section 1 shows 'CreditCard-test.csv'. Section 2 shows '2 items selected.' Section 3 shows 'SELECT SCORER FOR TEST SCORE' with 'AUC' selected. The main table lists three experiments: 'wikikubi' (selected), 'honedume' (selected), and 'mabuvivi'. The columns include Name, A, T, I, Scorer, Status, Train Time, Val. Score, Test Score, and Test Time. The 'COMPARE 2 ITEMS' button is also visible in the table header.

	Name	A	T	I	Scorer	Status	Train Time	Val. Score	Test Score	Test Time	
<input checked="" type="checkbox"/>	wikikubi	5	4	6	AUC	Completed	00:07:26	0.7790	0.8013	9.761 sec	<input type="checkbox"/>
<input checked="" type="checkbox"/>	honedume	3	5	3	AUC	Completed	00:03:17	0.7787	NA	N/A	<input type="checkbox"/>
<input type="checkbox"/>	mabuvivi	3	3	3	LOGLOSS	Completed	00:05:45	0.1251	NA	N/A	<input type="checkbox"/>

This opens the **Compare Experiments** page. This page includes the experiment summary and metric plots for each experiment. The metric plots vary depending on whether this is a classification or regression experiment.

For classification experiments, this page includes:

- Variable Importance list
- Confusion Matrix
- ROC Curve
- Precision Recall Curve
- Lift Chart
- Gains Chart
- Kolmogorov-Smirnov Chart

For regression experiments, this page includes:

- Variable Importance list
- Actual vs. Predicted Graph

Compare Experiments

EXPERIMENT SUMMARY	
Experiment: <code>celmossa (2e355d0-55b1-1e9-9c07-0242ac110002)</code> Version: 1.7.0, 2019-04-24 16:57 Settings: 5/3/5, seed=13063021, GPUs enabled Train data: <code>CreditCard_train.csv</code> (16784, 24) Validation data: N/A Test data: N/A Target column: default payment next month (binary, 22.283% target class) System specs: Docker/Linux, 240 GB, 32 CPU cores, 4/4 GPUs Max memory usage: 4.16 GB, 1.8 GB GPU Recipe: AutoDL (8 Iterations, 4 individuals) Validation scheme: stratified, 2 Internal holdouts Feature engineering: 1691 features scored (23 selected) Timing: Data preparation: 3.49 secs Model and feature tuning: 90.43 secs (35 models trained) Feature evolution: 19.08 secs (12 of 128 models trained) Final pipeline building: 13.12 secs (3 models trained) Python / MOJO scoring: 16.01 secs / 0.00 secs Validation score: AUC = 0.7846 +/- 0.007905 (baseline) Validation score: AUC = 0.7806 +/- 0.008069 (final pipeline) Test score: AUC = N/A	

VARIABLE IMPORTANCE	
10_PAY_0	1.00
12_PAY_3	0.29
11_PAY_2	0.22
16_PAY_AMT1	0.15
13_PAY_4	0.15
1_BILL_AMT1	0.14
8_LIMIT_BAL	0.12
17_PAY_AMT2	0.12
18_PAY_AMT3	0.11
2_BILL_AMT2	0.10
19_PAY_AMT4	0.10

EXPERIMENT SUMMARY

EXPERIMENT SUMMARY	
Experiment: <code>lnteris (f99bf62e-62bb-11e9-9c07-0242ac110002)</code> Version: 1.7.0, 2019-04-19 16:00 Settings: 5/3/5, seed=212919763, GPUs enabled Train data: <code>CreditCard_train.csv</code> (16784, 24) Validation data: <code>CreditCard_vald.csv</code> (2387, 24) Test data: <code>CreditCard_test.csv</code> (4828, 23) Target column: default payment next month (binary, 22.253% target class) System specs: Docker/Linux, 240 GB, 32 CPU cores, 4/4 GPUs Max memory usage: 3.63 GB, 1.68 GB GPU Recipe: AutoDL (8 Iterations, 4 individuals) Validation scheme: user-given validation data Feature engineering: 1094 features scored (30 selected) Timing: Data preparation: 3.69 secs Model and feature tuning: 77.42 secs (18 models trained) Feature evolution: 19.88 secs (6 of 64 models trained) Final pipeline building: 16.33 secs (1 model trained) Python / MOJO scoring: 15.89 secs / 0.00 secs Validation score: AUC = 0.7762 +/- 0.0085769 (baseline) Validation score: AUC = 0.7809 +/- 0.011016 (final pipeline) Test score: AUC = 0.7603 +/- 0.0082957 (final pipeline)	

VARIABLE IMPORTANCE	
44_NumToCatWoE_PAY_0PAY_3.0	1.00
57_WoE-PAY_0PAY_2PAY_4.0	0.90
47_NumToCatWoE-MARRIAGE-PAY_0PAY_5.0	0.66
39_ClusterTE:ClusterID69AGE-PAY_0PAY_2SEX0	0.50
58_NumToCatWoE-PAY_0PAY_3.0	0.41
33_ClusterTE:ClusterID24PAY_3PAY_5.0	0.21
55_NumToCatTE-PAY_0	0.20
40_ClusterTE:ClusterID64PAY_0PAY_AMT1PAY_AMT4.0	0.13
24_ClusterTE:ClusterID92PAY_5PAY_AMT2.0	0.12
1_BILL_AMT1	0.11
28_NumToCatTE-MARRIAGE-PAY_AMT3.0	0.11

24.4 Unlinking Data on a Projects Page

Unlinking datasets and/or experiments does not delete that data from Driverless AI. The datasets and experiments will still be available on the **Datasets** and **Experiments** pages.

- Unlink a dataset by clicking on the dataset and selecting **Unlink** from the menu. **Note:** You cannot unlink datasets that are tied to experiments in the same project.
- Unlink an experiment by clicking on the experiment and selecting **Unlink** from the menu. Note that this will not automatically unlink datasets that were tied to the experiment.

24.5 Deleting Projects

To delete a project, click the **Projects** option on the top menu to open the main Projects page. Click the dotted menu the right-most column, and then select Delete. You will be prompted to confirm the deletion.

Note that deleting projects does not delete datasets and experiments from Driverless AI. Any datasets and experiments from deleted projects will still be available on the **Datasets** and **Experiments** pages.

The screenshot shows the H2O.ai Driverless AI interface. At the top, there is a navigation bar with links for PROJECTS, DATASETS, AUTOVIZ, EXPERIMENTS, DIAGNOSTICS, MLI, DEPLOYMENTS, RESOURCES, and LOGOUT. Below the navigation bar, it says "LICENSED TO H2O.ai (SN35). Current User - ANG". There is also a "+ NEW PROJECT" button. The main area is titled "Projects" and contains a table with one row. The table columns are: Name (with a checkbox), Description, Training Datasets, Validation Datasets, Test Datasets, and Experiments. The single row shows "Credit Card" as the name, "Credit card project" as the description, 2 training datasets, 0 validation datasets, 2 test datasets, and 3 experiments. Below the table are three buttons: OPEN, RENAME, and DELETE.

Name	Description	Training Datasets	Validation Datasets	Test Datasets	Experiments
Credit Card	Credit card project	2	0	2	3

OPEN

RENAME

DELETE

LEADERBOARDS

Driverless AI provides a feature to automatically create leaderboards.

The **Create Leaderboard** feature runs multiple diverse experiments that provide an overview of the dataset. This feature also provides you with relevant information for deciding on complexity, accuracy, size, and time tradeoffs when putting models into production. Refer to the *Expert Settings* topic for information on expert settings that can be used to control this feature. For more information on the default models built for a leaderboard, see *Leaderboard Models*.

The built models are placed under the *Project Workspace* page and can be simultaneously scored on the test dataset and compared.

25.1 Creating a Leaderboard

Creating a Leaderboard is similar to running a *new experiment*. Refer to the *Experiment Settings*, *Expert Settings*, and *Scorers* topics for more information about options you can set when running an experiment.

1. On the **Datasets** page, select the dataset that you want to use for the experiment, then click **Predict**
or
On the **Experiments** page, click **New Experiment**, then select the dataset that you want to use.
2. Specify whether to include dropped columns, a validation dataset, and a testing dataset.
3. Specify the Target column and optionally a fold column, weight column, and time column.
4. Optionally specify *Expert Settings*.
5. Optionally adjust the Accuracy/Time/Interpretability knobs.
6. Optionally override the default scorer.
7. Optionally override the Classification/Regression setting.
8. Optionally specify to make the experiments reproducible and/or whether to enable GPUs.
9. Click the **Create Leaderboard** button.

The screenshot shows the H2O.ai Experiment interface. At the top, it displays "DRIVERLESS AI 1.9.0 - AI TO DO AI" and "Licensed to H2O.ai (SN37). Current User - ANG". The main header is "H2O.ai Experiment". The top navigation bar includes PROJECTS, DATASETS, AUTOVIZ, EXPERIMENTS, DIAGNOSTICS, MLI, and DEPLOY.

EXPERIMENT SETUP

- DISPLAY NAME: Display name
- ROWS: 17K
- COLUMNS: 25
- DROPPED COLUMNS: 1
- VALIDATION DATASET: Yes (CreditCard_valid.csv)
- TEST DATASET: Yes (CreditCard_test.csv)
- TARGET COLUMN: default payment next
- FOLD COLUMN: Disabled
- WEIGHT COLUMN: --
- TIME COLUMN: Disabled
- TYPE: bool
- COUNT: 16784
- UNIQUE: 2
- TARGET FREQ: 3740

TRAINING SETTINGS

- ACCURACY: 6
- TIME: 3
- INTERPRETABILITY: 7
- SCORER: AUC

EXPERT SETTINGS

- CLASSIFICATION
- REPRODUCIBLE
- GPUS ENABLED

Buttons:

- LAUNCH EXPERIMENT
- CREATE LEADERBOARD

Left sidebar (What do these settings mean?)

- ACCURACY**: - Training data size: **16,784 rows, 24 cols**. - Feature evolution: **[Constant, DecisionTree, LightGBM, XGBoostGBM]**, external validation, 2 reps. - Final pipeline: Blend of up to 3 [Constant, DecisionTree, LightGBM, XGBoostGBM] models, external validation.
- TIME**: - Feature evolution: **4 individuals**, up to **36 iterations**. - Early stopping: After 5 iterations of no improvement.
- INTERPRETABILITY**: - Feature pre-pruning strategy: Permutation Importance FS. - Monotonicity constraints: enabled. - Feature engineering search space: [CVCatNumEncode, CVTargetEncode, CatOriginal, Cat, Frequent, Interactions, NumCatTE, NumToCatTE, NumToCatWoEMonotonic, NumToCatWoE, Original, WeightOfEvidence].
- Model Summary**: [Constant, DecisionTree, LightGBM, XGBoostGBM] models to train; Model and feature tuning: **144**; Feature evolution: **384**; Final pipeline: **3**.
- Runtime**: Estimated runtime: **minutes**. Auto-click Finish/Abort if not done in: **1 day/7 days**.

Driverless AI will create a new, randomly named project and begin automatically training models using the queuing mechanism. After all models have been built, you can *score each experiment* and *compare experiments*, as described in the *Project Workspace* topic.

The screenshot shows the H2O.ai Project tabemico interface. At the top, it displays "DRIVERLESS AI 1.9.0 - AI TO DO AI" and "Licensed to H2O.ai (SN37). Current User - ANG". The main header is "H2O.ai Project tabemico". The top navigation bar includes PROJECTS, DATASETS, AUTOVIZ, EXPERIMENTS, DIAGNOSTICS, MLI, and DEPLOYMENTS. There are also links for RESOURCES, MESSAGES [2], and LOGOUT.

DATASETS

Name	Rows	Columns
CreditCard_train.c...	17K	25

EXPERIMENTS

1. SELECT SCORING DATASET
1. Select Scoring Dataset

2. SELECT EXPERIMENTS
0 items selected.

3. SCORE DATASET ON EXPERIMENTS
SCORE ITEMS

SELECT SCORER FOR TEST SCORE
Select Scorer

Name	A	T	I	Scorer	Status	Train Time	Val. Score	Test Score	Test Time
Few Features ...	1	1	10	AUC	Completed	00:01:59	0.7534	NA	N/A
Simple LightG...	1	1	10	AUC	Completed	00:02:02	0.7803	NA	N/A
Constant Base...	1	1	10	AUC	Completed	00:01:21	0.5000	NA	N/A
Single Decision...	6	3	7	AUC	Completed	00:07:09	0.7721	NA	N/A
Single GLM-to...	6	3	7	AUC	Completed	00:14:48	0.7774	NA	N/A
Complex Light...	7	3	7	AUC	Completed	00:15:43	0.7837	NA	N/A
Few Features ...	6	3	7	AUC	Completed	00:07:28	0.7725	NA	N/A
Default Single ...	6	3	7	AUC	Completed	00:10:38	0.7806	NA	N/A
Default XGBoo...	6	3	7	AUC	Completed	00:07:12	0.7834	NA	N/A
Single FTRL-to...	6	3	7	AUC	Completed	00:09:02	0.7655	NA	N/A
Single TensorF...	6	3	7	AUC	Completed	00:14:15	0.7832	NA	N/A

© 2017-2020 H2O.ai. All rights reserved.

25.2 Leaderboard Models

When creating a leaderboard, the models that are built will vary based on whether you are running a regular experiment or a time-series experiment.

25.3 Regular Experiments

By default, the following models are built when creating a leaderboard for regular (non-time-series) experiments. You can omit models from being built by disabling those models in the *Expert Settings*.

Model	Accuracy	Time	Interpretability	Config Overrides
Few Features Decision Tree	1	1	10	max_orig_cols_selected=5 nfeatures_max=10
Simple LightGBM	1	1	10	
Constant Baseline	1	1	10	max_orig_cols_selected=1
Single Decision Tree	Specified in experiment	Specified in experiment	Specified in experiment	fixed_ensemble_level=0
Single GLM	Specified in experiment	Specified in experiment	Specified in experiment	fixed_ensemble_level=0
Complex LightGBM Ensemble	7	Specified in experiment	Specified in experiment	
Few Features Single LightGBM	Specified in experiment	Specified in experiment	Specified in experiment	max_orig_cols_selected=5 nfeatures_max=10 fixed_ensemble_level=0
Default Single LightGBM	Specified in experiment	Specified in experiment	Specified in experiment	fixed_ensemble_level=0
Default XG-Boost/LightGBM Ensemble	Specified in experiment	Specified in experiment	Specified in experiment	
Single FTRL	Specified in experiment	Specified in experiment	Specified in experiment	fixed_ensemble_level=0
Single TensorFlow	Specified in experiment	Specified in experiment	Specified in experiment	fixed_ensemble_level=0

25.4 Time Series Experiments

Driverless AI will build one time-series experiment using the default Driverless AI settings. The following additional models are also built when creating a leaderboard for time-series experiments.

Notes:

- Lag-based features are always included for time-series baseline models.
- Unlike in regular experiments, the leaderboard will only build models using the specified Accuracy/Time/Interpretability settings.
- Refer to the *Driverless AI Transformations* topic for information about each transformer.

Model	Included Models	Included Transformers	Config Overrides
Exponential Smoothing	GLM	EwmaLagsTransformer	nfeatures_max=10
Target Lags Decision-Tree	Decision Tree	LagsTransformer	nfeatures_max=10
Target Lags GLM	GLM	LagsTransformer, EwmaLagsTransformer, LagsAggregatesTransformer	nfeatures_max=50
Target Lags Light-GBM/XGBoost	LightGBM, XG-BoostGBM	LagsTransformer, EwmaLagsTransformer, LagsAggregatesTransformer	nfeatures_max=50

CHAPTER
TWENTYSIX

MLI OVERVIEW

Driverless AI provides robust interpretability of machine learning models to explain modeling results in a human-readable format. In the Machine Learning Interpretability (MLI) view, Driverless AI employs a host of different techniques and methodologies for interpreting and explaining the results of its models. A number of charts are generated automatically (depending on experiment type), including K-LIME, Shapley, Variable Importance, Decision Tree Surrogate, Partial Dependence, Individual Conditional Expectation, Sensitivity Analysis, NLP Tokens, NLP LOCO, and more. Additionally, you can download a CSV of LIME, Shapley, and Original (Kernel SHAP) Shapley reason codes as well as text and Python files of Decision Tree Surrogate model rules from this view.

This chapter describes Machine Learning Interpretability (MLI) in Driverless AI for both regular and time-series experiments. Refer to the following sections for more information:

- [The Interpreted Models Page](#)
- [MLI for Regular \(Non-Time-Series\) Experiments](#)
- [MLI for Time-Series Experiments](#)

Additional Resources

- [Click here](#) to download our MLI cheat sheet.
- [“An Introduction to Machine Learning Interpretability” book.](#)
- [Click here](#) to access the H2O.ai MLI Resources repository. This repo includes materials that illustrate applications or adaptations of various MLI techniques for practicing data scientists.
- [Click here](#) to view our H2O Driverless AI Machine Learning Interpretability walkthrough video.

Limitations

- This release deprecates experiments run in 1.7.0 and earlier. MLI will not be available for experiments from versions <= 1.7.0.
- MLI is not supported for Image or multiclass Time Series experiments.
- MLI does not require an Internet connection to run on current models.

CHAPTER
TWENTYSEVEN

THE INTERPRETED MODELS PAGE

Click the **MLI** link in the upper-right corner of the UI to view a list of interpreted models.

You can sort this page by Name, Target, Model, Dataset, N-Folds, Feature Set, Cluster Col, LIME Method, Status, or ETA/Runtime. You can also use the search bar to locate a specific interpreted model. To specify which columns are visible on this page, click the top right-most column, then select **Visible Columns**.

Click the right-most column of an interpreted model to view an additional menu. This menu allows you to open, rename, or delete the interpretation.

Note: Driverless AI version 1.9 features a redesigned MLI page for interpreted models. To view the legacy version of an interpreted model's MLI page, select **Open Legacy** from the menu.

Click on an interpeted model to view the MLI page for that interpretation. The MLI page that displays will vary depending on whether the experiment was a **regular experiment** or a **time series** experiment.

MLI FOR REGULAR (NON-TIME-SERIES) EXPERIMENTS

This section describes MLI functionality and features for regular experiments. Refer to [MLI for Time-Series Experiments](#) for MLI information with time-series experiments.

28.1 Interpreting a Model

There are two methods you can use for interpreting models:

- Using the **Interpret this Model** button on a completed experiment page to interpret a Driverless AI model on original and transformed features.
- Using the **MLI** link in the upper right corner of the UI to interpret either a Driverless AI model or an external model.

Notes:

- Experiments run in 1.7.0 and earlier are deprecated in this release. MLI will not be available for experiments from versions <= 1.7.0.
- MLI does not require an Internet connection to run on current models.

Intrepret from Completed Experiment Page

Interpret from MLI Page

Clicking the **Interpret this Model** button on a completed experiment page launches the Model Interpretation for that experiment. Python and Java logs can be viewed for non-time-series experiments while the interpretation is running.

For non-time-series experiments, this page provides several visual explanations and reason codes for the trained Driverless AI model and its results. More information about this page is available in the [Understanding the Model Interpretation Page](#) section later in this chapter.

Driverless AI Model

External Model

This method allows you to run model interpretation on a Driverless AI model. This method is similar to clicking “Interpret This Model” on an experiment summary page.

1. Click the **MLI** link in the upper-right corner of the UI to view a list of interpreted models.
2. Click the **New Interpretation** button.
3. Select the dataset that was used to train the model that you will use for interpretation.
4. Specify the Driverless AI model that you want to use for the interpretation. Once selected, the Target Column used for the model will be selected.
5. Optionally specify any additional MLI *Expert Settings* to use when running this interpretation.

6. Optionally specify a weight column.
7. Optionally specify one or more dropped columns. Columns that were dropped when the model was created are automatically dropped for the interpretation.
8. Click the **Launch MLI** button.

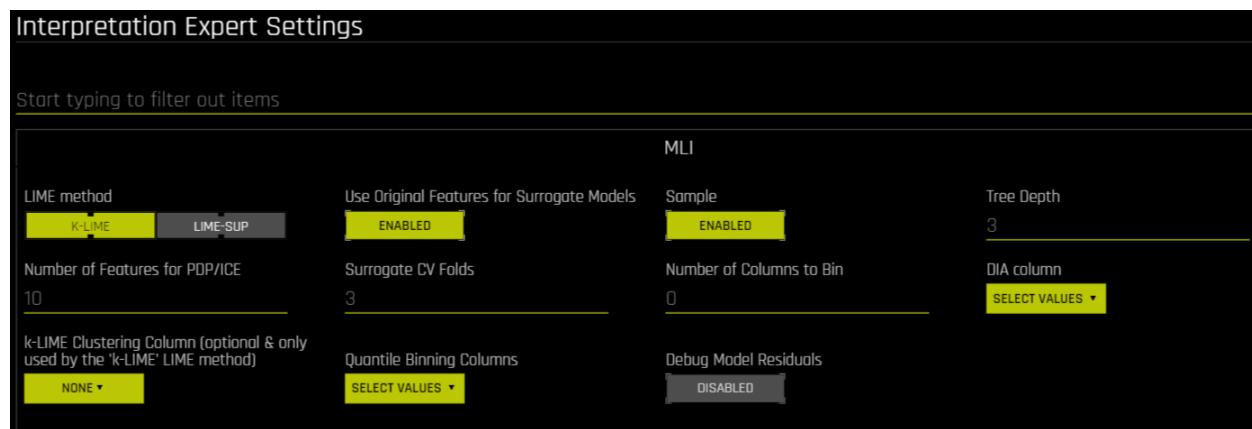
Model Interpretation does not need to be run on a Driverless AI experiment. You can train an external model and run Model Interpretability on the predictions.

1. Click the **MLI** link in the upper-right corner of the UI to view a list of interpreted models.
2. Click the **New Interpretation** button.
3. Select the dataset that you want to use for the model interpretation. This must include a prediction column that was generated by the external model. If the dataset does not have predictions, then you can join the external predictions. An example showing how to do this in Python is available in the [Run Model Interpretation on External Model Predictions](#) section of the Credit Card Demo.
4. Specify a Target Column (actuals) and the Prediction Column (scores from the model).
5. Optionally specify any additional MLI *Expert Settings* to use when running this interpretation.
6. Optionally specify a weight column.
7. Optionally specify one or more dropped columns. Columns that were dropped when the model was created are automatically dropped for the interpretation.
8. Click the **Launch MLI** button.

Note: When running interpretations on an external model, leave the **Select Model** option empty. That option is for selecting a Driverless AI model.

28.1.1 Interpretation Expert Settings

Driverless AI provides a variety of options in the Interpretation Expert Settings that allow you to customize interpretations. Use the search bar to refine the list of settings or locate a specific setting.



Notes:

- The selection of available expert settings is determined by the type of model you want to interpret and the specified LIME method.
- Expert settings are not available for time-series models.

LIME Method

Select a LIME method of either KLIME (default) or LIME-SUP.

- **KLIME** (default): creates one global surrogate GLM on the entire training data and also creates numerous local surrogate GLMs on samples formed from k -means clusters in the training data. The features used for k -means are selected from the Random Forest surrogate model's variable importance. The number of features used for k -means is the minimum of the top 25% of variables from the Random Forest surrogate model's variable importance and the max number of variables that can be used for k -means, which is set by the user in the config.toml setting for `mli_max_number_cluster_vars`. (Note, if the number of features in the dataset are less than or equal to 6, then all features are used for k -means clustering.) The previous setting can be turned off to use all features for k -means by setting `use_all_columns_klime_kmeans` in the config.toml file to `true`. All penalized GLM surrogates are trained to model the predictions of the Driverless AI model. The number of clusters for local explanations is chosen by a grid search in which the R^2 between the Driverless AI model predictions and all of the local KLIME model predictions is maximized. The global and local linear model's intercepts, coefficients, R^2 values, accuracy, and predictions can all be used to debug and develop explanations for the Driverless AI model's behavior.
- **LIME-SUP**: explains local regions of the trained Driverless AI model in terms of the original variables. Local regions are defined by each leaf node path of the decision tree surrogate model instead of simulated, perturbed observation samples - as in the original LIME. For each local region, a local GLM model is trained on the original inputs and the predictions of the Driverless AI model. Then the parameters of this local GLM can be used to generate approximate, local explanations of the Driverless AI model.

Use Original Features for Surrogate Models

Specify whether to use original features or transformed features in the surrogate model for the new interpretation. This is enabled by default.

Note: When this setting is disabled, the KLIME clustering column and quantile binning options are unavailable.

Sample

Specify whether to perform the interpretation on a sample of the training data. By default, MLI will sample the training dataset if it is greater than 100k rows. (Note that this value can be modified in the config.toml setting for `mli_sample_size`.) Turn this toggle off to run MLI on the entire dataset.

Tree Depth

For KLIME interpretations, specify the depth that you want for your decision tree surrogate model. The tree depth value can be a value from 2-5 and defaults to 3. For LIME-SUP interpretations, specify the LIME-SUP tree depth. This can be a value from 2-5 and defaults to 3.

Number of Features for PDP/ICE

Specify the number of top features for which partial dependence and ICE will be computed. This value defaults to 10. Setting a value greater than 10 can significantly increase the computation time. Setting this to -1 specifies to use all features.

Surrogate CV Folds

Specify the number of surrogate cross-validation folds to use (from 0 to 10). When running experiments, Driverless AI automatically splits the training data and uses the validation data to determine the performance of the model parameter tuning and feature engineering steps. For a new interpretation, Driverless AI uses 3 cross-validation folds by default for the interpretation.

Number of Columns to Bin

Specify the number of columns to bin. This value defaults to 0.

DIA Column

For binary classification and regression experiments, specify which columns to have Disparate Impact Analysis (DIA) applied to.

KLIME Clustering Column

For KLIME interpretations, specify which columns to have KLIME clustering applied to.

Quantile Binning Columns

For KLIME interpretations, specify one or more columns to generate decile bins (uniform distribution) to help with MLI accuracy. Columns selected are added to top n columns for quantile binning selection. If a column is not numeric or not in the dataset (transformed features), then the column will be skipped.

Note: This option is only available when the **Use Original Features for Surrogate Models** setting is enabled.

Debug Model Residuals

Specify whether to enable debugging on model residuals. This is disabled by default. Refer to the surrogate-models-on-residuals section for more information.

Class for Debugging Classification Model Logloss Residuals

Specify a target class when debugging classification models. When this setting is enabled, Driverless AI builds MLI surrogate models with the residual that is obtained for that particular class. Refer to the surrogate-models-on-residuals section for more information.

Note: This setting is only available for classification experiments when the **Debug Model Residuals** interpretation expert setting is enabled.

28.2 Understanding the Model Interpretation Page

This section describes the features on the Model Interpretation page for non-time-series experiments.

The Model Interpretation page is organized into three tabs:

Summary Tab

DAI Model Tab

Surrogate Model Tab

The Summary tab provides an overview of the interpretation, including the dataset and Driverless AI experiment (if available) that were used for the interpretation along with the feature space (original or transformed), target column, problem type, and k-Lime information. If the interpretation was created from a Driverless AI model, then a table with the Driverless AI model summary is also included along with the top variables for the model.

The screenshot shows the H2O.ai interface for MLI: Regression and Classification Explanations. At the top, there's a navigation bar with links for PROJECTS, DATASETS, AUTOVIZ, EXPERIMENTS, DIAGNOSTICS, MLI, DEPLOYMENTS, RESOURCES, MESSAGES (3), and LOGOUT. Below the navigation bar, the title 'MLI: Regression and Classification Explanations' is displayed. The main content area has three tabs: SUMMARY (selected), DAI MODEL, and SURROGATE MODEL. Under the SUMMARY tab, there's a section titled 'MACHINE LEARNING INTERPRETABILITY (MLI) EXPERIMENT OVERVIEW'. It states: 'MLI was applied to the **kunabiso** (2a8f9048-7ca0-11ea-8d96-0eo085361175) Driverless AI (DAI) model, in relation to the target column (**default payment next month**)'. Below this, there's a 'MLI Parameters Summary' table with the following data:

MLI Experiment:	minogito (4ba94462-7ca1-11ea-8d96-0eo085361175)
DAI Experiment:	kunabiso (2a8f9048-7ca0-11ea-8d96-0eo085361175)
Dataset:	CreditCard-train.csv
Feature Space for Surrogate Models:	Original Features
Target Column:	default payment next month
Problem Type:	classification
Surrogate CV Folds:	3
LIME Method:	k-LIME
LIME used k-means to cluster data based on these variables (top 6 from DRF surrogate model variable importance):	PAY_0, PAY_3, PAY_4, PAY_2, PAY_6, PAY_5
LIME Number of Clusters:	15
Tree Depth for Decision Tree Surrogate Model:	3
MLI Duration:	00:02:00

Below the parameters summary is a 'DAI Model Summary' table:

DAI Model	Validation AUC	Test AUC	Accuracy	Time	Interpretability	Training Duration

At the bottom of the page, there's a footer note: '© 2017-2020 H2O.ai. All rights reserved.'

The DAI Model tab is organized into tiles for each interpretation method. To view a specific plot, click the tile for the plot that you want to view.

For binary classification and regression experiments, this tab includes Feature Importance and Shapley (not supported for RuleFit and TensorFlow models) plots for original and transformed features as well as Partial Dependence/ICE, Disparate Impact Analysis (DIA), Sensitivity Analysis, NLP Tokens and NLP LOCO (for text experiments), and Permutation Feature Importance (if the `autodoc_include_permutation_feature_importance` configuration option is enabled) plots. For multiclass classification experiments, this tab includes Feature Importance and Shapley plots for original and transformed features. See the [DAI Model Tab Plots](#) section for more information on these plots.

Notes:

- Shapley plots are not supported for RuleFit, FTRL, and TensorFlow models.
- Shapley plots are only supported for BYOR models that implement the `has_pred_contribs` method (and return `True`) and implement proper handling of the argument `pred_contribs=True` in the `predict` method.

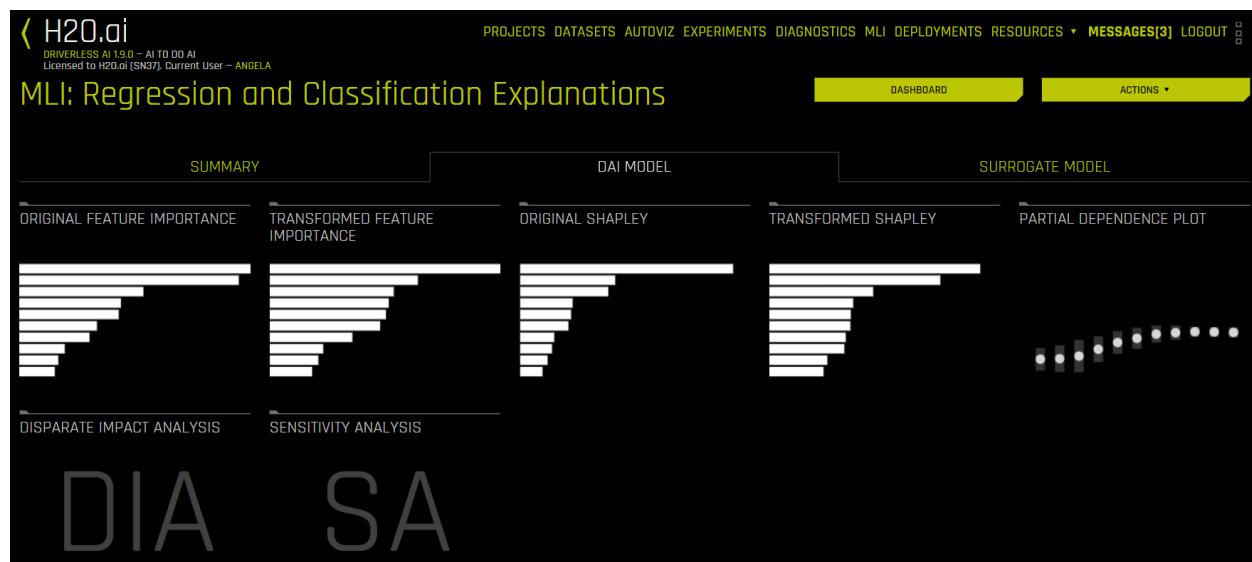
- The Permutation-based feature importance plot is only available when the `autodoc_include_permutation_feature_importance` configuration option is enabled when starting Driverless AI or when starting the experiment.

- On the Feature Importance and Shapley plots, the transformed feature names are encoded as follows:

```
<transformation/gene_details_id>_<transformation_name>:<orig>:<...>:<orig>.<extra>
```

So in `32_NumToCatTE:BILL_AMT1:EDUCATION:MARRIAGE:SEX.0`, for example:

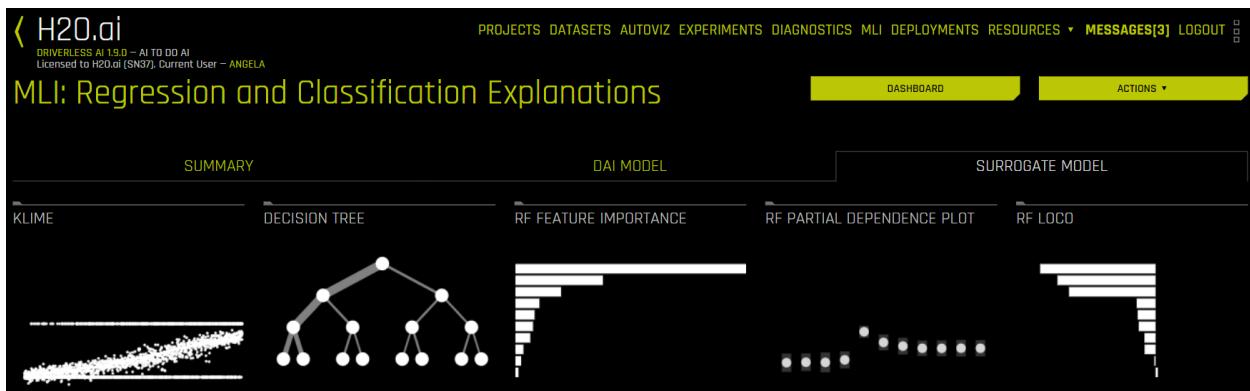
- `32_` is the transformation index for specific transformation parameters.
- `NumToCatTE` is the transformation type.
- `BILL_AMT1:EDUCATION:MARRIAGE:SEX` represent original features used.
- `0` represents the likelihood encoding for target[0] after grouping by switch and making out-of-fold estimates. For multiclass experiments this value will be > 0 . For binary experiments, this value will always be `0`.



A surrogate model is a data mining and engineering technique in which a generally simpler model is used to explain another, usually more complex, model or phenomenon. For example, the decision tree surrogate model is trained to predict the predictions of the more complex Driverless AI model using the original model inputs. The trained surrogate model enables a heuristic understanding (i.e., not a mathematically precise understanding) of the mechanisms of the highly complex and nonlinear Driverless AI model.

The Surrogate Model tab is organized into tiles for each interpretation method. To view a specific plot, click the tile for the plot that you want to view. For binary classification and regression experiments, this tab includes K-LIME/LIME-SUP and Decision Tree plots as well as Feature Importance, Partial Dependence, and LOCO plots for the Random Forest surrogate model. See the [Surrogate Model Tab Plots](#) section for more information on these plots.

Note: For multiclass classification experiments, only the Decision Tree and Random Forest Feature Importance plots are available in this tab.



The Model Interpretation page also features the **Dashboard** and **Actions** buttons, which are located in the upper-right corner:



Dashboard Button

Actions Button

Click the **Dashboard** button to view the Dashboard page. For binary classification and regression experiments, the Dashboard page provides a single page with the following surrogate plots:

- Global Interpretable Model Explanations
- Feature Importance
- Decision Tree
- Partial Dependence

You can also view explanations from this page by clicking the **Explanations** button located in the upper-right corner. Refer to the viewing-explanations section for more information.

Note: The Dashboard is not available for multiclass classification experiments.



Click the **Actions** button to view the following options:

- **Go to MLI Documentation:** View the Machine Learning Interpretability section of the Driverless AI documentation.
- **Download MLI Logs:** Download a ZIP file of the logs that were generated during the interpretation.
- **Go to Experiment:** View the experiment that was used to generate the interpretation.
- **Download Scoring Pipeline:** For binomial and regression experiments, download the scoring pipeline for the interpretation. This option is not available for multinomial experiments.
- **Download Reason Codes LIME:** For binomial experiments, download a CSV file of LIME reason codes.
- **Download Reason Codes Shapley:** For regression, binary, and multinomial experiments, download a CSV file of Shapley reason codes.
- **Display MLI Java Logs:** View MLI Java logs for the interpretation.
- **Display MLI Python Logs:** View MLI Python logs for the interpretation.
- **Download Decision Tree Surrogate Rules:** Download text and Python files of decision tree surrogate model rules for the interpretation.
- **Download Reason Codes Original Shapley (Kernel Shapley):** For regression, binary, and multinomial experiments, download a CSV file of Original Shapley reason codes.

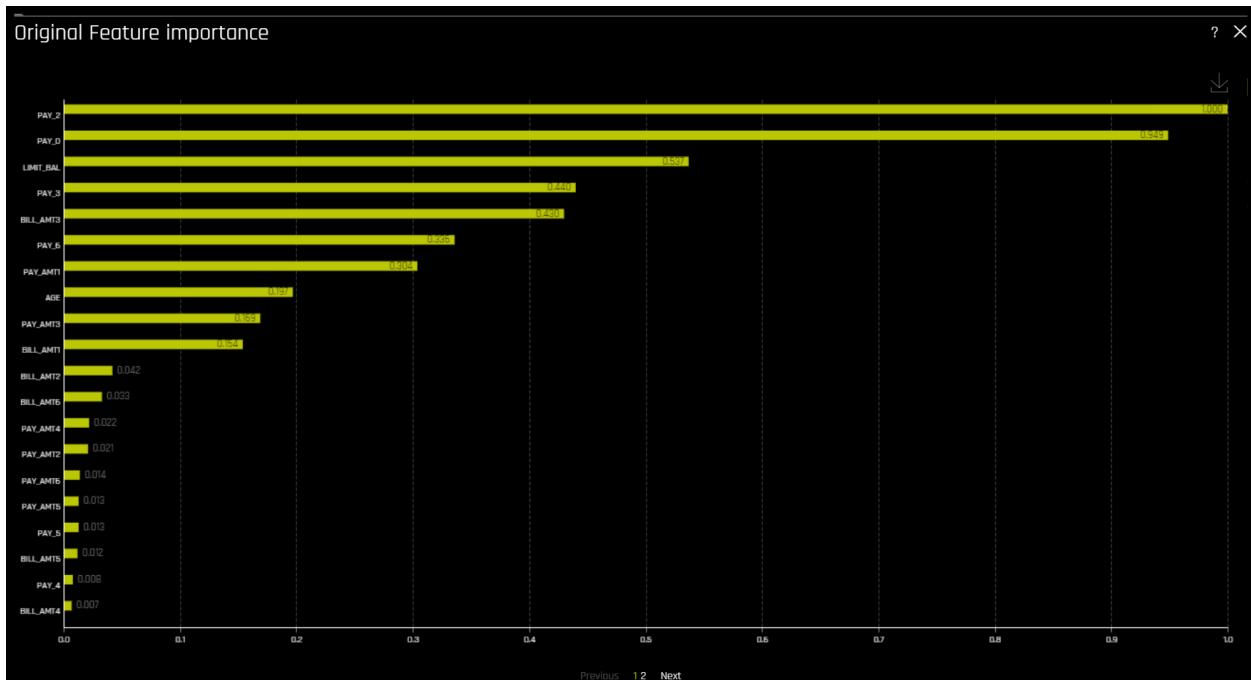
28.2.1 DAI Model Tab Plots

This section describes the plots that are available in the DAI Model Tab.

Feature Importance (Original and Transformed Features)

This plot is available for all models for binary classification, multiclass classification, and regression experiments.

This plot shows the Driverless AI feature importance. Driverless AI feature importance is a measure of the contribution of an input variable to the overall predictions of the Driverless AI model.



Shapley (Original and Transformed Features)

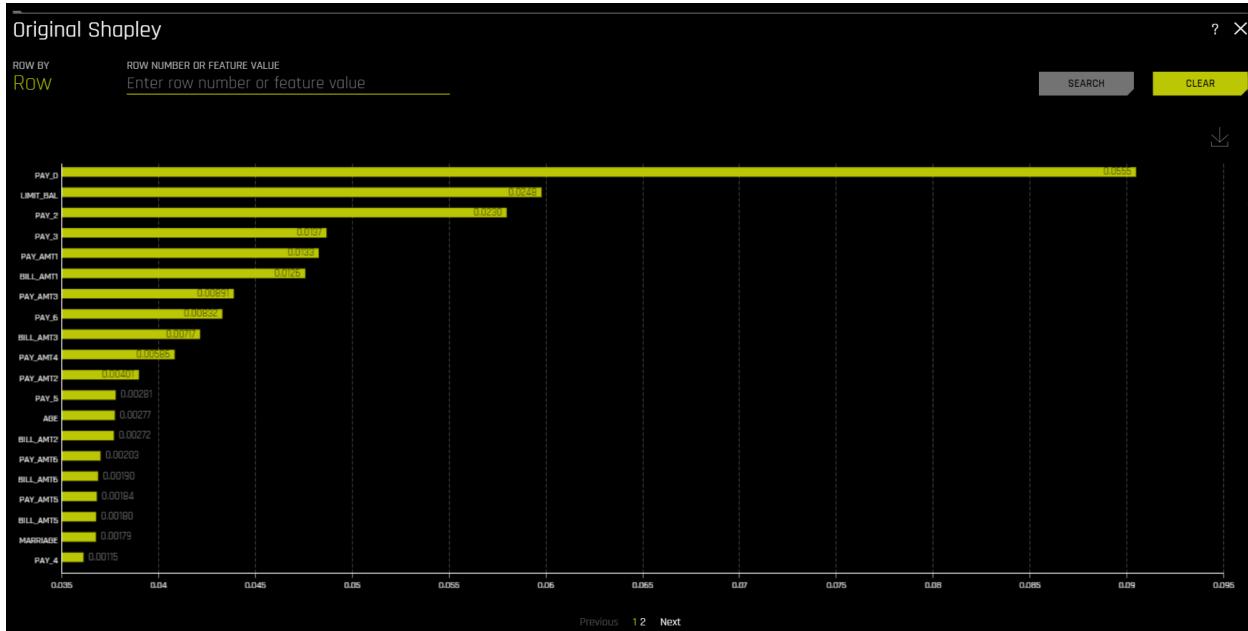
This plot is not available for RuleFit or TensorFlow models. For all other models, this plot is available for binary classification, multiclass classification, and regression experiments.

Shapley explanations are a technique with credible theoretical support that presents consistent global and local variable contributions. Local numeric Shapley values are calculated by tracing single rows of data through a trained tree ensemble and aggregating the contribution of each input variable as the row of data moves through the trained ensemble. For regression tasks, Shapley values sum to the prediction of the Driverless AI model. For classification problems, Shapley values sum to the prediction of the Driverless AI model before applying the link function. Global Shapley values are the average of the absolute Shapley values over every row of a dataset.

Notes:

- Shapley values for transformed features are calculated by tracing individual rows of data through a trained tree ensemble and then aggregating the contribution of each input variable as the row of data moves through the trained ensemble. More information about Shapley for tree-based models is available at <https://arxiv.org/abs/1706.06060>.
- Shapley values for original features are calculated with the Kernel Explainer method, which uses a special weighted linear regression to compute the importance of each feature. More information about Kernel SHAP is

available at <http://papers.nips.cc/paper/7062-a-unified-approach-to-interpreting-model-predictions.pdf>. If Kernel Explainer is disabled, Shapley values for original features are approximated from the accompanying Shapley values for transformed features with the Naive Shapley method. For example, if the transformed feature `feature1_feature2` has a Shapley value of 0.5, then the Shapley value of the original features `feature1` and `feature2` will be 0.25.



The **Showing n Features** dropdown for Feature Importance and Shapley plots allows you to select between original and transformed features. If there are a significant amount of features, they will be organized in numbered pages that can be viewed individually. **Note:** The provided original values are approximations derived from the accompanying transformed values. For example, if the transformed feature *feature1_feature2* has a value of 0.5, then the value of the original features (*feature1* and *feature2*) will be 0.25.

Partial Dependence (PDP) and Individual Conditional Expectation (ICE)

A Partial Dependence and ICE plot is available for both Driverless AI and surrogate models.

The Partial Dependence Technique

Partial dependence is a measure of the average model prediction with respect to an input variable. Partial dependence plots display how machine-learned response functions change based on the values of an input variable of interest, while taking nonlinearity into consideration and averaging out the effects of all other input variables. Partial dependence plots are well-known and described in the Elements of Statistical Learning (Hastie et al, 2001). Partial dependence plots enable increased transparency in Driverless AI models and the ability to validate and debug Driverless AI models by comparing a variable's average predictions across its domain to known standards, domain knowledge, and reasonable expectations.

The ICE Technique

This plot is available for binary classification and regression models.

Individual conditional expectation (ICE) plots, a newer and less well-known adaptation of partial dependence plots, can be used to create more localized explanations for a single individual using the same basic ideas as partial dependence plots. ICE Plots were described by Goldstein et al (2015). ICE values are simply disaggregated partial dependence, but ICE is also a type of nonlinear sensitivity analysis in which the model predictions for a single row are measured while a variable of interest is varied over its domain. ICE plots enable a user to determine whether the model's treatment of an individual row of data is outside one standard deviation from the average model behavior, whether the treatment of a specific row is valid in comparison to average model behavior, known standards, domain knowledge, and reasonable expectations, and how a model will behave in hypothetical situations where one variable in a selected row is varied across its domain.

Given the row of input data with its corresponding Driverless AI and K-LIME predictions:

debt_to_income_ratio	credit_score	savings_acct_balance	observed_default	H2OAI_predicted_default	K-LIME_predicted_default
30	600	1000	1	0.85	0.9

Taking the Driverless AI model as $F(\mathbf{X})$, assuming credit scores vary from 500 to 800 in the training data, and that increments of 30 are used to plot the ICE curve, ICE is calculated as follows:

$$\text{ICE}_{\text{credit_score}, 500} = F(30, 500, 1000)$$

$$\text{ICE}_{\text{credit_score}, 530} = F(30, 530, 1000)$$

$$\text{ICE}_{\text{credit_score}, 560} = F(30, 560, 1000)$$

...

$$\text{ICE}_{\text{credit_score}, 800} = F(30, 800, 1000)$$

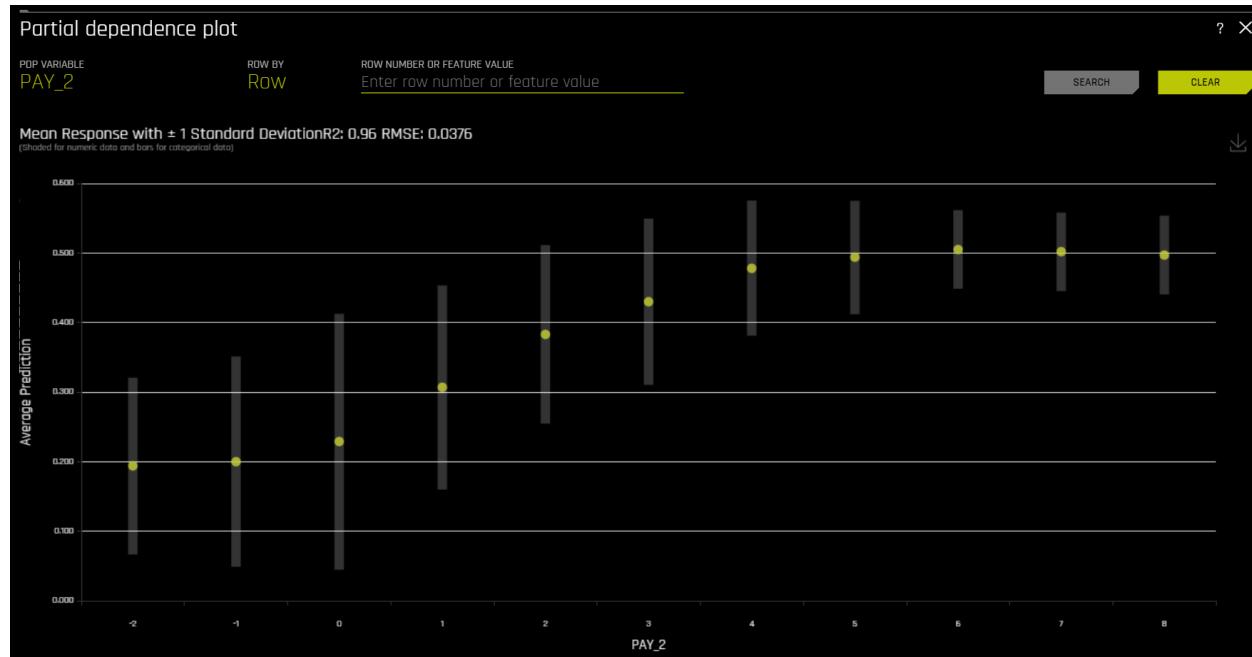
The one-dimensional partial dependence plots displayed here do not take interactions into account. Large differences in partial dependence and ICE are an indication that strong variable interactions may be present. In this case partial dependence plots may be misleading because average model behavior may not accurately reflect local behavior.

The Partial Dependence Plot

This plot is available for binary classification and regression models.

Overlaying ICE plots onto partial dependence plots allow the comparison of the Driverless AI model's treatment of certain examples or individuals to the model's average predictions over the domain of an input variable of interest.

This plot shows the partial dependence when a variable is selected and the ICE values when a specific row is selected. Users may select a point on the graph to see the specific value at that point. Partial dependence (yellow) portrays the average prediction behavior of the Driverless AI model across the domain of an input variable along with ± 1 standard deviation bands. ICE (grey) displays the prediction behavior for an individual row of data when an input variable is toggled across its domain. Currently, partial dependence and ICE plots are only available for the top ten most important original input variables. Categorical variables with 20 or more unique values are never included in these plots.



Disparate Impact Analysis

This plot is available for binary classification and regression models.

DIA is a technique that is used to evaluate fairness. Bias can be introduced to models during the process of collecting, processing, and labeling data—as a result, it is important to determine whether a model is harming certain users by making a significant number of biased decisions.

DIA typically works by comparing aggregate measurements of unprivileged groups to a privileged group. For instance, the proportion of the unprivileged group that receives the potentially harmful outcome is divided by the proportion of the privileged group that receives the same outcome—the resulting proportion is then used to determine whether the model is biased. Refer to the **Summary** section to determine if a categorical level (for example, Fairness Female) is fair in comparison to the specified reference level and user-defined thresholds. **Fairness All** is a true or false value that is only true if every category is fair in comparison to the reference level.

Disparate impact testing is best suited for use with constrained models in Driverless AI, such as linear models, monotonic GBMs, or RuleFit. The average group metrics reported in most cases by DIA may miss cases of local discrimination, especially with complex, unconstrained models that can treat individuals very differently based on small changes in their data attributes.

DIA allows you to specify a **disparate impact variable** (the group variable that is analyzed), a **reference level** (the group level that other groups are compared to), and **user-defined thresholds for disparity**. Several tables are provided as part of the analysis:

- **Group metrics:** The aggregated metrics calculated per group. For example, true positive rates per group.
- **Group disparity:** This is calculated by dividing the `metric_for_group` by the `reference_group_metric`. Disparity is observed if this value falls outside of the user-defined thresholds.
- **Group parity:** This builds on Group disparity by converting the above calculation to a true or false value by applying the user-defined thresholds to the disparity values.

In accordance with the established four-fifths rule, user-defined thresholds are set to 0.8 and 1.25 by default. These thresholds will generally detect if the model is (on average) treating the non-reference group 20% more or less favor-

ably than the reference group. Users are encouraged to set the user-defined thresholds to align with their organization's guidance on fairness thresholds.

Metrics - Binary Classification

The following are formulas for error metrics and parity checks utilized by binary DIA. Note that in the tables below:

- tp = true positive
- fp = false positive
- tn = true negative
- fn = false negative

Error Metric / Parity Metric	Formula
Adverse Impact	$(tp + fp) / (tp + tn + fp + fn)$
Accuracy	$(tp + tn) / (tp + tn + fp + fn)$
True Positive Rate	$tp / (tp + fn)$
Precision	$tp / (tp + fp)$
Specificity	$tn / (tn + fp)$
Negative Predicted Value	$tn / (tn + fn)$
False Positive Rate	$fp / (tn + fp)$
False Discovery Rate	$fp / (tp + fp)$
False Negative Rate	$fn / (tp + fn)$
False Omissions Rate	$fn / (tn + fn)$

Parity Check	Description
Type I Parity	Fairness in both FDR Parity and FPR Parity
Type II Parity	Fairness in both FOR Parity and FNR Parity
Equalized Odds	Fairness in both FPR Parity and TPR Parity
Supervised Fairness	Fairness in both Type I and Type II Parity
Overall Fairness	Fairness across all parities for all metrics where: <ul style="list-style-type: none"> • tp == true positive • fp == false positive • tn == true negative • fn == false negative

Metrics - Regression

The following are metrics utilized by regression DIA:

- **Mean Prediction:** The mean of all predictions
- **Std.Dev Prediction:** The standard deviation of all predictions
- **Maximum Prediction:** The prediction with the highest value
- **Minimum Prediction:** The prediction with the lowest value
- **R2:** The measure that represents the proportion of the variance for a dependent variable that is explained by an independent variable or variables
- **RMSE:** The measure of the differences between values predicted by a model and the values actually observed

Fairness Metrics

DIA calculates Marginal Error (ME), Adverse Impact Ratio (AIR), and Standardized Mean Difference (SMD) for binary models and SMD for regression models.

- **ME** is the difference between the percent of the control group members receiving a favorable outcome and the percent of the protected class members receiving a favorable outcome:

$$ME \equiv 100 \cdot (\Pr(\hat{y} = 1|X_c = 1) - \Pr(\hat{y} = 1|X_p = 1))$$

Where:

- \hat{y} is the model decisions.
- X_c and X_p are binary markers created from some demographic attribute.
- c is the control group.
- p is the protected group.
- $Pr(\cdot)$ is the operator for conditional probability.
- **AIR** is equal to the ratio of the proportion of the protected class that receives a favorable outcome and the proportion of the control class that receives a favorable outcome:

$$AIR \equiv \frac{\Pr(\hat{y} = 1|X_p = 1)}{\Pr(\hat{y} = 1|X_c = 1)}$$

Where:

- \hat{y} is the model decisions.
- X_p and X_c are binary markers created from some demographic attribute.
- c is the control group.
- p is the protected group.
- $Pr(\cdot)$ is the operator for conditional probability.
- **SMD** is used to assess disparities in continuous features such as income differences in employment analyses or interest rate differences in lending:

$$SMD \equiv \frac{\bar{\hat{y}}_p - \bar{\hat{y}}_c}{\sigma_{\hat{y}}}$$

Where:

- $\bar{\hat{y}}$ is the difference in the average protected class outcome.
- $\bar{\hat{y}}$ is the control class outcome.
- $\sigma_{\hat{y}}$ is a measure of the standard deviation of the population.

Notes:

- Although the process of DIA is the same for both classification and regression experiments, the returned information is dependent on the type of experiment being interpreted. An analysis of a regression experiment returns an actual vs. predicted plot, while an analysis of a binary classification experiment returns confusion matrices.
- Users are encouraged to consider the explanation dashboard to understand and augment results from disparate impact analysis. In addition to its established use as a fairness tool, users may want to consider disparate impact for broader model debugging purposes. For example, users can analyze the supplied confusion matrices and group metrics for important, non-demographic features in the Driverless AI model.

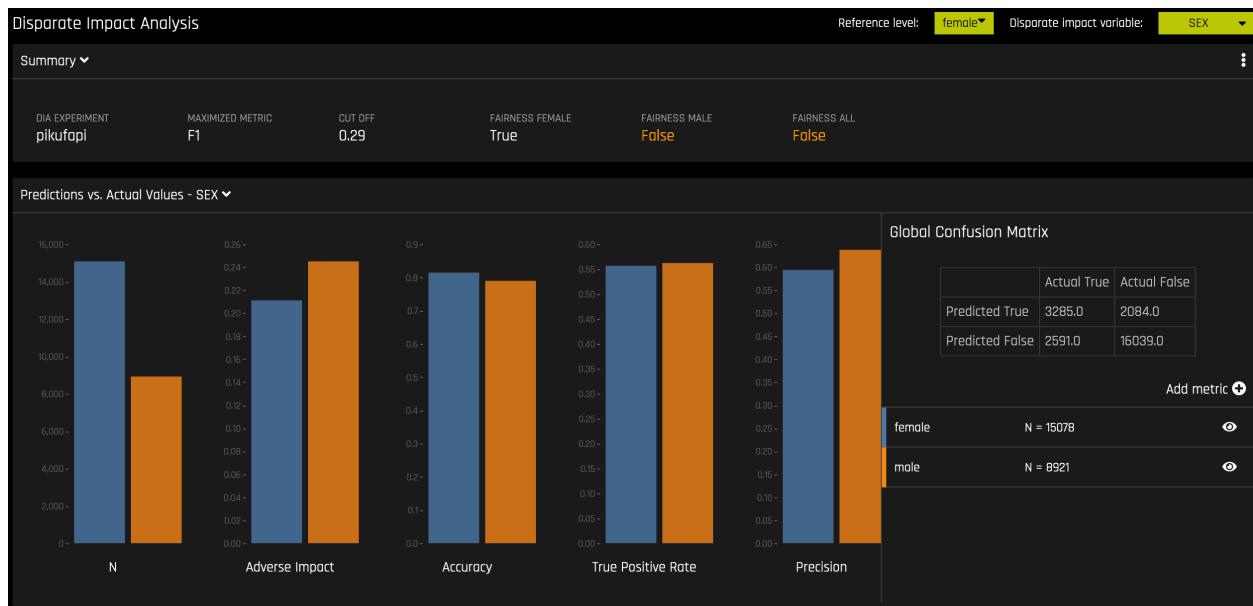


Fig. 1: Classification Experiment

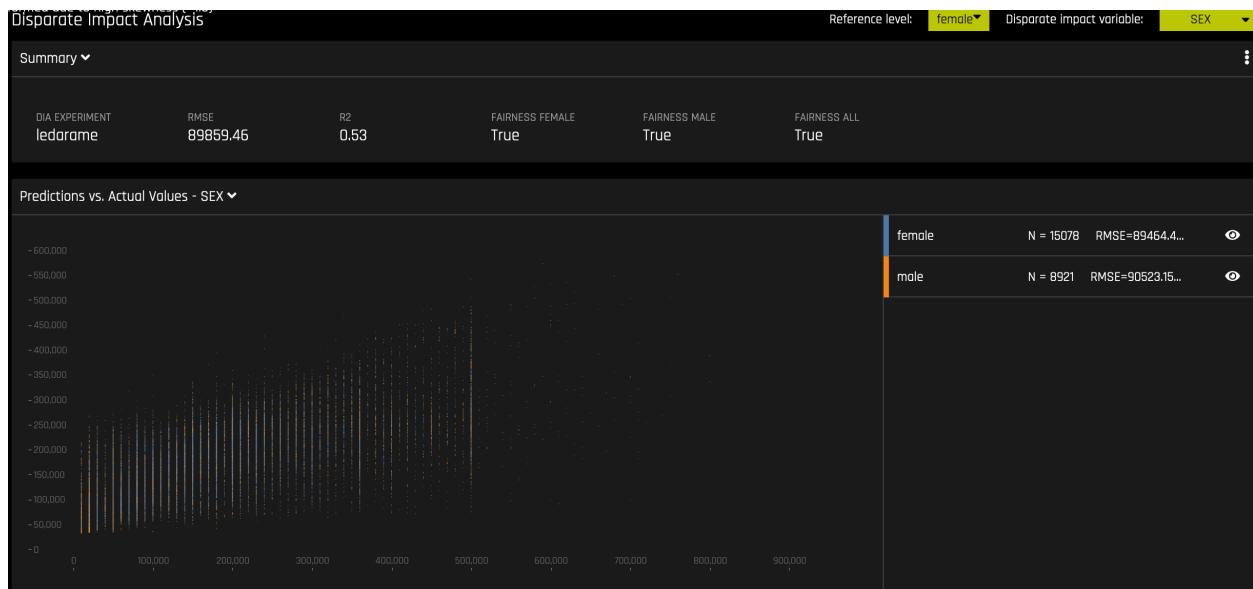


Fig. 2: Regression Experiment

Sensitivity Analysis

Note: Sensitivity Analysis (SA) is not available for multiclass experiments.

Sensitivity Analysis (or “What if?”) is a simple and powerful model debugging, explanation, fairness, and security tool. The idea behind SA is both direct and simple: Score your trained model on a single row, on multiple rows, or on an entire dataset of potentially interesting simulated values and compare the model’s new outcome to the predicted outcome on the original data.

Beyond traditional assessment practices, sensitivity analysis of machine learning model predictions is perhaps the most important validation technique for machine learning models. Sensitivity analysis investigates whether model behavior and outputs remain stable when data is intentionally perturbed or other changes are simulated in the data. Machine learning models can make drastically differing predictions for only minor changes in input variable values. For example, when looking at predictions that determine financial decisions, SA can be used to help you understand the impact of changing the most important input variables and the impact of changing socially sensitive variables (such as Sex, Age, Race, etc.) in the model. If the model changes in reasonable and expected ways when important variable values are changed, this can enhance trust in the model. Similarly, if the model changes to sensitive variables have minimal impact on the model, then this is an indication of fairness in the model predictions.

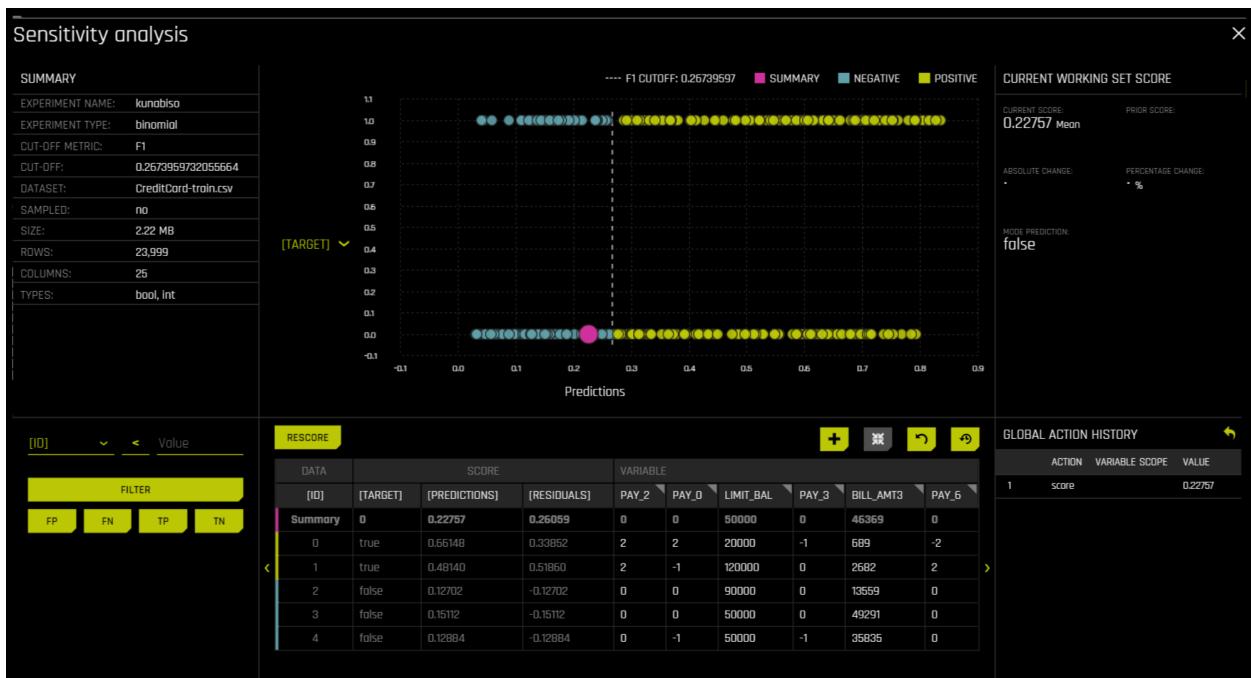
This page utilizes the [What If Tool](#) for displaying the SA information.

The top portion of this page includes:

- A summary of the experiment
- Predictions for a specified column. Change the column on the Y axis to view predictions for that column.
- The current working score set. This updates each time you rescore.

The bottom portion of this page includes:

- A filter tool for filtering the analysis. Choose a different column, predictions, or residuals. Set the filter type (<, >, etc.). Choose to filter by False Positive, False Negative, True Positive, or True Negative.
- Scoring chart. Click the **Rescore** button after applying a filter to update the scoring chart. This chart also allows you to add or remove variables, toggle the main chart aggregation, reset the data, and delete the global history while resetting the data.
- The current history of actions taken on this page. You can delete individual actions by selecting the action and then clicking the Delete button that appears.



Use Case 1: Using SA on a Single Row or on a Small Group of Rows

This section describes scenarios for using SA for explanation, debugging, security, or fairness when scoring a trained model on a single row or on a small group of rows.

- **Explanation:** Change values for a variable, and then rescore the model. View the difference between the original prediction and the new model prediction. If the change is big, then the changed variable is locally important.
- **Debugging:** Change values for a variable, and then rescore the model. View the difference between the original prediction and the new model prediction and determine whether the change to variable made the model more or less accurate.
- **Security:** Change values for a variable, and then rescore the model. View the difference between the original prediction and the new model prediction. If the change is big, then the user can, for example, inform their IT department that this variable can be used in an adversarial attack or inform the model makers that this variable should be more regularized.
- **Fairness:** Change values for a demographic variable, and then rescore the model. View the difference between the original prediction and the new model prediction. If change is big, then the user can consider using a different model, regularizing the model more, or applying post-hoc bias remediation techniques.
- **Random:** Set variables to random values, and then rescore the model. This can help you look for things the you might not have thought of.

Use Case 2: Using SA on an Entire Dataset and Trained Model

This section describes scenarios for using SA for explanation, debugging, security, or fairness when scoring a trained model for an entire dataset and trained predictive model.

- **Financial Stress Testing:** Assume the user wants to see how their loan default rates will change (according to their trained probability of default model) when they change an entire dataset to simulate that all their customers are under more financial stress (such as lower FICO scores, lower savings balances, higher unemployment, etc). Change the values of the variables in their entire dataset, and look at the **Percentage Change** in the average model score (default probability) on the original and new data. They can then use this discovered information along with external information and processes to understand whether their institution has enough cash on hand to be prepared for the simulated crisis.

- **Random:** Set variables to random values, and then rescore the model. This allows the user to look for things the user might not have thought of.

Additional Resources

Sensitivity Analysis on a Driverless AI Model: This ipynb uses the UCI credit card default data to perform sensitivity analysis and test model performance.

NLP Tokens

This plot is available for natural language processing (NLP) models. It is located in the **Dataset** tab on the Model Interpretation page (only visible for NLP models).

This plot shows both the global and local importance values of each token in a corpus (a large and structured set of texts). The corpus is automatically generated from text features used by Driverless AI models prior to the process of tokenization.

Local importance values are calculated by using the term frequency-inverse document frequency (TFIDF) as a weighting factor for each token in each row. The TFIDF increases proportionally to the number of times a token appears in a given document and is offset by the number of documents in the corpus that contain the token. Specify the row that you want to view, then click the **Search** button to see the local importance of each token in that row.

Global importance values are calculated by using the inverse document frequency (IDF), which measures how common or rare a given token is across all documents. (Default View)

Notes:

- MLI support for NLP is not available for multinomial experiments.
- MLI for NLP does not currently feature the option to remove stop words.
- By default, up to 10,000 tokens are created during the tokenization process. This value can be changed in the configuration.
- By default, Driverless AI uses up to 10,000 documents to extract tokens from. This value can be changed with the `config.mli_nlp_sample_limit` parameter. Downsampling is used for datasets that are larger than the default sample limit.
- Driverless AI does not currently generate a K-LIME scoring pipeline for MLI NLP problems.



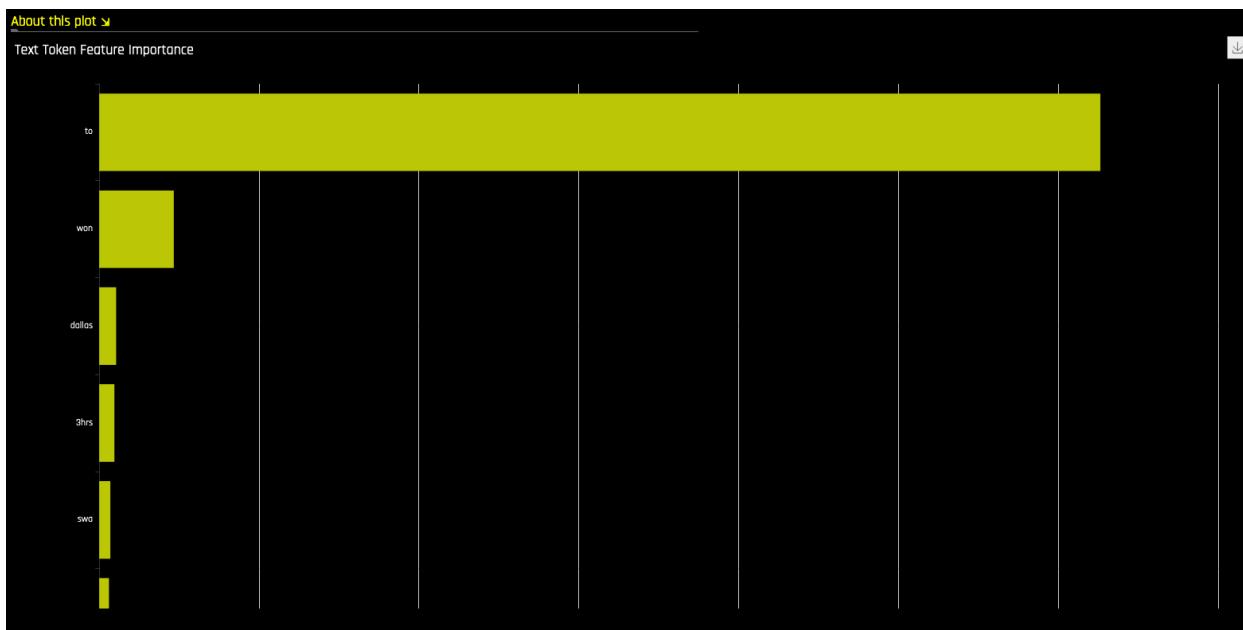
NLP LOCO

This plot is available for natural language processing (NLP) models.

This plot applies a leave-one-covariate-out (LOCO) styled approach to NLP models by removing a specific token from all text features in a record and predicting local importance without that token. The difference between the resulting score and the original score (token included) is useful when trying to determine how specific changes to text features alter the predictions made by the model.

Notes:

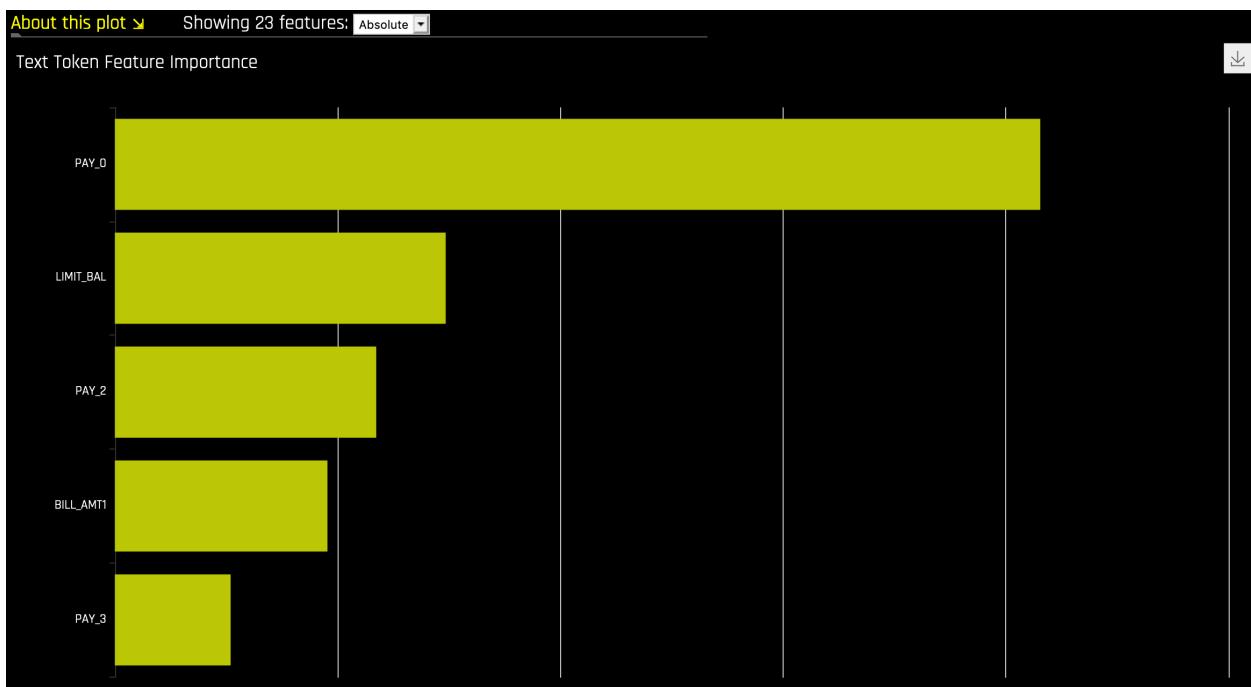
- MLI support for NLP is not available for multinomial experiments.
- Due to computational complexity, the global importance value is only calculated for N (20 by default) tokens. This value can be changed with the `mli_nlp_top_n` configuration option.
- A specific token selection method can be used by specifying one of the following options for the `mli_nlp_min_token_mode` configuration option:
 - `linspace`: Selects N evenly spaced tokens according to their IDF score (Default)
 - `top`: Selects top N tokens by IDF score
 - `bottom`: Selects bottom N tokens by IDF score
- Local values for NLP LOCO can take a significant amount of time to calculate depending on the specifications of your hardware.
- Driverless AI does not currently generate a K-LIME scoring pipeline for MLI NLP problems.



Permutation Feature Importance

Note: The Permutation-based feature importance plot is only available if the `autodoc_include_permutation_feature_importance` configuration option was enabled when starting Driverless AI or when starting the experiment. In addition, this plot is only available for binary classification and regression experiments.

Permutation-based feature importance shows how much a model's performance would change if a feature's values were permuted. If the feature has little predictive power, shuffling its values should have little impact on the model's performance. If a feature is highly predictive, however, shuffling its values should decrease the model's performance. The difference between the model's performance before and after permuting the feature provides the feature's absolute permutation importance.



28.2.2 Surrogate Model Tab Plots

This section describes the plots that are available in the Surrogate Model Tab.

K-LIME and LIME-SUP

The MLI screen includes a K-LIME or LIME-SUP graph. A K-LIME graph is available by default when you interpret a model from the experiment page. When you create a new interpretation, you can instead choose to use LIME-SUP as the LIME method. Note that these graphs are essentially the same, but the K-LIME/LIME-SUP distinction provides insight into the LIME method that was used during model interpretation.

The K-LIME Technique

This plot is available for binary classification and regression models.

K-LIME is a variant of the LIME technique proposed by Ribeiro et al (2016). K-LIME generates global and local explanations that increase the transparency of the Driverless AI model, and allow model behavior to be validated and debugged by analyzing the provided plots, and comparing global and local explanations to one-another, to known standards, to domain knowledge, and to reasonable expectations.

K-LIME creates one global surrogate GLM on the entire training data and also creates numerous local surrogate GLMs on samples formed from k -means clusters in the training data. The features used for k -means are selected from the Random Forest surrogate model's variable importance. The number of features used for k -means is the minimum of the top 25% of variables from the Random Forest surrogate model's variable importance and the max number of variables that can be used for k -means, which is set by the user in the config.toml setting for `mli_max_number_cluster_vars`. (Note, if the number of features in the dataset are less than or equal to 6, then all features are used for k -means clustering.) The previous setting can be turned off to use all features for k -means by setting `use_all_columns_klime_kmeans` in the config.toml file to `true`. All penalized GLM surrogates are trained to model the predictions of the Driverless AI model. The number of clusters for local explanations is chosen by a grid search in which the R^2 between the Driverless AI model predictions and all of the local K-LIME

model predictions is maximized. The global and local linear model's intercepts, coefficients, R^2 values, accuracy, and predictions can all be used to debug and develop explanations for the Driverless AI model's behavior.

The parameters of the global K-LIME model give an indication of overall linear feature importance and the overall average direction in which an input variable influences the Driverless AI model predictions. The global model is also used to generate explanations for very small clusters ($N < 20$) where fitting a local linear model is inappropriate.

The in-cluster linear model parameters can be used to profile the local region, to give an average description of the important variables in the local region, and to understand the average direction in which an input variable affects the Driverless AI model predictions. For a point within a cluster, the sum of the local linear model intercept and the products of each coefficient with their respective input variable value are the K-LIME prediction. By disaggregating the K-LIME predictions into individual coefficient and input variable value products, the local linear impact of the variable can be determined. This product is sometimes referred to as a reason code and is used to create explanations for the Driverless AI model's behavior.

In the following example, reason codes are created by evaluating and disaggregating a local linear model.

Given the row of input data with its corresponding Driverless AI and K-LIME predictions:

debt_to_income_ratio	credit_score	savings_acct_balance	observed_default	H2OAI_predicted_default	K-LIME_predicted_default
30	600	1000	1	0.85	0.9

And the local linear model:

$$y_{\text{K-LIME}} = 0.1 + 0.01 * \text{debt_to_income_ratio} + 0.0005 * \text{credit_score} + 0.0002 * \text{savings_account_balance}$$

It can be seen that the local linear contributions for each variable are:

- *debt_to_income_ratio*: $0.01 * 30 = 0.3$
- *credit_score*: $0.0005 * 600 = 0.3$
- *savings_acct_balance*: $0.0002 * 1000 = 0.2$

Each local contribution is positive and thus contributes positively to the Driverless AI model's prediction of 0.85 for *H2OAI_predicted_default*. By taking into consideration the value of each contribution, reason codes for the Driverless AI decision can be derived. *debt_to_income_ratio* and *credit_score* would be the two largest negative reason codes, followed by *savings_acct_balance*.

The local linear model intercept and the products of each coefficient and corresponding value sum to the K-LIME prediction. Moreover it can be seen that these linear explanations are reasonably representative of the nonlinear model's behavior for this individual because the K-LIME predictions are within 5.5% of the Driverless AI model prediction. This information is encoded into English language rules which can be viewed by clicking the **Explanations** button.

Like all LIME explanations based on linear models, the local explanations are linear in nature and are offsets from the baseline prediction, or intercept, which represents the average of the penalized linear model residuals. Of course, linear approximations to complex non-linear response functions will not always create suitable explanations and users are urged to check the K-LIME plot, the local model R^2 , and the accuracy of the K-LIME prediction to understand the validity of the K-LIME local explanations. When K-LIME accuracy for a given point or set of points is quite low, this can be an indication of extremely nonlinear behavior or the presence of strong or high-degree interactions in this local region of the Driverless AI response function. In cases where K-LIME linear models are not fitting the Driverless AI model well, nonlinear LOCO feature importance values may be a better explanatory tool for local model behavior. As K-LIME local explanations rely on the creation of k -means clusters, extremely wide input data or strong correlation between input variables may also degrade the quality of K-LIME local explanations.

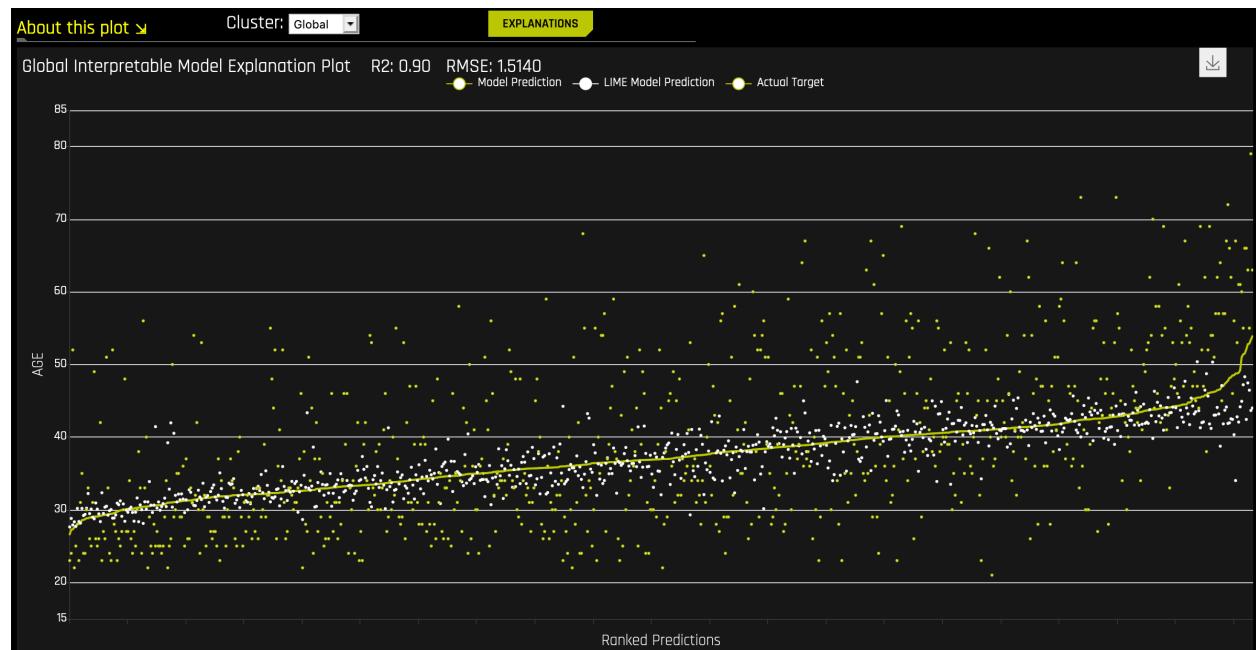
The LIME-SUP Technique

This plot is available for binary classification and regression models.

LIME-SUP explains local regions of the trained Driverless AI model in terms of the original variables. Local regions are defined by each leaf node path of the decision tree surrogate model instead of simulated, perturbed observation samples - as in the original LIME. For each local region, a local GLM model is trained on the original inputs and the predictions of the Driverless AI model. Then the parameters of this local GLM can be used to generate approximate, local explanations of the Driverless AI model.

The Global Interpretable Model Explanation Plot

This plot shows Driverless AI model predictions and LIME model predictions in sorted order by the Driverless AI model predictions. This graph is interactive. Hover over the **Model Prediction**, **LIME Model Prediction**, or **Actual Target** radio buttons to magnify the selected predictions. Or click those radio buttons to disable the view in the graph. You can also hover over any point in the graph to view LIME reason codes for that value. By default, this plot shows information for the global LIME model, but you can change the plot view to show local results from a specific cluster. The LIME plot also provides a visual indication of the linearity of the Driverless AI model and the trustworthiness of the LIME explanations. The closer the local linear model approximates the Driverless AI model predictions, the more linear the Driverless AI model and the more accurate the explanation generated by the LIME local linear models.



Decision Tree

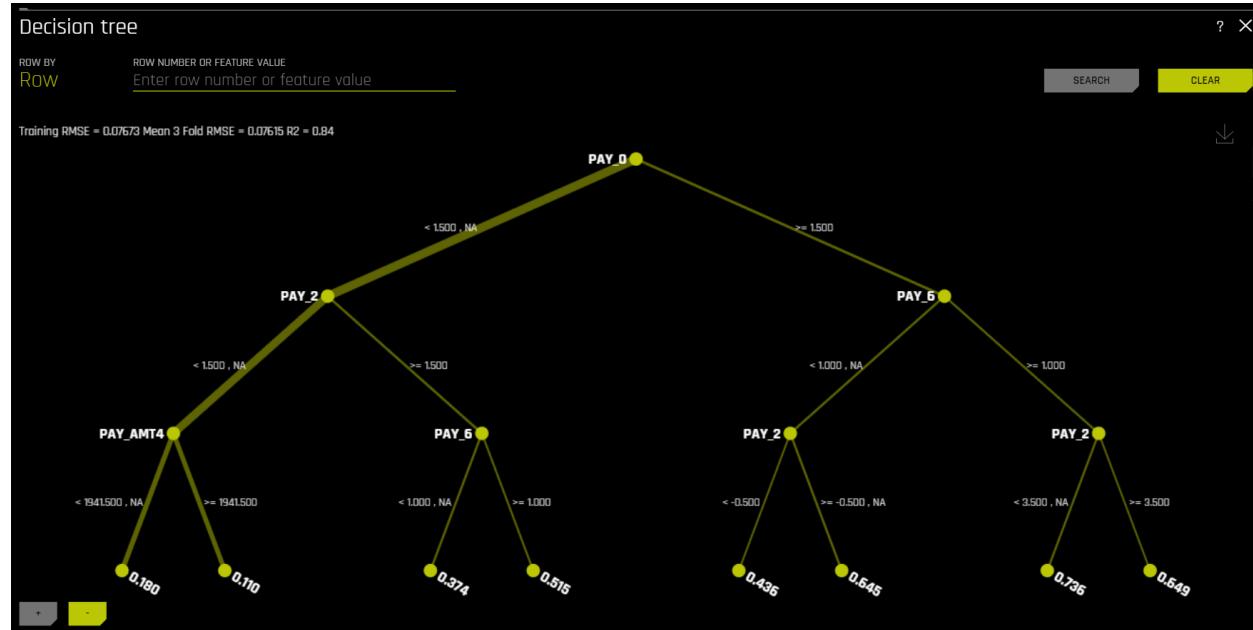
The Decision Tree Surrogate Model Technique

The decision tree surrogate model increases the transparency of the Driverless AI model by displaying an *approximate* flow-chart of the complex Driverless AI model's decision making process. It also displays the most important variables in the Driverless AI model and the most important interactions in the Driverless AI model. The decision tree surrogate model can be used for visualizing, validating, and debugging the Driverless AI model by comparing the displayed decision-process, important variables, and important interactions to known standards, domain knowledge, and reasonable expectations. It is known to date back at least to 1996 (Craven and Shavlik).

The Decision Tree Plot

This plot is available for binary and multinomial classification models as well as regression models.

In the Decision Tree plot, the highlighted row shows the path to the highest probability leaf node and indicates the globally important variables and interactions that influence the Driverless AI model prediction for that row.



For multinomial models, decision trees are created for each class. To view a decision tree for a specific class, click **Class** in the upper-left corner of the page and select the class you want to view a decision tree for.

Random Forest Feature Importance

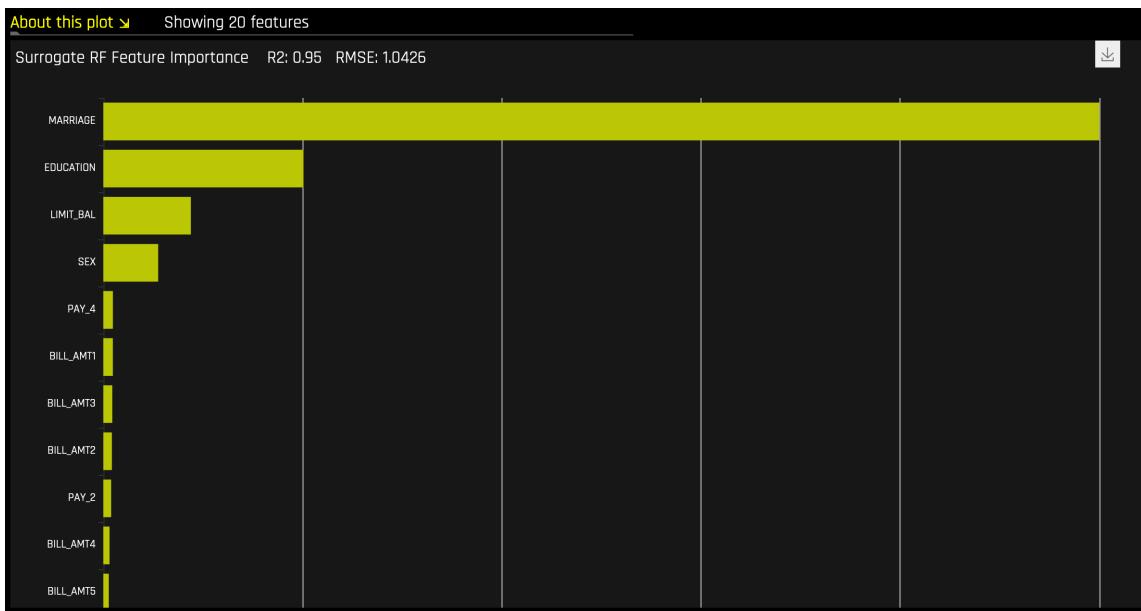
Global Feature Importance vs Local Feature Importance

Global feature importance (yellow) is a measure of the contribution of an input variable to the overall predictions of the Driverless AI model. Global feature importance is calculated by aggregating the improvement in splitting criterion caused by a single variable across all of the decision trees in the Driverless AI model.

Local feature importance (grey) is a measure of the contribution of an input variable to a single prediction of the Driverless AI model. Local feature importance is calculated by removing the contribution of a variable from every decision tree in the Driverless AI model and measuring the difference between the prediction with and without the variable.

Both global and local variable importance are scaled so that the largest contributor has a value of 1.

Note: Engineered features are used for MLI when a time series experiment is built. This is because munged time series features are more useful features for MLI than raw time series features, as raw time series features are not IID (Independent and Identically Distributed).



Random Forest Partial Dependence and Individual Conditional Expectation

A Partial Dependence and ICE plot is available for both Driverless AI and surrogate models. Refer to the previous pdp-ice section for more information about this plot.

Random Forest LOCO

This plot is available for binary and multinomial classification models as well as regression models.

Local feature importance describes how the combination of the learned model rules or parameters and an individual row's attributes affect a model's prediction for that row while taking nonlinearity and interactions into effect. Local feature importance values reported in this plot are based on a variant of the leave-one-covariate-out (LOCO) method (Lei et al, 2017).

The LOCO-variant method for binary and regression models calculates each local feature importance by re-scoring the trained Driverless AI model for each feature in the row of interest, while removing the contribution to the model prediction of splitting rules that contain that feature throughout the ensemble. The original prediction is then subtracted from this modified prediction to find the raw, signed importance for the feature. All local feature importance values for the row are then scaled between 0 and 1 for direct comparison with global feature importance values.

The LOCO-variant method for multinomial models differs slightly in that it calculates row-wise local feature importance values by re-scoring the trained supervised model and measuring the impact of setting each variable to missing. The sum of the absolute value of differences across classes is then calculated for each dropped or replaced column.

Given the row of input data with its corresponding Driverless AI and K-LIME predictions:

debt_to_income_ratio	credit_score	savings_acct_balance	observed_default	H2OAI_predicted_default	K-LIME_predicted_default
30	600	1000	1	0.85	0.9

Taking the Driverless AI model as $F(\mathbf{X})$, LOCO-variant feature importance values are calculated as follows.

First, the modified predictions are calculated:

$$F_{debt_to_income_ratio} = F(NA, 600, 1000) = 0.99$$

$$F_{credit_score} = F(30, NA, 1000) = 0.73$$

$$F_{savings_acct_balance} = F(30, 600, NA) = 0.82$$

Second, the original prediction is subtracted from each modified prediction to generate the unscaled local feature importance values:

$$\text{LOCO}_{debt_to_income_ratio} = F_{debt_to_income_ratio} - 0.85 = 0.99 - 0.85 = 0.14$$

$$\text{LOCO}_{credit_score} = F_{credit_score} - 0.85 = 0.73 - 0.85 = -0.12$$

$$\text{LOCO}_{savings_acct_balance} = F_{savings_acct_balance} - 0.85 = 0.82 - 0.85 = -0.03$$

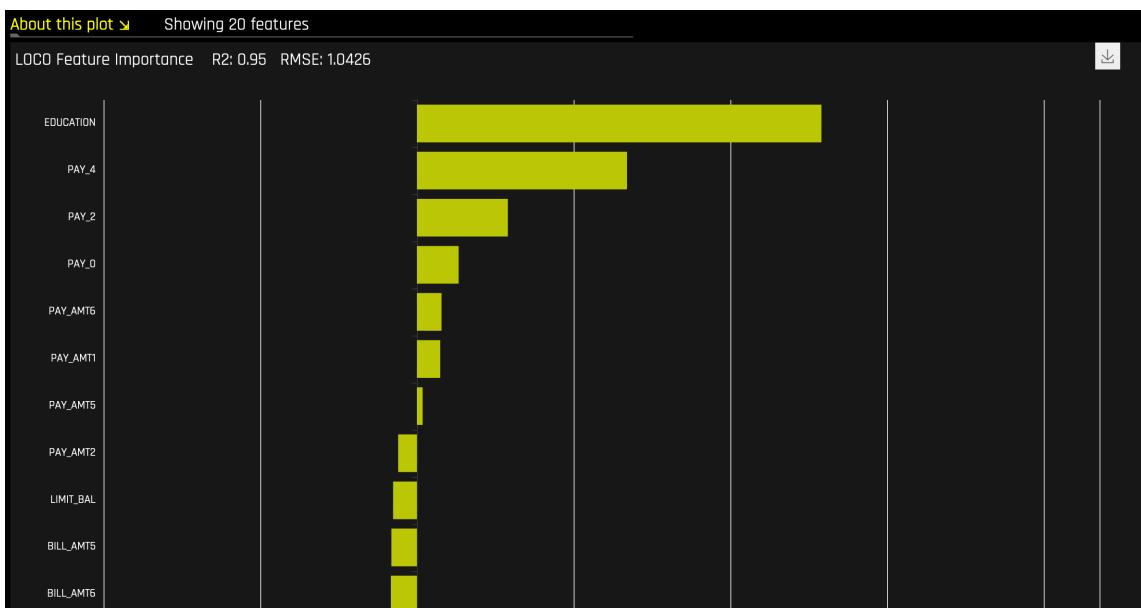
Finally LOCO values are scaled between 0 and 1 by dividing each value for the row by the maximum value for the row and taking the absolute magnitude of this quotient.

$$\text{Scaled}(\text{LOCO}_{debt_to_income_ratio}) = \text{Abs}(\text{LOCO}_{debt_to_income_ratio}/0.14) = 1$$

$$\text{Scaled}(\text{LOCO}_{credit_score}) = \text{Abs}(\text{LOCO}_{credit_score}/0.14) = 0.86$$

$$\text{Scaled}(\text{LOCO}_{savings_acct_balance}) = \text{Abs}(\text{LOCO}_{savings_acct_balance}/0.14) = 0.21$$

One drawback to these LOCO-variant feature importance values is, unlike K-LIME, it is difficult to generate a mathematical error rate to indicate when LOCO values may be questionable.



NLP Surrogate Models

These plots are available for natural language processing (NLP) models.

For NLP surrogate models, Driverless AI creates a TFIDF matrix by tokenizing all text features. The resulting frame is appended to numerical or categorical columns from the training dataset, and the original text columns are removed. This frame is then used for training surrogate models that have prediction columns consisting of tokens and the original numerical or categorical features.

Notes:

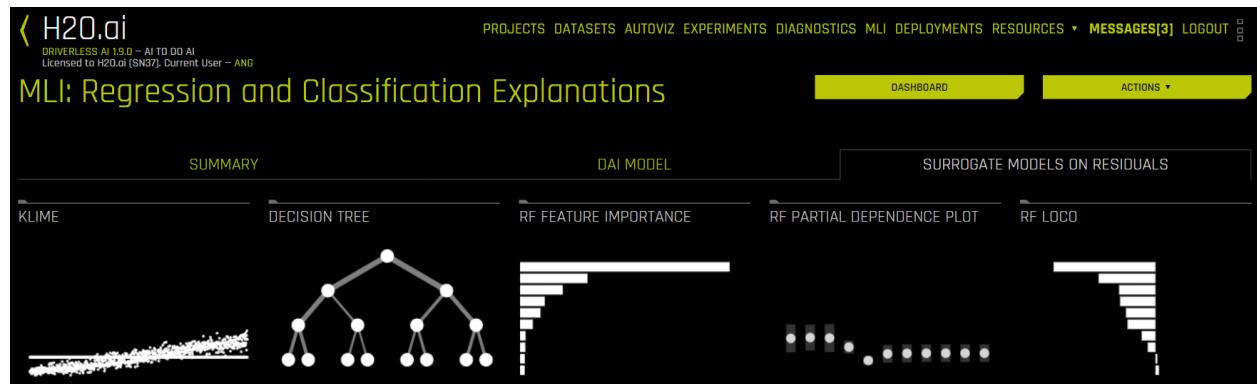
- MLI support for NLP is not available for multinomial experiments.

- Each row in the TFIDF matrix contains N columns, where N is the total number of tokens in the corpus with values that are appropriate for that row (0 if absent).
- Driverless AI does not currently generate a K-LIME scoring pipeline for MLI NLP problems.

Running Surrogate Models on Residuals

In Driverless AI, residuals (differences between observed and predicted values) can be used as targets in MLI surrogate models for the purpose of debugging models. The method used to calculate residuals varies depending on the type of problem. For classification problems, logloss residuals are calculated for a specified class. For regression problems, residuals are determined by calculating the square of the difference between targeted and predicted values.

To run MLI surrogate models on residuals, enable the **Debug Model Residuals** interpretation expert setting. For classification experiments, specify a class to use as an outcome of interest with the **Class for Debugging Classification Model Logloss Residuals** interpretation expert setting (not visible for regression problems). You can view the models by clicking the **Surrogate Models on Residuals** tab once the interpretation is complete.



28.3 Viewing Explanations

Note: Not all explanatory functionality is available for multinomial classification scenarios.

Driverless AI provides easy-to-read explanations for a completed model. You can view these by clicking the **Explanations** button on the **Model Interpretation > Dashboard** page for an interpreted model.

The UI allows you to view global, cluster-specific, and local reason codes. You can also export the explanations to CSV.

- **Global Reason Codes:** To view global reason codes, click **Cluster** and select **Global** from the list of options. With Global selected, click the **Explanations** button located in the upper-right corner.
- **Cluster Reason Codes:** To view reason codes for a specific cluster, click **Cluster** and select a specific cluster from the list of options. With a cluster selected, click the **Explanations** button.
- **Local Reason Codes by Row Number:** To view local reason codes for a specific row, select a point on the graph or type a value in the **Row Number or Feature Value** field. With a value selected, click the **Explanations** button.
- **Local Reason Codes by ID:** Identifier columns cannot be specified by the user - MLI makes this choice automatically by choosing columns whose values are unique (dataset row count equals the number of unique values in a column). To find a row by identifier column, choose **ID** from the drop-down menu (if it meets the logic of being an identifier column), and then specify a value. With a value selected, click the **Explanations** button.

28.4 General Considerations

28.4.1 Machine Learning and Approximate Explanations

For years, common sense has deemed the complex, intricate formulas created by training machine learning algorithms to be uninterpretable. While great advances have been made in recent years to make these often nonlinear, non-monotonic, and non-continuous machine-learned response functions more understandable (Hall et al, 2017), it is likely that such functions will never be as directly or universally interpretable as more traditional linear models.

Why consider machine learning approaches for inferential purposes? In general, linear models focus on understanding and predicting average behavior, whereas machine-learned response functions can often make accurate, but more difficult to explain, predictions for subtler aspects of modeled phenomenon. In a sense, linear models create very exact interpretations for approximate models. The approach here seeks to make approximate explanations for very exact models. It is quite possible that an approximate explanation of an exact model may have as much, or more, value and meaning than the exact interpretations of an approximate model. Moreover, the use of machine learning techniques for inferential or predictive purposes does not preclude using linear models for interpretation (Ribeiro et al, 2016).

28.4.2 The Multiplicity of Good Models in Machine Learning

It is well understood that for the same set of input variables and prediction targets, complex machine learning algorithms can produce multiple accurate models with very similar, but not exactly the same, internal architectures (Breiman, 2001). This alone is an obstacle to interpretation, but when using these types of algorithms as interpretation tools or with interpretation tools it is important to remember that details of explanations will change across multiple accurate models.

28.4.3 Expectations for Consistency Between Explanatory Techniques

- The decision tree surrogate is a global, nonlinear description of the Driverless AI model behavior. Variables that appear in the tree should have a direct relationship with variables that appear in the global feature importance plot. For certain, more linear Driverless AI models, variables that appear in the decision tree surrogate model may also have large coefficients in the global K-LIME model.
- K-LIME explanations are linear, do not consider interactions, and represent offsets from the local linear model intercept. LOCO importance values are nonlinear, do consider interactions, and do not explicitly consider a linear intercept or offset. LIME explanations and LOCO importance values are not expected to have a direct relationship but can align roughly as both are measures of a variable's local impact on a model's predictions, especially in more linear regions of the Driverless AI model's learned response function.
- ICE is a type of nonlinear sensitivity analysis which has a complex relationship to LOCO feature importance values. Comparing ICE to LOCO can only be done at the value of the selected variable that actually appears in the selected row of the training data. When comparing ICE to LOCO the total value of the prediction for the row, the value of the variable in the selected row, and the distance of the ICE value from the average prediction for the selected variable at the value in the selected row must all be considered.
- ICE curves that are outside the standard deviation of partial dependence would be expected to fall into less populated decision paths of the decision tree surrogate; ICE curves that lie within the standard deviation of partial dependence would be expected to belong to more common decision paths.
- Partial dependence takes into consideration nonlinear, but average, behavior of the complex Driverless AI model without considering interactions. Variables with consistently high partial dependence or partial dependence that swings widely across an input variable's domain will likely also have high global importance values. Strong interactions between input variables can cause ICE values to diverge from partial dependence values.

MLI FOR TIME-SERIES EXPERIMENTS

This section describes how to run MLI for time-series experiments. Refer to [MLI for Regular \(Non-Time-Series\) Experiments](#) for MLI information with regular experiments.

There are two methods you can use for interpreting time-series models:

- Using the **Interpret this Model** button on a completed experiment page to interpret a Driverless AI model on original and transformed features. (See below.)
- Using the **MLI** link in the upper right corner of the UI to interpret either a Driverless AI model or an external model. This process is described in the [Model Interpretation on Driverless AI Models](#) and [Model Interpretation on External Models](#) sections.

Limitations

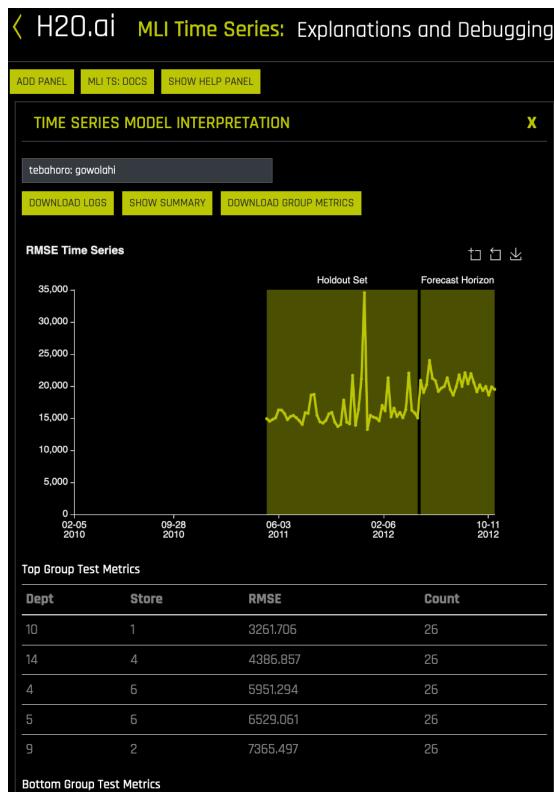
- This release deprecates experiments run in 1.7.0 and earlier. MLI will not be available for experiments from versions <= 1.7.0.
- MLI is not available for NLP experiments or for multiclass Time Series.
- When the test set contains actuals, you will see the time series metric plot and the group metrics table. If there are no actuals, MLI will run, but you will see only the prediction value time series and a Shapley table.
- MLI does not require an Internet connection to run on current models.

29.1 Multi-Group Time Series MLI

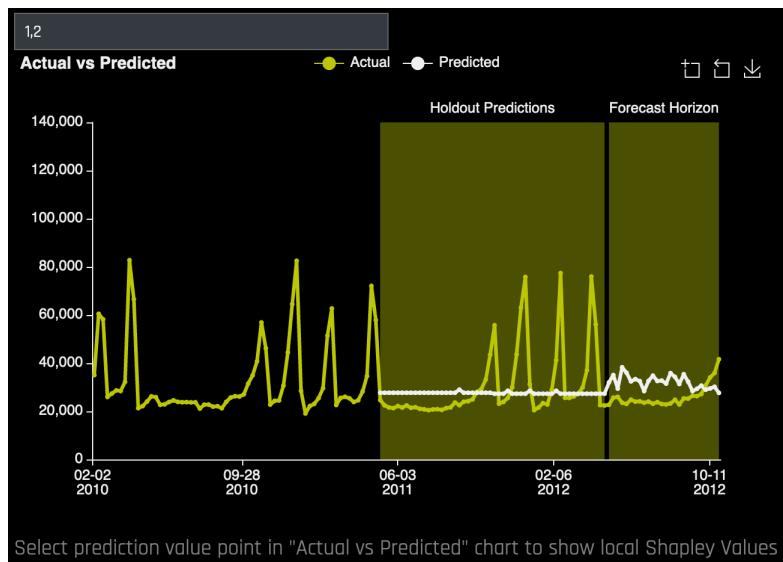
This section describes how to run MLI on time series data for multiple groups.

1. Click the **Interpret this Model** button on a completed time series experiment to launch Model Interpretation for that experiment. This page includes the following:
 - A Help panel describing how to read and use this page. Click the **Hide Help Button** to hide this text.
 - If a test set is provided and the test set includes actuals, then a panel will display showing a time series plot and the top and bottom group matrix tables based on the scorer that was used in the experiment. The metric plot will show the metric of interest per time point for holdout predictions and the test set. Likewise, the actual vs. predicted plot will show actuals vs. predicted values per time point for the holdout set and the test set. Note that this panel can be resized if necessary.
 - If a test set is not provided, then internal validation predictions will be used. The metric plot will only show the metric of interest per time point for holdout predictions. Likewise, the actual vs. predicted plot will only show actuals vs. predicted values per time point for the holdout set.
 - A **Download Logs** button for retrieving logs that were generated when this interpretation was built.

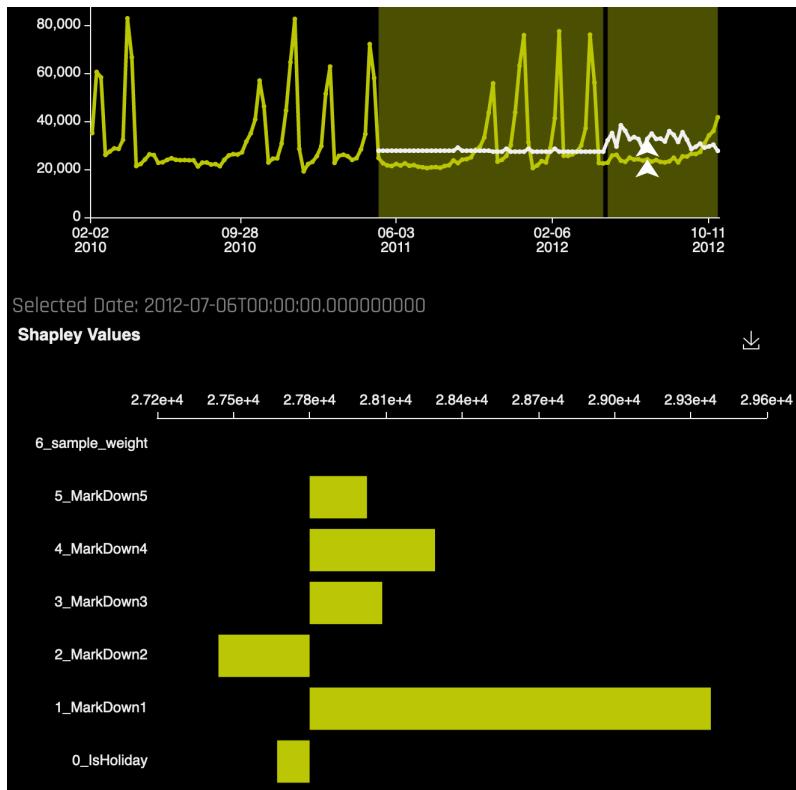
- A **Download Group Metrics** button for retrieving the averages of each group's scorer, as well as each group's sample size.
- A **Show Summary** button that provide details about the experiment settings that were used.
- A **Group Search** entry field (scroll to bottom) for selecting the groups to view.
- Use the zoom feature to magnify any portion of a graph by clicking the **Enable Zoom** icon near the top-right corner of a graph. While this icon is selected, click and drag to draw a box over the portion of the graph you want to magnify. Click the **Disable Zoom** icon to return to the default view.



2. Scroll to the bottom of the panel and select a grouping in the **Group Search** field to view a graph of Actual vs. Predicted values for the group. The outputted graph can be downloaded to your local machine.



- Click on a prediction point in the plot (white line) to view Shapley values for that prediction point. The Shapley values plot can also be downloaded to your local machine.

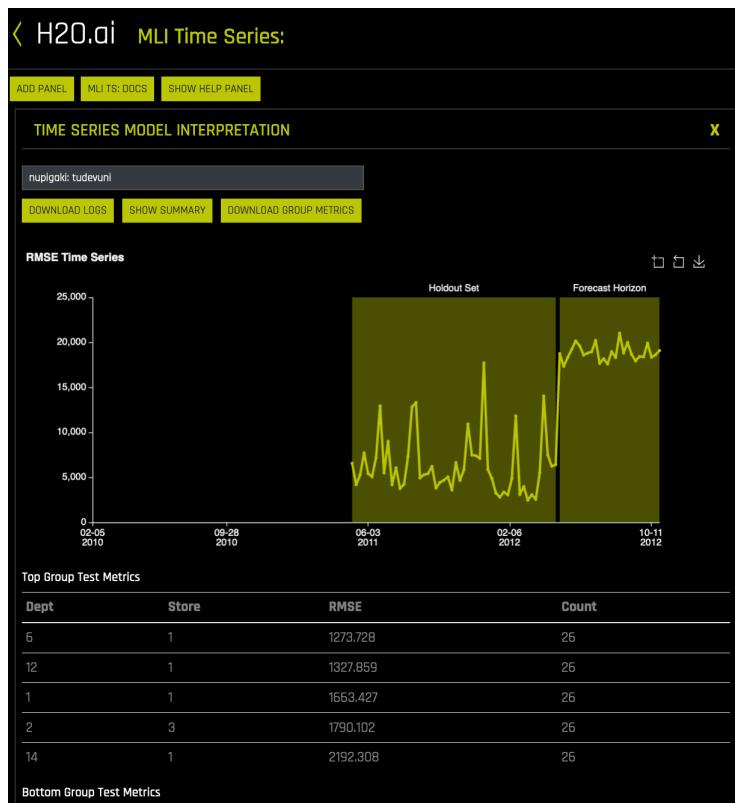


- Click **Add Panel** to add a new MLI Time Series panel. This allows you to compare different groups in the same model and also provides the flexibility to do a “side-by-side” comparison between different models.

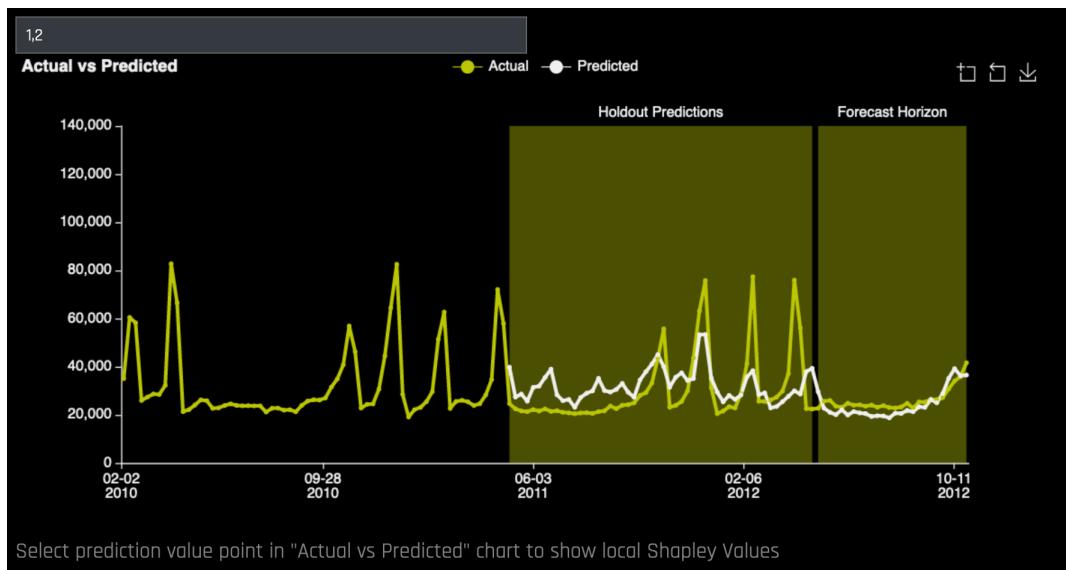
29.2 Single Time Series MLI

Time Series MLI can also be run when only one group is available.

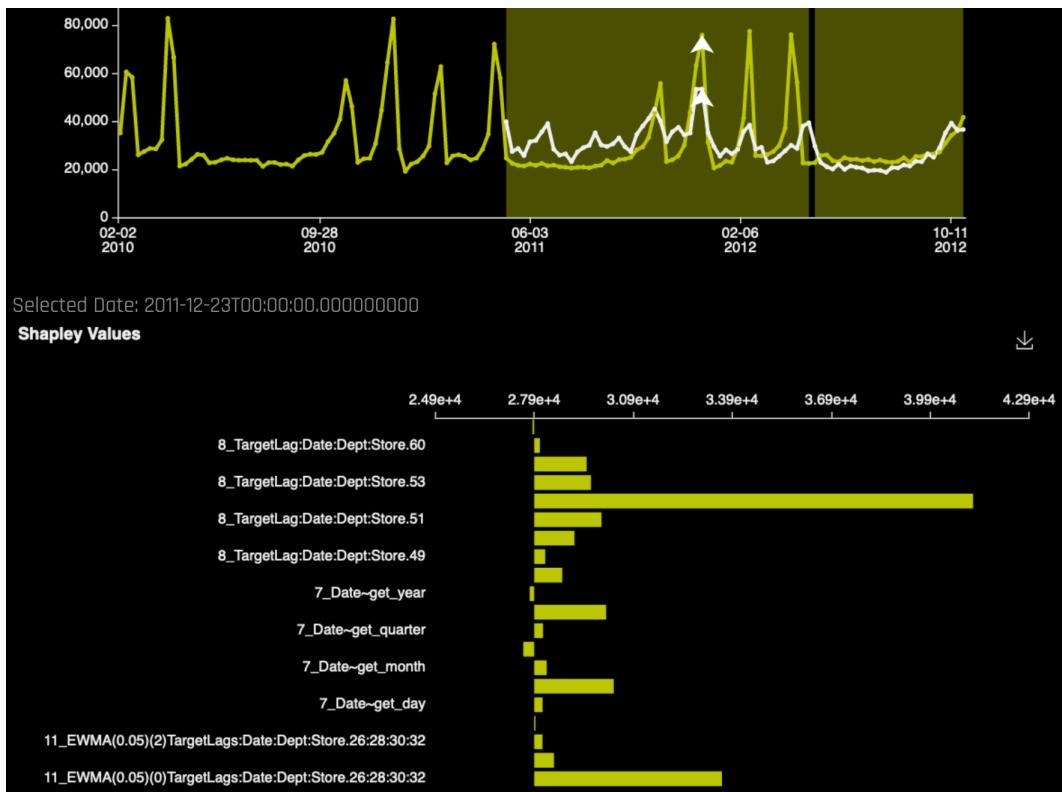
1. Click the **Interpret this Model** button on a completed time series experiment to launch Model Interpretation for that experiment. This page includes the following:
 - A Help panel describing how to read and use this page. Click the **Hide Help Button** to hide this text.
 - If a test set is provided and the test set includes actuals, then a panel will display showing a time series plot and the top and bottom group matrix tables based on the scorer that was used in the experiment. The metric plot will show the metric of interest per time point for holdout predictions and the test set. Likewise, the actual vs. predicted plot will show actuals vs. predicted values per time point for the holdout set and the test set. Note that this panel can be resized if necessary.
 - If a test set is not provided, then internal validation predictions will be used. The metric plot will only show the metric of interest per time point for holdout predictions. Likewise, the actual vs. predicted plot will only show actuals vs. predicted values per time point for the holdout set.
 - A **Download Logs** button for retrieving logs that were generated when this interpretation was built.
 - A **Download Group Metrics** button for retrieving the average of the group's scorer, as well as the group's sample size.
 - A **Show Summary** button that provides details about the experiment settings that were used.
 - A **Group Search** entry field for selecting the group to view. Note that for Single Time Series MLI, there will only be one option in this field.
 - Use the zoom feature to magnify any portion of a graph by clicking the leftmost square icon near the top-right corner of a graph. While this icon is selected, click and drag to draw a box over the portion of the graph you want to magnify. To return to the default view, click the square-shaped arrow icon to the right of the zoom icon.



2. Scroll to the bottom of the panel and select an option in the **Group Search** field to view a graph of Actual vs. Predicted values for the group. (Note that for Single Time Series MLI, there will only be one option in this field.) The outputted graph can be downloaded to your local machine.



3. Click on a prediction point in the plot (white line) to view Shapley values for that prediction point. The Shapley values plot can also be downloaded to your local machine.



4. Click **Add Panel** to add a new MLI Time Series panel. This allows you to do a “side-by-side” comparison between different models.

SCORE ON ANOTHER DATASET

After you generate a model, you can use that model to make predictions on another dataset.

1. Click the **Experiments** link in the top menu and select the experiment that you want to use.
2. On the Experiment page, click the **Score on Another Dataset** button.
3. Locate the new dataset (test set) that you want to score on. Note that this new dataset must include the same columns as the dataset used in selected experiment.
4. Select the columns from the test set to include in the predictions frame.
5. Click **Done** to start the scoring process.
6. Click the **Download Predictions** button after scoring is complete.

Note: This feature runs batch scoring on a new dataset. You may notice slow speeds if you attempt to perform single-row scoring.

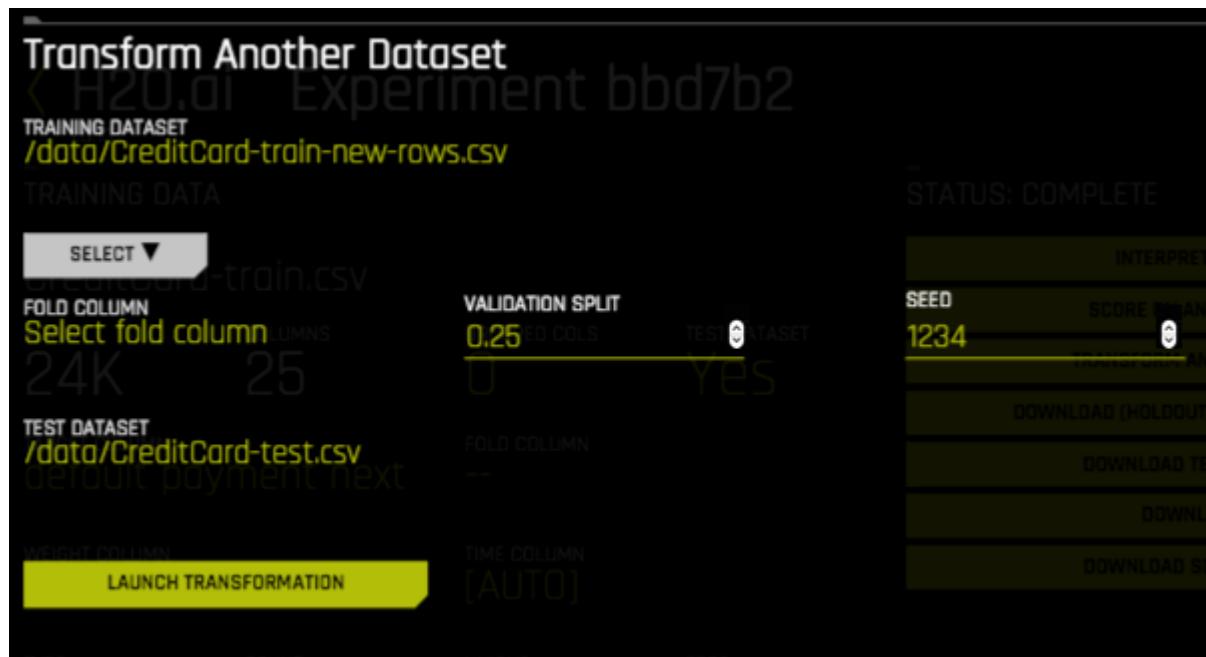
TRANSFORM ANOTHER DATASET

When a training dataset is used in an experiment, Driverless AI transforms the data into an improved, feature engineered dataset. (Refer to [Driverless AI Transformations](#) for more information about the transformations that are provided in Driverless AI.) But what happens when new rows are added to your dataset? In this case, you can specify to transform the new dataset after adding it to Driverless AI, and the same transformations that Driverless AI applied to the original dataset will be applied to these new rows.

Follow these steps to transform another dataset. Note that this assumes the new dataset has been added to Driverless AI already.

Note: **Transform Another Dataset** is not available for Time Series experiments.

1. On the completed experiment page for the original dataset, click the **Transform Another Dataset** button.
2. Select the new training dataset that you want to transform. Note that this must have the same number columns as the original dataset.
3. In the **Select** drop down, specify a validation dataset to use with this dataset, or specify to split the training data. If you specify to split the data, then you also specify the split value (defaults to 25%) and the seed (defaults to 1234). **Note:** To ensure the transformed dataset respects the row order, choose a validation dataset instead of splitting the training data. Splitting the training data will result in a shuffling of the row order.
4. Optionally specify a test dataset. If specified, then the output also include the final test dataset for final scoring.
5. Click **Launch Transformation**.



The following datasets will be available for download upon successful completion:

- Training dataset (not for cross validation)
- Validation dataset for parameter tuning
- Test dataset for final scoring. This option is available if a test dataset was used.



CHAPTER
THIRTYTWO

SCORING PIPELINES OVERVIEW

Driverless AI provides several Scoring Pipelines for experiments and/or interpreted models.

- A standalone Python Scoring Pipeline is available for experiments and interpreted models.
- A low-latency, standalone MOJO Scoring Pipeline is available for experiments, with both Java and C++ backends.

The Python Scoring Pipeline is implemented as a Python whl file. While this allows for a single process scoring engine, the scoring service is generally implemented as a client/server architecture and supports interfaces for TCP and HTTP.

The MOJO Scoring Pipeline provides a standalone scoring pipeline that converts experiments to MOJOS, which can be scored in real time. The MOJO Scoring Pipeline is available as either a *Java runtime* or a *C++ runtime*. For the C++ runtime, both Python and R wrappers are provided.

Examples are included with each scoring package.

Note: These sections describe scoring pipelines and not deployments of scoring pipelines. For information on how to deploy a MOJO scoring pipeline, refer to the *Deploying the MOJO Pipeline* section.

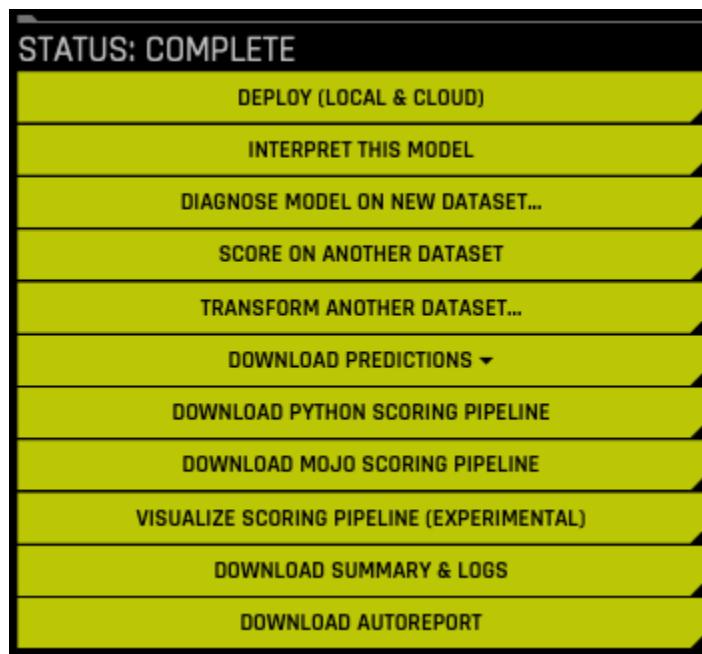
CHAPTER
THIRTYTHREE

VISUALIZING THE SCORING PIPELINE

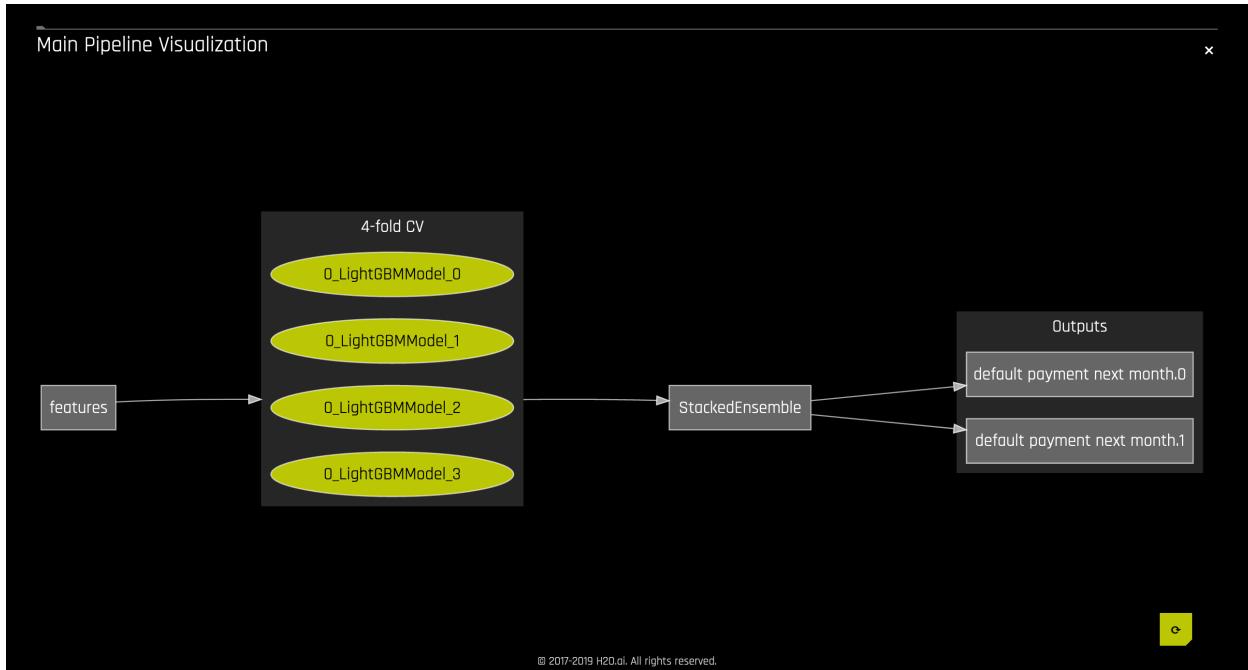
A visualization of the scoring pipeline is available for each completed experiment.

Notes:

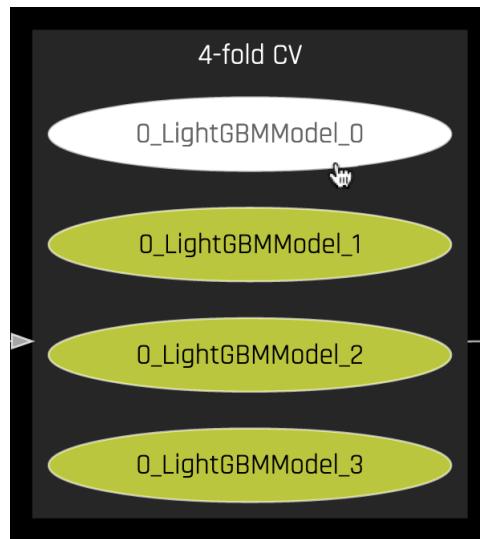
- This pipeline is best viewed in the latest version of Chrome.
- A .png image of this pipeline is available in the [Autoreport](#) and in the mojo.zip file ONLY with the Driverless AI Docker image. For tar, deb, and rpm installs, you must install [Graphviz](#) manually in order for the visualization pipeline to be included in the Autoreport and mojo.zip.

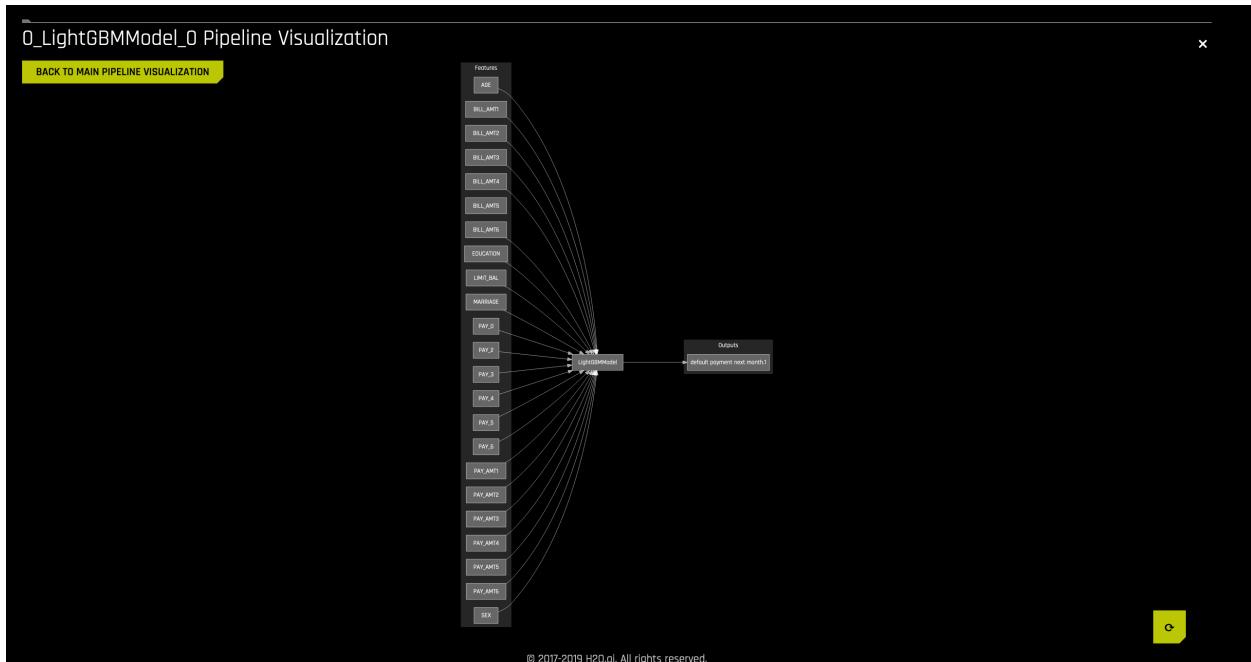


Click the **Visualize Scoring Pipeline (Experimental)** button on the completed experiment page to view the visualization.

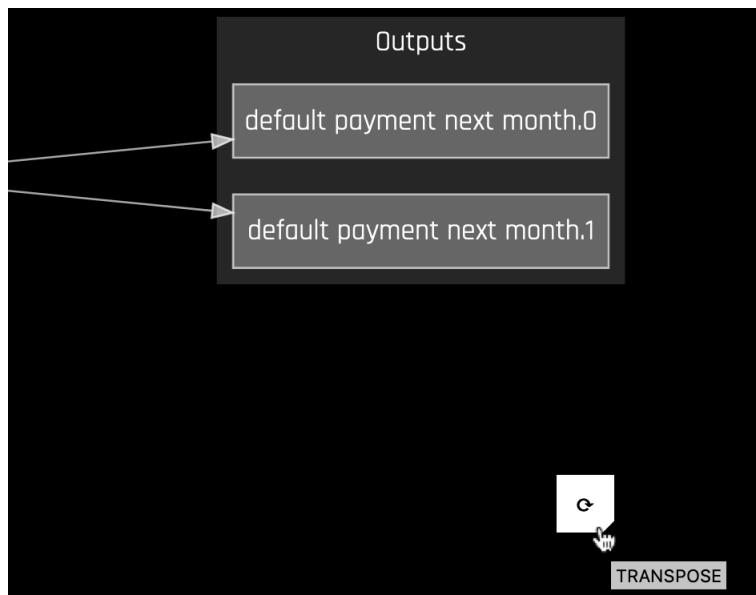


To view a visual representation of a specific model, click on the oval that corresponds with that model.





To change the orientation of the visualization, click the **Transpose** button in the bottom right corner of the screen.



CHAPTER
THIRTYFOUR

WHICH PIPELINE SHOULD I USE?

Driverless AI provides a Python Scoring Pipeline, an MLI Standalone Scoring Pipeline, and a MOJO Scoring Pipeline. Consider the following when determining the scoring pipeline that you want to use.

- For all pipelines, the higher the accuracy, the slower the scoring.
- The Python Scoring Pipeline is slower but easier to use than the MOJO scoring pipeline.
- When running the Python Scoring Pipeline:
 - HTTP is easy and is supported by virtually any language. HTTP supports RESTful calls via curl, wget, or supported packages in various scripting languages.
 - TCP is a bit more complex, though faster. TCP also requires Thrift, which currently does not handle NAs.
- The MOJO Scoring Pipeline is flexible and is faster than the Python Scoring Pipeline, but it requires a bit more coding. The MOJO Scoring Pipeline is available as either a *Java runtime* or a *C++ runtime*.
- The MLI Standalone Python Scoring Pipeline can be used to score interpreted models but only supports k-LIME reason codes.
 - For obtaining k-LIME reason codes from an MLI experiment, use the MLI Standalone Python Scoring Pipeline. k-LIME reason codes are available for all models.
 - For obtaining Shapley reason codes from an MLI experiment, use the DAI Standalone Python Scoring Pipeline. Shapley is only available for XGBoost and LightGBM models. Note that obtaining Shapley reason codes through the Python Scoring Pipeline can be time consuming.

DRIVERLESS AI STANDALONE PYTHON SCORING PIPELINE

As indicated earlier, a scoring pipeline is available after a successfully completed experiment. This package contains an exported model and Python 3.6 source code examples for productionizing models built using H2O Driverless AI.

The files in this package allow you to transform and score on new data in a couple of different ways:

- From Python 3.6, you can import a scoring module, and then use the module to transform and score on new data.
- From other languages and platforms, you can use the TCP/HTTP scoring service bundled with this package to call into the scoring pipeline module through remote procedure calls (RPC).

Note about custom recipes and the Python Scoring Pipeline: By default, if a custom recipe was has been uploaded into Driverless AI but then not used in the experiment, the Python Scoring Pipeline still contains the H2O recipe server. If this pipeline is then deployed in a container, the H2O recipe server causes the size of the pipeline to be much larger. In addition, Java has to be installed in the container, which further increases the runtime storage and memory requirements. A workaround is to set the following environment variable before running the Python Scoring Pipeline:

```
export dai_enable_custom_recipes=0
```

35.1 Python Scoring Pipeline Files

The **scoring-pipeline** folder includes the following notable files:

- **example.py**: An example Python script demonstrating how to import and score new records.
- **run_example.sh**: Runs example.py (also sets up a virtualenv with prerequisite libraries).
- **tcp_server.py**: A standalone TCP server for hosting scoring services.
- **http_server.py**: A standalone HTTP server for hosting scoring services.
- **run_tcp_server.sh**: Runs TCP scoring service (runs tcp_server.py).
- **run_http_server.sh**: Runs HTTP scoring service (runs http_server.py).
- **example_client.py**: An example Python script demonstrating how to communicate with the scoring server.
- **run_tcp_client.sh**: Demonstrates how to communicate with the scoring service via TCP (runs example_client.py).
- **run_http_client.sh**: Demonstrates how to communicate with the scoring service via HTTP (using curl).

35.2 Quick Start

There are two methods for starting the Python Scoring Pipeline.

35.2.1 Quick Start - Recommended Method

This is the recommended method for running the Python Scoring Pipeline. Use this method if:

- You have an air gapped environment with no access to the Internet.
- You are running Power.
- You want an easy quick start approach.

Prerequisites

- A valid Driverless AI license key.
- A completed Driverless AI experiment.
- Downloaded Python Scoring Pipeline.

Running the Python Scoring Pipeline - Recommended

1. Download the TAR SH version of Driverless AI from <https://www.h2o.ai/download/> (for either Linux or IBM Power).
2. Use bash to execute the download. This creates a new **dai-<dai_version>** folder, where **<dai_version>** represents your version of Driverless AI, for example, **1.7.1-linux-x86_64**.)
3. Change directories into the new Driverless AI folder. (Replace **<dai_version>** below with your version that was created in Step 2.)

```
cd dai-<dai_version>
```

4. Run the following to change permissions:

```
chmod -R a+w python
```

5. Run the following to install the Python Scoring Pipeline for your completed Driverless AI experiment:

```
./dai-env.sh pip install /path/to/your/scoring_experiment.whl
```

6. Run the following command to run the included scoring pipeline example:

```
DRIVERLESS_AI_LICENSE_KEY="pastekeyhere" SCORING_PIPELINE_INSTALL_DEPENDENCIES=0 ./dai-env.sh /path/to/your/run_example.sh
```

35.2.2 Quick Start - Alternative Method

This section describes an alternative method for running the Python Scoring Pipeline. This version requires Internet access. It is also not supported on Power machines.

Prerequisites

- The scoring module and scoring service are supported only on Linux with Python 3.6 and OpenBLAS.
- The scoring module and scoring service download additional packages at install time and require Internet access. Depending on your network environment, you might need to set up internet access via a proxy.
- Valid Driverless AI license. Driverless AI requires a license to be specified in order to run the Python Scoring Pipeline.
- Apache Thrift (to run the scoring service in TCP mode)
- Linux environment
- Python 3.6
- libopenblas-dev (required for H2O4GPU)
- OpenCL

Examples of how to install these prerequisites are below.

Installing Python 3.6

Installing Python 3.6 and OpenBLAS on Ubuntu 16.10+

```
sudo apt install python3.6 python3.6-dev python3-pip python3-dev \
python-virtualenv python3-virtualenv libopenblas-dev
```

Installing Python 3.6 and OpenBLAS on Ubuntu 16.04

```
sudo add-apt-repository ppa:deadsnakes/ppa
sudo apt-get update
sudo apt-get install python3.6 python3.6-dev python3-pip python3-dev \
python-virtualenv python3-virtualenv libopenblas-dev
```

Installing Conda 3.6:

You can install Conda using either Anaconda or Miniconda. Refer to the links below for more information:

- Anaconda - <https://docs.anaconda.com/anaconda/install.html>
- Miniconda - <https://docs.conda.io/en/latest/miniconda.html>

Installing OpenCL

Install OpenCL on RHEL

```
yum -y clean all
yum -y makecache
yum -y update
wget http://dl.fedoraproject.org/pub/epel/7/x86_64/Packages/c/clinfo-2.1.17.02.09-1.el7.x86_64.rpm
wget http://dl.fedoraproject.org/pub/epel/7/x86_64/Packages/o/ocl-icd-2.2.12-1.el7.x86_64.rpm
rpm -if clinfo-2.1.17.02.09-1.el7.x86_64.rpm
rpm -if ocl-icd-2.2.12-1.el7.x86_64.rpm
clinfo

mkdir -p /etc/OpenCL/vendors && \
echo "libnvidia-opencl.so.1" > /etc/OpenCL/vendors/nvidia.icd
```

Install OpenCL on Ubuntu

```
sudo apt install ocl-icd-libopencl1
mkdir -p /etc/OpenCL/vendors && \
echo "libnvidia-opencl.so.1" > /etc/OpenCL/vendors/nvidia.icd
```

License Specification

Driverless AI requires a license to be specified in order to run the Python Scoring Pipeline. The license can be specified via an environment variable in Python:

```
# Set DRIVERLESS_AI_LICENSE_FILE, the path to the Driverless AI license file  
%env DRIVERLESS_AI_LICENSE_FILE="/home/ubuntu/license/license.sig"  
  
# Set DRIVERLESS_AI_LICENSE_KEY, the Driverless AI license key (Base64 encoded string)  
%env DRIVERLESS_AI_LICENSE_KEY="oLqLZXMI0y..."
```

The examples that follow use DRIVERLESS_AI_LICENSE_FILE. Using DRIVERLESS_AI_LICENSE_KEY would be similar.

Installing the Thrift Compiler

Thrift is required to run the scoring service in TCP mode, but it is not required to run the scoring module. The following steps are available on the Thrift documentation site at: <https://thrift.apache.org/docs/BuildingFromSource>.

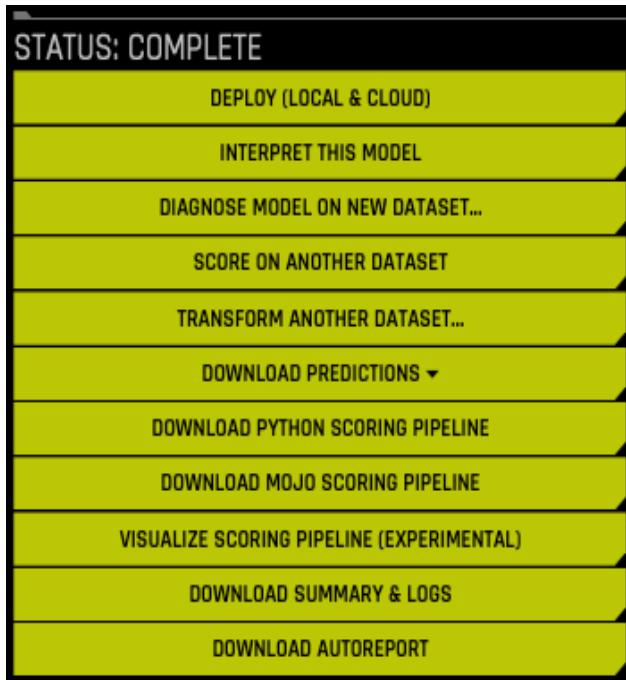
```
sudo apt-get install automake bison flex g++ git libevent-dev \  
libssl-dev libtool make pkg-config libboost-all-dev ant  
wget https://github.com/apache/thrift/archive/0.10.0.tar.gz  
tar -xvf 0.10.0.tar.gz  
cd thrift-0.10.0  
.bootstrap.sh  
.configure  
make  
sudo make install
```

Run the following to refresh the runtime shared after installing Thrift:

```
sudo ldconfig /usr/local/lib
```

Running the Python Scoring Pipeline - Alternative Method

1. On the completed Experiment page, click on the **Download Python Scoring Pipeline** button to download the **scorer.zip** file for this experiment onto your local machine.



2. Unzip the scoring pipeline.

After the pipeline is downloaded and unzipped, you will be able to run the scoring module and the scoring service.

Score from a Python Program

If you intend to score from a Python program, run the scoring module example. (Requires Linux and Python 3.6.)

```
export DRIVERLESS_AI_LICENSE_FILE="/path/to/license.sig"
bash run_example.sh
```

Score Using a Web Service

If you intend to score using a web service, run the HTTP scoring server example. (Requires Linux x86_64 and Python 3.6.)

```
export DRIVERLESS_AI_LICENSE_FILE="/path/to/license.sig"
bash run_http_server.sh
bash run_http_client.sh
```

Score Using a Thrift Service

If you intend to score using a Thrift service, run the TCP scoring server example. (Requires Linux x86_64, Python 3.6 and Thrift.)

```
export DRIVERLESS_AI_LICENSE_FILE="/path/to/license.sig"
bash run_tcp_server.sh
bash run_tcp_client.sh
```

Note: By default, the `run_*.sh` scripts mentioned above create a virtual environment using `virtualenv` and `pip`, within which the Python code is executed. The scripts can also leverage Conda (Anaconda/Miniconda) to create Conda virtual environment and install required package dependencies. The package manager to use is provided as an argument to the script.

```
# to use conda package manager
export DRIVERLESS_AI_LICENSE_FILE="/path/to/license.sig"
bash run_example.sh --pm conda

# to use pip package manager
export DRIVERLESS_AI_LICENSE_FILE="/path/to/license.sig"
bash run_example.sh --pm pip
```

If you experience errors while running any of the above scripts, please check to make sure your system has a properly installed and configured Python 3.6 installation. Refer to the [Troubleshooting Python Environment Issues](#) section that follows to see how to set up and test the scoring module using a cleanroom Ubuntu 16.04 virtual machine.

35.3 The Python Scoring Module

The scoring module is a Python module bundled into a standalone wheel file (name `scoring_*.whl`). All the prerequisites for the scoring module to work correctly are listed in the `requirements.txt` file. To use the scoring module, all you have to do is create a Python `virtualenv`, install the prerequisites, and then import and use the scoring module as follows:

```
# See 'example.py' for complete example.
from scoring_487931_20170921174120_b4066 import Scorer
scorer = Scorer()           # Create instance.
score = scorer.score([    # Call score()
    7.416,                 # sepal_len
    3.562,                 # sepal_wid
    1.049,                 # petal_len
    2.388,                 # petal_wid
])
```

The `scorer` instance provides the following methods (and more):

- `score(list)`: Score one row (list of values).
- `score_batch(df)`: Score a Pandas dataframe.

- `fit_transform_batch(df)`: Transform a Pandas dataframe.
- `get_target_labels()`: Get target column labels (for classification problems).

The process of importing and using the scoring module is demonstrated by the bash script `run_example.sh`, which effectively performs the following steps:

```
# See 'run_example.sh' for complete example.
virtualenv -p python3.6 env
source env/bin/activate
pip install -r requirements.txt
export DRIVERLESS_AI_LICENSE_FILE="/path/to/license.sig"
python example.py
```

35.4 The Scoring Service

The scoring service hosts the scoring module as an HTTP or TCP service. Doing this exposes all the functions of the scoring module through remote procedure calls (RPC). In effect, this mechanism allows you to invoke scoring functions from languages other than Python on the same computer or from another computer on a shared network or on the Internet.

The scoring service can be started in two ways:

- In TCP mode, the scoring service provides high-performance RPC calls via Apache Thrift (<https://thrift.apache.org/>) using a binary wire protocol.
- In HTTP mode, the scoring service provides JSON-RPC 2.0 calls served by Tornado (<http://www.tornadoweb.org>).

Scoring operations can be performed on individual rows (row-by-row) or in batch mode (multiple rows at a time).

35.4.1 Scoring Service - TCP Mode (Thrift)

The TCP mode allows you to use the scoring service from any language supported by Thrift, including C, C++, C#, Cocoa, D, Dart, Delphi, Go, Haxe, Java, Node.js, Lua, perl, PHP, Python, Ruby and Smalltalk.

To start the scoring service in TCP mode, you will need to generate the Thrift bindings once, then run the server:

```
# See 'run_tcp_server.sh' for complete example.
thrift --gen py scoring.thrift
python tcp_server.py --port=9090
```

Note that the Thrift compiler is only required at build-time. It is not a run time dependency, i.e. once the scoring services are built and tested, you do not need to repeat this installation process on the machines where the scoring services are intended to be deployed.

To call the scoring service, simply generate the Thrift bindings for your language of choice, then make RPC calls via TCP sockets using Thrift's buffered transport in conjunction with its binary protocol.

```
# See 'run_tcp_client.sh' for complete example.
thrift --gen py scoring.thrift

# See 'example_client.py' for complete example.
socket = TSocket.TSocket('localhost', 9090)
transport = TTransport.TBufferedTransport(socket)
protocol = TBinaryProtocol.TBinaryProtocol(transport)
client = ScoringService.Client(protocol)
transport.open()
row = Row()
row.sepalLen = 7.416 # sepal_len
row.sepalWid = 3.562 # sepal_wid
row.petalLen = 1.049 # petal_len
row.petalWid = 2.388 # petal_wid
scores = client.score(row)
transport.close()
```

You can reproduce the exact same result from other languages, e.g. Java:

```

thrift --gen java scoring.thrift

// Dependencies:
// commons-codec-1.9.jar
// commons-logging-1.2.jar
// httpclient-4.1.1.jar
// httpcore-4.4.1.jar
// libthrift-0.10.0.jar
// slf4j-api-1.7.12.jar

import ai.h2o.scoring.Row;
import ai.h2o.scoring.ScoringService;
import org.apache.thrift.TException;
import org.apache.thrift.protocol.TBinaryProtocol;
import org.apache.thrift.transport.TSocket;
import org.apache.thrift.transport.TTransport;
import java.util.List;

public class Main {
    public static void main(String[] args) {
        try {
            TTransport transport = new TSocket("localhost", 9090);
            transport.open();

            ScoringService.Client client = new ScoringService.Client(
                new TBinaryProtocol(transport));

            Row row = new Row(7.642, 3.436, 6.721, 1.020);
            List<Double> scores = client.score(row);
            System.out.println(scores);

            transport.close();
        } catch (TException ex) {
            ex.printStackTrace();
        }
    }
}

```

Scoring Service - HTTP Mode (JSON-RPC 2.0)

The HTTP mode allows you to use the scoring service using plaintext JSON-RPC calls. This is usually less performant compared to Thrift, but has the advantage of being usable from any HTTP client library in your language of choice, without any dependency on Thrift.

For JSON-RPC documentation, see <http://www.jsonrpc.org/specification>.

To start the scoring service in HTTP mode:

```

# See 'run_http_server.sh' for complete example.
export DRIVERLESS_AI_LICENSE_FILE="/path/to/license.sig"
python http_server.py --port=9090

```

To invoke scoring methods, compose a JSON-RPC message and make a HTTP POST request to <http://host:port/rpc> as follows:

```

# See 'run_http_client.sh' for complete example.
curl http://localhost:9090/rpc \
--header "Content-Type: application/json" \
--data @- <<EOF
{
    "id": 1,
    "method": "score",
    "params": {
        "row": [ 7.486, 3.277, 4.755, 2.354 ]
    }
}
EOF

```

Similarly, you can use any HTTP client library to reproduce the above result. For example, from Python, you can use the `requests` module as follows:

```

import requests
row = [7.486, 3.277, 4.755, 2.354]
req = dict(id=1, method='score', params=dict(row=row))
res = requests.post('http://localhost:9090/rpc', data=req)
print(res.json()['result'])

```

35.5 Python Scoring Pipeline FAQ

Why am I getting a “TensorFlow is disabled” message when I run the Python Scoring Pipeline?

If you ran an experiment when TensorFlow was enabled and then attempt to run the Python Scoring Pipeline, you may receive a message similar to the following:

```
TensorFlow is disabled. To enable, export DRIVERLESS_AI_ENABLE_TENSORFLOW=1 or set enable_tensorflow=true in config.toml.
```

To successfully run the Python Scoring Pipeline, you must enable the `DRIVERLESS_AI_ENABLE_TENSORFLOW` flag. For example:

```
export DRIVERLESS_AI_LICENSE_FILE="/path/to/license.sig"
DRIVERLESS_AI_ENABLE_TENSORFLOW=1 bash run_example.sh
```

35.6 Troubleshooting Python Environment Issues

The following instructions describe how to set up a cleanroom Ubuntu 16.04 virtual machine to test that this scoring pipeline works correctly.

Prerequisites:

- Install Virtualbox: `sudo apt-get install virtualbox`
 - Install Vagrant: <https://www.vagrantup.com/downloads.html>
1. Create configuration files for Vagrant.
 - `bootstrap.sh`: contains commands to set up Python 3.6 and OpenBLAS.
 - `Vagrantfile`: contains virtual machine configuration instructions for Vagrant and VirtualBox.

```
----- bootstrap.sh -----
#!/usr/bin/env bash

sudo apt-get -y update
sudo apt-get -y install apt-utils build-essential python-software-properties software-properties-common zip libopenblas-dev
sudo add-apt-repository -y ppa:deadsnakes/ppa
sudo apt-get update -yqq
sudo apt-get install -y python3.6 python3.6-dev python3-pip python3-dev python-virtualenv python3-virtualenv

# end of bootstrap.sh

----- Vagrantfile -----
# -*- mode: ruby -*-
# vi: set ft=ruby :

Vagrant.configure(2) do |config|
  config.vm.box = "ubuntu/xenial64"
  config.vm.provision :shell, path: "bootstrap.sh", privileged: false
  config.vm.hostname = "h2o"
  config.vm.provider "virtualbox" do |vb|
    vb.memory = "4096"
  end
end

# end of Vagrantfile
```

2. Launch the VM and SSH into it. Note that we’re also placing the scoring pipeline in the same directory so that we can access it later inside the VM.

```
cp /path/to/scorer.zip .
vagrant up
vagrant ssh
```

3. Test the scoring pipeline inside the virtual machine.

```
cp /vagrant/scorer.zip .
unzip scorer.zip
cd scoring-pipeline/
export DRIVERLESS_AI_LICENSE_FILE="/path/to/license.sig"
bash run_example.sh
```

At this point, you should see scores printed out on the terminal. If not, contact us at support@h2o.ai.

DRIVERLESS AI MLI STANDALONE PYTHON SCORING PACKAGE

This package contains an exported model and Python 3.6 source code examples for productionizing models built using H2O Driverless AI Machine Learning Interpretability (MLI) tool. This is only available for interpreted models and can be downloaded by clicking the **Scoring Pipeline** button the Interpreted Models page.

The files in this package allow you to obtain reason codes for a given row of data a couple of different ways:

- From Python 3.6, you can import a scoring module, and then use the module to transform and score on new data.
- From other languages and platforms, you can use the TCP/HTTP scoring service bundled with this package to call into the scoring pipeline module through remote procedure calls (RPC).

36.1 MLI Python Scoring Package Files

The **scoring-pipeline-mli** folder includes the following notable files:

- **example.py**: An example Python script demonstrating how to import and interpret new records.
- **run_example.sh**: Runs example.py (This also sets up a virtualenv with prerequisite libraries.)
- **run_example_shapley.sh**: Runs example_shapley.py. This compares K-LIME and Driverless AI Shapley reason codes.
- **tcp_server.py**: A standalone TCP server for hosting MLI services.
- **http_server.py**: A standalone HTTP server for hosting MLI services.
- **run_tcp_server.sh**: Runs the TCP scoring service (specifically, tcp_server.py).
- **run_http_server.sh**: Runs HTTP scoring service (runs http_server.py).
- **example_client.py**: An example Python script demonstrating how to communicate with the MLI server.
- **example_shapley.py**: An example Python script demonstrating how to compare K-LIME and Driverless AI Shapley reason codes.
- **run_tcp_client.sh**: Demonstrates how to communicate with the MLI service via TCP (runs example_client.py).
- **run_http_client.sh**: Demonstrates how to communicate with the MLI service via HTTP (using curl).

36.2 Quick Start

There are two methods for starting the MLI Standalone Scoring Pipeline.

36.2.1 Quick Start - Recommended Method

This is the recommended method for running the MLI Scoring Pipeline. Use this method if:

- You have an air gapped environment with no access to the Internet.
- You are running Power.
- You want an easy quick start approach.

Prerequisites

- A valid Driverless AI license key.
- A completed Driverless AI experiment.
- Downloaded MLI Scoring Pipeline.

Running the MLI Scoring Pipeline - Recommended

1. Download the TAR SH version of Driverless AI from <https://www.h2o.ai/download/> (for either Linux or IBM Power).
2. Use bash to execute the download. This creates a new **dai-nnn** folder.
3. Change directories into the new Driverless AI folder.

```
cd dai-nnn directory.
```

4. Run the following to install the Python Scoring Pipeline for your completed Driverless AI experiment:

```
./dai-env.sh pip install /path/to/your/scoring_experiment.whl
```

5. Run the following command to run the included scoring pipeline example:

```
DRIVERLESS_AI_LICENSE_KEY="pastekeyhere" SCORING_PIPELINE_INSTALL_DEPENDENCIES=0 ./dai-env.sh /path/to/your/run_example.sh
```

36.2.2 Quick Start - Alternative Method

This section describes an alternative method for running the MLI Standalone Scoring Pipeline. This version requires Internet access. It is also not supported on Power machines.

Prerequisites

- Valid Driverless AI license.
- The scoring module and scoring service are supported only on Linux with Python 3.6 and OpenBLAS.
- The scoring module and scoring service download additional packages at install time and require internet access. Depending on your network environment, you might need to set up internet access via a proxy.
- Apache Thrift (to run the scoring service in TCP mode)

Examples of how to install these prerequisites are below.

Installing Python 3.6

Installing Python3.6 on Ubuntu 16.10+:

```
sudo apt install python3.6 python3.6-dev python3-pip python3-dev \
python-virtualenv python3-virtualenv
```

Installing Python3.6 on Ubuntu 16.04:

```
sudo add-apt-repository ppa:deadsnakes/ppa
sudo apt-get update
sudo apt-get install python3.6 python3.6-dev python3-pip python3-dev \
python-virtualenv python3-virtualenv
```

Installing Conda 3.6:

You can install Conda using either Anaconda or Miniconda. Refer to the links below for more information:

- Anaconda - <https://docs.anaconda.com/anaconda/install.html>
- Miniconda - <https://docs.conda.io/en/latest/miniconda.html>

Installing the Thrift Compiler

Refer to Thrift documentation at <https://thrift.apache.org/docs/BuildingFromSource> for more information.

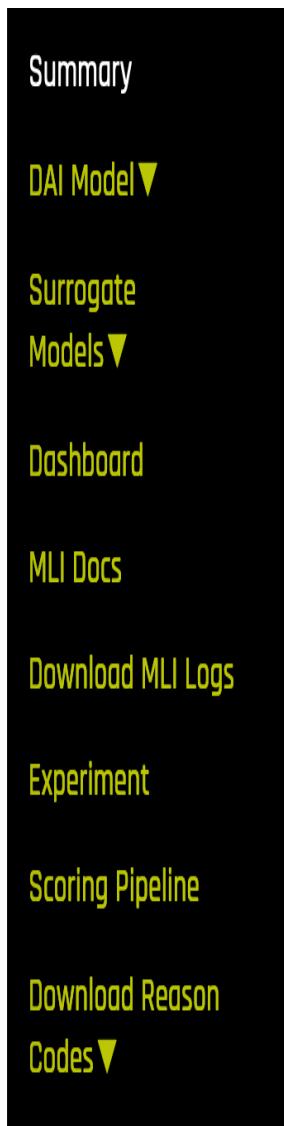
```
sudo apt-get install automake bison flex g++ git libevent-dev \
libssl-dev libtool make pkg-config libboost-all-dev ant
wget https://github.com/apache/thrift/archive/0.10.0.tar.gz
tar -xvf 0.10.0.tar.gz
cd thrift-0.10.0
./bootstrap.sh
./configure
make
sudo make install
```

Run the following to refresh the runtime shared after installing Thrift.

```
sudo ldconfig /usr/local/lib
```

Running the MLI Scoring Pipeline - Alternative Method

1. On the MLI page, click the **Scoring Pipeline** button.



2. Unzip the scoring pipeline, and run the following examples in the **scoring-pipeline-mli** folder.

Run the scoring module example. (This requires Linux and Python 3.6.)

```
bash run_example.sh
```

Run the TCP scoring server example. Use two terminal windows. (This requires Linux, Python 3.6 and Thrift.)

```
bash run_tcp_server.sh  
bash run_tcp_client.sh
```

Run the HTTP scoring server example. Use two terminal windows. (This requires Linux, Python 3.6 and Thrift.)

```
bash run_http_server.sh  
bash run_http_client.sh
```

Note: By default, the `run_*.sh` scripts mentioned above create a virtual environment using `virtualenv` and `pip`, within which the Python code is executed. The scripts can also leverage Conda (Ana-

conda/Miniconda) to create Conda virtual environment and install required package dependencies. The package manager to use is provided as an argument to the script.

```
# to use conda package manager
bash run_example.sh --pm conda

# to use pip package manager
bash run_example.sh --pm pip
```

36.3 MLI Python Scoring Module

The MLI scoring module is a Python module bundled into a standalone wheel file (name scoring_*.whl). All the prerequisites for the scoring module to work correctly are listed in the ‘requirements.txt’ file. To use the scoring module, all you have to do is create a Python virtualenv, install the prerequisites, and then import and use the scoring module as follows:

```
----- See 'example.py' for complete example. -----
from scoring_467931_20170921174120_b4066 import Scorer
scorer = KLimeScorer()           # Create instance.
score = scorer.score_reason_codes([ # Call score_reason_codes()
    7.416,                      # sepal_len
    3.562,                      # sepal_wid
    1.049,                      # petal_len
    2.388,                      # petal_wid
])
```

The scorer instance provides the following methods:

- `score_reason_codes (list)`: Get K-LIME reason codes for one row (list of values).
- `score_reason_codes_batch (dataframe)`: Takes and outputs a Pandas Dataframe
- `get_column_names ()`: Get the input column names
- `get_reason_code_column_names ()`: Get the output column names

The process of importing and using the scoring module is demonstrated by the bash script `run_example.sh`, which effectively performs the following steps:

```
----- See 'run_example.sh' for complete example. -----
virtualenv -p python3.6 env
source env/bin/activate
pip install -r requirements.txt
python example.py
```

36.4 K-LIME vs Shapley Reason Codes

There are times when the K-LIME model score is not close to the Driverless AI model score. In this case it may be better to use reason codes using the Shapley method on the Driverless AI model. Please note: the reason codes from Shapley will be in the transformed feature space.

To see an example of using both K-LIME and Driverless AI Shapley reason codes in the same Python session, run:

```
bash run_example_shapley.sh
```

For this batch script to succeed, MLI must be run on a Driverless AI model. If you have run MLI in standalone (external model) mode, there will not be a Driverless AI scoring pipeline.

If MLI was run with transformed features, the Shapley example scripts will not be exported. You can generate exact reason codes directly from the Driverless AI model scoring pipeline.

36.5 MLI Scoring Service Overview

The MLI scoring service hosts the scoring module as a HTTP or TCP service. Doing this exposes all the functions of the scoring module through remote procedure calls (RPC).

In effect, this mechanism allows you to invoke scoring functions from languages other than Python on the same computer, or from another computer on a shared network or the internet.

The scoring service can be started in two ways:

- In TCP mode, the scoring service provides high-performance RPC calls via Apache Thrift (<https://thrift.apache.org/>) using a binary wire protocol.
- In HTTP mode, the scoring service provides JSON-RPC 2.0 calls served by Tornado (<http://www.tornadoweb.org>).

Scoring operations can be performed on individual rows (row-by-row) using `score` or in batch mode (multiple rows at a time) using `score_batch`. Both functions allow you to specify `pred_contribs=[True|False]` to get MLI predictions (KLime/Shapley) on a new dataset. See the `example_shapley.py` file for more information.

36.5.1 MLI Scoring Service - TCP Mode (Thrift)

The TCP mode allows you to use the scoring service from any language supported by Thrift, including C, C++, C#, Cocoa, D, Dart, Delphi, Go, Haxe, Java, Node.js, Lua, perl, PHP, Python, Ruby and Smalltalk.

To start the scoring service in TCP mode, you will need to generate the Thrift bindings once, then run the server:

```
----- See 'run_tcp_server.sh' for complete example. -----
thrift --gen py scoring.thrift
python tcp_server.py --port=9090
```

Note that the Thrift compiler is only required at build-time. It is not a run time dependency, i.e. once the scoring services are built and tested, you do not need to repeat this installation process on the machines where the scoring services are intended to be deployed.

To call the scoring service, simply generate the Thrift bindings for your language of choice, then make RPC calls via TCP sockets using Thrift's buffered transport in conjunction with its binary protocol.

```
----- See 'run_tcp_client.sh' for complete example. -----
thrift --gen py scoring.thrift

----- See 'example_client.py' for complete example. -----
socket = TSocket.TSocket('localhost', 9090)
transport = TTransport.TBufferedTransport(socket)
protocol = TBinaryProtocol.TBinaryProtocol(transport)
client = ScoringService.Client(protocol)
transport.open()
row = Row()
row.sepalLen = 7.416 # sepal_len
row.sepalWid = 3.562 # sepal_wid
row.petalLen = 1.049 # petal_len
row.petalWid = 2.388 # petal_wid
scores = client.score_reason_codes(row)
transport.close()
```

You can reproduce the exact same result from other languages, e.g. Java:

```
thrift --gen java scoring.thrift

// Dependencies:
// commons-codec-1.9.jar
// commons-logging-1.2.jar
// httpclient-4.4.1.jar
// httpcore-4.4.1.jar
// libthrift-0.10.0.jar
// slf4j-api-1.7.12.jar

import ai.h2o.scoring.Row;
import ai.h2o.scoring.ScoringService;
import org.apache.thrift.TException;
import org.apache.thrift.protocol.TBinaryProtocol;
import org.apache.thrift.transport.TSocket;
import org.apache.thrift.transport.TTransport;
```

(continues on next page)

(continued from previous page)

```

import java.util.List;

public class Main {
    public static void main(String[] args) {
        try {
            Transport transport = new TSocket("localhost", 9090);
            transport.open();

            ScoringService.Client client = new ScoringService.Client(
                new TBinaryProtocol(transport));

            Row row = new Row(7.642, 3.436, 6.721, 1.020);
            List<Double> scores = client.score_reason_codes(row);
            System.out.println(scores);

            transport.close();
        } catch (TException ex) {
            ex.printStackTrace();
        }
    }
}

```

36.5.2 Scoring Service - HTTP Mode (JSON-RPC 2.0)

The HTTP mode allows you to use the scoring service using plaintext JSON-RPC calls. This is usually less performant compared to Thrift, but has the advantage of being usable from any HTTP client library in your language of choice, without any dependency on Thrift.

For JSON-RPC documentation, see <http://www.jsonrpc.org/specification>.

To start the scoring service in HTTP mode:

```
----- See 'run_http_server.sh' for complete example. -----
python http_server.py --port=9090
```

To invoke scoring methods, compose a JSON-RPC message and make a HTTP POST request to <http://host:port/rpc> as follows:

```
----- See 'run_http_client.sh' for complete example. -----
curl http://localhost:9090/rpc \
--header "Content-Type: application/json" \
--data @- <<EOF
{
    "id": 1,
    "method": "score_reason_codes",
    "params": {
        "row": [ 7.486, 3.277, 4.755, 2.354 ]
    }
}
EOF
```

Similarly, you can use any HTTP client library to reproduce the above result. For example, from Python, you can use the requests module as follows:

```

import requests
row = [7.486, 3.277, 4.755, 2.354]
req = dict(id=1, method='score_reason_codes', params=dict(row=row))
res = requests.post('http://localhost:9090/rpc', data=req)
print(res.json()['result'])

```


MOJO SCORING PIPELINES

As indicated previously, the MOJO Scoring Pipeline provides a standalone scoring pipeline that converts experiments to MOJOS, which can be scored in real time. The MOJO Scoring Pipeline is available as either a Java runtime or a C++ runtime (with Python and R wrappers).

37.1 Driverless AI MOJO Scoring Pipeline - Java Runtime

For completed experiments, Driverless AI automatically converts models to MOJOS (Model Objects, Optimized). The MOJO Scoring Pipeline is a scoring engine that can be deployed in any Java environment for scoring in real time. (Refer to *Driverless AI MOJO Scoring Pipeline - C++ Runtime with Python and R Wrappers* for information about the C++ scoring runtime with Python and R wrappers.)

Keep in mind that, similar to H2O-3, MOJOS are tied to experiments. Experiments and MOJOS are not automatically upgraded when Driverless AI is upgraded.

Notes:

- This scoring pipeline is not currently available for TensorFlow or RuleFit models.
- To disable the automatic creation of this scoring pipeline, set the **Make MOJO Scoring Pipeline** expert setting to **Off**.
- You can have Driverless AI attempt to reduce the size of the MOJO scoring pipeline when it is being built by enabling the *Attempt to Reduce the Size of the MOJO* expert setting.

37.1.1 Prerequisites

The following are required in order to run the MOJO scoring pipeline.

- Java 7 runtime (JDK 1.7) or newer. **NOTE:** We recommend using Java 11+ due to a bug in Java. (See <https://bugs.openjdk.java.net/browse/JDK-8186464>.)
- Valid Driverless AI license. You can download the `license.sig` file from the machine hosting Driverless AI (usually in the **license** folder). Copy the license file into the downloaded `mojo-pipeline` folder.
- `mojo2-runtime.jar` file. This is available from the top navigation menu in the Driverless AI UI and in the downloaded `mojo-pipeline.zip` file for an experiment.

License Specification

Driverless AI requires a license to be specified in order to run the MOJO Scoring Pipeline. The license can be specified in one of the following ways:

- Via an environment variable:
 - DRIVERLESS_AI_LICENSE_FILE: Path to the Driverless AI license file, or
 - DRIVERLESS_AI_LICENSE_KEY: The Driverless AI license key (Base64 encoded string)
- Via a system property of JVM (-D option):
 - ai.h2o.mojos.runtime.license.file: Path to the Driverless AI license file, or
 - ai.h2o.mojos.runtime.license.key: The Driverless AI license key (Base64 encoded string)
- Via an application classpath:
 - The license is loaded from a resource called /license.sig.
 - The default resource name can be changed via the JVM system property ai.h2o.mojos.runtime.license.filename.

For example:

```
# Specify the license via a temporary environment variable
export DRIVERLESS_AI_LICENSE_FILE="path/to/license.sig"
```

37.1.2 MOJO Scoring Pipeline Files

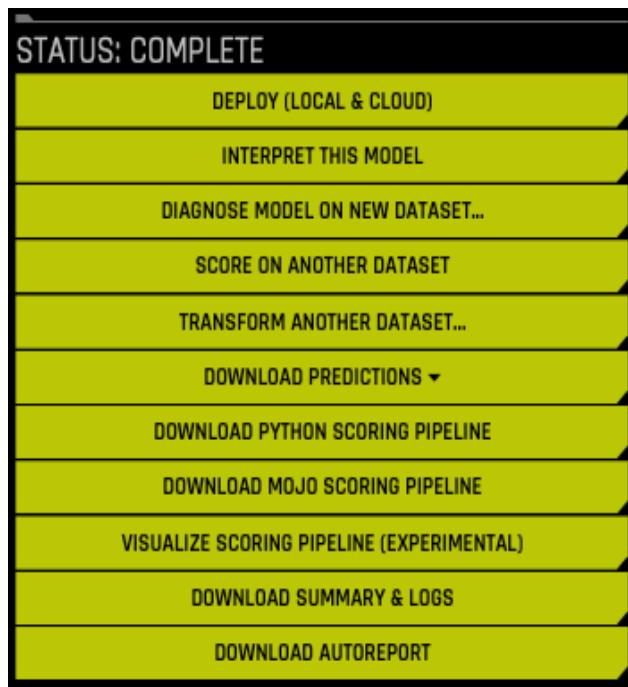
The **mojo-pipeline** folder includes the following files:

- **run_example.sh**: An bash script to score a sample test set.
- **pipeline.mojo**: Standalone scoring pipeline in MOJO format.
- **mojo2-runtime.jar**: MOJO Java runtime.
- **example.csv**: Sample test set (synthetic, of the correct format).
- **DOT files**: Text files that can be rendered as graphs that provide a visual representation of the MOJO scoring pipeline (can be edited to change the appearance and structure of a rendered graph).
- **PNG files**: Image files that provide a visual representation of the MOJO scoring pipeline.

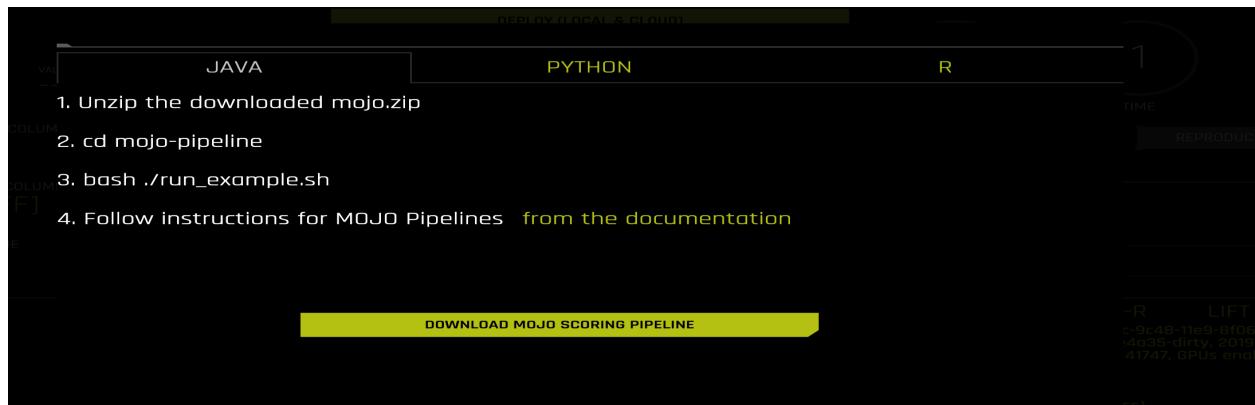
37.1.3 Quickstart

Before running the quickstart examples, be sure that the MOJO scoring pipeline is already downloaded and unzipped:

1. On the completed Experiment page, click on the **Download MOJO Scoring Pipeline** button.



2. In the pop-up menu that appears, click on the **Download MOJO Scoring Pipeline** button once again to download the **scorer.zip** file for this experiment onto your local machine. Refer to the provided instructions for Java, Python, or R.



3. Run the following to score all rows in the sample test set with the file paths to the test set (example.csv), MOJO pipeline (pipeline.mojo) and license (license.sig) stored in environment variables TEST_SET_FILE, MOJO_PIPELINE_FILE, DRIVERLESS_AI_LICENSE_KEY:

```
bash run_example.sh
```

4. Run the following to score a specific test set (example.csv) with MOJO pipeline (pipeline.mojo) and the license file (license.sig):

```
bash run_example.sh pipeline.mojo example.csv license.sig
```

5. To run the Java application for data transformation directly:

```
java -Dai.h2o.mojos.runtime.license.file=license.sig -cp mojo2-runtime.jar ai.h2o.mojos.ExecuteMojo pipeline.mojo example.csv
```

Note: For very large models, it may be necessary to increase the memory limit when running the Java application for data transformation. This can be done by specifying `-Xmx25g` when running the above command.

37.1.4 MOJO Scoring Command Line Options

Executing the Java Runtime

The following is an example of how the Java runtime can be executed from the command line:

```
java <JVM options> -cp mojo2-runtime-x.y.z.jar:your-other.jar:many-more-libs.jar ai.h2o.mojos.ExecuteMojo path-to/pipeline.mojo path-to/input.csv path-to/  
→output.csv
```

The following is an example of how the Java runtime can be executed if additional libraries are not needed:

```
java <JVM options> -jar mojo2-runtime-x.y.z.jar path-to/pipeline.mojo path-to/input.csv path-to/output.csv
```

Note: Data can be streamed from stdin to stdout by replacing both the input and output CSV arguments with ` -`.

JVM Options

- `keepCarriageReturn` (boolean) - Specify whether to keep the carriage return after parsing. This value defaults to True.
- `stripCrFromLastColumn` (boolean) - Workaround for issues relating to the OpenCSV parser. This value defaults to True.
- `quotedHeaders` (boolean) - Specify whether to quote header names in the output CSV file. This value defaults to False.
- `separator` (char) - Specify the separator used between CSV fields. The special value `TAB` can be used for tab-separated values. This value defaults to ` `.
- `escapeChar` (char) - Specify the escape character for parsing CSV fields. If this value is not specified, then no escaping is attempted. This value defaults to an empty string.
- `batch` (int) - Specify the number of input records brought into memory for batch processing (determines consumed memory). This value defaults to 1000.
- `pipelineFormats` (string) - When multiple formats are recognized, this option specifies the order in which they are tried. This value defaults to `pb,toml,klime,h2o3`.
- `date.formats` (string) - Specify a format for dates. This value defaults to an empty string.
- `exposedInputs` (string) - Specify a comma separated list of input cols that are needed on output. The special value `ALL` takes all inputs. This defaults to a null value.

37.1.5 Execute the MOJO from Java

1. Open a new terminal window, create an experiment folder, and change directories to that new folder:

```
mkdir experiment && cd experiment
```

2. Create your main program in the **experiment** folder by creating a new file called **Main.java** (for example, using `vim Main.java`). Include the following contents.

```
import ai.h2o.mojos.runtime.MojoPipeline;
import ai.h2o.mojos.runtime.frame.MojoFrame;
import ai.h2o.mojos.runtime.frame.MojoFrameBuilder;
import ai.h2o.mojos.runtime.frame.MojoRowBuilder;
import ai.h2o.mojos.runtime.lic.LicenseException;
import ai.h2o.mojos.runtime.utils.CsvWritingBatchHandler;
import com.opencsv.CSVWriter;
import java.io.BufferedWriter;
import java.io.IOException;
import java.io.OutputStreamWriter;
import java.io.Writer;

public class DocsExample {
    public static void main(String[] args) throws IOException, LicenseException {
        // Load model and csv
        final MojoPipeline model = MojoPipeline.loadFrom("pipeline.mojo");

        // Get and fill the input columns
        final MojoFrameBuilder frameBuilder = model.getInputFrameBuilder();
        final MojoRowBuilder rowBuilder = frameBuilder.getMojoRowBuilder();
        rowBuilder.setValue("AGE", "68");
        rowBuilder.setValue("RACE", "2");
        rowBuilder.setValue("DCAPS", "2");
        rowBuilder.setValue("VOL", "0");
        rowBuilder.setValue("GLEASON", "6");
        frameBuilder.addRow(rowBuilder);

        // Create a frame which can be transformed by MOJO pipeline
        final MojoFrame iframe = frameBuilder.toMojoFrame();

        // Transform input frame by MOJO pipeline
        final MojoFrame oframe = model.transform(iframe);
        // `MojoFrame.debug()` can be used to view the contents of a Frame
        // oframe.debug();

        // Output prediction as CSV
        final Writer writer = new BufferedWriter(new OutputStreamWriter(System.out));
        final CSVWriter csvWriter = new CSVWriter(writer);
        CsvWritingBatchHandler.csvWriteFrame(csvWriter, oframe, true);
    }
}
```

3. Compile the source code with the files of the MOJO runtime (`mojo2-runtime.jar`) and MOJO pipeline (`pipeline.mojo`) copied into the experiment:

```
javac -cp mojo2-runtime.jar -J-Xms2g -J-XX:MaxPermSize=128m Main.java
```

4. Run the MOJO example with the license (`license.sig`) copied into the experiment:

```
# Linux and OS X users
java -Dai.h2o.mojos.runtime.license.file=license.sig -cp .:mojo2-runtime.jar Main

# Windows users
java -Dai.h2o.mojos.runtime.license.file=license.sig -cp .;mojo2-runtime.jar Main
```

5. The following output is displayed:

```
CAPSULE.True
0.5442205910902282
```

37.1.6 Using the MOJO Scoring Pipeline with Spark/Sparkling Water

Note: The Driverless AI 1.5 release will be the last release with TOML-based MOJO2. Releases after 1.5 will include protobuf-based MOJO2.

MOJO scoring pipeline artifacts can be used in Spark to deploy predictions in parallel using the Sparkling Water API. This section shows how to load and run predictions on the MOJO scoring pipeline in Spark using Scala and the Python API.

In the event that you upgrade H2O Driverless AI, we have a good news! Sparkling Water is backwards compatible with MOJO versions produced by older Driverless AI versions.

Requirements

- You must have a Spark cluster with the Sparkling Water JAR file passed to Spark.
- To run with PySparkling, you must have the PySparkling zip file.

The H2OContext does not have to be created if you only want to run predictions on MOJOs using Spark. This is because the scoring is independent of the H2O run-time.

Preparing Your Environment

In order use the MOJO scoring pipeline, Driverless AI license has to be passed to Spark. This can be achieved via --jars argument of the Spark launcher scripts.

Note: In Local Spark mode, please use --driver-class-path to specify path to the license file.

PySparkling

First, start PySpark with PySparkling Python package and Driverless AI license.

```
./bin/pyspark --jars license.sig --py-files pysparkling.zip
```

or, you can download official Sparkling Water distribution from H2O Download page. Please follow steps on the Sparkling Water download page. Once you are in the Sparkling Water directory, you can call:

```
./bin/pysparkling --jars license.sig
```

At this point, you should have available a PySpark interactive terminal where you can try out predictions. If you would like to productionalize the scoring process, you can use the same configuration, except instead of using ./bin/pyspark, you would use ./bin/spark-submit to submit your job to a cluster.

```
# First, specify the dependencies
from pysparkling.ml import H2OMOJOPipelineModel, H2OMOJOSettings
```

```
# The 'namedMojoOutputColumns' option ensures that the output columns are named properly.
# If you want to use old behavior when all output columns were stored inside an array,
# set it to False. However we strongly encourage users to use True which is defined as a default value.
settings = H2OMOJOSettings(namedMojoOutputColumns = True)

# Load the pipeline. 'settings' is an optional argument. If it's not specified, the default values are used.
mojo = H2OMOJOPipelineModel.createFromMojo("file:///path/to/the/pipeline.mojo", settings)
```

```
# Load the data as Spark's Data Frame
dataFrame = spark.read.csv("file:///path/to/the/data.csv", header=True)
```

```
# Run the predictions. The predictions contain all the original columns plus the predictions
# added as new columns
predictions = mojo.transform(dataFrame)

# You can easily get the predictions for a desired column using the helper function as
predictions.select(mojo.selectPredictionUDF("AGE")).collect()
```

Sparkling Water

First, start Spark with Sparkling Water Scala assembly and Driverless AI license.

```
./bin/spark-shell --jars license.sig,sparkling-water-assembly.jar
```

or, you can download official Sparkling Water distribution from H2O Download page. Please follow steps on the Sparkling Water download page. Once you are in the Sparkling Water directory, you can call:

```
./bin/sparkling-shell --jars license.sig
```

At this point, you should have available a Sparkling Water interactive terminal where you can carry out predictions. If you would like to productionalize the scoring process, you can use the same configuration, except instead of using `./bin/spark-shell`, you would use `./bin/spark-submit` to submit your job to a cluster.

```
// First, specify the dependencies
import ai.h2o.sparkling.ml.models.{H2OMOJOPipelineModel, H2OMOJOSettings}

// The 'namedMojoOutputColumns' option ensures that the output columns are named properly.
// If you want to use old behavior when all output columns were stored inside an array,
// set it to false. However we strongly encourage users to use true which is defined as a default value.
val settings = H2OMOJOSettings(namedMojoOutputColumns = true)

// Load the pipeline. 'settings' is an optional argument. If it's not specified, the default values are used.
val mojo = H2OMOJOPipelineModel.createFromMojo("file:///path/to/the/pipeline.mojo", settings)

// Load the data as Spark's Data Frame
val dataFrame = spark.read.option("header", "true").csv("file:///path/to/the/data.csv")

// Run the predictions. The predictions contain all the original columns plus the predictions
// added as new columns
val predictions = mojo.transform(dataFrame)

// You can easily get the predictions for desired column using the helper function as follows:
predictions.select(mojo.selectPredictionUDF("AGE")).show()
```

37.2 Driverless AI MOJO Scoring Pipeline - C++ Runtime with Python and R Wrappers

The C++ Scoring Pipeline is provided as R and Python packages for the protobuf-based MOJO2 protocol. The packages are self contained, so no additional software is required. Simply build the MOJO Scoring Pipeline and begin using your preferred method.

Notes:

- These scoring pipelines are currently not available for RuleFit models.
- The **Download MOJO Scoring Pipeline** button appears as **Build MOJO Scoring Pipeline** if the MOJO Scoring Pipeline is disabled.
- You can have Driverless AI attempt to reduce the size of the MOJO scoring pipeline when it is being built by enabling the *Attempt to Reduce the Size of the MOJO* expert setting.

37.2.1 Downloading the Scoring Pipeline Runtimes

The R and Python packages can be downloaded from within the Driverless AI application. To do this, click **Resources**, then click **MOJO2 R Runtime** and **MOJO2 Py Runtime** from the drop-down menu. In the pop-up menu that appears, click the button that corresponds to the OS you are using. Choose from Linux, Mac OS X, and IBM PowerPC.

37.2.2 Examples

The following examples show how to use the R and Python APIs of the C++ MOJO runtime.

R Example

Python Example

Prerequisites

- Linux OS (x86 or PPC) or Mac OS X (10.9 or newer)
- Driverless AI License (either file or environment variable)

- Rcpp (>=1.0.0)
- data.table

Running the MOJO2 R Runtime

```
# Install the R MOJO runtime using one of the methods below

# Install the R MOJO runtime on PPC Linux
install.packages("./daimojo_2.4.8_ppc64le-linux.tar.gz")

# Install the R MOJO runtime on x86 Linux
install.packages("./daimojo_2.4.8_x86_64-linux.tar.gz")

#Install the R MOJO runtime on Mac OS X
install.packages("./daimojo_2.4.8_x86_64-darwin.tar.gz")

# Load the MOJO
library(daimojo)
m <- load.mojo("./mojo-pipeline/pipeline.mojo")

# retrieve the creation time of the MOJO
create.time(m)
## [1] "2019-11-18 22:00:24 UTC"

# retrieve the UUID of the experiment
uuid(m)
## [1] "65875c15-943a-4bc0-a162-b8984fe8e50d"

# Load data and make predictions
col_class <- setNames(feature.types(m), feature.names(m)) # column names and types

library(data.table)
d <- fread("./mojo-pipeline/example.csv", colClasses=col_class, header=TRUE, sep=",")

predict(m, d)
##          label.B      label.M
## 1  0.08287659 0.91712341
## 2  0.77655075 0.22344925
## 3  0.58438434 0.41561566
## 4  0.10570505 0.89429495
## 5  0.01685609 0.98314391
## 6  0.23656610 0.76343390
## 7  0.17410333 0.82889667
## 8  0.10157948 0.89842052
## 9  0.13546191 0.86453809
## 10 0.94778244 0.05221756
```

Prerequisites

- Linux OS (x86 or PPC) or Mac OS X (10.9 or newer)
- Driverless AI License (either file or environment variable)
- Python 3.6
- datatable. Run the following to install:

```
# Install on Linux PPC, Linux x86, or Mac OS X
pip install datatable
```

- Non-binary version of protobuf:

```
pip install --no-binary=protobuf protobuf
```

- Python MOJO runtime. Run one of the following commands after downloading from the GUI:

```
# Install the MOJO runtime on Linux PPC
pip install daimojo-2.4.8-cp36-cp36m-linux_ppc64le.whl

# Install the MOJO runtime on Linux x86
pip install daimojo-2.4.8-cp36-cp36m-linux_x86_64.whl

# Install the MOJO runtime on Mac OS X
pip install daimojo-2.4.8-cp36-cp36m-macosx_10_7_x86_64.whl
```

Running the MOJO2 Python Runtime

```
# import the daimojo model package
import daimojo.model

# specify the location of the MOJO
m = daimojo.model("./mojo-pipeline/pipeline.mojo")

# retrieve the creation time of the MOJO
m.created_time
# 'Mon November 18 14:00:24 2019'
```

(continues on next page)

(continued from previous page)

```

# retrieve the UUID of the experiment
m.uuid

# retrieve a list of missing values
m.missing_values
# ['!',
# '?',
# 'None',
# 'nan',
# 'NA',
# 'N/A',
# 'unknown',
# 'inf',
# '-inf',
# '1.7976931348623157e+308',
# '-1.7976931348623157e+308']

# retrieve the feature names
m.feature_names
# ['clump_thickness',
# 'uniformity_cell_size',
# 'uniformity_cell_shape',
# 'marginal_adhesion',
# 'single_epithelial_cell_size',
# 'bare_nuclei',
# 'bland_chromatin',
# 'normal_nucleoli',
# 'mitoses']

# retrieve the feature types
m.feature_types
# ['float32',
# 'float32',
# 'float32']

# retrieve the output names
m.output_names
# ['label.B', 'label.M']

# retrieve the output types
m.output_types
# ['float64', 'float64']

# import the datatable module
import datatable as dt

# parse the example.csv file
pydt = dt.fread("./mojo-pipeline/example.csv", na_strings=m.missing_values, header=True, separator=',')
pydt
#   clump_thickness uniformity_cell_size uniformity_cell_shape marginal_adhesion single_epithelial_cell_size bare_nuclei bland_chromatin normal_
#   nucleoli mitoses
# 0          8             1              3            10               6             6              9
# 1          1             2              1              2               5             3              4
# 2          8             8              1              1               9             4             10              3
# 3          5             4              6              9              10              4             8              1
# 4          2             3             10              8              1               8             3              6
# 5          3             4              3              4               5              10             1              2
# 6          5             3              8              4               5              10             1              2
# 7          3             2             10              2               9              1              2              9
# 8          5             4              8              9               2              10             10              3
# 9          3             6              3              8               5              2              3              5
# 10         4             4              2              2               8              1              2              8

# [10 rows x 9 columns]

# retrieve the column types
pydt.stypes
# (stype.float64,
# stype.float64,
# stype.float64)

# make predictions on the example.csv file
res = m.predict(pydt)

# retrieve the predictions
res
#      label.B    label.M
# 0  0.0828766  0.917123
# 1  0.776551   0.223449
# 2  0.584384   0.415616
# 3  0.105705   0.894295
# 4  0.0168561  0.983144

```

(continues on next page)

(continued from previous page)

```
# 5      0.236566      0.763434
# 6      0.174103      0.825897
# 7      0.101579      0.898421
# 8      0.135462      0.864538
# 9      0.947782      0.0522176

# [10 rows x 2 columns]

# retrieve the prediction column names
res.names
# ('label.B', 'label.M')

# retrieve the prediction column types
res.types
# (stype.float64, stype.float64)

# convert datatable results to common data types
# res.to_pandas() # need pandas
# res.to_numpy() # need numpy
res.to_list()
```

37.3 MOJO2 Javadoc

The downloaded **mojo.zip** file contains the entire scoring pipeline. This pipeline also includes a MOJO2 Javadoc, which can be opened by running the following in the **mojo-pipeline** folder:

```
jar -xf mojo2-runtime-javadoc.jar
```

This opens the following files:

```
SaraJanes-MBP13TB:mojo-pipeline sarajane$ jar -xf mojo2-runtime-javadoc.jar
SaraJanes-MBP13TB:mojo-pipeline sarajane$ ls
META-INF           index-all.html
README.txt         index.html
ai                 mojo2-runtime-javadoc.jar
allclasses-frame.html   mojo2-runtime.jar
allclasses-noframe.html  overview-frame.html
classestree.txt    overview-summary.html
constant-values.html  overview-tree.html
deprecated-list.html package-list
example.csv        pipeline.mojo
help-doc.html      run_example.sh
highlight-LICENSE.txt  script.js
highlight.css       serialized-form.html
highlight.pack.js   stylesheet.css
SaraJanes-MBP13TB:mojo-pipeline sarajane$
```

Open the **index.html** file to view the MOJO2 Javadoc.

CHAPTER
THIRTYEIGHT

DEPLOYING THE MOJO PIPELINE

Driverless AI can deploy the MOJO scoring pipeline for you to test and/or to integrate into a final product.

Notes:

- Only MOJO Java runtime deployments are supported.
- This section describes how to deploy a MOJO scoring pipeline and assumes that a MOJO scoring pipeline exists. Refer to the [MOJO Scoring Pipelines](#) section for information on how to build a MOJO scoring pipeline.
- This is an early feature that will continue to support additional deployments.

38.1 Deployments Overview Page

All of the existing MOJO scoring pipeline deployments are available in the Deployments Overview page, which is available from the top menu. This page lists all active deployments and the information needed to access the respective endpoints. In addition, it allows you to stop any deployments that are no longer needed.

Deployments						SELECT	SORT BY ▾
NAME	DEPLOYED ON	API KEY	LOCATION	STATUS	OPTIONS		
wavobacu-lambda2 wavobacu	DEPLOYED ON Lambda	lysNa0qoL86PNv4olbu3D8HyWAN...	https://h9kvsd85c.execute-api.us...	Running			
wavobacu-lambda wavobacu	DEPLOYED ON Lambda	xUu9xtXB8F4bHEsgbYnPT58QQfUT...	https://2hvdcvb18.execute-api.us...	Running			

38.2 Available Deployments

The following deployments are available in Driverless AI.

- [Amazon Lambda Deployment](#)
- [REST Server Deployment](#)

38.3 Amazon Lambda Deployment

Driverless AI can deploy the trained MOJO scoring pipeline as an [AWS Lambda Function](#), i.e., a server-less scorer running in Amazon Cloud and charged by the actual usage.

38.3.1 Additional Resources

Refer to the [aws-lambda-scorer](#) folder in the dai-deployment-templates repository to see different deployment templates for AWS Lambda scorer.

38.3.2 Driverless AI Prerequisites

- Driverless AI MOJO Scoring Pipeline: To deploy a MOJO scoring pipeline as an AWS Lambda function, the MOJO pipeline archive has to be created first by choosing the **Build MOJO Scoring Pipeline** option on the completed experiment page. Refer to the [MOJO Scoring Pipelines](#) section for information on how to build a MOJO scoring pipeline.
- Current Driverless AI license. The Driverless AI deployment pipeline to AWS Lambdas explicitly sets the license key as an environment variable. You will not be able to use MOJOS if your Driverless AI license is expired. If you have an expired license, you can update this manually for each MOJO in AWS, or you can update all MOJOS for a deployment region using a script. Refer to [Updating Driverless AI Licenses on AWS Lambda](#) for more information.

38.3.3 AWS Prerequisites

Usage Plans

Usage plans must be enabled in the target AWS region in order for API keys to work when accessing the AWS Lambda via its REST API. Refer to <https://aws.amazon.com/blogs/aws/new-usage-plans-for-amazon-api-gateway/> for more information.

Access Permissions

The following AWS access permissions need to be provided to the role in order for Driverless AI Lambda deployment to succeed.

- AWSLambdaFullAccess
- IAMFullAccess
- AmazonAPIGatewayAdministrator

The screenshot shows the 'Permissions' tab of an AWS IAM role configuration. It lists three attached policies: 'AWSLambdaFullAccess', 'IAMFullAccess', and 'AmazonAPIGatewayAdministrator'. Each policy is identified as an 'AWS managed policy'. There are also tabs for 'Groups', 'Tags', 'Security credentials', and 'Access Advisor'.

Policy name	Policy type	Action
AWSLambdaFullAccess	AWS managed policy	X
IAMFullAccess	AWS managed policy	X
AmazonAPIGatewayAdministrator	AWS managed policy	X

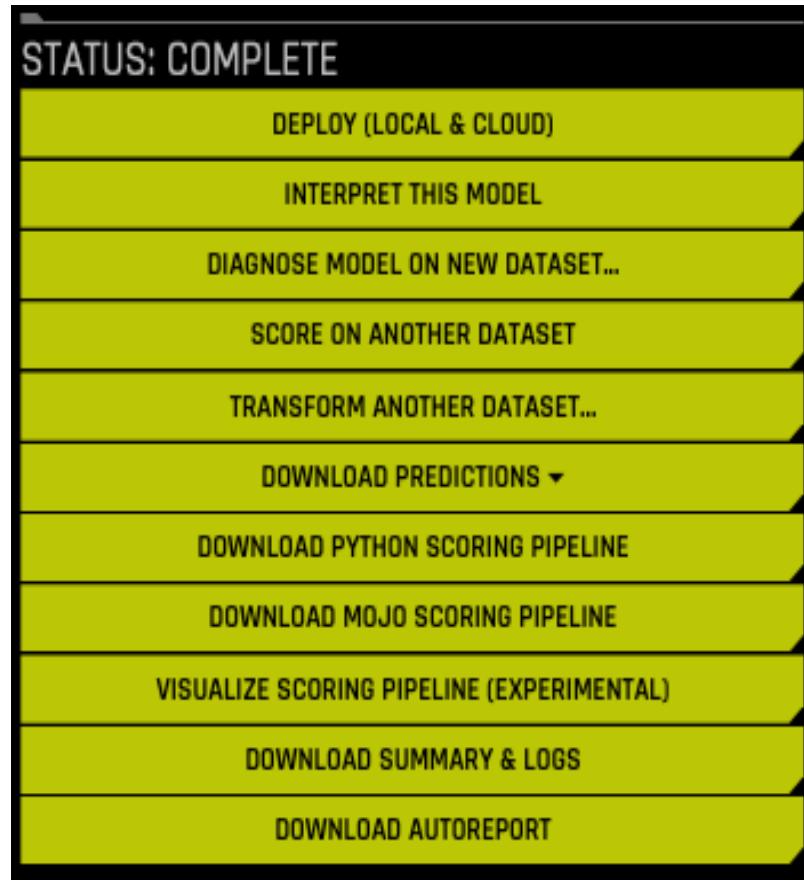
The policy can be further stripped down to restrict Lambda and S3 rights using the JSON policy definition as follows:

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "VisualEditor0",
            "Effect": "Allow",
            "Action": [
                "iam:GetPolicyVersion",
                "iam>DeletePolicy",
                "iam>CreateRole",
                "iam:AttachRolePolicy",
                "iam>ListInstanceProfilesForRole",
                "iam:PassRole",
                "iam:DetachRolePolicy",
                "iam>ListAttachedRolePolicies",
                "iam:GetRole",
                "iam:GetPolicy",
                "iam>DeleteRole",
                "iam>CreatePolicy",
                "iam>ListPolicyVersions"
            ],
            "Resource": [
                "arn:aws:iam:::role/h2oai*",
                "arn:aws:iam:::policy/h2oai*"
            ]
        },
        {
            "Sid": "VisualEditor1",
            "Effect": "Allow",
            "Action": "apigateway:/*",
            "Resource": "*"
        },
        {
            "Sid": "VisualEditor2",
            "Effect": "Allow",
            "Action": [
                "lambda>CreateFunction",
                "lambda>ListFunctions",
                "lambda:InvokeFunction",
                "lambda:GetFunction",
                "lambda:UpdateFunctionConfiguration",
                "lambda>DeleteFunctionConcurrency",
                "lambda:RemovePermission",
                "lambda:UpdateFunctionCode",
                "lambda>AddPermission",
                "lambda>ListVersionsByFunction",
                "lambda:GetFunctionConfiguration",
                "lambda>DeleteFunction",
                "lambda:PutFunctionConcurrency",
                "lambda:GetPolicy"
            ],
            "Resource": "arn:aws:lambda:::function:h2oai*"
        },
        {
            "Sid": "VisualEditor3",
            "Effect": "Allow",
            "Action": "s3:/*",
            "Resource": [
                "arn:aws:s3:::h2oai/*",
                "arn:aws:s3:::h2oai*"
            ]
        }
    ]
}
```

38.3.4 Deploying on Amazon Lambda

Once the MOJO pipeline archive is ready, Driverless AI provides a **Deploy (Local & Cloud)** option on the completed experiment page.

Notes: This button is only available after the MOJO Scoring Pipeline has been built.



This option opens a new dialog for setting the AWS account credentials (or use those supplied in the Driverless AI configuration file or environment variables), AWS region, and the desired deployment name (which must be unique per Driverless AI user and AWS account used).

Deploy

Deployment for: kabopull

Amazon Lambda

REST Server

Deploy experiment kabopull to Amazon Lambda.

Deployment Name: kabopull-lambda

Use AWS environment variables

AWS Access Key ID

AWS Secret Access Key

Region: SELECT REGION ▾

Price: [AWS Lambda Pricing](#)

Prior to filling these fields, first log in to your Amazon AWS account and configure your account based on the configuration you need. Copy the Secret Access Key and Access Key ID provided by AWS and fill in these fields.

DEPLOY

Amazon Lambda deployment parameters:

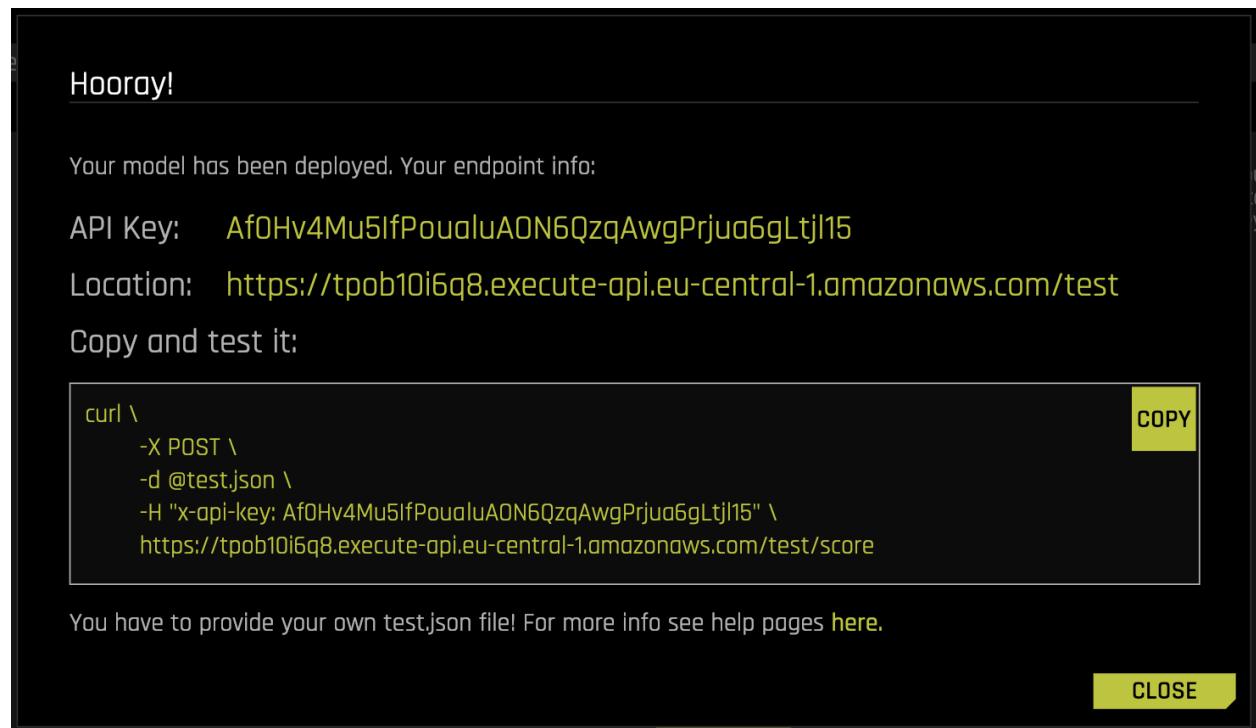
- **Deployment Name:** A unique name of the deployment. By default, Driverless AI offers a name based on the

name of the experiment and the deployment type. This has to be unique both for Driverless AI user and the AWS account used.

- **Region:** The AWS region to deploy the MOJO scoring pipeline to. It makes sense to choose a region geographically close to any client code calling the endpoint in order to minimize request latency. (See also [AWS Regions and Availability Zones](#).)
- **Use AWS environment variables:** If enabled, the AWS credentials are taken from the Driverless AI configuration file (see `records` `deployment_aws_access_key_id` and `deployment_aws_secret_access_key`) or environment variables (`DRIVERLESS_AI_DEPLOYMENT_AWS_ACCESS_KEY_ID` and `DRIVERLESS_AI_DEPLOYMENT_AWS_SECRET_ACCESS_KEY`). This would usually be entered by the Driverless AI installation administrator.
- **AWS Access Key ID and AWS Secret Access Key:** Credentials to access the AWS account. This pair of secrets identifies the AWS user and the account and can be obtained from the AWS account console.

38.3.5 Testing the Lambda Deployment

On a successful deployment, all the information needed to access the new endpoint (URL and an API Key) is printed, and the same information is available in the [Deployments Overview Page](#) after clicking on the deployment row.



Hooray!

Your model has been deployed. Your endpoint info:

API Key: Af0Hv4Mu5lfPoualuAON6QzqAwgPrju6gLtjl15

Location: <https://tpob10i6q8.execute-api.eu-central-1.amazonaws.com/test>

Copy and test it:

```
curl \
-X POST \
-d @test.json \
-H "x-api-key: Af0Hv4Mu5lfPoualuAON6QzqAwgPrju6gLtjl15" \
https://tpob10i6q8.execute-api.eu-central-1.amazonaws.com/test/score
```

You have to provide your own test.json file! For more info see help pages [here](#).

COPY

CLOSE

Note that the actual scoring endpoint is located at the path `/score`. In addition, to prevent DDoS and other malicious activities, the resulting AWS lambda is protected by an API Key, i.e., a secret that has to be passed in as a part of the request using the `x-api-key` HTTP header.

The request is a JSON object containing attributes:

- **fields:** A list of input column names that should correspond to the training data columns.
- **rows:** A list of rows that are in turn lists of cell values to predict the target values for.

- optional **includeFieldsInOutput**: A list of input columns that should be included in the output.

An example request providing 2 columns on the input and asking to get one column copied to the output looks as follows:

```
{  
    "fields": [  
        "age", "salary"  
    ],  
    "includeFieldsInOutput": [  
        "salary"  
    ],  
    "rows": [  
        [  
            "48.0", "15000.0"  
        ],  
        [  
            "35.0", "35000.0"  
        ],  
        [  
            "18.0", "22000.0"  
        ]  
    ]  
}
```

Assuming the request is stored locally in a file named `test.json`, the request to the endpoint can be sent, e.g., using the `curl` utility, as follows:

```
URL=(place the endpoint URL here)  
API_KEY=(place the endpoint API key here)  
curl \  
  -d @test.json \  
  -X POST \  
  -H "x-api-key: ${API_KEY}" \  
  ${URL}/score
```

The response is a JSON object with a single attribute `score`, which contains the list of rows with the optional copied input values and the predictions.

For the example above with a two class target field, the result is likely to look something like the following snippet. The particular values would of course depend on the scoring pipeline:

```
{  
    "score": [  
        [  
            "48.0",  
            "0.6240277982943945",  
            "0.045458571508101536",  
        ],  
        [  
            "35.0",  
            "0.7209441819603676",  
            "0.06299909138586585",  
        ],  
        [  
            "18.0",  
            "0.7209441819603676",  
            "0.06299909138586585",  
        ]  
    ]  
}
```

38.3.6 AWS Deployment Issues

We create a new S3 bucket per AWS Lambda deployment. The bucket names have to be unique throughout AWS S3, and one user can create a maximum of 100 buckets. Therefore, we recommend setting the bucket name used for deployment with the `deployment_aws_bucket_name` config option.

38.4 REST Server Deployment

This section describes how to deploy the trained MOJO scoring pipeline as a local Representational State Transfer (REST) Server.

38.4.1 Additional Resources

The REST server deployment supports API endpoints such as model metadata, file/CSV scoring, etc. It uses SpringFox for both programmatic and manual inspection of the API. Refer to the `local-rest-scorer` folder in the `dai-deployment-templates` repository to see different deployment templates for Local REST scorers.

38.4.2 Prerequisites

- Driverless AI MOJO Scoring Pipeline: To deploy a MOJO scoring pipeline as a Local REST Scorer, the MOJO pipeline archive has to be created first by choosing the **Build MOJO Scoring Pipeline** option on the completed experiment page. Refer to the [MOJO Scoring Pipelines](#) section for information on how to build a MOJO scoring pipeline.
- When using a firewall or a virtual private cloud (VPC), the ports that are used by the REST server must be exposed.
- Ensure that you have enough memory and CPUs to run the REST scorer. Typically, a good estimation for the amount of required memory is 12 times the size of the pipeline.mojo file. For example, a 100MB pipeline.mojo file will require approximately 1200MB of RAM. (**Note:** To conveniently view in-depth information about your system in Driverless AI, click on **Resources** at the top of the screen, then click **System Info**.)
- When running Driverless AI in a Docker container, you must expose ports on Docker for the REST service deployment within the Driverless AI Docker container. For example, the following exposes the Driverless AI Docker container to listen to port 8094 for requests arriving at the host port at 18094.

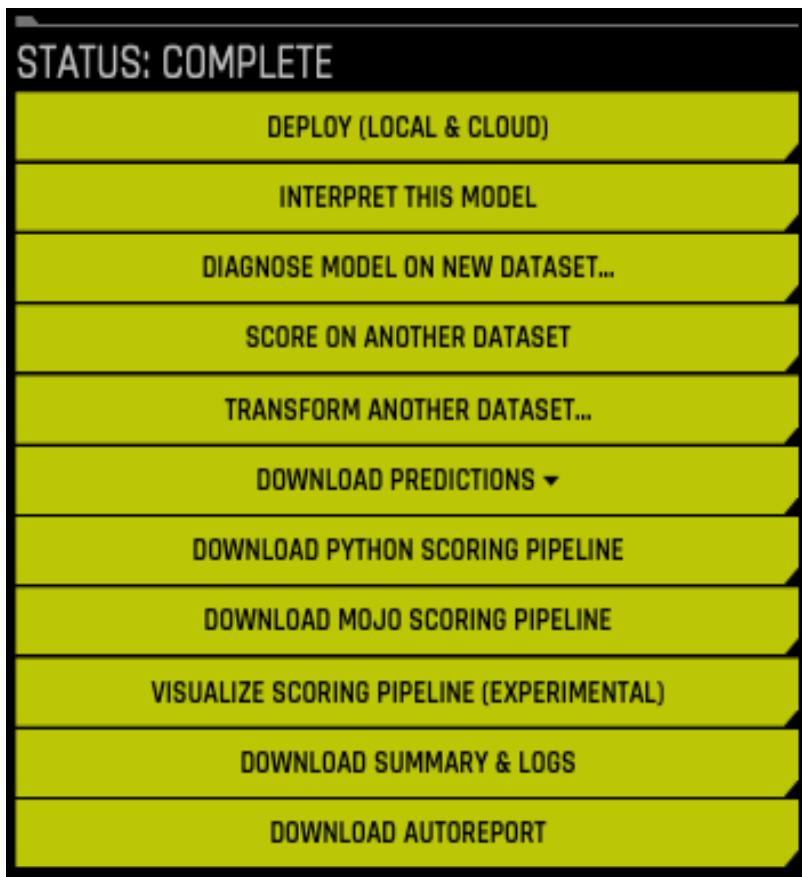
```
docker run \
-d \
--pid=host \
--init \
--rm \
--shm-size=256m \
-u `id -u`: `id -g` \
-p 12181:12345 \
-p 18094:8094 \
-v `pwd`:/data:/data \
-v `pwd`:/log:/log \
-v `pwd`:/license:/license \
-v `pwd`:/tmp:/tmp \
h2oai/<dai-image-name>:1.9.0-cuda10.0.23
```

38.4.3 Deploying on REST Server

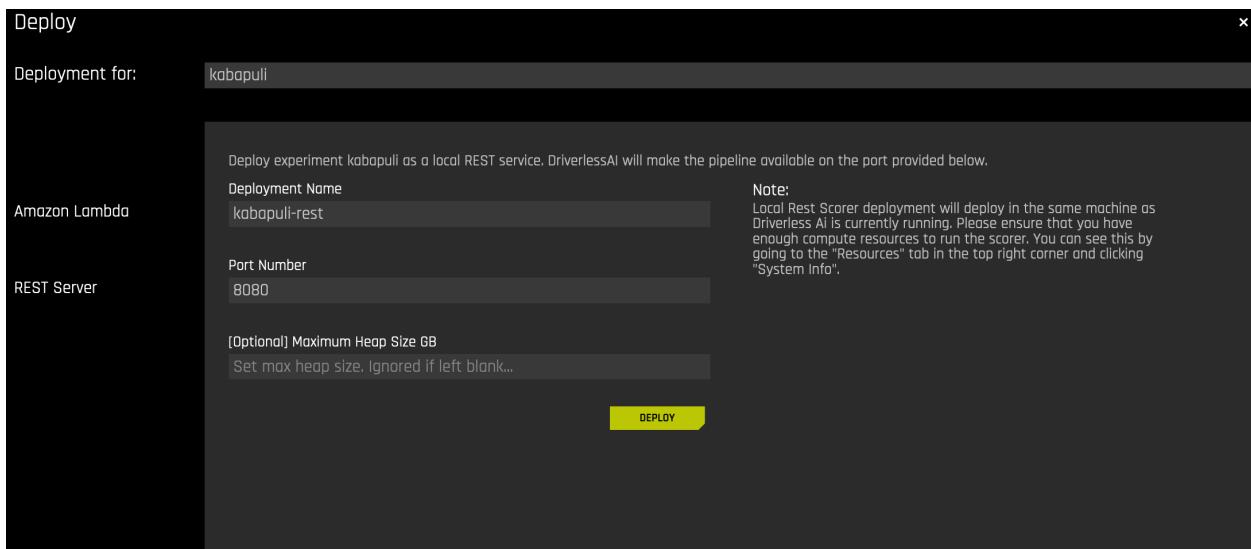
Once the MOJO pipeline archive is ready, Driverless AI provides a **Deploy (Local & Cloud)** option on the completed experiment page.

Notes:

- This button is only available after the MOJO Scoring Pipeline has been built.
- This button is not available on PPC64LE environments.



This option opens a new dialog for setting the REST Server deployment name, port number, and maximum heap size (optional).



1. Specify a name for the REST scorer in order to help track the deployed REST scorers.
2. Provide a port number on which the REST scorer will run. For example, if port number 8081 is selected, the scorer will be available at <http://my-ip-address:8081/models>
3. Optionally specify the maximum heap size for the Java Virtual Machine (JVM) running the REST scorer. This can help constrain the REST scorer from overconsuming memory of the machine. Because the REST scorer is running on the same machine as Driverless AI, it may be helpful to limit the amount of memory that is allocated to the REST scorer. This option will limit the amount of memory the REST scorer can use, but it will also produce an error if the memory allocated is not enough to run the scorer. (The amount of memory required is mostly dependent on the size of MOJO. See **Prerequisites** for more information.)

38.4.4 Testing the REST Server Deployment

Hooray

Your model has been deployed. Your endpoint info:

Model ID: 3b211958-b573-11e9-b073-0242ac110002

Location: <http://localhost:8080/models>

Copy and test it:

```
curl \
-X POST \
-d '{"fields": ["LIMIT_BAL", "EDUCATION", "MARRIAGE", "AGE", "PAY_0", "PAY_2", "PAY_3", "PAY_4",
"PAY_5", "PAY_6", "BILL_AMT1", "BILL_AMT2", "BILL_AMT3", "BILL_AMT4", "BILL_AMT5", "BILL_AMT6",
"PAY_AMT1", "PAY_AMT2", "PAY_AMT3", "PAY_AMT4", "PAY_AMT5", "PAY_AMT6", "default payment next"]}
```

COPY

You have to provide your own json data! For more info see help pages [here](#).

CLOSE

Note that the actual scoring endpoint is located at the path /score.

The request is a JSON object containing attributes:

- **fields:** A list of input column names that should correspond to the training data columns.
- **rows:** A list of rows that are in turn lists of cell values to predict the target values for.
- optional **includeFieldsInOutput:** A list of input columns that should be included in the output.

An example request providing 2 columns on the input and asking to get one column copied to the output looks as follows:

```
{
  "fields": [
    "age", "salary"
  ],
  "includeFieldsInOutput": [
    "salary"
  ],
  "rows": [
    [
      "48.0", "15000.0"
    ],
    [
      "35.0", "35000.0"
    ],
    [
      "18.0", "22000.0"
    ]
  ]
}
```

Assuming the request is stored locally in a file named test.json, the request to the endpoint can be sent, e.g., using the curl utility, as follows:

```
URL=(place the endpoint URL here)
curl \
-X POST \
-d {"fields": ["age", "salary", "education"], "rows": [1, 2, 3], "includeFieldsInOutput": ["education"]} \
-H "Content-Type: application/json" \
$URL/score
```

The response is a JSON object with a single attribute `score`, which contains the list of rows with the optional copied input values and the predictions.

For the example above with a two class target field, the result is likely to look something like the following snippet. The particular values would of course depend on the scoring pipeline:

```
{
  "score": [
    [
      "48.0",
      "0.6240277982943945",
      "0.045458571508101536",
    ],
    [
      "35.0",
      "0.7209441819603676",
      "0.06299909138586585",
    ],
    [
      "18.0",
      "0.7209441819603676",
      "0.06299909138586585",
    ]
  ]
}
```

38.4.5 REST Server Deployment Issues

- Local REST server deployments are useful for determining the behavioral characteristics of a MOJO that is intended for deployment. However, we don't recommend using the REST Server deployment as a production level scoring service. The REST Server deployment runs in the same machine as the core of Driverless AI, and therefore has to share system resources with all other Driverless AI processes. This can lead to unexpected scenarios in which competition for compute resources causes the REST Server to fail.

Additionally, given that the memory and CPU resource requirements for the REST scorer and for scoring the MOJO are typically smaller than the requirements for training, it is typically much more cost effective to deploy the MOJO in a smaller instance/machine. Templates can be easily found/built from the repository: <https://github.com/h2oai/dai-deployment-templates>.

- The REST Server will be shut down if Driverless AI is restarted.

CHAPTER
THIRTYNINE

WHAT'S HAPPENING IN DRIVERLESS AI?

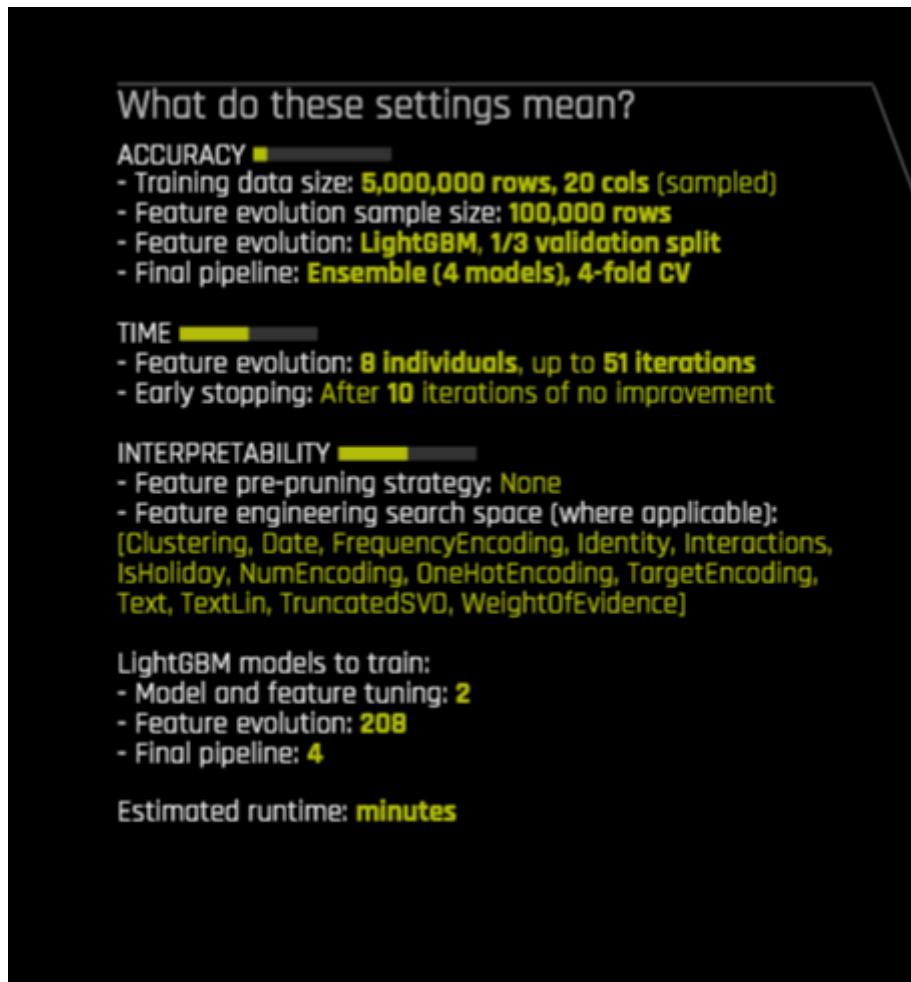
H2O Driverless AI is an automatic machine learning platform that uses feature engineering recipes from some of the world's best data scientists to deliver highly accurate machine learning models. As part of the automatic feature engineering process, the system uses a variety of transformers to enhance the available data. This section describes what's happening underneath the hood, including details about the feature engineering transformations and time series and natural language processing functionality.

Refer to one of the following topics:

- *Data Sampling*
- *Driverless AI Transformations*
- *Internal Validation Technique*
- *Missing and Unseen Levels Handling*
- *Imputation in Driverless AI*
- *Time Series in Driverless AI*
- *NLP in Driverless AI*
- *Image Processing in Driverless AI*

DATA SAMPLING

Driverless AI does not perform any type of data sampling unless the dataset is big or highly imbalanced (for improved accuracy). What is considered big is dependent on your accuracy setting and the `statistical_threshold_data_size_large` parameter in the config.toml or in the Expert Settings. You can see if the data will be sampled by viewing the Experiment Preview when you set up the experiment. In the experiment preview below, I can see that my data was sampled down to 5 million rows for the final model, and to 100k rows for the feature evolution part of the experiment.



If Driverless AI decides to sample the data based on these settings and the data size, then Driverless AI will perform the following types of sampling at the start of (and/or throughout) the experiment:

- Random sampling for regression problems
- Stratified sampling for classification problems
- Imbalanced sampling for binary problems where the target distribution is considered imbalanced and imbalanced sampling methods are enabled (`imbalance_sampling_method` not set to "off")

40.1 Imbalanced Sampling Methods

Imbalanced sampling techniques can help in binary classification use cases with highly imbalanced outcomes (churn, fraud, rare event modeling, etc.)

In Driverless AI, imbalanced sampling is an optional technique, implemented via a special custom recipe, called Imbalanced Model. There are two such Imbalanced Models: `ImbalancedLightGBMModel` and `ImbalancedXGBoostGBMModel`. Both perform repeated stratified sampling (bagging) inside their `fit()` method, in an attempt to speed up modeling and to improve the resolution of the decision boundary between the two classes. Since these model are presented a training dataset with a different prior than the original data, they require a probability correction that is performed as part of postprocessing in the `predict()` method.

When imbalanced sampling is enabled, no sampling is done at the start of the experiment, neither for the feature evolution part, nor for the final model pipeline. Instead, sampling (with replacement) is performed during model fitting, and the model is presented a more balanced target class distribution than the original data. Since the sample is usually much smaller than the original data, this process can be repeated many times and each internal model's prediction can be averaged to improve accuracy (bagging). By default, the number of bags is automatically determined, but can be specified in expert settings (`imbalance_sampling_number_of_bags=-1` means automatic). For "`over_under_sampling`", each bag can have a slightly different balance between minority and majority classes.

There are multiple settings for imbalanced sampling:

- Disabled (`imbalance_sampling_method="off"`, the default)
- Automatic (`imbalance_sampling_method="auto"`). A combination of the two methods below.
- Under- and over-sample both minority and majority classes to reach roughly class balance in each sampled bag (`imbalance_sampling_method="over_under_sampling"`). If original data has 500:10000 imbalance, this method could sample 1000:1500 samples for the first bag, 500:400 samples for the second bag, and so on.
- Under-sample the majority class to reach exact class balance in each sampled bag (`imbalance_sampling_method="under_sampling"`). Would create 500:500 samples per bag for the same example imbalance ratio . Each bag would then sample the 500 rows from each class with replacement, so each bag is still different.

The amount of imbalance controls how aggressively imbalanced models are used for the experiment (if `imbalance_sampling_method` is not "off"):

- By default, imbalanced is defined as when the majority class is 5 times more common than the minority class (`imbalance_ratio_sampling_threshold=5`, configurable). In this case, imbalanced models are added to the list of models for the experiment to select from.
- By default, heavily imbalanced is defined as when the majority class is 25 times more common than the minority class (`heavy_imbalance_ratio_sampling_threshold=25`, configurable). In highly imbalanced cases, imbalanced models are used exclusively.

Notes:

- The data has to be large enough to enable imbalanced sampling: by default, `imbalance_sampling_threshold_min_rows_original` is set to 100,000 rows.

- If `imbalance_sampling_number_of_bags=-1` (automatic) and `imbalance_sampling_method="auto"`, the number of bags will be automatically determined by the experiment's accuracy settings and by the total size of all bags together, controlled by `imbalance_sampling_max_multiple_data_size`, which defaults to 1. So all bags together will be no larger than 1x the original data by default. For an imbalance of 1:19, each balanced 1:1 sample would be as large as 10% of the data, so it would take up to 10 such 1:1 bags (or approximately 10 if the balance is different or slightly random) to reach that limit. That's still a bit much for feature evolution and could cost too much slowdown. That's why the other limit of 3 (by default) for feature evolution exists. Feel free to adjust to your preferences.
- If `imbalance_sampling_number_of_bags=-1` (automatic) and `imbalance_sampling_method="over_under_sampling"` or `"under_sampling"`, the number of bags will be equal to the experiment's accuracy settings (accuracy 7 will use 7 bags).
- The upper limit for the number of bags can be specified separately for feature evolution (`imbalance_sampling_max_number_of_bags_feature_evolution`) and globally (i.e., final model) set by (`imbalance_sampling_max_number_of_bags`) and both will be strictly enforced.
- Instead of balancing the target class distribution via default value of `imbalance_sampling_target_minority_fraction=-1` (same as setting it to 0.5), one can control the target fraction of the minority class. So if the data starts with a 1:1000 imbalance and you wish to model with a 1:9 imbalance, specify `imbalance_sampling_target_minority_fraction=0.1`.
- To force the number of bags to 15 for all models, set `imbalance_sampling_number_of_bags=imbalance_sampling` and `imbalance_sampling_method` to either “`over_under_sampling`” or `under_sampling`, based on your preferences.
- By default, Driverless AI will notify you if the imbalance ratio (majority to minority) is larger than 2, controlled by `imbalance_ratio_notification_threshold`.
- Each parameter mentioned above is described in detail in the config.toml file, and also displayed in the expert options (on mouse-over hover), and each actual Imbalanced Model displays its parameter in the experiment logs.

Summary:

There are many options to control imbalanced sampling in Driverless AI. In most cases, it suffices to set `imbalance_sampling_method="auto"` and to compare results with an experiment without imbalanced sampling techniques `imbalance_sampling_method="off"`. In many cases, however, imbalanced models do not perform better than regular “full” models (especially if the majority class examples are important for the outcome, so any amount of sampling would hurt). It is never a bad idea to try imbalanced sampling techniques though, especially if the dataset is extremely large and highly imbalanced.

DRIVERLESS AI TRANSFORMATIONS

Transformations in Driverless AI are applied to columns in the data. The transformers create the engineered features in experiments.

Driverless AI provides a number of transformers. The downloaded experiment logs include the transformations that were applied to your experiment.

Notes:

- You can include or exclude specific transformers in your Driverless AI environment using the `included_transformers` or `excluded_transformers` config options.
- You can control which transformers to use in individual experiments with the *Include Specific Transformers* Expert Setting.
- Transformed feature names are encoded as follows:

`<transformation/gene_details_id>_<transformation_name>:<orig>:<...>:<orig>.<extra>`

So in `32_NuToCatTE:BILL_AMT1:EDUCATION:MARRIAGE:SEX.0`, for example:

- `32_` is the transformation index for specific transformation parameters.
- `NuToCatTE` is the transformation type.
- `BILL_AMT1:EDUCATION:MARRIAGE:SEX` represent original features used.
- `0` represents the likelihood encoding for target[0] after grouping by switch and making out-of-fold estimates. For multiclass experiments this value will be > 0 . For binary experiments, this value will always be 0.

41.1 Available Transformers

The following transformers are available for classification (multiclass and binary) and regression experiments.

41.1.1 Numeric Transformers (Integer, Real, Binary)

- ClusterDist Transformer

The Cluster Distance Transformer clusters selected numeric columns and uses the distance to a specific cluster as a new feature.

- ClusterTE Transformer

The Cluster Target Encoding Transformer clusters selected numeric columns and calculates the mean of the response column for each cluster. The mean of the response is used as a new feature. Cross Validation is used to calculate mean response to prevent overfitting.

- Interactions Transformer

The Interactions Transformer adds, divides, multiplies, and subtracts two numeric columns in the data to create a new feature. This transformation uses a smart search to identify which feature pairs to transform. Only interactions that improve the baseline model score are kept.

- InteractionsSimple Transformer

The InteractionsSimple Transformer adds, divides, multiplies, and subtracts two numeric columns in the data to create a new feature. This transformation randomly selects pairs of features to transform.

- NumCatTE Transformer

The Numeric Categorical Target Encoding Transformer calculates the mean of the response column for several selected columns. If one of the selected columns is numeric, it is first converted to categorical by binning. The mean of the response column is used as a new feature. Cross Validation is used to calculate mean response to prevent overfitting.

- NumToCatTE Transformer

The Numeric to Categorical Target Encoding Transformer converts numeric columns to categoricals by binning and then calculates the mean of the response column for each group. The mean of the response for the bin is used as a new feature. Cross Validation is used to calculate mean response to prevent overfitting.

- NumToCatWoEMonotonic Transformer

The Numeric to Categorical Weight of Evidence Monotonic Transformer converts a numeric column to categorical by binning and then calculates Weight of Evidence for each bin. The monotonic constraint ensures the bins of values are monotonically related to the Weight of Evidence value. The Weight of Evidence is used as a new feature. Weight of Evidence measures the “strength” of a grouping for separating good and bad risk and is calculated by taking the log of the ratio of distributions for a binary response column.

- NumToCatWoE Transformer

The Numeric to Categorical Weight of Evidence Transformer converts a numeric column to categorical by binning and then calculates Weight of Evidence for each bin. The Weight of Evidence is used as a new feature. Weight of Evidence measures the “strength” of a grouping for separating good and bad risk and is calculated by taking the log of the ratio of distributions for a binary response column.

- Original Transformer

The Original Transformer applies an identity transformation to a numeric column.

- TruncSVDNum Transformer

Truncated SVD Transformer trains a Truncated SVD model on selected numeric columns and uses the components of the truncated SVD matrix as new features.

41.1.2 Time Series Experiments Transformers

- DateOriginal Transformer

The Date Original Transformer retrieves date values such as year, quarter, month, day, day of the year, week, and weekday values.

- DateTimeOriginal Transformer

The Date Time Original Transformer retrieves date *and* time values such as year, quarter, month, day, day of the year, week, weekday, hour, minute, and second values.

- EwmaLags Transformer

The Exponentially Weighted Moving Average (EWMA) Transformer calculates the exponentially weighted moving average of target or feature lags.

- LagsAggregates Transformer

The Lags Aggregates Transformer calculates aggregations of target/feature lags like mean(lag7, lag14, lag21) with support for mean, min, max, median, sum, skew, kurtosis, std. The aggregation is used as a new feature.

- LagsInteraction Transformer

The Lags Interaction Transformer creates target/feature lags and calculates interactions between the lags (lag2 - lag1, for instance). The interaction is used as a new feature.

- Lags Transformer

The Lags Transformer creates target/feature lags, possibly over groups. Each lag is used as a new feature. Lag transformers may apply to categorical (strings) features or binary/multiclass string valued targets after they have been internally numerically encoded.

- LinearLagsRegression Transformer

The Linear Lags Regression transformer trains a linear model on the target or feature lags to predict the current target or feature value. The linear model prediction is used as a new feature.

41.1.3 Categorical Transformers (String)

- Cat Transformer

The Cat Transformer sorts a categorical column in lexicographical order and uses the order index created as a new feature. This transformer works with models that can handle categorical features.

- CatOriginal Transformer

The Categorical Original Transformer applies an identity transformation that leaves categorical features as they are. This transformer works with models that can handle non-numeric feature values.

- CVCatNumEncode Transformer

The Cross Validation Categorical to Numeric Encoding Transformer calculates an aggregation of a numeric column for each value in a categorical column (ex: calculate the mean Temperature for each City) and uses this aggregation as a new feature.

- CVTargetEncode Transformer

The Cross Validation Target Encoding Transformer calculates the mean of the response column for each value in a categorical column and uses this as a new feature. Cross Validation is used to calculate mean response to prevent overfitting.

- Frequent Transformer

The Frequent Transformer calculates the frequency for each value in categorical column(s) and uses this as a new feature. This count can be either the raw count or the normalized count.

- LexiLabelEncoder Transformer

The Lexi Label Encoder sorts a categorical column in lexicographical order and uses the order index created as a new feature.

- NumCatTE Transformer

The Numeric Categorical Target Encoding Transformer calculates the mean of the response column for several selected columns. If one of the selected columns is numeric, it is first converted to categorical by binning. The mean of the response column is used as a new feature. Cross Validation is used to calculate mean response to prevent overfitting.

- OneHotEncoding Transformer

The One-hot Encoding transformer converts a categorical column to a series of boolean features by performing one-hot encoding. The boolean features are used as new features. If there are more than a specific number of unique values in the column, then they will be binned to the max number (10 by default) in lexicographical order. This value can be changed with the `ohe_bin_list config.toml` configuration option.

- SortedLE Transformer

The Sorted Label Encoding Transformer sorts a categorical column by the response column and uses the order index created as a new feature.

- WeightOfEvidence Transformer

The Weight of Evidence Transformer calculates Weight of Evidence for each value in categorical column(s). The Weight of Evidence is used as a new feature. Weight of Evidence measures the “strength” of a grouping for separating good and bad risk and is calculated by taking the log of the ratio of distributions for a binary response column.

$$WOE = \ln \left(\frac{\text{Distribution of Goods}}{\text{Distribution of Bads}} \right)$$

This only works with a binary target variable. The likelihood needs to be created within a stratified kfold if a `fit_transform` method is used. More information can be found here: <http://ucanalytics.com/blogs/information-value-and-weight-of-evidencebanking-case/>.

41.1.4 Text Transformers (String)

- BERT Transformer

The Bidirectional Encoder Representations from Transformers (BERT) Transformer creates new features for each text column based on the pre-trained model embeddings and is ideally suited for datasets that contain additional important non-text features.

Note: If your dataset is large or contains many text columns, then using the BERT transformer may significantly increase the time it takes for your experiment to complete.

- TextBiGRU Transformer

The Text Bidirectional GRU Transformer trains a bi-directional GRU TensorFlow model on word embeddings created from a text feature to predict the response column. The GRU prediction is used as a new feature. Cross Validation is used when training the GRU model to prevent overfitting.

- TextCharCNN Transformer

The Text Character CNN Transformer trains a CNN TensorFlow model on character embeddings created from a text feature to predict the response column. The CNN prediction is used as a new feature. Cross Validation is used when training the CNN model to prevent overfitting.

- TextCNN Transformer

The Text CNN Transformer trains a CNN TensorFlow model on word embeddings created from a text feature to predict the response column. The CNN prediction is used as a new a feature. Cross Validation is used when training the CNN model to prevent overfitting.

- TextLinModel Transformer

The Text Linear Model Transformer trains a linear model on a TF-IDF matrix created from a text feature to predict the response column. The linear model prediction is used as a new feature. Cross Validation is used when training the linear model to prevent overfitting.

- Text Transformer

The Text Transformer tokenizes a text column and creates a TFIDF matrix (term frequency-inverse document frequency) or count (count of the word) matrix. This may be followed by dimensionality reduction using truncated SVD. Selected components of the TF-IDF/Count matrix are used as new features.

41.1.5 Time Transformers (Date, Time)

- Dates Transformer

The Dates Transformer retrieves any date values, including:

- Year
- Quarter
- Month
- Day
- Day of year
- Week
- Week day
- Hour
- Minute
- Second

- IsHoliday Transformer

The Is Holiday Transformer determines if a date column is a holiday. A boolean column indicating if the date is a holiday is added as a new feature. Creates a separate feature for holidays in the United States, United Kingdom, Germany, Mexico, and the European Central Bank. Other countries available in the python Holiday package can be added via the configuration file.

41.1.6 Image Transformers

- ImageOriginal Transformer

The Image Original Transformer passes image paths to the model without performing any feature engineering.

- ImageVectorizer Transformer

The Image Vectorizer Transformer uses pre-trained [ImageNet](#) models to convert a column with an image path or URI to an embeddings (vector) representation that is derived from the last global average pooling layer of the model.

Note: Fine-tuning of the pre-trained image models can be enabled with the [*Enable Fine-Tuning of the Pretrained Image Models Used for the Image Vectorizer Transformers*](#) expert setting.

41.2 Example Transformations

In this section, we will describe some of the available transformations using the example of predicting house prices on the example dataset.

Date Built	Square Footage	Num Beds	Num Baths	State	Price
01/01/1920	1700	3	2	NY	\$700K

41.2.1 Frequent Transformer

- the count of each categorical value in the dataset
- the count can be either the raw count or the normalized count

Date Built	Square Footage	Num Beds	Num Baths	State	Price	Freq_State
01/01/1920	1700	3	2	NY	700,000	4,500

There are 4,500 properties in this dataset with state = NY.

41.2.2 Bulk Interactions Transformer

- add, divide, multiply, and subtract two columns in the data

Date Built	Square Footage	Num Beds	Num Baths	State	Price	Interaction_NumBeds#subtract#NumBaths
01/01/1920	1700	3	2	NY	700,000	1

There is one more bedroom than there are number of bathrooms for this property.

41.2.3 Truncated SVD Numeric Transformer

- truncated SVD trained on selected numeric columns of the data
- the components of the truncated SVD will be new features

Date Built	Square Footage	Num Beds	Num Baths	State	Price	TruncSVD_Price_NumBeds_NumBaths_1
01/01/1920	1700	3	2	NY	700,000	0.632

The first component of the truncated SVD of the columns Price, Number of Beds, Number of Baths.

41.2.4 Dates Transformer

- get year, get quarter, get month, get day, get day of year, get week, get week day, get hour, get minute, get second

Date Built	Square Footage	Num Beds	Num Baths	State	Price	DateBuilt_Month
01/01/1920	1700	3	2	NY	700,000	1

The home was built in the month January.

41.2.5 Text Transformer

- transform text column using methods: TFIDF or count (count of the word)
- this may be followed by dimensionality reduction using truncated SVD

41.2.6 Categorical Target Encoding Transformer

- cross validation target encoding done on a categorical column

Date Built	Square Footage	Num Beds	Num Baths	State	Price	CV_TE_State
01/01/1920	1700	3	2	NY	700,000	550,000

The average price of properties in NY state is \$550,000*.

*In order to prevent overfitting, Driverless AI calculates this average on out-of-fold data using cross validation.

41.2.7 Numeric to Categorical Target Encoding Transformer

- numeric column converted to categorical by binning
- cross validation target encoding done on the binned numeric column

Date Built	Square Footage	Num Beds	Num Baths	State	Price	CV_TE_SquareFootage
01/01/1920	1700	3	2	NY	700,000	345,000

The column Square Footage has been bucketed into 10 equally populated bins. This property lies in the Square Footage bucket 1,572 to 1,749. The average price of properties with this range of square footage is \$345,000*.

*In order to prevent overfitting, Driverless AI calculates this average on out-of-fold data using cross validation.

41.2.8 Cluster Target Encoding Transformer

- selected columns in the data are clustered
- target encoding is done on the cluster ID

Date Built	Square Footage	Num Beds	Num Baths	State	Price	ClusterTE_4_NumBeds_NumBaths_SquareFootage
01/01/1920	1700	3	2	NY	700,000	450,000

The columns: Num Beds, Num Baths, Square Footage have been segmented into 4 clusters. The average price of properties in the same cluster as the selected property is \$450,000*.

*In order to prevent overfitting, Driverless AI calculates this average on out-of-fold data using cross validation.

41.2.9 Cluster Distance Transformer

- selected columns in the data are clustered
- the distance to a chosen cluster center is calculated

Date Built	Square Footage	Num Beds	Num Baths	State	Price	ClusterDist_4_NumBeds_NumBaths_SquareFootage_1
01/01/1920	1700	3	2	NY	700,000	0.83

The columns: Num Beds, Num Baths, Square Footage have been segmented into 4 clusters. The difference from this record to Cluster 1 is 0.83.

INTERNAL VALIDATION TECHNIQUE

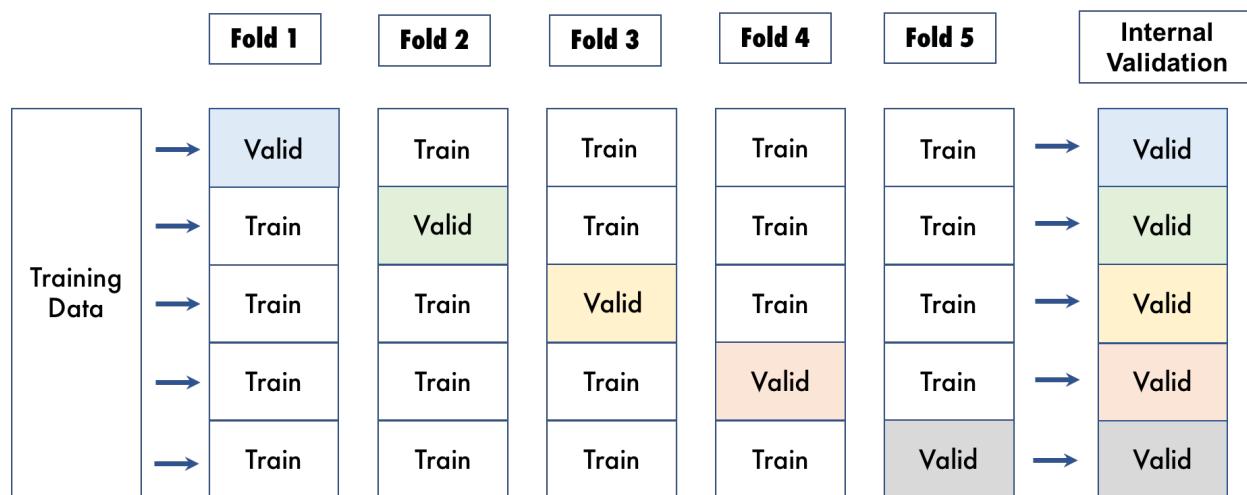
This section describes the technique behind internal validation in Driverless AI.

For the experiment, Driverless AI will either:

- (1) split the data into a training set and internal validation set
or
- (2) use cross validation to split the data into n folds

Driverless AI chooses the method based on the size of the data and the Accuracy setting. For method 1, part of the data is removed to be used for internal validation. (**Note:** This train and internal validation split may be repeated if the data is small so that more data can be used for training.)

For method 2, however, no data is wasted for internal validation. With cross validation, the whole dataset is utilized, and each model is trained on a different subset of the training data. The following visualization shows an example of cross validation with 5 folds.



Driverless AI randomly splits the data into the specified number of folds for cross validation. With cross validation, the whole dataset is utilized, and each model is trained on a different subset of the training data.

Driverless AI will not automatically create the internal validation data randomly if a user provides a Fold Column or a Validation Dataset. If a Fold Column or a Validation Dataset is provided, Driverless AI will use that data to calculate the performance of the Driverless AI models and to calculate all performance graphs and statistics.

If the experiment is a Time Series use case, and a Time Column is selected, Driverless AI will change the way the internal validation data is created. In the case of temporal data, it is important to train on historical data and validate on more recent data. Driverless AI does not perform random splits, but instead respects the temporal nature of the data

to prevent any data leakage. In addition, the train/validation split is a function of the time gap between train and test as well as the forecast horizon (amount of time periods to predict). If test data is provided, Driverless AI will suggest values for these parameters that lead to a validation set that resembles the test set as much as possible. But users can control the creation of the validation split in order to adjust it to the actual application.

MISSING AND UNSEEN LEVELS HANDLING

This section describes how missing and unseen levels are handled by each algorithm during training and scoring.

43.1 How Does the Algorithm Handle Missing Values During Training?

43.1.1 LightGBM, XGBoost, RuleFit

Driverless AI treats missing values natively. (I.e., a missing value is treated as a special value.) Experiments rarely benefit from imputation techniques, unless the user has a strong understanding of the data.

43.1.2 GLM

Driverless AI automatically performs mean value imputation (equivalent to setting the value to zero after standardization).

43.1.3 TensorFlow

Driverless AI provides an imputation setting for TensorFlow in the config.toml file: `tf_nan_impute_value` (post-normalization). If you set this option to 0, then missing values will be imputed by the mean. Setting it to (for example) +5 will specify 5 standard deviations above the mean of the distribution. The default value in Driverless AI is -5, which specifies that TensorFlow will treat missing values as outliers on the negative end of the spectrum. Specify 0 if you prefer mean imputation.

43.1.4 FTRL

In FTRL, missing values have their own representation for each datable column type. These representations are used to hash the missing value, with their column's name, to an integer. This means FTRL replaces missing values with special constants that are the same for each column type, and then treats these special constants like a normal data value.

43.2 How Does the Algorithm Handle Missing Values During Scoring (Production)?

43.2.1 LightGBM, XGBoost, RuleFit

If missing data is present during training, these tree-based algorithms learn the optimal direction for missing data for each split (left or right). This optimal direction is then used for missing values during scoring. If no missing data is present during scoring (for a particular feature), then the majority path is followed if the value is missing.

43.2.2 GLM

Missing values are replaced by the mean value (from training), same as in training.

43.2.3 TensorFlow

Missing values are replaced by the same value as specified during training (parameterized by `tf_nan_impute_value`).

43.2.4 FTRL

To ensure consistency, FTRL treats missing values during scoring in exactly the same way as during training.

43.2.5 Clustering in Transformers

Missing values are replaced with the mean along each column. This is used only on numeric columns.

43.2.6 Isolation Forest Anomaly Score Transformer

Isolation Forest uses out-of-range imputation that fills missing values with the values beyond the maximum.

43.3 What Happens When You Try to Predict on a Categorical Level Not Seen During Training?

43.3.1 XGBoost, LightGBM, RuleFit, TensorFlow, GLM

Driverless AI's feature engineering pipeline will compute a numeric value for every categorical level present in the data, whether it's a previously seen value or not. For frequency encoding, unseen levels will be replaced by 0. For target encoding, the global mean of the target value will be used.

43.3.2 FTRL

FTRL models don't distinguish between categorical and numeric values. Whether or not FTRL saw a particular value during training, it will hash all the data, row by row, to numeric and then make predictions. Because you can think of FTRL as learning all the possible values in the dataset by heart, there is no guarantee it will make accurate predictions for unseen data. Therefore, it is important to ensure that the training dataset has a reasonable "overlap" in terms of unique values with the ones used to make predictions.

43.4 What Happens if the Response Has Missing Values?

All algorithms will skip an observation (aka record) if the response value is missing.

IMPUTATION IN DRIVERLESS AI

The impute feature allows you to fill in missing values with substituted values. Missing values can be imputed based on the column's mean, median, minimum, maximum, or mode value. You can also impute based on a specific percentile or by a constant value.

The imputation is precomputed on all data or inside the pipeline (based on what's in the train split).

The following guidelines should be followed when performing imputation:

- For constant imputation on numeric columns, constant must be numeric.
- For constant imputation on string columns, constant must be a string.
- For percentile imputation, the percentage value must be between 0 and 100.

Notes:

- This feature is experimental.
- Time columns cannot be imputed.

44.1 Enabling Imputation

Imputation is disabled by default. It can be enabled by setting `enable_imputation=true` in the `config.toml` (for native installs) or via the `DRIVERLESS_AI_ENABLE_IMPUTATION=true` environment variable (Docker image installs). This enables imputation functionality in transformers.

44.2 Running an Experiment with Imputation

Once imputation is enabled, you will have the option when running an experiment to add imputation columns.

1. Click on **Columns Imputation** in the Experiment Setup page.

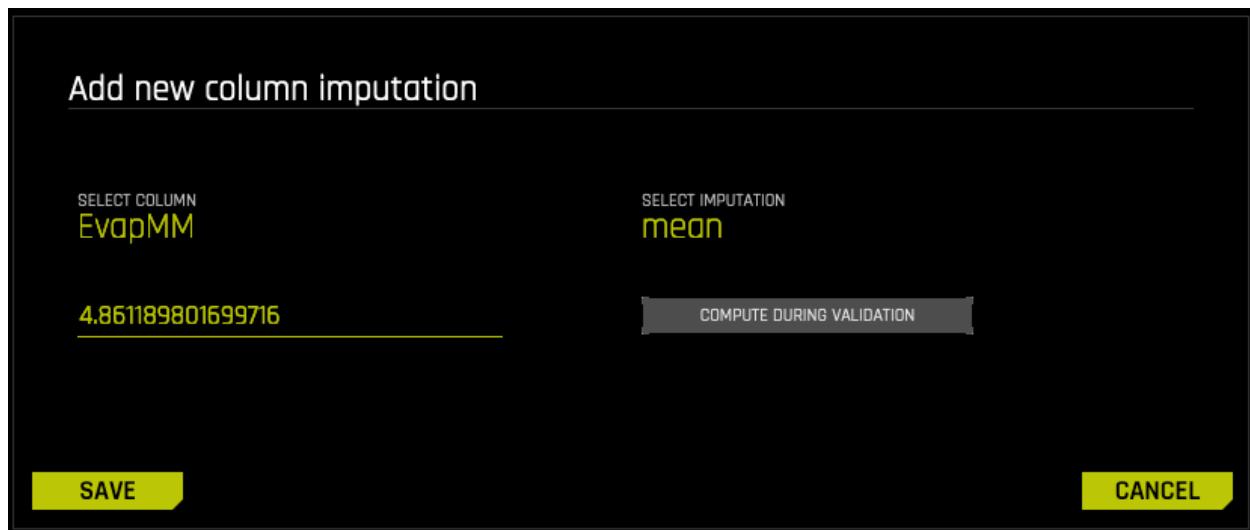
The screenshot shows the H2O.ai Experiment interface. At the top, there's a navigation bar with links to Projects, Datasets, AutoViz, Experiments, and Diagnostics. Below the navigation bar, the main area is divided into several sections:

- EXPERIMENT SETUP:** Displays dataset information: Display name (weather.csv), Rows (355), Columns (24), Dropped columns (--), Validation dataset (Yes, weather_valid), Test dataset (Yes, weather_test). It also shows the Target column (Sunshine) and Fold column (Disabled).
- WEIGHT COLUMN:** Shows a dropdown menu with options like "None", "Poisson", "Gamma", etc.
- TIME COLUMN:** Shows a dropdown menu with options like "None", "Time", "TimeSqrt", etc.
- COLUMNS IMPUTATION:** A section with a dropdown menu and a "Disabled" button.
- TRAINING SETTINGS:** Includes four circular sliders for Accuracy (7), Time (2), Interpretability (8), and Scorer (RMSE). Below these are buttons for Regression, Reproducible, and GPU Enabled.
- EXPERT SETTINGS:** Buttons for Launch Experiment and Save.

On the left side of the interface, there's a sidebar with the title "What do these settings mean?". It contains several sections with detailed explanations:

- ACCURACY:** Training data size: 355 rows, 24 cols; Feature evolution: [DecisionTree, LightGBM, XGBoostGBM], external validation, 2 reps; Final pipeline: 1 model, external validation.
- TIME:** Feature evolution: 8 individuals, up to 48 iterations; Early stopping: After 5 iterations of no improvement.
- INTERPRETABILITY:** Feature pre-pruning strategy: Permutation Importance FS; Monotonicity constraints: enabled; Feature engineering search space: [CVCatNumEncode, CVTargetEncode, Dates, Frequent, Interactions, lSHoliday, NumCatTE, Original, Text].
- Model Tuning:** [DecisionTree, LightGBM, XGBoostGBM] models to train; Target transform tuning: 36; Model and feature tuning: 192; Feature evolution: 288; Final pipeline: 1.
- Runtime:** Estimated runtime: minutes. Auto-click Finish/Abort if not done in: 1 day/7 days.

2. Click on **Add Imputation** in the upper-right corner.
3. Select the column that contains missing values you want to impute.
4. Select the imputation type. Available options are:
 - mean: The column's numeric mean value displays by default. (Default method for numeric values.)
 - median: When selected, the column's numeric median value displays by default.
 - min: When selected, the column's numeric minimum value displays by default.
 - max: When selected, the column's numeric maximum value displays by default.
 - const: Enter a string of characters. (Default method for string columns)
 - mode: When selected, the column's numeric mode value displays by default.
 - percentile: Specify a percentile rank value between 0 and 100. (Defaults to 95.) In addition, specify a numeric imputed value.
5. Optionally allow Driverless AI to compute the imputation value during validation instead of using the inputted imputed value.
6. Click **Save** when you are done.



7. At this point, you can add additional imputations, delete the imputation you just created, or close this form and return to the experiment. Note that each column can have only a single imputation.

TIME SERIES IN DRIVERLESS AI

Time series forecasting is one of the most common and important tasks in business analytics. There are many real-world applications like sales, weather, stock market, and energy demand, just to name a few. At H2O, we believe that automation can help our users deliver business value in a timely manner. Therefore, we combined advanced time series analysis and our Kaggle Grand Masters' time series recipes into Driverless AI.

The key features/recipes that make automation possible are:

- Automatic handling of time groups (e.g., different stores and departments)
- Robust time series validation
 - Accounts for gaps and forecast horizon
 - Uses past information only (i.e., no data leakage)
- Time series-specific feature engineering recipes
 - Date features like day of week, day of month, etc.
 - AutoRegressive features, like optimal lag and lag-features interaction
 - Different types of exponentially weighted moving averages
 - Aggregation of past information (different time groups and time intervals)
 - Target transformations and differentiation
- Integration with existing feature engineering functions (recipes and optimization)
- Rolling-window based predictions for time series experiments with test-time augmentation or re-fit
- Automatic pipeline generation (See “From Kaggle Grand Masters’ Recipes to Production Ready in a Few Clicks” [blog post](#).)

45.1 Understanding Time Series

45.1.1 Modeling Approach

Driverless AI uses GBMs, GLMs and neural networks with a focus on time series-specific feature engineering. The feature engineering includes:

- Autoregressive elements: creating lag variables
- Aggregated features on lagged variables: moving averages, exponential smoothing descriptive statistics, correlations
- Date-specific features: week number, day of week, month, year

- Target transformations: Integration/Differentiation, univariate transforms (like logs, square roots)

This approach is combined with AutoDL features as part of the genetic algorithm. The selection is still based on validation accuracy. In other words, the same transformations/genes apply; plus there are new transformations that come from time series. Some transformations (like target encoding) are deactivated.

When running a time series experiment, Driverless AI builds multiple models by rolling the validation window back in time (and potentially using less and less training data).

45.1.2 User-Configurable Options

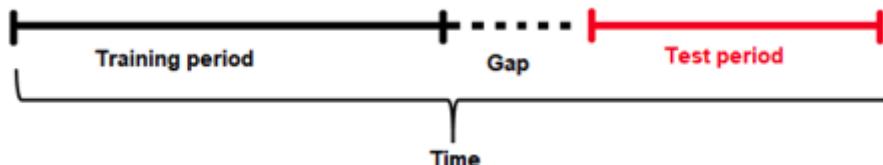
Gap

The guiding principle for properly modeling a time series forecasting problem is to use the historical data in the model training dataset such that it mimics the data/information environment at scoring time (i.e. deployed predictions). Specifically, you want to partition the training set to account for: 1) the information available to the model when making predictions and 2) the number of units out that the model should be optimized to predict.

Given a training dataset, the gap and forecast horizon are parameters that determine how to split the training dataset into training samples and validation samples.

Gap is the amount of missing time bins between the end of a training set and the start of test set (with regards to time). For example:

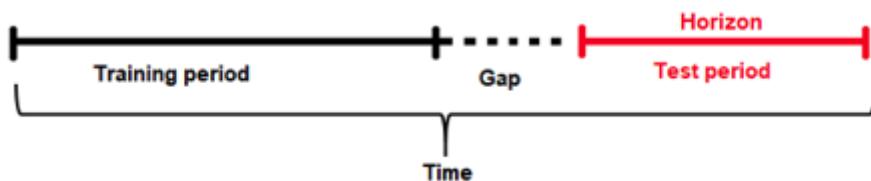
- Assume there are daily data with days 1/1/2020, 2/1/2020, 3/1/2020, 4/1/2020 in train. There are 4 days in total for training.
- In addition, the test data will start from 6/1/2020. There is only 1 day in the test data.
- The previous day (5/1/2020) does not belong to the train data. It is a day that cannot be used for training (i.e. because information from that day may not be available at scoring time). This day cannot be used to derive information (such as historical lags) for the test data either.
- Here the time bin (or time unit) is 1 day. This is the time interval that separates the different samples/rows in the data.
- In summary, there are 4 time bins/units for the train data and 1 time bin/unit for the test data plus the Gap.
- In order to estimate the Gap between the end of the train data and the beginning of the test data, the following formula is applied.
- $\text{Gap} = \min(\text{time bin test}) - \max(\text{time bin train}) - 1$.
- In this case $\min(\text{time bin test})$ is 6 (or 6/1/2020). This is the earliest (and only) day in the test data.
- $\max(\text{time bin train})$ is 4 (or 4/1/2020). This is the latest (or the most recent) day in the train data.
- Therefore the GAP is 1 time bin (or 1 day in this case), because $\text{Gap} = 6 - 4 - 1$ or $\text{Gap} = 1$



Forecast Horizon

Quite often, it is not possible to have the most recent data available when applying a model (or it is costly to update the data table too often); hence models need to be built accounting for a “future gap”. For example, if it takes a week to update a certain data table, ideally we would like to predict “7 days ahead” with the data as it is “today”; hence a gap of 7 days would be sensible. Not specifying a gap and predicting 7 days ahead with the data as it is is unrealistic (and cannot happen, as we update the data on a weekly basis in this example). Similarly, gap can be used for those who want to forecast further in advance. For example, users want to know what will happen 7 days in the future, they will set the gap to 7 days.

Forecast Horizon (or prediction length) is the period that the test data spans for (for example, one day, one week, etc.). In other words it is the future period that the model can make predictions for (or the number of units out that the model should be optimized to predict). Forecast horizon is used in feature selection and engineering and in model selection. Note that forecast horizon might not equal the number of predictions. The actual predictions are determined by the test dataset.



The periodicity of updating the data may require model predictions to account for significant time in the future. In an ideal world where data can be updated very quickly, predictions can always be made having the most recent data available. In this scenario there is no need for a model to be able to predict cases that are well into the future, but rather focus on maximizing its ability to predict short term. However this is not always the case, and a model needs to be able to make predictions that span deep into the future because it may be too costly to make predictions every single day after the data gets updated.

In addition, each future data point is not the same. For example, predicting tomorrow with today’s data is easier than predicting 2 days ahead with today’s data. Hence specifying the forecast horizon can facilitate building models that optimize prediction accuracy for these future time intervals.

Prediction Intervals

For regression problems, enable the *Compute Prediction Intervals* expert setting to have Driverless AI provide two additional columns *y.lower* and *y.upper* in the prediction frame. The true target value *y* for a predicted sample is expected to lie within $[y.lower, y.upper]$ with a certain probability. The default value for this confidence level can be specified with the *Confidence Level for Prediction Intervals* expert setting, which has a default value of 0.9.

Driverless AI uses holdout predictions to determine intervals empirically (Williams, W.H. and Goodman, M.L. “A Simple Method for the Construction of Empirical Confidence Limits for Economic Forecasts.” *Journal of the American Statistical Association*, 66, 752-754. 1971). This method makes no assumption about the underlying model or the distribution of error and has been shown to outperform many other approaches (Lee, Yun Shin and Scholtes, Stefan. “Empirical prediction intervals revisited.” *International Journal of Forecasting*, 30, 217-234. 2014).

Notes:

- This feature applies to regression tasks (i.i.d. and time series).
- This feature works with all model types.
- MOJO support is not currently implemented for this feature.
- Prediction intervals are computed for each individual time group.

time_period_in_seconds

Note: `time_period_in_seconds` is only available in the Python and R clients. Time period in seconds cannot be specified in the UI.

In Driverless AI, the forecast horizon (a.k.a., `num_prediction_periods`) needs to be in periods, and the size is unknown. To overcome this, you can use the optional `time_period_in_seconds` parameter when running `start_experiment_sync` (in Python) or `train` (in R). This is used to specify the forecast horizon in real time units (as well as for `gap`.) If this parameter is not specified, then Driverless AI will automatically detect the period size in the experiment, and the forecast horizon value will respect this period. I.e., if you are sure that your data has a 1 week period, you can say `num_prediction_periods=14`; otherwise it is possible that the model will not work correctly.

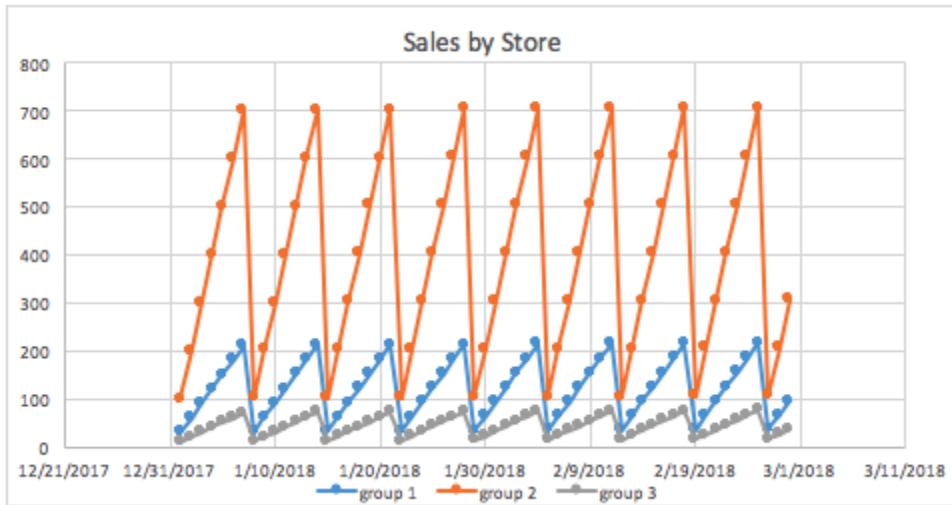
Groups

Groups are categorical columns in the data that can significantly help predict the target variable in time series problems. For example, one may need to predict sales given information about stores and products. Being able to identify that the combination of store and products can lead to very different sales is key for predicting the target variable, as a big store or a popular product will have higher sales than a small store and/or with unpopular products.

For example, if we don't know that the store is available in the data, and we try to see the distribution of sales along time (with all stores mixed together), it may look like that:

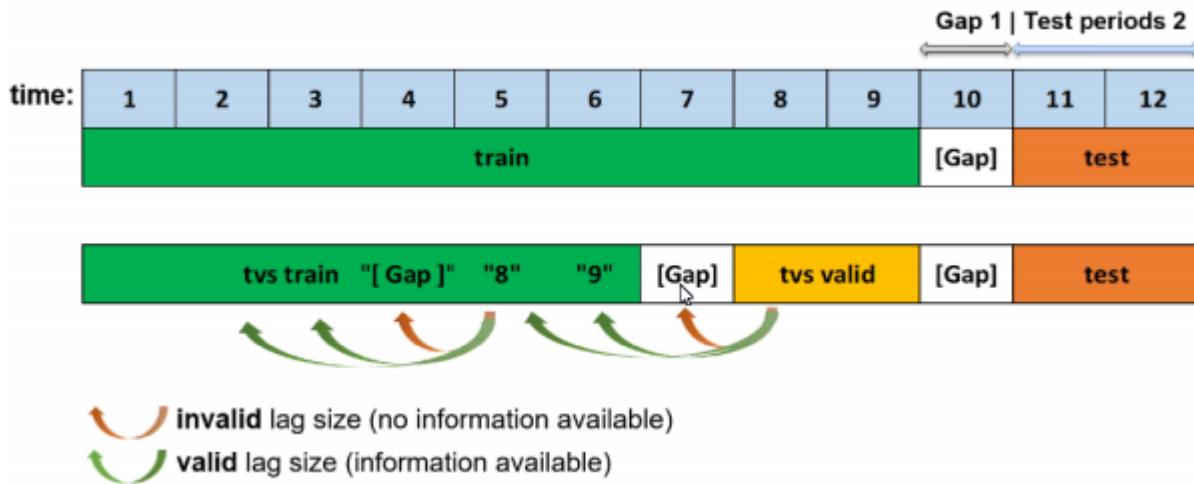


The same graph grouped by store gives a much clearer view of what the sales look like for different stores.



Lag

The primary generated time series features are lag features, which are a variable's past values. At a given sample with time stamp t , features at some time difference T (lag) in the past are considered. For example, if the sales today are 300, and sales of yesterday are 250, then the lag of one day for sales is 250. Lags can be created on any feature as well as on the target.



As previously noted, the training dataset is appropriately split such that the amount of validation data samples equals that of the testing dataset samples. If we want to determine valid lags, we must consider what happens when we will evaluate our model on the testing dataset. Essentially, the minimum lag size must be greater than the gap size.

Aside from the minimum useable lag, Driverless AI attempts to discover predictive lag sizes based on auto-correlation.

“Lagging” variables are important in time series because knowing what happened in different time periods in the past can greatly facilitate predictions for the future. Consider the following example to see the lag of 1 and 2 days:

Date	Sales	Lag1	Lag2
1/1/2020	100	-	-
2/1/2020	150	100	-
3/1/2020	160	150	100
4/1/2020	200	160	150
5/1/2020	210	200	160
6/1/2020	150	210	200
7/1/2020	160	150	210
8/1/2020	120	160	150
9/1/2020	80	120	160
10/1/2020	70	80	120

45.1.3 Settings Determined by Driverless AI

Window/Moving Average

Using the above Lag table, a moving average of 2 would constitute the average of Lag1 and Lag2:

Date	Sales	Lag1	Lag2	MA2
1/1/2020	100	-	-	-
2/1/2020	150	100	-	-
3/1/2020	160	150	100	125
4/1/2020	200	160	150	155
5/1/2020	210	200	160	180
6/1/2020	150	210	200	205
7/1/2020	160	150	210	180
8/1/2020	120	160	150	155
9/1/2020	80	120	160	140
10/1/2020	70	80	120	100

Aggregating multiple lags together (instead of just one) can facilitate stability for defining the target variable. It may include various lags values, for example lags [1-30] or lags [20-40] or lags [7-70 by 7].

Exponential Weighting

Exponential weighting is a form of weighted moving average where more recent values have higher weight than less recent values. That weight is exponentially decreased over time based on an **alpha** (α) (hyper) parameter (0,1), which is normally within the range of [0.9 - 0.99]. For example:

- Exponential Weight = $\alpha^{**}(\text{time})$
- If sales 1 day ago = 3.0 and 2 days ago = 4.5 and $\alpha=0.95$:
- Exp. smooth = $3.0*(0.95^{**1}) + 4.5*(0.95^{**2}) / ((0.95^{**1}) + (0.95^{**2})) = 3.73$ approx.

45.2 Rolling-Window-Based Predictions

Driverless AI supports rolling-window-based predictions for time series experiments with two options: [Test Time Augmentation \(TTA\)](#) or re-fit.

Both options are useful to assess the performance of the pipeline for predicting not just a single forecast horizon, but many in succession. TTA simulates the process where the model stays the same but the features are refreshed using newly available data. Re-fit simulates the process of re-fitting the entire pipeline (including the model) once new data is available.

This process is automated when the test set spans for a longer period than the forecast horizon and if the target values of the test set are known. If the user scores a test set that meets these conditions after the experiment is finished, rolling predictions with TTA will be applied. Re-fit, on the other hand, is only applicable for test sets provided during an experiment.

TTA is the default option and can be changed with the [Method to Create Rolling Test Set Predictions](#) expert setting.





45.3 Time Series Constraints

45.3.1 Dataset Size

Usually, the forecast horizon (prediction length) H equals the number of time periods in the testing data N_{TEST} (i.e. $N_{TEST} = H$). You want to have enough training data time periods N_{TRAIN} to score well on the testing dataset. At a minimum, the training dataset should contain at least three times as many time periods as the testing dataset (i.e. $N_{TRAIN} \geq 3N_{TEST}$). This allows for the training dataset to be split into a validation set with the same amount of time periods as the testing dataset while maintaining enough historical data for feature engineering.

45.4 Time Series Use Case: Sales Forecasting

Below is a typical example of sales forecasting based on the [Walmart competition on Kaggle](#). In order to frame it as a machine learning problem, we formulate the historical sales data and additional attributes as shown below:

Raw data

Forecasting for these groups		Time Unit	What we want to predict	Additional Attributes	
Store	Department	Date	Weekly Sales	Mark Down 1	Mark Down 2
1	1	2/5/10	\$24,924.50	-1	-1
1	2	2/5/10	\$50,605.27	-1	-1
1	3	2/5/10	\$13,740.12	-1	-1
1	4	2/5/10	\$39,954.04	-1	-1

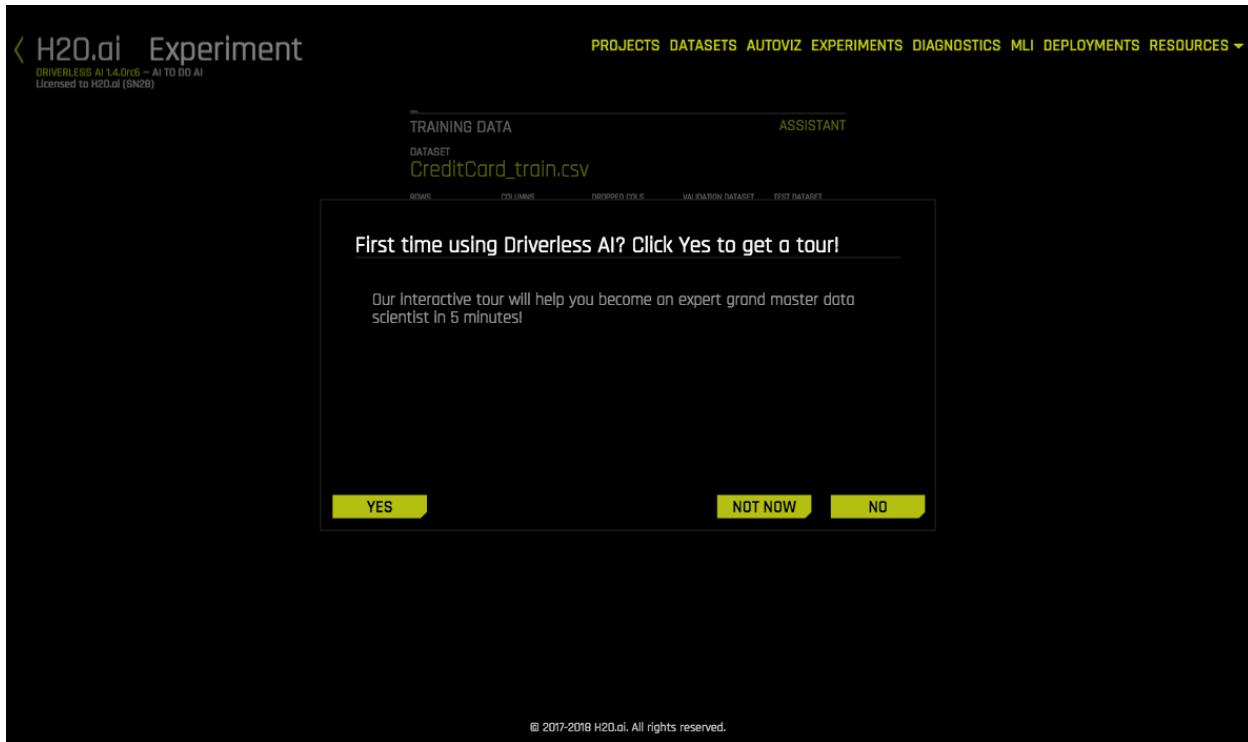
Data formulated for machine learning

Time Groups		Attributes					What we want to predict
Store	Department	Mark Down 1	Mark Down 2	Weekly Sales 2 Weeks Ago	Weekly Sales Last Week	Weekly Sales Next Week	
1	1	-1	-1	NA	\$24,924.50	\$46,039.49	
1	1	-1	-1	\$24,924.50	\$46,039.49	\$41,595.55	
1	1	-1	-1	\$46,039.49	\$41,595.55	\$19,403.54	
1	1	-1	-1	\$41,595.55	\$19,403.54	\$21,827.90	

The additional attributes are attributes that we will know at time of scoring. In this example, we want to forecast the next week of sales. Therefore, all of the attributes included in our data must be known at least one week in advance. In this case, we assume that we will know whether or not a Store and Department will be running a promotional markdown. We will not use features like the temperature of the Week since we will not have that information at the time of scoring.

Once you have your data prepared in tabular format (see raw data above), Driverless AI can formulate it for machine learning and sort out the rest. If this is your very first session, the Driverless AI assistant will guide you through the journey.

Using Driverless AI, Release 1.9.0.1



Similar to previous Driverless AI examples, you need to select the dataset for training/test and define the target. For time series, you need to define the time column (by choosing AUTO or selecting the date column manually). If weighted scoring is required (like the Walmart Kaggle competition), you can select the column with specific weights for different samples.

A screenshot of the H2O.ai Experiment interface showing the 'EXPERIMENT SETUP' tab. It includes sections for 'DISPLAY NAME' (set to 'walmart_train.csv'), 'ROWS' (422K), 'COLUMNS' (15), 'TARGET COLUMN' ('Weekly_Sales'), 'WEIGHT COLUMN' (disabled), 'TYPE' (real), 'COUNT' (421570), 'MEAN' (15981.258), and 'STDEV' (22711.184). In the 'TIME SERIES SETTINGS' section, 'TIME GROUPS COLUMNS' is set to '[AUTO]'. Other settings include 'UNAVAILABLE COLUMNS AT PREDICTION TIME' (1), 'FORECAST HORIZON' (39 weeks), 'GAP BETWEEN TRAIN/TEST PERIOD' (0 weeks), 'EST. FREQUENCY' (1.65μHz), 'EST. PERIOD' (1wk), 'TRAINSET TIMESPAN' (143 periods, 143 weeks), and 'TESTSET TIMESPAN' (39 periods, 39 weeks). There are also 'INTERPRETABILITY' and 'TRAINING SETTINGS' sections with various configuration options. A callout box highlights the 'What do these settings mean?' link, which points to detailed descriptions of accuracy, time, and interpretability metrics, as well as training settings like feature engineering search space and model tuning.

If you prefer to use automatic handling of time groups, you can leave the setting for time groups columns as AUTO, or you can define specific time groups. You can also specify the columns that will be unavailable at prediction time (see [More About Unavailable Columns at Time of Prediction](#) below for more information), the forecast horizon (in weeks), and the gap (in weeks) between the train and test periods.

Once the experiment is finished, you can make new predictions and download the scoring pipeline just like any other Driverless AI experiments.

45.4.1 More About Unavailable Columns at Time of Prediction

The **Unavailable Columns at Prediction Time** (UCAPT) option is a way to mark features that will not be available in the test dataset or at the time of prediction but might still be predictive when looking at historical values. These features will only be used in historical feature engineering recipes, such as Lagging or Exponential Weighted Moving Average.

For example, if we were predicting the sales amount each day, we might have the number of customers each day as a feature in our training dataset. In the future, we won't know how many customers will be coming into the store, so this would be a leaky feature to use. However, the average number of customers last week might be predictive and is something that we could calculate ahead of time. So in this case, looking at the historical values would be better than just dropping the feature.

The default value for this setting is often `--`, meaning that all features can be used as they are. If you include a test dataset before selecting a time column, and that test dataset is missing any columns, then you will see a number as the default for **Unavailable Columns at Prediction Time**, which will be the number of columns that are in the training dataset but not the testing dataset. All of these features will only be looked at historically, and you can see a list of them by clicking on this setting.

45.5 Using a Driverless AI Time Series Model to Forecast

When you set the experiment's forecast horizon, you are telling the Driverless AI experiment the dates this model will be asked to forecast for. In the Walmart Sales example, we set the Driverless AI forecast horizon to 1 (1 week in the future). This means that Driverless AI expects this model to be used to forecast 1 week after training ends. Since the training data ends on 2020-10-26, then this model should be used to score for the week of 2020-11-02.

What should the user do once the 2020-11-02 week has passed?

There are two options:

- Option 1: Trigger a Driverless AI experiment to be trained once the forecast horizon ends. A Driverless AI experiment will need to be re-trained every week.
- Option 2: Use **Test Time Augmentation** to update historical features so that we can use the same model to forecast outside of the forecast horizon.

Test Time Augmentation refers to the process where the model stays the same but the features are refreshed using the latest data. In our Walmart Sales Forecasting example, a feature that may be very important is the Weekly Sales from the previous week. Once we move outside of the forecast horizon, our model no longer knows the Weekly Sales from the previous week. By performing Test Time Augmentation, Driverless AI will automatically generate these historical features if new data is provided.

In Option 1, we would launch a new Driverless AI experiment every week with the latest data and use the resulting model to forecast the next week. In Option 2, we would continue using the same Driverless AI experiment outside of the forecast horizon by using Test Time Augmentation.

Both options have their advantages and disadvantages. By retraining an experiment with the latest data, Driverless AI has the ability to possibly improve the model by changing the features used, choosing a different algorithm, and/or selecting different parameters. As the data changes over time, for example, Driverless AI may find that the best algorithm for this use case has changed.

There may be clear advantages for retraining an experiment after each forecast horizon or for using Test Time Augmentation. Refer to [this example](#) to see how to use the scoring pipeline to predict future data instead of using the prediction endpoint on the Driverless AI server.

Using Test Time Augmentation to be able to continue using the same experiment over a longer period of time means there would be no need to continually repeat a model review process. The model may become out of date, however, and the MOJO scoring pipeline is not supported.

Scoring Supported	Retraining Model	Test Time Augmentation
Driverless AI Scoring	Supported	Supported
Python Scoring Pipeline	Supported	Supported
MOJO Scoring Pipeline	Supported	Not Supported

For different use cases, there may be clear advantages for retraining an experiment after each forecast horizon or for using Test Time Augmentation. In this notebook, we show how to perform both and compare the performance: [Time Series Model Rolling Window](#).

45.5.1 Triggering Test Time Augmentation

To perform Test Time Augmentation, create your forecast data to include any data that occurred after the training data ended up to the dates you want a forecast for. The dates that you want Driverless AI to forecast should have missing values (NAs) where the target column is. Target values for the remaining dates must be filled in.

The following is an example of forecasting for 2020-11-23 and 2020-11-30 with the remaining dates being used for TTA:

Date	Store	Dept	Mark Down 1	Mark Down 2	Weekly_Sales
2020-11-02	1	1	-1	-1	\$35,000
2020-11-09	1	1	-1	-1	\$40,000
2020-11-16	1	1	-1	-1	\$45,000
2020-11-23	1	1	-1	-1	NA
2020-11-30	1	1	-1	-1	NA

Notes:

- Although TTA can span any length of time into the future, the dates that are being predicted cannot exceed the horizon.
- If the date being forecasted contains any non-missing value in the target column, then TTA is not triggered for that row.

45.5.2 Forecasting Future Dates

To forecast or predict future dates, upload a dataset that contains the future dates of interest and provide additional information such as group IDs or features known in the future. The dataset can then be used to run and score your predictions.

The following is an example of a model that was trained up to 2020-05-31:

Date	Group_ID	Known_Feature_1	Known_Feature_2
2020-06-01	A	3	1
2020-06-02	A	2	2
2020-06-03	A	4	1
2020-06-01	B	3	0
2020-06-02	B	2	1
2020-06-03	B	4	0

45.6 Time Series Expert Settings

The user may further configure the time series experiments with a dedicated set of options available through the **Expert Settings** panel. This panel is available from within the experiment page right above the Scorer knob.

Refer to [Time Series Settings](#) for more information on these options.

The screenshot shows the 'Expert Experiment Settings' interface. At the top, there are three buttons: '+ UPLOAD CUSTOM RECIPE', '+ LOAD CUSTOM RECIPE FROM URL', and 'OFFICIAL RECIPES (EXTERNAL)'. On the right, there is a 'TOML' button. Below these are tabs: EXPERIMENT, MODEL, FEATURES, TIMESERIES, NLP, IMAGE, RECIPES, and SYSTEM. The TIMESERIES tab is selected. The interface contains several configuration sections:

- Time-series log-based recipe**: Includes a dropdown menu with 'ENABLED'.
- Time-series lags override, e.g. [7, 14, 21]**: Includes a dropdown menu with '[]'.
- Allowed date and date-time transformations**: Includes a dropdown menu with 'SELECT VALUES'.
- Always group by all time groups columns for creating lag features**: Includes a dropdown menu with 'ENABLED'.
- Maximum number of splits used for creating final time-series model's holdout predictions**: Includes a dropdown menu with '-1'.
- Lower limit on interpretability setting for**: Includes a dropdown menu with '0.5'.
- Custom validation splits for time-series experiments**: Includes a dropdown menu with '30'.
- Smallest considered log size (-1 = auto)**: Includes a dropdown menu with '-1'.
- Consider time groups columns as standalone features**: Includes a dropdown menu with 'DISABLED'.
- Generate Time-Series Holdout Predictions**: Includes a dropdown menu with 'ENABLED'.
- Whether to speed up calculation of Time-Series Holdout Predictions**: Includes a dropdown menu with 'DISABLED'.
- Timeout in seconds for time-series properties detection in UI**: Includes a dropdown menu with '30'.
- Enable feature engineering from time column**: Includes a dropdown menu with 'ENABLED'.
- Which tgc feature types to consider as standalone features**: Includes a dropdown menu with 'SELECT VALUES'.
- Number of time-based splits for internal model validation (-1 = auto)**: Includes a dropdown menu with '-1'.
- Whether to speed up calculation of Shapley values for Time-Series Holdout Predictions**: Includes a dropdown menu with 'ENABLED'.
- Probability to create non-target log**: Includes a dropdown menu with '0.5'.
- Generate Holiday Features**: Includes a dropdown menu with 'ENABLED'.
- Allow integer time column as numeric feature**: Includes a dropdown menu with 'DISABLED'.
- Enable time unaware transformers**: Includes a dropdown menu with 'AUTO'.
- Maximum overlap between two time-based splits**: Includes a dropdown menu with '0.5'.
- Generate Shapley values for Time-Series Holdout Predictions at the time of experiment**: Includes a dropdown menu with 'ENABLED'.

At the bottom left is a 'SAVE' button, and at the bottom right is a 'CANCEL' button.

45.7 Additional Resources

Refer to the following for examples showing how to run Time Series examples in Driverless AI:

- [Training a Time Series Model](#)
- [Time Series Recipes with Rolling Window](#)
- [Time Series Pipeline with Time Test Augmentation](#)

NLP IN DRIVERLESS AI

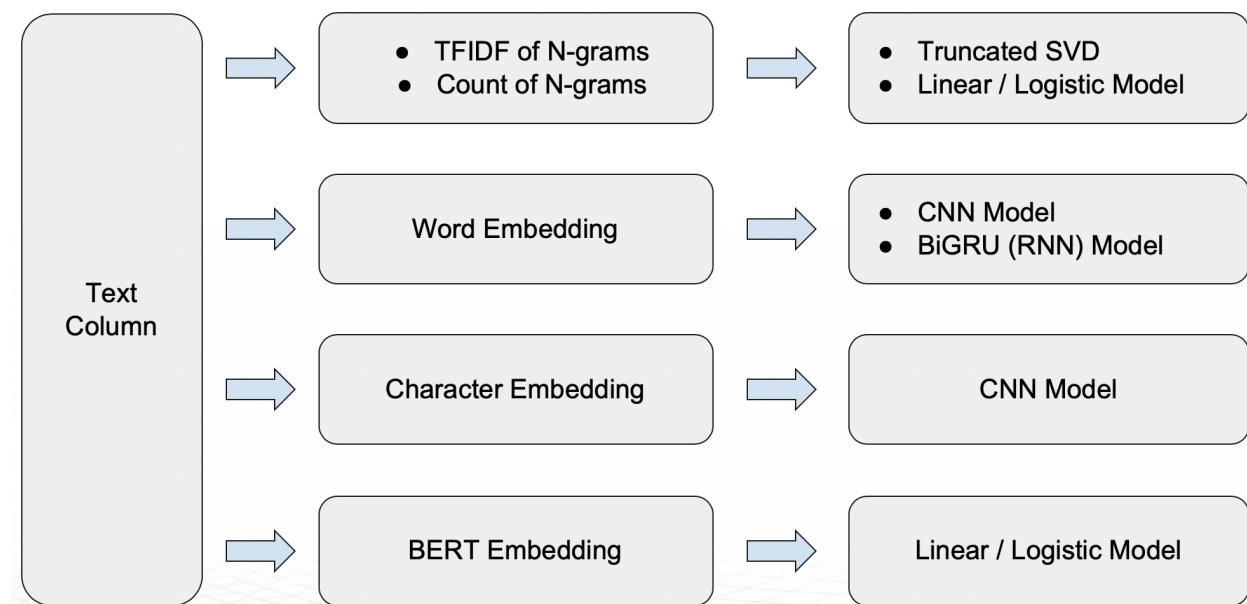
Driverless AI version 1.9 introduces support for PyTorch Transformer Architectures (for example, BERT) that can be used for Feature Engineering or as Modeling Algorithms. The Driverless AI platform has the ability to support both standalone text and text with other columns as predictive features.

The following NLP recipes are available for a given text column:

- n-gram frequency / TF-IDF followed by Truncated SVD
- n-gram frequency / TF-IDF followed by Linear / Logistic regression
- Word embeddings followed by CNN model (TensorFlow)
- Word embeddings followed by BiGRU model (TensorFlow)
- Character embeddings followed by CNN model (TensorFlow)
- BERT/DistilBERT based embeddings for Feature Engineering (PyTorch)
- Support for multiple Transformer Architectures (eg.BERT) as modeling algorithms (PyTorch)

In addition to these techniques, Driverless AI supports custom NLP recipes using, for example, PyTorch or Flair.

46.1 NLP Feature Engineering



46.2 n-gram

An **n-gram** is a contiguous sequence of n items from a given sample of text or speech.

46.2.1 n-gram Frequency

Frequency-based features represent the count of each word from a given text in the form of vectors. These are created for different n-gram values. For example, a one-gram is equivalent to a single word, a two-gram is equivalent to two consecutive words paired together, and so on.

Words and n-grams that occur more often will receive a higher weightage. The ones that are rare will receive a lower weightage.

46.2.2 TF-IDF of n-grams

Frequency-based features can be multiplied with the inverse document frequency to get term frequency-inverse document frequency (TF-IDF) vectors. Doing so also gives importance to the rare terms that occur in the corpus, which may be helpful in certain classification tasks.

$$\text{tfidf}_{i,j} = \text{tf}_{i,j} \times \log \left(\frac{N}{\text{df}_i} \right)$$

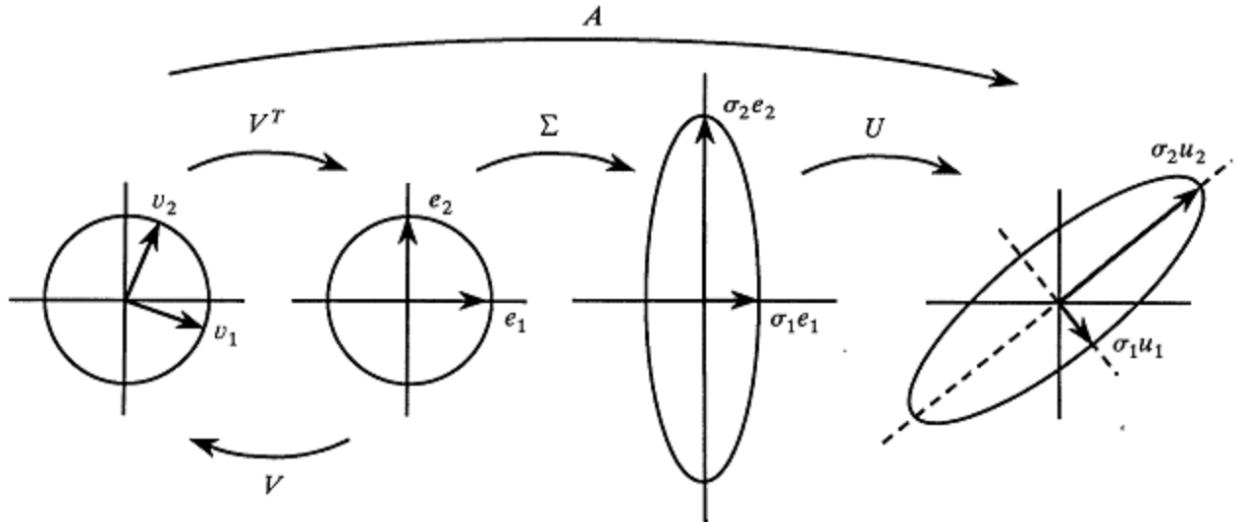
tf_{ij} = total number of occurrences of i in j

df_i = total number of documents (speeches) containing i

N = total number of documents (speeches)

46.3 Truncated SVD Features

TF-IDF and the frequency of n-grams both result in higher dimensions of the representational vectors. To counteract this, **Truncated SVD** is commonly used to decompose the vectorized arrays into lower dimensions.

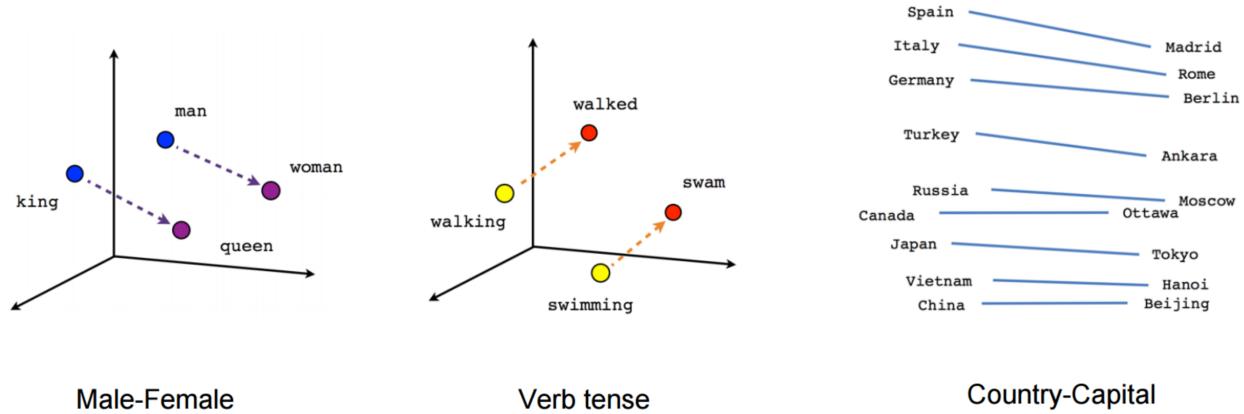


46.4 Linear Models for TF-IDF Vectors

Linear models are also available in the Driverless AI NLP recipe. These capture linear dependencies that are crucial to the process of achieving high accuracy rates and are used as features in the base DAI model.

46.5 Word Embeddings

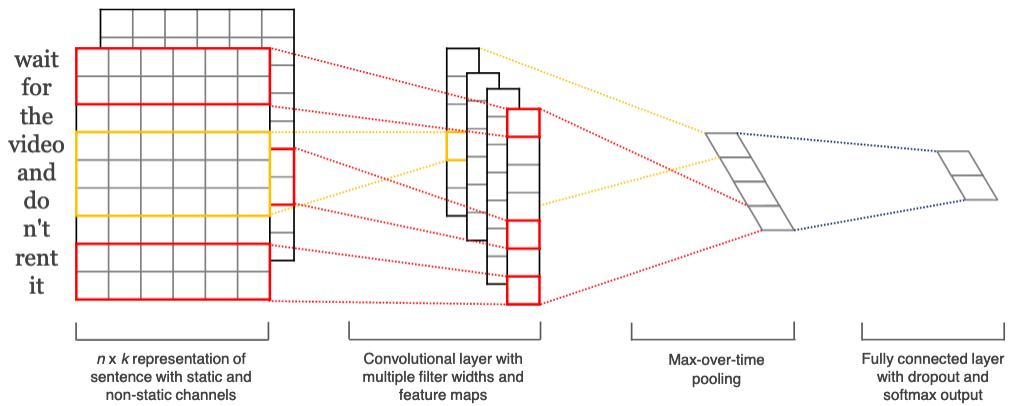
Word embeddings is the term for a collective set of feature engineering techniques for text where words or phrases from the vocabulary are mapped to vectors of real numbers. Representations are made so that words with similar meanings are placed close to or equidistant from one another. For example, the word “king” is closely associated with the word “queen” in this kind of vector representation.



TF-IDF and frequency-based models represent counts and significant word information, but they lack the semantic context for these words. Word embedding techniques are used to make up for this lack of semantic information.

46.5.1 CNN Models for Word Embedding

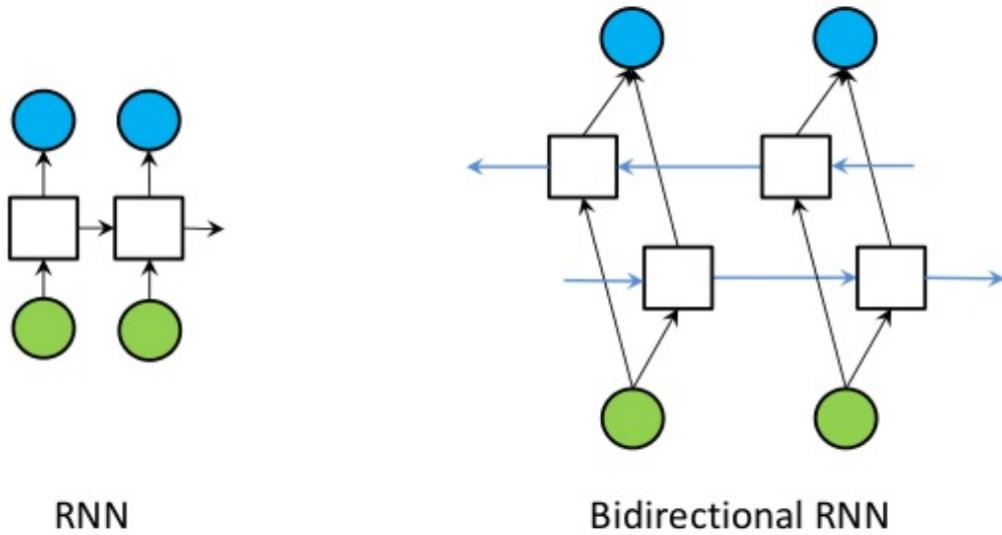
Although Convolutional Neural Network (CNN) models are primarily used on image-level machine learning tasks, their use case on representing text as information has proven to be quite efficient and faster compared to RNN models. In Driverless AI, we pass word embeddings as input to CNN models, which return cross validated predictions that can be used as a new set of features.



46.5.2 Bi-directional GRU Models for Word Embedding

Recurrent neural networks, like long short-term memory units (LSTM) and gated recurrent units (GRU), are state-of-the-art algorithms for NLP problems. In Driverless AI, we implement bi-directional GRU features for previous word steps and for later steps to predict the current state. For example, in the sentence "John is walking on the golf course," a unidirectional model would represent states that represent "golf" based on "John is walking on," but would not represent "course." Using a bi-directional model, the representation would also account the later representations, giving the model more predictive power.

In simple terms, a bi-directional GRU model combines two independent RNN models into a single model. A GRU architecture provides high speeds and accuracy rates similar to a LSTM architecture. As with CNN models, we pass word embeddings as input to these models, which return cross validated predictions that can be used as a new set of features.

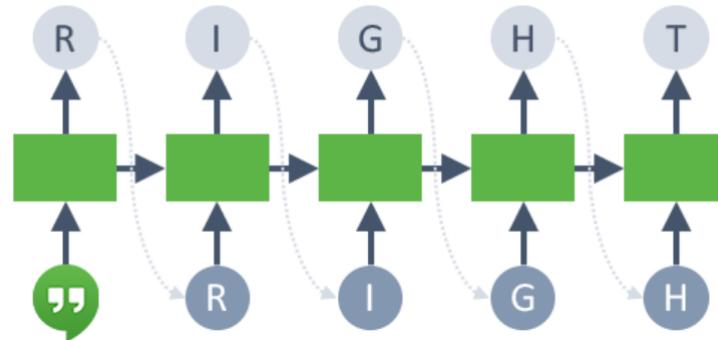


46.5.3 CNN Models for Character Embedding

For languages like Japanese and Mandarin Chinese, where characters play a major role, character level embedding is available as an NLP recipe.

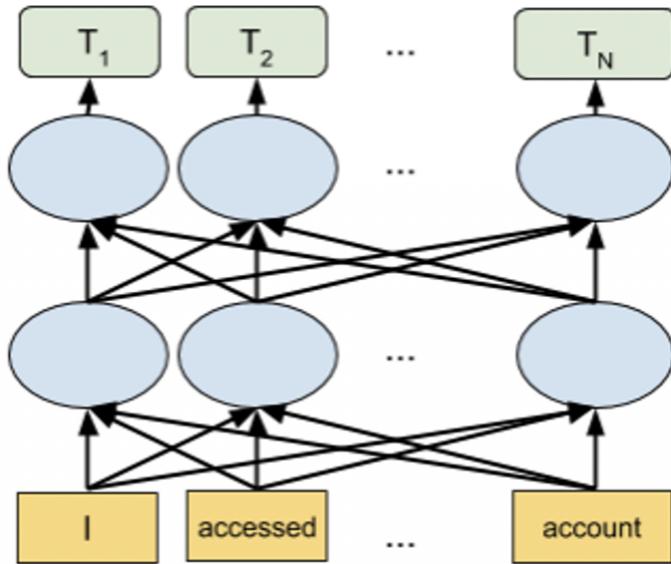
In character embedding, each character is represented in the form of vectors rather than words. Driverless AI uses character level embedding as the input to CNN models and later extracts class probabilities to feed as features for downstream models.

The image below represents the overall set of features created by this NLP recipe:



46.5.4 BERT/DistilBERT Models for Feature Engineering

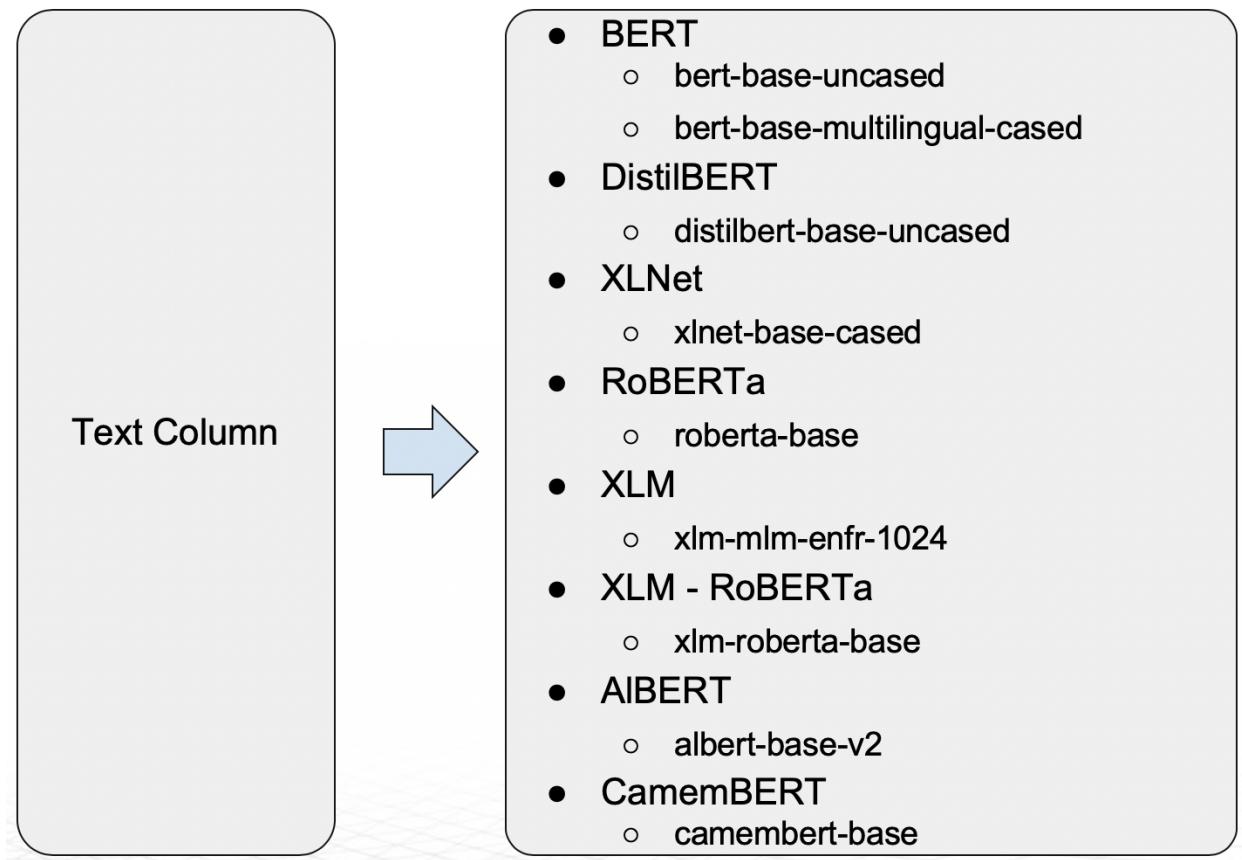
Transformer based language models like BERT are state-of-the-art NLP models that can be used for a wide variety of NLP tasks. These models capture the contextual relation between words by using an attention mechanism. Unlike directional models that read text sequentially, a Transformer-based model reads the entire sequence of text at once, allowing it to learn the context of the word based on all of its surrounding words. The embeddings obtained by these models show improved results in comparison to earlier embedding approaches.



BERT and DistilBERT models can be used for generating embeddings for any text columns. These pretrained models are used to get embeddings for the text followed by Linear/Logistic Regression to generate features that can then be used for any downstream models in Driverless AI. Refer to [NLP Settings](#) in the Expert Settings topic for more information on how to enable these models for feature engineering. We recommend using GPU(s) to leverage the power of these models and accelerate the feature engineering process.

46.6 PyTorch Transformer Architecture Models (eg. BERT) as Modeling Algorithms

Starting with Driverless AI 1.9 release, the Transformer-based architectures shown in the diagram below will be supported as models in Driverless AI.



The BERT model support multiple languages. DistilBERT is a distilled version of BERT that has fewer parameters compared to BERT (40% less) and it is faster (60% speedup) while retaining 95% of BERT level performance. The DistilBERT model can be useful when training time and model size is important. Refer to [NLP Settings](#) in the Expert Settings topic for more information on how to enable these models as modeling algorithms. We recommend using GPU(s) to leverage the power of these models and accelerate the model training time.

46.7 NLP Naming Conventions

The naming conventions of the NLP features help to understand the type of feature that has been created.

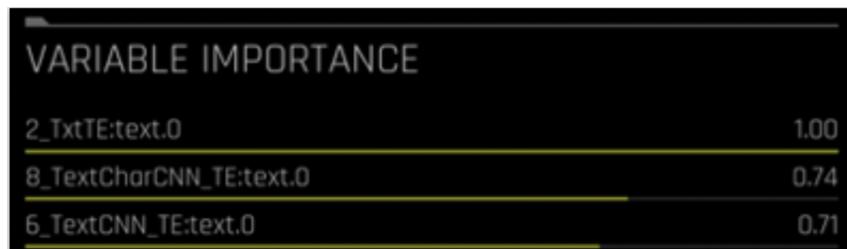
The syntax for the feature names is as follows:

[FEAT TYPE]:[COL].[TARGET_CLASS]

- **[FEAT TYPE]** represents one of the following:
 - Txt – Frequency / TF-IDF of n-grams followed by Truncated SVD
 - TxtTE - Frequency / TF-IDF of n-grams followed by a Linear model
 - TextCNN_TE – Word embeddings followed by CNN model
 - TextBiGRU_TE – Word embeddings followed by Bi-directional GRU model
 - TextCharCNN_TE – Character embeddings followed by CNN model
- **[COL]** represents the name of the text column.

- [TARGET_CLASS] represents the target class for which the model predictions are made.

For example, TxtTE:text.0 equates to class 0 predictions for the text column “text” using Frequency / TF-IDF of n-grams followed by a linear model.



46.8 NLP Expert Settings

A number of configurable settings are available for NLP in Driverless AI. Refer to [NLP Settings](#) in the Expert Settings topic for more information.

EXPERIMENT	MODEL	FEATURES	TIMESERIES	NLP	IMAGE	RECIPES	SYSTEM
Enable word-based CNN TensorFlow models for NLP	<input checked="" type="radio"/> AUTO <input type="radio"/> ON <input type="radio"/> OFF	Enable word-based BiGRU TensorFlow models for NLP	<input checked="" type="radio"/> AUTO <input type="radio"/> ON <input type="radio"/> OFF	Enable character-based CNN TensorFlow models for NLP	<input checked="" type="radio"/> AUTO <input type="radio"/> ON <input type="radio"/> OFF	Enable PyTorch models for NLP (Experimental)	<input checked="" type="radio"/> AUTO <input type="radio"/> ON <input type="radio"/> OFF
Select which pretrained PyTorch NLP model(s) to use.	<input checked="" type="button"/> SELECT VALUES	Max. TensorFlow epochs for NLP	2	Accuracy above enable TensorFlow NLP by default for all models	5	Path to pretrained embeddings for TensorFlow NLP models. If empty, will train from scratch.	
For TensorFlow NLP, allow training of unfrozen pretrained embeddings (in addition to fine-tuning of the rest of the graph)	<input checked="" type="button"/> DISABLED	Number of epochs for fine-tuning of PyTorch NLP models.	2	Batch size for PyTorch NLP models.	10	Maximum sequence length (padding length) for PyTorch NLP models.	100
Path to pretrained PyTorch NLP models. If empty, will get models from S3		Fraction of text columns out of all features to be considered a text-dominated problem	0.3	Fraction of text per all transformers to trigger that text dominated	0.3	Threshold for string columns to be treated as text (0.0 - text, 1.0 - string)	0.3

Buttons:

- SAVE
- CANCEL

46.9 A Typical NLP Example: Sentiment Analysis

The following section provides an NLP example. This information is based on the [Automatic Feature Engineering for Text Analytics](#) blog post. A similar example using the Python Client is available in [The Python Client](#).

This example uses a classic example of sentiment analysis on tweets using the [US Airline Sentiment dataset](#) from [Figure Eight's Data for Everyone](#) library. Note that the sentiment of the tweets have been previously labeled and our model will be used to label new tweets. We can split the dataset into training and test with this simple [script](#). We will just use the tweets in the ‘text’ column and the sentiment (positive, negative or neutral) in the ‘airline_sentiment’ column for this demo. Here are some samples from the dataset:

text	airline_sentiment
@JetBlue @roxydigital HAHA. you didn<89>t disappoint. Well done.	positive
@USAirways will do. Hoping for a voucher for a future flight	positive
@SouthwestAir hello I was talking with @SouthwestVerity but it says you moved here, they sent me a voucher but does not work can u help?	neutral
@AmericanAir when will tomorrow's flight Cancelled Flightlations at Dfw for AA flights be posted? We are on 2424 at 7am from LAX!	negative
@united 3 hrs searching for flts, find 1 on site & can't book the \$ offered bc 1 seg isn't really available. #lies #falseadvertising	negative
@AmericanAir even with calls you haven't been able to help us anyway.	negative

Once we have our dataset ready in the tabular format, we are all set to use the Driverless AI. Similar to other problems in the Driverless AI setup, we need to choose the dataset, and then specify the target column (‘airline_sentiment’).

The screenshot shows the 'EXPERIMENT SETUP' and 'ASSISTANT' sections of the Driverless AI interface. In the 'EXPERIMENT SETUP' section, the 'DISPLAY NAME' is 'Airline NLP'. The 'DATASET' is 'train_airline_sentiment.' The 'ROWS' are '12K', 'COLUMNS' are '20', 'DROPPED COLUMNS' are '18', 'VALIDATION DATASET' is '--', 'TEST DATASET' is 'Yes' (with 'test_airline_sentimen' listed), 'TARGET COLUMN' is 'airline_sentiment', 'FOLD COLUMN' is '--', 'WEIGHT COLUMN' is '--', 'TIME COLUMN' is '[OFF]', 'TYPE' is 'str', 'COUNT' is '11712', 'UNIQUE' is '3', and 'MOST FREQ' is '7334'. In the 'ASSISTANT' section, there is a note: 'Because we don't want to use any other columns in the dataset, we need to click on Dropped Cols, and then exclude everything but text as shown below:'.

Because we don't want to use any other columns in the dataset, we need to click on **Dropped Cols**, and then exclude everything but **text** as shown below:

Select columns to drop (then click done at bottom of page):
18 columns selected.

Licensed to H2O.ai (SN28)

Start typing to filter out items

UNCHECK ALL INVERT SELECTION CHECK ALL

TRAINING DATA

DATASET: train_airline_sentiment.csv

ROWS: 12K COLUMNS: 20 DROPPED COLS: -- VALIDATION DATASET: Yes TEST DATASET: test_airline_sentiment

TARGET COLUMN: airline_sentiment FOLD COLUMN: --

WEIGHT COLUMN: -- TIME COLUMN: [OFF]

TYPE: STR COUNT: 11712 UNIQUE: 3 MOST FREQ: 7334

EXPERIMENT SETTINGS

ACCURACY: 7 TIME: 2 INTERPRETABILITY: 6

EXPERT SETTINGS

SCORER: RNN MCC F05 F1 F2 LOGLOSS AUC AUCPR MACROAUC

CLASSIFICATION REPRODUCIBLE ENABLE GPUs

LAUNCH EXPERIMENT

DONE

© 2017-2018 H2O.ai. All rights reserved.

Next, we will turn on our TensorFlow NLP recipes. We can go to **Expert Settings, NLP** and turn on the following: **CNN TensorFlow models, BiGRU TensorFlow models, character-based TensorFlow models or pretrained PyTorch NLP models**

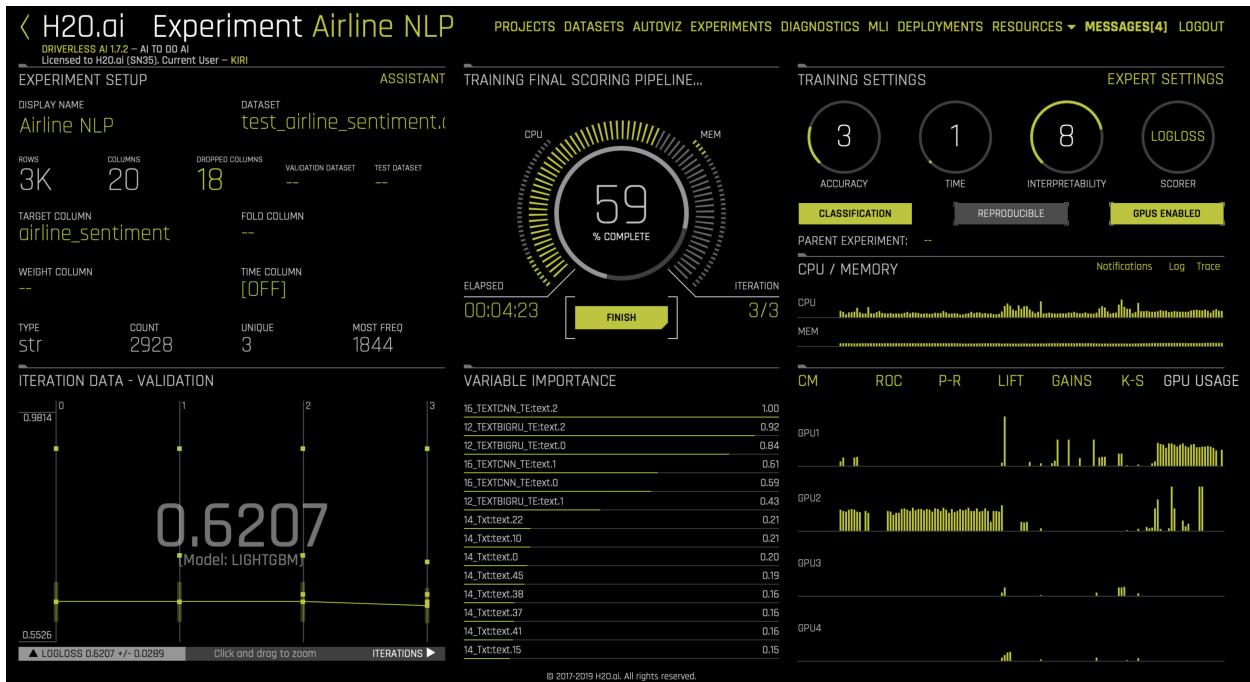
+ UPLOAD CUSTOM RECIPE + LOAD CUSTOM RECIPE FROM URL OFFICIAL RECIPES (EXTERNAL) TOML

Start typing to filter out items

EXPERIMENT	MODEL	FEATURES	TIMESERIES	NLP	IMAGE	RECIPES	SYSTEM
Enable word-based CNN TensorFlow models for NLP <input checked="" type="radio"/> AUTO <input type="radio"/> ON <input type="radio"/> OFF	Enable word-based BiGRU TensorFlow models for NLP <input checked="" type="radio"/> AUTO <input type="radio"/> ON <input type="radio"/> OFF	Enable character-based CNN TensorFlow models for NLP <input checked="" type="radio"/> AUTO <input type="radio"/> ON <input type="radio"/> OFF	Enable PyTorch models for NLP (Experimental) <input checked="" type="radio"/> AUTO <input type="radio"/> ON <input type="radio"/> OFF				
Select which pretrained PyTorch NLP model(s) to use. <input type="button" value="SELECT VALUES"/>	Max. TensorFlow epochs for NLP 2	Accuracy above enable TensorFlow NLP by default for all models 5	Path to pretrained embeddings for TensorFlow NLP models. If empty, will train from scratch. <input type="text" value="disabled"/>				
For TensorFlow NLP, allow training of unfrozen pretrained embeddings (in addition to fine-tuning of the rest of the graph) <input type="radio"/> DISABLED	Number of epochs for fine-tuning of PyTorch NLP models. 2	Batch size for PyTorch NLP models. 10	Maximum sequence length (padding length) for PyTorch NLP models. 100				
Path to pretrained PyTorch NLP models. If empty, will get models from S3 <input type="text"/>	Fraction of text columns out of all features to be considered a text-dominated problem 0.3	Fraction of text per all transformers to trigger that text dominated 0.3	Threshold for string columns to be treated as text (0.0 - text, 1.0 - string) 0.3				

SAVE CANCEL

At this point, we are ready to launch an experiment. Text features will be automatically generated and evaluated during the feature engineering process. Note that some features such as TextCNN rely on TensorFlow models. We recommend using GPU(s) to leverage the power of TensorFlow or the PyTorch Transformer models and accelerate the feature engineering process.



Once the experiment is done, users can make new predictions and download the scoring pipeline just like any other Driverless AI experiments.

Resources:

- **fastText:** <https://fasttext.cc/>
- **GloVe:** <https://nlp.stanford.edu/projects/glove/>

IMAGE PROCESSING IN DRIVERLESS AI

Image processing in Driverless AI is a powerful tool that can be used to gain insight from digital images. This section describes Driverless AI's image processing capabilities.

47.1 Uploading Data for Image Processing

Driverless AI supports multiple methods for uploading image datasets:

- Archive with images in directories for each class. Labels for each class are automatically created based on directory hierarchy
- Archive with images and a CSV file that contains at least one column with relative image paths and a target column (best method for regression)
- CSV file with local paths to the images on the disk
- CSV file with remote URLs to the images

47.2 Modeling Images

Driverless AI features two different approaches to modeling images.

47.2.1 Embeddings Transformer (Image Vectorizer)

The Image Vectorizer transformer utilizes pre-trained [ImageNet](#) models to convert a column with an image path or URI to an embeddings (vector) representation that is derived from the last global average pooling layer of the model. The resulting vector is then used for modeling in Driverless AI.

There are several options in the **Expert Settings** panel that allow you to configure the Image Vectorizer transformer. This panel is available from within the experiment page above the Scorer knob. Refer to [Image Settings](#) for more information on these options.

Notes:

- This modeling approach supports classification and regression experiments.
- This modeling approach supports the use of mixed data types (any number of image columns, text columns, numeric or categorical columns)

47.2.2 Automatic Image Model (GPU Only)

Automatic Image Model is an AutoML model that accepts only an image and a label as input features. This model automatically selects hyperparameters such as learning rate, optimizer, batch size, and image input size. It also automates the training process by selecting the number of epochs, cropping strategy, augmentations, and learning rate scheduler.

Automatic Image Model uses pre-trained [ImageNet](#) models and starts the training process from them. The possible architectures list includes all the well-known models: (SE)-ResNe(X)ts; DenseNets; EfficientNets; Inceptions; etc.

Unique insights that provide information and sample images for the current best individual model are available for Automatic Image Model. To view these insights, click on the **Insights** option while an experiment is running or after an experiment is complete. Refer to [Automatic Image Model Insights](#) for more information.

Enabling Automatic Image Model

To enable Automatic Image Model, navigate to the *Pipeline Building Recipe* expert setting and select the **image_model** option:

After confirming your selection, click **Save**. The experiment preview section updates to include information about Automatic Image Model:

What do these settings mean?

- ACCURACY**
 - Training data size: **20,000 rows, 2 cols**
 - Final pipeline: **ImageAuto, single final model**
- TIME**
 - Only Training Final Pipeline
- INTERPRETABILITY**
 - Feature pre-pruning strategy: **None**
 - Monotonicity constraints: **enabled**
 - Feature engineering search space: **ImageOriginal**

ImageAuto models to train:
- Final pipeline: **1**
Auto-click Finish/Abort if not done in: **1 day/7 days**

PROJECT

EXPERIMENT SETUP

DISPLAY NAME
Display name

ROWS 20K **COLUMNS** 2

TARGET COLUMN label

WEIGHT COLUMN --

TYPE bool

TRAINING SETTINGS

ACCURACY: 7

CLASSIFICATION

Notes:

- This modeling approach only supports a single image column as an input.
- This modeling approach does not support any transformers.
- This modeling approach supports classification and regression experiments.
- This modeling approach does not support the use of mixed data types because of its limitation on input features.
- This modeling approach does not use Genetic Algorithm (GA).

CHAPTER
FORTYEIGHT

QUEUEING

Driverless AI supports automatic queuing of experiments to avoid system overload. You can launch multiple experiments simultaneously that are automatically queued and run when the necessary resources become available.

Note: By default, each node runs two experiments at a time. This is controlled by the `worker_remote_processors` option in the `config.toml` file. Additional options that control resource allocation can also be configured in the `config.toml` file.

Also see [Multinode Training \(Alpha\)](#)

THE PYTHON CLIENT

This section describes how to install the Driverless AI Python Client. It also provides some end-to-end examples showing how to use the Driverless AI Python client. Additional examples are available in the <https://github.com/h2oai/driverlessai-tutorials> repository.

49.1 Installing the Python Client

The Python Client is available on the Driverless AI UI and published on the h2oai channel at <https://anaconda.org/h2oai/repo>.

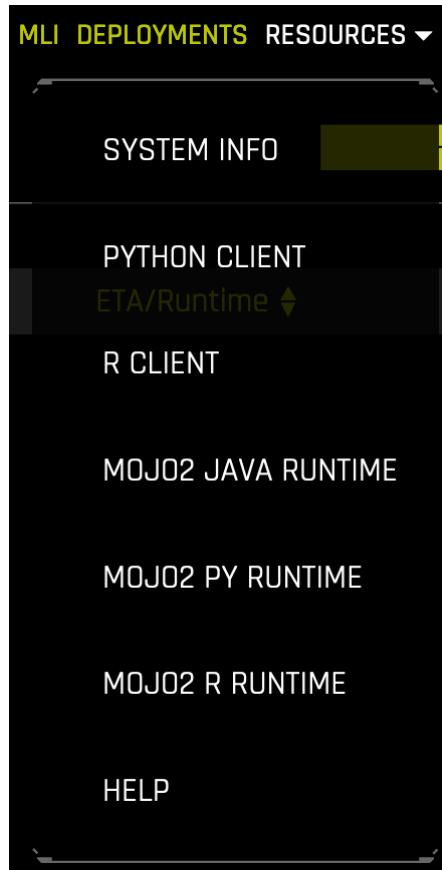
49.1.1 Installing from Driverless AI

Requirements

- Python 3.6. This is the only supported version.

Download from UI

On the Driverless AI top menu, select the **RESOURCES > PYTHON CLIENT** link. This downloads the **h2oai_client** wheel.



Download from Command Line

The Driverless AI Python client is exposed as the `/clients/py` HTTP end point. This can be accessed via the command line:

```
wget --trust-server-names http://<Driverless AI address>/clients/py
```

Wheel Installation

Install this wheel to your local Python via `pip` command. Once installed, you can launch a Jupyter notebook and begin using the Driverless AI Python Client.

49.1.2 Installing from Anaconda Cloud

Note: Conda installs of the Python client are not supported on Windows.

Requirements

- Conda Package Manager. You can install the Conda Package Manager either through Anaconda or Miniconda. Note that the Driverless AI Python client requires Python 3.6, so ensure that you install the Python 3 version of Anaconda or Miniconda.
 - [Anaconda Install Instructions](#)
 - [Miniconda Install Instructions](#)

Installation Procedure

After Conda is installed and the Conda executable is available in \$PATH, create a new Anaconda environment for h2oai_client:

```
conda create -n h2oaiclientenv -c h2oai -c conda-forge h2oai_client
```

The above command installs the latest version of the Python client. Include the version number to install a specific version. For example, the following command installs the 1.6.3 Python client:

```
conda create -n h2oaiclientenv -c h2oai -c conda-forge h2oai_client=1.6.3
```

A list of available Python client versions is available here: https://anaconda.org/h2oai/h2oai_client/files

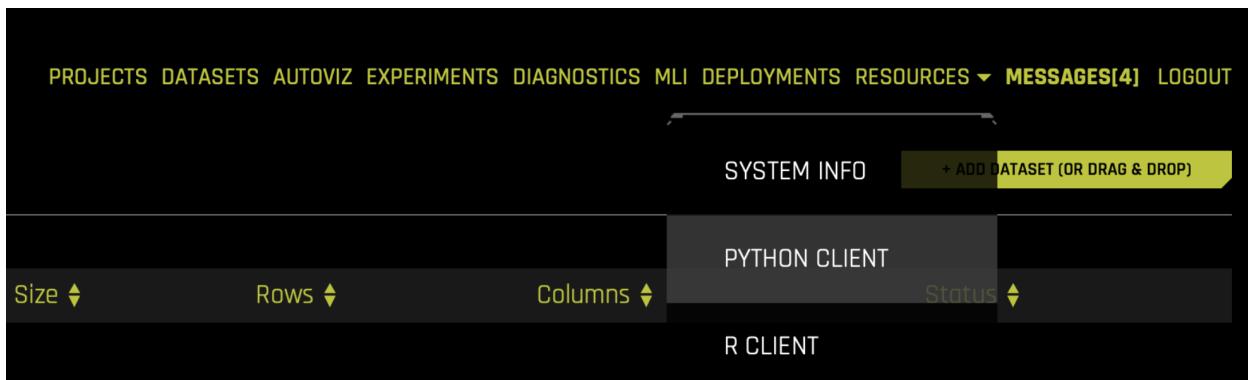
Upon completion, you can launch a Jupyter notebook and begin using the Driverless AI Python Client.

49.2 Driverless AI: Credit Card Demo

This notebook provides an H2OAI Client workflow, of model building and scoring, that parallels the Driverless AI workflow.

Notes:

- This is an early release of the Driverless AI Python client.
- This notebook was tested in Driverless AI version 1.8.5.
- Python 3.6 is the only supported version.
- You must install the h2oai_client wheel to your local Python. This is available from the RESOURCES link in the top menu of the UI.



49.2.1 Workflow Steps

Build an Experiment with Python API:

1. Sign in
2. Import train & test set/new data
3. Specify experiment parameters
4. Launch Experiment
5. Examine Experiment
6. Download Predictions

Build an Experiment in Web UI and Access Through Python:

1. Get pointer to experiment

Score on New Data:

1. Score on new data with H2OAI model

Model Diagnostics on New Data:

1. Run model diagnostics on new data with H2OAI model

Run Model Interpretation

1. Run model interpretation on the raw features
2. Run Model Interpretation on External Model Predictions

Build Scoring Pipelines

1. Build Python Scoring Pipeline
2. Build MOJO Scoring Pipeline

49.2.2 Build an Experiment with Python API

1. Sign In

Import the required modules and log in.

Pass in your credentials through the Client class which creates an authentication token to send to the Driverless AI Server. In plain English: to sign into the Driverless AI web page (which then sends requests to the Driverless Server), instantiate the Client class with your Driverless AI address and login credentials.

```
[1]: from h2oai_client import Client
import matplotlib.pyplot as plt
import pandas as pd

[2]: address = 'http://ip_where_driverless_is_running:12345'
username = 'username'
password = 'password'
h2oai = Client(address = address, username = username, password = password)
# make sure to use the same user name and password when signing in through the GUI
```

Equivalent Steps in Driverless: Signing In

2. Upload Datasets

Upload training and testing datasets from the Driverless AI /data folder.

You can provide a training, validation, and testing dataset for an experiment. The validation and testing dataset are optional. In this example, we will provide only training and testing.

```
[3]: train_path = '/data/Kaggle/CreditCard/CreditCard-train.csv'
test_path = '/data/Kaggle/CreditCard/CreditCard-test.csv'

train = h2oai.create_dataset_sync(train_path)
test = h2oai.create_dataset_sync(test_path)
```

Equivalent Steps in Driverless: Uploading Train & Test CSV Files

Name	Path	Size	Rows	Columns	Status
CreditCard-test.csv	..CreditCard/CreditCard-test.csv	570KB	6K	25	[Click for Actions]
CreditCard-train.csv	..CreditCard/CreditCard-train.csv	2MB	24K	25	[Click for Actions]

3. Set Experiment Parameters

We will now set the parameters of our experiment. Some of the parameters include:

- Target Column: The column we are trying to predict.
- Dropped Columns: The columns we do not want to use as predictors such as ID columns, columns with data leakage, etc.
- Weight Column: The column that indicates the per row observation weights. If `None`, each row will have an observation weight of 1.
- Fold Column: The column that indicates the fold. If `None`, the folds will be determined by Driverless AI.
- Is Time Series: Whether or not the experiment is a time-series use case.

For information on the experiment settings, refer to the [Experiment Settings](#).

For this example, we will be predicting `“default payment next month”`. The parameters that control the experiment process are: accuracy, time, and interpretability. We can use the `get_experiment_preview_sync` function to get a sense of what will happen during the experiment.

We will start out by seeing what the experiment will look like with accuracy, time, and interpretability all set to 5.

```
[4]: target="default payment next month"
exp_preview = h2oai.get_experiment_preview_sync(dataset_key= train.key
                                                , validset_key=''
                                                , classification=True
                                                , dropped_cols=[]
                                                , target_col=target
                                                , is_time_series=False
                                                , enable_gpus=False
                                                , accuracy=5, time=5, interpretability=5
                                                , reproducible=True
                                                , resumed_experiment_id=''
                                                , time_col=''
                                                , config_overrides=None)

exp_preview
```

```
[4]: ['ACCURACY [5/10]:',
      '- Training data size: *23,999 rows, 25 cols*',
      '- Feature evolution: *[LightGBM, XGBoostGBM]*, *1/3 validation split*',
      '- Final pipeline: *Ensemble (4 models), 4-fold CV*',
      '',
      'TIME [5/10]:',
      '- Feature evolution: *4 individuals*, up to *66 iterations*',
      '- Early stopping: After *10* iterations of no improvement',
      '',
      'INTERPRETABILITY [5/10]:',
      '- Feature pre-pruning strategy: None',
      '- Monotonicity constraints: disabled',
      '- Feature engineering search space: [CVCatNumEncode, CVTargetEncode, ClusterDist, ClusterTE, Frequent, Interactions, NumCatTE, NumToCatTE, NumToCatWoE, ↴Original, TruncSVDNum, WeightOfEvidence]*',
      '',
      '[LightGBM, XGBoostGBM] models to train:',
      '- Model and feature tuning: *16*',
      '- Feature evolution: *104*',
      '- Final pipeline: *4*',
      '',
      'Estimated runtime: *minutes*',
      'Auto-click Finish/Abort if not done in: *1 day*/*7 days*']
```

With these settings, the Driverless AI experiment will train about 124 models:

- * 16 for model and feature tuning
- * 104 for feature evolution
- * 4 for the final pipeline

When we start the experiment, we can either:

- specify parameters
- use Driverless AI to suggest parameters

Driverless AI can suggest the parameters based on the dataset and target column. Below we will use the ```get_experiment_tuning_suggestion``` to see what settings Driverless AI suggests.

```
[5]: # let Driverless suggest parameters for experiment
params = h2oai.get_experiment_tuning_suggestion(dataset_key = train.key, target_col = target,
                                                is_classification = True, is_time_series = False,
                                                config_overrides = None, cols_to_drop=[])

params.dump()

[5]: {'dataset': {'key': '6f09ae32-33dc-11ea-ba27-0242ac110002',
  'display_name': ''},
'resumed_model': {'key': '', 'display_name': ''},
'target_col': 'default payment next month',
'weight_col': '',
'fold_col': '',
'orig_time_col': '',
'time_col': '',
'is_classification': True,
'cols_to_drop': [],
'validset': {'key': '', 'display_name': ''},
'testset': {'key': '', 'display_name': ''},
'enable_gpus': True,
'seed': False,
'accuracy': 5,
'time': 4,
'interpretability': 6,
'score_f_name': 'AUC',
'time_groups_columns': [],
'unavailable_columns_at_prediction_time': [],
'time_period_in_seconds': None,
'num_prediction_periods': None,
'num_gap_periods': None,
'is_timeseries': False,
'config_overrides': None}
```

Driverless AI has found that the best parameters are to set ``**accuracy = 5**``, ``**time = 4**``, ``**interpretability = 6**``. It has selected ``**AUC**`` as the scorer (this is the default scorer for binomial problems).

Equivalent Steps in Driverless: Set the Knobs, Configuration & Launch

The screenshot shows the H2O.ai Experiment interface. On the left, a sidebar displays performance metrics:

- What do these settings mean?**
- ACCURACY**: - Training data size: **23,999 rows, 25 cols**, - Feature evolution: **[GLM, LightGBM, XGBoostGBM]**, 1/3 **validation split**, - Final pipeline: **Ensemble (4 models), 4-fold CV**
- TIME**: - Feature evolution: **4 individuals**, up to **48 iterations**, - Early stopping: After **5** iterations of no improvement
- INTERPRETABILITY**: - Feature pre-pruning strategy: **FS**, - XGBoost Monotonicity constraints: **disabled**, - Feature engineering search space (where applicable): **[CVCatNumEncode, CVTargetEncode, ClusterTE, Dates, Frequent, Interactions, IsHoliday, NumCatTE, NumToCatTE, NumToCatWoE, Original, TextLinModel, Text, WeightOfEvidence]**
- [GLM, LightGBM, XGBoostGBM] models to train:** - Model and feature tuning: **16**, - Feature evolution: **84**, - Final pipeline: **4**
- Estimated runtime: **minutes**

The main interface shows the following configuration:

- EXPERIMENT SETUP**:
 - DISPLAY NAME**: CreditCard-train.csv
 - ROWS**: 24K
 - COLUMNS**: 25
 - DROPPED COLS**: --
 - VALIDATION DATASET**: --
 - TARGET COLUMN**: default payment next
 - FOLD COLUMN**: --
 - WEIGHT COLUMN**: --
 - TIME COLUMN**: [OFF]
 - TYPE**: bool
 - COUNT**: 23999
 - UNIQUE**: 2
 - TARGET FREQ**: 5369
- TRAINING SETTINGS**:
 - Accuracy: 5
 - Time: 4
 - Interpretability: 6
- EXPERT SETTINGS**:
 - Scorer: AUC
 - GPUs Disabled

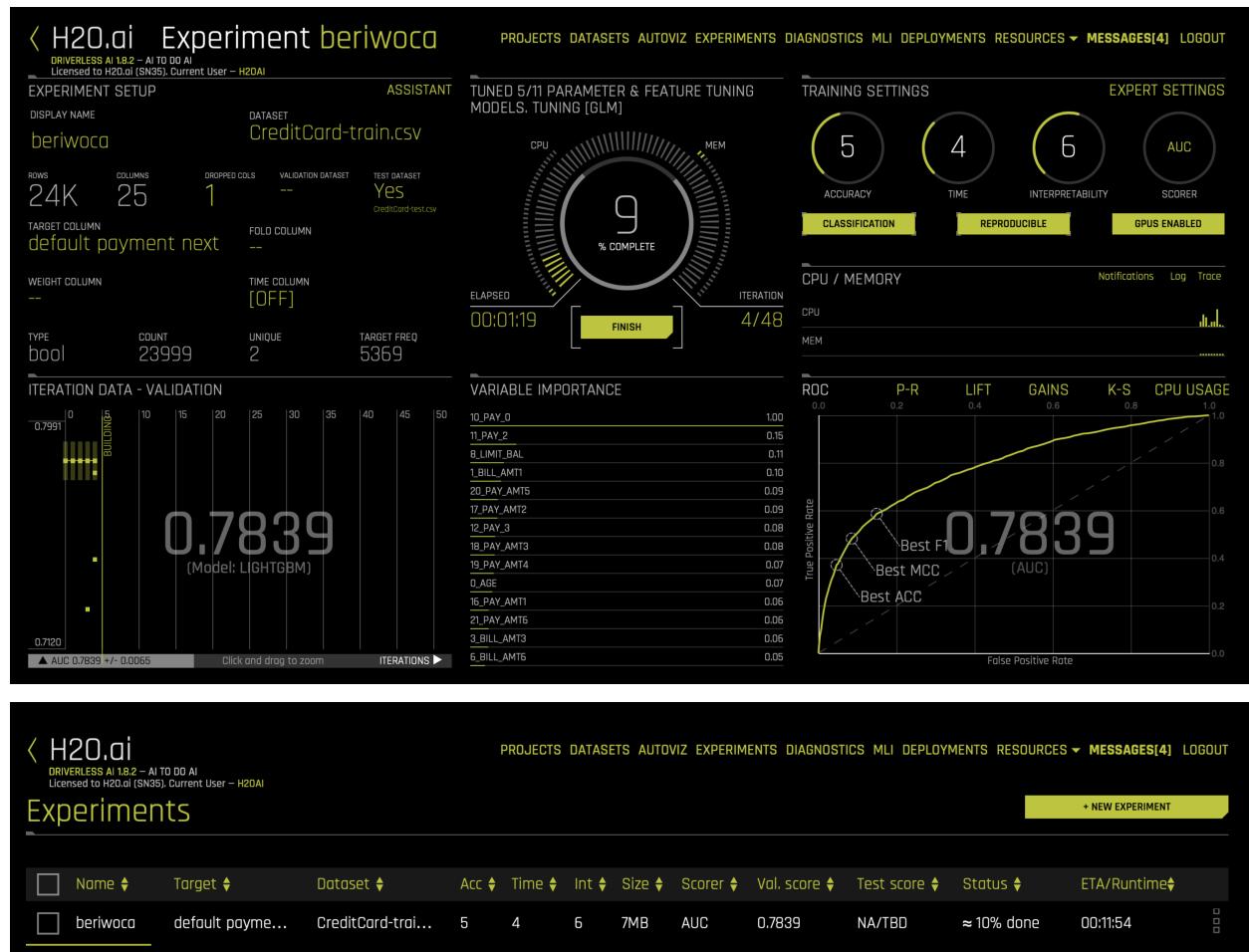
A large green button at the bottom right says **LAUNCH EXPERIMENT**.

4. Launch Experiment: Feature Engineering + Final Model Training

Launch the experiment using the parameters that Driverless AI suggested along with the testset, scorer, and seed that were added. We can launch the experiment with the suggested parameters or create our own.

```
[6]: experiment = h2oai.start_experiment_sync(dataset_key=train.key,
                                             testset_key = test.key,
                                             target_col=target,
                                             is_classification=True,
                                             accuracy=5,
                                             time=4,
                                             interpretability=6,
                                             scorer="AUC",
                                             enable_gpus=True,
                                             seed=1234,
                                             cols_to_drop=['ID'])
```

Equivalent Steps in Driverless: Launch Experiment



5. Examine Experiment

View the final model score for the validation and test datasets. When feature engineering is complete, an ensemble model can be built depending on the accuracy setting. The experiment object also contains the score on the validation and test data for this ensemble model. In this case, the validation score is the score on the training cross-validation predictions.

```
[7]: print("Final Model Score on Validation Data: " + str(round(experiment.valid_score, 3)))
print("Final Model Score on Test Data: " + str(round(experiment.test_score, 3)))

Final Model Score on Validation Data: 0.779
Final Model Score on Test Data: 0.8
```

The experiment object also contains the scores calculated for each iteration on bootstrapped samples on the validation data. In the iteration graph in the UI, we can see the mean performance for the best model (yellow dot) and +/- 1 standard deviation of the best model performance (yellow bar).

This information is saved in the experiment object.

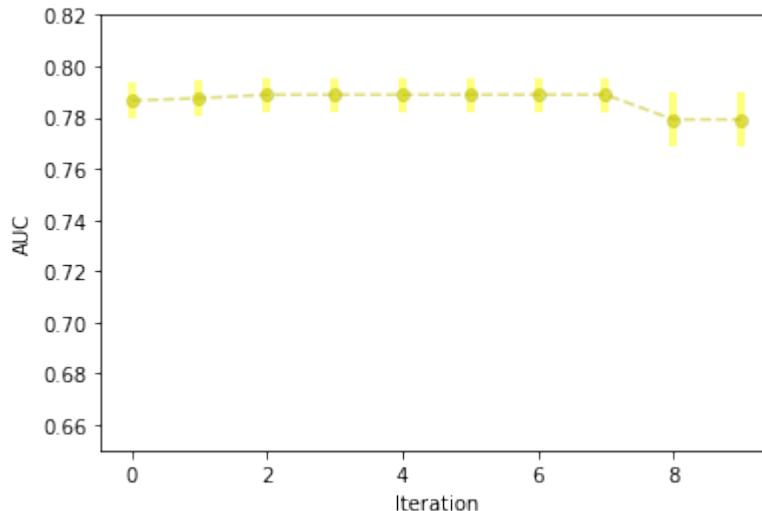
```
[8]: # Add scores from experiment iterations
iteration_data = h2oai.list_model_iteration_data(experiment.key, 0, len(experiment.iteration_data))
iterations = list(map(lambda iteration: iteration.iteration, iteration_data))
scores_mean = list(map(lambda iteration: iteration.score_mean, iteration_data))
scores_sd = list(map(lambda iteration: iteration.score_sd, iteration_data))
```

(continues on next page)

(continued from previous page)

```
# Add score from final ensemble
iterations = iterations + [max(iterations) + 1]
scores_mean = scores_mean + [experiment.valid_score]
scores_sd = scores_sd + [experiment.valid_score_sd]

plt.figure()
plt.errorbar(iterations, scores_mean, yerr=scores_sd, color = "y",
             ecolor='yellow', fmt = '--o', elinewidth = 4, alpha = 0.5)
plt.xlabel("Iteration")
plt.ylabel("AUC")
plt.ylim([0.65, 0.82])
plt.show();
```



Equivalent Steps in Driverless: View Results

H2O.ai Experiment beriwoca

DRIVERLESS AI 1.7.1 - AI TO DO AI
Licensed to H2O.ai (SN35). Current User - H2OAI

EXPERIMENT SETUP

DISPLAY NAME: beriwoca
ROWS: 24K COLUMNS: 25 DROPPED COLS: 1 VALIDATION DATASET: Yes
TARGET COLUMN: default payment next FOLD COLUMN: --
WEIGHT COLUMN: -- TIME COLUMN: [OFF]
TYPE: bool COUNT: 23999 UNIQUE: 2 TARGET FREQ: 5369

ITERATION DATA - VALIDATION

0.7780 (Model: [LIGHTGBM])

PROJECTS
DATASETS
AUTOVIZ
EXPERIMENTS
DIAGNOSTICS
MLI
DEPLOYMENTS
RESOURCES
MESSAGES [4]
LOGOUT

STATUS: COMPLETE

- DEPLOY (LOCAL & CLOUD)
- INTERPRET THIS MODEL
- DIAGNOSE MODEL ON NEW DATASET...
- SCORE ON ANOTHER DATASET...
- TRANSFORM ANOTHER DATASET...
- DOWNLOAD PREDICTIONS ▾
- DOWNLOAD PYTHON SCORING PIPELINE
- DOWNLOAD MOJO SCORING PIPELINE
- VISUALIZE SCORING PIPELINE (EXPERIMENTAL)
- DOWNLOAD SUMMARY & LOGS
- DOWNLOAD AUTOREPORT

TRAINING SETTINGS

5
ACCURACY

4
TIME

6
INTERPRETABILITY

AUC
SCORER

EXPERT SETTINGS

CLASSIFICATION

REPRODUCIBLE

GPUS ENABLED

CPU / MEMORY

CPU: Notifications Log Trace

MEM:

VARIABLE IMPORTANCE

Variable	Importance
10_PAY_0	1.00
40_NumToCatWoE_PAY_0_0	0.27
39_InteractionSubLIMIT_BAL_PAY_0	0.14
1_BILL_AMT1	0.12
17_PAY_AMT2	0.12
11_PAY_2	0.12
18_PAY_AMT3	0.10
16_PAY_AMT1	0.10
21_PAY_AMT6	0.09
19_PAY_AMT4	0.08
0_AGE	0.08
20_PAY_AMT5	0.08
12_PAY_3	0.07
2_BILL_AMT2	0.06

ROC P-R LIFT GAINS K-S SUMMARY

Experiment: beriwoca (0abb4dc0-c460-11eb-b100-0242ac10002)
Version: 1.8.2, 2020-01-10 18:40
Settings: 5/4/6, seed=1234, GPUs enabled
Train data: CreditCard-train.csv (23999, 24)
Validation data: CreditCard-test.csv (6000, 23)
Target column: default payment next month (binary, 22.372% target class)
System Specs: Docker/Linux, 240 GB, 32 CPU cores, 4/4 GPUs
Max memory usage: 157 GB, 158 GB GPU
Recipe: AutoDL (7 iterations, 8 individuals)
Validation scheme: stratified, 1 internal holdout
Feature engineering: 218 features scored (23 selected)
Timing: MOJO latency: 0.09432 millis (769.0KB)
Shift/Leakage detection: 29.38 secs
Model and feature tuning: 108.64 secs (16 models trained)
Feature evolution: 47.93 secs (12 of 168 models trained)
Final pipeline training: 4112 secs (4 models trained)
Python / MOJO scorer building: 22.06 secs / 2.32 secs
Validation score: AUC = 0.7865391 +/- 0.006819665 (baseline)
Validation score: AUC = 0.7791419 +/- 0.01039529 (final pipeline)
Test score: AUC = 0.8003248 +/- 0.01039523 (final pipeline)

6. Download Results

Once an experiment is complete, we can see that the UI presents us options of downloading the:

- predictions
 - on the (holdout) train data
 - on the test data
- experiment summary - summary of the experiment including feature importance

We will show an example of downloading the test predictions below. Note that equivalent commands can also be run for downloading the train (holdout) predictions.

```
[9]: h2oai.download(src_path=experiment.test_predictions_path, dest_dir=".")
[9]: './test_preds.csv'

[10]: test_preds = pd.read_csv("./test_preds.csv")
[10]: test_preds.head()

default payment next month.0  default payment next month.1
0          0.399347          0.600653
1          0.858302          0.141698
2          0.938646          0.061354
3          0.619994          0.380006
4          0.865251          0.134749
```

49.2.3 Build an Experiment in Web UI and Access Through Python

It is also possible to use the Python API to examine an experiment that was started through the Web UI using the experiment key.

Name	Target	Dataset	Acc	Time	Int	Size	Scorer	Vol. Score	Test Score	Status	ETA/Runtime
tadicefe	default payme...	CreditCard-trai...	5	4	6	464MB	AUC	0.7791	0.80032	Completed	00:04:36
bugaboni	default payme...	CreditCard_trai...	6	3	7	462MB	AUC	0.7826	0.76728	Completed	00:04:55

1. Get pointer to experiment

You can get a pointer to the experiment by referencing the experiment key in the Web UI.

```
[11]: # Get list of experiments
experiment_list = list(map(lambda x: x.key, h2oai.list_models(offset=0, limit=100).models))
experiment_list

[11]: ['7fb429e-33dc-11ea-ba27-0242ac110002',
       '0be7d94a-33d8-11ea-ba27-0242ac110002',
       '2e6bbcfa-30a1-11ea-83f9-0242ac110002',
       '3c06c58c-27fd-11ea-9e09-0242ac110002',
       'a3c6dfda-2353-11ea-9f4a-0242ac110002',
       '15fe0c0a-203d-11ea-97b6-0242ac110002']
```

```
[12]: # Get pointer to experiment
experiment = h2oai.get_model_job(experiment_list[0]).entity
```

49.2.4 Score on New Data

You can use the Python API to score on new data. This is equivalent to the **SCORE ON ANOTHER DATASET** button in the Web UI. The example below scores on the test data and then downloads the predictions.

Pass in any dataset that has the same columns as the original training set. If you passed a test set during the H2OAI model building step, the predictions already exist.

1. Score Using the H2OAI Model

The following shows the predicted probability of default for each record in the test.

```
[13]: prediction = h2oai.make_prediction_sync(experiment.key, test.key, output_margin = False, pred_contribs = False)
pred_path = h2oai.download(prediction.predictions_csv_path, '.')
pred_table = pd.read_csv(pred_path)
pred_table.head()
```

	default payment next month.0	default payment next month.1
0	0.399347	0.600653
1	0.858302	0.141698
2	0.938646	0.061354
3	0.619994	0.380006
4	0.865251	0.134749

We can also get the contribution each feature had to the final prediction by setting `pred_contribs = True`. This will give us an idea of how each feature effects the predictions.

```
[14]: prediction_contributions = h2oai.make_prediction_sync(experiment.key, test.key,
                                                       output_margin = False, pred_contribs = True)
pred_contributions_path = h2oai.download(prediction_contributions.predictions_csv_path, '.')
pred_contributions_table = pd.read_csv(pred_contributions_path)
pred_contributions_table.head()
```

	contrib_0_AGE	contrib_10_PAY_0	contrib_11_PAY_2	contrib_12_PAY_3	contrib_13_PAY_4	contrib_14_PAY_5	contrib_15_PAY_6	contrib_16_PAY_AMT1	contrib_17_PAY_AMT2	contrib_18_PAY_AMT3	...	contrib_22_SEX
0	-0.017284	1.559683	0.336257	-0.004992	-0.009970	-0.028749	-0.046268	0.064267	-0.021834	-0.015015	...	0.000940
1	-0.006587	-0.283259	-0.058643	-0.014668	-0.025089	-0.021999	-0.021543	0.091787	-0.018144	-0.011283	-0.017874	0.047035
2	-0.017300	-0.347706	-0.061856	-0.027669	-0.016219	0.170304	0.081330	0.153046	-0.029481	-0.260402	-0.058225	-0.155098
3	-0.064728	0.916047	0.081330	0.059009	-0.047545	-0.022028	-0.024870	0.076568	-0.019231	0.123152	...	0.021810
4	-0.029481	-0.260402	-0.058225	-0.076568	-0.015791	-0.161386	-0.032228	-0.039787	-0.010946	-0.015015	-0.027173	-0.000515

We will examine the contributions for our first record more closely.

```
[15]: contrib = pd.DataFrame(pred_contributions_table.iloc[0][1:])
contrib.columns = ["contribution"]
contrib["abs_contribution"] = contrib.contribution.abs()
contrib.sort_values(by="abs_contribution", ascending=False)[["contribution"]].head()
```

contribution	
contrib_10_PAY_0	1.559683
contrib_bias	-1.507163
contrib_11_PAY_2	0.336257
contrib_8_LIMIT_BAL	0.163888
contrib_1_BILL_AMT1	-0.100378

The clusters from this customer's: PAY_0, PAY_2, and LIMIT_BAL had the greatest impact on their prediction. Since the contribution is positive, we know that it increases the probability that they will default.

49.2.5 Model Diagnostics on New Data

You can use the Python API to also perform model diagnostics on new data. This is equivalent to the **Model Diagnostics** tab in the Web UI.

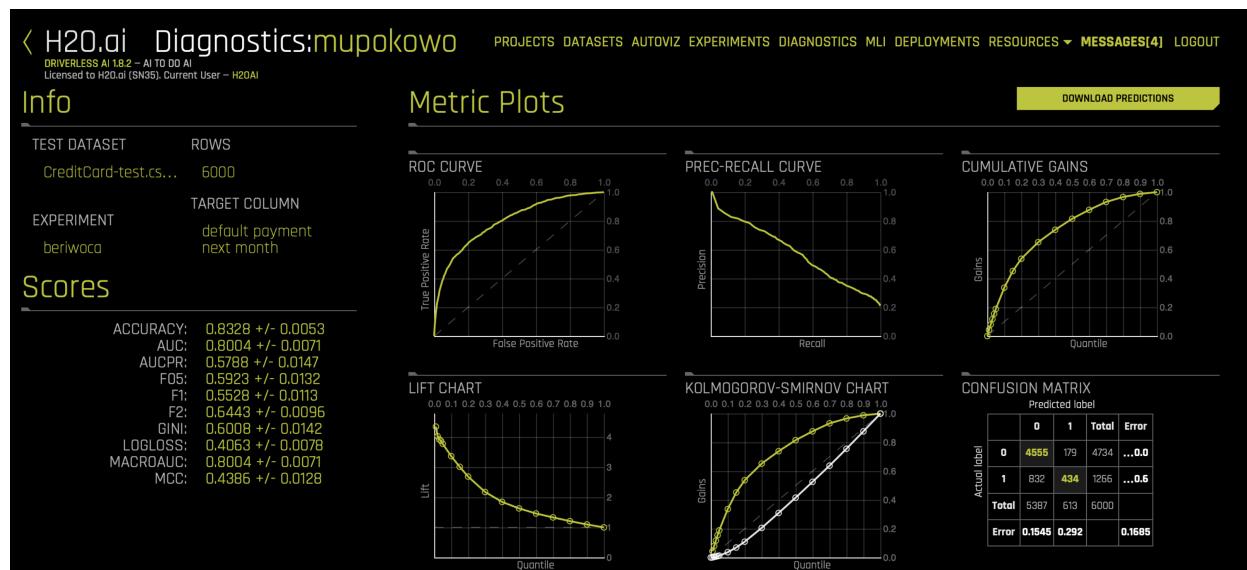
1. Run model diagnostics on new data with H2OAI model

The example below performs model diagnostics on the test dataset but any data with the same columns can be selected.

```
[16]: test_diagnostics = h2oai.make_model_diagnostic_sync(experiment.key, test.key)

[17]: [ {'scorer': x.score_f_name, 'score': x.score} for x in test_diagnostics.scores]
```

Here is the same model diagnostics displayed in the UI:



49.2.6 Run Model Interpretation

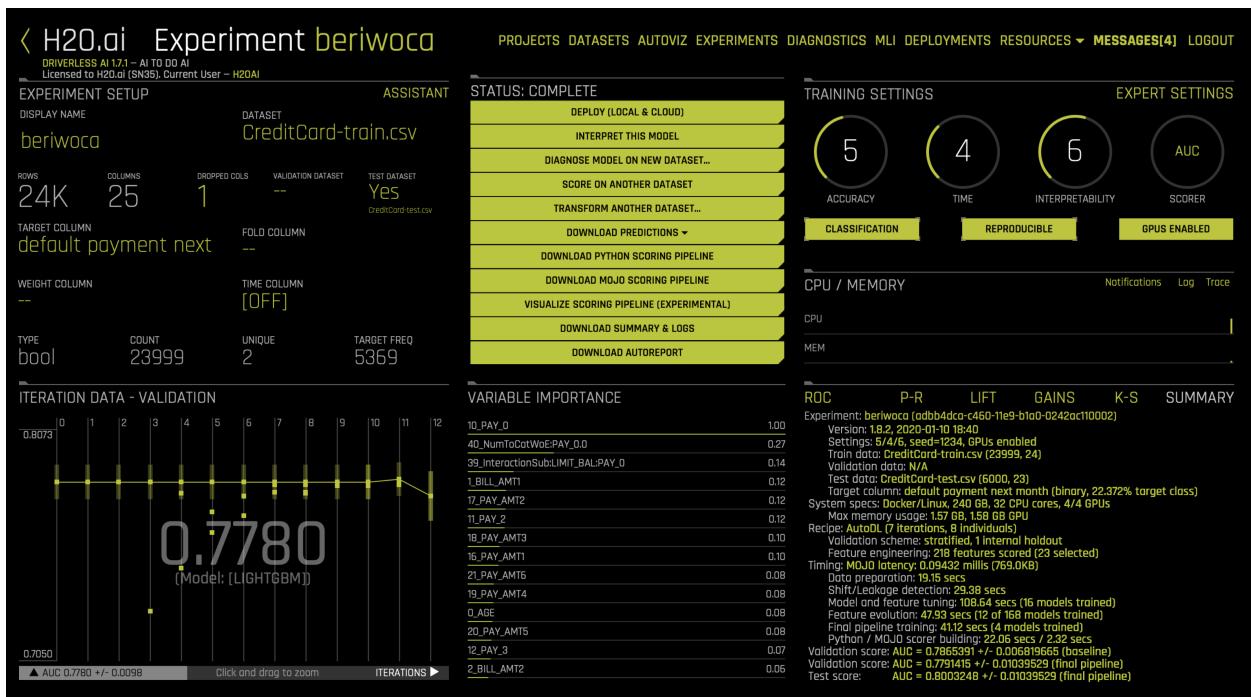
Once we have completed an experiment, we can interpret our H2OAI model. Model Interpretability is used to provide model transparency and explanations. More information on Model Interpretability can be found here: <http://docs.h2o.ai/driverless-ai/latest-stable/docs/userguide/interpreting.html>.

1. Run Model Interpretation on the Raw Data

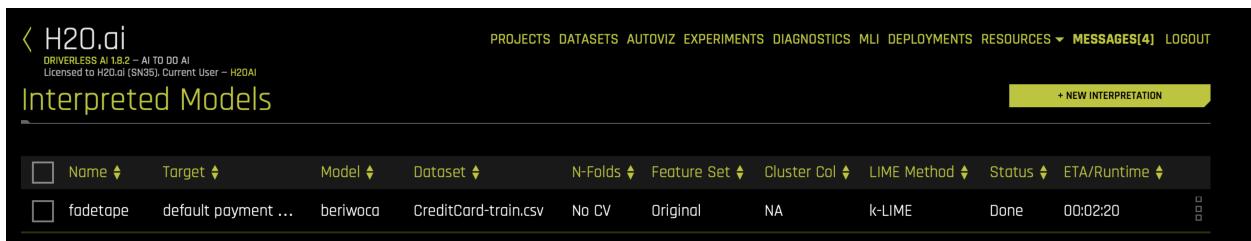
We can run the model interpretation in the Python client as shown below. By setting the parameter, `use_raw_features` to True, we are interpreting the model using only the raw features in the data. This will not use the engineered features we saw in our final model's features to explain the data.

```
[18]: mli_experiment = h2oai.run_interpretation_sync(dai_model_key = experiment.key,
                                                dataset_key = train.key,
                                                target_col = target,
                                                use_raw_features = True)
```

This is equivalent to clicking **Interpret this Model on Original Features** in the UI once the experiment has completed.



Once our interpretation is finished, we can navigate to the MLI tab in the UI to see our interpreted model.



We can also see the list of interpretations using the Python Client:

```
[19]: # Get list of interpretations
mli_list = list(map(lambda x: x.key, h2oai.list_interpretations(offset=0, limit=100)))
mli_list

[19]: ['2ff44e94-33de-11ea-ba27-0242ac110002',
 '6ce9e9f6-33db-11ea-ba27-0242ac110002']
```

2. Run Model Interpretation on External Model Predictions

Model Interpretation does not need to be run on a Driverless AI experiment. We can also train an external model and run Model Interpretability on the predictions. In this next section, we will walk through the steps to interpret an external model.

Train External Model

We will begin by training a model with scikit-learn. Our end goal is to use Driverless AI to interpret the predictions made by our scikit-learn model.

```
[25]: # Dataset must be located where Python client is running - you may need to download it locally  
train_pd = pd.read_csv(train_path)
```

```
[26]: from sklearn.ensemble import GradientBoostingClassifier  
  
predictors = list(set(train_pd.columns) - set([target]))  
  
gbm_model = GradientBoostingClassifier(random_state=10)  
gbm_model.fit(train_pd[predictors], train_pd[target])  
  
[26]: GradientBoostingClassifier(criterion='friedman_mse', init=None,  
                                learning_rate=0.1, loss='deviance', max_depth=3,  
                                max_features=None, max_leaf_nodes=None,  
                                min_impurity_decrease=0.0, min_impurity_split=None,  
                                min_samples_leaf=1, min_samples_split=2,  
                                min_weight_fraction_leaf=0.0, n_estimators=100,  
                                n_iter_no_change=None, presort='auto',  
                                random_state=10, subsample=1.0, tol=0.0001,  
                                validation_fraction=0.1, verbose=0,  
                                warm_start=False)
```

```
[27]: predictions = gbm_model.predict_proba(train_pd[predictors])  
predictions[0:5]
```

```
[27]: array([[0.20060823, 0.79939177],  
           [0.57311744, 0.42688256],  
           [0.88034896, 0.11965104],  
           [0.86483594, 0.13516406],  
           [0.87349721, 0.12650279]])
```

Interpret on External Predictions

Now that we have the predictions from our scikit-learn GBM model, we can call Driverless AI's ```h2oai.run_interpretation_sync``` to create the interpretation screen.

```
[28]: train_gbm_path = "./CreditCard-train-gbm_pred.csv"  
predictions = pd.concat([train_pd, pd.DataFrame(predictions[:, 1], columns = ["p1"])], axis = 1)  
predictions.to_csv(path_or_buf=train_gbm_path, index = False)
```

```
[29]: train_gbm_pred = h2oai.upload_dataset_sync(train_gbm_path)
```

```
[30]: mli_external = h2oai.run_interpretation_sync(dai_model_key = "", # no experiment key  
                                                dataset_key = train_gbm_pred.key,  
                                                target_col = target,  
                                                prediction_col = "p1")
```

We can also run Model Interpretability on an external model in the UI as shown below:

SELECT OR IMPORT DATA

DATASET
CreditCard-train-gbm_pred.csv

ROWS 24K	COLUMNS 26		
TARGET COLUMN default payment next	PREDICTION COLUMN default payment next		
TYPE bool	COUNT 23999	UNIQUE 2	FREQ 5369

LIME METHOD K-LIME	DECISION TREE SURROGATE TREE DEPTH 3	SAMPLE
DIA COLUMNS --	NUMBER OF FEATURES FOR PD/ICE 10	
WEIGHT COLUMN --	DROPPED COLS --	
K-LIME CLUSTERING COLUMN (OPTIONAL & ONLY USED BY THE "K-LIME" LIME METHOD) --	SURROGATE CV FOLDS 3	
QUANTILE BINNING COLS --	NUMBER OF COLUMNS TO BIN 0	

LAUNCH ML!

```
[31]: # Get list of interpretations
mi_list = list(map(lambda x: x.key, h2oai.list_interpretations(offset=0, limit=100)))
mi_list

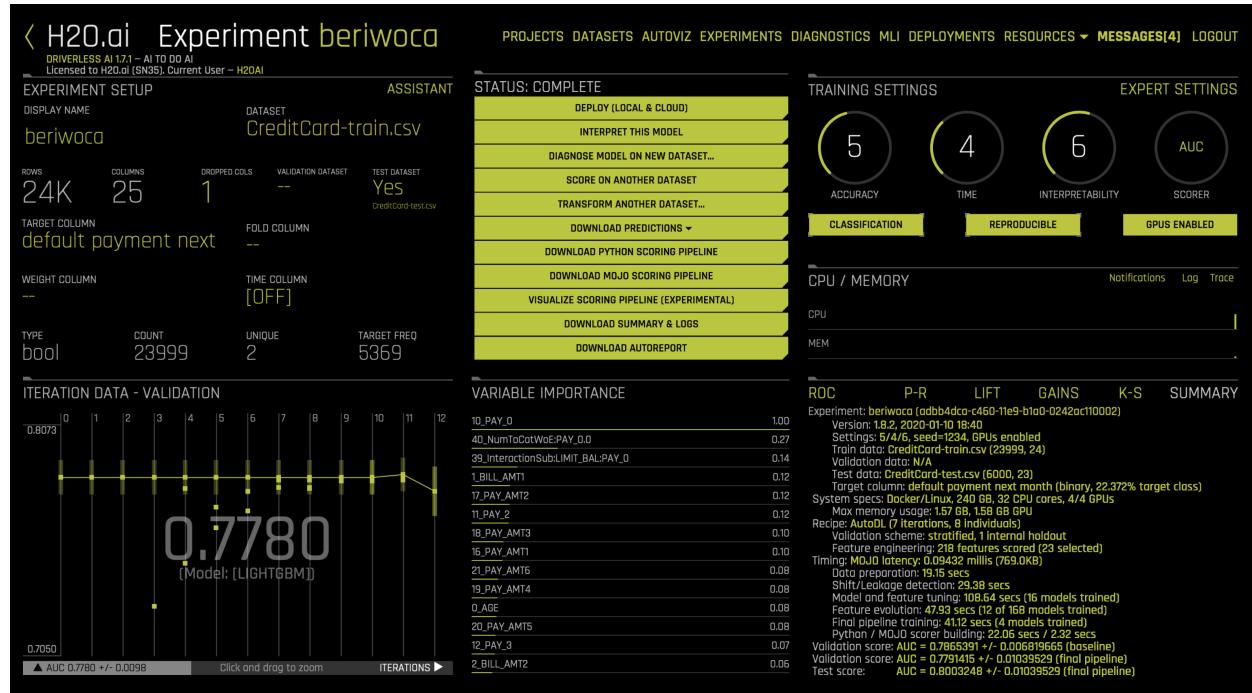
[31]: ['minemutu', 'nutiduha', 'senikahi']
```

49.2.7 Build Scoring Pipelines

In our last section, we will build the scoring pipelines from our experiment. There are two scoring pipeline options:

- Python Scoring Pipeline: requires Python runtime
- MOJO Scoring Pipeline: requires Java runtime

Documentation on the scoring pipelines is provided here: <http://docs.h2o.ai/driverless-ai/latest-stable/docs/userguide/python-mojo-pipelines.html>.



The experiment screen shows two scoring pipeline buttons: **Download Python Scoring Pipeline** or **Build MOJO Scoring Pipeline**. Driverless AI determines if any scoring pipeline should be automatically built based on the config.toml file. In this example, we have run Driverless AI with the settings:

```
# Whether to create the Python scoring pipeline at the end of each experiment
make_python_scoring_pipeline = true

# Whether to create the MOJO scoring pipeline at the end of each experiment
# Note: Not all transformers or main models are available for MOJO (e.g. no gblinear main model)
make_mojo_scoring_pipeline = false
```

Therefore, only the Python Scoring Pipeline will be built by default.

1. Build Python Scoring Pipeline

The Python Scoring Pipeline has been built by default based on our config.toml settings. We can get the path to the Python Scoring Pipeline in our experiment object.

```
[32]: experiment.scoring_pipeline_path
[32]: 'h2oai_experiment_daguwofe/scoring_pipeline/scorer.zip'
```

We can also build the Python Scoring Pipeline - this is useful if the ``make_python_scoring_pipeline`` option was set to false.

```
[58]: python_scoring_pipeline = h2oai.build_scoring_pipeline_sync(experiment.key)
```

```
[59]: python_scoring_pipeline.file_path  
[59]: 'h2oai_experiment_adbb4dca-c460-11e9-b1a0-0242ac110002/scoring_pipeline/scorer.zip'
```

Now we will download the scoring pipeline zip file.

```
[60]: h2oai.download(python_scoring_pipeline.file_path, dest_dir=".")  
[60]: './scorer.zip'
```

2. Build MOJO Scoring Pipeline

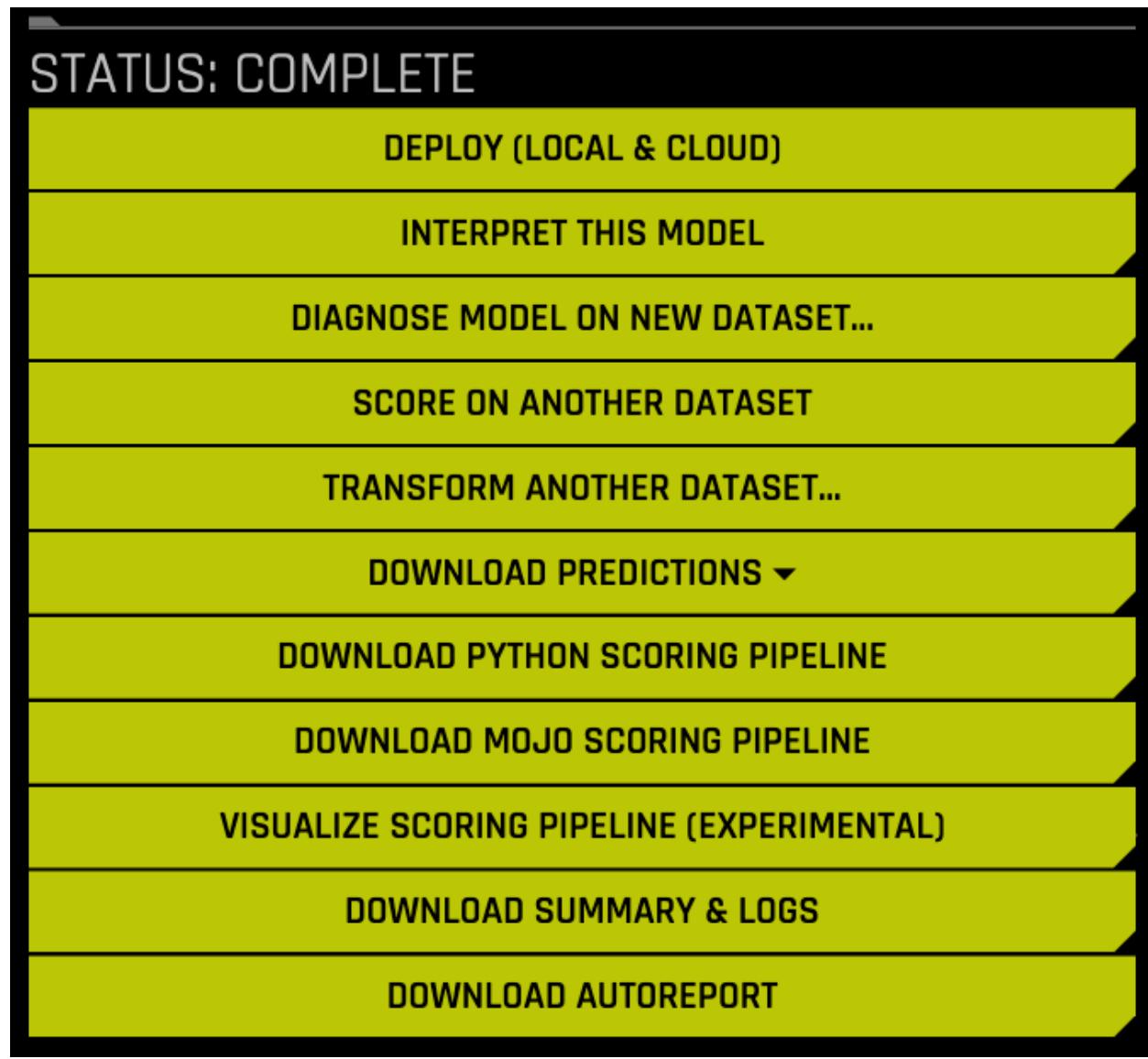
The MOJO Scoring Pipeline has not been built by default because of our config.toml settings. We can build the MOJO Scoring Pipeline using the Python client. This is equivalent to selecting the **Build MOJO Scoring Pipeline** on the experiment screen.

```
[61]: mojo_scoring_pipeline = h2oai.build_mojo_pipeline_sync(experiment.key)  
  
[62]: mojo_scoring_pipeline.file_path  
[62]: 'h2oai_experiment_adbb4dca-c460-11e9-b1a0-0242ac110002/mojo_pipeline/mojo.zip'
```

Now we can download the scoring pipeline zip file.

```
[63]: h2oai.download(mojo_scoring_pipeline.file_path, dest_dir=".")  
[63]: './mojo.zip'
```

Once the MOJO Scoring Pipeline is built, the **Build MOJO Scoring Pipeline** changes to **Download MOJO Scoring Pipeline**.



[]:

49.3 Driverless AI - Training Time Series Model

The purpose of this notebook is to show an example of using Driverless AI to train a time series model. Our goal will be to forecast the Weekly Sales for a particular Store and Department for the next week. The data used in this notebook is from the: [Walmart Kaggle Competition](#) where `features.csv` and `train.csv` have been joined together.

Note: This notebook was tested and run on Driverless AI 1.8.1.

49.3.1 Workflow

1. Import data into Python
2. Format data for Time Series
3. Upload data to Driverless AI
4. Launch Driverless AI Experiment
5. Evaluate model performance

```
[1]: import pandas as pd
from h2oai_client import Client

%matplotlib inline
import matplotlib.pyplot as plt
import matplotlib.dates as mdates
```

49.3.2 Step 1: Import Data

We will begin by importing our data using pandas. We are going to first work with the data in Python to correctly format it for a Driverless AI time series use case.

```
[2]: sales_data = pd.read_csv("./walmart_train.csv")
sales_data.head()

[2]:   Store  Dept       Date  Weekly_Sales  Temperature  Fuel_Price  MarkDown \
0      1      1  2010-02-05      24924.50      42.31      2.572    -1.0
1      1      2  2010-02-05      50605.27      42.31      2.572    -1.0
2      1      3  2010-02-05     13740.12      42.31      2.572    -1.0
3      1      4  2010-02-05     39954.04      42.31      2.572    -1.0
4      1      5  2010-02-05     32229.38      42.31      2.572    -1.0

   MarkDown2  MarkDown3  MarkDown4  MarkDown5        CPI  Unemployment \
0      -1.0      -1.0      -1.0      -1.0  211.096358      8.106
1      -1.0      -1.0      -1.0      -1.0  211.096358      8.106
2      -1.0      -1.0      -1.0      -1.0  211.096358      8.106
3      -1.0      -1.0      -1.0      -1.0  211.096358      8.106
4      -1.0      -1.0      -1.0      -1.0  211.096358      8.106

   IsHoliday  sample_weight
0          0              1
1          0              1
2          0              1
3          0              1
4          0              1
```

```
[3]: # Convert Date column to datetime
sales_data["Date"] = pd.to_datetime(sales_data["Date"], format="%Y-%m-%d")
```

49.3.3 Step 2: Format Data for Time Series

The data has one record per Store, Department, and Week. Our goal for this use case will be to forecast the total sales for the next week.

The only features we should use as predictors are ones that we will have available at the time of scoring. Features like the Temperature, Fuel Price, and Unemployment will not be known in advance. Therefore, before we start our Driverless AI experiments, we will choose to use the previous week's Temperature, Fuel Price, Unemployment, and CPI attributes. This information we will know at time of scoring.

```
[4]: lag_variables = ["Temperature", "Fuel_Price", "CPI", "Unemployment"]
dai_data = sales_data.set_index(["Date", "Store", "Dept"])
lagged_data = dai_data.loc[:, lag_variables].groupby(level=["Store", "Dept"]).shift(1)

[5]: # Join lagged predictor variables to training data
dai_data = dai_data.join(lagged_data.rename(columns=lambda x: x + "_lag"))

[6]: # Drop original predictor variables - we do not want to use these in the model
dai_data = dai_data.drop(lagged_data, axis=1)
dai_data = dai_data.reset_index()
```

```
[7]: dai_data.head()
[7]:
   Date  Store  Dept  Weekly_Sales  Markdown1  Markdown2  Markdown3  \
0 2010-02-05      1     1    24924.50     -1.0     -1.0     -1.0
1 2010-02-05      1     2    50605.27     -1.0     -1.0     -1.0
2 2010-02-05      1     3    13740.12     -1.0     -1.0     -1.0
3 2010-02-05      1     4    39954.04     -1.0     -1.0     -1.0
4 2010-02-05      1     5    32229.38     -1.0     -1.0     -1.0

  Markdown4  Markdown5  IsHoliday  sample_weight  Temperature_lag  \
0     -1.0     -1.0          0           1        NaN
1     -1.0     -1.0          0           1        NaN
2     -1.0     -1.0          0           1        NaN
3     -1.0     -1.0          0           1        NaN
4     -1.0     -1.0          0           1        NaN

  Fuel_Price_lag  CPI_lag  Unemployment_lag
0            NaN     NaN             NaN
1            NaN     NaN             NaN
2            NaN     NaN             NaN
3            NaN     NaN             NaN
4            NaN     NaN             NaN
```

Now that our training data is correctly formatted, we can run a Driverless AI experiment to forecast the next week's sales.

49.3.4 Step 3: Upload Data to Driverless AI

We will split out data into two pieces: training and test (which consists of the last week of data).

```
[8]: train_data = dai_data.loc[dai_data["Date"] < "2012-10-26"]
test_data = dai_data.loc[dai_data["Date"] == "2012-10-26"]
```

To upload the datasets, we will sign into Driverless AI.

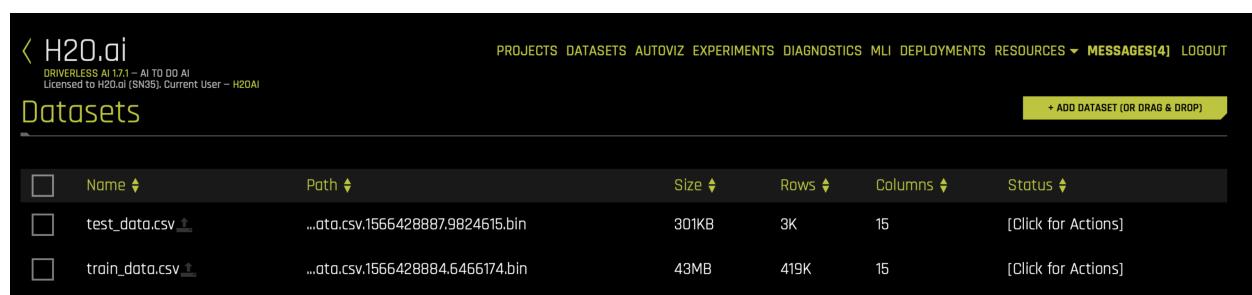
```
[9]: address = 'http://<ip_where_driverless_is_running>:12345'
username = 'username'
password = 'password'
h2oai = Client(address = address, username = username, password = password)
# make sure to use the same user name and password when signing in through the GUI
```

```
[10]: train_path = "./train_data.csv"
test_path = "./test_data.csv"

train_data.to_csv(train_path, index = False)
test_data.to_csv(test_path, index = False)
```

```
[11]: # Add datasets to Driverless AI
train_dai = h2oai.upload_dataset_sync(train_path)
test_dai = h2oai.upload_dataset_sync(test_path)
```

Equivalent Steps in Driverless: Uploading Train & Test CSV Files



The screenshot shows the H2O.ai interface for managing datasets. At the top, there are navigation links: PROJECTS, DATASETS, AUTOVIZ, EXPERIMENTS, DIAGNOSTICS, MLI, DEPLOYMENTS, RESOURCES, MESSAGES (4), and LOGOUT. Below the header, a banner indicates 'DRIVERLESS AI 1.7.1 - AI TO DO AI' and 'Licensed to H2O.ai (SN35). Current User - H2OAI'. A large green button on the right says '+ ADD DATASET (OR DRAG & DROP)'. The main area is titled 'Datasets' and contains a table with two rows of uploaded files:

	Name	Path	Size	Rows	Columns	Status
<input type="checkbox"/>	test_data.csv	...ata.csv.1566428887.9824615.bin	301KB	3K	15	[Click for Actions]
<input type="checkbox"/>	train_data.csv	...ata.csv.1566428884.6466174.bin	43MB	419K	15	[Click for Actions]

49.3.5 Step 4: Launch Driverless AI Experiment

We will now launch the Driverless AI experiment. To do that we will need to specify the parameters for our experiment. Some of the parameters include:

- Target Column: The column we are trying to predict.
- Dropped Columns: The columns we do not want to use as predictors such as ID columns, columns with data leakage, etc.
- Is Time Series: Whether or not the experiment is a time-series use case.
- Time Column: The column that contains the date/date-time information.
- Time Group Columns: The categorical columns that indicate how to group the data so that there is one time series per group. In our example, our Time Groups Columns are `Store` and `Dept`. Each `Store` and `Dept`, corresponds to a single time series.
- Number of Prediction Periods: How far in the future do we want to predict?
- Number of Gap Periods: After how many periods can we start predicting? If we assume that we can start forecasting right after the training data ends, then the Number of Gap Periods will be 0.

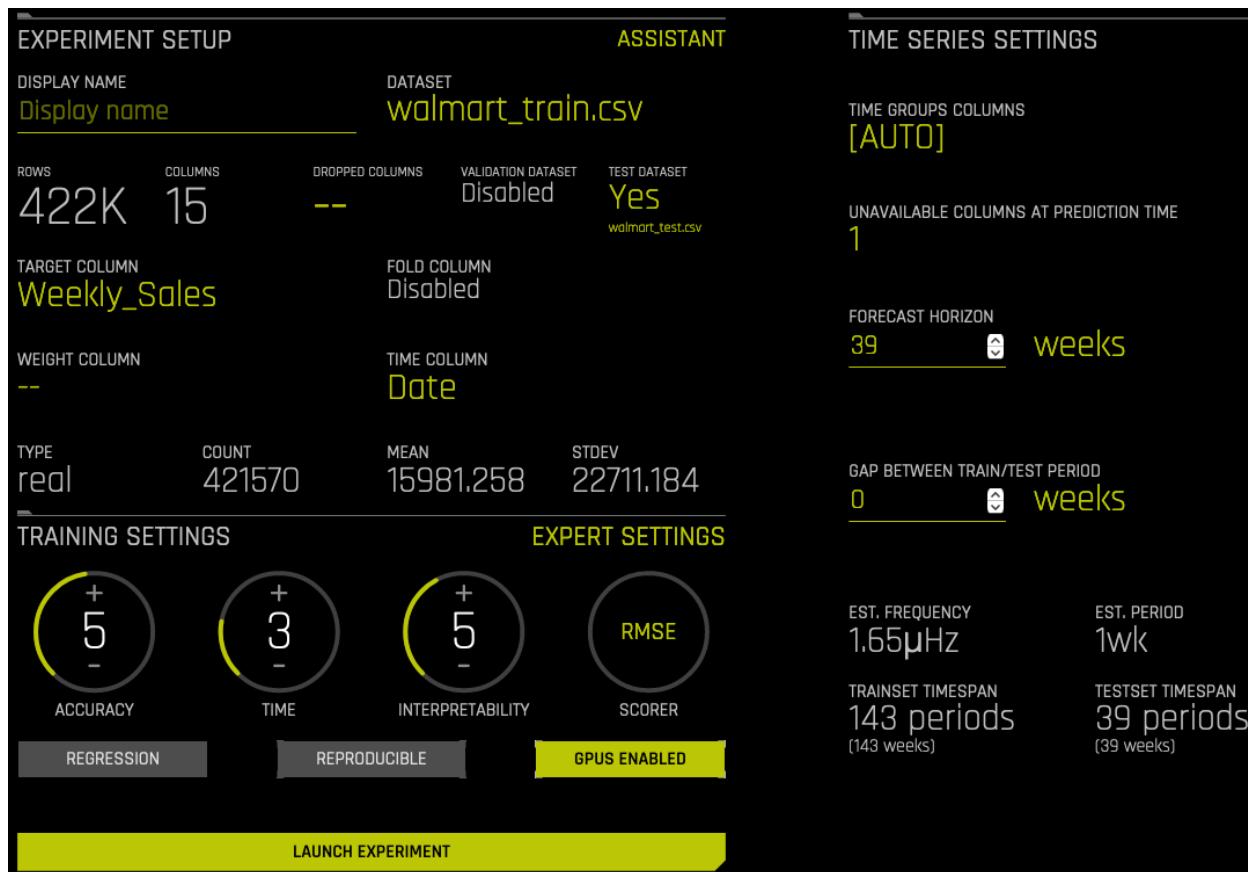
For this experiment, we want to forecast next week's sales for each `Store` and `Dept`. Therefore, we will use the following time series parameters:

- Time Group Columns: `[Store, Dept]`
- Number of Prediction Periods: 1 (a.k.a., horizon)
- Number of Gap Periods: 0

Note that the period size is unknown to the Python client. To overcome this, you can also specify the optional `time_period_in_seconds` parameter, which can help specify the horizon in real time units. If this parameter is omitted, Driverless AI will automatically detect the period size in the experiment, and the horizon value will respect this period. I.e., if you are sure your data has 1 week period, you can say `num_prediction_periods=14`, otherwise it is possible that the model may not work out correctly.

```
[12]: experiment = h2oai.start_experiment_sync(dataset_key=train_dai.key,
                                             testset_key = test_dai.key,
                                             target_col="Weekly_Sales",
                                             is_classification=False,
                                             cols_to_drop = ["sample_weight"],
                                             accuracy=5,
                                             time=3,
                                             interpretability=1,
                                             scorer="RMSE",
                                             enable_gpus=True,
                                             seed=1234,
                                             time_col = "Date",
                                             time_groups_columns = ["Store", "Dept"],
                                             num_prediction_periods = 1,
                                             num_gap_periods = 0)
```

Equivalent Steps in Driverless: Launching Driverless AI Experiment



49.3.6 Step 5. Evaluate Model Performance

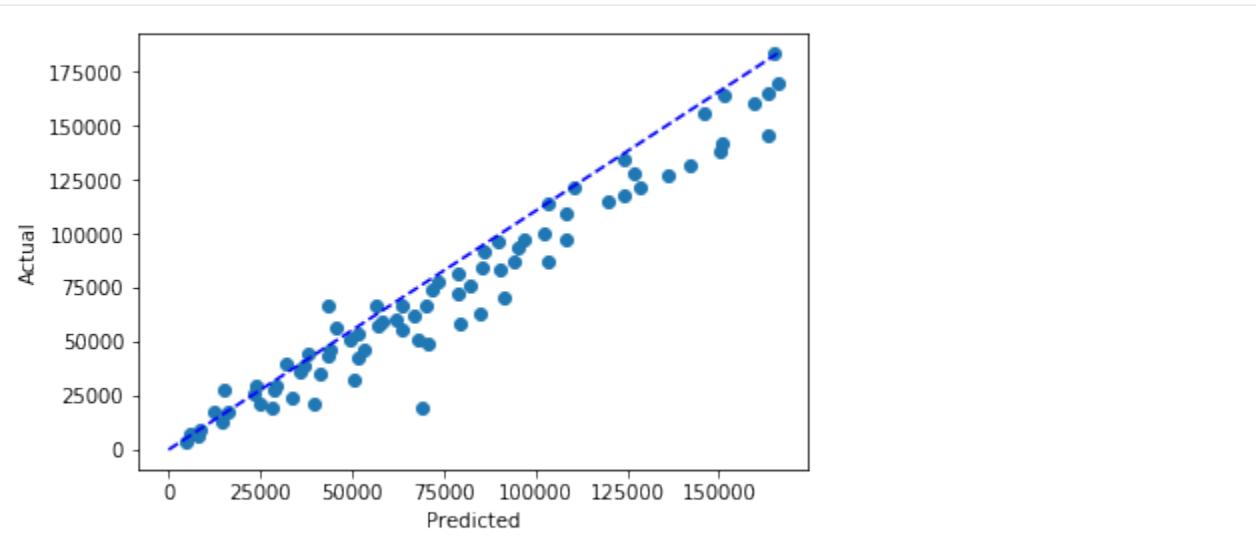
Now that our experiment is complete, we can view the model performance metrics within the experiment object.

```
[13]: print("Validation RMSE: ${:.0f}".format(experiment.valid_score))
print("Test RMSE: ${:.0f}".format(experiment.test_score))

Validation RMSE: $2,281
Test RMSE: $2,483
```

We can also plot the actual versus predicted values from the test data.

```
[14]: plt.scatter(experiment.test_act_vs_pred.x_values, experiment.test_act_vs_pred.y_values)
plt.plot([0, max(experiment.test_act_vs_pred.x_values)], [0, max(experiment.test_act_vs_pred.y_values)], 'b--')
plt.xlabel("Predicted")
plt.ylabel("Actual")
plt.show()
```



Lastly, we can download the test predictions from Driverless AI and examine the forecasted sales vs actual for a selected store and department.

```
[15]: preds_path = h2oai.download(src_path=experiment.test_predictions_path, dest_dir=".")
forecast_predictions = pd.read_csv(preds_path)
forecast_predictions.columns = ["predicted_Weekly_Sales"]

actual = test_data[["Date", "Store", "Dept", "Weekly_Sales"]].reset_index(drop = True)
forecast_predictions = pd.concat([actual, forecast_predictions], axis = 1)
forecast_predictions.head()

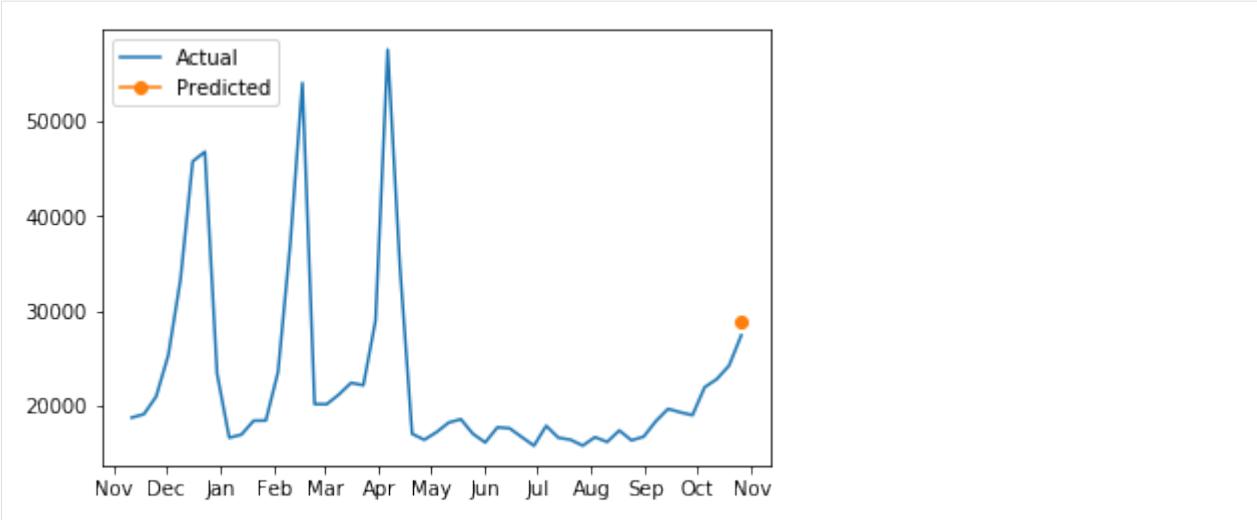
[15]:
      Date  Store  Dept  Weekly_Sales  predicted_Weekly_Sales
0 2012-10-26      1     1        27390.81            28837.857422
1 2012-10-26      1     2        43134.88            43528.121094
2 2012-10-26      1     3        9350.90             8774.910156
3 2012-10-26      1     4       36292.60            35721.511719
4 2012-10-26      1     5       25846.94            23501.814453

[16]: selected_ts = sales_data.loc[(sales_data["Store"] == 1) & (sales_data["Dept"] == 1)].tail(n = 51)

selected_ts_forecast = forecast_predictions.loc[(forecast_predictions["Store"] == 1) &
                                                (forecast_predictions["Dept"] == 1)]
```

```
[17]: # Plot the forecast of a select store and department
years = mdates.MonthLocator()
yearsFmt = mdates.DateFormatter('%b')

fig, ax = plt.subplots()
ax.plot(selected_ts["Date"], selected_ts["Weekly_Sales"], label = "Actual")
ax.plot(selected_ts_forecast["Date"], selected_ts_forecast["predicted_Weekly_Sales"], marker='o', label = "Predicted")
ax.xaxis.set_major_locator(years)
ax.xaxis.set_major_formatter(yearsFmt)
plt.legend(loc='upper left')
plt.show()
```



[1]:

49.4 Driverless AI - Time Series Recipes with Rolling Window

The purpose of this notebook is to show an example of using Driverless AI to train experiments on different subsets of data. This would result in a collection of forecasted values that can be evaluated. The data used in this notebook is a public dataset: S+P 500 Stock Data. In this example, we are using the `all_stocks_5yr.csv` dataset.

49.4.1 Workflow

1. Import data into Python
2. Create function that slices data by index
3. For each slice of data:
 - import data into Driverless AI
 - train an experiment
 - combine test predictions

49.4.2 Import Data

We will begin by importing our data using pandas.

```
[1]: import pandas as pd  
  
stock_data = pd.read_csv("./all_stocks_5yr.csv")  
stock_data.head()  
  
[1]:
```

	date	open	high	low	close	volume	Name
0	2013-02-08	15.07	15.12	14.63	14.75	8407500	AAL
1	2013-02-11	14.89	15.01	14.26	14.46	8882000	AAL
2	2013-02-12	14.45	14.51	14.10	14.27	8126000	AAL
3	2013-02-13	14.30	14.94	14.25	14.66	10259500	AAL
4	2013-02-14	14.94	14.96	13.16	13.99	31879900	AAL

```
[2]: # Convert Date column to datetime  
stock_data["date"] = pd.to_datetime(stock_data["date"], format="%Y-%m-%d")
```

We will add a new column which is the index. We will use this later on to do a rolling window of training and testing. We will use this index instead of the actual date because this data only occurs on weekdays (when the stock market is opened). When you use Driverless AI to perform a forecast, it will forecast the next n days. In this particular case, we never want to forecast Saturday's and Sunday's. We will instead treat our time column as the index of the record.

```
[3]: dates_index = pd.DataFrame(sorted(stock_data["date"].unique()), columns = ["date"])
dates_index["index"] = range(len(dates_index))
stock_data = pd.merge(stock_data, dates_index, on = "date")
stock_data.head()

[3]:
```

	date	open	high	low	close	volume	Name	index
0	2013-02-08	15.0700	15.1200	14.6300	14.7500	8407500	AAL	0
1	2013-02-08	67.7142	68.4014	66.8928	67.8542	158168416	AAPL	0
2	2013-02-08	78.3400	79.7200	78.0100	78.9000	1298137	AAP	0
3	2013-02-08	36.3700	36.4200	35.8250	36.2500	13858795	ABV	0
4	2013-02-08	46.5200	46.8950	46.4600	46.8900	1232802	ABC	0

49.4.3 Create Moving Window Function

Now we will create a function that can split our data by time to create multiple experiments.

We will start by first logging into Driverless AI.

```
[4]: import h2oai_client
import numpy as np
import pandas as pd
# import h2o
import requests
import math
from h2oai_client import Client, ModelParameters

[10]: address = 'http://ip_where_driverless_is_running:12345'
username = 'username'
password = 'password'
h2oai = Client(address = address, username = username, password = password)
# make sure to use the same user name and password when signing in through the GUI
```

Our function will split the data into training and testing based on the training length and testing length specified by the user. It will then run an experiment in Driverless AI and download the test predictions.

```
[11]: def dai_moving_window(dataset, train_len, test_len, target, predictors, index_col, time_group_cols,
accuracy, time, interpretability):

    # Calculate windows for the training and testing data based on the train_len and test_len arguments
    num_dates = max(dataset[index_col])
    num_windows = (num_dates - train_len) // test_len

    windows = []
    for i in range(num_windows):
        train_start_id = i*test_len
        train_end_id = train_start_id + (train_len - 1)
        test_start_id = train_end_id + 1
        test_end_id = test_start_id + (test_len - 1)

        window = {'train_start_index': train_start_id,
                  'train_end_index': train_end_id,
                  'test_start_index': test_start_id,
                  'test_end_index': test_end_id}
        windows.append(window)

    # Split the data by the window
    forecast_predictions = pd.DataFrame([])
    for window in windows:
        train_data = dataset[(dataset[index_col] >= window.get("train_start_index")) &
                             (dataset[index_col] <= window.get("train_end_index"))]

        test_data = dataset[(dataset[index_col] >= window.get("test_start_index")) &
                            (dataset[index_col] <= window.get("test_end_index"))]

        # Get the Driverless AI forecast predictions
        window_preds = dai_get_forecast(train_data, test_data, predictors, target, index_col, time_group_cols,
                                         accuracy, time, interpretability)
        forecast_predictions = forecast_predictions.append(window_preds)

    return forecast_predictions

[12]: def dai_get_forecast(train_data, test_data, predictors, target, index_col, time_group_cols,
accuracy, time, interpretability):

    # Save dataset
    train_path = "./train_data.csv"
    test_path = "./test_data.csv"
    keep_cols = predictors + [target, index_col] + time_group_cols
    train_data[keep_cols].to_csv(train_path)
    test_data[keep_cols].to_csv(test_path)
```

(continues on next page)

(continued from previous page)

```
# Add datasets to Driverless AI
train_dai = h2oai.upload_dataset_sync(train_path)
test_dai = h2oai.upload_dataset_sync(test_path)

# Run Driverless AI Experiment
experiment = h2oai.start_experiment_sync(dataset_key = train_dai.key,
                                         testset_key = test_dai.key,
                                         target_col = target,
                                         cols_to_drop = [],
                                         is_classification = False,
                                         accuracy = accuracy,
                                         time = time,
                                         interpretability = interpretability,
                                         scorer = "RMSE",
                                         seed = 1234,
                                         time_col = index_col,
                                         time_groups_columns = time_group_cols,
                                         num_prediction_periods = test_data[index_col].nunique(),
                                         num_gap_periods = 0)

# Download the predictions on the test dataset
test_predictions_path = h2oai.download(experiment.test_predictions_path, "./")
test_predictions = pd.read_csv(test_predictions_path)
test_predictions.columns = ["Prediction"]

# Add predictions to original test data
keep_cols = [target, index_col] + time_group_cols
test_predictions = pd.concat([test_data[keep_cols].reset_index(drop=True), test_predictions], axis = 1)

return test_predictions
```

```
[13]: predictors = ["Name", "index"]
target = "close"
index_col = "index"
time_group_cols = ["Name"]
```

```
[1]: # We will filter the dataset to the first 1030 dates for demo purposes
filtered_stock_data = stock_data[stock_data["index"] <= 1029]
forecast_predictions = dai_moving_window(filtered_stock_data, 1000, 3, target, predictors, index_col, time_group_cols,
                                         accuracy = 1, time = 1, interpretability = 1)
```

```
[25]: forecast_predictions.head()

[25]:
   close  index  Name  Prediction
0    44.90    1000   AAL    48.050527
1   121.63    1000   AAPL   119.485352
2   164.63    1000    AAP   167.960700
3    60.43    1000   ABBV   60.784213
4    83.62    1000    ABC   86.939174
```

```
[26]: # Calculate some error metric
mae = (forecast_predictions[target] - forecast_predictions["Prediction"]).abs().mean()
print("Mean Absolute Error: ${:, .2f}".format(mae))

Mean Absolute Error: $6.79
```

```
[ ]:
```

49.5 Driverless AI NLP Demo - Airline Sentiment Dataset

In this notebook, we will see how to use Driverless AI python client to build text classification models using the Airline sentiment twitter dataset.

Import the necessary python modules to get started including the Driverless AI client. If not already installed, please download and install the python client from Driverless AI GUI.

This notebook was tested in Driverless AI version 1.8.2.

```
[1]: import pandas as pd
from sklearn import model_selection
from h2oai_client import Client
```

The below code downloads the twitter airline sentiment dataset and save it in the current folder.

```
[2]: ! wget https://www.figure-eight.com/wp-content/uploads/2016/03/Airline-Sentiment-2-w-AA.csv
--2020-01-17 09:38:39-- https://www.figure-eight.com/wp-content/uploads/2016/03/Airline-Sentiment-2-w-AA.csv
Resolving www.figure-eight.com (www.figure-eight.com)... 54.86.123.68, 35.169.155.50
Connecting to www.figure-eight.com (www.figure-eight.com)|54.86.123.68|:443... connected.
```

(continues on next page)

(continued from previous page)

```
HTTP request sent, awaiting response... 200 OK
Length: 3704908 (3.5M) [application/octet-stream]
Saving to: 'Airline-Sentiment-2-w-AA.csv.l'

Airline-Sentiment-2 100%[=====] 3.53M 2.26MB/s in 1.6s
2020-01-17 09:38:41 (2.26 MB/s) - 'Airline-Sentiment-2-w-AA.csv.l' saved [3704908/3704908]
```

We can now split the data into training and testing datasets.

```
[3]: al = pd.read_csv("Airline-Sentiment-2-w-AA.csv", encoding='ISO-8859-1')
train_al, test_al = model_selection.train_test_split(al, test_size=0.2, random_state=2018)
train_al.to_csv("train_airline_sentiment.csv", index=False)
test_al.to_csv("test_airline_sentiment.csv", index=False)
```

The first step is to establish a connection to Driverless AI using Client. Please key in your credentials and the url address.

```
[4]: address = 'http://ip_where_driverless_is_running:12345'
username = 'username'
password = 'password'
h2oai = Client(address = address, username = username, password = password)
# # make sure to use the same user name and password when signing in through the GUI
```

Read the train and test files into Driverless AI using the upload_dataset_sync command.

```
[5]: train_path = './train_airline_sentiment.csv'
test_path = './test_airline_sentiment.csv'

train = h2oai.upload_dataset_sync(train_path)
test = h2oai.upload_dataset_sync(test_path)
```

Now let us look at some basic information about the dataset.

```
[6]: print('Train Dataset: ', len(train.columns), 'x', train.row_count)
print('Test Dataset: ', len(test.columns), 'x', test.row_count)

[c.name for c in train.columns]

Train Dataset: 20 x 11712
Test Dataset: 20 x 2928

[6]: ['_unit_id',
 '_golden',
 '_unit_state',
 '_trusted_judgments',
 '_last_judgment_at',
 'airline_sentiment',
 'airline_sentiment:confidence',
 'negativereson',
 'negativereson:confidence',
 'airline',
 'airline_sentiment_gold',
 'name',
 'negativereson_gold',
 'retweet_count',
 'text',
 'tweet_coord',
 'tweet_created',
 'tweet_id',
 'tweet_location',
 'user_timezone']
```

We just need two columns for our experiment. `text` which contains the text of the tweet and `airline_sentiment` which contains the sentiment of the tweet (target column). We can drop the remaining columns for this experiment.

We will enable tensorflow models and transformations to take advantage of CNN based text features.

```
[7]: exp_preview = h2oai.get_experiment_preview_sync(
    dataset_key=train.key
    , validset_key=''
    , target_col='airline_sentiment'
    , classification=True
    , dropped_cols=["_unit_id", "_golden", "_unit_state", "_trusted_judgments", "_last_judgment_at",
        "airline_sentiment:confidence", "negativereson", "negativereson:confidence", "airline",
        "airline_sentiment_gold", "name", "negativereson_gold", "retweet_count",
        "tweet_coord", "tweet_created", "tweet_id", "tweet_location", "user_timezone"]
    , accuracy=6
    , time=4
    , interpretability=5
    , is_time_series=False
    , time_col=''
    , enable_gpus=True
    , reproducible=False
    , resumed_experiment_id=''
    , config_overrides="\"enable_tensorflow='on'"
    )
```

(continues on next page)

(continued from previous page)

```
enable_tensorflow_charcnn='on'
enable_tensorflow_textcnn='on'
enable_tensorflow_textbigru='on'
"""
)
exp_preview

[7]: ['ACCURACY [6/10]:',
'- Training data size: *11,712 rows, 2 cols*',
'- Feature evolution: *[LightGBM, TensorFlow, XGBoostGBM]*, *3-fold CV**, 2 reps*',
'- Final pipeline: *Ensemble (6 models), 3-fold CV*',
 '',
'TIME [4/10]:',
'- Feature evolution: *4 individuals*, up to *46 iterations*',
'- Early stopping: After *5* iterations of no improvement',
 '',
'INTERPRETABILITY [5/10]:',
'- Feature pre-pruning strategy: None',
'- Monotonicity constraints: disabled',
'- Feature engineering search space: [CVTargetEncode, Frequent, TextBiGRU, TextCNN, TextCharCNN, Text]',
 '',
'[LightGBM, TensorFlow, XGBoostGBM] models to train:',
'- Model and feature tuning: *144*',
'- Feature evolution: *504*',
'- Final pipeline: *6*',
 '',
'Estimated runtime: *minutes*',
'Auto-click Finish/Abort if not done in: *1 day*/*7 days*']
```

Please note that the `Text` and `TextCNN` features are enabled for this experiment.

Now we can start the experiment.

```
[8]: model = h2oai.start_experiment_sync(
    dataset_key=train.key,
    testset_key=test.key,
    target_col='airline_sentiment',
    scorer='F1',
    is_classification=True,
    cols_to_drop=["_unit_id", "_golden", "_unit_state", "_trusted_judgments", "_last_judgment_at",
                  "airline_sentiment:confidence", "negativereson", "negativereson:confidence", "airline",
                  "airline_sentiment_gold", "name", "negativereson_gold", "retweet_count",
                  "tweet_coord", "tweet_created", "tweet_id", "tweet_location", "user_timezone"],
    accuracy=6,
    time=2,
    interpretability=5,
    enable_gpus=True,
    config_overrides="""
        enable_tensorflow='on'
        enable_tensorflow_charcnn='on'
        enable_tensorflow_textcnn='on'
        enable_tensorflow_textbigru='on'
        """
)
```

```
[9]: print('Modeling completed for model ' + model.key)

Modeling completed for model ce5935e6-3950-11ea-9465-0242ac110002
```

```
[10]: logs = h2oai.download(model.log_file_path, '.')
print('Logs available at', test_preds)

Logs available at ./test_preds.csv
```

We can download the predictions to the current folder.

```
[11]: test_preds = h2oai.download(model.test_predictions_path, '.')
print('Test set predictions available at', test_preds)

Test set predictions available at ./test_preds.csv
```

```
[ ]:
```

49.6 Driverless AI Autoviz Python Client Example

This example shows how to use the Autoviz Python client in Driverless AI to retrieve graphs in Vega Lite format. (See <https://vega.github.io/vega-lite/>.)

When running the Autoviz Python client in Jupyter Notebook you can use <https://github.com/ipyvega> (installed through pip) and render the graph directly in Jupyter notebook. You can also copy paste the result e.g. to <https://vega.github.io/vega-editor/?mode=vega-lite>. The final graph can then be downloaded to png/svg/json.

The end of this document includes the available API methods.

49.6.1 Prerequisites

Using DriverlessAI Autoviz python client doesn't require any additional packages, other than [DriverlessAI client](#). However, if you are using Jupyter notebooks or labs, installing [Vega](#) package can help better user experience, as it allows you to render the produced graphs directly inside Jupyter environment. In addition, it provides options to download the generated files in SVG, PNG or JSON formats.

49.6.2 Initialization

To initialize the Autoviz Python client, follow the same steps as when initializing the client for new experiment. You need to import the `Client` and initialize it, providing the DriverlessAI host address and login credentials.

```
[1]: from h2oai_client import Client
address = 'http://ip.where_driverless_is_running:12345'
username = 'username'
password = 'password'
cli = Client(address=address, username=username, password=password)
```

49.6.3 Upload the dataset

```
[18]: dataset_local_path = '/data/Kaggle/CreditCard/CreditCard-train.csv'
dataset = cli.create_dataset_sync(dataset_local_path)
```

49.6.4 Request specific visualizations

```
[19]: hist = cli.autoviz.get_histogram(dataset.key, variable_name='PAY_0')

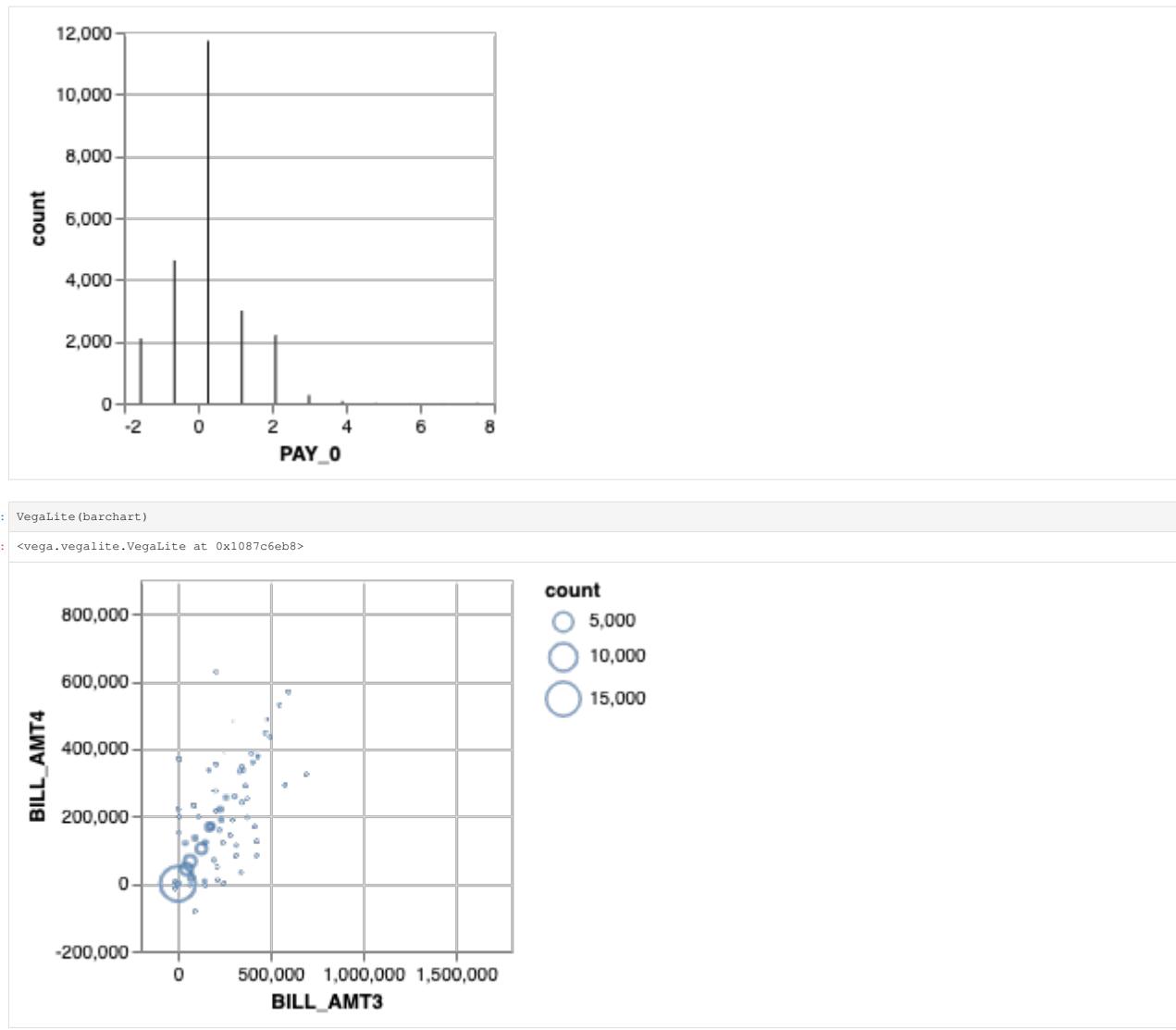
[29]: barchart = cli.autoviz.get_scatterplot(dataset.key, x_variable_name='BILL_AMT3', y_variable_name='BILL_AMT4')
```

49.6.5 Visualize using Vega (Optional)

All of the methods provided in `Client.autoviz` return graphics in [Vega Lite \(v3\)](#) format. In order to visualize them, you can either paste the returned graph (`hist`) into [Online editor](#), or you can utilize the Python [Vega package](#), which can visualize the charts directly in Jupyter environment.

```
[32]: from vega import VegaLite

[33]: VegaLite(hist)
[33]: <vega.vegalite.VegaLite at 0x1098b90f0>
```



49.6.6 API Methods

```

get_histogram(
    dataset_key: str,
    variable_name: str,
    number_of_bars: int = 0,
    transformation: str = "none",
    mark: str = "bar",
) -> dict:
    """
    Required Keyword arguments:
    dataset_key -- str, Key of visualized dataset in DriverlessAI
    variable_name -- str, name of variable

    Optional Keyword arguments:
    number_of_bars -- int, number of bars
    transformation -- str, default value is "none"
        (otherwise, "log" or "square_root")
    mark -- str, default value is "bar" (use "area" to get a density polygon)
    """

get_scatterplot(
    dataset_key: str,

```

(continues on next page)

(continued from previous page)

```

x_variable_name: str,
y_variable_name: str,
mark: str = "point",
) -> dict:
"""

-----
Required Keyword arguments:
-----
dataset_key -- str, Key of visualized dataset in DriverlessAI
x_variable_name -- str, name of x variable
(y variable assumed to be counts if no y variable specified)
y_variable_name -- str, name of y variable

-----
Optional Keyword arguments:
-----

get_bar_chart(
    dataset_key: str,
    x_variable_name: str,
    y_variable_name: str = "",
    transpose: bool = False,
    mark: str = "bar",
) -> dict:
"""

-----
Required Keyword arguments:
-----
dataset_key -- str, Key of visualized dataset in DriverlessAI
x_variable_name -- str, name of x variable
(y variable assumed to be counts if no y variable specified)

-----
Optional Keyword arguments:
-----
y_variable_name -- str, name of y variable
transpose -- Boolean, default value is false
mark -- str, default value is "bar" (use "point" to get a Cleveland dot plot)
"""

get_parallel_coordinates_plot(
    dataset_key: str,
    variable_names: list = [],
    permute: bool = False,
    transpose: bool = False,
    cluster: bool = False,
) -> dict:
"""

-----
Required Keyword arguments:
-----
dataset_key -- str, Key of visualized dataset in DriverlessAI

-----
Optional Keyword arguments:
-----
variable_names -- str, name of variables
(if no variables specified, all in dataset will be used)
permute -- Boolean, default value is false
(if true, use SVD to permute variables)
transpose -- Boolean, default value is false
cluster -- Boolean, k-means cluster variables and color plot by cluster IDs,
default value is false
"""

get_heatmap(
    dataset_key: str,
    variable_names: list = [],
    permute: bool = False,
    transpose: bool = False,
    matrix_type: str = "rectangular",
) -> dict:
"""

-----
Required Keyword arguments:
-----
dataset_key -- str, Key of visualized dataset in DriverlessAI

-----
Optional Keyword arguments:
-----
variable_names -- str, name of variables
(if no variables specified, all in dataset will be used)
permute -- Boolean, default value is false
(if true, use SVD to permute rows and columns)
transpose -- Boolean, default value is false
matrix_type -- str, default value is "rectangular" (alternative is "symmetric")
"""

get_boxplot(
    dataset_key: str,
    variable_name: str,
    group_variable_name: str = "",
    transpose: bool = False,
) -> dict:
"""

-----
Required Keyword arguments:
-----
dataset_key -- str, Key of visualized dataset in DriverlessAI
variable_name -- str, name of variable for box

-----
Optional Keyword arguments:
-----
```

(continues on next page)

(continued from previous page)

```
-----  
group_variable_name -- str, name of grouping variable  
transpose -- Boolean, default value is false  
"""  
  
get_linear_regression(  
    dataset_key: str,  
    x_variable_name: str,  
    y_variable_name: str,  
    mark: str = "point",  
) -> dict:  
"""  
  
-----  
Required Keyword arguments:  
-----  
dataset_key -- str, Key of visualized dataset in DriverlessAI  
x_variable_name -- str, name of x variable  
    (y variable assumed to be counts if no y variable specified)  
y_variable_name -- str, name of y variable  
  
-----  
Optional Keyword arguments:  
-----  
mark -- str, default value is "point" (alternative is "square")  
"""  
  
get_loess_regression(  
    dataset_key: str,  
    x_variable_name: str,  
    y_variable_name: str,  
    mark: str = "point",  
    bandwidth: float = 0.5,  
) -> dict:  
"""  
  
-----  
Required Keyword arguments:  
-----  
dataset_key -- str, Key of visualized dataset in DriverlessAI  
x_variable_name -- str, name of x variable  
    (y variable assumed to be counts if no y variable specified)  
y_variable_name -- str, name of y variable  
  
-----  
Optional Keyword arguments:  
-----  
mark -- str, default value is "point" (alternative is "square")  
bandwidth -- float, number in the (0,1)  
    interval denoting proportion of cases in smoothing window (default is 0.5)  
"""  
  
get_dotplot(  
    dataset_key: str,  
    variable_name: str,  
    mark: str = "point",  
) -> dict:  
"""  
  
-----  
Required Keyword arguments:  
-----  
dataset_key -- str, Key of visualized dataset in DriverlessAI  
variable_name -- str, name of variable on which dots are calculated  
  
-----  
Optional Keyword arguments:  
-----  
mark -- str, default value is "point" (alternative is "square" or "bar")  
"""  
  
get_distribution_plot(  
    dataset_key: str,  
    x_variable_name: str,  
    y_variable_name: str = "",  
    subtype: str = "probability_plot",  
    distribution: str = "normal",  
    mark: str = "point",  
    transpose: bool = False,  
) -> dict:  
"""  
  
-----  
Required Keyword arguments:  
-----  
dataset_key -- str, Key of visualized dataset in DriverlessAI  
x_variable_name -- str, name of x variable  
  
-----  
Optional Keyword arguments:  
-----  
y_variable_name -- str, name of y variable for quantile plot  
subtype -- str "probability_plot" or "quantile_plot"  
    (default is "probability_plot" done on x variable)  
distribution -- str, type of distribution, "normal" or "uniform"  
    ("normal" is default)  
mark -- str, default value is "point" (alternative is "square")  
transpose -- Boolean, default value is false  
"""
```

THE R CLIENT

This section describes how to install the Driverless AI R Client. It also provides an example tutorial showing how to use the Driverless AI R client. More information about how to use the Driverless AI R Client is available in the Driverless AI R Client documentation.

50.1 Installing the R Client

The R Client is available on the Driverless AI UI and from the command line. The installation process includes downloading the R client and then installing the source package.

50.1.1 Prerequisites

The R client requires R version 3.3 or greater. In addition, the following R packages must be installed:

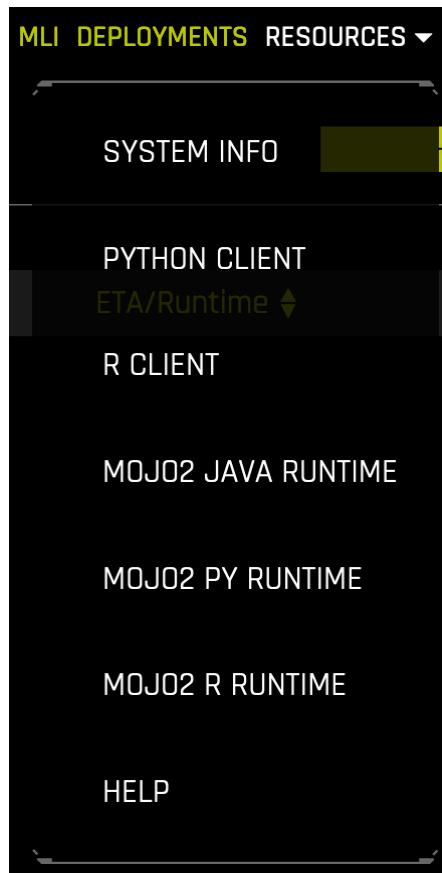
- `Rcurl`
- `jsonlite`
- `rlang`
- `methods`

50.1.2 Download the R Client

The R Client can be downloaded from within Driverless AI or from the command line.

Download from UI

On the Driverless AI top menu, select the **RESOURCES > R CLIENT** link. This downloads the `dai_<version>.tar.gz` file.



Download from Command Line

The Driverless AI R client is exposed as the `/clients/` HTTP end point. This can be accessed via the command line:

```
wget --trust-server-names http://<Driverless AI address>/clients/r
```

50.1.3 Install the Source Package

After you have downloaded the R client, the next step is to install the source package in R. This can be done by running the following command in R.

```
install.packages('~/Downloads/dai_VERSION.tar.gz', type = 'source', repos = NULL)
```

After the package is installed, you can run the available `dai-tutorial` vignette to see an example of how to use the client:

```
vignette('dai-tutorial', package = 'dai')
```

50.2 R Client Tutorial

This tutorial describes how to use the Driverless AI R client package to use and control the Driveless AI platform. It covers the main predictive data-science workflow, including:

1. Data load
2. Automated feature engineering and model tuning
3. Model inspection
4. Predicting on new data
5. Managing the datasets and models

Note: These steps assume that you have entered your license key in the Driverless AI UI.

50.2.1 Loading the Data

Before we can start working with the Driverless.ai platform (DAI), we have to import the package and initialize the connection:

```
library(dai)
dai.connect(uri = "http://localhost:12345", username = 'h2oai', password = 'h2oai')
creditcard <- dai.create_dataset('/data/smldata/kaggle/CreditCard/creditcard_train_cat.csv')
#>
| |
| |
| ====== | 0%
| ====== | 24%
| ====== | 100%
```

The function `dai.create_dataset()` loads the data located at the machine that hosts DAI. The above command assumes that the `creditcard_train_cat.csv` is in the `/data` folder on the machine running Driverless AI. This file is available at https://s3.amazonaws.com/h2o-public-test-data/smldata/kaggle/CreditCard/creditcard_train_cat.csv.

If you want to upload the data located at your workstation, use `dai.upload_dataset()` instead.

If you already have the data loaded into R `data.frame`, you can simply convert it into a DAIFrame. For example:

```
iris.dai <- as.DAIFrame(iris)
#>
| |
| |
| ====== | 0%
| ====== | 100%

print(iris.dai)
#> DAI frame '7c38cb84-5baa-11e9-a50b-b938de969cdb': 150 obs. of 5 variables
#> File path: ./tmp/7c38cb84-5baa-11e9-a50b-b938de969cdb/iris9elf15d2df00.csv.1554912339.9424415.bin
```

You can switch off the progress bar whenever it is displayed by setting `progress = FALSE`.

Upon creation of the dataset, you can display the basic information and summary statistics by calling generics `print` and `summary`:

```
print(creditcard)
#> DAI frame '7abe28b2-5baa-11e9-a50b-b938de969cdb': 23999 obs. of 25 variables
#> File path: tests/smldata/kaggle/CreditCard/creditcard_train_cat.csv

summary(creditcard)
#>
#> 1           ID      num_classes 0          TRUE 23999
#> 2          LIMIT_BAL num_classes 79         TRUE 23999
#> 3            SEX      num_classes 2          FALSE 23999
#> 4        EDUCATION num_classes 4          FALSE 23999
#> 5       MARRIAGE num_classes 4          FALSE 23999
```

(continues on next page)

Using Driverless AI, Release 1.9.0.1

(continued from previous page)

```

#> 6          AGE      55      TRUE 23999
#> 7          PAY_1     11      TRUE 23999
#> 8          PAY_2     11      TRUE 23999
#> 9          PAY_3     11      TRUE 23999
#> 10         PAY_4     11      TRUE 23999
#> 11         PAY_5     10      TRUE 23999
#> 12         PAY_6     10      TRUE 23999
#> 13         BILL_AMT1    0      TRUE 23999
#> 14         BILL_AMT2    0      TRUE 23999
#> 15         BILL_AMT3    0      TRUE 23999
#> 16         BILL_AMT4    0      TRUE 23999
#> 17         BILL_AMT5    0      TRUE 23999
#> 18         BILL_AMT6    0      TRUE 23999
#> 19         PAY_AMT1    0      TRUE 23999
#> 20         PAY_AMT2    0      TRUE 23999
#> 21         PAY_AMT3    0      TRUE 23999
#> 22         PAY_AMT4    0      TRUE 23999
#> 23         PAY_AMT5    0      TRUE 23999
#> 24         PAY_AMT6    0      TRUE 23999
#> 25 DEFAULT_PAYMENT_NEXT_MONTH 2      TRUE 23999
#>           mean        std   min    max unique freq
#> 1 12000.6928.058891204666 1 23999 23999 1
#> 2 165498.715779824 129130.743065318 10000 1000000 79 2740
#> 3                                     2 8921
#> 4                                     4 11360
#> 5                                     4 12876
#> 6 35.3808492020501 9.2710457493384 21 79 55 1284
#> 7 -0.00312513021375891 1.12344874325651 -2 8 11 11738
#> 8 -0.123463477644902 1.20059118344043 -2 8 11 12543
#> 9 -0.154756448185341 1.20405796618856 -2 8 11 12576
#> 10 -0.211675486478603 1.16657279943005 -2 8 11 13250
#> 11 -0.252885553697371 1.13700672904 -2 8 10 135520
#> 12 -0.27801583615992 1.1581916495226 -2 8 10 12876
#> 13 50598.9286636943 72650.1978092856 -165580 964511 18717 1607
#> 14 48648.943139297471 70365.3956426641 -69777 983931 18367 2049
#> 15 46368.9035376474 68194.7195202748 -157264 1664089 18131 2325
#> 16 42369.8728280345 63071.4551670874 -170000 891586 17719 2547
#> 17 40002.3330972124 60345.7282797424 -81334 927171 17284 2840
#> 18 38565.26663631098 59156.5011434754 -339603 961664 16906 3258
#> 19 5543.09804575191 15068.86272958 0 505000 6918 4270
#> 20 5815.52852202175 20797.443884891 0 1684259 6839 4362
#> 21 4969.43139297471 16095.9292948255 0 896040 6424 4853
#> 22 4743.65686070253 14883.5548720259 0 497000 6028 5200
#> 23 4783.64369348723 15270.7039035392 0 417990 5984 5407
#> 24 5189.57360723363 17630.7185745277 0 528666 5988 5846
#> 25 0.223717654902288 0.41674368928609 FALSE TRUE 2 5369
#>
#> num_hist_ticks
#> 1 21599.2, 23999.0
#> 2 901000.0, 1000000.0
#> 3
#> 4
#> 5
#> 6 1.0, 2400.8, 4800.6, 7200.400000000001, 9600.2, 12000.0, 14399.800000000001, 16799.600000000002, 19199.4
#> 7 21.0, 26.8, 32.6, 38.4, 44.2, 50.0, 55.8, 61.6, 67.4, 73.
#> 8 -2, -1, 0, 1, 2,
#> 9 -2, -1, 0, 1, 2,
#> 10 -2, -1, 0, 1, 2,
#> 11 -2, -1, 0, 2,
#> 12 -2, -1, 0, 2,
#> 13 -2, -1, 0, 1, 2,
#> 14 -2, -1, 0, 1, 2,
#> 15 -2, -1, 0, 2,
#> 16 -2, -1, 0, 2,
#> 17 -2, -1, 0, 2,
#> 18 -2, -1, 0, 2,
#> 19 0.0, 50500.0, 101000.0, 151500.0, 202000.0, 252500.0, 303000.0, 353500.0, 404000.0
#> 20 0.0, 168425.9, 336851.8, 505277.6999999995, 673703.6, 842129.5, 101055.399999999, 1178981.3, 1347407.2, 1515833.
#> 21 0.0, 89604.0, 179208.0, 268812.0, 358416.0, 448020.0, 537624.0, 627228.0, 716832.0
#> 22 0.0, 49700.0, 99400.0, 149100.0, 198800.0, 248500.0, 298200.0, 347900.0, 397600.0
#> 23 0.0, 41799.0, 83598.0, 125397.0, 167196.0, 208995.0, 250794.0, 292593.0, 334392.0
#> 24 0.0, 52866.6, 105733.2, 158599.8, 211466.4, 264333.0, 317199.6, 370066.2, 422932.8, 475799.
#> 25 False, True
#> num_hist_counts top
#> 1 2400, 2400, 2400, 2400, 2399, 2400, 2400, 2400, 2400, 2400
#> 2 10151, 6327, 3965, 2149, 1251, 96, 44, 15, 0, 1
#> 3
#> 4 female
#> 5 university
#> 6 single
#> 7 4285, 6546, 5187, 3780, 2048, 1469, 501, 147, 34, 2
#> 8 2086, 4625, 11738, 2994, 2185, 254, 66, 17, 9, 7, 18
#> 9 2953, 4886, 12543, 20, 3204, 268, 76, 21, 9, 18, 1

```

(continues on next page)

(continued from previous page)

```

#> 9      3197, 4787, 12576, 4, 3121, 183, 64, 17, 21, 27, 2
#> 10     3382, 4555, 13250, 2, 2515, 158, 55, 29, 5, 46, 2
#> 11     3539, 4482, 13520, 2178, 147, 71, 11, 3, 47, 1
#> 12     3818, 4722, 12876, 2324, 158, 37, 9, 16, 37, 2
#> 13     2, 17603, 4754, 1193, 316, 111, 18, 1, 0, 1
#> 14     14571, 7214, 1578, 429, 155, 43, 7, 1, 0, 1
#> 15     12977, 10150, 767, 99, 5, 0, 0, 0, 0, 1
#> 16     2, 16619, 5775, 1181, 311, 89, 20, 1, 0, 1
#> 17     12722, 9033, 1720, 374, 113, 31, 4, 0, 1, 1
#> 18     1, 1, 18312, 4788, 745, 131, 19, 1, 0, 1
#> 19     23643, 249, 56, 26, 14, 8, 0, 1, 1, 1
#> 20     23936, 50, 11, 1, 0, 0, 0, 0, 0, 1
#> 21     23836, 130, 20, 9, 3, 0, 0, 0, 0, 1
#> 22     23647, 235, 65, 29, 11, 5, 4, 0, 2, 1
#> 23     23588, 234, 94, 40, 22, 7, 3, 8, 0, 3
#> 24     23605, 235, 77, 56, 15, 5, 1, 3, 0, 2
#> 25     18630, 5369
#>
#> nonnum_hist_ticks nonnum_hist_counts
#> 1
#> 2
#> 3      female, male, Other    15078, 8921, 0
#> 4      university, graduate, Other 11360, 8442, 4197
#> 5      single, married, Other 12876, 10813, 310
#> 6
#> 7
#> 8
#> 9
#> 10
#> 11
#> 12
#> 13
#> 14
#> 15
#> 16
#> 17
#> 18
#> 19
#> 20
#> 21
#> 22
#> 23
#> 24
#> 25

```

A couple of other generics works as usual on a DAIFrame: dim, head, and format.

```

dim(creditcard)
#> [1] 23999   25

head(creditcard, 10)
#> ID LIMIT_BAL SEX EDUCATION MARRIAGE AGE PAY_1 PAY_2 PAY_3 PAY_4
#> 1 1 20000 female university married 24 2 2 -1 -1
#> 2 2 120000 female university single 26 -1 2 0 0
#> 3 3 90000 female university single 34 0 0 0 0
#> 4 4 50000 female university married 37 0 0 0 0
#> 5 5 50000 male university married 57 -1 0 -1 0
#> 6 6 50000 male graduate single 37 0 0 0 0
#> 7 7 50000 male graduate single 29 0 0 0 0
#> 8 8 100000 female university single 23 0 -1 -1 0
#> 9 9 140000 female highschool married 28 0 0 2 0
#> 10 10 20000 male highschool single 35 -2 2 2 -2
#> PAY_5 PAY_6 BILL_AMT1 BILL_AMT2 BILL_AMT3 BILL_AMT4 BILL_AMT5 BILL_AMT6
#> 1 -2 -2 3913 3102 689 0 0 0
#> 2 0 2 2682 1725 2682 3272 3455 3261
#> 3 0 0 29239 14027 13559 14331 14948 15549
#> 4 0 0 46990 48233 49291 28314 28959 29547
#> 5 0 0 8617 5670 53835 20940 19146 19131
#> 6 0 0 64400 57069 57608 19394 19619 20024
#> 7 0 0 367965 412023 445007 542653 483003 473944
#> 8 0 -1 11876 380 601 221 -159 567
#> 9 0 0 11285 14096 12108 12211 11793 3719
#> 10 -1 -1 0 0 0 0 13007 13912
#> PAY_AMT1 PAY_AMT2 PAY_AMT3 PAY_AMT4 PAY_AMT5 PAY_AMT6
#> 1 0 689 0 0 0 0
#> 2 0 1000 1000 1000 0 2000
#> 3 1518 1500 1000 1000 1000 5000
#> 4 2000 2019 1200 1100 1069 1000
#> 5 2000 36681 10000 9000 689 679
#> 6 2500 1815 657 1000 1000 800
#> 7 55000 40000 38000 20239 13750 13770
#> 8 380 601 0 581 1687 1542
#> 9 3329 0 432 1000 1000 1000
#> 10 0 0 0 13007 1122 0
#> DEFAULT_PAYMENT_NEXT_MONTH
#> 1 TRUE
#> 2 TRUE
#> 3 FALSE
#> 4 FALSE
#> 5 FALSE
#> 6 FALSE
#> 7 FALSE
#> 8 FALSE
#> 9 FALSE
#> 10 FALSE

```

You cannot, however, use DAIFrame to access all its data, nor can you use it to modify the data. It only represents the data set loaded into the DAI platform. The head function gives access only to example data:

```

creditcard$example_data[1:10, ]
#>   ID LIMIT_BAL SEX EDUCATION MARRIAGE AGE PAY_1 PAY_2 PAY_3 PAY_4
#> 1 1 20000 female university married 24 2 2 -1 -1
#> 2 2 120000 female university single 26 -1 2 0 0
#> 3 3 90000 female university single 34 0 0 0 0
#> 4 4 50000 female university married 37 0 0 0 0
#> 5 5 50000 male university married 57 -1 0 -1 0
#> 6 6 50000 male graduate single 37 0 0 0 0
#> 7 7 50000 male graduate single 29 0 0 0 0
#> 8 8 100000 female university single 23 0 -1 -1 0
#> 9 9 140000 female highschool married 28 0 0 2 0
#> 10 10 20000 male highschool single 35 -2 -2 -2 -2
#> PAY_5 PAY_6 BILL_AMT1 BILL_AMT2 BILL_AMT3 BILL_AMT4 BILL_AMT5 BILL_AMT6
#> 1 -2 -2 3913 3102 689 0 0 0
#> 2 0 2 2682 1725 2682 3272 3455 3261
#> 3 0 0 29239 14027 13559 14331 14948 15549
#> 4 0 0 46990 48233 49291 28314 28959 29547
#> 5 0 0 8617 5670 53835 20940 19146 19131
#> 6 0 0 64400 57069 57608 19394 19619 20024
#> 7 0 0 367965 412023 445007 542653 483003 473944
#> 8 0 -1 11876 380 601 221 -159 567
#> 9 0 0 11285 14096 12108 12211 11793 3719
#> 10 -1 -1 0 0 0 0 13007 13912
#> PAY_AMT1 PAY_AMT2 PAY_AMT3 PAY_AMT4 PAY_AMT5 PAY_AMT6
#> 1 0 689 0 0 0 0
#> 2 0 1000 1000 1000 0 2000
#> 3 1518 1500 1000 1000 1000 5000
#> 4 2000 2019 1200 1100 1069 1000
#> 5 2000 36681 10000 9000 689 679
#> 6 2500 1815 657 1000 1000 800
#> 7 55000 40000 38000 20239 13750 13770
#> 8 380 601 0 581 1687 1542
#> 9 3329 0 432 1000 1000 1000
#> 10 0 0 0 13007 1122 0
#> DEFAULT_PAYMENT_NEXT_MONTH
#> 1 TRUE
#> 2 TRUE
#> 3 FALSE
#> 4 FALSE
#> 5 FALSE
#> 6 FALSE
#> 7 FALSE
#> 8 FALSE
#> 9 FALSE
#> 10 FALSE

```

A dataset can be split into e.g. training and test sets directly in R:

```

creditcard.splits <- dai.split_dataset(creditcard,
                                       output_name1 = 'train',
                                       output_name2 = 'test',
                                       ratio = .8,
                                       seed = 25,
                                       progress = FALSE)

```

In this case the creditcard.splits is a list with two elements with names “train” and “test”, where 80% of the data went into train and 20% of the data went into test.

```

creditcard.splits$strain
#> DAI frame: '7cf3024c-5baa-11e9-a50b-b938de969cdb': 19199 obs. of 25 variables
#> File path: ./tmp/7cf3024c-5baa-11e9-a50b-b938de969cdb/train.1554912341.0864356.bin

creditcard.splits$test
#> DAI frame: '7cf613a6-5baa-11e9-a50b-b938de969cdb': 4800 obs. of 25 variables
#> File path: ./tmp/7cf613a6-5baa-11e9-a50b-b938de969cdb/test.1554912341.0966916.bin

```

By default it yields a simple random sample, but you can do stratified or time-based splits as well. See the function’s documentation for more details.

50.2.2 Automated Feature Engineering and Model Tuning

One of the main strengths of Driverless AI is the fully automated feature engineering along with hyperparameter tuning, model selection and ensembling. The function `dai.train()` executes the experiment that results in a `DAIModel` instance that represents the model.

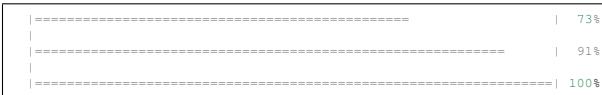
```

model <- dai.train(training_frame = creditcard.splits$strain,
                    testing_frame = creditcard.splits$test,
                    target_col = 'DEFAULT_PAYMENT_NEXT_MONTH',
                    is_classification = T,
                    is_timeseries = F,
                    accuracy = 1, time = 1, interpretability = 10,
                    seed = 25)
#>
| | 0%
| |
| ====== | 40%
|

```

(continues on next page)

(continued from previous page)



If you do not specify the accuracy, time, or interpretability, they will be suggested by the DAI platform. (See `dai.suggest_model_params()`.)

50.2.3 Model Inspection

As with DAIFrame, generic methods such as `print`, `format`, `summary`, and `predict` work with DAIModel:

```
print(model)
#> Status: Complete
#> Experiment: 7e2b70ae-5baa-11e9-a50b-b938de969cddb, 2019-04-10 18:06, 1.7.0+local_0c7d019-dirty
#> Settings: 1/1/10, seed=25, GPUs enabled
#> Train data: train (19199, 25)
#> Validation data: N/A
#> Test data: test (4800, 24)
#> Target column: DEFAULT_PAYMENT_NEXT_MONTH (binary, 22.366% target class)
#> System specs: Linux, 126 GB, 40 CPU cores, 2/2 GPUs
#> Max memory usage: 0.406 GB, 0.167 GB GPU
#> Recipe: AutoDL (2 iterations, 2 individuals)
#> Validation scheme: stratified, 1 internal holdout
#> Feature engineering: 33 features scored (18 selected)
#> Timing:
#>   Data preparation: 4.94 secs
#>   Model and feature tuning: 10.13 secs (3 models trained)
#>   Feature evolution: 5.54 secs (1 of 3 model trained)
#>   Final pipeline training: 7.85 secs (1 model trained)
#>   Python / MOJO scorer building: 42.05 secs / 0.00 secs
#> Validation score: AUC = 0.77802 +/- 0.0077539 (baseline)
#> Validation score: AUC = 0.77802 +/- 0.0077539 (final pipeline)
#> Test score:      AUC = 0.7861 +/- 0.0064711 (final pipeline)

summary(model)$score
#> [1] 0.7780229
```

50.2.4 Predicting on New Data

New data can be scored in two different ways:

- Call `predict()` directly on the model in R session.
- Download a scoring pipeline and embed that into your Python or Java workflow.

Predicting in R

Generic `predict()` either directly returns an R data.frame with the results (by default) or it returns a URL pointing to a CSV file with the results (`return_df=FALSE`). The latter option may be useful when you predict on a large dataset.

```
predictions <- predict(model, newdata = creditcard.splits$test)
#>
#> |          0%
#> =====| 100%
#> Loading required package: bitops

head(predictions)
#>   DEFAULT_PAYMENT_NEXT_MONTH.0 DEFAULT_PAYMENT_NEXT_MONTH.1
#> 1       0.8879988           0.11200116
#> 2       0.9289870           0.07101299
#> 3       0.9550328           0.04496716
#> 4       0.3513577           0.64864230
#> 5       0.9183724           0.08162758
#> 6       0.9154425           0.08455751

predict(model, newdata = creditcard.splits$test, return_df = FALSE)
#>
#> |          0%
#> =====| 100%
#> [1] "h2oai_experiment_7e2b70ae-5baa-11e9-a50b-b938de969cddb/7e2b70ae-5baa-11e9-a50b-b938de969cddb_preds_f854b49f.csv"
```

Downloading Python or MOJO Scoring Pipelines

For productizing your model in a Python or Java, you can download full Python or MOJO pipelines, respectively. For more information about how to use the pipelines, please see the documentation.

```
dai.download_mojo(model, path = tempdir(), force = TRUE)
#> 
#> | 
#> |           0%
#> ======| 100%
#> Downloading the pipeline:
#> [1] "/tmp/RtmpplsLT29/mojo-7e2b70ae-5baa-11e9-a50b-b938de969cdb.zip"
dai.download_python_pipeline(model, path = tempdir(), force = TRUE)
#> 
#> | 
#> |           0%
#> ======| 100%
#> Downloading the pipeline:
#> [1] "/tmp/RtmpplsLT29/python-pipeline-7e2b70ae-5baa-11e9-a50b-b938de969cdb.zip"
```

50.2.5 Managing the Datasets and Models

After some time, you may have multiple datasets and models on your DAI server. The dai package offers a few utility functions to find, reuse, and remove the existing datasets and models.

If you already have the dataset loaded into DAI, you can get the DAIFrame object by either `dai.get_frame` (if you know the frame's key) or `dai.find_dataset` (if you know the original path or at least a part of it):

```
dai.get_frame(creditcard$key)
#> DAI frame '7abe28b2-5baa-11e9-a50b-b938de969cdb': 23999 obs. of 25 variables
#> File path: tests/smldata/kaggle/CreditCard/creditcard_train_cat.csv

dai.find_dataset('creditcard')
#> DAI frame '7abe28b2-5baa-11e9-a50b-b938de969cdb': 23999 obs. of 25 variables
#> File path: tests/smldata/kaggle/CreditCard/creditcard_train_cat.csv
```

The latter directly returns you the frame if there's only one match. Otherwise it let you select which frame to return from all the matching candidates.

Furthermore, you can get a list of datasets or models:

```
datasets <- dai.list_datasets()
head(datasets)
#>          key          name
#> 1 7cf613a6-5baa-11e9-a50b-b938de969cdb      test
#> 2 7cf3024c-5baa-11e9-a50b-b938de969cdb      train
#> 3 7c38cb84-5baa-11e9-a50b-b938de969cdb  iris9elf15d2df00.csv
#> 4 7abe28b2-5baa-11e9-a50b-b938de969cdb creditcard_train_cat.csv
#> 
#>   file_path
#> 1 ./tmp/7cf613a6-5baa-11e9-a50b-b938de969cdb/test.1554912341.0966916.bin
#> 2 ./tmp/7cf3024c-5baa-11e9-a50b-b938de969cdb/train.1554912341.0864356.bin
#> 3 ./tmp/7c38cb84-5baa-11e9-a50b-b938de969cdb/iris9elf15d2df00.csv.1554912339.9424415.bin
#> 4 tests/smldata/kaggle/CreditCard/creditcard_train_cat.csv
#> 
#>   file_size data_source row_count column_count import_status import_error
#> 1     567584    upload       4800        25            0
#> 2     2265952    upload      19199        25            0
#> 3      7064    upload       150         5            0
#> 4    2832040      file      23999        25            0
#> 
#>   aggregation_status aggregation_error aggregated_frame mapping_frame
#> 1          -1
#> 2          -1
#> 3          -1
#> 4          -1
#> 
#>   uploaded
#> 1      TRUE
#> 2      TRUE
#> 3      TRUE
#> 4     FALSE

models <- dai.list_models()
head(models)
#>          key description
#> 1 7e2b70ae-5baa-11e9-a50b-b938de969cdb  mupulori
#> 
#>   dataset_name parameters.dataset_key
#> 1 train.1554912341.0864356.bin 7cf3024c-5baa-11e9-a50b-b938de969cdb
#>   parameters.resumed_model_key parameters.target_col
#> 1                               DEFAULT_PAYMENT_NEXT_MONTH
#>   parameters.weight_col parameters.fold_col parameters.orig_time_col
#> 1
#>   parameters.time_col parameters.is_classification parameters.cols_to_drop
#> 1           [OF]          TRUE             NULL
#>   parameters.validset_key parameters.testset_key
#> 1                         7cf613a6-5baa-11e9-a50b-b938de969cdb
#>   parameters.enable_gpus parameters.seed parameters.accuracy
```

(continues on next page)

(continued from previous page)

```
#> 1      TRUE      25      1
#> parameters.time parameters.interpretability parameters.scorer
#> 1      1          10      AUC
#> parameters.time_groups_columns parameters.time_period_in_seconds
#> 1      NULL      NA
#> parameters.num_prediction_periods parameters.num_gap_periods
#> 1      NA        NA
#> parameters.is_timeseries parameters.config_overrides
#> 1      FALSE     NA
#> 1                                     log_file_path
#> 1 h2oai_experiment_7e2b70ae-5baa-11e9-a50b-b938de969cdb/h2oai_experiment_logs_7e2b70ae-5baa-11e9-a50b-b938de969cdb.zip
#> 1                                         pickle_path
#> 1 h2oai_experiment_7e2b70ae-5baa-11e9-a50b-b938de969cdb/best_individual.pickle
#> 1                                     summary_path
#> 1 h2oai_experiment_7e2b70ae-5baa-11e9-a50b-b938de969cdb/h2oai_experiment_summary_7e2b70ae-5baa-11e9-a50b-b938de969cdb.zip
#> train_predictions_path valid_predictions_path
#> 1
#> 1                                     test_predictions_path
#> 1 h2oai_experiment_7e2b70ae-5baa-11e9-a50b-b938de969cdb/test_preds.csv
#> progress status training_duration scorer      score test_score deprecated
#> 1      1      0    71.43582    AUC 0.7780229    0.7861      FALSE
#> model_file_size diagnostic_keys
#> 1      695996094      NULL
```

If you know the key of the dataset or model, you can obtain the instance of DAIFrame or DAIModel by `dai.get_model` and `dai.get_frame`:

```
dai.get_model(modelss$key[1])
#> Status: Complete
#> Experiment: 7e2b70ae-5baa-11e9-a50b-b938de969cdb, 2019-04-10 18:06, 1.7.0+local_0c7d019-dirty
#> Settings: 1/1/10, seed=25, GPUs enabled
#> Train data: train (19199, 25)
#> Validation data: N/A
#> Test data: test (4800, 24)
#> Target column: DEFAULT_PAYMENT_NEXT_MONTH (binary, 22.366% target class)
#> System specs: Linux, 126 GB, 40 CPU cores, 2/2 GPUs
#> Max memory usage: 0.406 GB, 0.167 GB GPU
#> Recipe: AutoDL (2 iterations, 2 individuals)
#> Validation scheme: stratified, 1 internal holdout
#> Feature engineering: 33 features scored (18 selected)
#> Timing:
#> Data preparation: 4.94 secs
#> Model and feature tuning: 10.13 secs (3 models trained)
#> Feature evolution: 5.54 secs (1 of 3 model trained)
#> Final pipeline training: 7.85 secs (1 model trained)
#> Python / MOJO scorer Building: 42.05 secs / 0.00 secs
#> Validation score: AUC = 0.77802 +/- 0.0077539 (baseline)
#> Validation score: AUC = 0.77802 +/- 0.0077539 (final pipeline)
#> Test score:      AUC = 0.7861 +/- 0.0064711 (final pipeline)
dai.get_frame(datasets$key[1])
#> DAI frame '7cf613a6-5baa-11e9-a50b-b938de969cdb': 4800 obs. of 25 variables
#> File path: ./tmp/7cf613a6-5baa-11e9-a50b-b938de969cdb/test.1554912341.0966916.bin
```

Finally, the datasets and models can be removed by `dai.rm`:

```
dai.rm(model, creditcard, creditcard.splits$train, creditcard.splits$test)
#> Model 7e2b70ae-5baa-11e9-a50b-b938de969cdb removed
#> Dataset 7abe28b2-5baa-11e9-a50b-b938de969cdb removed
#> Dataset 7cf3024c-5baa-11e9-a50b-b938de969cdb removed
#> Dataset 7cf613a6-5baa-11e9-a50b-b938de969cdb removed
```

The function `dai.rm` deletes the objects by default both from the server and the R session. If you wish to remove it only from the server, you can set `from_session=FALSE`. Please note that only objects can be removed from the session, i.e. in the example above the `creditcard.splits$train` and `creditcard.splits$test` objects will not be removed from R session because they are actually function calls (recall that \$ is a function).

MONITORING PENDING JOBS

Driverless AI features a **Pending Jobs** panel that allows you to monitor the progress of various long-running jobs that can be started from the *Completed Experiment* page. To view this panel, click the group of square icons located in the upper-right corner.

The following jobs are monitored in this panel:

- Create Autoreport
- Create MOJO Scoring Pipeline
- Create Python Scoring Pipeline
- Create Test Set Predictions
- Create Training Predictions
- Score Model
- Transform Data

The circular icon next to the description of a pending job indicates its status:

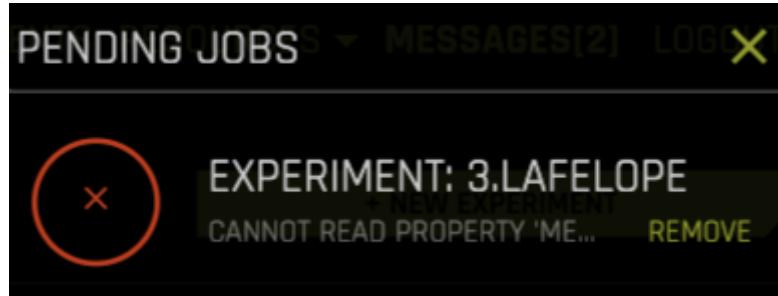
Icon	Status
	Complete
	Failed
	Running

Navigate to a completed job by clicking the **Open** icon. You can also clear a completed job from the panel by clicking **Remove** or cancel an ongoing job by clicking **Abort**.

Note: Certain jobs cannot be cancelled.

51.1 Failed Jobs

If a job fails, you can review the logs associated with that job type to determine what caused the failure. Refer to the [Driverless AI Logs](#) section for more information.



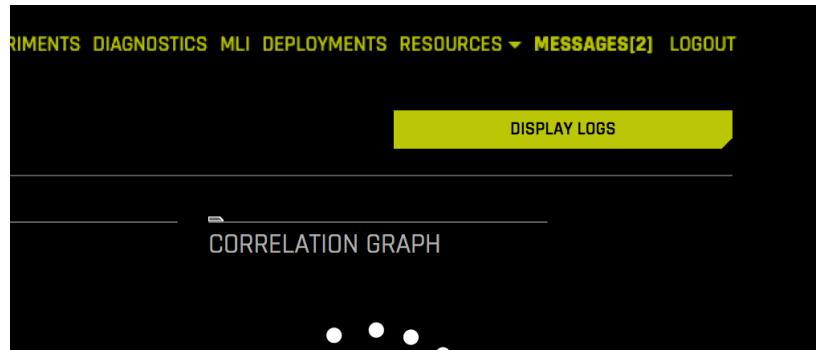
CHAPTER
FIFTYTWO

DRIVERLESS AI LOGS

Driverless AI provides a number of logs that can be viewed and/or retrieved when performing different tasks. This section describes how to access those Driverless AI logs.

52.1 While Visualizing Datasets

When running Autovizualization, you can access the Autoviz logs by clicking the **Display Logs** button on the Visualize Datasets page.

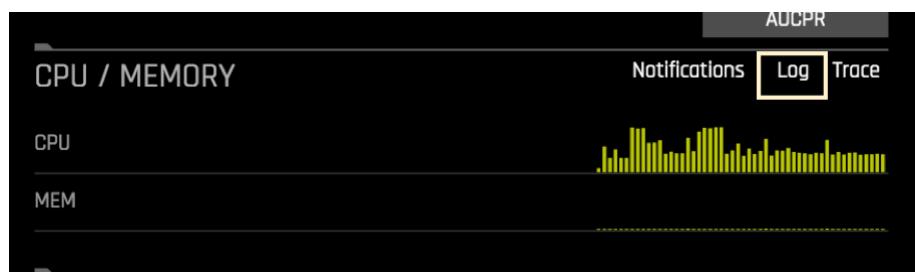


Using Driverless AI, Release 1.9.0.1

This page presents logs created while the dataset visualization was being performed. You can download the **vis-data-server.log** file by clicking the **Download Logs** button on this page. This file can be used to troubleshoot any issues encountered during dataset visualization.

52.2 While an Experiment is Running

While the experiment is running, you can access the logs by clicking on the **Log** button on the experiment screen. The **Log** button can be found in the CPU/Memory section.



Clicking on the **Log** button will present the experiment logs in real time. You can download these logs by clicking on the **Download Logs** button in the upper right corner.

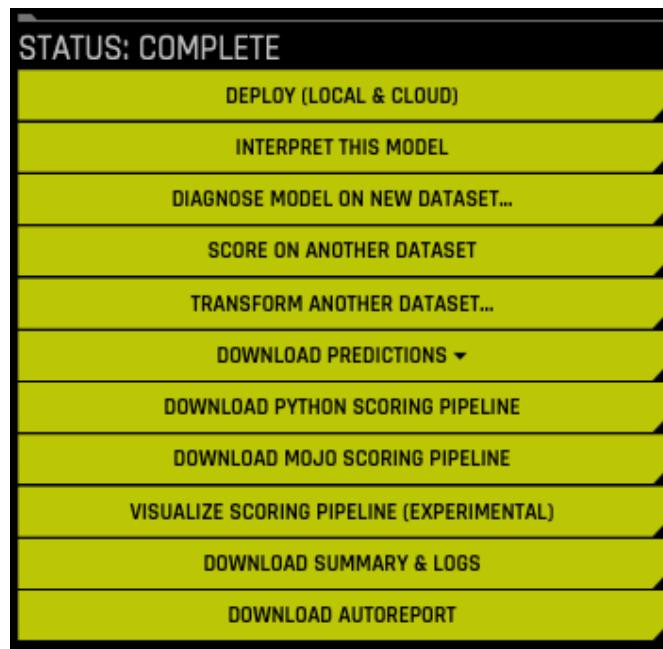


Only the **h2oai_experiment.log** can be downloaded while the experiment is running (for example: **h2oai_experiment_tobosoru.log**). It will have the same information as the logs being presented in real time on the screen.

For troubleshooting purposes, it is best to view the complete **h2oai_experiment.log** (or **h2oai_experiment_anonymized.log**). These will be available after the experiment finishes, as described in the next section.

52.3 After an Experiment has Finished

If the experiment has finished, you can download the logs by clicking on the **Download Experiment & Logs** button on the completed experiment screen.

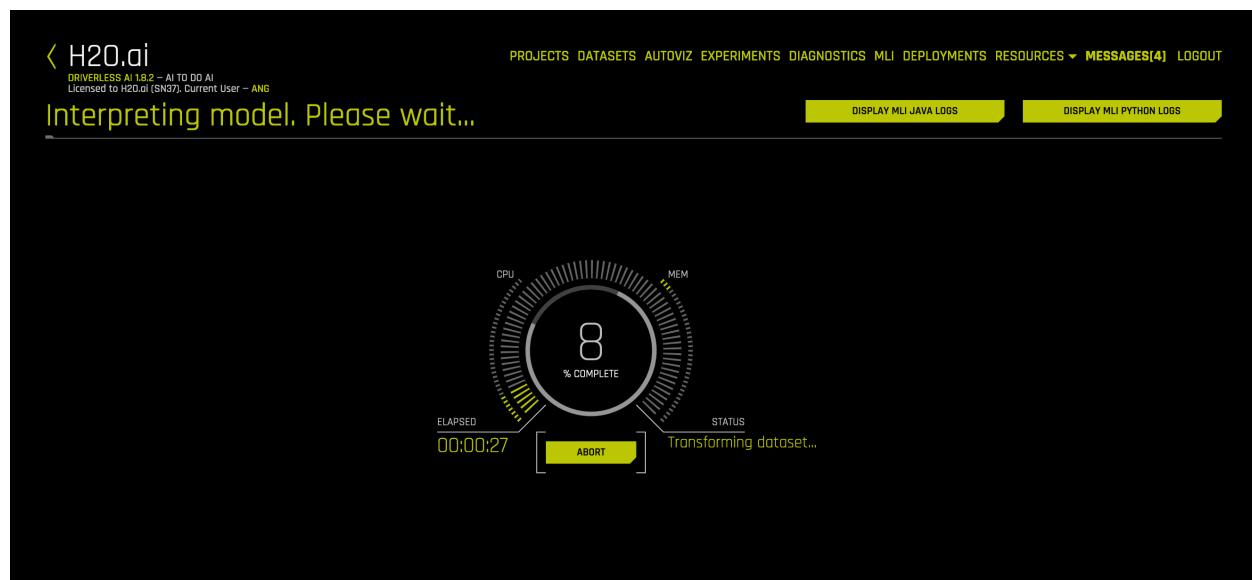


This will download a zip file that includes the following logs along with a summary of the experiment:

- **h2oai_experiment.log**: This is the log corresponding to the experiment.
- **h2oai_experiment_anonymized.log**: This is the log corresponding to the experiment where all data in the log is anonymized.

52.4 During Model Interpretation

Driverless AI allows you to view and download Python and/or Java logs while MLi is running. Note that these logs are not available for time-series experiments.



- The **Display MLI Python Logs** button allows you to view or download the Python log for the model interpretation. The downloaded file is named **h2oai_experiment_{mli_key}.log**.
- The **Display MLI Java Logs** button allows you to view or download the Java log for the model interpretation. The downloaded file is named **mli_experiment_{mli_key}.log**.

52.5 After Model Interpretation

You can view an MLI log for completed model interpretations by selecting the **Download MLI Logs** link on the MLI page.

MLI Parameters Summary:

- MLI Experiment:** vohokigu (5b3a9c88-8629-11e9-ac04-0242ac110002)
- DAI Experiment:** tocepobi (15e04816-8617-11e9-ac04-0242ac110002)
- Dataset:** CreditCard_train.csv
- Feature Space for Surrogate Models:** Original Features
- Target Column:** default payment next month
- Problem Type:** Classification
- Surrogate CV Folds:** 3
- LIME Method:** k-LIME
- LIME used k-means to cluster data based on these variables (top 6 from original features):** (list of variables)
- LIME Number of Clusters:** 4
- Tree Depth for Decision Tree Surrogate Model:** 3
- MLI Duration:** 00:01:42

This will download a zip file which includes the following logs:

- h2oai_experiment_{mli_key}.log:** This is the log corresponding to the model interpretation.
- h2oai_experiment_{mli_key}_anonymized.log:** This is the log corresponding to the model interpretation where all data in the log is anonymized.
- mli_experiment_{mli_key}.log:** This is the Java log corresponding to the model interpretation.

This file can be used to view logging information for successful interpretations. If MLI fails, then those logs are in `./tmp/h2oai_experiment_{mli_key}.log`, `./tmp/h2oai_experiment_{mli_key}_anonymized.log`, and `./tmp/mli_experiment_{mli_key}.log`.

SENDING LOGS TO H2O.AI

This section describes the logs to send in the event of failures when running Driverless AI.

53.1 Dataset Failures

- **Adding Datasets:** If a dataset fails to import, a message on the screen should provide the reason for the failure. The logs to send are available in the Driverless AI `./tmp` folder.
- **Dataset Details:** If a failure occurs when attempting to view Dataset Details, the logs to send are available in the Driverless AI `./tmp` folder.
- **Autoviz:** If a failure occurs when attempting to Visualize Datasets, a message on the screen should provide a reason for the failure. The logs to send are available in the Driverless AI `./tmp` folder.

53.2 Experiments

- **While Running an Experiment:** As indicated previously, a **Log** button is available on the Experiment page. Clicking on the **Log** button will present the experiment logs in real time. You can download these logs by clicking on the **Download Logs** button in the upper right corner. You can also retrieve the `h2oai_experiment.log` for the corresponding experiment in the Driverless AI `./tmp` folder.

53.3 MLI

- **During Model Interpretation:** If a failure occurs during model interpretation, then the logs to send are `./tmp/h2oai_experiment_{mli_key}.log` and `./tmp/h2oai_experiment_{mli_key}_anonymized.log`.

53.4 Custom Recipes

- **After Running an Experiment:** If a Custom Recipe is producing errors, the entire zip file obtained by clicking on the **Download Summary & Logs** button can be sent for troubleshooting. Note that these files may contain information that is not anonymized.

CHAPTER
FIFTYFOUR

SYSTEM LOGS

System logs can also include useful information about Driverless AI. Collecting system logs varies depending on the way Driverless AI was installed.

54.1 Docker Installs

If you installed the Driverless AI Docker image, you can view the `./log/{date_time}/dai.log` file from the Driverless AI `./log` folder.

54.2 Native Installs

- For RPM and Deb installs that do NOT use systemd, the log file will be in `opt/h2oai/dai/log`. For example:

```
sudo less /opt/h2oai/dai/log/dai.log
sudo less /opt/h2oai/dai/log/h2o.log
sudo less /opt/h2oai/dai/log/procsy.log
sudo less /opt/h2oai/dai/log/vis-server.log
```

- For RPM and Deb installs that use systemd, you can use journalctl to view logs collected by systemd. For example:

```
sudo systemctl status dai-dai
sudo systemctl status dai-h2o
sudo systemctl status dai-procsy
sudo systemctl status dai-vis-server
sudo journalctl -u dai-dai
sudo journalctl -u dai-h2o
sudo journalctl -u dai-procsy
sudo journalctl -u dai-vis-server
```

- For Tar installs, use the following commands to view system logs:

```
less log/dai.log
less log/h2o.log
less log/procsy.log
less log/vis-server.log
```


DRIVERLESS AI SECURITY

55.1 Objective

The goal of this document is to describe different aspects of Driverless AI security and to provide guidelines to secure the system by reducing its surface of vulnerability.

This section covers the following areas of the product:

- User access
- Authentication
- Authorization
- Data security
- Data import
- Data export
- Logs
- User data isolation
- Transfer security
- Custom recipes security
- Web UI security

55.2 Important things to know

Warning: **WARNING** Security in a default installation of Driverless AI is DISABLED! By default, a Driverless AI installation targets ease-of-use and does not enable all security features listed in this document. For production environments, we recommend following this document and performing a secure Driverless AI installation.

55.3 User Access

55.3.1 Authentication Methods

Option	Default Value	Recommended Value	Description
authentication_method	"unvalidated" "PAM"	Any supported authentication (e.g., LDAP, PAM) method except "unvalidated" and "none".	Define user authentication method.
authentication_default_time	Consult your security requirements.	Number of hours after which a user has to re-login.	

55.3.2 Authorization Methods

At this point, Driverless AI does not perform any authorization.

55.4 Data Security

55.4.1 Data Import

Option	Default Value	Recommended Value	Description
enabled_filesystems	file, hdfs, s3	Configure only needed data sources.	Control list of available/configured data sources.
max_file_size	104857600	Configure based on expected file size and size of Driverless AI deployment.	Limit maximum size of uploaded file.
supported_file_extensions	fig.toml	It is recommended to limit file types to extension used in the target environment (e.g., parquet).	Supported file formats listed in filesystem browsers.
show_all_filesystems	false		Show all available data sources in WebUI (even though there are not configured). It is recommended to show only configured data sources.

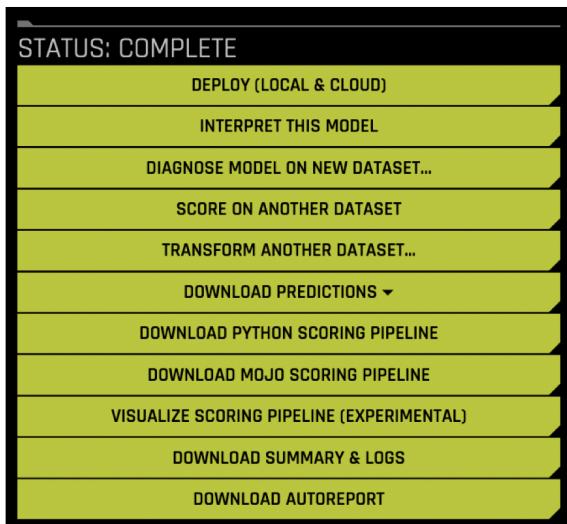
55.4.2 Data Export

Option	De-fault Value	Recom-mended Value	Description
enable_dataset_download	true	false	Control ability to download any datasets (uploaded, predictions, ML). Note: if dataset download is disabled, we strongly suggest to disable custom recipes as well to remove another way how data could be exported from the application.
enable_artifact_exports	false	false	Replace all downloads on the experiment page to “exports”, and allow users to push to the artifact store configured with artifacts_store. (See notes below.)
artifacts_store	file_system	file_system	Stores a MOJO on a file system directory denoted by artifacts_file_system_directory. (See notes below.)
artifacts_file_system_directory	tmp	tmp	File system location where artifacts will be copied in case artifacts_store is set to file_system. (See notes below.)

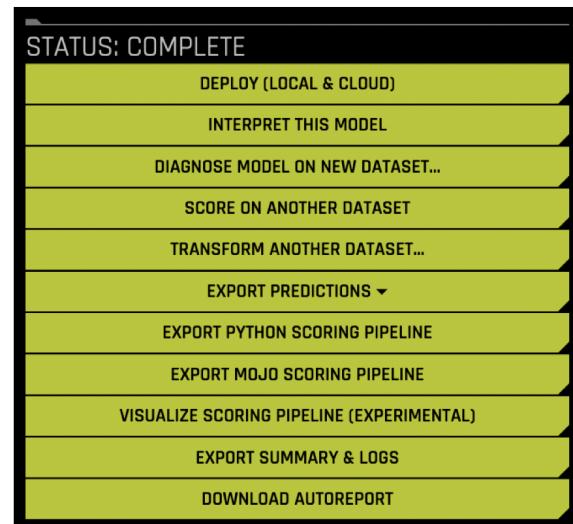
Notes about Artifacts:

- Currently, file_system is the only option that can be specified for artifacts_store. Additional options will be available in future releases.
- The location for artifacts_file_system_directory is expected to be a directory on your server.
- When these artifacts are enabled/configured, the menu options on the *Completed Experiment* page will change. Specifically, all “Download” options (with the exception of Autoreport) will change to “Export.” Refer to *Export Artifacts* for more information.

Driverless AI completed experiment menu with default configuration



Driverless AI completed experiment menu with artifact exporting configured



55.4.3 Logs

The Driverless AI produces several logs:

- audit logs
- server logs
- experiment logs

The administrator of Driverless AI application (i.e., person who is responsible for configuration and setup of the application) has control over content which is written to the logs.

Option	De-fault Value	Recom-mended Value	Description
audit_log_retention	0 period (days)	5 period (disable audit log rotation)	Number of days to keep audit logs. The value 0 disable rotation.
do_not_log_list	see config.toml	—	Contain list of configuration options which are not recorded in logs.
log_level	1	see config.toml	Define verbosity of logging
collect_server_logs	false	see experiment_level	Dump server logs with experiment. Dangerous since server logs can contain information about experiments of other users using Driverless AI.
h2o_recipes_log_level	None	—	Log level for OSS H2O instances used by custom recipes.
debug_log	false	false	Enable debug logs.
write_recipes_to_expansion_file_logger	expansion_file_logger	—	Dump a custom recipe source code into logs.

55.4.4 User Data Isolation

Option	De-fault Value	Recommended Value	Description
data_directory	"tmp"	Specify proper name and location of directory.	Directory where Driverless AI stores all computed experiments and datasets
file_hide_data_directory	data_directory	—	Hide data_directory in file-system browser. It is recommended to hide it to protect data_directory from browsing and corruption.
file_path_filtering_enabled	false	—	Enable path filter for file-system browser (file data source). By default the filter is disabled which means users can browse the entire application-local filesystem.
file_path_filters	include a list of folder paths or {{DAI_USERNAME}} for the logged user's directory. For example, "['/home/{{DAI_USERNAME}}/', '/data/prod']".	—	List of absolute path prefixes to restrict access to in file-browser.

55.5 Client-Server Communication Security

Option	Default Value	Recommended Value	Description
enable_https	false	true	Enable HTTPS
ssl_key_file	"/etc/dai/private_key.pem"	Correct private key.	Private key to setup HTTPS/SSL communication.
ssl_crt_file	"/etc/dai/cert.pem"	Correct public certificate.	Public certificate to setup HTTPS/SSL.
ssl_no_sslv2	true	true	Prevents an SSLv2 connection.
ssl_no_sslv3	true	true	Prevents an SSLv3 connection.
ssl_no_tlsv1	true	true	Prevents an TLSv1 connection.
ssl_no_tlsv1.1	true	true	Prevents an TLSv1.1 connection.
ssl_no_tlsv1.2	false (disable TLSv1.2 only if TLSv1.3 is available)	false	Prevents a TLSv1.2 connection.
ssl_no_tlsv1.3	false	false	Prevents a TLSv1.3 connection.

55.5.1 Response Headers

The response headers which are passed between Driverless AI server and client (browser, Python/R clients) are controlled via the following option:

Option	Default Value	Recommended Value	Description
extra_http_headers	"{}"	See below	Configure HTTP header returned in server response.

Recommended Response Headers

Header	Description	Example value	Link
Strict-Transport-Security	The header lets a web site tell browsers that it should only be accessed using HTTPS, instead of using HTTP. The max-age specifies time, in seconds, that the browser should remember that a site is only to be accessed using HTTPS.	max-age=63072000	https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/Strict-Transport-Security
Content-Security-Policy	Content Security Policy (CSP) is an added layer of security that helps to detect and mitigate certain types of attacks, including Cross Site Scripting and data injection attacks. Controls from where the page can download source.	default-src https: ; font-src 'self'; script-src 'self' 'unsafe-eval' 'unsafe-inline'; style-src 'self' 'unsafe-inline'; object-src 'none' Note: The Driverless AI is still requires to have unsafe-eval and unsafe-inline configured, which potentially makes the server vulnerable to XSS attacks.	https://developer.mozilla.org/en-US/docs/Web/HTTP/CSP https://infosec.mozilla.org/guidelines/web_security#Examples_5
X-Frame-Options	Controls where a page can get source to render in a frame. The value here overrides the default, which is SAMEORIGIN.	deny	https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/X-Frame-Options
X-Content-Type-Options	Prevents the browser from trying to determine the content-type of a resource that is different than the declared content-type.	nosniff	https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/X-Content-Type-Options
X-XSS-Protection	The HTTP X-XSS-Protection response header is a feature of Internet Explorer, Chrome and Safari that stops pages from loading when they detect reflected cross-site scripting (XSS) attacks. When value is set to 1 and a cross-site scripting attack is detected, the browser will sanitize the page (remove the unsafe parts).	1; mode=block	https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/X-XSS-Protection

Other Headers to Consider

Header	Documentation
Public-Key-Pins	https://developer.mozilla.org/en-US/docs/Web/HTTP/Public_Key_Pinning
CORS-related headers	https://developer.mozilla.org/en-US/docs/Web/HTTP/CORS

55.6 Web UI Security

Note: The Driverless AI UI is design to be user-friendly, and by default all features like auto-complete are enabled. Disabling the user-friendly features increases security of the application, but impacts user-friendliness and usability of the application.

Option	Default Value	Recommended Value	Description
allow_form_auto_complete	left	false	Control auto-completion in Web UI elements (e.g., login inputs).
allow_localstorage	danger	false	Disable use of web browser local storage.
show_all_filesystems	systems	false	Show all available data sources in WebUI (even though there are not configured). It is recommended to show only configured data sources.

55.7 Custom Recipe Security

Note: By default Driverless AI enables custom recipes as a main route for the way data-science teams can extend the application capabilities. In enterprise environments, it is recommended to follow best software engineering practices for development of custom recipes (i.e., code reviews, testing, stage releasing, etc.) and bundle only a pre-defined and approved set of custom Driverless AI extensions.

Option	Default Value	Recommended Value	Description
enable_custom_recipes	true	false	Enable custom Python recipes.
enable_custom_recipes_upload	upload	false	Enable uploading of custom recipes.
enable_custom_recipes_from_url	from_url	false	Enable downloading of custom recipes from external URL.
include_custom_recipes	sf_bysdefaf	false	Include custom recipes in default inclusion lists. (warning: enables all custom recipes)

55.8 Baseline Secure Configuration

The following Driverless AI configuration is an example of secure configuration. Please, make sure that you fill all necessary config options.

```
# 
# Auth
#
# Configure auth method
#authentication_method="PAM"
# Force user re-login after 24hours
authentication_default_timeout_hours=24
```

(continues on next page)

(continued from previous page)

```
# Data
#
# Configure available connectors
enabled_file_systems="hdfs"
show_all_filesystems=false

# Restrict Downloads
enable_dataset_downloading=false

#
# Logs
#
audit_log_retention_period=0
collect_server_logs_in_experiment_logs=false

#
# User data isolation
#
file_hide_data_directory=true
#file_path_filter_include=true
#file_path_filter_include=[]

#
# Client-Server Communication
enable_https=true
ssl_key_file="<>ILL ME>>"
ssl_crt_file="<>ILL ME>>"
# Disable support of TLSv1.2 on server side only if your environment supports TLSv1.3
#ssl_no_tisv1_2=true

#
# Web UI security
#
allow_form_autocomplete=false
allow_localstorage=false

extra_http_headers={ "Strict-Transport-Security" = "max-age=63072000", "Content-Security-Policy" = "default-src https: ; font-src 'self'; script-src 'self' ↪ 'unsafe-eval' 'unsafe-inline'; style-src 'self' 'unsafe-inline'; object-src 'none'", "X-Frame-Options" = "deny", "X-Content-Type-Options" = "nosniff", ↪ "X-XSS-Protection" = "1; mode=block" }

#
# Custom Recipes
#
enable_custom_recipes=false
enable_custom_recipes_upload=false
enable_custom_recipes_from_url=false
include_custom_recipes_by_default=false
```

CHAPTER
FIFTYSIX

FAQ

H2O Driverless AI is an artificial intelligence (AI) platform for automatic machine learning. Driverless AI automates some of the most difficult data science and machine learning workflows such as feature engineering, model validation, model tuning, model selection and model deployment. It aims to achieve highest predictive accuracy, comparable to expert data scientists, but in much shorter time thanks to end-to-end automation. Driverless AI also offers automatic visualizations and machine learning interpretability (MLI). Especially in regulated industries, model transparency and explanation are just as important as predictive performance. Modeling pipelines (feature engineering and models) are exported (in full fidelity, without approximations) both as Python modules and as Java standalone scoring artifacts.

This section provides answers to frequently asked questions. If you have additional questions about using Driverless AI, post them on Stack Overflow using the driverless-ai tag at <http://stackoverflow.com/questions/tagged/driverless-ai>. You can also post questions on the H2O.ai Community Slack workspace in the **#driverlessai** channel. If you have not signed up for the H2O.ai Community Slack workspace, you can do so here: <https://www.h2o.ai/community/>.

General

- *How is Driverless AI different than any other black box ML algorithm?*
- *How often do new versions come out?*

Installation/Upgrade/Authentication

- *How can I change my username and password?*
- *Can Driverless AI run on CPU-only machines?*
- *How can I upgrade to a newer version of Driverless AI?*
- *What kind of authentication is supported in Driverless AI?*
- *How can I automatically turn on persistence each time the GPU system reboots?*
- *How can I start Driverless AI on a different port than 12345?*
- *Can I set up TLS/SSL on Driverless AI?*
- *Can I set up TLS/SSL on Driverless AI in AWS?*
- *Why do I receive a “package dai-<version>.x86_64 does not verify: no digest” error during the installation?*
- *I received a “Must have exactly one OpenCL platform ‘NVIDIA CUDA’” error. How can I fix that?*
- *Is it possible for multiple users to share a single Driverless AI instance?*
- *Can multiple Driverless AI users share a GPU server?*
- *How can I retrieve a list of Driverless AI users?*
- *Start of Driverless AI fails on the message “Segmentation fault (core dumped)” on Ubuntu 18/RHEL 7.6. How can I fix this?*
- *Why do I receive a “shared object file” error when running on RHEL 8?*

Data

- *Is there a file size limit for datasets?*
- *How can I import CSV files that use UTF-8 encoding into Excel?*
- *Can a byte order mark be used when writing CSV files with datatable?*

Connectors

- *Why can't I import a folder as a file when using a data connector on Windows?*
- *I get a ClassNotFoundException error when I try to select a JDBC connection. How can I fix that?*
- *I get a org.datanucleus.exceptions.NucleusUserException: Please check your CLASSPATH and plugin specification error when attempting to connect to hive. How can I fix that?*

Recipes

- *Where can I retrieve H2O's custom recipes?*
- *How can I create my own custom recipe?*
- *Are MOJOs supported for experiments that use custom recipes?*
- *How can I use BYOR in my airgapped installation?*

Experiments

- *How much memory does Driverless AI require in order to run experiments?*
- *How many columns can Driverless AI handle?*
- *How should I use Driverless AI if I have large data?*
- *How does Driverless AI detect the ID column?*
- *Can Driverless AI handle data with missing values/nulls?*
- *How does Driverless AI deal with categorical variables? What if an integer column should really be treated as categorical?*
- *How are outliers handled?*
- *If I drop several columns from the Train dataset, will Driverless AI understand that it needs to drop the same columns from the Test dataset?*
- *Does Driverless AI treat numeric variables as categorical variables?*
- *Which algorithms are used in Driverless AI?*
- *Why do my selected algorithms not show up in the Experiment Preview?*
- *How can we turn on TensorFlow Neural Networks so they are evaluated?*
- *Does Driverless AI standardize the data?*
- *What objective function is used in XGBoost?*
- *Does Driverless AI perform internal or external validation?*
- *How does Driverless AI prevent overfitting?*
- *How does Driverless AI avoid the multiple hypothesis (MH) problem?*
- *How does Driverless AI suggest the experiment settings?*
- *What happens when I set Interpretability and Accuracy to the same number?*
- *Can I specify the number of GPUs to use when running Driverless AI?*

- *How can I create the simplest model in Driverless AI?*
- *Why is my experiment suddenly slow?*
- *When I run multiple experiments with different seeds, why do I see different scores, runtimes, and sizes on disk in the Experiments listing page?*
- *Why does the final model performance appear to be worse than previous iterations?*
- *How can I find features that may be causing data leakages in my Driverless AI model?*
- *How can I see the performance metrics on the test data?*
- *How can I see all the performance metrics possible for my experiment?*
- *What if my training/validation and testing data sets come from different distributions?*
- *Does Driverless AI handle weighted data?*
- *How does Driverless AI handle fold assignments for weighted data?*
- *Why do I see that adding new features to a dataset deteriorates the performance of the model?*
- *How does Driverless AI handle imbalanced data for binary classification experiments?*

Feature Transformations

- *Where can I get details of the various transformations performed in an experiment?*

Predictions

- *How can I download the predictions onto the machine where Driverless AI is running?*
- *Why are predicted probabilities not available when I run an experiment without ensembling?*

Deployment

- *What drives the size of a MOJO?*
- *Are MOJOS thread safe?*
- *Running the scoring pipeline for my MOJO is taking several hours. How can I get this to run faster?*
- *Why have I encountered a “Best Score is not finite” error?*

Time Series

- *What if my data has a time dependency?*
- *What is a lag, and why does it help?*
- *Why can't I specify a validation data set for time-series problems? Why do you look at the test set for time-series problems*
- *Why does the gap between train and test matter? Is it because of creating the lag features on the test set?*
- *In regards to applying the target lags to different subsets of the time group columns, are you saying Driverless AI perform auto-correlation at “levels” of the time series? For example, consider the Walmart dataset where I have Store and Dept (and my target is Weekly Sales). Are you saying that Driverless AI checks for auto-correlation in Weekly Sales based on just Store, just Dept, and both Store and Dept?*
- *How does Driverless AI detect the time period?*
- *What is the logic behind the selectable numbers for forecast horizon length?*
- *Assume that in my Walmart dataset, all stores provided data at the week level, but one store provided data at the day level. What would Driverless AI do?*

- Assume that in my Walmart dataset, all stores and departments provided data at the weekly level, but one department in a specific store provided weekly sales on a bi-weekly basis (every two weeks). What would Driverless AI do?
- Why does the number of weeks that you want to start predicting matter?
- Are the scoring components of time series sensitive to the order in which new pieces of data arrive? I.e., is each row independent at scoring time, or is there a real-time windowing effect in the scoring pieces?
- What happens if the user, at predict time, gives a row with a time value that is too small or too large?
- What's the minimum data size for a time series recipe?
- How long must the training data be compared to the test data?
- How does the time series recipe deal with missing values?
- Can the time information be distributed acrosss multiple columns in the input data (such as [year, day, month])?
- What type of modeling approach does Driverless AI use for time series?
- What's the idea behind exponential weighting of moving averages?

Logging

- How can I reduce the size of the Audit Logger?

56.1 General

How is Driverless AI different than any other black box ML algorithm?

Driverless AI uses many techniques (some older and some cutting-edge) for interpreting black box models including creating reason codes for every prediction the system makes. We have also created numerous open source code examples and free publications that explain these techniques. See the list below for links to these resources and for references for the interpretability techniques.

- Open source interpretability examples:
 - https://github.com/jphall663/interpretable_machine_learning_with_python
 - <https://content.oreilly.com/oriole/Interpretable-machine-learning-with-Python-XGBoost-and-H2O>
 - <https://github.com/h2oai/mli-resources>
- Free Machine Learning Interpretability publications:
 - <http://www.oreilly.com/data/free/an-introduction-to-machine-learning-interpretability.csp>
 - <http://docs.h2o.ai/driverless-ai/latest-stable/docs/booklets/MLIBooklet.pdf>
- Machine Learning Techniques *already* in Driverless AI:
 - Tree-based Variable Importance: https://web.stanford.edu/~hastie/ElemStatLearn/printings/ESLII_print12.pdf
 - Partial Dependence: https://web.stanford.edu/~hastie/ElemStatLearn/printings/ESLII_print12.pdf
 - LIME: <http://www.kdd.org/kdd2016/papers/files/rfp0573-ribeiroA.pdf>
 - LOCO: <http://www.stat.cmu.edu/~ryantibs/papers/conformal.pdf>
 - ICE: <https://arxiv.org/pdf/1309.6392.pdf>
- Surrogate Models:

- <https://papers.nips.cc/paper/1152-extracting-tree-structured-representations-of-trained-networks.pdf>
- <https://arxiv.org/pdf/1705.08504.pdf>
- Shapely Explanations: <http://papers.nips.cc/paper/7062-a-unified-approach-to-interpreting-model-predictions>

How often do new versions come out?

The frequency of major new Driverless AI releases has historically been about every two months.

56.2 Installation/Upgrade/Authentication

How can I change my username and password?

The username and password are tied to the experiments you have created. For example, if I log in with the username/password: megan/megan and start an experiment, then I would need to log back in with the same username and password to see those experiments. The username and password, however, does not limit your access to Driverless AI. If you want to use a new user name and password, you can log in again with a new username and password, but keep in mind that you won't see your old experiments.

Can Driverless AI run on CPU-only machines?

Yes, Driverless AI can run on machines with CPUs only, though GPUs are recommended. Installation instructions are available for GPU and CPU systems. Refer to *Installing and Upgrading Driverless AI* for more information.

How can I upgrade to a newer version of Driverless AI?

Upgrade instructions vary depending on your environment. Refer to the installation section for your environment. Upgrade instructions are included there.

What kind of authentication is supported in Driverless AI?

Driverless AI supports Client Certificate, LDAP, Local, mTLS, OpenID, none, and unvalidated (default) authentication. These can be configured by setting the appropriate environment variables in the config.toml file or by specifying the environment variables when starting Driverless AI. Refer to *Configuring Authentication* for more information.

How can I automatically turn on persistence each time the GPU system reboots?

For GPU machines, the sudo nvidia-persistenced --user dai command can be run after each reboot to enable persistence. For systems that have systemd, it is possible to automatically enable persistence after each reboot by removing the --no-persistence-mode flag from nvidia-persistenced.service. Before running the steps below, be sure to review the following for more information:

- <https://docs.nvidia.com/deploy/driver-persistence/index.html#persistence-daemon>
- <https://docs.nvidia.com/deploy/driver-persistence/index.html#installation>

1. Run the following to stop the nvidia-persistenced.service:

```
sudo systemctl stop nvidia-persistenced.service
```

2. Open the file /lib/systemd/system/nvidia-persistenced.service. This file includes a line "ExecStart=/usr/bin/nvidia-persistenced --user nvidia-persistenced --no-persistence-mode --verbose".
3. Remove the flag --no-persistence-mode from that line so that it reads:

```
ExecStart=/usr/bin/nvidia-persistenced --user nvidia-persistenced --verbose
```

- Run the following command to start the nvidia-persistenced.service:

```
sudo systemctl start nvidia-persistenced.service
```

How can I start Driverless AI on a different port than 12345?

Docker Image Installs

Native Installs

When starting Driverless AI in Docker, the `-p` option specifies the port on which Driverless AI will run. Change this option in the start script if you need to run on a port other than 12345. The following example shows how to run on port 22345. (Change `nvidia-docker run` to `docker-run` if needed.) Keep in mind that [privileged ports will require root access](#).

```
nvidia-docker run \
--pid=host \
--init \
--rm \
--shm-size=256m \
-u `id -u`:`id -g` \
-p 22345:12345 \
-v `pwd`/data:/data \
-v `pwd`/log:/log \
-v `pwd`/license:/license \
-v `pwd`/tmp:/tmp \
harbor.h2o.ai/h2oai/dai-centos7-x86_64:1.9.0-cudal0.0.23
```

To run on a port other than 12345, update the port value in the **config.toml** file. The following example shows how to run Driverless AI on port 22345. Keep in mind that [privileged ports will require root access](#).

```
# Export the Driverless AI config.toml file (or add it to ~/.bashrc)
export DRIVERLESS_AI_CONFIG_FILE="/config/config.toml"

# IP address and port for Driverless AI HTTP server.
ip = "127.0.0.1"
port = 22345
```

Point to this updated config file when restarting Driverless AI.

Can I set up TLS/SSL on Driverless AI?

Yes, Driverless AI provides configuration options that allow you to set up HTTPS/TLS/SSL. You will need to have your own SSL certificate, or you can create a self-signed certificate for yourself.

To enable HTTPS/TLS/SSL on the Driverless AI server, add the following to the config.toml file:

```
enable_https = true
ssl_key_file = "/etc/dai/private_key.pem"
ssl_crt_file = "/etc/dai/cert.pem"
```

You can make a self-signed certificate for testing with the following commands:

```
umask 077
openssl req -x509 -newkey rsa:4096 -keyout private_key.pem -out cert.pem -days 20 -nodes -subj '/O=Driverless AI'
sudo chown dai:dai cert.pem private_key.pem
sudo mv cert.pem private_key.pem /etc/dai
```

To configure specific versions of TLS/SSL, enable or disable the following settings in the config.toml file:

```
ssl_no_sslv2 = true
ssl_no_sslv3 = true
ssl_no_tslv1 = true
ssl_no_tslv1_1 = true
ssl_no_tslv1_2 = false
ssl_no_tslv1_3 = false
```

Can I set up TLS/SSL on Driverless AI in AWS?

Yes, you can set up HTTPS/TLS/SSL on Driverless AI running in an AWS environment. HTTPS/TLS/SSL needs to be configured on the host machine, and the necessary ports will need to be opened on the AWS side. You will need to have your own TLS/SSL cert or you can create a self signed cert for yourself.

The following is a very simple example showing how to configure HTTPS with a proxy pass to the port on the container 12345 with the keys placed in /etc/nginx/. Replace <server_name> with your server name.

```
server {
    listen 80;
    return 301 https://$host$request_uri;
}

server {
    listen 443;

    # Specify your server name here
    server_name <server_name>;

    ssl_certificate      /etc/nginx/cert.crt;
    ssl_certificate_key  /etc/nginx/cert.key;
    ssl on;
    ssl_session_cache builtin:1000 shared:SSL:10m;
    ssl_protocols TLSv1 TLSv1.1 TLSv1.2;
    ssl_ciphers HIGH:!aNULL:!eNULL:!EXPORT:!CAMELLIA:!DES:!MD5:!PSK:!RC4;
    ssl_prefer_server_ciphers on;

    access_log            /var/log/nginx/dai.access.log;

    location / {
        proxy_set_header Host $host;
        proxy_set_header X-Real-IP $remote_addr;
        proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
        proxy_set_header X-Forwarded-Proto $scheme;

        # Fix the "It appears that your reverse proxy set up is broken" error.
        proxy_pass          http://localhost:12345;
        proxy_read_timeout  90;

        # Specify your server name for the redirect
        proxy_redirect      http://localhost:12345 https://<server_name>;
    }
}
```

More information about SSL for Nginx in Ubuntu 16.04 can be found here: <https://www.digitalocean.com/community/tutorials/how-to-create-a-self-signed-ssl-certificate-for-nginx-in-ubuntu-16-04>.

I received a “package dai-<version>.x86_64 does not verify: no digest” error during the installation. How can I fix this?

You will receive a “package dai-<version>.x86_64 does not verify: no digest” error when installing the rpm using an RPM version newer than 4.11.3. You can run the following as a workaround, replacing <version> with your DAI version:

```
rpm --nodigest -i dai-<version>.x86_64.rpm
```

I received a “Must have exactly one OpenCL platform ‘NVIDIA CUDA’” error. How can I fix that?

If you encounter problems with opencl errors at server time, you may see the following message:

```
2018-11-08 14:26:15,341 C: D:452.2GB M:246.0GB 21603 ERROR : Must have exactly one OpenCL platform 'NVIDIA CUDA', but got:
Platform #0: Clover
Platform #1: NVIDIA CUDA
++- Device #0: GeForce GTX 1080 Ti
++- Device #1: GeForce GTX 1080 Ti
++- Device #2: GeForce GTX 1080 Ti

Uninstall all but 'NVIDIA CUDA' platform.
```

For Ubuntu, the solution is to run the following:

```
sudo apt-get remove mesa-opencl-icd
```

Is it possible for multiple users to share a single Driverless AI instance?

Driverless AI supports multiple users, and Driverless AI is licensed per a single named user. Therefore, in order, to have different users run experiments simultaneously, they would each need a license. Driverless AI manages the GPU(s) that it is given and ensures that different experiments from different users can run safely simultaneously and don’t interfere with each other. So when two licensed users log in with different credentials, then neither of them will see the other’s experiment. Similarly, if a licensed user logs in using a different set of credentials, then that user will not see any previously run experiments.

Can multiple Driverless AI users share a GPU server?

Yes, you can allocate multiple users in a single GPU box. For example, a single box with four GPUs can allocate that User1 has two GPUs and User2 has the other two GPUs. This is accomplished by having two separated Driverless AI instances running on the same server.

There are two ways to assign specific GPUs to Driverless AI. And in the scenario with four GPUs (two GPUs allocated to two users), both of these options allow each Docker container only to see two GPUs.

- Use the CUDA_VISIBLE_DEVICES environment variable. In the case of Docker deployment, this will translate in passing the `-e CUDA_VISIBLE_DEVICES="0,1"` to the `nvidia-docker run` command.
- Passing the NV_GPU option at the beginning of the `nvidia-docker run` command. (See example below.)

```
#Team 1
NV_GPU='0,1' nvidia-docker run
--pid=host
--init
--rm
--shm-size=256m
-u id -u:id -g
-p port-to-team:12345
-e DRIVERLESS_AI_CONFIG_FILE="/config/config.toml"
-v /data:/data
-v /log:/log
-v /license:/license
-v /tmp:/tmp
-v /config:/config
harbor.h2o.ai/h2oai/dai-centos7-x86_64:1.9.0-cuda10.0.23

#Team 2
NV_GPU='0,1' nvidia-docker run
--pid=host
--init
--rm
--shm-size=256m
-u id -u:id -g
-p port-to-team:12345
-e DRIVERLESS_AI_CONFIG_FILE="/config/config.toml"
-v /data:/data
-v /log:/log
-v /license:/license
-v /tmp:/tmp
-v /config:/config
harbor.h2o.ai/h2oai/dai-centos7-x86_64:1.9.0-cuda10.0.23
```

Note, however, that a Driverless AI instance expects to fully utilize and not share the GPUs that are assigned to it. Sharing a GPU with other Driverless AI instances or other running programs can result in out-of-memory issues.

How can I retrieve a list of Driverless AI users?

A list of users can be retrieved using the Python client.

```
h2o = Client(address='http://<client_url>:12345', username='<username>', password='<password>')
h2o.get_users()
```

Start of Driverless AI fails on the message ``Segmentation fault (core dumped)`` on Ubuntu 18/RHEL 7.6. How can I fix this?

This problem is caused by the font `NotoColorEmoji.ttf`, which cannot be processed by the Python matplotlib library. A workaround is to disable the font by renaming it. (Do not use `fontconfig` because it is ignored by `matplotlib`.) The following will print out the command that should be executed.

```
sudo find / -name "NotoColorEmoji.ttf" 2>/dev/null | xargs -I{} echo sudo mv {} {}.backup
```

Why do I receive a “shared object file” error when running on RHEL 8?

RHEL 8 is not currently supported. Supported Linux systems include x86_64 RHEL 7, CentOS 7, and SLES 12.

56.3 Data

Is there a file size limit for datasets?

For GBMs, the file size for datasets is limited by the collective CPU or GPU memory on the system, but we continue to make optimizations for getting more data into an experiment, such as using TensorFlow streaming to stream to arbitrarily large datasets.

How can I import CSV files that use UTF-8 encoding into Excel?

Excel requires a byte order mark (BOM) to correctly identify CSV files that use UTF-8 encoding. Refer to the following [FAQ entry](#) for more information on how to use a BOM when writing CSV files with datatable.

Can a byte order mark be used when writing CSV files with datatable?

Yes, a byte order mark (BOM) can be used when writing CSV files with datatable by enabling `datatable_bom_csv` in the config.toml file when starting Driverless AI.

Note: Support for UTF-8 encoding in Excel requires the use of a BOM.

56.4 Connectors

Why can't I import a folder as a file when using a data connector on Windows?

If you try to use the **Import Folder as File** option via a data connector on Windows, the import will fail if the folder contains files that do not have file extensions. For example, if a folder contains the files `file1.csv`, `file2.csv`, `file3.csv`, and `_SUCCESS`, the function will fail due to the presence of the `_SUCCESS` file.

Note that this only occurs if the data is sourced from a volume that is mounted from the Windows filesystem onto the Docker container via `-v /path/to/windows/filesystem:/path/in/docker/container` flags. This error occurs because of the difference in how files without file extensions are treated in Windows and in the Docker container (CentOS Linux).

I get a ClassNotFoundException error when I try to select a JDBC connection. How can I fix that?

The folder storing the JDBC jar file must be visible/readable by the dai process user.

If you downloaded the JDBC jar file from Oracle, they may provide you with a tar.gz file that you can unpack with the following command:

```
tar --no-same-permissions --no-same-owner -xzvf <my-jdbc-driver.tar>.gz
```

Alternatively you can ensure that the permissions on the file are correct in general by running the following:

```
chmod -R o+rx /path/to/folderContaining_jar_file
```

Finally, if you just want to check the permissions use the command `ls -altr` and check the final 3 values in the permissions output.

I get a org.datanucleus.exceptions.NucleusUserException: Please check your CLASSPATH and plugin specification error when attempting to connect to hive. How can I fix that?

Make sure `hive-site.xml` is configured in `/etc/hive/conf` and not in `/etc/hadoop/conf`.

56.5 Recipes

Where can I retrieve H2O's custom recipes?

H2O's custom recipes can be obtained from the official [Recipes for Driverless AI repository](#).

How can I create my own custom recipe?

Refer to the [How to Write a Recipe](#) guide for details on how to create your own custom recipe.

Are MOJOs supported for experiments that use custom recipes?

In most cases, MOJOs will not be available for custom recipes. Unless the recipe is simple, creating the MOJO is only possible with additional MOJO runtime support. Contact support@h2o.ai for more information about creating MOJOs for custom recipes. (**Note:** The Python Scoring Pipeline features full support for custom recipes.)

How can I use BYOR in my airgapped installation?

If your Driverless AI environment cannot access Internet and, thus, cannot access Driverless AI's "Bring Your Own Recipes" from GitHub, please contact [H2O support](#). We can work with you directly to help you access recipes.

56.6 Experiments

How much memory does Driverless AI require in order to run experiments?

Right now, Driverless AI requires approximately 10x the size of the data in system memory.

How many columns can Driverless AI handle?

Driverless AI has been tested on datasets with 10k columns. When running experiments on wide data, Driverless AI automatically checks if it is running out of memory, and if it is, it reduces the number of features until it can fit in memory. This may lead to a worse model, but Driverless AI shouldn't crash because the data is wide.

How should I use Driverless AI if I have large data?

Driverless AI can handle large datasets out of the box. For very large datasets (more than 10 billion rows x columns), we recommend sampling your data for Driverless AI. Keep in mind that the goal of driverless AI is to go through many features and models to find the best modeling pipeline, and not to just train a few models on the raw data (H2O-3 is ideally suited for that case).

For large datasets, the recommended steps are:

1. Run with the recommended accuracy/time/interpretability settings first, especially accuracy ≤ 7
2. Gradually increase accuracy settings to 7 and choose accuracy 9 or 10 only after observing runs with ≤ 7 .

How does Driverless AI detect the ID column?

The ID column logic is one of the following:

- The column is named 'id', 'Id', 'ID' or 'iD' exactly
- The column contains a significant number of unique values (above `max_relative_cardinality` in the config.toml file or **Max. allowed fraction of uniques for integer and categorical cols** in Expert settings)

Can Driverless AI handle data with missing values/nulls?

Yes, data that is imported into Driverless AI can include missing values. Feature engineering is fully aware of missing values, and missing values are treated as information - either as a special categorical level or as a special number. So for target encoding, for example, rows with a certain missing feature will belong to the same group. For Categorical Encoding where aggregations of a numeric columns are calculated for a grouped categorical column, missing values are kept. The formula for calculating the mean is the sum of non-missing values divided by the count of all non-missing values. For clustering, we impute missing values. And for frequency encoding, we count the number of rows that have a certain missing feature.

The imputation strategy is as follows:

- XGBoost/LightGBM do not need missing value imputation and may, in fact, perform worse with any specific other strategy unless the user has a strong understanding of the data.
- Driverless AI automatically imputes missing values using the mean for GLM.
- Driverless AI provides an imputation setting for TensorFlow in the config.toml file: `tf_nan_impute_value` post-normalization. If you set this option to 0, then missing values will be imputed. Setting it to (for example) +5 will specify 5 standard deviations outside the distribution. The default for TensorFlow is -5, which specifies that TensorFlow will treat NAs like a missing value. We recommend that you specify 0 if the mean is better.

More information is available in the [Missing and Unseen Values Handling](#) section.

How does Driverless AI deal with categorical variables? What if an integer column should really be treated as categorical?

If a column has string values, then Driverless AI will treat it as a categorical feature. There are multiple methods for how Driverless AI converts the categorical variables to numeric. These include:

- One Hot Encoding: creating dummy variables for each value
- Frequency Encoding: replace category with how frequently it is seen in the data
- Target Encoding: replace category with the average target value (additional steps included to prevent overfitting)
- Weight of Evidence: calculate weight of evidence for each category (<http://ucanalytics.com/blogs/information-value-and-weight-of-evidencebanking-case/>)

Driverless AI will try multiple methods for representing the column and determine which representation(s) are best.

If the column has integers, Driverless AI will try treating the column as a categorical column and numeric column. It will treat any integer column as both categorical and numeric if the number of unique values is less than 50.

This is configurable in the config.toml file:

```
# Whether to treat some numerical features as categorical
# For instance, sometimes an integer column may not represent a numerical feature but
# represents different numerical codes instead.
num_as_cat = true

# Max number of unique values for integer/real columns to be treated as categoricals (test applies to first statistical_threshold_data_
# size_small rows only)
max_int_as_cat_uniques = 50
```

(Note: Driverless AI will also check if the distribution of any numeric column differs significantly from the distribution of typical numerical data using Benford's Law. If the column distribution does not obey Benford's Law, we will also try to treat it as categorical even if there are more than 50 unique values.)

How are outliers handled?

Outliers are not removed from the data. Instead Driverless AI finds the best way to represent data with outliers. For example, Driverless AI may find that binning a variable with outliers improves performance.

For target columns, Driverless AI first determines the best representation of the column. It may find that for a target column with outliers, it is best to predict the log of the column.

If I drop several columns from the Train dataset, will Driverless AI understand that it needs to drop the same columns from the Test dataset?

If you drop columns from the training dataset, Driverless AI will do the same for the validation and test datasets (if the columns are present). There is no need for these columns because no features will be created from them.

Does Driverless AI treat numeric variables as categorical variables?

In certain cases, yes. You can prevent this behavior by setting the num_as_cat variable in your installation's config.toml file to false. You can have finer grain control over this behavior by excluding the Numeric to Categorical Target Encoding Transformer and the Numeric To Categorical Weight of Evidence Transformer and their corresponding genes in your installation's config.toml file. To learn more about the config.toml file, see the [Using the config.toml File](#) section.

Which algorithms are used in Driverless AI?

Features are engineered with a proprietary stack of Kaggle-winning statistical approaches including some of the most sophisticated target encoding and likelihood estimates based on groupings, aggregations and joins, but we also employ linear models, neural nets, clustering and dimensionality reduction models and many traditional approaches such as one-hot encoding etc.

On top of the engineered features, sophisticated models are fitted, including, but not limited to: XGBoost (both original XGBoost and ‘lossguide’ (LightGBM) mode), Decision Trees, GLM, TensorFlow (including a TensorFlow NLP recipe based on CNN Deeplearning models), RuleFit, FTRL (Follow the Regularized Leader), Isolation Forest, and Constant Models. (Refer to [Supported Algorithms](#) for more information.) And additional algorithms can be added via [Recipes](#).

In general, GBMs are the best single-shot algorithms. Since 2006, boosting methods have proven to be the most accurate for noisy predictive modeling tasks outside of pattern recognition in images and sound (<https://www.cs.cornell.edu/~caruana/ctp/ct.papers/caruana.icml06.pdf>). The advent of XGBoost and Kaggle only cemented this position.

Why do my selected algorithms not show up in the Experiment Preview?

When changing the algorithms used via **Expert Settings > Model** and **Expert Settings > Recipes**, you may notice in the Experiment Preview that those changes are not applied. Driverless AI determines whether to include models and/or recipes based on a hierarchy of those expert settings.

The screenshots illustrate the 'Expert Experiment Settings' interface. The top screenshot shows the 'MODEL' tab selected, listing various machine learning models with configuration options: XGBoost GBM models (AUTO, ON, OFF), XGBoost Dart models (AUTO, ON, OFF), GLM models (AUTO, ON, OFF), Decision Tree models (AUTO, ON, OFF), LightGBM models (AUTO, ON, OFF), TensorFlow models (DISABLED), FTRL models (AUTO, ON, OFF), RuleFit models (AUTO, ON, OFF), LightGBM Boosting types (SELECT VALUES), and Parameters for TensorFlow (0) with a max value of 3000. The bottom screenshot shows the 'RECIPES' tab selected, displaying probability settings for various experimental operations: Probability to add transformers (0.5), Probability to prune transformers (0.5), Probability to mutate model parameters (0.25), and Probability to prune weak features (0.25).

- Setting an Algorithm to “OFF” in Expert Settings: If an algorithm is turned OFF in Expert Settings (for example, **GLM Models**) when running, then that algorithm will not be included in the experiment.
- Algorithms Not Included from Recipes (BYOR): If an algorithm from a custom recipe is not selected for the experiment in the **Include specific models** option, then that algorithm will not be included in the experiment, regardless of whether that same algorithm is set to AUTO or ON on the **Expert Settings > Model** page.
- Algorithms Not Specified as “OFF” and Included from Recipes: If a Driverless AI algorithm is specified as either “AUTO” or “ON” and additional models are selected for the experiment in the **Include specific models** option, than those algorithms may or may not be included in the experiment. Driverless AI will determine the algorithms to use based on the data and experiment type.

How can we turn on TensorFlow Neural Networks so they are evaluated?

Neural networks are considered by Driverless AI, although they may not be evaluated by default. To ensure that neural networks are tried, you can turn on TensorFlow in the Expert Settings:

Expert Experiment Settings

+ UPLOAD CUSTOM RECIPE + LOAD CUSTOM RECIPE FROM URL

GENERAL	MODEL	TIMESERIES	NLP	SYSTEM	CUSTOM RECIPES	OTHERS
XGBoost GBM models	XGBoost Dart models	XGBoost Dart models	GLM models	ON	LightGBM models	
AUTO ON OFF	AUTO ON OFF	AUTO ON OFF	AUTO ON OFF	ON OFF	AUTO ON OFF	
LightGBM Random Forest models	TensorFlow models	TensorFlow models	RuleFit models	ON	FTRL models	
AUTO▼	AUTO ON OFF	AUTO ON OFF	AUTO ON OFF	ON OFF	AUTO ON OFF	
Max. number of trees/iterations 3000	Reduction factor for number of trees/iterations during feature evolution 0.2	Max. learning rate for tree models 0.5	Max. number of epochs for TensorFlow / FTRL 10			
Max. number of rules for RuleFit (-1 for all) -1						

Once you have set TensorFlow to ON. You should see the Experiment Preview on the left hand side change and mention that it will evaluate TensorFlow models:

What do these settings mean?

ACCURACY
 - Training data size: **23,999 rows, 25 cols**
 - Feature evolution: **[LightGBM, XGBoost, TensorFlow], 1/3 validation split**
 - Final pipeline: **Ensemble (10 models), 5-fold CV**

TIME
 - Feature evolution: **8 individuals, up to 48 iterations**
 - Early stopping: **After 5 iterations of no improvement**

INTERPRETABILITY
 - Feature pre-pruning strategy: **FS**
 - XGBoost Monotonicity constraints: **disabled**
 - Feature engineering search space (where applicable): **(Date, FrequencyEncoding, Identity, Interactions, IsHoliday, NumEncoding, OneHotEncoding, TargetEncoding, Text, TextLin, WeightOfEvidence)**

[LightGBM, XGBoost, TensorFlow] models to train:
 - Model and feature tuning: **32**
 - Feature evolution: **168**
 - Final pipeline: **10**

TARGET COLUMN
DEFAULT_PAYMENT_NE --

WEIGHT COLUMN
--

TIME COLUMN
[OFF]

TYPE bool **COUNT** 23999 **UNIQUE** 2 **TARGET FREQ** 5369

EXPERIMENT SETTINGS
 ACCURACY: 6 TIME: 4 INTERPRETABILITY: 6

EXPERT SETTINGS
 CLASSIFICATION REPRODUCIBLE ENABLE GPUs

SCORER
 GINI
 MCC
 F05
 F1
 F2
 ACCURACY
 LOGLOSS
 AUC
 AUCPR

LAUNCH EXPERIMENT

We recommend using TensorFlow neural networks if you have a multinomial use case with more than 5 unique values.

Does Driverless AI standardize the data?

Driverless AI will automatically do variable standardization for certain algorithms. For example, with Linear Models and Neural Networks, the data is automatically standardized. For decision tree algorithms, however, we do not perform standardization since these algorithms do not benefit from standardization.

What objective function is used in XGBoost?

The objective function used in XGBoost is:

- reg:linear and custom absolute error objective function for regression
- binary:logistic or multi:softprob for classification

The objective function does not change depending on the scorer chosen. The scorer influences parameter tuning only.

For regression, Tweedie/Gamma/Poisson/etc. regression is not yet supported, but Driverless AI handles various target transforms so many target distributions can be handled very effectively already. Driverless AI handles quantile regression for alpha=0.5 (media), and general quantiles are on the roadmap.

Further details for the XGBoost instantiations can be found in the logs and in the model summary, both of which can be downloaded from the GUI or are found in the `/tmp/h2oai_experiment_<name>/` folder on the server.

Does Driverless AI perform internal or external validation?

Driverless AI does internal validation when only training data is provided. It does external validation when training and validation data are provided. In either scenario, the validation data is used for all parameter tuning (models and features), not just for feature selection. Parameter tuning includes target transformation, model selection, feature engineering, feature selection, stacking, etc.

Specifically:

- Internal validation (only training data given):
 - Ideal when data is either close to i.i.d., or for time-series problems
 - Internal holdouts are used for parameter tuning, with temporal causality for time-series problems
 - Will do the full spectrum from single holdout split to 5-fold CV, depending on accuracy settings
 - No need to split training data manually
 - Final models are trained using CV on the training data
- External validation (training + validation data given):
 - Ideal when there's some amount of drift in the data, and the validation set mimics the test set data better than the training data
 - No training data wasted during training because training data not used for parameter tuning
 - Validation data is used only for parameter tuning, and is not part of training data
 - No CV possible because we explicitly do not want to overfit on the training data
 - Not allowed for time-series problems (see Time Series FAQ section that follows)

Tip: If you want both training and validation data to be used for parameter tuning (the training process), just concatenate the datasets together and turn them both into training data for the “internal validation” method.

How does Driverless AI prevent overfitting?

Driverless AI performs a number of checks to prevent overfitting. For example, during certain transformations, Driverless AI calculates the average on out-of-fold data using cross validation. Driverless AI also performs early stopping for every model built, ensuring that the model build will stop when it ceases to improve on holdout data. And additional steps to prevent overfitting include checking for i.i.d. and avoiding leakage during feature engineering.

A blog post describing Driverless AI overfitting protection in greater detail is available here: <https://www.h2o.ai/blog/driverless-ai-prevents-overfitting-leakage/>.

How does Driverless AI avoid the multiple hypothesis (MH) problem?

Or more specifically, like many brute force methods for tuning hyperparameters/model selection, Driverless AI runs up against the multihypothesis problem (MH). For example, if I randomly generated a gazillion models, the odds that a few will do awesome on the test data that they are all measured against is pretty large, simply by sheer luck. How does Driverless AI address this?

Driverless AI uses a variant of the reusable holdout technique to address the multiple hypothesis problem. Refer to <https://pdfs.semanticscholar.org/25fe/96591144f4af3d8f8f79c95b37f415e5bb75.pdf> for more information.

How does Driverless AI suggest the experiment settings?

When you run an experiment on a dataset, the experiment settings (Accuracy, Time, and Interpretability) are automatically suggested by Driverless AI. For example, Driverless AI may suggest the parameters Accuracy = 7, Time = 3, Interpretability = 6, based on your data.

Driverless AI will automatically suggest experiment settings based on the number of columns and number of rows in your dataset. The settings are suggested to ensure best handling when the data is small. If the data is small, Driverless AI will suggest the settings that prevent overfitting and ensure the full dataset is utilized.

If the number of rows and number of columns are each below a certain threshold, then:

- Accuracy will be increased up to 8.
 - The accuracy is increased so that cross validation is done. (We don’t want to “throw away” any data for internal validation purposes.)
- Interpretability will be increased up to 8.
 - The higher the interpretability setting, the smaller the number of features in the final model.
 - More complex features are not allowed.
 - This prevents overfitting.
- Time will be decreased down to 2.
 - There will be fewer feature engineering iterations to prevent overfitting.

What happens when I set Interpretability and Accuracy to the same number?

The answer is currently that interpretability controls which features are created and what features are kept. (Also above interpretability = 6, monotonicity constraints are used in XGBoost GBM, XGBoost Dart, LightGBM, and Decision Tree models.) The accuracy refers to how hard Driverless AI then tries to make those features into the most accurate model

Can I specify the number of GPUs to use when running Driverless AI?

When running an experiment, the Expert Settings allow you to specify the starting GPU ID for Driverless AI to use. You can also specify the maximum number of GPUs to use per model and per experiment. Refer to the [Expert Settings](#) section for more information.

How can I create the simplest model in Driverless AI?

To create the simplest model in Driverless AI, set the following Experiment Settings:

- Set Accuracy to 1. Note that this can hurt performance as a sample will be used. If necessary, adjust the knob until the preview shows no sampling.
- Set Time to 1.
- Set Interpretability to 10.

Next, configure the following Expert Settings:

- Turn OFF all algorithms except GLM.
- Set **GLM models** to ON.
- Set **Ensemble level** to 0.
- Set **Select target transformation of the target for regression problems** to Identity.
- Disable **Data distribution shift detection**.
- Disable **Target Encoding**.

Alternatively, you can set **Pipeline Building Recipe** to Compliant. Compliant automatically configures the following experiment and expert settings:

- interpretability=10 (To avoid complexity. This overrides GUI or Python client settings for Interpretability.)
- enable_glm='on' (Remaining algos are 'off', to avoid complexity and be compatible with algorithms supported by MLI.)
- num_as_cat=true: Treat some numerical features as categorical. For instance, sometimes an integer column may not represent a numerical feature but represent different numerical codes instead.
- fixed_ensemble_level=0: Don't use any ensemble (to avoid complexity).
- feature_brain_level=0: No feature brain used (to ensure every restart is identical).
- max_feature_interaction_depth=1: Interaction depth is set to 1 (no multi-feature interactions to avoid complexity).
- target_transformer="identity": For regression (to avoid complexity).
- check_distribution_shift="off": Don't use distribution shift between train, valid, and test to drop features (bit risky without fine-tuning).

Why is my experiment suddenly slow?

It is possible that your experiment has gone from using GPUs to using CPUs due to a change of the host system outside of Driverless AI's control. You can verify this using any of the following methods:

- Check GPU usage by going to your Driverless AI experiment page and clicking on the GPU USAGE tab in the lower-right quadrant of the experiment.
- Run `nvidia-smi` in a terminal to see if any processes are using GPU resources in an unexpected way (such as those using a large amount of memory).
- Check if System/GPU memory is being consumed by prior jobs or other tasks or if older jobs are still running some tasks.

- Check and disable automatic NVIDIA driver updates on your system (as they can interfere with running experiments).

The general solution to these kind of sudden slowdown problems is to restart:

- Restart Docker if using Docker
- `pkill --signal 9 h2oai` if using the native installation method
- Restart the system if `nvidia-smi` does not work as expected (e.g., after a driver update)

More ML-related issues that can lead to a slow experiment are:

- Choosing high accuracy settings on a system with insufficient memory
- Choosing low interpretability settings (can lead to more feature engineering which can increase memory usage)
- Using a dataset with a lot of columns (> 500)
- Doing multi-class classification with a GBM model when there are many target classes (> 5)

When I run multiple experiments with different seeds, why do I see different scores, runtimes, and sizes on disk in the Experiments listing page?

When running multiple experiments with all of the same settings except the seed, understand that a feature brain level > 0 can lead to variations in models, features, timing, and sizes on disk. (The default value is 2.) These variations can be disabled by setting the **Feature Brain Level** to 0 in the *Expert Settings* or in the config.toml file.

In addition, if you use a different seed for each experiment, then each experiment can be different due to the randomness in the genetic algorithm that searches for the best features and model parameters. Only if **Reproducible** is set with the same seed and with a feature brain level of 0 should users expect the same outcome. Once a different seed is set, the models, features, timing, and sizes on disk can all vary within the constraints set by the choices made for the experiment. (I.e., accuracy, time, interpretability, expert settings, etc., all constrain the outcome, and then a different seed can change things within those constraints.)

Why does the final model performance appear to be worse than previous iterations?

There are a few things to remember:

- Driverless AI creates a best effort *estimate* of the *generalization performance* of the best modeling pipeline found so far.
- The performance estimation is always based on holdout data (data unseen by the model).
- If no validation dataset is provided, the training data is split internally to create *internal validation* holdout data (once or multiple times or cross-validation, depending on the *accuracy* settings).
- If no validation dataset is provided, for accuracy ≤ 7 , a single holdout split is used, and a “lucky” or “unlucky” split can bias estimates for small datasets or datasets with high variance.
- If a validation dataset is provided, then all performance estimates are solely based on the entire validation dataset (independent of *accuracy* settings).
- All scores reported are based on bootstrapped-based statistical methods and come with *error bars* that represent a range of estimate uncertainty.

After the final iteration, a *best* final model is trained on a final set of engineered features. Depending on *accuracy* settings, a more accurate estimation of generalization performance may be done using cross-validation. Also, the final model may be a stacked ensemble consisting of multiple base models, which generally leads to better performance. Consequently, in rare cases, the difference in performance estimation method can lead to the final model’s estimated performance seeming poorer than those from previous

iterations. (i.e., The final model's estimated score is significantly worse than the last iteration score and error bars don't overlap.) In that case, it is very likely that the final model performance estimation is more accurate, and the prior estimates were biased due to a "lucky" split. To confirm this, you can re-run the experiment multiple times (without setting the reproducible flag).

If you would like to minimize the likelihood of the final model performance appearing worse than previous iterations, here are some recommendations:

- Increase accuracy settings
- Provide a validation dataset
- Provide more data

How can I find features that may be causing data leakages in my Driverless AI model?

To find original features that are causing leakage, have a look at features_orig.txt in the experiment summary download. Features causing leakage will have high importance there. To get a hint at derived features that might be causing leakage, create a new experiment with dials set to 2/2/8, and run the new experiment on your data with all your features and response. Then analyze the top 1-2 features in the model variable importance. They are likely the main contributors to data leakage if it is occurring.

How can I see the performance metrics on the test data?

As long as you provide a target column in the test set, Driverless AI will show the best estimate of the final model's performance on the test set at the end of the experiment. The test set is never used to tune parameters (unlike to what Kagglers often do), so this is purely a convenience. Of course, you can still make test set predictions and compute your own metrics using a method of your choice.

How can I see all the performance metrics possible for my experiment?

At the end of the experiment, the model's estimated performance on all provided datasets with a target column is printed in the experiment logs. For example, for the test set:

```
Final scores on test (external holdout) +/- stddev:
  GINI = 0.87794 +/- 0.035305 (more is better)
  MCC = 0.71124 +/- 0.043232 (more is better)
  F05 = 0.79175 +/- 0.04209 (more is better)
  F1 = 0.75823 +/- 0.038675 (more is better)
  F2 = 0.82752 +/- 0.03604 (more is better)
  ACCURACY = 0.91513 +/- 0.011975 (more is better)
  LOGLOSS = 0.28429 +/- 0.016682 (less is better)
  AUCPR = 0.79074 +/- 0.046223 (more is better)
optimized: AUC = 0.93386 +/- 0.018856 (more is better)
```

What if my training/validation and testing data sets come from different distributions?

In general, Driverless AI uses training data to engineer features and train models and validation data to tune all parameters. If no external validation data is given, the training data is used to create internal holdouts. The way holdouts are created internally depends on whether there is a strong time dependence, see the point below. If the data has no obvious time dependency (e.g., if there is no time column neither implicit or explicit), or if the data can be sorted arbitrarily and it won't affect the outcome (e.g., Iris data, predicting flower species from measurements), and if the test dataset is different (e.g., new flowers or only large flowers), then the model performance on validation (either internal or external) as measured during training won't be achieved during final testing due to the obvious inability of the model to generalize.

Does Driverless AI handle weighted data?

Yes. You can optionally provide an extra weight column in your training (and validation) data with non-negative observation weights. This can be useful to implement domain-specific effects such as exponential weighting in time or class weights. All of our algorithms and metrics in Driverless AI support observation weights, but note that estimated likelihoods can be skewed as a consequence.

How does Driverless AI handle fold assignments for weighted data?

Currently, Driverless AI does not take the weights into account during fold creation, but you can provide a fold column to enforce your own grouping, i.e., to keep rows that belong to the same group together

(either in train or valid). The fold column has to be a categorical column (integers ok) that assigns a group ID to each row. (It needs to have at least 5 groups because we do up to 5-fold CV.)

Why do I see that adding new features to a dataset deteriorates the performance of the model?

You may notice that after adding one or more new features to a dataset, it deteriorates the performance of the Driverless AI model. In Driverless AI, the feature engineering sequence is fairly random and may end up not doing same things with original features if you restart entirely fresh with new columns.

Beginning in Driverless AI v1.4.0, you now have the option to **Restart from Last Checkpoint**. This allows you to pull in a new dataset with more columns, and Driverless AI will more iteratively take advantage of the new columns.

How does Driverless AI handle imbalanced data for binary classification experiments?

If you have data that is imbalanced, a binary imbalanced model can help to improve scoring with a variety of imbalanced sampling methods. An imbalanced model is able to take advantage of most (or even all) of the imbalanced dataset's positive values during sampling, while a regular model significantly limits the population of positive values. Imbalanced models, however, take more time to make predictions, and they are not always more accurate than regular models. We still recommend that you try using an imbalanced model if your data is imbalanced to see if scoring is improved over a regular model. Note that this information only applies to binary models.

56.7 Feature Transformations

Where can I get details of the various transformations performed in an experiment?

Download the experiment's log .zip file from the GUI. This zip file includes summary information, log information, and a gene_summary.txt file with details of the transformations used in the experiment. Specifically, there is a **details** folder with all subprocess logs.

On the server, the experiment specific files are inside the `/tmp/h2oai_experiment_<name>/` folder after the experiment completes, particularly `h2oai_experiment_logs_<name>.zip` and `h2oai_experiment_summary_<name>.zip`.

56.8 Predictions

How can I download the predictions onto the machine where Driverless AI is running?

When you select **Score on Another Dataset**, the predictions will automatically be stored on the machine where Driverless AI is running. They will be saved in the following locations (and can be opened again by Driverless AI, both for .csv and .bin):

- Training Data Predictions: `tmp/h2oai_experiment_<name>/train_preds.csv` (also saved as .bin)
- Testing Data Predictions: `tmp/h2oai_experiment_<name>/test_preds.csv` (also saved as .bin)
- New Data Predictions: `tmp/h2oai_experiment_<name>/automatically_generated_name.csv`. Note that the automatically generated name will match the name of the file downloaded to your local computer.

Why are predicted probabilities not available when I run an experiment without ensembling?

When Driverless AI provides pre-computed predictions after completing an experiment, it uses only those parts of the modeling pipeline that were not trained on the particular rows for which the predictions are made. This means that Driverless AI needs holdout data in order to create predictions, such as validation or test sets, where the model is trained on training data only. In the case of ensembles, Driverless AI

uses cross-validation to generate holdout folds on the training data, so we are able to provide out-of-fold estimates for every row in the training data and, hence, can also provide training holdout predictions (that will provide a good estimate of generalization performance). In the case of a single model, though, that is trained on 100% of the training data. There is no way to create unbiased estimates for any row in the training data. While DAI uses an internal validation dataset, this is a re-usable holdout, and therefore will not contain holdout predictions for the full training dataset. You need cross-validation in order to get out-of-fold estimates, and then that's not a single model anymore. If you want to still get predictions for the training data for a single model, then you have to use the scoring API to create predictions on the training set. From the GUI, this can be done using the **Score on Another Dataset** button for a completed experiment. Note, though, that the results will likely be overly optimistic, too good to be true, and virtually useless.

56.9 Deployment

What drives the size of a MOJO?

The size of the MOJO is based on the complexity of the final modeling pipeline (i.e., feature engineering and models). One of the biggest factors is the amount of higher-order interactions between features, especially target encoding and related features, which have to store lookup tables for all possible combinations observed in the training data. You can reduce the amount of these transformations by reducing the value of **Max. feature interaction depth** and/or **Feature engineering effort** under Expert Settings, or by increasing the interpretability settings for the experiment. Ensembles also contribute to the final modeling pipeline's complexity as each model has its own pipeline. Lowering the accuracy settings or increasing the `ensemble_accuracy_switch` setting in the config.toml file can help here. The number of features **Max. pipeline features** also affects the MOJO size. Text transformers are pretty bulky as well and can add to the MOJO size.

Are MOJOS thread safe?

Yes, all Driverless AI MOJOS are thread safe.

Running the scoring pipeline for my MOJO is taking several hours. How can I get this to run faster?

When running example.sh, Driverless AI implements a memory setting, which is suitable for most use cases. For very large models, however, it may be necessary to increase the memory limit when running the Java application for data transformation. This can be done using the `-Xmx25g` parameter. For example:

```
java -Xmx25g -Dai.h2o.mojos.runtime.license.file=license.sig -cp mojo2-runtime.jar ai.h2o.mojos.ExecuteMojo pipeline.mojo example.csv
```

Why have I encountered a “Best Score is not finite” error?

Driverless AI uses 32-bit floats by default. You may encounter this error if your data value exceeds 1E38 or if you are resolving more than 1 part in 10 million. You can resolve this error using one of the following methods:

- Enable the **Force 64-bit Precision** option in the experiment's Expert Settings.
- or
- Set `data_precision="float64"` and `transformer_precision="float64"` in config.toml.

56.10 Time Series

What if my data has a time dependency?

If you know that your data has a strong time dependency, select a time column before starting the experiment. The time column must be in a Datetime format that can be parsed by pandas, such as “2017-11-06 14:32:21”, “Monday, June 18, 2012” or “Jun 18 2018 14:34:00” etc., or contain only integers.

If you are unsure about the strength of the time dependency, run two experiments: One with time column set to “[OFF]” and one with time column set to “[AUTO]” (or pick a time column yourself).

What is a lag, and why does it help?

A lag is a feature value from a previous point in time. Lags are useful to take advantage of the fact that the current (unknown) target value is often correlated with previous (known) target values. Hence, they can better capture target patterns along the time axis.

Why can't I specify a validation data set for time-series problems? Why do you look at the test set for time-series problems

The problem with validation vs test in the time series setting is that there is only one valid way to define the split. If a test set is given, its length in time defines the validation split and the validation data has to be part of train. Otherwise the time-series validation won't be useful.

For instance: Let's assume we have train = [1,2,3,4,5,6,7,8,9,10] and test = [12,13], where integers define time periods (e.g., weeks). For this example, the most natural train/valid split that mimics the test scenario would be: train = [1,2,3,4,5,6,7] and valid = [9,10], and month 8 is not included in the training set to allow for a gap. Note that we will look at the start time and the duration of the test set only (if provided), and not at the contents of the test data (neither features nor target). If the user provides validation = [8,9,10] instead of test data, then this could lead to inferior validation strategy and worse generalization. Hence, we use the user-given test set only to create the optimal internal train/validation splits. If no test set is provided, the user can provide the length of the test set (in periods), the length of the train/test gap (in periods) and the length of the period itself (in seconds).

Why does the gap between train and test matter? Is it because of creating the lag features on the test set?

Taking the gap into account is necessary in order to avoid too optimistic estimates of the true error and to avoid creating history-based features like lags for the training and validation data (which cannot be created for the test data due to the missing information).

In regards to applying the target lags to different subsets of the time group columns, are you saying Driverless AI perform auto-correlation at “levels” of the time series? For example, consider the Walmart dataset where I have Store and Dept (and my target is Weekly Sales). Are you saying that Driverless AI checks for auto-correlation in Weekly Sales based on just Store, just Dept, and both Store and Dept?

Currently, auto-correlation is only applied on the detected superkey (entire TGC) of the training dataset relation at the very beginning. It's used to rank potential lag-sizes, with the goal to prune the search space for the GA optimization process, which is responsible for selecting the lag features.

How does Driverless AI detect the time period?

Driverless AI treats each time series as a function with some frequency 1/ns. The actual value is estimated by the median of time deltas across maximal length TGC subgroups. The chosen SI unit minimizes the distance to all available SI units.

What is the logic behind the selectable numbers for forecast horizon length?

The shown forecast horizon options are based on quantiles of valid splits. This is necessary because Driverless AI cannot display all possible options in general.

Assume that in my Walmart dataset, all stores provided data at the week level, but one store provided data at the day level. What would Driverless AI do?

Driverless AI would still assume “weekly data” in this case because the majority of stores are yielding this property. The “daily” store would be resampled to the detected overall frequency.

Assume that in my Walmart dataset, all stores and departments provided data at the weekly level, but one department in a specific store provided weekly sales on a bi-weekly basis (every two weeks). What would Driverless AI do?

That’s similar to having missing data. Due to proper resampling, Driverless AI can handle this without any issues.

Why does the number of weeks that you want to start predicting matter?

That’s an option to provide a train-test gap if there is no test data is available. That is to say, “I don’t have my test data yet, but I know it will have a gap to train of x.”

Are the scoring components of time series sensitive to the order in which new pieces of data arrive? I.e., is each row independent at scoring time, or is there a real-time windowing effect in the scoring pieces?

Each row is independent at scoring time.

What happens if the user, at predict time, gives a row with a time value that is too small or too large?

Internally, “out-of bounds” time values are encoded with special values. The samples will still be scored, but the predictions won’t be trustworthy.

What’s the minimum data size for a time series recipe?

We recommended that you have around 10,000 validation samples in order to get a reliable estimate of the true error. The time series recipe can still be applied for smaller data, but the validation error might be inaccurate.

How long must the training data be compared to the test data?

At a minimum, the training data has to be at least twice as long as the test data along the time axis. However, we recommended that the training data is at least three times as long as the test data.

How does the time series recipe deal with missing values?

Missing values will be converted to a special value, which is different from any non-missing feature value. Explicit imputation techniques won’t be applied.

Can the time information be distributed across multiple columns in the input data (such as [year, day, month])?

Currently Driverless AI requires the data to have the time stamps given in a single column. Driverless AI will create additional time features like [year, day, month] on its own, if they turn out to be useful.

What type of modeling approach does Driverless AI use for time series?

Driverless AI combines the creation of history-based features like lags, moving averages etc. with the modeling techniques, which are also applied for i.i.d. data. The primary model of choice is XGBoost.

What’s the idea behind exponential weighting of moving averages?

Exponential weighting accounts for the possibility that more recent observations are better suited to explain the present than older observations.

56.11 Logging

How can I reduce the size of the Audit Logger?

An Audit Logger file is created every day that Driverless AI is in use. The `audit_log_retention_period` config variable allows you to specify the number of days, after which the `audit.log` will be overwritten. This option defaults to 5 days, which means that Driverless AI will maintain Audit Logger files for the last 5 days, and `audit.log` files older than 5 days are removed and replaced with newer log files. When this option is set to 0, the `audit.log` file will not be overwritten.

TIPS ‘N TRICKS

This section includes Arno’s tips for running Driverless AI.

57.1 Pipeline Tips

Given training data and a target column to predict, H2O Driverless AI produces an end-to-end pipeline tuned for high predictive performance (and/or high interpretability) for general classification and regression tasks. The pipeline has only one purpose: to take a test set, row by row, and turn its feature values into predictions.

A typical pipeline creates dozens or even hundreds of derived features from the user-given dataset. Those transformations are often based on precomputed lookup tables and parameterized mathematical operations that were selected and optimized during training. It then feeds all these derived features to one or several machine learning algorithms such as linear models, deep learning models, or gradient boosting models (and several more derived models). If there are multiple models, then their output is post-processed to form the final prediction (either probabilities or target values). The pipeline is a directed acyclic graph.

It is important to note that the training dataset is processed as a whole for better results (e.g., aggregate statistics). For scoring, however, every row of the test dataset must be processed independently to mimic the actual production scenario.

To facilitate deployment to various production environments, there are multiple ways to obtain predictions from a completed Driverless AI experiment, either from the GUI, from the R or Python client API, or from a standalone pipeline.

GUI

- **Score on Another Dataset** - Convenient, parallelized, ideal for imported data
- **Download Predictions** - Available if a test set was provided during training
- **Deploy** - Creates an Amazon Lambda endpoint (more endpoints coming soon)
- **Diagnostics** - Useful if the test set includes a target column

Client APIs

- **Python client** - Use the `make_prediction_sync()` method. An optional argument can be used to get per-row and per-feature ‘Shapley’ prediction contributions. (Pass `pred_contribs=True`.)
- **R client** - Use the `predict()` method. An optional argument can be used to get per-row and per-feature ‘Shapley’ prediction contributions. (Pass `pred_contribs=True`.)

Standalone Pipelines

- **Python** - Supports all models and transformers, and supports ‘Shapley’ prediction contributions and MLI reason codes

- **Java** - Most portable, low latency, supports all models and transformers that are enabled by default (except TensorFlow NLP transformers), can be used in Spark/H2O-3/SparklingWater for scale
- **C++** - Highly portable, low latency, standalone runtime with a convenient Python and R wrapper

57.2 Time Series Tips

H2O Driverless AI handles time-series forecasting problems out of the box.

All you need to do when starting a time-series experiment is to provide a regular columnar dataset containing your features. Then pick a target column and also pick a “time column” - a designated column containing time stamps for every record (row) such as “April 10 2019 09:13:41” or “2019/04/10”. If you have a test set for which you want predictions for every record, make sure to provide future time stamps and features as well.

In most cases, that’s it. You can launch the experiment and let Driverless AI do the rest. It will even auto-detect multiple time series in the same dataset for different groups such as weekly sales for stores and departments (by finding the columns that identify stores and departments to group by). Driverless AI will also auto-detect the time period including potential gaps during weekends, as well as the forecast horizon, a possible time gap between training and testing time periods (to optimize for deployment delay) and even keeps track of holiday calendars. Of course, it automatically creates multiple causal time-based validation splits (sliding time windows) for proper validation, and incorporates many other related grand-master recipes such as automatic target and non-target lag feature generation as well as interactions between lags, first and second derivatives and exponential smoothing.

- If you find that the automatic lag-based time-series recipe isn’t performing well for your dataset, we recommend that you try to disable the creation of lag-based features by disabling “Time-series lag-based recipe” in the expert settings. This will lead to regular feature engineering but with time-based causal validation splits. Especially for small datasets and short forecast periods, this can lead to better results.
- If the target column is present in the test set and has partially filled information (non-missing values), then Driverless AI will automatically augment the model with those future target values to make better predictions. This can be used to extend the usable lifetime of the model into the future without the need for retraining by providing past known outcomes. Contact us if you’re interested in learning more about test-time augmentation.
- For now, training and test datasets should have the same input features available, so think about which of the predictors (input features) will be available during production time and drop the rest (or create your own lag features that can be available to both train and test sets).
- For datasets that are non-stationary in time, create a test set from the last temporal portion of data, and create time-based features. This allows the model to be optimized for your production scenario.
- We are working on further improving many aspects of our time-series recipe. For example, we will add support to automatically generate lags for features that are only available in the training set, but not in the test set, such as environmental or economic factors. We’ll also improve the performance of back-testing using rolling windows.
- In 1.7.x, you will have the option to bring your own recipes (BYOR) for features, models and scorers, and that includes time-series recipes! We are very excited about that. Please contact us if you are interested in learning more about BYOR.

57.3 Scorer Tips

A core capability of H2O Driverless AI is the creation of automatic machine learning modeling pipelines for supervised problems. In addition to the data and the target column to be predicted, the user can pick a scorer. A scorer is a function that takes actual and predicted values for a dataset and returns a number. Looking at this single number is the most common way to estimate the generalization performance of a predictive model on unseen data by comparing the model's predictions on the dataset with its actual values. There are more detailed ways to estimate the performance of a machine learning model such as residual plots (available on the Diagnostics page in Driverless AI), but we will focus on scorers here.

For a given scorer, Driverless AI optimizes the pipeline to end up with the best possible score for this scorer. The default scorer for regression problems is RMSE (root mean squared error), where 0 is the best possible value. For example, for a dataset containing 4 rows, if actual target values are [1, 1, 10, 0], but predictions are [2, 3, 4, -1], then the RMSE is $\sqrt{(1+4+36+1)/4}$ and the largest misprediction dominates the overall score (quadratically). Driverless AI will focus on improving the predictions for the third data point, which can be very difficult when hard-to-predict outliers are present in the data. If outliers are not that important to get right, a metric like the MAE (mean absolute error) can lead to better results. For this case, the MAE is $(1+2+6+1)/4$ and the optimization process will consider all errors equally (linearly). Another scorer that is robust to outliers is RMSLE (root mean square logarithmic error), which is like RMSE but after taking the logarithm of actual and predicted values - however, it is restricted to positive values. For price predictions, scorers such as MAPE (mean absolute percentage error) or MER (median absolute percentage error) are useful, but have problems with zero or small positive values. SMAPE (symmetric mean absolute percentage error) is designed to improve upon that.

For classification problems, the default scorer is either the AUC (area under the receiver operating characteristic curve) or LOGLOSS (logarithmic loss) for imbalanced problems. LOGLOSS focuses on getting the probabilities right (strongly penalizes wrong probabilities), while AUC is designed for ranking problems. Gini is similar to the AUC, but measures the quality of ranking (inequality) for regression problems. For general imbalanced classification problems, AUCPR and MCC are good choices, while F05, F1 and F2 are designed to balance recall against precision.

We highly suggest experimenting with different scorers and to study their impact on the resulting models. Using the Diagnostics page in Driverless AI, all applicable scores can be computed for any given model, no matter which scorer was used during training.

57.4 Knob Settings Tips

H2O Driverless AI allows you to customize every experiment in great detail via the expert settings. The most important controls however are the three knobs for accuracy, time and interpretability. A higher accuracy setting results in a better estimate of the model generalization performance, usually through using more data, more holdout sets, more parameter tuning rounds and other advanced techniques. Higher time settings means the experiment is given more time to converge to an optimal solution. Higher interpretability settings reduces the model's complexity through less feature engineering and using simpler models. In general, a setting of 1/1/10 will lead to the simplest and usually least accurate modeling pipeline, while a setting of 10/10/1 will lead to the most complex and most time consuming experiment possible. Generally, it is sufficient to use settings of 7/5/5 or similar, and we recommend to start with the default settings. We highly recommend studying the experiment preview on the left-hand side of the GUI before each experiment - it can help you fine-tune the settings and save time overall.

Note that you can always finish an experiment early, either by clicking 'Finish' to get the deployable final pipeline out, or by clicking 'Abort' to instantly terminate the experiment. In either case, the experiment can be continued seamlessly at a later time with 'Restart from last Checkpoint' or 'Retrain Final Pipeline', and you can always turn the knobs (or modify the expert settings) to adapt to your requirements.

57.5 Tips for Running an Experiment

H2O Driverless AI is an automatic machine learning platform designed to create highly accurate modeling pipelines from tabular training data. The predictive performance of the pipeline is a function of both the training data and the parameters of the pipeline (details of feature engineering and modeling). During an experiment, Driverless AI automatically tunes these parameters by scoring candidate pipelines on held out (“validation”) data. This important validation data is either provided by the user (for experts) or automatically created (random, time-based or fold-based) by Driverless AI. Once a final pipeline has been created, it should be scored on yet another held out dataset (“test data”) to estimate its generalization performance. Understanding the origin of the training, validation and test datasets (“the validation scheme”) is critical for success with machine learning, and we welcome your feedback and suggestions to help us create the right validation schemes for your use cases.

57.6 Expert Settings Tips

H2O Driverless AI offers a range of ‘Expert Settings’ that allow you to customize each experiment. For example, you can limit the amount of feature engineering by reducing the value for ‘Feature engineering effort’ or ‘Max. feature interaction depth’ or by disabling ‘Target Encoding’. You can also select the model types to be used for training on the engineered features (such as XGBoost, LightGBM, GLM, TensorFlow, FTRL, or RuleFit). For time-series problems where the selected time_column leads to an error message (this can currently happen if the time structure is not regular enough - we are working on an improved version), you can disable the ‘Time-series lag-based recipe’ and Driverless AI will create train/validation splits based on the time order instead, which can increase the model’s performance if the time column is important.

57.7 Checkpointing Tips

Driverless AI provides the option to checkpoint experiments to speed up feature engineering and model tuning when running multiple experiments on the same dataset. By default, H2O Driverless AI automatically scans all prior experiments (including aborted ones) for an optimal checkpoint to restart from. You can select a specific prior experiment to restart a new experiment from with “Restart from Last Checkpoint” in the experiment listing page (click on the 3 yellow bars on the right). You can disable checkpointing by setting ‘Feature Brain Level’ in the expert settings (or feature_brain_level in the configuration file) to 0 to force the experiment to start from scratch.

57.8 Text Data Tips

For datasets that contain text (string) columns - where each value can be a few words, a paragraph or an entire document - Driverless AI automatically creates NLP features based on bag of words, tf-idf, singular value decomposition and out-of-fold likelihood estimates. In versions 1.3 and above, you can enable TensorFlow in the expert settings to see how CNN (convolutional neural net) based learned word embeddings can improve predictive accuracy even more. Try this for sentiment analysis, document classification, and generic text-enriched datasets.

APPENDIX A: CUSTOM RECIPES

This appendix describes how to use custom recipes in Driverless AI. You're welcome to create your own recipes, or you can select from a number of recipes available in the [driverlessai-recipes repository](#).

In most cases (especially for complex recipes), MOJOs won't be available out of the box. But, it is possible to get the MOJO. Contact support@h2o.ai for more information about creating MOJOs for custom recipes. (Note that the Python Scoring Pipeline features full support for custom recipes.)

58.1 Additional Resources

- [Custom Recipes FAQ](#): For answers to common questions about custom recipes.
- [How to Write a Recipe](#): A guide for writing your own recipes.
- [Data Template](#): A template for creating your own Data recipe.
- [Model Template](#): A template for creating your own Model recipe.
- [Scorer Template](#): A template for creating your own Scorer recipe.
- [Transformer Template](#): A template for creating your own Transformer recipe.

58.2 Examples

58.2.1 Adding a Data Recipe

Driverless AI allows you to create a new dataset by modifying an existing dataset with a data recipe. (Refer to the `modify_by_recipe` section for more information.) This example shows you how to use the **Live Code** option to create a new dataset by adding a data recipe.

1. Navigate to the Datasets page, then click on the dataset you want to modify.
2. Click **Details** from the submenu that appears to open the Dataset Details page.
3. Click the **Modify by Recipe** button in the top right portion of the UI, then click **Live Code** from the submenu that appears.
4. Enter the code for the data recipe you want to use to modify the dataset. Click the **Get Preview** button to see a preview of how the data recipe will modify the dataset. In this simple example, the data recipe modifies the number of rows and columns in the dataset.
5. Click the **Save** button to confirm the changes and create a new dataset. (The original dataset will still be available on the Datasets page.)

58.2.2 Driverless AI with H2O-3 Algorithms

Driverless AI already supports a variety of *algorithms*. This example shows how you can use our **h2o-3-models-py** recipe to include H2O-3 supervised learning algorithms in your experiment. The available H2O-3 algorithms in the recipe include:

- Naive Bayes
- GBM
- Random Forest
- Deep Learning
- GLM
- AutoML

Caution: Because AutoML is treated as a regular ML algorithm here, the runtime requirements can be large. We recommend that you adjust the `max_runtime_secs` parameters as suggested here: <https://github.com/h2oai/driverlessai-recipes/blob/rel-1.9.0/models/algorithms/h2o-3-models.py#L41>

1. Start an experiment in Driverless AI by selecting your training dataset along with (optionally) validation and testing datasets and then specifying a Target Column. Notice the list of algorithms that will be used in the **Feature evolution** section of the experiment summary. In the example below, the experiment will use LightGBM and XGBoostGBM.
2. Click on **Expert Settings**.
3. Specify the custom recipe using one of the following methods:
 - On your local machine, clone the [driverlessai-recipes repo](#), then use the **Upload Custom Recipe** button to upload the **driverlessai-recipes/models/h2o-3-models.py** file.
 - Click the **Load Custom Recipe from URL** button, then enter the URL for the Python file. (Both HTML and raw versions of the file are supported.)

Note: Click the **Official Recipes (External)** button to browse the driverlessai-recipes repository.

Driverless AI will begin uploading and verifying the new custom recipe.

4. In the Expert Settings page, specify any additional settings and then click **Save**. This returns you to the experiment summary.
5. To include each of the new models in your experiment, return to the Expert Settings option. Click the **Recipes > Include Specific Models** option. Select the algorithm(s) that you want to include. Click **Done** to return to the experiment summary.

Notice the updated list of available algorithms in the experiment.

6. Edit any additional experiment settings, and then click **Launch Experiment**.

Upon completion, you can download the Experiment Summary and review the Model Tuning section of the **report.docx** file to see how each of the algorithms compare.

58.2.3 Using a Custom Scorer

Driverless AI supports a number of scorers, including:

- Regression: GINI, MAE, MAPE, MER, MSE, R2, RMSE (default), RMSLE, RMSPE, SMAPE, TOPDECILE
- Classification: ACCURACY, AUC (default), AUCPR, F05, F1, F2, GINI, LOGLOSS, MACROAUC, MCC

This example shows how you can include a custom scorer in your experiment. This example will use the [Explained Variance scorer](#), which is used for regression experiments.

1. Start an experiment in Driverless AI by selecting your training dataset along with (optionally) validation and testing datasets and then specifying a (regression) Target Column.
2. The scorer defaults to RMSE. Click on **Expert Settings**.
3. Specify the custom scorer recipe using one of the following methods:
 - On your local machine, clone the [driverlessai-recipes repo](#), then use the **Upload Custom Recipe** button to upload the `driverlessai-recipes/scorers/explained_variance.py` file.
 - Click the **Load Custom Recipe from URL** button, then enter the URL for the Python file. (Both HTML and raw versions of the file are supported.)

Note: Click the **Official Recipes (External)** button to browse the driverlessai-recipes repository.

Driverless AI will begin uploading and verifying the new custom recipe.

4. In the Experiment Summary page, select the new Explained Variance (**EXPVAR**) scorer. (**Note:** If you do not see the **EXPVAR** option, return to the Expert Settings, select **Recipes > Include Specific Scorers**, then click the **Enable Custom** button in the top right corner. Click **Done** and then **Save** to return to the Experiment Summary.)
5. Edit any additional experiment settings, and then click **Launch Experiment**. The experiment will run using the custom Explained Variance scorer.

58.2.4 Using a Custom Transformer

Driverless AI supports a number of feature transformers as described in [Driverless AI Transformations](#). This example shows how you can include a custom transformer in your experiment. Specifically, this example will show how to add the ExpandingMean transformer.

1. Start an experiment in Driverless AI by selecting your training dataset along with (optionally) validation and testing datasets and then specifying a Target Column. Notice the list of transformers that will be used in the **Feature engineering search space (where applicable)** section of the experiment summary. Driverless AI determines this list based on the dataset and experiment.
2. Click on **Expert Settings**.
3. Specify the custom recipe using one of the following methods:
 - On your local machine, clone the [driverlessai-recipes repo](#), then use the **Upload Custom Recipe** button to upload the `driverlessai-recipes/transformers/targetencoding/ExpandingMean.py` file.
 - Click the **Load Custom Recipe from URL** button, then enter the URL for the Python file. (Both HTML and raw versions of the file are supported.)

Note: Click the **Official Recipes (External)** button to browse the driverlessai-recipes repository.

Driverless AI will begin uploading and verifying the new custom recipe.

4. Navigate to the **Expert Settings > Recipes** tab and click the **Include Specific Transformers** button. Notice that all transformers are selected by default, including the new ExpandingMean transformer (bottom of page).
 5. Select the transformers that you want to include in the experiment. Use the **Check All/Uncheck All** button to quickly add or remove all transformers at once. This example removes all transformers except for OriginalTransformer and ExpandingMean.
- Note:** If you uncheck all transformers so that none is selected, Driverless AI will ignore this and will use the default list of transformers for that experiment. This list of transformers will vary for each experiment.
6. Edit any additional experiment settings, and then click **Launch Experiment**. The experiment will run using the custom ExpandingMean transformer.

APPENDIX B: THIRD-PARTY INTEGRATIONS

H2O Driverless AI integrates with a (continuously growing) number of third-party products. Please contact sales@h2o.ai to schedule a discussion with one of our Solution Engineers for more information.

If you are interested in a product not yet listed here, please ask us about it!

59.1 Instance Life-Cycle Management

The following products are able to manage (start and stop) Driverless AI instances themselves:

Name	Notes
BlueData	DAI runs in a BlueData container
Domino	DAI runs in a Domino container
IBM Spectrum Conductor	DAI runs in user mode via TAR SH distribution
IBM Cloud Private (ICP)	Uses Kubernetes underneath; DAI runs in a docker container; requires HELM chart
Kubernetes	DAI runs in as a long running service via Docker container
Kubeflow	Abstraction of Kubernetes; allows additional monitoring and management of Kubernetes deployments. Click here for more information .
Puddle (from H2O.ai)	Multi-tenant orchestration platform for DAI instances (not a third party, but listed here for completeness)
SageMaker	Bring your own algorithm docker container

59.2 API Clients

The following products have Driverless AI client API integrations:

Name	Notes
Alteryx	Allows users to interact with a remote DAI server from Alteryx Designer
Cinchy	Data collaboration for the Enterprise, use MOJOS to enrich data and use Cinchy data network to train models
Jupyter/Python	DAI Python API client library can be downloaded from the Web UI of a running instance
KDB	Use KDB as a data source in Driverless AI for training
RStudio/R	DAI R API client library can be downloaded from the Web UI of a running instance. More information available here .

59.3 Scoring

The following products have Driverless AI scoring integrations:

Name	Notes
KDB	Call a MOJO to score streaming data from KDB Ticker Service
ParallelM	Deploy and monitor MOJO models
Qlik	Call a MOJO from a Qlik dashboard
SageMaker	Host scoring-only docker image that uses a MOJO
Trifacta	Call a MOJO as a UDF
UiPath	Call a MOJO from within an RPA workflow

59.4 Storage

Name	Notes
Network Appliance	A mounted expandable volume is convenient for the Driverless AI working (tmp) directory

59.5 Data Sources

Please visit the section on [Enabling Data Connectors](#) for information about data sources supported by Driverless AI.

**CHAPTER
SIXTY**

REFERENCES

- Adebayo, Julius A. "Fairml: Toolbox for diagnosing bias in predictive modeling." Master's Thesis, MIT, 2016.
- Breiman, Leo. "Statistical Modeling: The Two Cultures (with comments and a rejoinder by the author)." *Statistical Science* 16, no. 3, 2001.
- Craven, Mark W. and Shavlik, Jude W. "Extracting tree structured representations of trained networks." *Advances in Neural Information Processing Systems*, 1996.
- Goldstein, Alex, Kapelner, Adam, Bleich, Justin, and Pitkin, Emil. "Peeking inside the black box: Visualizing statistical learning with plots of individual conditional expectation." *Journal of Computational and Graphical Statistics*, no. 24, 2015.
- Groeneveld, R.A. and Meeden, G. (1984), "Measuring Skewness and Kurtosis." *The Statistician*, 33, 391-399.
- Hall, Patrick, Wen Phan, and SriSatish Ambati. "Ideas for Interpreting Machine Learning." O'Reilly Ideas. O'Reilly Media, 2017.
- Hartigan, J. A. and Mohanty, S. (1992), "The RUNT test for multimodality," *Journal of Classification*, 9, 63–70.
- Hastie, Trevor, Tibshirani, Robert, and Friedman, Jerome. *The Elements of Statistical Learning*. Springer, 2008.
- Lei, Jing, Max G'Sell, Alessandro Rinaldo, Ryan J. Tibshirani, and Larry Wasserman. "Distribution-Free Predictive Inference for Regression." *Journal of the American Statistical Association* (just-accepted), 2017.
- Ribeiro, Marco Tulio, Sameer Singh, and Carlos Guestrin. "Why Should I Trust You?: Explaining the Predictions of Any Classifier." In Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2016.
- Tukey, J. W. (1977). *Exploratory Data Analysis*. Addison-Wesley, Reading, MA.
- Wilkinson, L. (1999). "Dot plots." *The American Statistician*, 53, 276–281.
- Wilkinson, L., Anand, A., and Grossman, R. (2005), "Graph-theoretic Scagnostics," in Proceedings of the IEEE Information Visualization 2005, pp. 157–164.
- I.K. Yeo and R.A. Johnson, "A new family of power transformations to improve normality or symmetry." *Biometrika*, 87(4), (2000).