

# Scalable Automatic Machine Learning in H2O



Statistical Computing Workshop  
July 2019

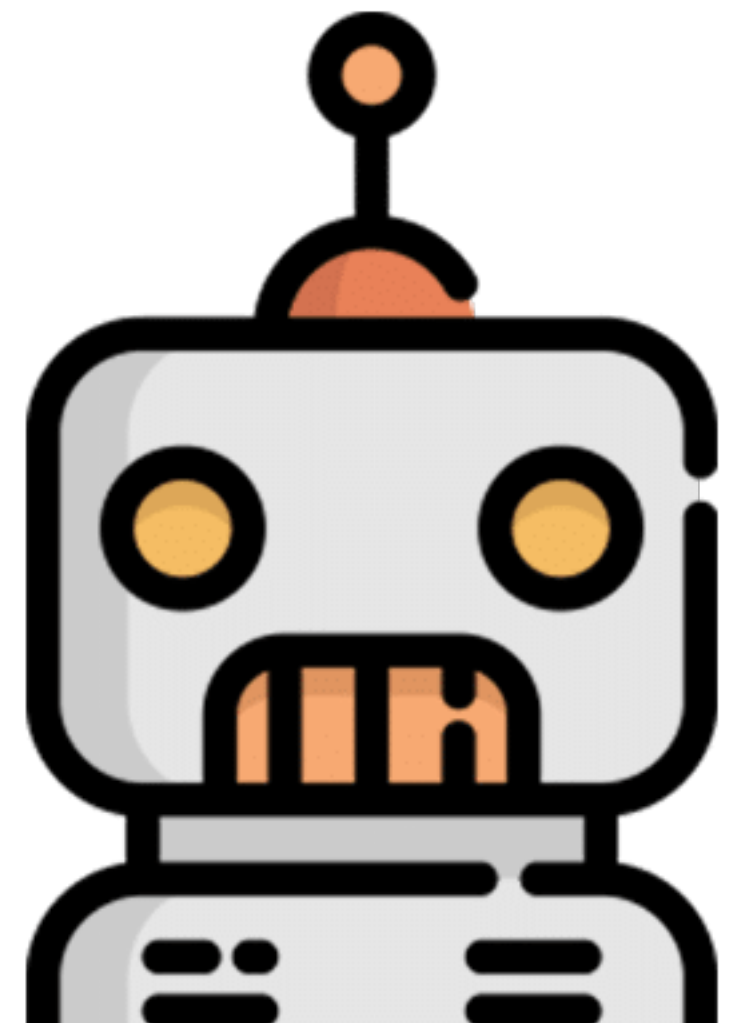
**H<sub>2</sub>O.ai**

Erin LeDell Ph.D.  
@ledell

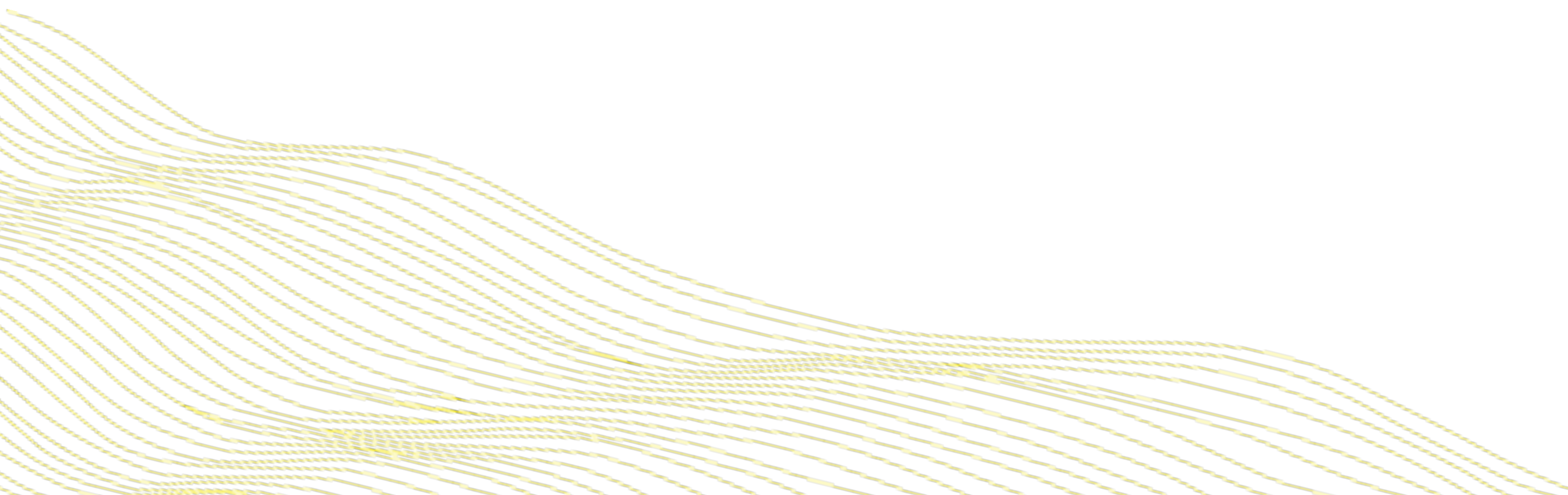
# Agenda

- H2O Platform
- Automatic Machine Learning (AutoML)
- H2O AutoML Overview
- AutoML Pro Tips
- Demo

Slides  <https://tinyurl.com/automl-statcomp>



# H2O Platform





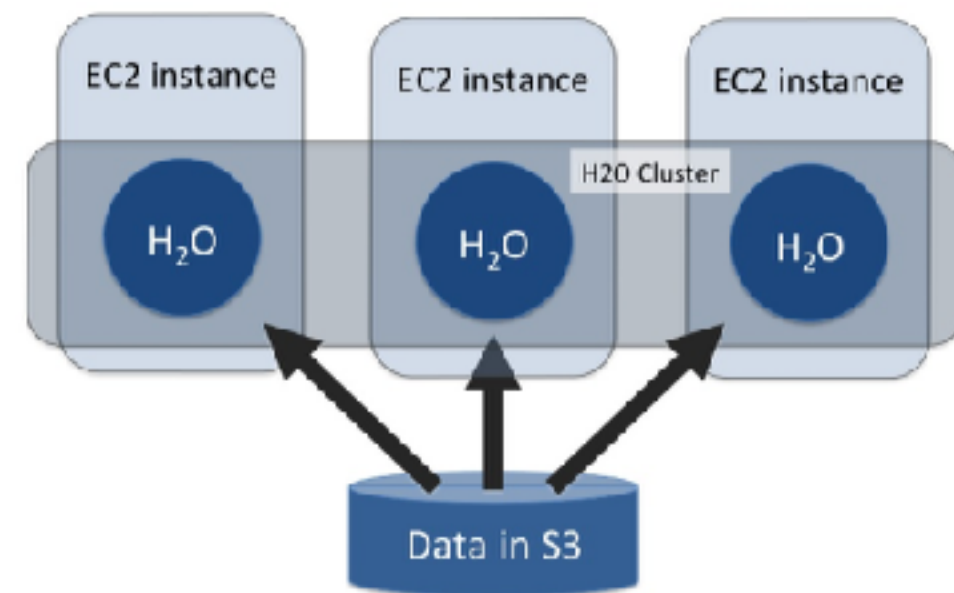
# H2O Machine Learning Platform

- Distributed (multi-core + multi-node) implementations of cutting edge ML algorithms.
- Core algorithms written in high performance Java.
- APIs available in R, Python, Scala; web GUI.
- Easily deploy models to production as pure Java code.
- Works on Hadoop, Spark, EC2, your laptop, etc.



# H2O Distributed Computing

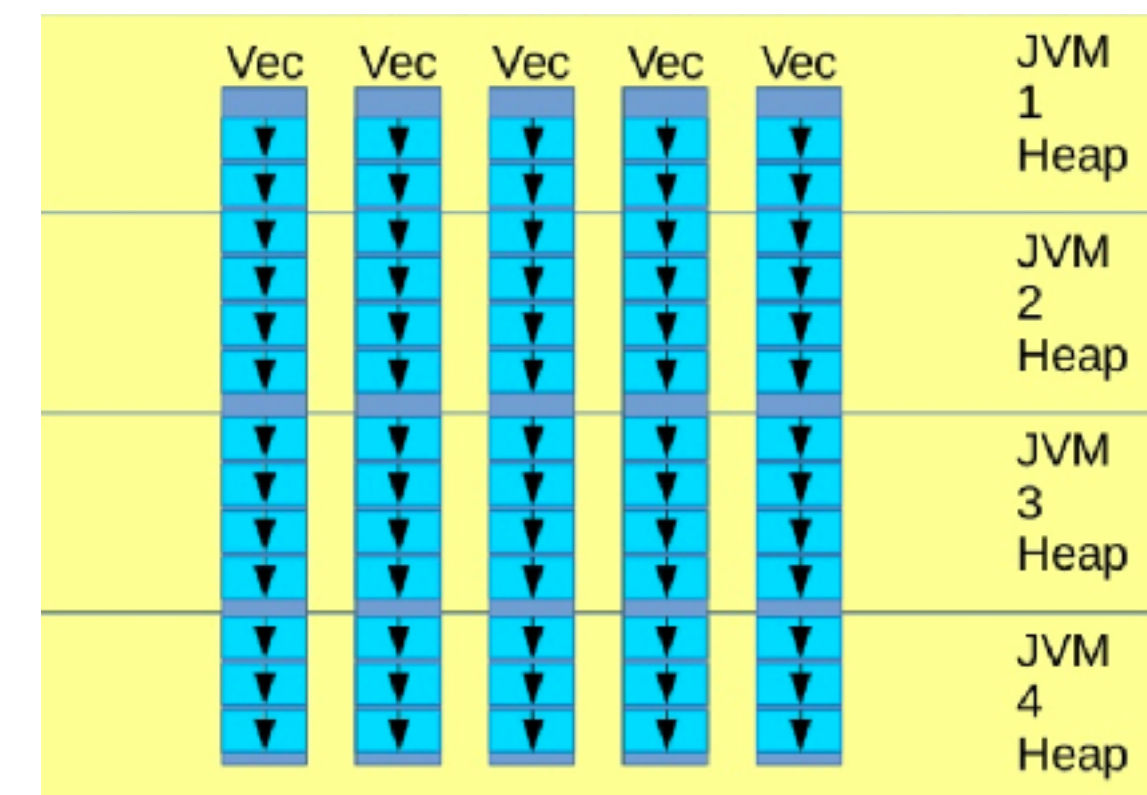
## H2O Cluster



- Multi-node cluster with shared memory model.
- All computations in memory.
- Each node sees only some rows of the data.
- No limit on cluster size.

## H2O Frame

- Distributed data frames (collection of vectors).
- Columns are distributed (across nodes) arrays.
- Works just like R's data.frame or Python Pandas DataFrame

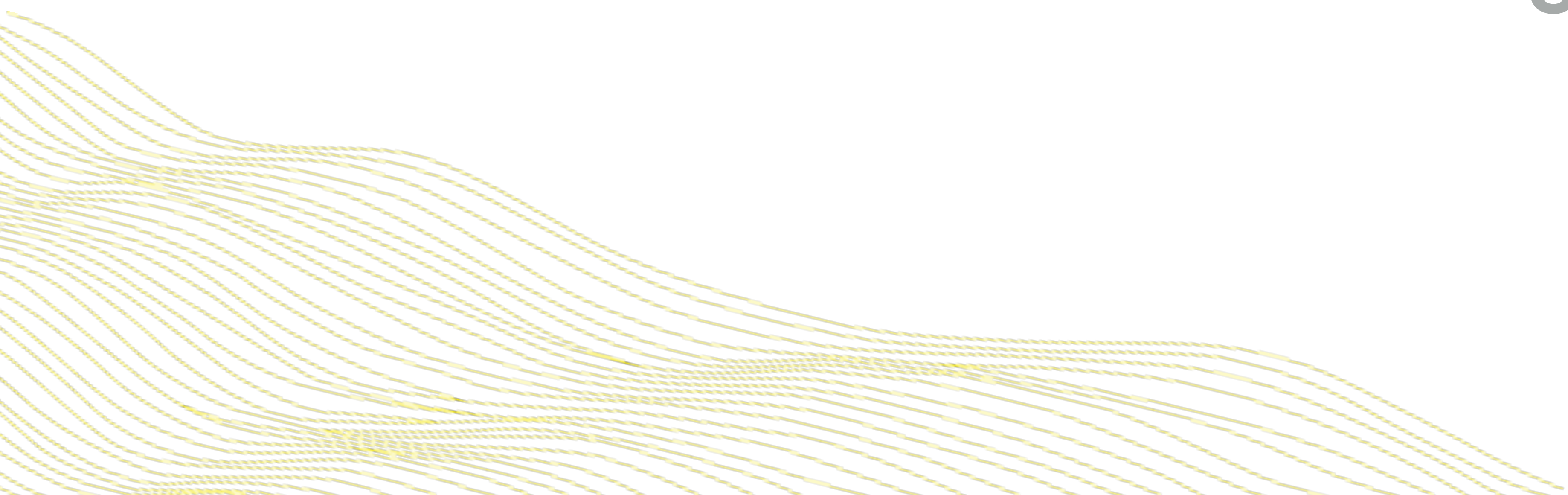


# H2O Machine Learning Features

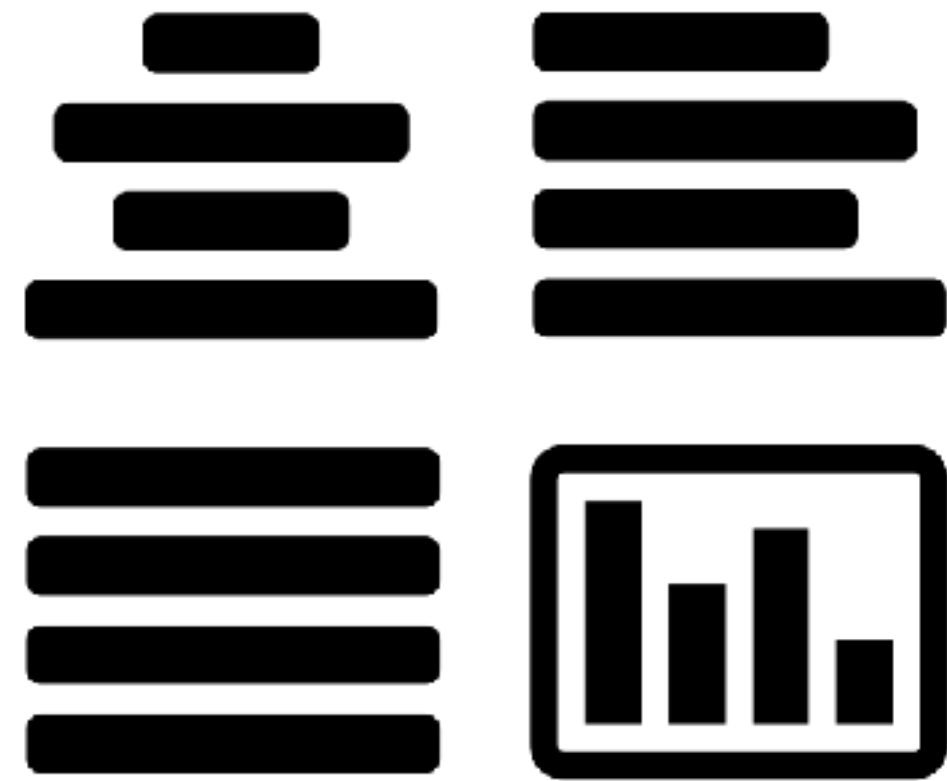


- Supervised & unsupervised machine learning algos (GBM, RF, DNN, GLM, Stacked Ensembles, etc.)
- Imputation, normalization & auto one-hot-encoding
- Automatic early stopping
- Cross-validation, grid search & random search
- Variable importance, model evaluation metrics, plots

# Intro to Automatic Machine Learning

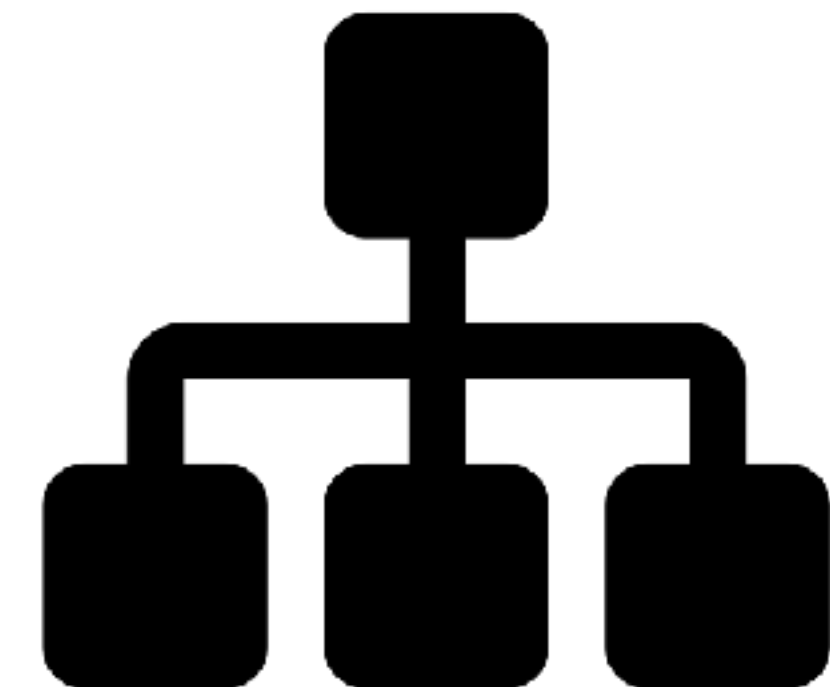
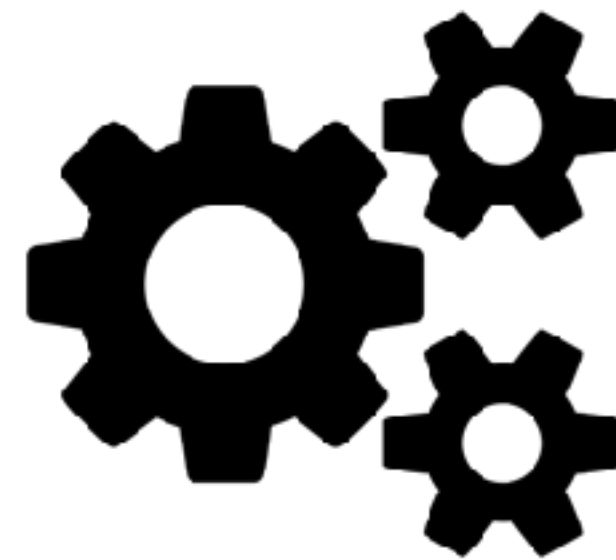


# Aspects of Automatic Machine Learning



Data Prep

Model  
Generation



Ensembles



# Aspects of Automatic Machine Learning

## Data Preprocessing

- Imputation, one-hot encoding, standardization
  - Feature selection and/or feature extraction (e.g. PCA)
  - Count/Label/Target encoding of categorical features
- 

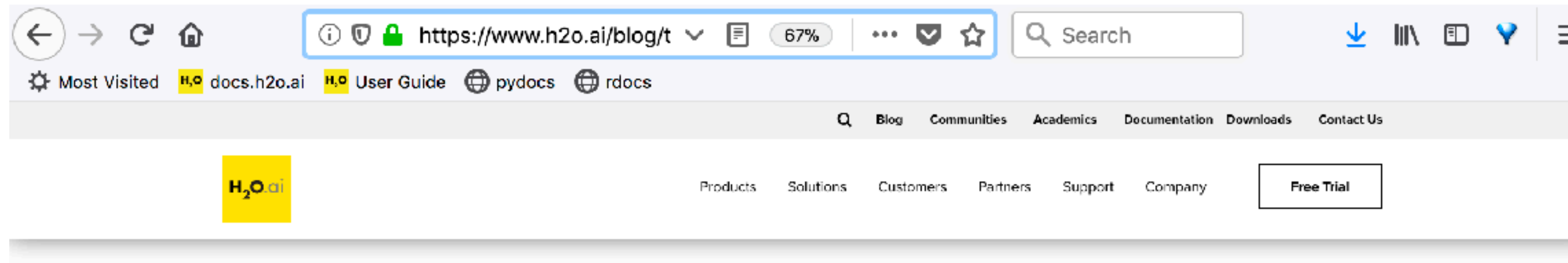
## Model Generation

- Cartesian grid search or random grid search
  - Bayesian Hyperparameter Optimization
  - Individual models can be tuned using a validation set
- 

## Ensembles

- Ensembles often out-perform individual models
- Stacking / Super Learning (Wolpert, Breiman)
- Ensemble Selection (Caruana)

# Different Flavors of AutoML



## The different flavors of AutoML

Share

Category: AutoML, Data Science, Driverless AI, H2O



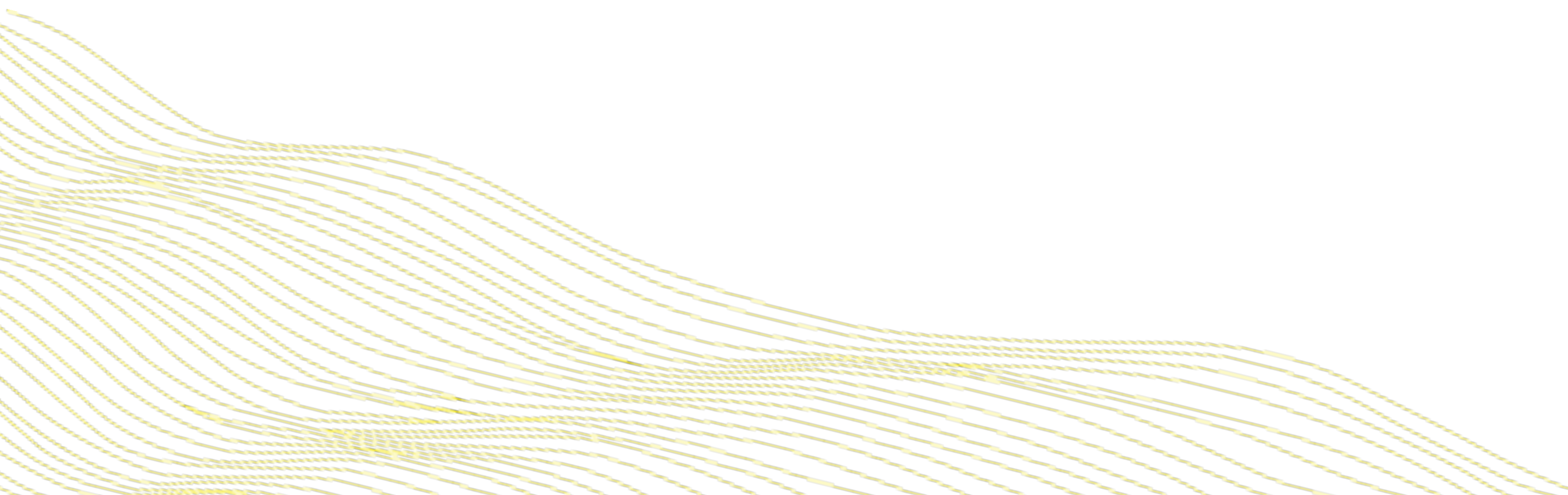
By: Erin LeDell

In recent years, the demand for machine learning experts has outpaced the supply, despite the surge of people entering the field. To address this gap, there have been big strides in the development of user-friendly machine learning software (e.g. [H2O](#), [scikit-learn](#), [keras](#)). Although these tools have made it easy to train and evaluate machine learning models, there is still a good amount of data science knowledge that's required in order to create the *highest-quality* model, given your dataset. Writing the code to perform a hyperparameter search over many different types of algorithms can also be time consuming and repetitive work.

**What is AutoML?**

<https://tinyurl.com/flavors-of-automl>

# H2O's AutoML



# H2O AutoML (v3.24)

## Data Preprocessing

- Imputation, one-hot encoding, standardization
  - Feature selection and/or feature extraction (e.g. PCA)
  - Count/Label/Target encoding of categorical features
- 

## Model Generation

- Cartesian grid search or random grid search
  - Bayesian Hyperparameter Optimization
  - Individual models can be tuned using a validation set
- 

## Ensembles

- Ensembles often out-perform individual models:
- Stacking / Super Learning (Wolpert, Breiman)
- Ensemble Selection (Caruana)



# Random Grid Search & Stacking

- Random Grid Search combined with Stacked Ensembles is a powerful combination.
- Ensembles perform particularly well if the models they are based on (1) are individually strong, and (2) make uncorrelated errors.
- Stacking uses a second-level metalearning algorithm to find the optimal combination of base learners.

# Stacking (aka Super Learner Algorithm)

$$\begin{matrix} & \overbrace{\hspace{1.5cm}}^m & \\ n \left\{ \begin{bmatrix} X \end{bmatrix} \right. & \begin{bmatrix} y \end{bmatrix} \end{matrix} \quad \begin{matrix} \text{"Level-zero"} \\ \text{data} \end{matrix}$$

- Start with design matrix,  $X$ , and response,  $y$
- Specify  $L$  base learners (with model params)
- Specify a metalearner (just another algorithm)
- Perform  $k$ -fold CV on each of the  $L$  learners

# Stacking (aka Super Learner Algorithm)

$$n \left\{ \begin{bmatrix} p_1 \end{bmatrix} \cdots \begin{bmatrix} p_L \end{bmatrix} \begin{bmatrix} y \end{bmatrix} \right\} \rightarrow n \left\{ \overbrace{\begin{bmatrix} Z \end{bmatrix}}^L \begin{bmatrix} y \end{bmatrix} \right\} \quad \text{"Level-one" data}$$

- Collect the predicted values from k-fold CV that was performed on each of the  $L$  base learners
- Column-bind these prediction vectors together to form a new design matrix,  $Z$
- Train the metalearner using  $Z, y$

# H2O AutoML

- Basic data pre-processing (as in all H2O algos).
- Trains a random grid of GBMs, DNNs, GLMs, etc. using a carefully chosen hyper-parameter space.
- Individual models are tuned using cross-validation.
- Two Stacked Ensembles are trained (“All Models” ensemble & a lightweight “Best of Family” ensemble).
- Returns a sorted “Leaderboard” of all models.
- All models can be easily exported to production.





# H2O AutoML in Python

## Example

```
import h2o
from h2o.automl import H2OAutoML
h2o.init()

train = h2o.import_file("train.csv")

aml = H2OAutoML(max_runtime_secs = 600)
aml.train(y = "response_colname",
          training_frame = train)

lb = aml.leaderboard
```

# H2O AutoML in R

## Example

```
library(h2o)
h2o.init()

train <- h2o.importFile("train.csv")

aml <- h2o.automl(y = "response_colname",
                 training_frame = train,
                 max_runtime_secs = 600)

lb <- aml@leaderboard
```





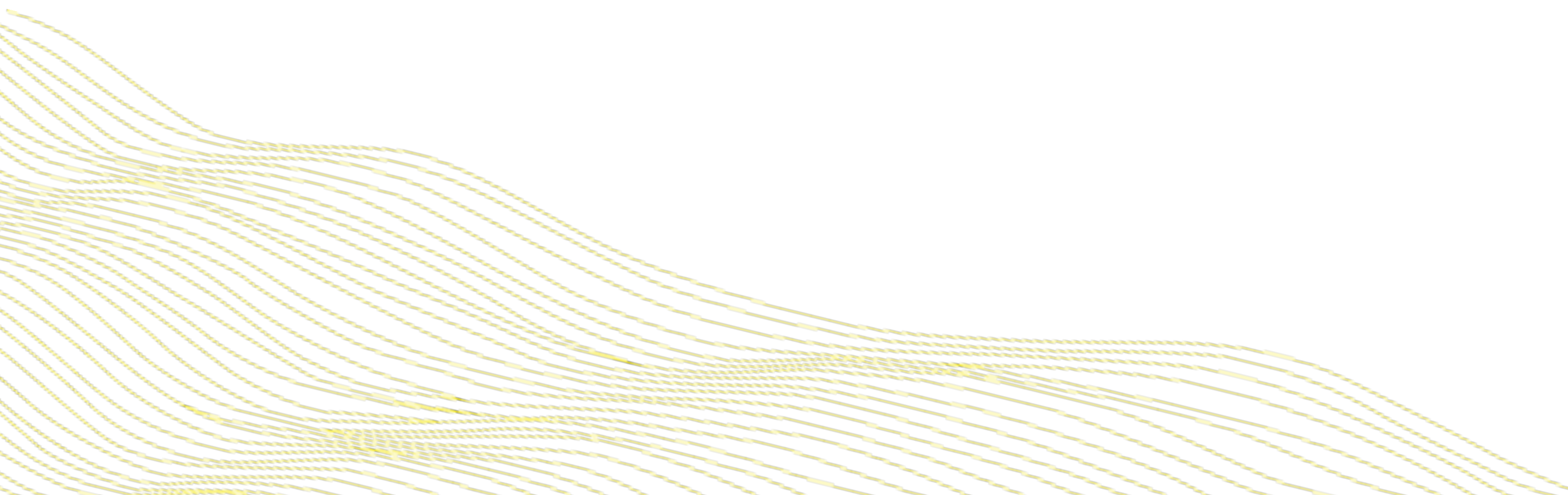
# H2O AutoML Leaderboard

model_id	auc	logloss	mean_per_class_error	rmse	mse
StackedEnsemble_AllModels_AutoML_20181212_105540	0.7898014	0.5511086	0.3331737	0.4321104	0.1867194
StackedEnsemble_BestOfFamily_AutoML_20181212_105540	0.7884246	0.5521454	0.3231919	0.4326254	0.1871647
XGBoost_1_AutoML_20181212_105540	0.7846510	0.5575305	0.3254707	0.4349489	0.1891806
XGBoost_grid_1_AutoML_20181212_105540_model_4	0.7835232	0.5578542	0.3188188	0.4352486	0.1894413
XGBoost_grid_1_AutoML_20181212_105540_model_3	0.7830043	0.5596125	0.3250808	0.4357077	0.1898412
XGBoost_2_AutoML_20181212_105540	0.7813603	0.5588797	0.3470738	0.4359074	0.1900153
XGBoost_3_AutoML_20181212_105540	0.7808475	0.5595886	0.3307386	0.4361295	0.1902090
GBM_5_AutoML_20181212_105540	0.7808366	0.5599029	0.3408479	0.4361915	0.1902630
GBM_2_AutoML_20181212_105540	0.7800361	0.5598060	0.3399258	0.4364149	0.1904580
GBM_1_AutoML_20181212_105540	0.7798274	0.5608570	0.3350957	0.4366159	0.1906335
GBM_3_AutoML_20181212_105540	0.7786685	0.5617903	0.3255378	0.4371886	0.1911339
XGBoost_grid_1_AutoML_20181212_105540_model_2	0.7744105	0.5750165	0.3228112	0.4427003	0.1959836
GBM_4_AutoML_20181212_105540	0.7714260	0.5697120	0.3374203	0.4410703	0.1945430
GBM_grid_1_AutoML_20181212_105540_model_1	0.7697524	0.5725826	0.3443314	0.4424524	0.1957641
GBM_grid_1_AutoML_20181212_105540_model_2	0.7543664	0.9185673	0.3558550	0.4966377	0.2466490
DRF_1_AutoML_20181212_105540	0.7428924	0.5958832	0.3554027	0.4527742	0.2050045
XRT_1_AutoML_20181212_105540	0.7420910	0.5993457	0.3565826	0.4531168	0.2053148
DeepLearning_grid_1_AutoML_20181212_105540_model_2	0.7417952	0.6014974	0.3682910	0.4549035	0.2069372
XGBoost_grid_1_AutoML_20181212_105540_model_1	0.6935538	0.6207021	0.4058805	0.4657911	0.2169614
DeepLearning_1_AutoML_20181212_105540	0.6913704	0.6379538	0.4093513	0.4717801	0.2225765
DeepLearning_grid_1_AutoML_20181212_105540_model_1	0.6900835	0.6617941	0.4184695	0.4766352	0.2271811
GLM_grid_1_AutoML_20181212_105540_model_1	0.6826481	0.6385205	0.3972341	0.4726827	0.2234290

Example  
Leaderboard for  
binary classification



# AutoML Pro Tips!



Before you press the “red button”



# AutoML Pro Tips: Exclude Algos

- If you have sparse, wide data (e.g. text), use the `exclude_algos` argument to turn off the tree-based models (GBM, RF).
- If you want tree-based algos only, turn off GLM and DNNs via `exclude_algos`.

# AutoML Pro Tips: Time & Model Limits

- AutoML will stop after 1 hour unless you change `max_runtime_secs`.
- Running with `max_runtime_secs` is not reproducible since available resources on a machine may change from run to run. Set `max_runtime_secs` to a big number (e.g. 9999999999) and use `max_models` instead.



# AutoML Pro Tips: Cluster memory

- Reminder: All H2O models are stored in H2O Cluster memory.
- Make sure to give the H2O Cluster a lot of memory if you're going to create hundreds or thousands of models.
- e.g. `h2o.init(max_mem_size = "80G")`



After you press the “red button”





# AutoML Pro Tips: Early Stopping

- If you're expecting more models than are listed in the leaderboard, or the run is stopping earlier than `max_runtime_secs`, this is a result of the default "early stopping" settings.
- To allow more time, increase the number of `stopping_rounds` and/or decrease value of `stopping_tolerance`.

# AutoML Pro Tips: Add More Models

- If you want to add (train) more models to an existing AutoML project, just make sure to use the same training set and `project_name`.
- If you set the same seed twice it will give you identical models as the first run (not useful), so change the seed or leave it unset.



# H2O AutoML Roadmap

- Automatic target encoding of high cardinality categorical cols
- Better support for wide datasets via feature selection/extraction
- Support text input directly via Word2Vec
- Variable importance for Stacked Ensembles
- Better exposure of what AutoML does (expose log in R, Python)
- Improvements to the models we train based on benchmarking
- Fully customizable model list (grid space, etc)

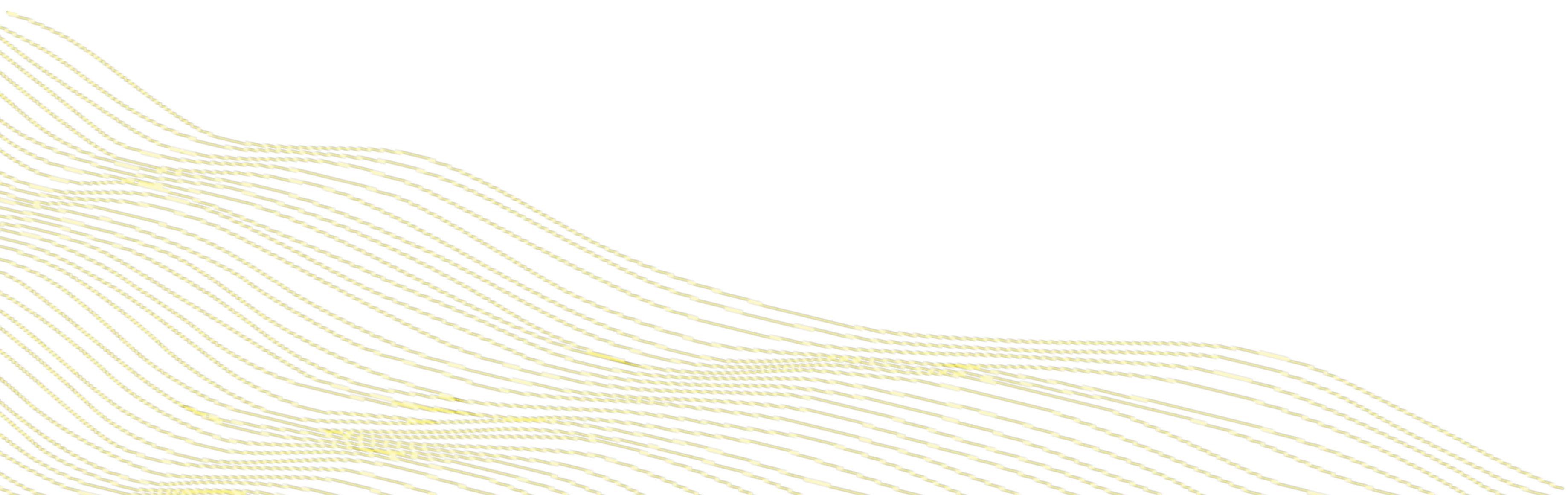


# AutoML Benchmarks

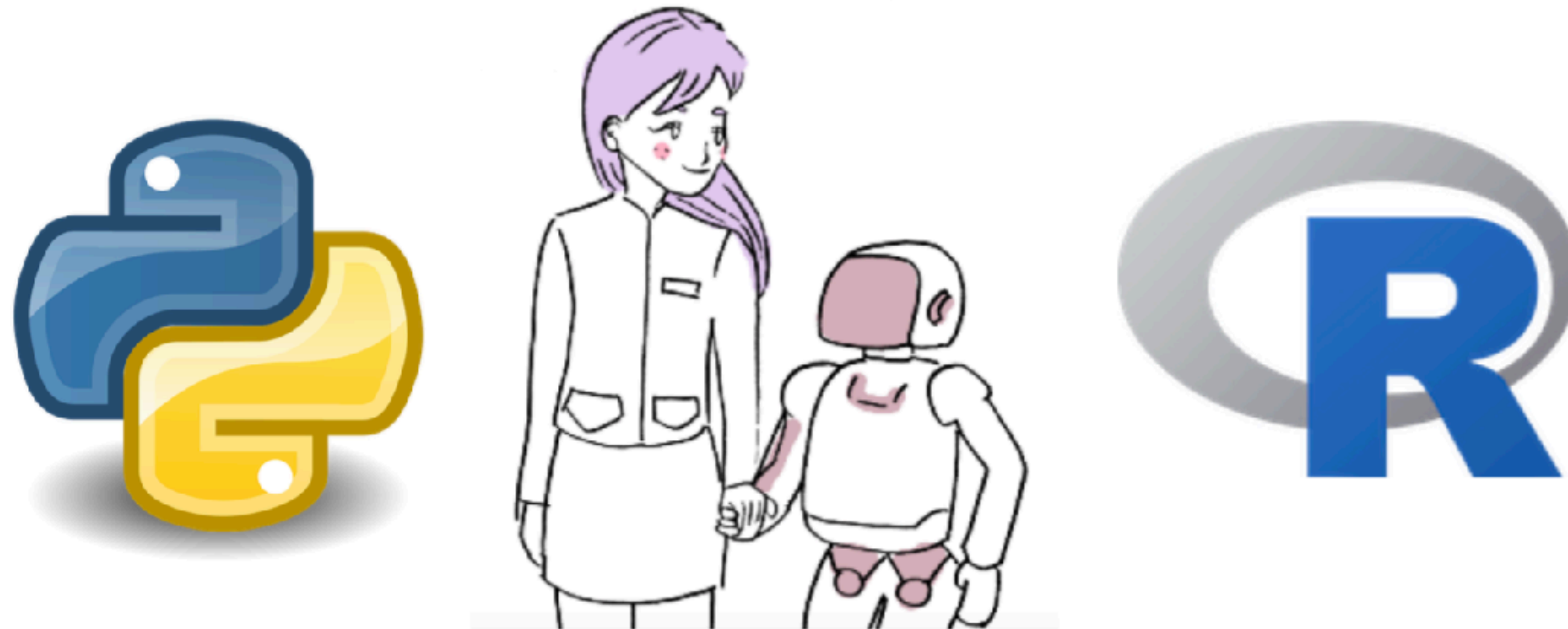
arXiv paper 

<https://tinyurl.com/automlbenchmark>

# H2O AutoML Tutorial



# Learn H2O AutoML!



- Docs: <https://tinyurl.com/h2o-automl-docs>
- R & Py tutorials: <https://tinyurl.com/h2o-automl-tutorials>



# H2O Resources

- Documentation: <http://docs.h2o.ai>
- Tutorials: <https://github.com/h2oai/h2o-tutorials>
- Slidedecks: <https://github.com/h2oai/h2o-meetups>
- Videos: <https://www.youtube.com/user/0xdata>
- Stack Overflow: <https://stackoverflow.com/tags/h2o>
- Google Group: <https://tinyurl.com/h2ostream>
- Gitter: <http://gitter.im/h2oai/h2o-3>
- Events & Meetups: <http://h2o.ai/events>



# Thank you!

@ledell on Github, Twitter  
erin@h2o.ai

