

Productionizing H2O Models using Apache Spark



HUG: Big Data Meetup Prague, September 26, 2019

Jakub Háva

jakub.hava@h2o.ai

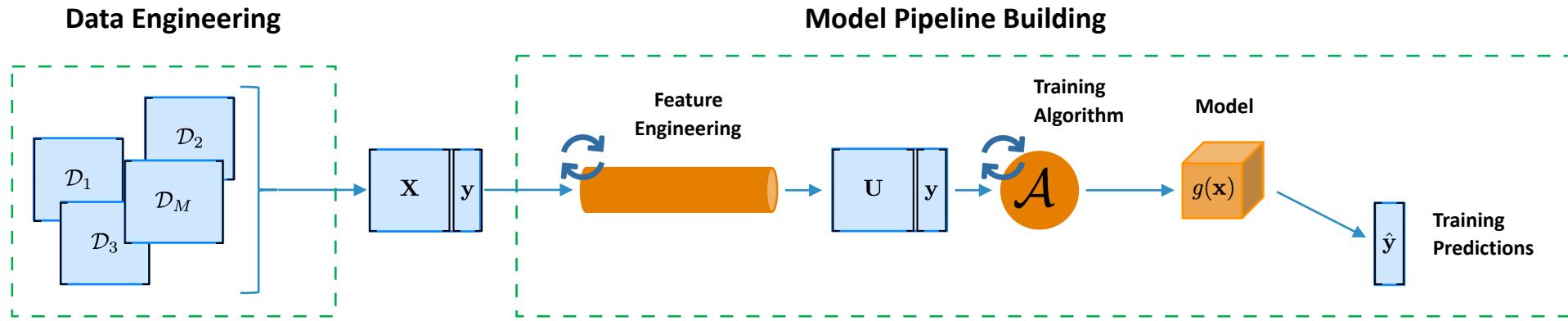
<https://www.linkedin.com/in/havaj/>

Who Are We?

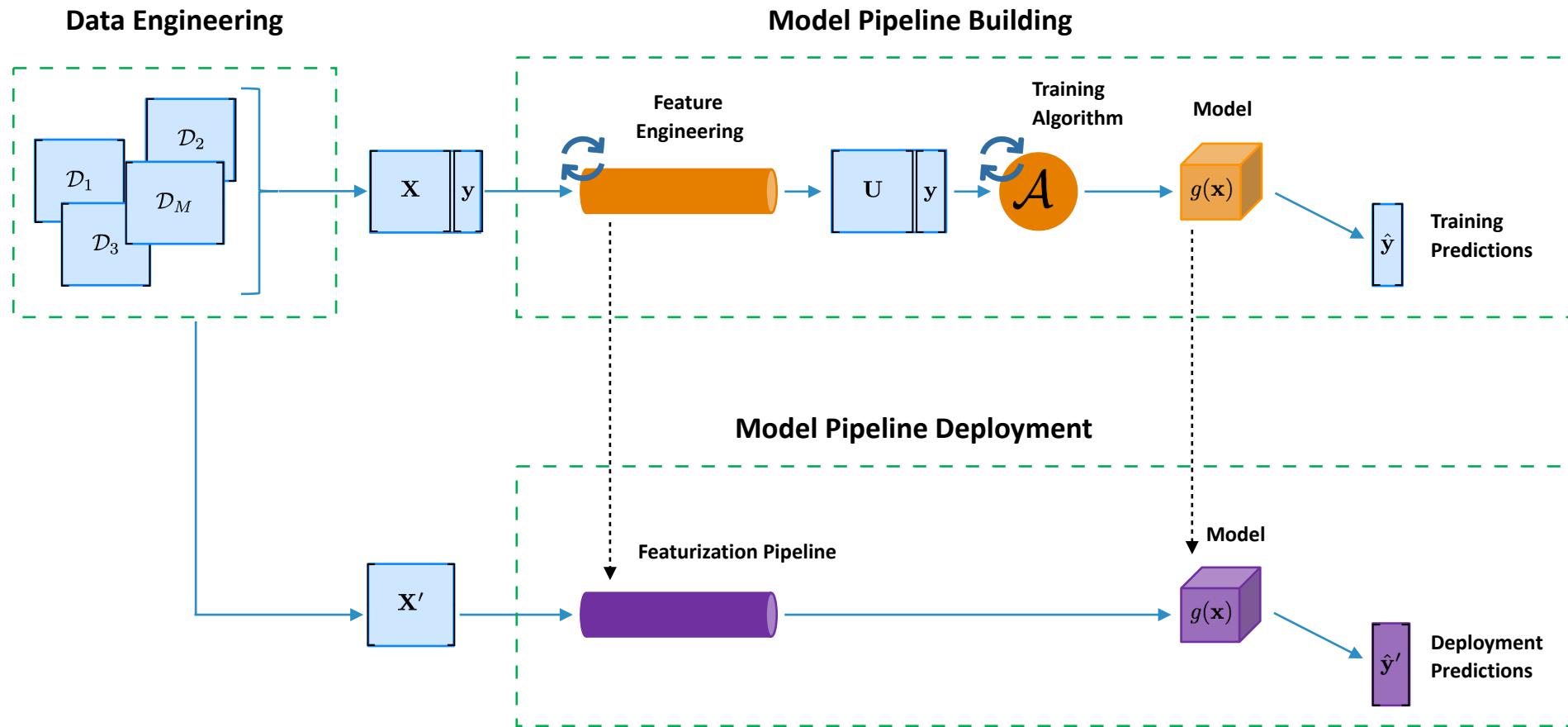
- **Kuba**
 - Senior Software engineer at H2O.ai
- **Marek**
 - Senior Software engineer at H2O.ai
- **Michał**
 - VP of Engineering at H2O.ai

Machine Learning (ML) Lifecycle

Basic ML Lifecycle



Basic ML Lifecycle



Example Implementations

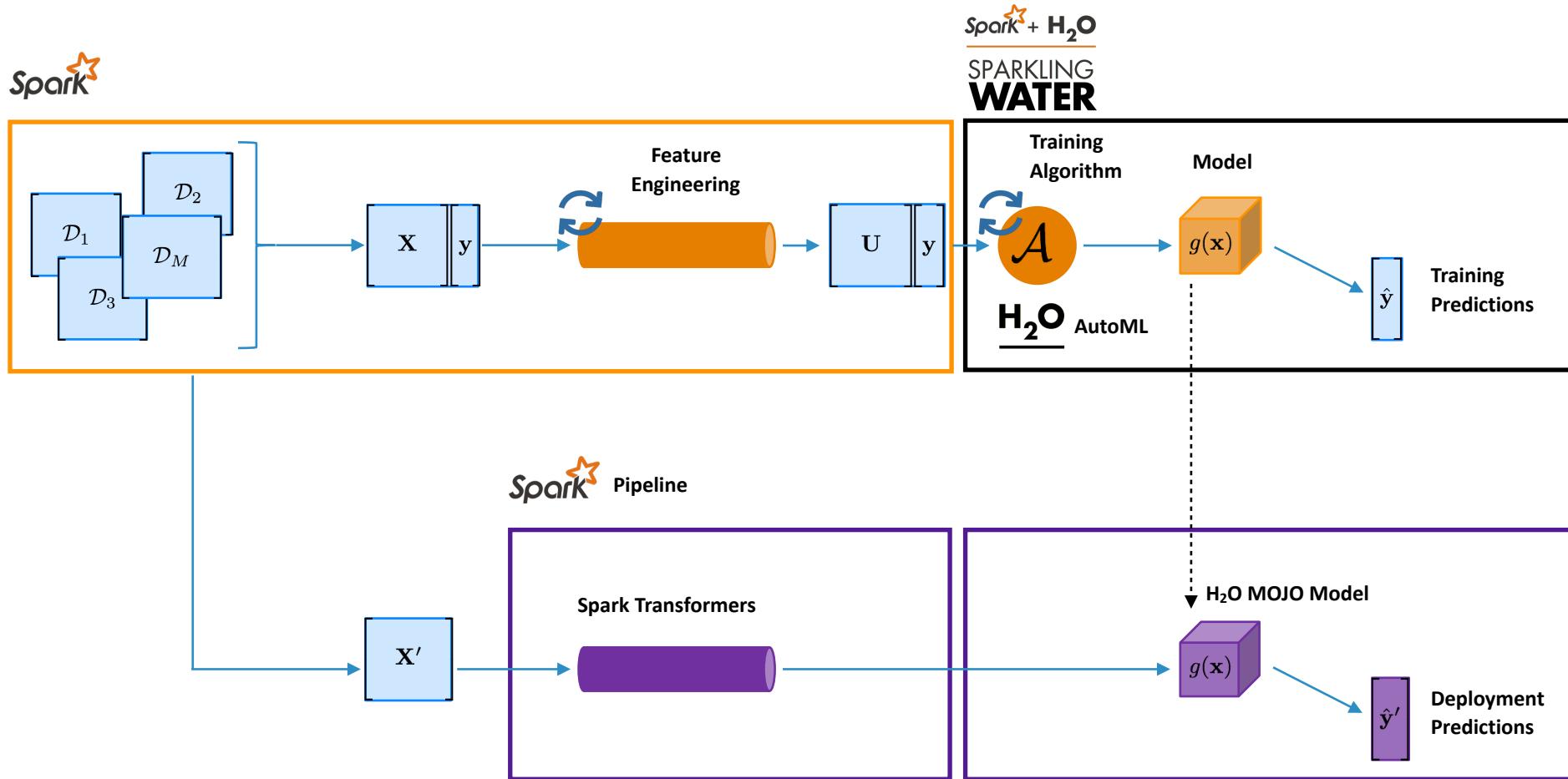
Model Building		Model Deployment		
Data Engineering	Feature Engineering	Training Algorithm	Deployment Pipeline	Model
	Spark	H2O	Spark	H2O MOJO
Spark		H2O Driverless AI	Spark	H2O Driverless AI MOJO

**H₂O + Spark =
Sparkling
Water**

H2O + Spark

- H2O
 - Machine Learning Library
 - Distributed Algorithms
 - For ML experts
- Sparkling Water
 - Integrates H2O & Spark Ecosystems
 - Transparent for Spark users
 - Based on Spark pipelines & H2O

Basic ML Lifecycle: Sparkling Water



Example:

Spark Pipeline

Start Sparkling Water

```
# Necessary PySparkling imports
from pysparkling import *
from pysparkling.ml import *
```

```
# Start H2O Context
hc = H2OContext.getOrCreate(spark)
```

Connecting to H2O server at <http://172.30.43.36:54321>... successful.

Create the Pipeline

```
# Filter out the outliers
from pyspark.ml.feature import SQLTransformer
removeOutliers = SQLTransformer(
    statement="SELECT * FROM __THIS__ WHERE nrows_train > 100")
```

```
# Define the H2O AutoML pipeline stage for Spark pipeline
from pysparkling.ml import *
algoStage = H2OAutoML(
    maxRuntimeSecs=60, # 1 minute
    seed=1,
    convertUnknownCategoricalLevelsToNa=True,
    predictionCol="t_total")
```

```
# Define the Spark pipeline
from pyspark.ml import Pipeline, PipelineModel
pipeline = Pipeline(stages=[removeOutliers, algoStage])
```

Export the Pipeline

```
# Convert data to Spark frame from H2O frame
train_spark_frame = hc.as_spark_frame(fr)
train_spark_frame
```

```
DataFrame[GPU: string, sys_CPU: tinyint, sys_GPU: tinyint, config_accu: tinyint, config_time: tinyint, config_MLI: tinyint, nrows_train: int, ncols_train: smallint, target_binary_dist: double, target_nclasses: smallint, recipe_individuals: tinyint, train_data_size: bigint, weight: tinyint, t_total: double]
```

```
# Train the pipeline model
model = pipeline.fit(train_spark_frame)
```

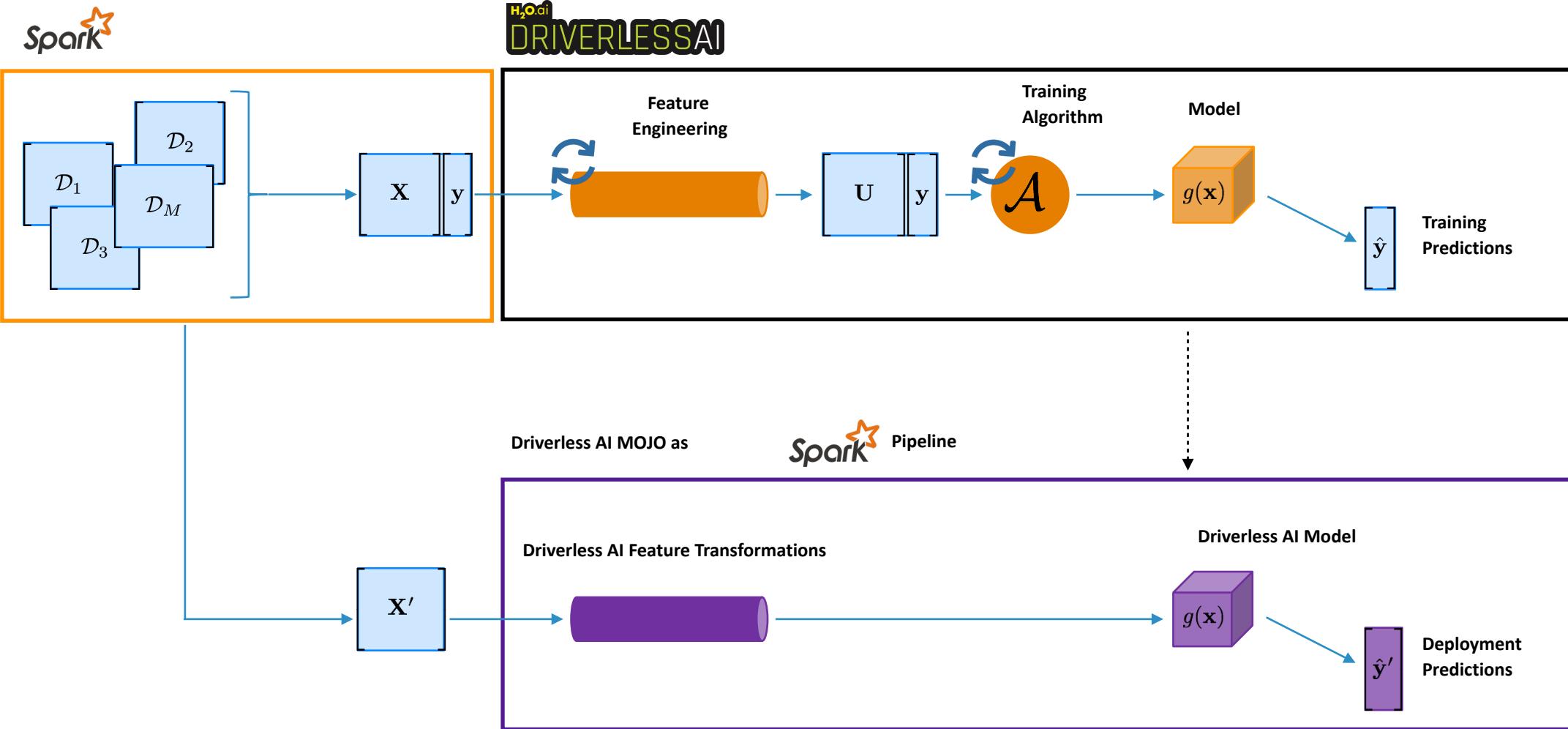
```
# Export the Pipeline Model
model.write().overwrite().save("spark.pipeline")
```

H2O Driverless AI

H2O Driverless AI

- What if I'm not expert ?
 - H2O Driverless AI
 - H2O Driverless AI
 - No expert knowledge required
 - Automatic Feature Engineering & ML

Basic ML Lifecycle: Driverless AI



**Example:
Driverless AI as
Spark Pipeline**

TRAINING DATA

DATASET

train.csv

ROWS

24K

COLUMNS

25

DROPPED COLS

--

VALIDATION DATASET

--

TEST DATASET

--

TARGET COLUMN

default payment next

FOLD COLUMN

--

WEIGHT COLUMN

--

TIME COLUMN

[OFF]

TYPE

int

COUNT

23999

UNIQUE

2

TARGET FREQ

18630

What do these settings mean?

ACCURACY

- Training data size: **4,000 rows, 25 cols** (sampled)
- Feature evolution: **XGBoost, 1/3 validation split, 2 reps**
- Final pipeline: **XGBoost, 4-fold CV**

TIME

- Feature evolution: **8 individuals**, up to **500 iterations**
- Early stopping: After **50** iterations of no improvement

INTERPRETABILITY

- Feature pre-pruning strategy: **None**
- Monotonicity constraints: **disabled**
- Feature engineering search space (where applicable):
['Clustering', 'Date', 'FrequencyEncoding', 'Identity',
'Interactions', 'TargetEncoding', 'Text', 'TruncatedSVD',
'WeightOfEvidence']

XGBoost models to train:

- Feature evolution: **4024**
- Final pipeline: **1**

Estimated max. total memory usage:

- Feature engineering: **8.0MB**
- GPU XGBoost: **1.2GB**

Estimated runtime: **20 minutes**

EXPERIMENT SETTINGS

HELP



ACCURACY



TIME



INTERPRETABILITY

CLASSIFICATION

REPRODUCIBLE

ENABLE GPUS

SCORER

GINI
MCC
F05
F1
F2
ACCURACY
LOGLOSS
AUC
AUCPR

LAUNCH EXPERIMENT

TRAINING DATA

DATASET

Engine_6988_494b344516674125abd1b4673ff39b

ROWS

24K

COLUMNS

25

DROPPED COLS

0

VALIDATION DATASET

--

TEST DATASET

Yes

Engine_6988_9c1a007...

TARGET COLUMN

default payment next

WEIGHT COLUMN

--

FOLD COLUMN

--

TIME COLUMN

[OFF]

TYPE

bool

COUNT

24000

UNIQUE

2

TARGET FREQ

5265

ITERATION DATA - VALIDATION



ASSISTANT

STATUS: COMPLETE

- [INTERPRET THIS MODEL](#)
- [DIAGNOSE MODEL ON NEW DATASET...](#)
- [SCORE ON ANOTHER DATASET](#)
- [TRANSFORM ANOTHER DATASET...](#)
- [DOWNLOAD PREDICTIONS ▾](#)
- [DOWNLOAD PYTHON SCORING PIPELINE](#)
- [BUILD MOJO SCORING PIPELINE](#)
- [DOWNLOAD EXPERIMENT SUMMARY](#)
- [DOWNLOAD LOGS](#)

EXPERIMENT SETTINGS

1

ACCURACY

1

TIME

10

INTERPRETABILITY

CLASSIFICATION

REPRODUCIBLE

ENABLE GPUs

EXPERT SETTINGS

SCORER
GINI
MCC
F05
F1
F2
ACCURACY
LOGLOSS
AUC
AUCPR

VARIABLE IMPORTANCE

4_Freq:PAY_0	1.00
21_PAY_0	0.88
44_Freq:PAY_0:PAY_2	0.68
22_PAY_2	0.41
38_CVTE:PAY_2:0	0.31
12_BILL_AMT1	0.30
23_PAY_3	0.29
19_LIMIT_BAL	0.24
27_PAY_AMT1	0.24
29_PAY_AMT3	0.22
43_CVCatNumEnc:LIMIT_BAL:PAY_AMT5:sd	0.21
28_PAY_AMT2	0.20
34_CVTE:PAY_4:0	0.20
14_BILL_AMT3	0.20

ROC

P-R

LIFT

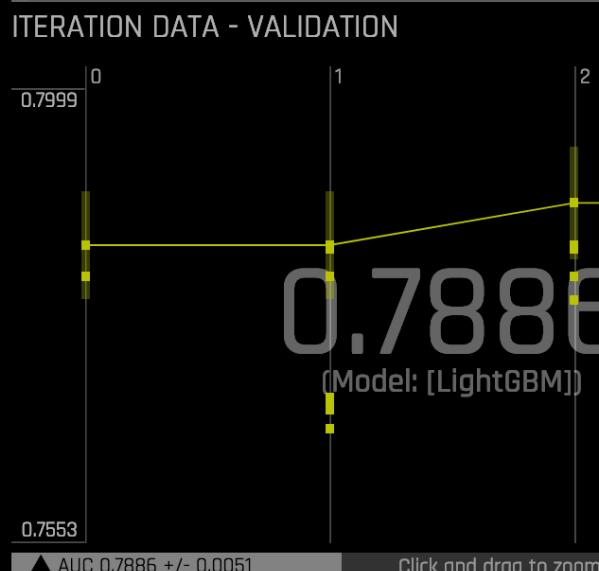
GAINS

K-S

SUMMARY

Experiment: wavuliwu, 2019-02-21 02:24, 1.5.4
Settings: 1/10, seed=265172352, GPUs enabled
Train data:
Engine_6988_494b344516674125abd1b4673ff39b13_daiplugin.csv (24000, 25)
Validation data: N/A
Test data:
Engine_6988_9c1a0076c6d2491b898c5f1a5ab4566a_daiplugin.csv (5700, 24)
Target column: default payment next month (binary, 21.938% target class)
System specs: Docker/Linux, 240 GB, 32 CPU cores, 4/4 GPUs
Max memory usage: 1.52 GB, 1.49 GB GPU
Recipe: AutoDL (2 iterations, 8 individuals)
Validation scheme: stratified, 1 internal holdout
Feature engineering: 87 features scored (32 selected)
Timing:
Data preparation: 2.70 secs
Model and feature tuning: 5.42 secs (4 models trained)
Feature evolution: 15.43 secs (11 of 18 models trained)
Final pipeline training: 7.05 secs (1 model trained)
Python / MOJO scorer building: 11.53 secs / 0.00 secs
Validation score: AUC = 0.78451 +/- 0.0051346 (baseline)
Validation score: AUC = 0.78865 +/- 0.0050898 (final pipeline)

TRAINING DATA				ASSISTANT	
DATASET		TEST DATASET			
ROWS	COLUMNS	DROPPED COLS	VALIDATION DATASET	TEST DATASET	
24K	25	0	--	Yes	Engine_6988_9c1a0076c6d2491b898c5f1a5ab4566a_daiplugin.csv
TARGET COLUMN	default payment next	FOLD COLUMN	--		
WEIGHT COLUMN	--	TIME COLUMN	[OFF]		
TYPE	COUNT	UNIQUE	TARGET FREQ		
bool	24000	2	5265		



- STATUS: COMPLETE
- [DEPLOY](#)
 - [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](#)
 - [DOWNLOAD EXPERIMENT SUMMARY](#)
 - [DOWNLOAD LOGS](#)

EXPERIMENT SETTINGS		EXPERT SETTINGS	
ACCURACY	1	TIME	1
CLASSIFICATION	REPRODUCIBLE	INTERPRETABILITY	10

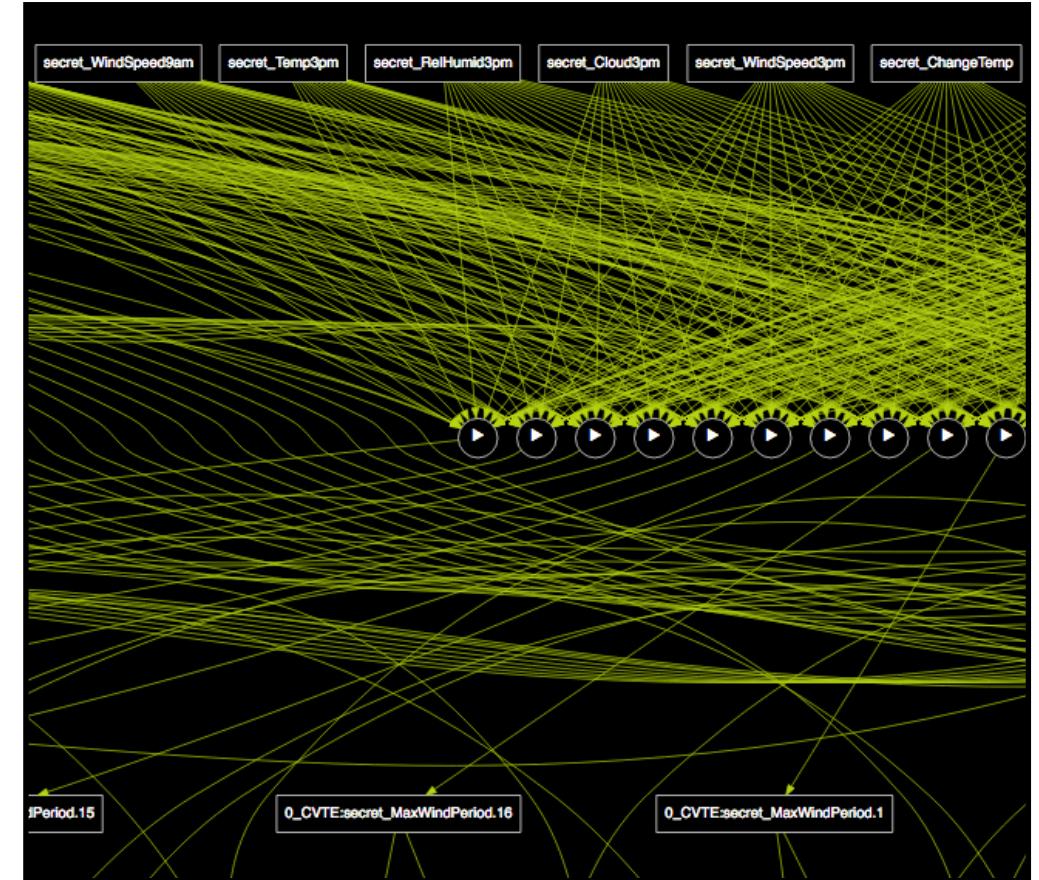
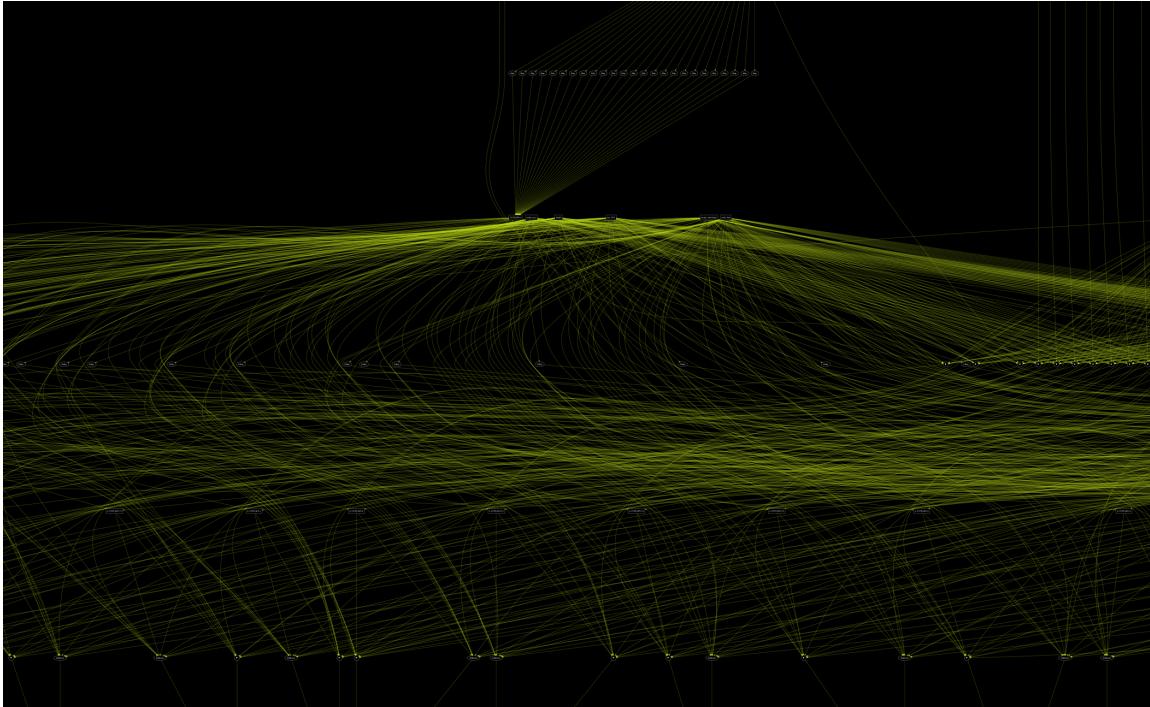
CPU / MEMORY Notifications Log Trace

CPU

MEM

ROC	P-R	LIFT	GAINS	K-S	SUMMARY
Experiment: wavuliwu, 2019-02-21 02:24, 1.5.4					
Settings: 1/1/10, seed=265172352, GPUs enabled					
Train data:					
Engine_6988_494b344516674125abd1b4673ff39b13_daiplugin.csv (24000, 25)					
Validation data: N/A					
Test data:					
Engine_6988_9c1a0076c6d2491b898c5f1a5ab4566a_daiplugin.csv (5700, 24)					
Target column: default payment next month (binary, 21.938% target class)					
System specs: Docker/Linux, 240 GB, 32 CPU cores, 4/4 GPUs					
Max memory usage: 1.52 GB, 1.49 GB GPU					
Recipe: AutoDL (2 Iterations, 8 individuals)					
Validation scheme: stratified, 1 internal holdout					
Feature engineering: 87 features scored (32 selected)					
Timing:					
Data preparation: 2.70 secs					
Model and feature tuning: 5.42 secs (4 models trained)					
Feature evolution: 15.43 secs (11 of 18 models trained)					
Final pipeline training: 7.05 secs (1 model trained)					
Python / MOJO scorer building: 11.53 secs / 0.00 secs					
Validation score: AUC = 0.78451 +/- 0.0051346 (baseline)					
Validation score: AUC = 0.78865 +/- 0.0050898 (final pipeline)					

Driverless AI Pipeline



Example:

Deployment

Load The MOJOs

```
spark = SparkSession.builder.getOrCreate()
spark.sparkContext.setLogLevel("OFF")

# Add sparkling water to all spark executors
Initializer.load_sparkling_jar(spark)

mojo1 = PipelineModel.load(sys.argv[2])
print("Spark Pipeline Loaded")

mojo2 = H2OMOJOPipelineModel.create_from_mojo(sys.argv[1])
print("Driverless AI as Spark Pipeline Loaded")
```

Define Schema

```
schema = StructType([
    StructField("ID", StringType(), True),
    StructField("GPU", StringType(), True),
    StructField("sys_CPU", DoubleType(), True),
    StructField("sys_GPU", DoubleType(), True),
    StructField("config_accu", DoubleType(), True),
    StructField("config_time", DoubleType(), True),
    StructField("config_MLI", DoubleType(), True),
    StructField("nrows_train", DoubleType(), True),
    StructField("ncols_train", DoubleType(), True),
    StructField("target_binary_dist", DoubleType(), True),
    StructField("target_nclasses", DoubleType(), True),
    StructField("recipe_individuals", DoubleType(), True),
    StructField("train_data_size", DoubleType(), True),
    StructField("weight", IntegerType(), True)])
```

Define Input Stream

```
input_data_stream = spark.readStream.schema(schema).csv("s3a://h2o-world-sf-streaming-data/*.csv")
```

Run the Transformations

```
input_data_stream = spark.readStream.schema(schema).csv("s3a://h2o-world-sf-streaming-data/*.csv")
```

```
output_data_stream_1 = mojo1.transform(input_data_stream)
output_data_stream_2 = mojo2.transform(input_data_stream)
```

Store the Result

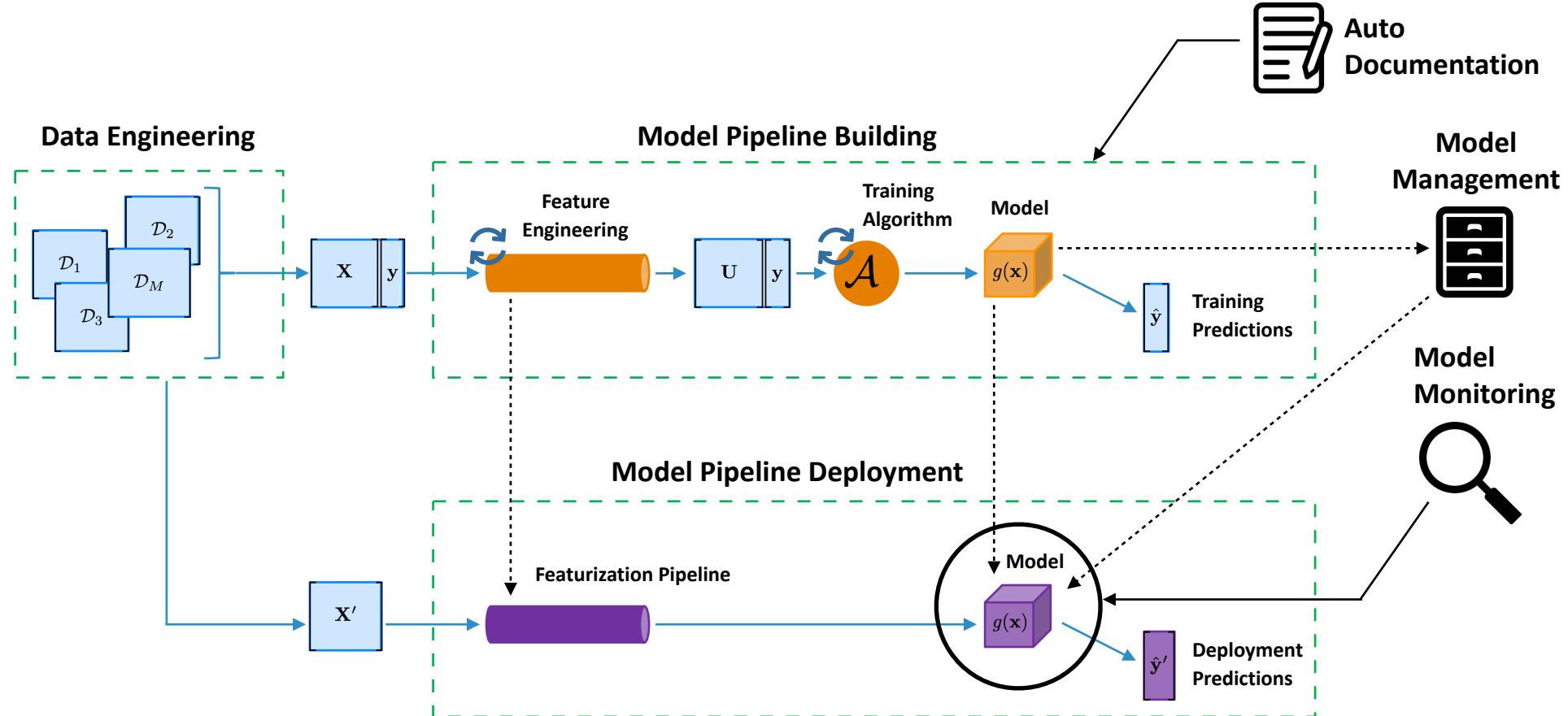
```
input_data_stream = spark.readStream.schema(schema).csv("s3a://h2o-world-sf-streaming-data/*.csv")
```

```
output_data_stream_1 = mojo1.transform(input_data_stream)
output_data_stream_2 = mojo2.transform(input_data_stream)
```

```
output_data_stream_1.writeStream.format("memory").queryName("spark").start()
output_data_stream_2.writeStream.format("memory").queryName("dai").start()
```

Governed ML Lifecycle

Governed ML Lifecycle



Materials



<https://bit.ly/2RCStWM>

Thank you!

**Sparkling
Water enables
deployment of
H2O ML
models with
Spark
Pipelines**

