

---

# Using Driverless AI

*Release 1.0.24*

**H2O.ai**

**Mar 09, 2018**



# RELEASE NOTES

<b>1</b>	<b>H2O Driverless AI Release Notes</b>	<b>3</b>
1.1	Architecture . . . . .	4
1.2	Roadmap . . . . .	5
1.3	ChangeLog . . . . .	5
<b>2</b>	<b>Installing Driverless AI</b>	<b>11</b>
2.1	Quick-Start Tables by Environment . . . . .	11
2.2	Install on NVIDIA GPU/DGX . . . . .	13
2.3	Install on AWS . . . . .	15
2.4	Install on Azure . . . . .	20
2.5	Install on Google Compute with GPUs . . . . .	25
2.6	Install on Google Compute with CPUs . . . . .	30
2.7	Install on Ubuntu with GPUs . . . . .	35
2.8	Install on Ubuntu with CPUs . . . . .	36
2.9	Install on RHEL with GPUs . . . . .	38
2.10	Install on RHEL with CPUs . . . . .	39
2.11	Install on Mac OS X . . . . .	40
2.12	Install on Windows 10 Pro . . . . .	43
<b>3</b>	<b>Upgrading Driverless AI</b>	<b>47</b>
<b>4</b>	<b>Running an Experiment</b>	<b>49</b>
4.1	Experiment Settings . . . . .	54
<b>5</b>	<b>Interpreting a Model</b>	<b>61</b>
5.1	Interpret this Model Button . . . . .	61
5.2	Model Interpretation on Driverless AI Models . . . . .	62
5.3	Model Interpretation on External Models . . . . .	63
5.4	The Model Interpretation Page . . . . .	64
5.5	K-LIME . . . . .	65
5.6	Global and Local Variable Importance . . . . .	67
5.7	Decision Tree Surrogate Model . . . . .	69
5.8	Partial Dependence and Individual Conditional Expectation (ICE) . . . . .	69
5.9	General Considerations . . . . .	71
<b>6</b>	<b>Viewing Explanations</b>	<b>73</b>
<b>7</b>	<b>Score on Another Dataset</b>	<b>77</b>
<b>8</b>	<b>Transform Another Dataset</b>	<b>79</b>

<b>9</b>	<b>The Scoring Pipelines</b>	<b>81</b>
9.1	Driverless AI Standalone Scoring Pipeline . . . . .	81
9.2	Driverless AI MLI Standalone Scoring Package . . . . .	88
<b>10</b>	<b>Viewing Experiments</b>	<b>93</b>
10.1	Rerunning Experiments . . . . .	93
10.2	Deleting Experiments . . . . .	94
<b>11</b>	<b>Visualizing Datasets</b>	<b>95</b>
<b>12</b>	<b>Launching H2O Flow</b>	<b>99</b>
<b>13</b>	<b>Data Connectors</b>	<b>101</b>
13.1	HDFS Setup . . . . .	101
13.2	S3 Setup . . . . .	103
<b>14</b>	<b>The Config.toml File</b>	<b>105</b>
14.1	Sample config.toml File . . . . .	105
<b>15</b>	<b>FAQ</b>	<b>109</b>
<b>16</b>	<b>Appendix A: Driverless AI Transformations</b>	<b>113</b>
16.1	Example Transformations . . . . .	114
<b>17</b>	<b>Appendix B: Using the Driverless AI Python Client</b>	<b>119</b>
17.1	Running an Experiment . . . . .	119
17.2	Access an Experiment Object that was Run through the Web UI . . . . .	123
17.3	Score on New Data . . . . .	124
17.4	Run Model Interpretation . . . . .	125
<b>18</b>	<b>References</b>	<b>129</b>

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.

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 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, please see <https://www.h2o.ai/driverless-ai/>.

For a third-party review, please 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>.



## H2O DRIVERLESS AI RELEASE NOTES

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 plain text sources and automates data visualization, feature engineering, model training, and model explanation. 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. The ability to read data from HDFS and 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
- Automatically creates machine learning modeling pipelines for highest predictive accuracy
- Automatically creates stand-alone scoring pipeline for in-process scoring or client/server scoring via http or tcp protocols.
- Python API or GUI
- 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

Problem types supported:

- Regression (continuous target variable, for age, income, house price, loss prediction)
- Binary classification (0/1 or “N”/”Y”, for fraud prediction, churn prediction, failure prediction, etc.)
- Multinomial classification (0/1/2/3 or “A”/”B”/”C”/”D” for categorical target variables, for prediction of membership type, next-action, product recommendation, etc.)

Data types supported:

- Tabular structured data, rows are observations, columns are fields/features/variables
- i.i.d. (identically and independently distributed) data
- Numeric, categorical and textual fields
- Missing values are allowed
- Time-series data with a single time-series (time flows across the entire dataset, not per block of data)

Data types *NOT* supported:

- Image/video/audio
- Grouped time-series (e.g., sales per item per store per hour, all in one file)

File formats supported:

- Plain text formats of columnar data (.csv, .tsv, .txt)
- Compressed archives (.zip, .gz)

## 1.1 Architecture

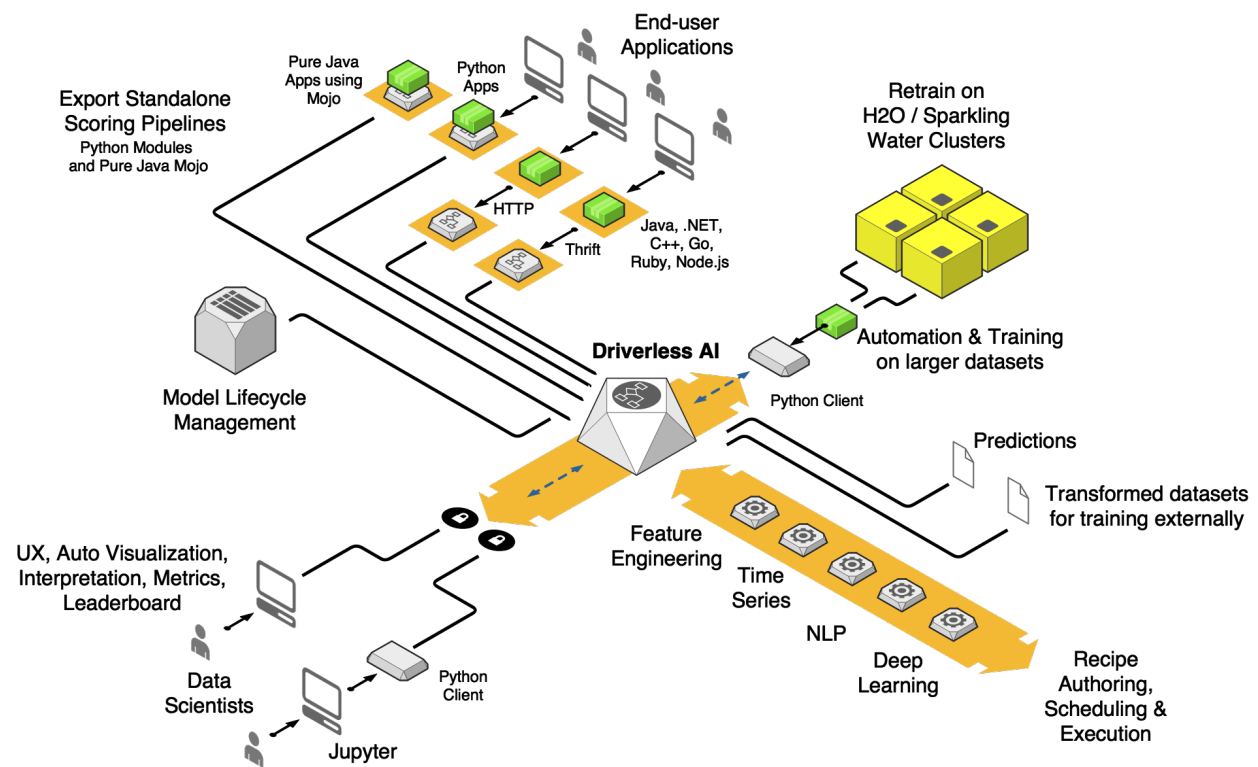


Fig. 1: DAI architecture



## 1.2 Roadmap

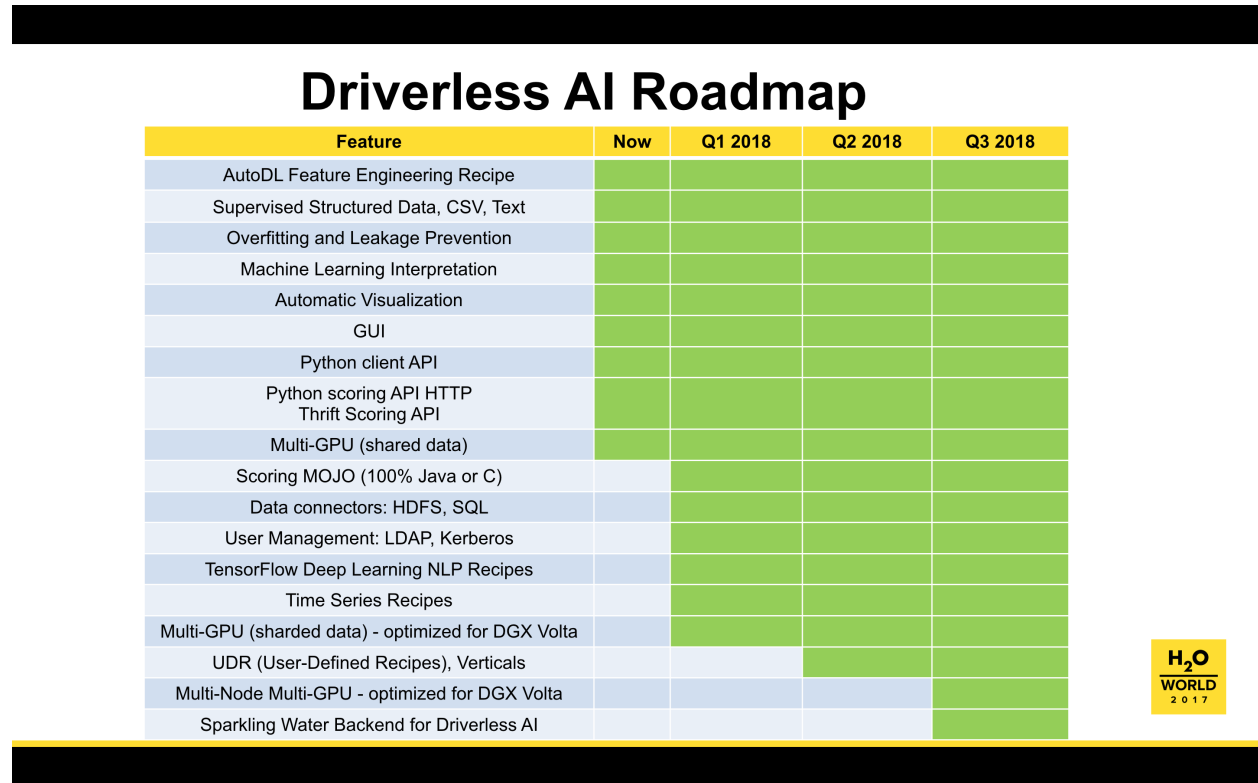


Fig. 2: DAI roadmap

## 1.3 ChangeLog

### 1.3.1 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

### 1.3.2 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

### 1.3.3 Version 1.0.22 (Feb 23 2018)

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

### 1.3.4 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

### 1.3.5 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

### 1.3.6 Version 1.0.19 (Jan 28 2018)

- Fix hang during final ensemble (accuracy  $\geq 5$ ) 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  $\pm 1 \times \text{stddev}$  (instead of  $2 \times \text{stddev}$ )
- Added RMSPE scorer (RMS Percentage Error)
- Added SMAPE scorer (Symmetric Mean Abs. Percentage Error)
- Added AUCPR scorer (Area under Precision-Recall Curve)
- Gracefully handle  $\text{inf}/-\text{inf}$  in data
- Various UI improvements
- Various bug fixes

### 1.3.7 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

### 1.3.8 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

### 1.3.9 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 (+/- 1 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

### 1.3.10 Version 1.0.15 (Jan 11 2018)

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

### 1.3.11 Version 1.0.14 (Jan 11 2018)

- Improved performance

### 1.3.12 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  $\geq 6$
- Use squared Pearson correlation for  $R^2$  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

### 1.3.13 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

#### **1.3.14 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

#### **1.3.15 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

#### **1.3.16 Version 1.0.8 (Nov 21 2017)**

- Various bug fixes

#### **1.3.17 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

### 1.3.18 Version 1.0.5 (Oct 24 2017)

- Only display scorers that are allowed
- Various bug fixes

### 1.3.19 Version 1.0.4 (Oct 19 2017)

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

### 1.3.20 Version 1.0.3 (Oct 9 2017)

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

### 1.3.21 Version 1.0.2 (Oct 5 2017)

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

### 1.3.22 Version 1.0.1 (Oct 4 2017)

- Improved speed of final ensemble
- Various bug fixes

### 1.3.23 Version 1.0.0 (Sep 24 2017)

- Initial stable release

## INSTALLING 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 requires 10 GB of free disk space to run and will stop working if less than 10 GB is available. You should have lots of system CPU memory (64 GB or more) and free disk space (at least 30 GB and/or 10x your dataset size) available.

To simplify cloud installation, Driverless AI is provided as an AMI. To simplify local installation, Driverless AI is provided as a Docker image. 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).

Driverless AI supports HDFS and S3 access. Refer to the [Data Connectors](#) section for more information on how to start Driverless AI with authentication. Or if preferred, a `config.toml` can instead be referenced when starting Driverless AI. Refer to [The Config.toml File](#) section for more information.

Driverless AI also supports basic, LDAP, and PAM authentication, which admins can configure via a `config.toml` file. Refer to [The Config.toml File](#) section to view the properties to set up.

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, contact [sales@h2o.ai](mailto:sales@h2o.ai).

### 2.1 Quick-Start Tables by Environment

Use the following tables for Cloud, Server, and Desktop to find the right setup instructions for your environment.

#### 2.1.1 Cloud

Refer to the following for more information about instance types:

- [AWS Instance Types](#)
- [Azure Instance Types](#)
- [Google Compute Instance Types](#)

Provider	Instance Type	Num GPUs	Suitable for	Refer to Section
NVIDIA GPU Cloud			Serious use	<i>Install on NVIDIA GPU/DGX</i>
AWS	p2.xlarge	1	Experimentation	<i>Install on AWS</i>
	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	
Azure	Standard_NV6	1	Experimentation	<i>Install on Azure</i>
	Standard_NV12	2	Experimentation	
	Standard_NV24	4	Serious use	
	Standard_NC6	1	Experimentation	
	Standard_NC12	2	Experimentation	
	Standard_NC24	4	Serious use	
Google Compute with GPUs				<i>Install on Google Compute with GPUs</i>
Google Compute with CPUs				<i>Install on Google Compute with CPUs</i>

## 2.1.2 Server

Operating System	GPUs?	Min Mem	Refer to Section
NVIDIA DGX-1	Yes	128 GB	<i>Install on NVIDIA GPU/DGX</i>
Ubuntu 16.04	Yes	64 GB	<i>Install on Ubuntu with GPUs</i>
Ubuntu with CPUs	No	64 GB	<i>Install on Ubuntu with CPUs</i>
RHEL with GPUs	Yes	64 GB	<i>Install on RHEL with GPUs</i>
RHEL with CPUs	No	64 GB	<i>Install on RHEL with CPUs</i>
IBM Power (Minsky)	Yes	64 GB	Contact sales@h2o.ai



### 2.1.3 Desktop

Operating System	GPU Support?	Min Mem	Suitable for	Refer to Section
NVIDIA DGX Station	Yes	64 GB	Serious Use	<a href="#">Install on NVIDIA GPU/DGX</a>
Mac OS X	No	16 GB	Experimentation	<a href="#">Install on Mac OS X</a>
Windows 10 Pro	No	16 GB	Experimentation	<a href="#">Install on Windows 10 Pro</a>
Linux	See server table above			

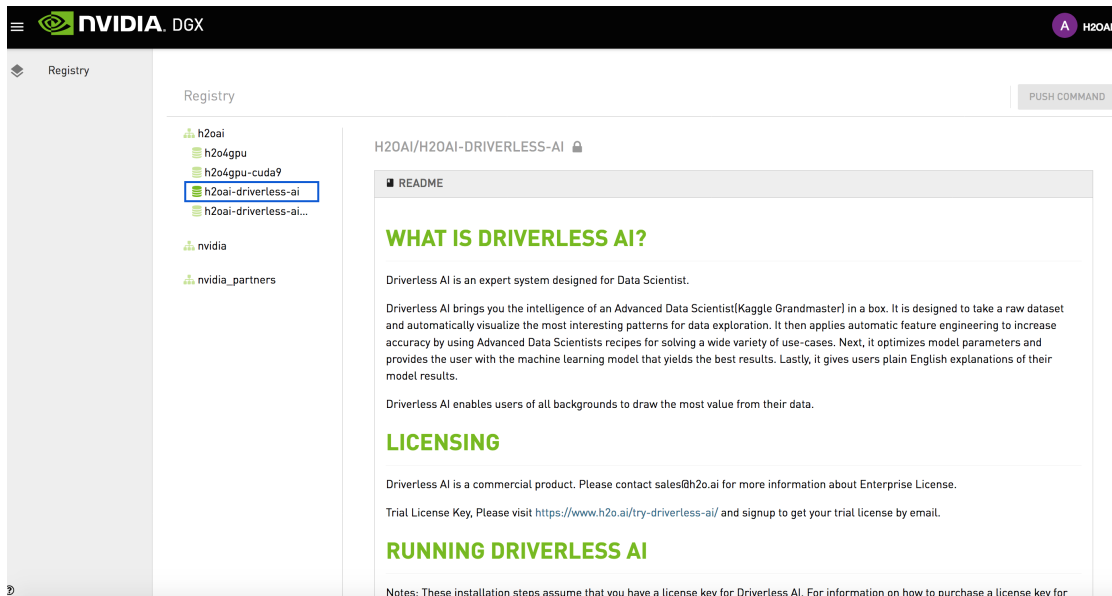
## 2.2 Install on NVIDIA GPU/DGX

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 Station

**Note:** The DGX installation instructions assume that you are running on an NVIDIA DGX machine. Driverless AI is only available in the registry for DGX machines. If you are not running on an NVIDIA DGX machine, then follow the installation instructions for [Ubuntu](#).

1. Log in to your NVIDIA DGX account at <https://compute.nvidia.com/registry>.
2. In the Registry menu, select one of the **h2oai-driverless-ai** options. Note that one registry is for Cuda 8 and the other is for Cuda 9.



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

TAGS		
NAME	LAST MODIFIED *	SIZE
latest	3 days ago	2 GB
1.0.19	3 days ago	2 GB
1.0.11	2 months ago	2.38 GB
1.0.10	2 months ago	2.25 GB
1.0.5	4 months ago	2.25 GB

- 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
```

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

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

- At this point, you can copy data into the data directory on the host machine. The data will be visible inside the Docker container.
- Start the Driverless AI Docker image:

```
nvidia-docker run \
  --rm \
  -u `id -u`:`id -g` \
  -p 12345:12345 \
  -p 54321:54321 \
  -p 9090:9090 \
  -v `pwd`/data:/data \
  -v `pwd`/log:/log \
  -v `pwd`/license:/license \
  -v `pwd`/tmp:/tmp \
  nvcr.io/h2oai/h2oai-driverless-ai:latest
```

Driverless AI will begin running:

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

version: X.Y.Z

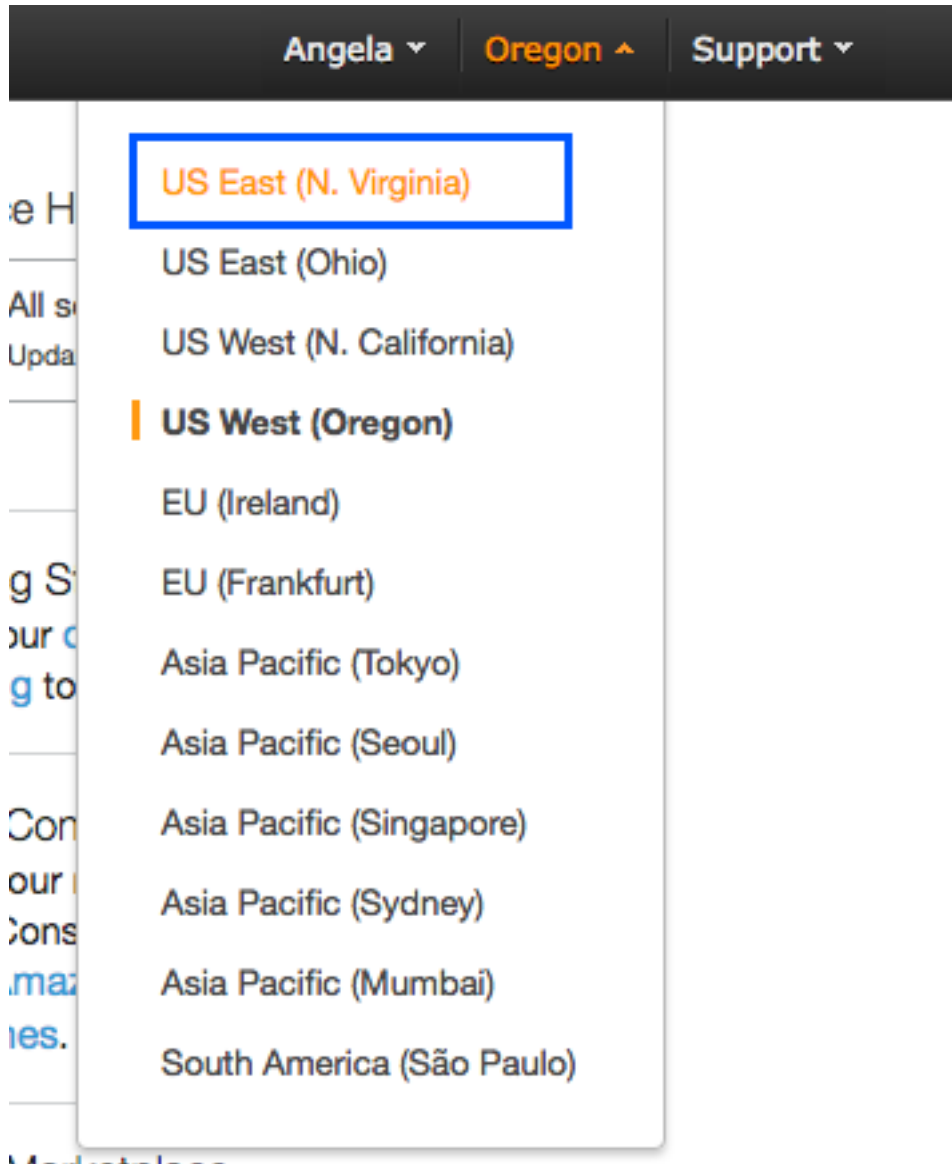
- Put data in the volume mounted at /data
- Logs are written to the volume mounted at /log/YYYYMMDD-HHMMSS
- Connect to Driverless AI on port 12345 inside the container
- Connect to Jupyter notebook on port 8888 inside the container
```

- Connect to Driverless AI with your browser:

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

## 2.3 Install on AWS

1. Log in to your AWS account at <https://aws.amazon.com>.
2. In the upper right corner of the Amazon Web Services page, make sure that the location drop-down is US East (N Virginia).



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

Quick Starts [Hide](#)

Build a web app  
[Start now](#)

Launch a virtual machine  
[Learn more](#)

Build a back end for your mobile app  
[Start now](#)

Host a static website  
[Learn more](#)

AWS Services [Show categories](#)

Search services

**Compute**

**EC2**

EC2 Container Service

Elastic Beanstalk

Lambda

Server Migration

**Developer Tools**

CodeCommit

CodeDeploy

CodePipeline

**Management Tools**

CloudWatch

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

aws Services Resource Groups

EC2 Dashboard

Events

Tags

Reports

Limits

INSTANCES

Instances

Spot Requests

Reserved Instances

Scheduled Instances

Dedicated Hosts

IMAGES

AMIs

Bundle Tasks

ELASTIC BLOCK STORE

Volumes

Resources

You are using the following Amazon EC2 resources in the US East (N. Virginia) region:

0 Running Instances

0 Elastic IPs

0 Dedicated Hosts

0 Snapshots

0 Volumes

0 Load Balancers

1 Key Pairs

2 Security Groups

0 Placement Groups

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.](#)

Create Instance

To start using Amazon EC2 you will want to launch a virtual server, known as an Amazon EC2 instance.

**Launch Instance**

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.

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

**Step 1: Choose an Amazon Machine Image (AMI)** [Cancel and Exit](#)

AWS Marketplace, or you can select one of your own AMIs.

Quick Start

My AMIs

AWS Marketplace

**Community AMIs**

▼ Operating system

- ☐ Amazon Linux
- ☐ Cent OS
- ☐ Debian
- ☐ Fedora
- ☐ Gentoo
- ☐ OpenSUSE
- ☐ Other Linux
- ☐ Red Hat
- ☐ SUSE Linux
- ☐ Ubuntu
- ☐ Windows

▼ Architecture

- ☐ 32-bit

h2oai-driverless-ai-1.0.1 - ami-0531ce7f

h2oai-driverless-ai-1.0.1

Root device type: ebs Virtualization type: hvm

Select

64-bit

h2oai-driverless-ai-1.0.2 - ami-17fa3b6d

h2oai-driverless-ai-1.0.2

Root device type: ebs Virtualization type: hvm

Select

64-bit

h2oai-driverless-ai-1.0.4 - ami-321ec348

h2oai-driverless-ai-1.0.4

Root device type: ebs Virtualization type: hvm

Select

64-bit

h2oai-driverless-ai-1.0.0 - ami-3de71547

h2oai-driverless-ai-1.0.0

Root device type: ebs Virtualization type: hvm

Select

64-bit

h2oai-driverless-ai-1.0.3 - ami-56c10e2c

h2oai-driverless-ai-1.0.3

Select

64-bit

6. 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.

aws Services Resource Groups

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

**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.

Filter by: **GPU compute** Current generation Show/Hide Columns

Currently selected: p3.8xlarge (94 ECUs, 32 vCPUs, 2.7 GHz, Intel Xeon E5-2686 v4, 244 GiB memory, EBS only)

	Family	Type	vCPUs	Memory (GiB)	Instance Storage (GB)	EBS-Optimized Available	Network Performance	IPv6 Support
<input type="checkbox"/>	GPU compute	p2.xlarge	4	61	EBS only	Yes	High	Yes
<input type="checkbox"/>	GPU compute	p2.8xlarge	32	488	EBS only	Yes	10 Gigabit	Yes
<input type="checkbox"/>	GPU compute	p2.16xlarge	64	732	EBS only	Yes	25 Gigabit	Yes
<input type="checkbox"/>	GPU compute	p3.2xlarge	8	61	EBS only	Yes	Up to 10 Gigabit	Yes
<input checked="" type="checkbox"/>	GPU compute	p3.8xlarge	32	244	EBS only	Yes	10 Gigabit	Yes
<input type="checkbox"/>	GPU compute	p3.16xlarge	64	488	EBS only	Yes	25 Gigabit	Yes

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

7. 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**.

### 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, a

Number of instances ⓘ  [Launch into Auto Scaling Group ⓘ](#)

---

Purchasing option ⓘ ☐ Request Spot instances

---

Network ⓘ  [Create new VPC](#)

Subnet ⓘ  [Create new subnet](#)  
48 IP Addresses available

Auto-assign Public IP ⓘ

Placement group ⓘ

---

IAM role ⓘ  [Create new IAM role](#)

⚠ You do not have permissions to list any IAM roles. Contact your administrator, or check your IAM permissions.

---

Shutdown behavior ⓘ

8. 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**.

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	<input type="text" value="128"/>	General Purpose SSD (GP2)	384 / 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 and to (optionally) enable access to H2O Flow, 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
Custom TCP Rule	TCP	54321	Anywhere 0.0.0.0/0	Optional access to H2O Flow

**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:

Description:

Type	Protocol	Port Range	Source	Description
SSH	TCP	22	Custom 0.0.0.0/0	e.g. SSH for Admin Desktop
Custom TCP	TCP	12345	Anywhere 0.0.0.0/0, ::/0	Launch DAI
Custom TCP	TCP	54321	Anywhere 0.0.0.0/0, ::/0	Optional access H2O Flow

**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.

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.

**Select an existing key pair or create a new key pair**

A key pair consists of a **public key** that AWS stores, and a **private key file** that you store. Together, they allow you to connect to your instance securely. For Windows AMIs, the private key file is required to obtain the password used to log into your instance. For Linux AMIs, the private key file allows you to securely SSH into your instance.

Note: The selected key pair will be added to the set of keys authorized for this instance. Learn more about [removing existing key pairs from a public AMI](#).

Choose an existing key pair

**Select a key pair**

mykeypair

☒ I acknowledge that I have access to the selected private key file (mykeypair.pem), and that without this file, I won't be able to log into my 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
```

15. At this point, you can copy data into the data directory on the host machine using `scp`. (Note that the data folder already exists.) For example:

```
scp <data_file>.csv ubuntu@ec2-34-230-6-230.compute-1.amazonaws.com:/home//  
↪data
```

The data will be visible inside the Docker container.

16. Connect to Driverless AI with your browser:

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

### 2.3.1 Stopping the EC2 Instance

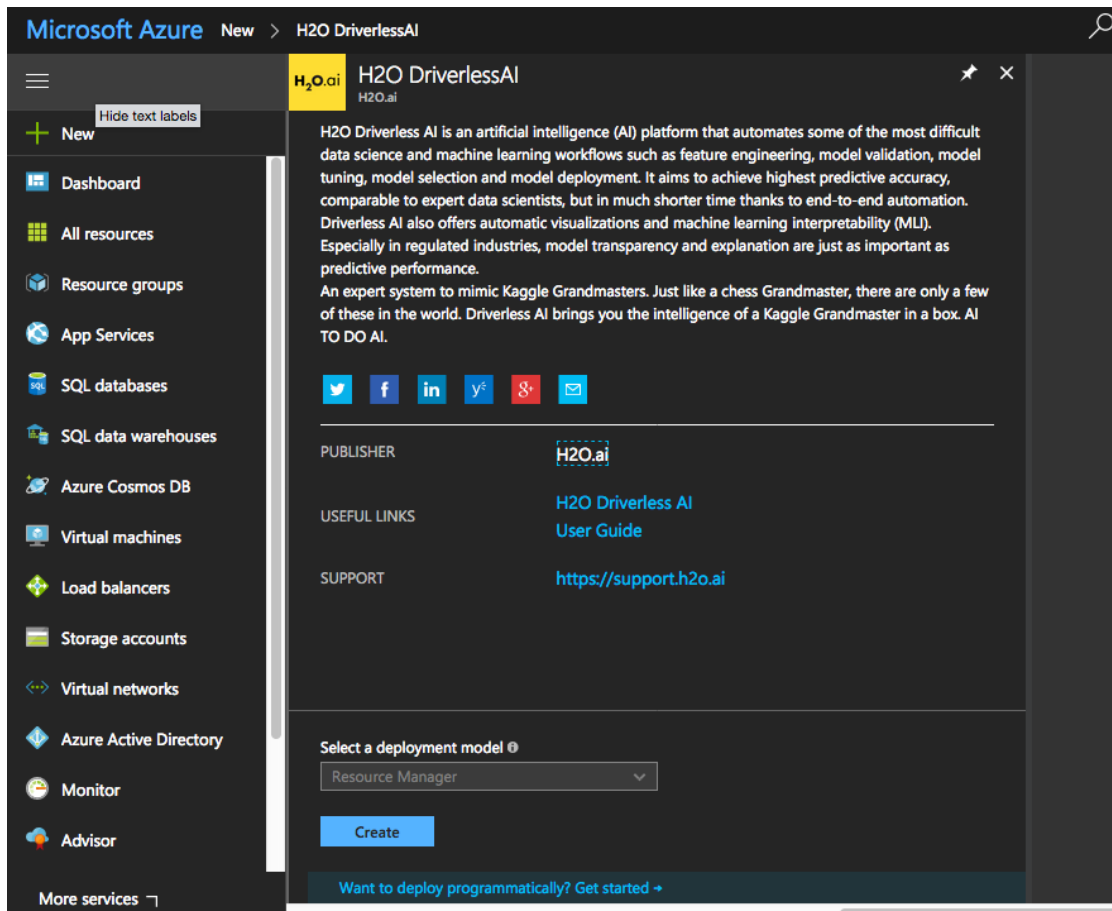
The EC2 instance will continue to run even when you close the `aws.amazon.com` portal. To stop the instance:

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.

## 2.4 Install on Azure

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





3. Click **Create**. This launches the H2O DriverlessAI Virtual Machine creation process.
4. On the **Basics** tab:
  1. Enter a name for the VM.
  2. Select the Disk Type for the VM. Use HDD for GPU instances.
  3. Enter the name that you will use when connecting to the machine through SSH.
  4. Enter and confirm a password that will be used when connecting to the machine through SSH.
  5. Specify the payment method.
  6. Enter a name unique name for the resource group.
  7. Specify the VM region.

Click **OK** when you are done.

Microsoft Azure New > H2O DriverlessAI > Create virtual machine > Basics

Create virtual machine

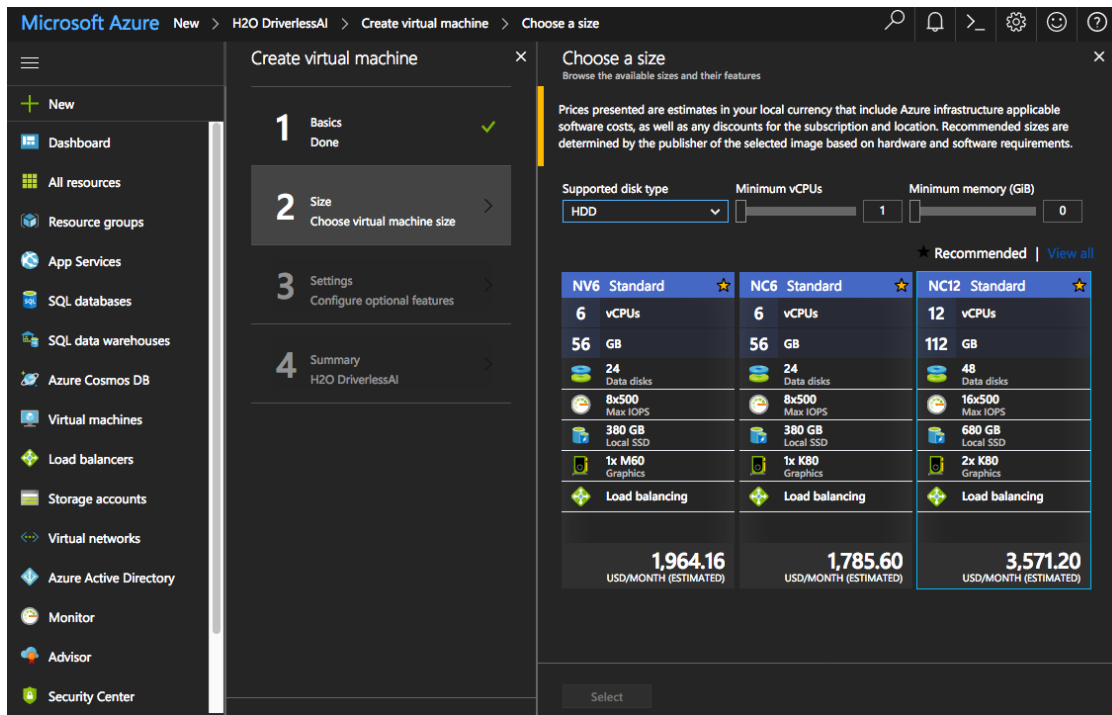
- 1 Basics  
Configure basic settings
- 2 Size  
Choose virtual machine size
- 3 Settings  
Configure optional features
- 4 Summary  
H2O DriverlessAI

Basics

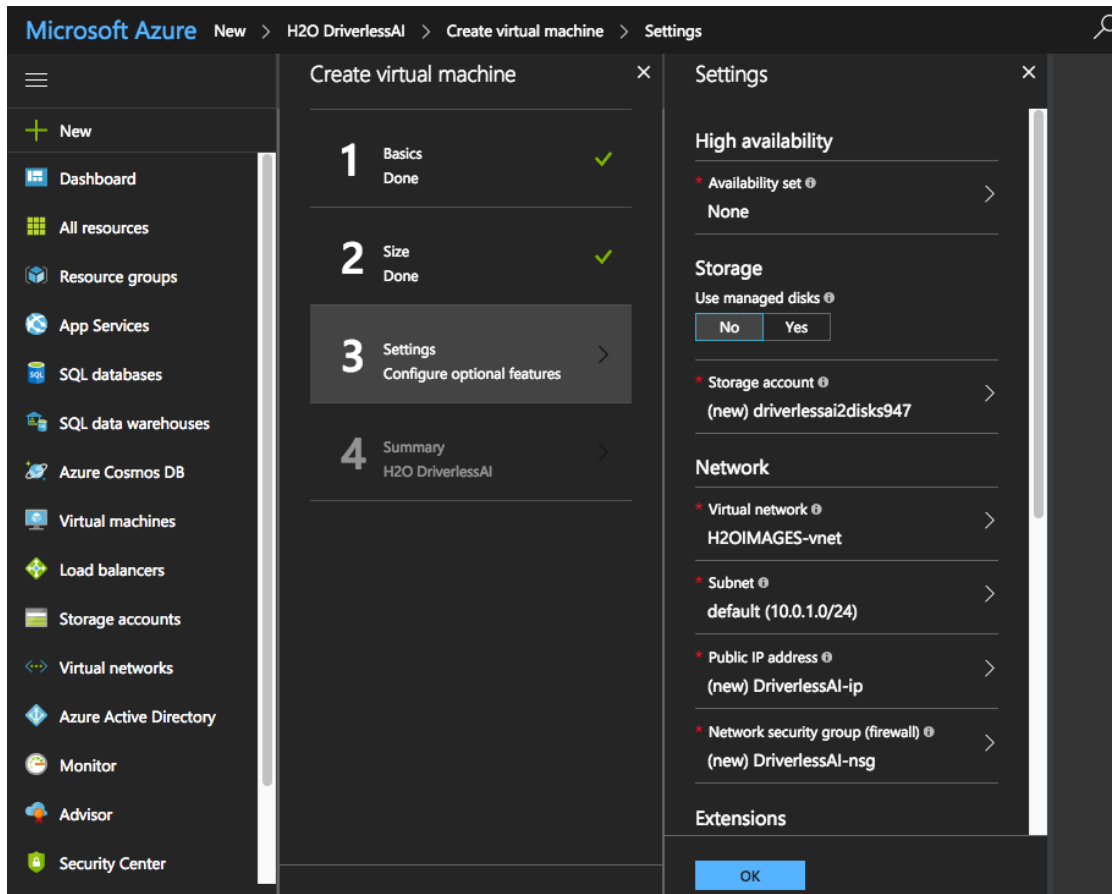
- \* Name: DriverlessAI ✓
- VM disk type ⓘ: HDD ✓
- \* User name: h2o ✓
- \* Authentication type: SSH public key Password
- \* Password: [masked] ✓
- \* Confirm password: [masked] ✓
- Subscription: Pay-As-You-Go ✓
- \* Resource group ⓘ: ☒ Create new ☐ Use existing  
DriverlessAI ✓
- \* Location: East US ✓

OK

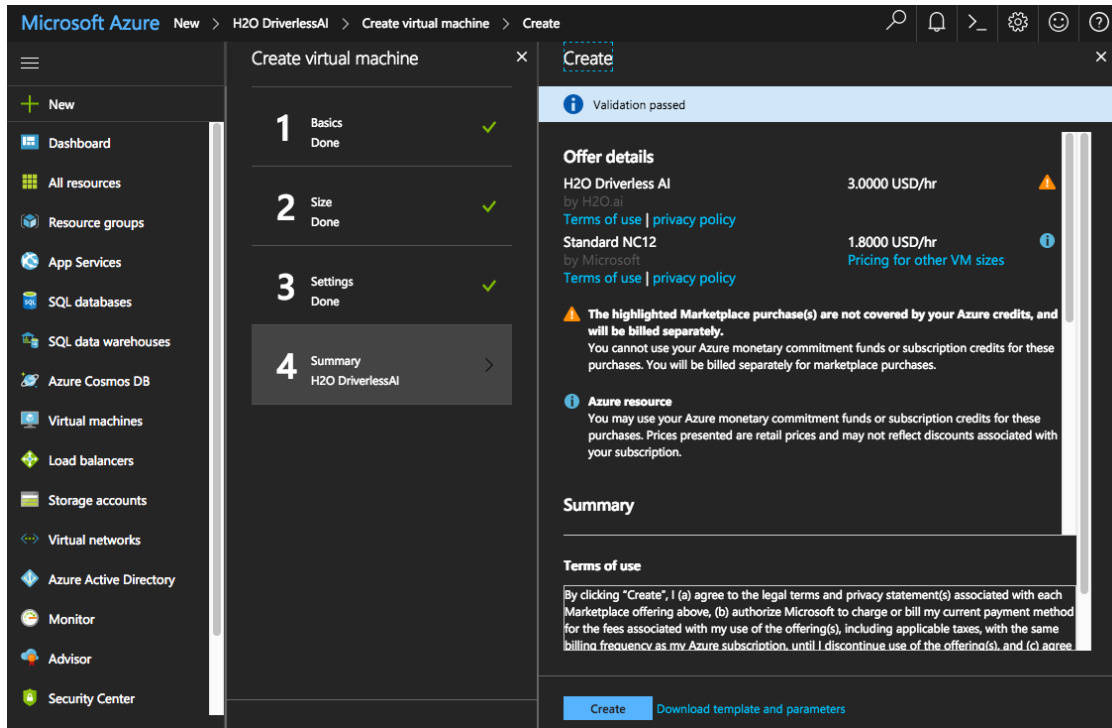
5. On the **Size** tab, select your virtual machine size. Specify the HDD disk type and select a configuration. We recommend using an N-Series type, which comes with a GPU. Also note that Driverless AI requires 10 GB of free space in order to run and will stop working if less than 10 GB is available.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



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. Then open a terminal window and ssh into the machine running the VM. Optionally run `pwd` to retrieve your current location in the VM, and optionally run `nvidia-smi` to verify that the NVIDIA driver is running.
9. Once you are logged in to the VM, use the following command to retrieve the latest Driverless AI version.

```
sudo h2oai update
```

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

```
scp <data_file>.csv <username>@<vm_address>:/etc/h2oai/data
```

The data will be visible inside the Docker container.

11. Start the Driverless AI Docker image

```
sudo h2oai start
```

Driverless AI will begin running:

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

```
version: X.Y.Z
```

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

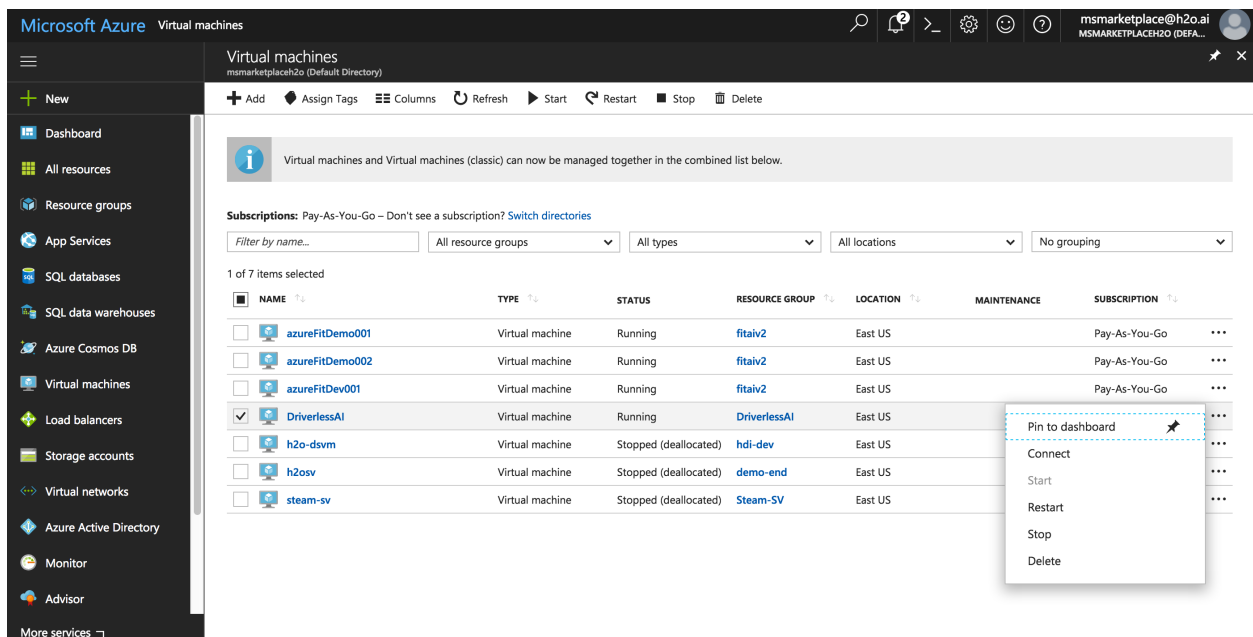
12. Connect to Driverless AI with your browser:

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

### 2.4.1 Stopping the Azure Instance

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

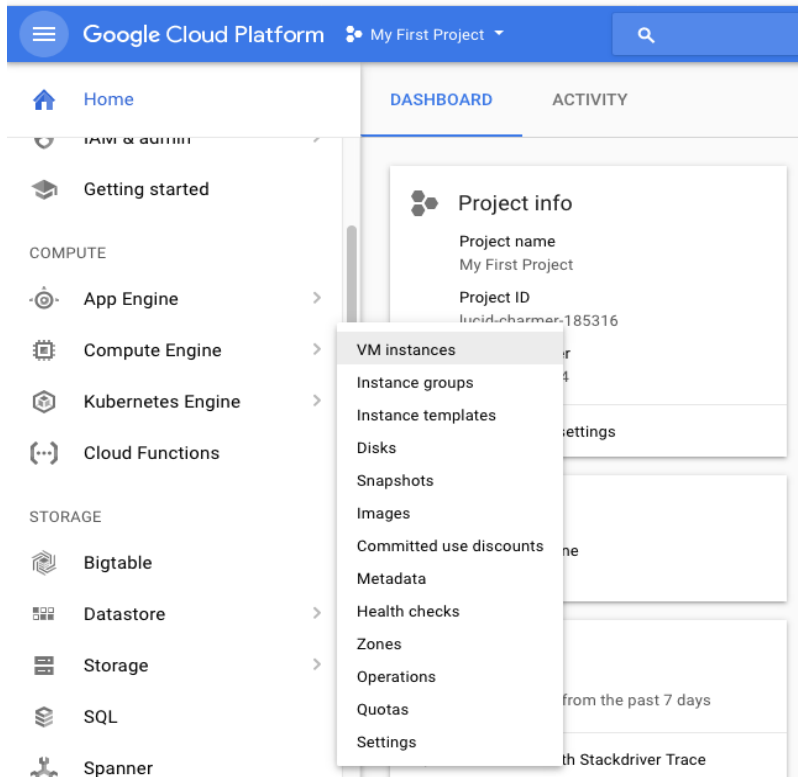
1. Click the **Virtual Machines** left menu item.
2. Select the checkbox beside your DriverlessAI virtual machine.
3. On the right side of the row, click the ... button, then select **Stop**. (Note that you can then restart this by selecting **Start**.)



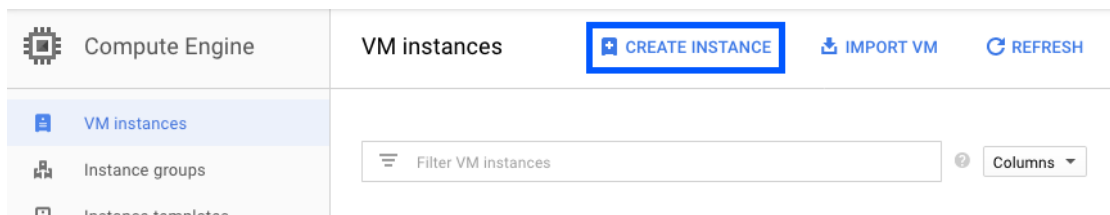
## 2.5 Install on Google Compute with GPUs

This section describes how to install and start Driverless AI in a Google Compute environment with GPUs.

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**.



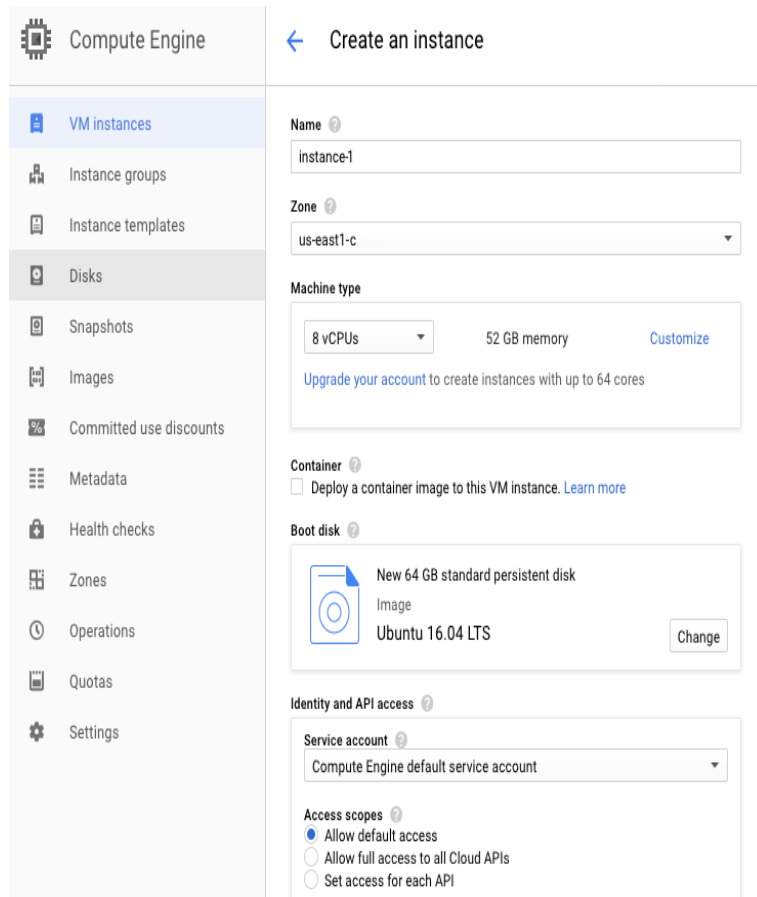
3. Click **Create Instance**.



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.



Compute Engine

← Create an instance

VM instances

Instance groups

Instance templates

Disks

Snapshots

Images

Committed use discounts

Metadata

Health checks

Zones

Operations

Quotas

Settings

Name ?

instance-1

Zone ?

us-east1-c

Machine type

8 vCPUs 52 GB memory [Customize](#)

[Upgrade your account](#) to create instances with up to 64 cores

Container ?

☐ Deploy a container image to this VM instance. [Learn more](#)

Boot disk ?

New 64 GB standard persistent disk

Image

Ubuntu 16.04 LTS [Change](#)

Identity and API access ?

Service account ?

Compute Engine default service account

Access scopes ?

☒ Allow default access

☐ Allow full access to all Cloud APIs

☐ Set access for each API

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:

- (Optional) Specify a unique name for this instance.
- Change the **Targets** dropdown to **All instances in the network**.
- Specify the **Source IP range** to be `0.0.0.0/0`.
- Under **Protocols and Ports**, specify 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](#)

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

**VM instances** [CREATE INSTANCE](#) [IMPORT VM](#) [REFRESH](#) [START](#)

Filter VM instances Columns

Name	Zone	Recommendation	Internal IP	External IP	Connect
<input checked="" type="checkbox"/> instance-1	us-west1-c		10.138.0.2	35.185.245.105	SSH

Open in browser window  
Open in browser window on custom port  
View gcloud command  
Use another SSH client

7. H2O provides a script for you to run in your VM instance. Open an editor in the VM instance (for example, vi). Enter the following text and save the script as **install.sh**.

```
apt-get -y update
apt-get -y --no-install-recommends install \
  curl \
  apt-utils \
  python-software-properties \
  software-properties-common
```

(continues on next page)



(continued from previous page)

```

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

wget -P /tmp https://github.com/NVIDIA/nvidia-docker/releases/download/v1.0.
↳1/nvidia-docker_1.0.1-1_amd64.deb
dpkg -i /tmp/nvidia-docker*.deb && rm /tmp/nvidia-docker*.deb

```

8. Type the following commands to run the install script.

```

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

```

9. 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

```

10. 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. Download the Driverless AI image, replacing X.Y.Z below with your Driverless AI Docker image version (for example, 1.0.21).

```

wget https://s3-us-west-2.amazonaws.com/h2o-internal-release/docker/
↳driverless-ai-docker-runtime-rel-X.Y.Z.gz

```

13. Load the Driverless AI Docker image, replacing X.Y.Z below with your Driverless AI Docker image version (for example, 1.0.21).

```

sudo docker load < driverless-ai-docker-runtime-rel-X.Y.Z.gz

```

14. Start the Driverless AI Docker image with nvidia-docker. Note that you must have write privileges for the folders that are created below. You can replace 'pwd' with the path to your user /home/<username> or start with sudo nvidia-docker run.

```

# Start the Driverless AI Docker image
nvidia-docker run \

```

(continues on next page)

(continued from previous page)

```
--rm \
-u `id -u`:`id -g` \
-p 12345:12345 \
-p 54321:54321 \
-p 9090:9090 \
-v `pwd`/data:/data \
-v `pwd`/log:/log \
-v `pwd`/license:/license \
-v `pwd`/tmp:/tmp \
opsh2oai/h2oai-runtime
```

Driverless AI will begin running:

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

version: X.Y.Z

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

15. Connect to Driverless AI with your browser:

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

## 2.5.1 Stopping the GCE Instance

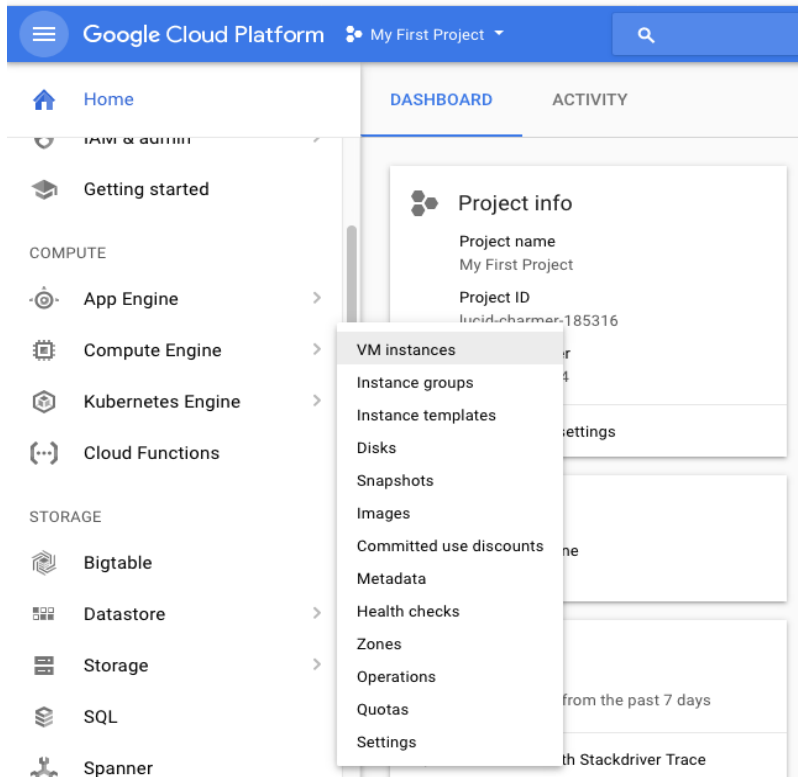
The Google Compute Engine instance will continue to run even when you close the aportal. To stop the instance:

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.

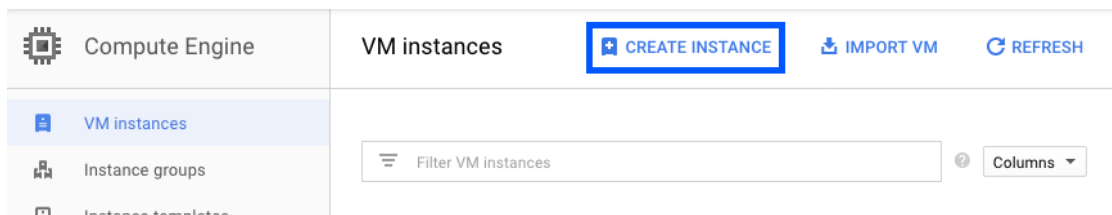
## 2.6 Install on Google Compute with CPUs

This section describes how to install and start Driverless AI in a Google Compute environment with CPUs. Note that this uses Docker CE and not NVIDIA Docker. GPU support will not be available.

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**.



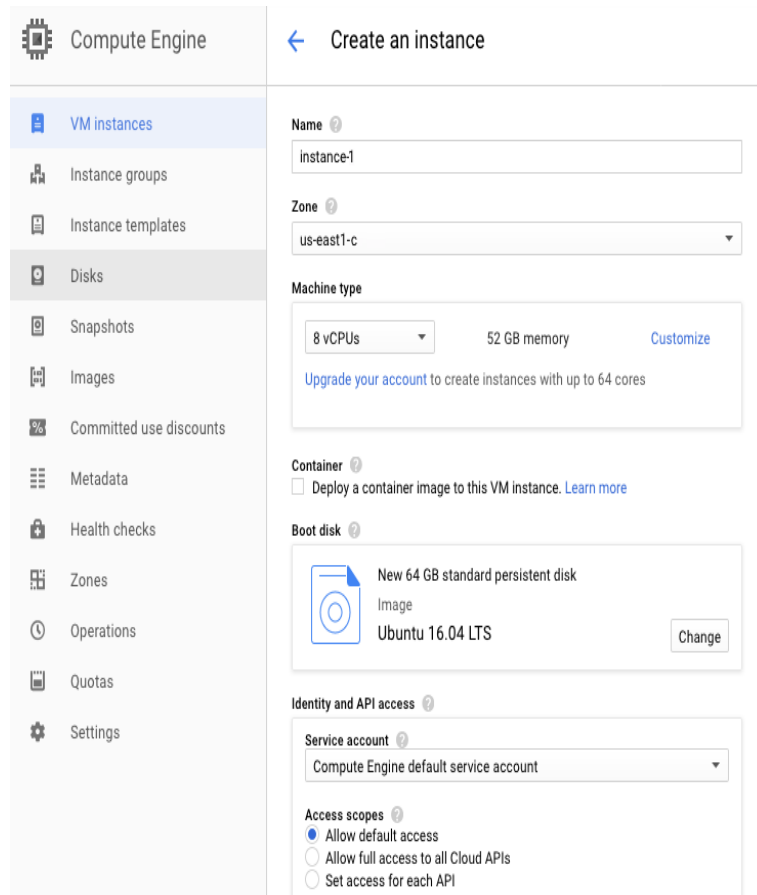
3. Click **Create Instance**.



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.



Compute Engine

← Create an instance

VM instances

Instance groups

Instance templates

Disks

Snapshots

Images

Committed use discounts

Metadata

Health checks

Zones

Operations

Quotas

Settings

Name ?

instance-1

Zone ?

us-east1-c

Machine type

8 vCPUs 52 GB memory [Customize](#)

[Upgrade your account](#) to create instances with up to 64 cores

Container ?

☐ Deploy a container image to this VM instance. [Learn more](#)

Boot disk ?

New 64 GB standard persistent disk

Image

Ubuntu 16.04 LTS [Change](#)

Identity and API access ?

Service account ?

Compute Engine default service account

Access scopes ?

☒ Allow default access

☐ Allow full access to all Cloud APIs

☐ Set access for each API

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:

- (Optional) Specify a unique name for this instance.
- Change the **Targets** dropdown to **All instances in the network**.
- Specify the **Source IP range** to be `0.0.0.0/0`.
- Under **Protocols and Ports**, specify 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](#)

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

**VM instances** [CREATE INSTANCE](#) [IMPORT VM](#) [REFRESH](#) [START](#)

Filter VM instances Columns

Name	Zone	Recommendation	Internal IP	External IP	Connect
<input checked="" type="checkbox"/> instance-1	us-west1-c		10.138.0.2	35.185.245.105	SSH

Open in browser window  
Open in browser window on custom port  
View gcloud command  
Use another SSH client

Select an LABELS  
Labels help on  
No insta

7. H2O provides a script for you to run in your VM instance. Open an editor in the VM instance (for example, vi). Enter the following text and save the script as **install.sh**.

```
apt-get -y update
apt-get -y --no-install-recommends install \
  curl \
  apt-utils \
  python-software-properties \
  software-properties-common
```

(continues on next page)

(continued from previous page)

```
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
```

8. Type the following commands to run the install script.

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

9. 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
```

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

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

11. Download the Driverless AI image, replacing X.Y.Z below with your Driverless AI Docker image version (for example, 1.0.21).

```
wget https://s3-us-west-2.amazonaws.com/h2o-internal-release/docker/
↳driverless-ai-docker-runtime-rel-X.Y.Z.gz
```

12. Load the Driverless AI Docker image, replacing X.Y.Z below with your Driverless AI Docker image version (for example, 1.0.21).

```
sudo docker load < driverless-ai-docker-runtime-rel-X.Y.Z.gz
```

13. Start the Driverless AI Docker image with nvidia-docker. Note that you must have write privileges for the folders that are created below. You can replace 'pwd' with the path to your user /home/<username> or start with sudo nvidia-docker run.

```
# Start the Driverless AI Docker image
nvidia-docker run \
  --rm \
  -u `id -u`:`id -g` \
  -p 12345:12345 \
  -p 9090:9090 \
  -v `pwd`/data:/data \
  -v `pwd`/log:/log \
  -v `pwd`/license:/license \
  -v `pwd`/tmp:/tmp \
  opsh2oai/h2oai-runtime
```

Driverless AI will begin running:

```

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

version: X.Y.Z

- Put data in the volume mounted at /data
- Logs are written to the volume mounted at /log/YYYYMMDD-HHMMSS
- 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:

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

## 2.6.1 Stopping the GCE Instance

The Google Compute Engine instance will continue to run even when you close the aportal. To stop the instance:

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.

## 2.7 Install on Ubuntu with GPUs

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 package from <https://www.h2o.ai/driverless-ai-download/>.
2. Install Docker on Ubuntu (if not already installed):

```

# Install 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

```

3. Install nvidia-docker on Ubuntu (if not already installed):

```

# Install nvidia-docker on Ubuntu
wget -P /tmp https://github.com/NVIDIA/nvidia-docker/releases/download/v1.0.
  ↪1/nvidia-docker_1.0.1-1_amd64.deb
dpkg -i /tmp/nvidia-docker*.deb
rm /tmp/nvidia-docker*.deb

```

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. Load the Driverless AI Docker image, replacing X.Y.Z below with your Driverless AI Docker image version (for example, 1.0.21).

```
# Load the Driverless AI docker image
docker load < driverless-ai-docker-runtime-rel-X.Y.Z.gz
```

6. 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
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. Start the Driverless AI Docker image with nvidia-docker:

```
# Start the Driverless AI Docker image
nvidia-docker run \
  --rm \
  -u `id -u`:`id -g` \
  -p 12345:12345 \
  -p 54321:54321 \
  -p 8888:8888 \
  -v `pwd`/data:/data \
  -v `pwd`/log:/log \
  -v `pwd`/license:/license \
  -v `pwd`/tmp:/tmp \
  opsh2oai/h2oai-runtime
```

Driverless AI will begin running:

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

version: X.Y.Z

- Put data in the volume mounted at /data
- Logs are written to the volume mounted at /log/YYYYMMDD-HHMMSS
- 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
```

## 2.8 Install on Ubuntu with CPUs

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

1. Open a Terminal and ssh to the machine that will run Driverless AI. Once you are logged in, perform the following steps.
2. Retrieve the Driverless AI package from <https://www.h2o.ai/driverless-ai-download/>.
3. Install Docker on Ubuntu (if not already installed):



```
# Install 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
```

4. Load the Driverless AI Docker image, replacing X.Y.Z below with your Driverless AI Docker image version (for example, 1.0.21):

```
# Load the Driverless AI Docker image
docker load < driverless-ai-docker-runtime-rel-X.Y.Z.gz
```

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

```
# 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. Start the Driverless AI Docker image:

```
# Start the Driverless AI Docker image
docker run \
  --rm \
  -u `id -u`:`id -g` \
  -p 12345:12345 \
  -p 54321:54321 \
  -p 8888:8888 \
  -v `pwd`/data:/data \
  -v `pwd`/log:/log \
  -v `pwd`/license:/license \
  -v `pwd`/tmp:/tmp \
  opsh2oai/h2oai-runtime
```

Driverless AI will begin running:

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

version: X.Y.Z

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

8. Connect to Driverless AI with your browser:

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

## 2.9 Install on RHEL with GPUs

This section describes how to install and start the Driverless AI Docker image on RHEL systems with GPUs. Note that the provided nvidia-docker rpm is for x86\_64 machines. nvidia-docker has limited support for ppc64le machines. More information is available [here](#).

**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. Retrieve the Driverless AI package from <https://www.h2o.ai/driverless-ai-download/>.
2. 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, which works even though it's not officially supported.

```
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-docker and the nvidia-docker plugin on RHEL (if not already installed):

```
# Install nvidia-docker and nvidia-docker-plugin
wget -P /tmp https://github.com/NVIDIA/nvidia-docker/releases/download/v1.0.
↪1/nvidia-docker-1.0.1-1.x86_64.rpm
sudo rpm -i /tmp/nvidia-docker*.rpm && rm /tmp/nvidia-docker*.rpm
sudo systemctl start nvidia-docker
```

**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. Load the Driverless AI Docker image, replacing X.Y.Z below with your Driverless AI Docker image version (for example, 1.0.21).

```
# Load the Driverless AI docker image
docker load < driverless-ai-docker-runtime-rel-X.Y.Z.gz
```

6. 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
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. Start the Driverless AI Docker image with nvidia-docker:

```
# Start the Driverless AI Docker image
nvidia-docker run \
  --rm \
  -u `id -u`:`id -g` \
  -p 12345:12345 \
  -p 54321:54321 \
  -p 8888:8888 \
  -v `pwd`/data:/data \
  -v `pwd`/log:/log \
  -v `pwd`/license:/license \
  -v `pwd`/tmp:/tmp \
  opsh2oai/h2oai-runtime
```

Driverless AI will begin running:

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

version: X.Y.Z

- Put data in the volume mounted at /data
- Logs are written to the volume mounted at /log/YYYYMMDD-HHMMSS
- 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>.

## 2.10 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 EE and not NVIDIA Docker. GPU support will not be available.

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

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

Alternatively, you can run on Docker CE, which works even though it's not officially supported.

```
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
```

(continues on next page)

(continued from previous page)

```
sudo yum -y install docker-ce
sudo systemctl start docker
```

3. Load the Driverless AI Docker image, replacing X.Y.Z below with your Driverless AI Docker image version (for example, 1.0.21):

```
# Load the Driverless AI Docker image
docker load < driverless-ai-docker-runtime-rel-X.Y.Z.gz
```

4. Set up the data, log, license, and tmp directories.

```
$ mkdir data
$ mkdir log
$ mkdir license
$ mkdir tmp
```

5. Copy data into the **data** directory on the host. The data will be visible inside the Docker container at **/<user-home>/data**.
6. Start the Driverless AI Docker image with docker. GPU support will not be available.

```
$ docker run \
  --rm \
  -u `id -u`:`id -g` \
  -p 12345:12345 \
  -p 54321:54321 \
  -p 8888:8888 \
  -v `pwd`/data:/data \
  -v `pwd`/log:/log \
  -v `pwd`/license:/license \
  -v `pwd`/tmp:/tmp \
  opsh2oai/h2oai-runtime
```

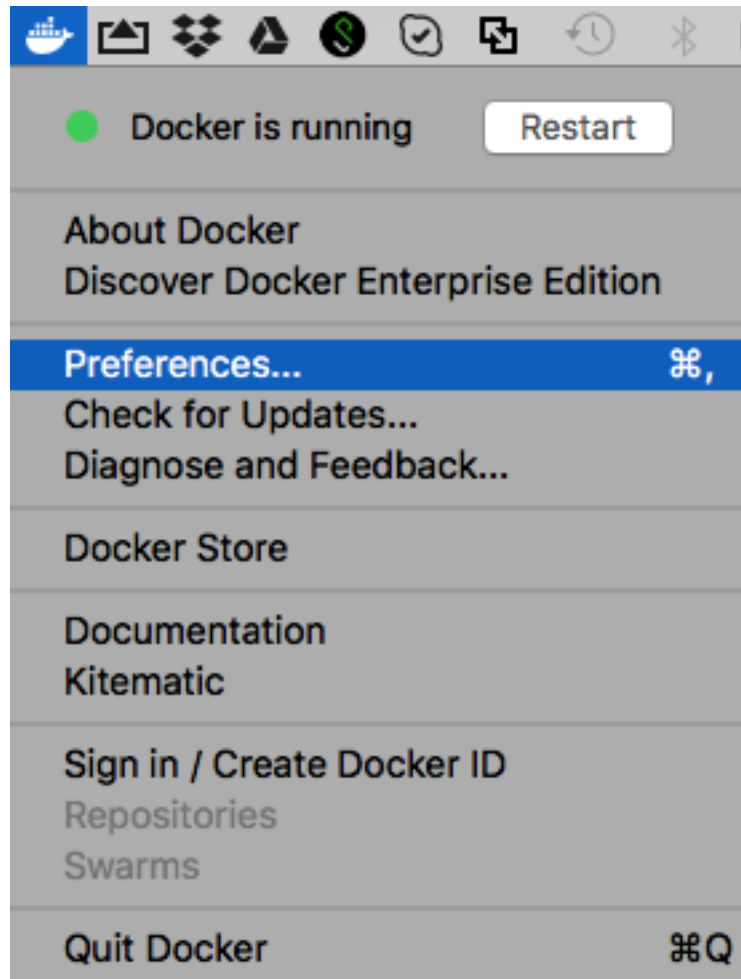
7. Connect to Driverless AI with your browser at <http://Your-Driverless-AI-Host-Machine:12345>.

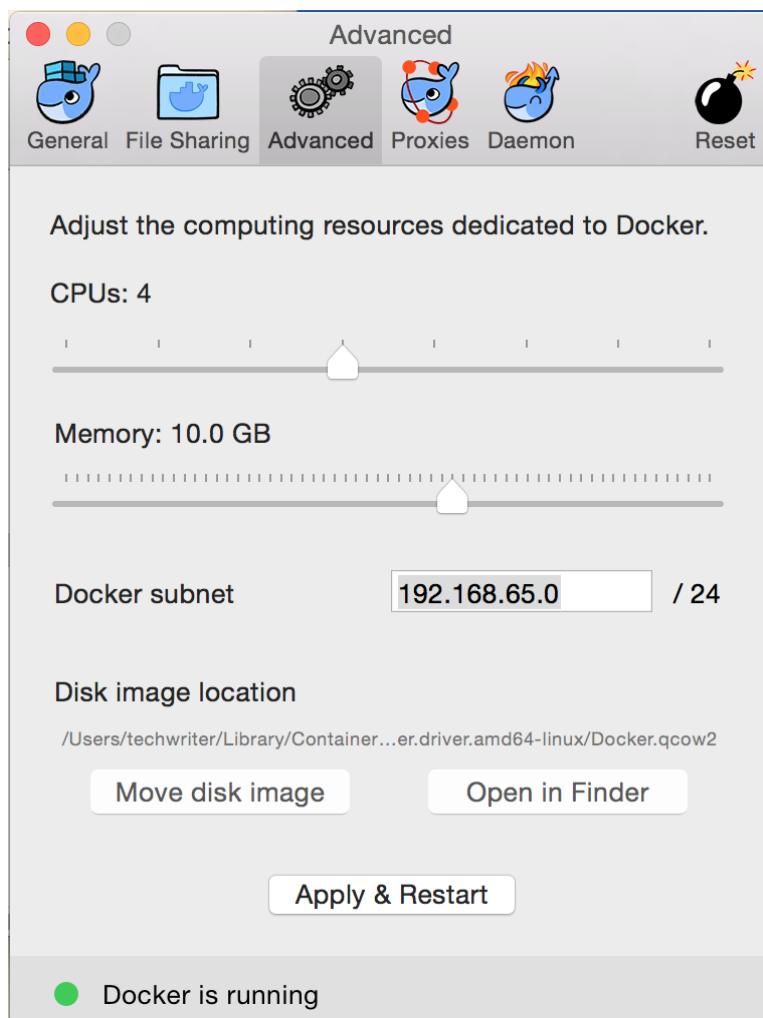
## 2.11 Install on Mac OS X

This section describes how to install and start the Driverless AI Docker image on Mac OS X. Note that this uses regular Docker and not NVIDIA Docker. GPU support will not be available.

### 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>.
1. Retrieve the Driverless AI package from <https://www.h2o.ai/driverless-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. With Docker running, open a Terminal. Navigate to the location of your downloaded Driverless AI and enter the following command, replacing X.Y.Z below with your Driverless AI Docker image version (for example, 1.0.21).

```
$ docker load < driverless-ai-docker-runtime-rel-X.Y.Z.gz
```

5. Set up the data, log, license, and tmp directories.

```
$ 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 **/data**.
7. Start the Driverless AI Docker image with docker. GPU support will not be available.

```
$ docker run \
  --rm \
  -u `id -u`:`id -g` \
  -p 12345:12345 \
  -p 54321:54321 \
  -p 9090:9090 \
```

(continues on next page)

(continued from previous page)

```
-v `pwd`/data:/data \  
-v `pwd`/log:/log \  
-v `pwd`/license:/license \  
-v `pwd`/tmp:/tmp \  
opsh2oai/h2oai-runtime
```

8. Connect to Driverless AI with your browser at <http://localhost:12345>.

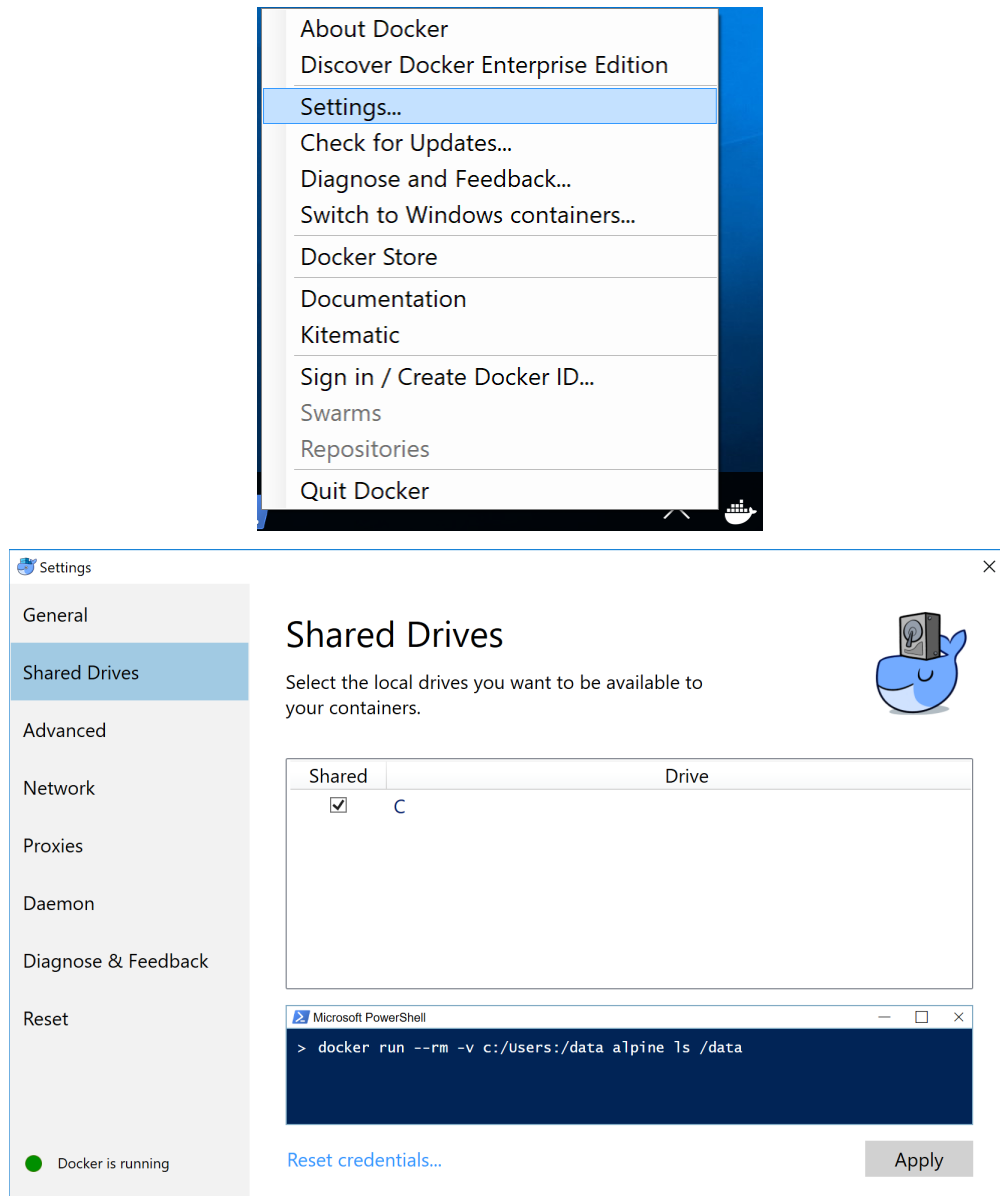
## 2.12 Install on Windows 10 Pro

This section describes how to install and start the Driverless AI Docker image on a Windows 10 Pro machine. Note that this uses regular Docker and not NVIDIA Docker. GPU support will not be available.

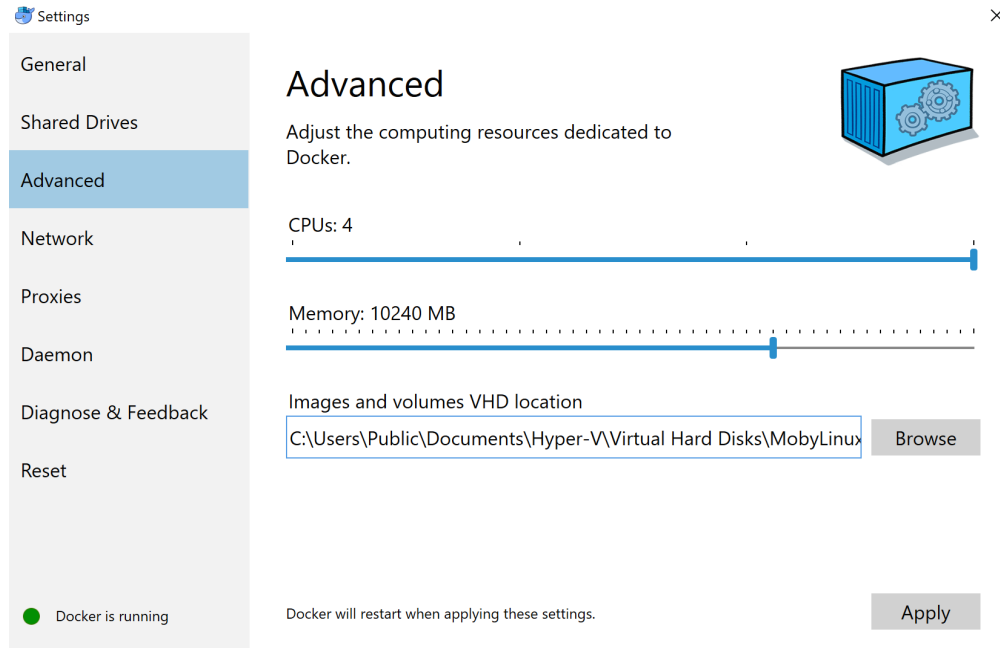
### Caution:

- This should be used only for experimental purposes only on small data. For serious use, please use Linux.
  - Please be aware that there are known issues with Docker for Windows. More information is available here:- <https://github.com/docker/for-win/issues/188>.
1. Retrieve the Driverless AI package from <https://www.h2o.ai/driverless-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**.







4. With Docker running, open a PowerShell terminal. Navigate to the location of your downloaded Driverless AI and enter the following command, replacing X.Y.Z below with your Driverless AI Docker image version (for example, 1.0.21).

```
C:\User>docker load -i .\driverless-ai-docker-runtime-rel-X.Y.Z.gz
```

5. Set up the data, log, license, and tmp directories.

```
C:\User>md data
C:\User>md log
C:\User>md license
C:\User>md tmp
```

6. Copy data into the **/data** directory. The data will be visible inside the Docker container at **/data**.
7. 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.

```
C:\User>docker run --rm -p 12345:12345 -p 54321:54321 -p 9090:9090 -v c:/
→path_to_data:/data -v c:/path_to_log:/log -v c:/path_to_license:/license -
→v c:/path_to_tmp:/tmp opsh2oai/h2oai-runtime
```

8. Connect to Driverless AI with your browser at <http://localhost:12345>.



## UPGRADING DRIVERLESS AI

If you have a valid license and are running the Driverless AI AMI, you can upgrade to the latest version of Driverless AI. Stop Driverless AI, then ssh into the machine that runs Driverless AI and run the following command. Note that this is not available for Mac or Windows users.

```
# Run the following command in Docker
h2oai update
```

And to see all available Driverless AI commands, type `h2oai`.



## RUNNING AN EXPERIMENT

1. After Driverless AI is installed and started, open a browser (Chrome recommended) 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:

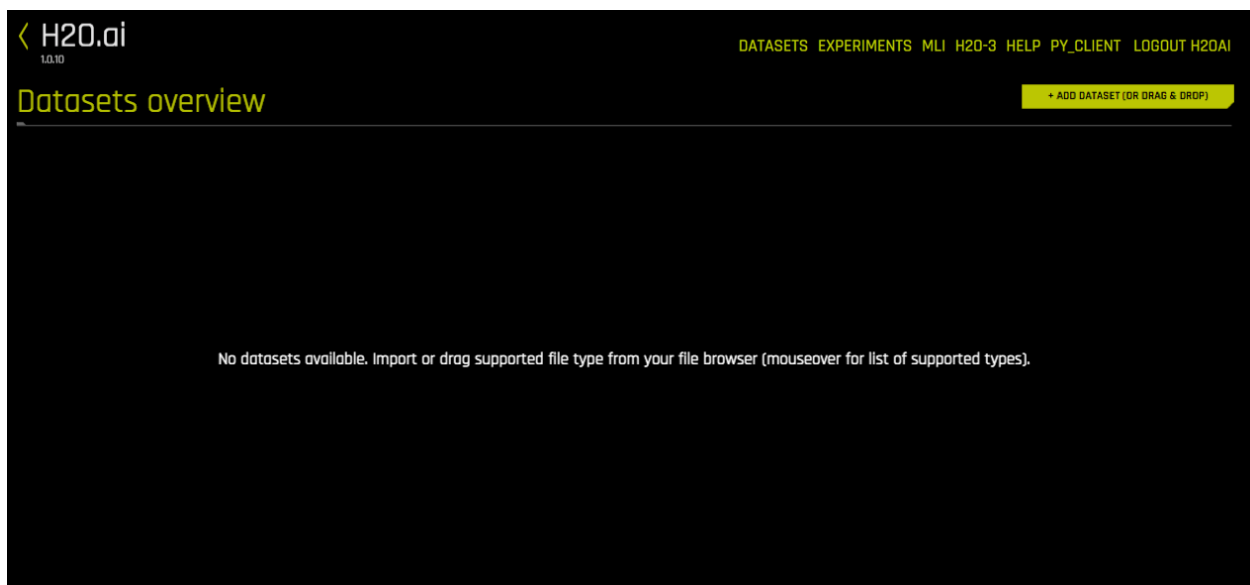
Username: h2oai  
Password: h2oai

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](mailto:sales@h2o.ai) for information on how to purchase a Driverless AI license.

5. The Home page appears, showing all datasets that have been imported. Note that the first time you log in, this list will be empty.



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

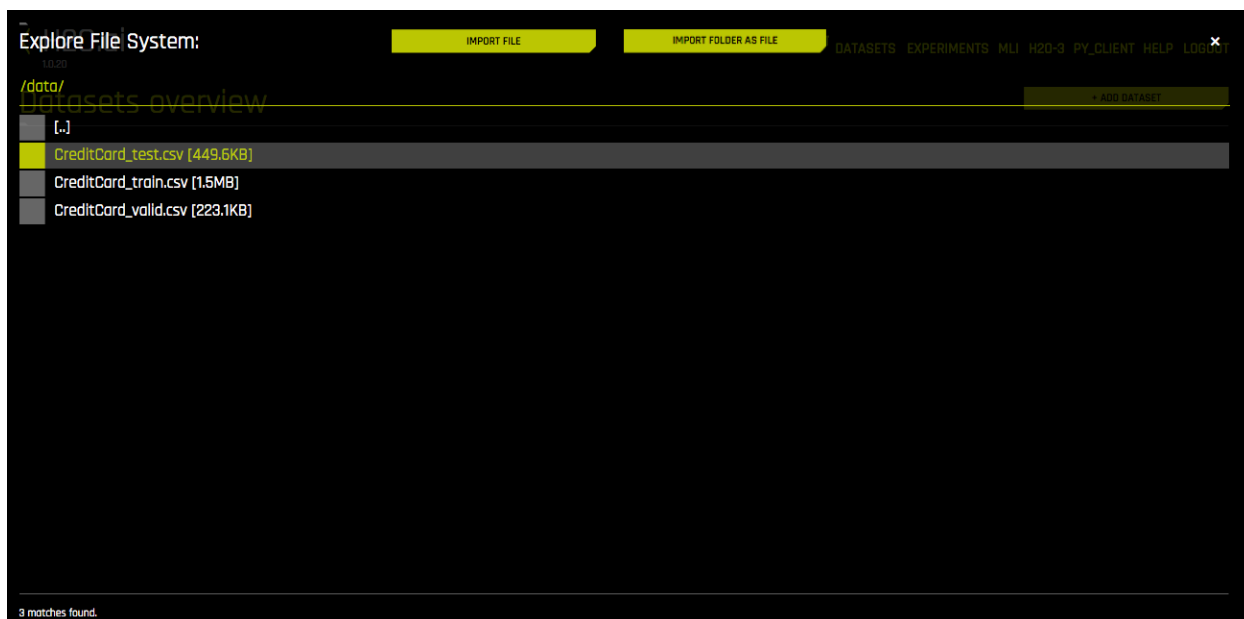
1. Click the **Add Dataset** button. Note that if Driverless AI was started with data connectors enabled for HDFS and/or S3, then a dropdown will appear allowing you to specify where to begin browsing for the dataset. Refer to [Data Connectors](#) for more information.



2. In the **Explore File System** field, type the location for the dataset. Driverless AI autofills the browse line as type in the file location. When you locate the folder that includes your datasets, you can specify to import the folder or to import one or more files.

**Notes:**

- 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.



6. After importing your data, you can run an experiment by selecting **[Click for Actions]** button beside the dataset that you want to use. This opens a submenu that allows you to Visualize or Predict a dataset. (**Note:** You can delete an unused dataset by hovering over it, clicking the **X** button, and then confirming the delete. You cannot delete a dataset that was used in an active experiment. You have to delete the experiment first.) Click **Predict** to begin an experiment.

**H2O.ai** 1.0.24

**Datasets overview**

+ ADD DATASET (OR DRAG & DROP)

	TYPE	ROWS	COLUMNS	STATUS
CreditCard_train.csv /tmp/uploads/CreditCard_train.csv	bin	16784	25	[Click for Actions]
CreditCard_test.csv /tmp/uploads/CreditCard_test.csv	bin	4828		
CreditCard_valid.csv /tmp/uploads/CreditCard_valid.csv	bin	2387		

Visualize

Predict

7. The Experiment Settings form displays and auto-fills with the selected dataset. 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).

8. Specify the target (response) column. Note that not all explanatory functionality will be available for multinomial classification scenarios (scenarios with more than two outcomes).

**H2O.ai**

Select target column:

- 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 month

**TRAINING DATA**

DATASET: credit\_card.csv

ROWS: 24K COLUMNS: 25 DROPPED: --- IGNORED: ---

**TARGET COLUMN**

default payment next month

TYPE: Int32 COUNT: 23999 MEAN: 0.2237 STD DEV: 0.4157

**EXPERIMENT SETTINGS**

WORKERS: 1 ITERATIONS: 50 CV FOLDS: 3 POPULATION: 4

CLASSIFICATION DETECT IDS DROP DUPS

LAUNCH EXPERIMENT

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  $\leq 1.0.19$ , 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.

9. The next step is to set the parameters and settings for the experiment. (Refer to the [Experiment Settings](#) section that follows 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:

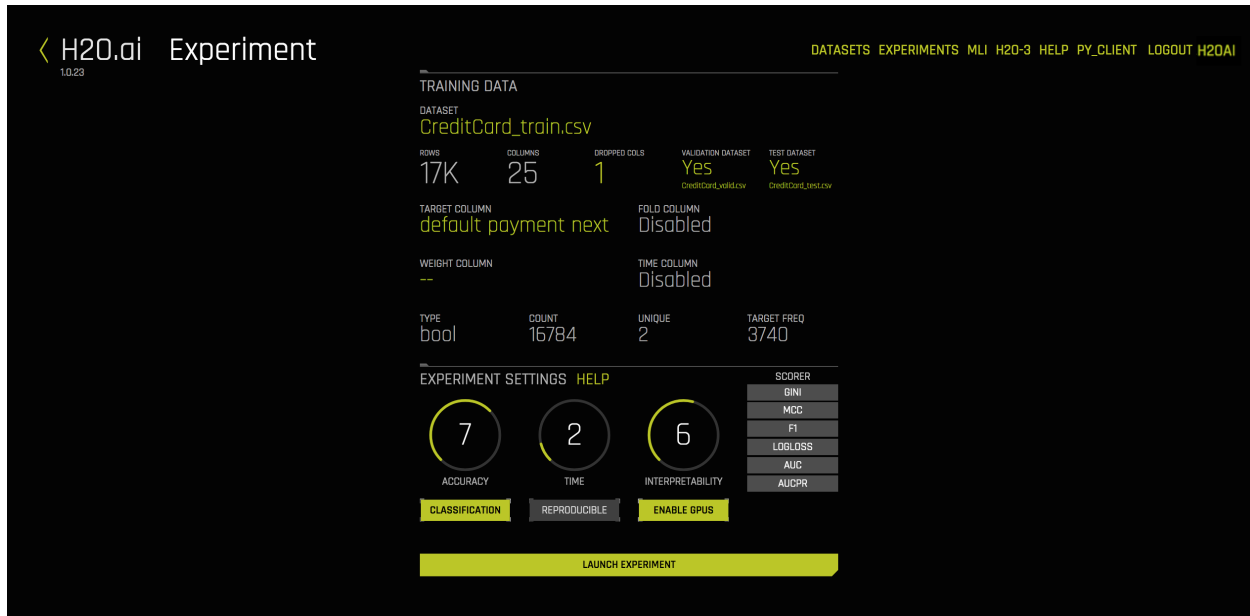
- Dropped Columns: The columns we do not want to use as predictors such as ID columns, columns with data leakage, etc.
- Weight Column: The column that indicates the per row observation weights. If “None” is specified, each row will have an observation weight of 1.
- Fold Column: The column that indicates 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.
- Desired relative accuracy from 1 to 10
- Desired relative time from 1 to 10
- Desired relative interpretability from 1 to 10
- Specify the scorer to use for this experiment. The scorers vary based on whether this is a classification or regression experiment. Available scorers include:
  - Regression: GINI, R2, MSE, RMSE, RMSLE, RMSPE, MAE, MAPE, SMAPE
  - Classification: GINI, MCC, F1, Logloss, AUC, AUCPR

If a scorer not selected, Driverless AI will select one based on the dataset and experiment.

Additional settings:

- If this is a classification problem, then click the **Classification** button. Note that Driverless AI determines the problem type based on the response column. Though not recommended, you can override this setting and specify whether this is a classification or regression problem.
- Click the **Reproducible** button to build this with a random seed.
- Specify whether to enable GPUs. (Note that this option is ignored on CPU-only systems.)





10. Click **Launch Experiment** to start the experiment.

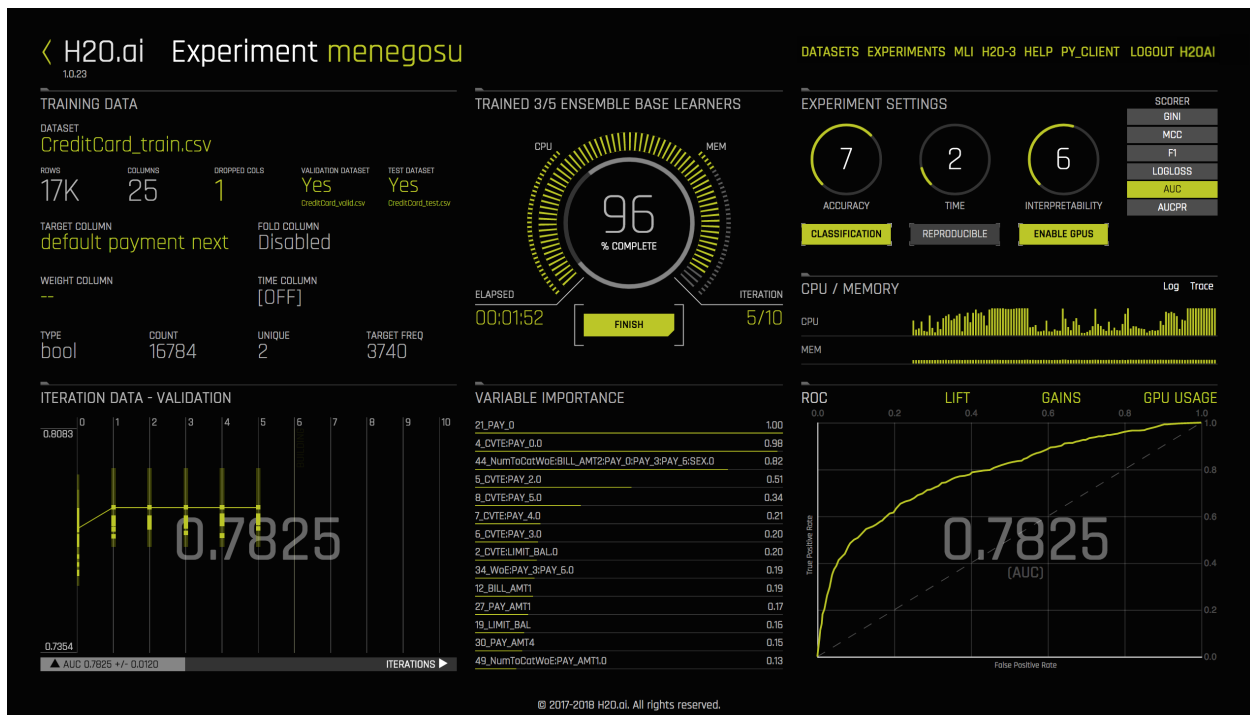
The experiment launches with a randomly generated experiment name. You can change this name at anytime during or after the experiment. Mouse over the name of the experiment to view an edit icon, then type in the desired name.

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.

In addition to the status, the UI also displays details about the dataset, the iteration score (internal validation) for each cross validation fold along with any specified scorer value, the variable importance values, CPU/Memory information, and a toggle between an ROC curve, Lift chart, Gains chart, and GPU Usage information (if GPUs are available). Upon completion, an Experiment Summary section will populate in the lower right section. In this section, you can toggle between the Experiment Summary, ROC curve, Lift chart, and Gains chart.

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.

You can stop experiments that are currently running. Click the **Finish** button to stop the experiment. This jumps the experiment to the end and completes the ensembling and the deployment package. You can also click **Abort** to terminate the experiment. (You will be prompted to confirm the abort.) Note that aborted experiments will not display on the Experiments page.



## 4.1 Experiment Settings

This section describes the settings that are available when running an experiment.

### Dropped Columns

Dropped columns are columns that you do not want to be used as predictors in the experiment.

### 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. This is not generally recommended, but can make sense if the data are non-stationary. In such a case, the validation dataset can help to improve the generalization performance on shifting data distributions.

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  $\geq 0$ . 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 scoring of the test set predictions can use the weight).

### Fold Column

Optional: Column to use to create stratification folds during (cross-)validation, if applicable. Must be of integer or categorical type. Rows with the same value in the fold column represent cohorts, and each cohort is assigned to exactly one fold. This can help to build better models when the data is grouped naturally. If left empty, the data is assumed to be i.i.d. (identically and independently distributed). 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. Note that a fold column cannot be specified if a validation set is used.

### Time Column

Optional: Column that provides a time order, if applicable. Can improve model performance and model validation accuracy for problems where the target values are auto-correlated with respect to the ordering. Each observation's time stamp is used to order the observations in a causal way (generally, to avoid training on the future to predict the past).

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". If [AUTO] is selected, all string columns are tested for potential date/datetime content and considered as potential time columns. The natural row order of the training data is also considered in case no date/datetime columns are detected. If the data is (nearly) identically and independently distributed (i.i.d.), then no time column is needed. If [OFF] is selected, no time order is used for modeling, and data may be shuffled randomly (any potential temporal causality will be ignored). Note that a time column cannot be specified if a validation set is used.

### Accuracy

The following table describes how the Accuracy value affects a Driverless AI experiment.

Ac-cu-racy	Max Rows	En-semble Level	Target Transfor-mation	Parameter Tuning Level	Num Individ-uals	Num Folds	Only First Fold Model	Distri-bution Check
1	10K	0	False	0	Auto	3	True	No
2	100K	0	False	0	Auto	3	True	No
3	500k	0	False	0	Auto	3	True	No
4	1M	0	False	0	Auto	3-4	True	No
5	2M	1	True	1	Auto	3-4	True	Yes
6	5M	1	True	1	Auto	3-5	True	Yes
7	10M	<=2	True	2	Auto	3-10	True	Yes
8	10M	<=2	True	2	Auto	4-10	if >= 5M rows	Yes
9	20M	<=3	True	3	Auto	4-10	if >= 5M rows	Yes
10	None	<=3	True	3	Auto	4-10	if >= 5M rows	Yes

**Note:** A check for a shift in the distribution between train and test is done for accuracy >= 5.

The list below includes more information about the parameters that are used when calculating accuracy.

- **Max Rows:** The maximum number of rows to use in model training
- For classification, stratified random sampling is performed
- For regression, random sampling is performed
- **Ensemble Level:** The level of ensembling done for the final model
- 0: single model

- 1: 2 4-fold models ensembled together
- 2: 5 5-fold models ensembled together
- 3: 8 5-fold models ensembled together
- **Target Transformation:** Try target transformations and choose the transformation that has the best score.

Possible transformations: identity, unit\_box, log, square, square root, inverse, Anscombe, logit, sigmoid

- **Parameter Tuning Level:** The level of parameter tuning done
- 0: no parameter tuning
- 1: 8 different parameter settings
- 2: 16 different parameter settings
- 3: 32 different parameter settings
- Optimal model parameters are chosen based on a combination of the model's accuracy, training speed, and complexity.
- **Num Individuals:** The number of individuals in the population for the genetic algorithms
- Each individual is a gene. The more genes, the more combinations of features are tried.
- The number of individuals is automatically determined and can depend on the number of GPUs. Typical values are between 4 and 16.
- **Num Folds:** The number of internal validation splits done for each pipeline
- If the problem is a classification problem, then stratified folds are created.
- **Only First Fold Model:** Whether to only use the first fold split for internal validation to save time
- Example: Setting Num Folds to 3 and Only First Fold Model = True means you are splitting the data into 67% training and 33% validation.
- **Early Stopping Rounds:** Time-based means based upon the Time table below.
- **Distribution Check:** Checks whether validation or test data are drawn from the same distribution as the training data. Note that this is purely informative to the user. Driverless AI does not take information from the test set into consideration during training.

**Strategy:** Feature selection strategy (to prune-away features that do not clearly give improvement to model score). Feature selection is triggered by interpretability. Strategy = "FS" if interpretability  $\geq 6$ ; otherwise strategy is None.

### Time

This specifies the relative time for completing the experiment (i.e., higher settings take longer). Early stopping will take place if the experiment doesn't improve the score for the specified amount of iterations.

Time	Iterations	Early Stopping Rounds
1	1	None
2	10	5
3	30	5
4	40	5
5	50	10
6	100	10
7	150	15
8	200	20
9	300	30
10	500	50

**Note:** See the Accuracy table for cases when not based upon time.

### Interpretability

Interpretability	Ensemble Level	Monotonicity Constraints
$\leq 5$	$\leq 3$	Disabled
$\geq 6$	$\leq 2$	Disabled
$\geq 7$	$\leq 2$	Enabled
$\geq 8$	$\leq 1$	Enabled
10	0	Enabled

Inter-pretability	Transformers**
$\leq 5$	All
6	Interpretability#5 - [TruncSvdNum, ClusterDist]
7	Interpretability#6 - [ClusterIDTargetEncodeMulti, ClusterIDTargetEncodeSingle]
8	Interpretability#7 - [NumToCatTargetEncodeSingle, NumToCatTargetEncodeMulti, Frequent]
9	Interpretability#8 - [NumToCatWeightOfEvidence, NumToCatWoE, NumCatTargetEncodeMulti, NumCatTargetEncodeSingle]
10	Interpretability#9 - [BulkInteractions, WeightOfEvidence, CvCatNumEncode, NumTo-CatWeightOfEvidenceMonotonic]

\*\* Interpretability# - [lost transformers] explains which transformers are lost by going up by 1 to that interpretability.

\*\* Exception - NumToCatWeightOfEvidenceMonotonic removed for interpretability $\leq 6$ .

\*\* For interpretability  $\leq 10$ , i.e. only [Filter for numeric, Frequent for categorical, DateTime for Date+Time, Date for dates, and Text for text]

- **Target Transformers:**

For regression, applied on target before any other transformations.

Interpretability	Target Transformer
<=10	TargetTransformer_identity
<=10	TargetTransformer_unit_box
<=10	TargetTransformer_log
<= 9	TargetTransformer_square
<= 9	TargetTransformer_sqrt
<= 6	TargetTransformer_logit
<= 6	TargetTransformer_sigmoid
<= 5	TargetTransformer_Anscombe
<= 4	TargetTransformer_inverse

- **Monotonicity Constraints:**

If 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.

- **Date Types Detected:**

- categorical
- date
- datetime
- numeric
- text

- **Transformers used on raw features to generate new features:**

Interpretability	Transformer
<=10	Filter
<=10	Frequent
<=10	DateTime
<=10	Date
<=10	Text
<=9	CvTargetEncodeMulti
<=9	CvTargetEncodeSingle
<=9	CvCatNumEncode
<=9	WeightOfEvidence
<=9 and >=7	NumToCatWeightOfEvidenceMonotonic
<=9	BulkInteractions
<=8	NumToCatWeightOfEvidence
<=8	NumCatTargetEncodeMulti
<=8	NumCatTargetEncodeSingle
<=7	NumToCatTargetEncodeMulti
<=7	NumToCatTargetEncodeSingle
<=6	ClusterIDTargetEncodeMulti
<=6	ClusterIDTargetEncodeSingle
<=5	TruncSvdNum
<=5	ClusterDist

**\*\* Default N-way interactions are up to 8-way except:**

- BulkInteractions are always 2-way.

- Interactions are minimal-way (e.g. 1-way for CvTargetEncode) if interpretability=10.

- **Variable importance threshold in below which features are removed**

Interpretability	Threshold
10	config.toml varimp_threshold_at_interpretability_10
9	varimp_threshold_at_interpretability_10/5.0
8	varimp_threshold_at_interpretability_10/7.0
7	varimp_threshold_at_interpretability_10/10.0
6	varimp_threshold_at_interpretability_10/20.0
5	varimp_threshold_at_interpretability_10/30.0
4	varimp_threshold_at_interpretability_10/50.0
3	varimp_threshold_at_interpretability_10/500.0
2	varimp_threshold_at_interpretability_10/5000.0
1	1E-30





## INTERPRETING A MODEL

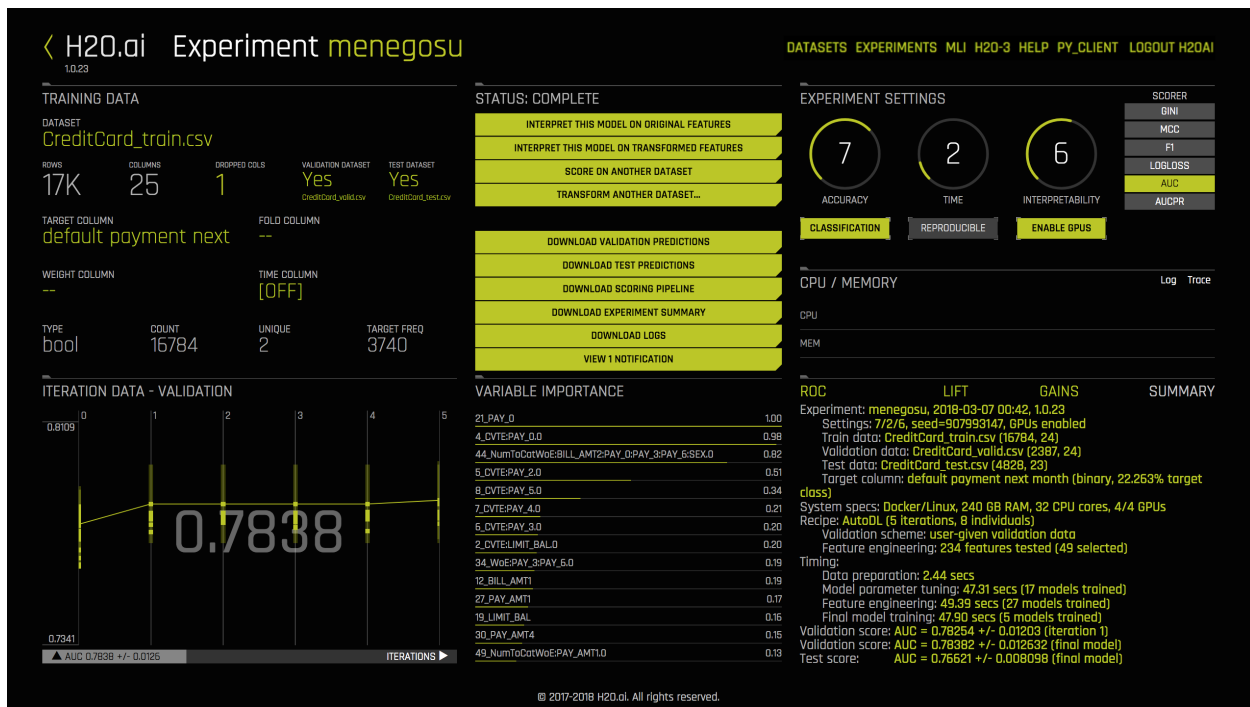
There are two methods you can use for interpreting models:

- Using the **Interpret this Model** button on a completed experiment page to interpret a Driverless AI model.
- Using the **MLI** link in the upper right corner of the UI to interpret either a Driverless AI model or an external model.

### 5.1 Interpret this Model Button

After an experiment status changes from RUNNING to COMPLETE, the UI provides you with several options:

- Interpret this Model on Original Features
- Interpret this Model on Transformed Features
- Score on Another Dataset (Refer to *Score on Another Dataset*.)
- Transform Another Dataset (Refer to *Transform Another Dataset*.)
- Download (Holdout) Training Predictions (in csv format, available if a validation set was NOT provided)
- Download Validation Predictions (in csv format, available if a validation set was provided)
- Download Test Predictions (in csv format, available if a test dataset is used)
- Download Scoring Pipeline (A standalone Python scoring pipeline for H2O Driverless AI. Refer to *The Scoring Pipelines*.)
- Download Experiment Summary (a zip file containing a summary of the experiment and the features along with their relative importance)
- Download Logs
- View Notifications/Warnings (if any existed)

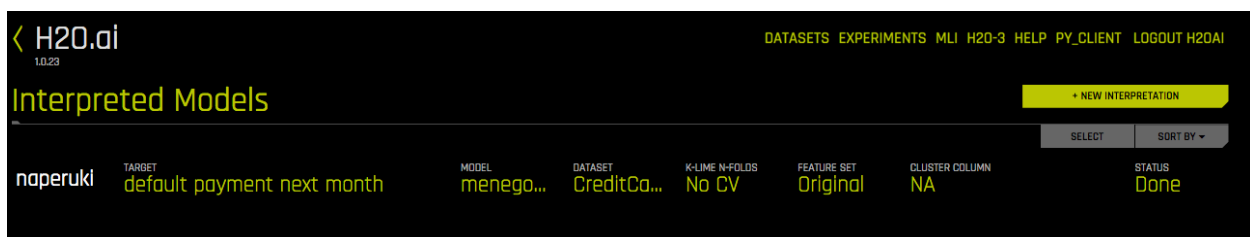


Click one of the **Interpret this Model** buttons to launch the Model Interpretation page. This page provides several visual explanations of the trained Driverless AI model and its results.

## 5.2 Model Interpretation on Driverless AI Models

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.
5. Specify the column of the target variable (the column of actuals for MLI).
6. Optionally specify weight and dropped columns.
7. Optionally specify a clustering column and whether to use the original features.

- Optionally specify the number of cross-validation folds to use in k-LIME. This defaults to 0, and the maximum value is 10.
- Click the **Launch MLI** button.

**SELECT OR IMPORT DATA**

**DATASET**  
CreditCard\_train.csv

**SELECT MODEL:**

**MODEL**  
menegosu

**ROWS**  
17K

**COLUMNS**  
25

**TARGET COLUMN**  
default payment next

**WEIGHT COLUMN**  
--

**DROPPED COLS**  
--

**TYPE**  
bool

**COUNT**  
16784

**UNIQUE**  
2

**FREQ**  
3740

**MLI SETTINGS**

**K-LIME CLUSTERING COLUMN (OPTIONAL)**  
Select clustering column...

☒ **USE ORIGINAL FEATURES**

**k-LIME n-folds:** 0

**LAUNCH MLI**

## 5.3 Model Interpretation on External Models

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.

- Click the **MLI** link in the upper-right corner of the UI to view a list of interpreted models.

**Interpreted Models**

**+ NEW INTERPRETATION**

**SELECT** **SORT BY**

NAME	TARGET	MODEL	DATASET	K-LIME N-FOLDS	FEATURE SET	CLUSTER COLUMN	STATUS
menegosu	default payment next month	menegosu	CreditCa...	No CV	Original	NA	Done

- Click the **New Interpretation** button.
- 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.

**Note:** When running interpretations on an external model, leave the **Select Model** option empty. That option is for selecting a Driverless AI model.

4. Specify a Target Column (actuals) and the Prediction Column (scores from the model).
5. Optionally specify weight and dropped columns.
6. Optionally specify a clustering column.
7. Optionally specify the number of cross-validation folds to use in k-LIME. This defaults to 0, and the maximum value is 10.
8. Click the **Launch MLI** button.

SELECT OR IMPORT DATA

DATASET  
creditcard\_scored.csv

ROWS	COLUMNS
17K	25

TARGET COLUMN  
default payment next

PREDICTION COLUMN  
predictions

WEIGHT COLUMN  
--

DROPPED COLS  
--

TYPE	COUNT	UNIQUE	FREQ
bool	16784	2	3740

MLI SETTINGS

KLIME CLUSTERING COLUMN (OPTIONAL)  
Select clustering column...

k-LIME nfolds: 0

LAUNCH MLI

## 5.4 The Model Interpretation Page

The Model Interpretation page includes the following information:

- Global interpretable model explanation plot
- Variable importance
- Decision tree surrogate model
- Partial dependence and individual conditional expectation plots

Each of these plots and techniques provide different types of insights and explanations regarding a model and its results.



## 5.5 K-LIME

### 5.5.1 The K-LIME Technique

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. 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 variable 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*:  $0.0002 * 1000 = 0.2$

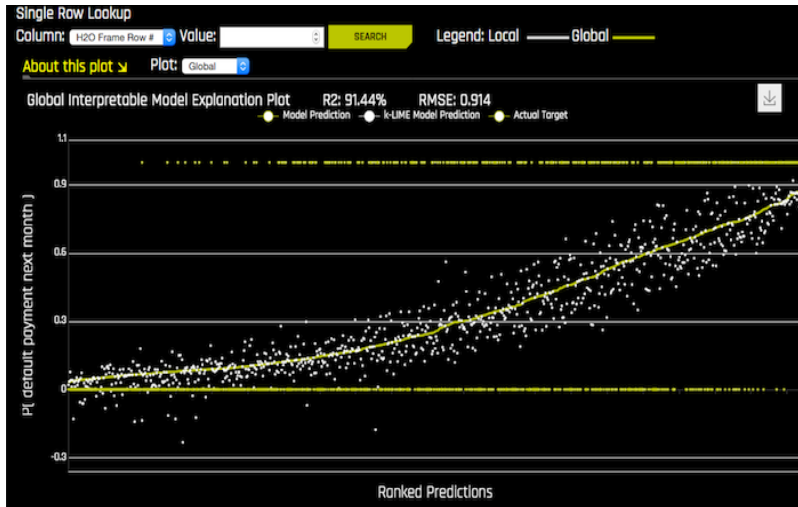
Each local contribution is positive and thus contributes positively to the Driverless AI model's prediction of 0.85 for *H2OAI\_predicted\_default*. By taking into consideration the value of each contribution, reason codes for the Driverless AI decision can be derived. *debt\_to\_income\_ratio* and *credit\_score* 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 variable 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.

## 5.5.2 The Global Interpretable Model Explanation Plot

This plot is in the upper-left quadrant of the UI. It shows Driverless AI model predictions and K-LIME model predictions in sorted order by the Driverless AI model predictions. This graph is interactive. Hover over the **Model Prediction**, **K-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 K-LIME reason codes for that value. By default, this plot shows information for the global K-LIME model, but you can change the plot view to show local results from a specific cluster. The K-LIME plot also provides a visual indication of the linearity of the Driverless AI model and the trustworthiness of the K-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 K-LIME local linear models.



## 5.6 Global and Local Variable Importance

Variable importance measures the effect that a variable has on the predictions of a model. Global and local variable importance values enable increased transparency in the Driverless AI model and enable validating and debugging of the Driverless AI model by comparing global model behavior to the local model behavior, and by comparing to global and local variable importance to known standards, domain knowledge, and reasonable expectations.

### 5.6.1 Global Variable Importance Technique

Global variable importance measures the overall impact of an input variable on the Driverless AI model predictions while taking nonlinearity and interactions into consideration. Global variable importance values give an indication of the magnitude of a variable's contribution to model predictions for all rows. Unlike regression parameters, they are often unsigned and typically not directly related to the numerical predictions of the model. The reported global variable importance values are calculated by aggregating the improvement in the split-criterion for a variable across all the trees in an ensemble. The aggregated feature importance values are then scaled between 0 and 1, such that the most important feature has an importance value of 1.

### 5.6.2 Local Variable Importance Technique

Local variable 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 variable importance values reported here are based on a variant of the leave-one-covariate-out (LOCO) method (Lei et al, 2017).

In the LOCO-variant method, each local variable importance is found 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 variable 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 variable importance values.

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 variable importance values are calculated as follows.

First, the modified predictions are calculated:

$$F_{\text{debt\_to\_income\_ratio}} = F(NA, 600, 1000) = 0.99$$

$$F_{\text{credit\_score}} = F(30, NA, 1000) = 0.73$$

$$F_{\text{savings\_acct\_balance}} = F(30, 600, NA) = 0.82$$

Second, the original prediction is subtracted from each modified prediction to generate the unscaled local variable importance values:

$$\text{LOCO}_{\text{debt\_to\_income\_ratio}} = F_{\text{debt\_to\_income\_ratio}} - 0.85 = 0.99 - 0.85 = 0.14$$

$$\text{LOCO}_{\text{credit\_score}} = F_{\text{credit\_score}} - 0.85 = 0.73 - 0.85 = -0.12$$

$$\text{LOCO}_{\text{savings\_acct\_balance}} = F_{\text{savings\_acct\_balance}} - 0.85 = 0.82 - 0.85 = -0.03$$

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.

$$\text{Scaled}(\text{LOCO}_{\text{debt\_to\_income\_ratio}}) = \text{Abs}(\text{LOCO}_{\text{debt\_to\_income\_ratio}}/0.14) = 1$$

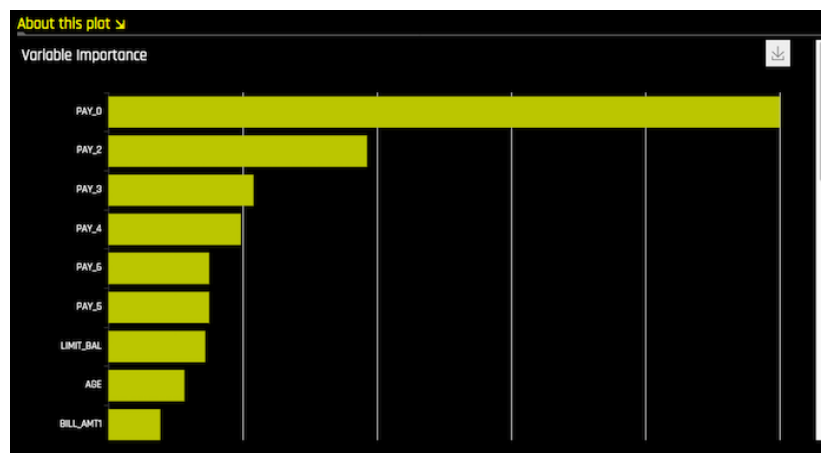
$$\text{Scaled}(\text{LOCO}_{\text{credit\_score}}) = \text{Abs}(\text{LOCO}_{\text{credit\_score}}/0.14) = 0.86$$

$$\text{Scaled}(\text{LOCO}_{\text{savings\_acct\_balance}}) = \text{Abs}(\text{LOCO}_{\text{savings\_acct\_balance}}/0.14) = 0.21$$

One drawback to these LOCO-variant variable importance values is, unlike K-LIME, it is difficult to generate a mathematical error rate to indicate when LOCO values may be questionable.

### 5.6.3 The Variable Importance Plot

The upper-right quadrant of the Model Interpretation page shows the scaled global variable importance values for the features in the model. Hover over each bar in the graph to view the scaled global importance value for that feature. When a specific row is selected, scaled local variable importance values are shown alongside scaled global variable importance values for comparison.





## 5.7 Decision Tree Surrogate Model

### 5.7.1 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. The decision tree surrogate model 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.

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. The decision tree surrogate is known to date back at least to 1996 (Craven and Shavlik). The decision tree surrogate model here 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.

### 5.7.2 The Decision Tree Surrogate Model Plot

The lower-left quadrant shows a decision tree surrogate for the generated model. 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.



## 5.8 Partial Dependence and Individual Conditional Expectation (ICE)

### 5.8.1 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.

### 5.8.2 The ICE Technique

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:

$$ICE_{credit\_score,500} = F(30, 500, 1000)$$

$$ICE_{credit\_score,530} = F(30, 530, 1000)$$

$$ICE_{credit\_score,560} = F(30, 560, 1000)$$

...

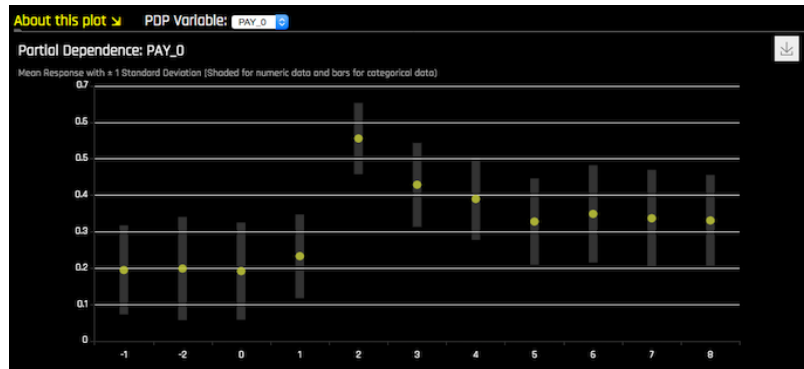
$$ICE_{credit\_score,800} = F(30, 800, 1000)$$

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.

### 5.8.3 The Partial Dependence and Individual Conditional Expectation Plot

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.

The lower-right quadrant shows the partial dependence for a selected variable 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. By default, this graph shows the partial dependence values for the top feature. Change this view by selecting a different feature in the feature drop-down. Note that this graph is available for the top five features.



## 5.9 General Considerations

### 5.9.1 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).

### 5.9.2 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.

### 5.9.3 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 variable 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 variable 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.

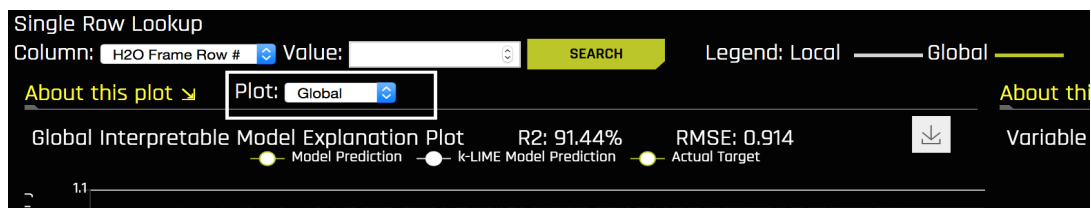
## VIEWING EXPLANATIONS

**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 in the upper-right corner of the Model Interpretation page. Note that this button is only available for completed experiments. Click **Close** when you are done to return to the Model Interpretations page.

The UI allows you to view global, cluster-specific, and local reason codes.

- **Global Reason Codes:** To view global reason codes, select the Global plot from the **Plot** dropdown.



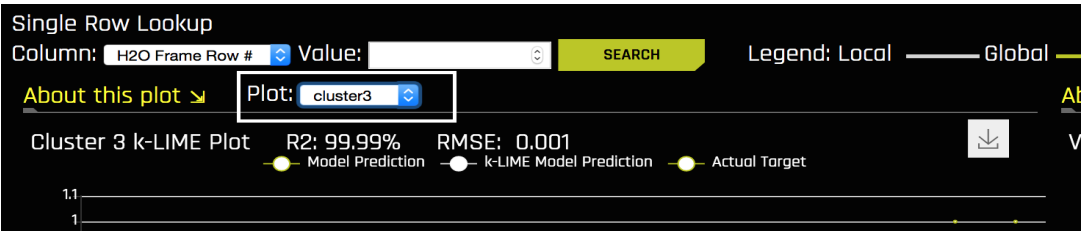
With Global selected, click the **Explanations** button in the upper-right corner.

**Global Reason Codes** [About reason codes](#)

Global interpretable model explains 91.44% in p(default payment next month) for the entire dataset with RMSE = 0.914.

Variable	with value/ 1 unit increase (if blank)	is associated with p(default payment next month)
<b>Top Positive Global Attributions</b>		
PAY_0	2	increase of 0.37
PAY_0	3	increase of 0.33
PAY_0	4	increase of 0.31
Skipped 44 additional attributions, click to view all ...		
<b>Top Negative Global Attributions</b>		
PAY_3	8	decrease of 0.13
PAY_4	7	decrease of 0.11
PAY_2	1	decrease of 0.095
Skipped 38 additional attributions, click to view all ...		

- **Cluster Reason Codes:** To view reason codes for a specific cluster, select a cluster from the **Plot** dropdown.



With a cluster selected, click the **Explanations** button in the upper-right corner.

**Cluster 3 Reason Codes**

k-LIME explains 99.99% in p(default payment next month) for this cluster with RMSE = 0.001.

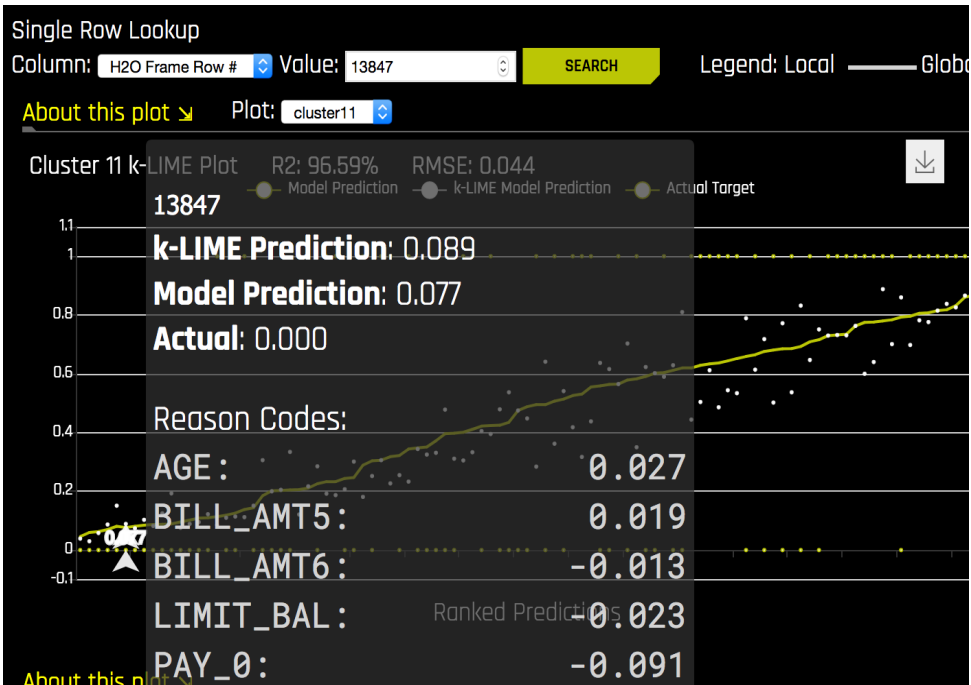
Variable	with value/ 1 unit increase (if blank)	Is associated with p(default payment next month)
Top Positive Cluster Attributions		
PAY_3	2	increase of 0.31
EDUCATION	5	increase of 0.052
PAY_4	-1	increase of 0.043
Skipped 22 additional attributions, click to view all ...		
Top Negative Cluster Attributions		
PAY_4	0	decrease of 0.068
PAY_5	-1	decrease of 0.052
PAY_5	-1	decrease of 0.042
Skipped 60 additional attributions, click to view all ...		

**Global Reason Codes** About reason codes

Global interpretable model explains 91.44% in p(default payment next month) for the entire dataset with RMSE = 0.914.

Variable	with value/ 1 unit increase (if blank)	Is associated with p(default payment next
----------	---	--

- **Local Reason Codes:** To view local reason codes, select a point on the graph or type a value in the Value field.



With a value selected, click the **Explanations** button in the upper-right corner.

Actual and Predicted Values

p(default payment next month) (Actual)	0
Model Prediction Value	0.08
k-LIME Prediction Value	0.09
k-LIME Prediction Accuracy	88.9%

Local Reason Codes

k-LIME Local Attributions	Variable	with value	is associated with p(default payment next month)	
Top Positive Local Attributions				
	AGE	37	increase of	0.027
	BILL_AMT5	91678	increase of	0.019
	BILL_AMT1	86701	increase of	0.0085
Skipped 5 additional attributions, click to view all ...				
Top Negative Local Attributions				
	PAY_0	0	decrease of	0.091
	LIMIT_BAL	240000	decrease of	0.023
	BILL_AMT6	94280	decrease of	0.013
Skipped 12 additional attributions, click to view all ...				





## 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 that you want to score on. Note that this new dataset must include the same columns as the dataset used in selected experiment.
4. Click **Select** at the top of the screen. This immediately starts the scoring process.
5. Click the **Download Predictions** button after scoring is complete.



## 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 [Appendix A: 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. The new rows will then get transformed into the original dataset.

Follow these steps to transform another dataset. Note that this assumes the new dataset has been added to Driverless AI already.

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).
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 screenshot shows the 'Transform Another Dataset' configuration window. At the top, it says 'Experiment bbd7b2'. The 'TRAINING DATASET' is set to '/data/CreditCard-train-new-rows.csv'. Below it, 'TRAINING DATA' is listed. A 'SELECT' dropdown menu is visible. The 'FOLD COLUMN' is set to '24K'. The 'TEST DATASET' is set to '/data/CreditCard-test.csv'. The 'VALIDATION SPLIT' is set to '0.25'. The 'SEED' is set to '1234'. A 'LAUNCH TRANSFORMATION' button is at the bottom left. On the right, a 'STATUS: COMPLETE' message is shown, along with a list of download links for the transformed data.

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.

**Transform Another Dataset** ×

**New datasets ready.** ×

**SELECT** ▼

**FOLD COLUMN**  
Select fold column  
1

**VALIDATION SPLIT**  
0.25

**SEED**  
1235

**TEST DATASET**  
/data/CreditCard-test.csv

**LAUNCH TRANSFORMATION**

**DOWNLOAD TRAINING DATASET (NOT FOR CROSS-VALIDATION)**

**DOWNLOAD VALIDATION DATASET (FOR PARAMETER TUNING)**

**DOWNLOAD TEST DATASET (FOR FINAL SCORING)**

## THE SCORING PIPELINES

Scoring Pipelines are available for productionizing models and for obtaining reason codes on interpreted models for a given row of data.

### 9.1 Driverless AI Standalone 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).

#### 9.1.1 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).

#### 9.1.2 Prerequisites

The following are required in order to run the scoring pipeline.

- Linux x86\_64 environment

- Python 3.6 (Note that Anaconda Python 3.6 distribution is not supported at this point in time.)
- libopenblas-dev (required for H2O4GPU)
- Apache Thrift (to run the scoring service in TCP mode)
- Internet access to download and install packages. Note that depending on your environment, you may also need to set up proxy.

The scoring pipeline has been tested on Ubuntu 16.04 and on 16.10+. Examples of how to install these prerequisites are below.

### 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 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
```

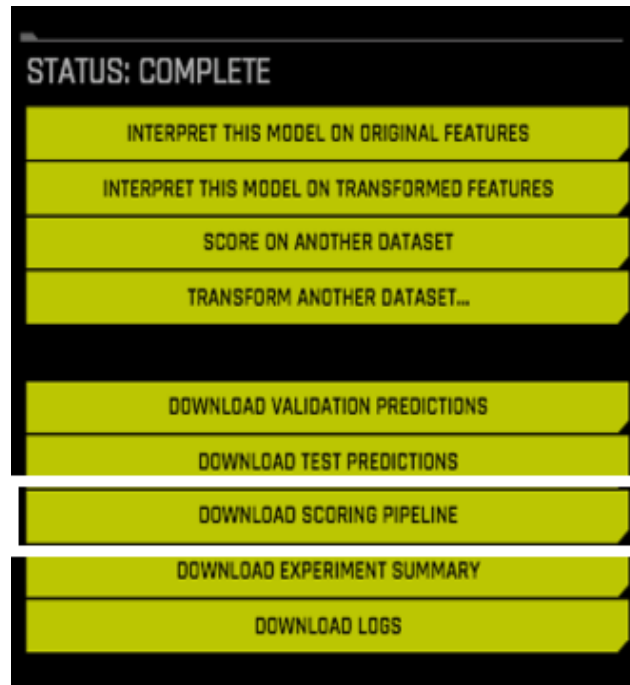
Run the following to refresh the runtime shared after installing Thrift:

```
$ sudo ldconfig /usr/local/lib
```

## 9.1.3 Quickstart

Before running the quickstart examples, be sure that the scoring pipeline is already downloaded and unzipped:

1. On the completed Experiment page, click on the **Download 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 x86\_64 and Python 3.6.)

```
$ 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.)

```
$ 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.)

```
$ bash run_tcp_server.sh
$ bash run_tcp_client.sh
```

**Note:** 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 at the end of this chapter to see how to set up and test the scoring module using a cleanroom Ubuntu 16.04 virtual machine.

### 9.1.4 The 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
$ python example.py
```

### 9.1.5 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).

#### 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-4.4.1.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;
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));
```

(continues on next page)

(continued from previous page)

```

    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.
$ 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'])

```

## 9.1.6 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/
bash run_example.sh

```

At this point, you should see scores printed out on the terminal. If not, contact us at [support@h2o.ai](mailto:support@h2o.ai).

## 9.2 Driverless AI MLI Standalone 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.

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).

### 9.2.1 MLI Scoring Package Files

The **scoring-pipeline-ml** 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.)
- **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.
- **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).

### 9.2.2 Prerequisites

- The scoring module and scoring service are supported only on Linux x86\_64 with Python 3.6.
- Apache Thrift (to run the scoring service in TCP mode)
- The scoring module and scoring service needs access to Internet to download and install packages. Depending on your environment, you may also need to set up proxy.

#### 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 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
```

### 9.2.3 Quickstart

Before running the quickstart examples, be sure that the MLI Scoring Package is already downloaded and unzipped.

1. On the MLI page, click the **Scoring Pipeline** button.



2. Unzip the scoring pipeline, and run the following examples in the **scoring-pipeline-ml** folder.

Run the scoring module example. (This requires Linux x86\_64 and Python 3.6.)

```
$ bash run_example.sh
```

Run the TCP scoring server example. Use two terminal windows. (This requires Linux x86\_64, 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 x86\_64, Python 3.6 and Thrift.)

```
$ bash run_http_server.sh
$ bash run_http_client.sh
```

### 9.2.4 MLI 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_487931_20170921174120_b4066 import Scorer
scorer = KTimeScorer()          # 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 KTime 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
```

## 9.2.5 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) or in batch mode (multiple rows at a time).

### 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;
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_reason_codes(row);
            System.out.println(scores);
        }
    }
}
```

(continues on next page)

(continued from previous page)

```
        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. -----
$ 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
```

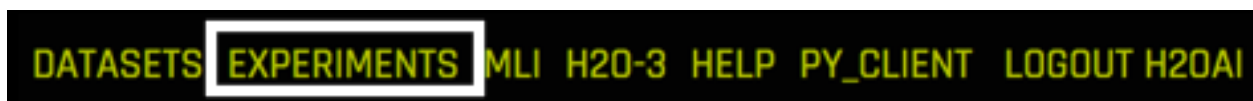
Similarly, you can use any HTTP client library to reproduce the above result. For example, from Python, you can use the requests module as follows:

```
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'])
```

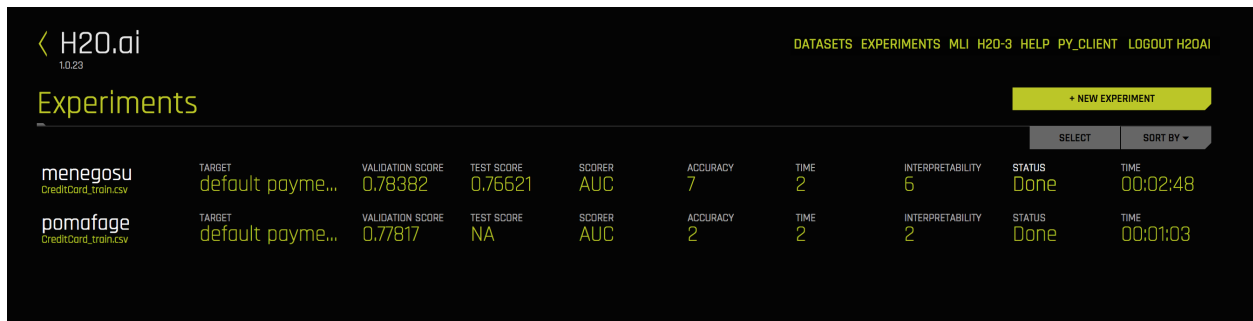


## VIEWING EXPERIMENTS

The upper-right corner of the Driverless AI UI includes an **Experiments** link.

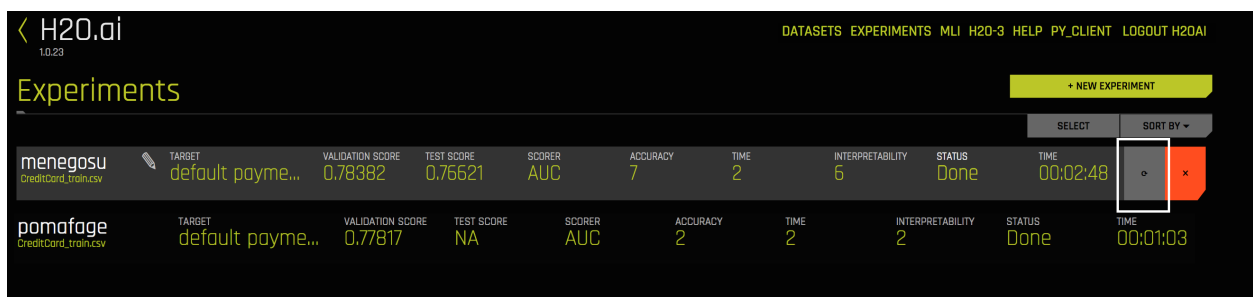


Click this link to open the Experiments page. From this page, you can rename an experiment, view previous experiments, begin a new experiment, re-run an experiment, and delete an experiment.

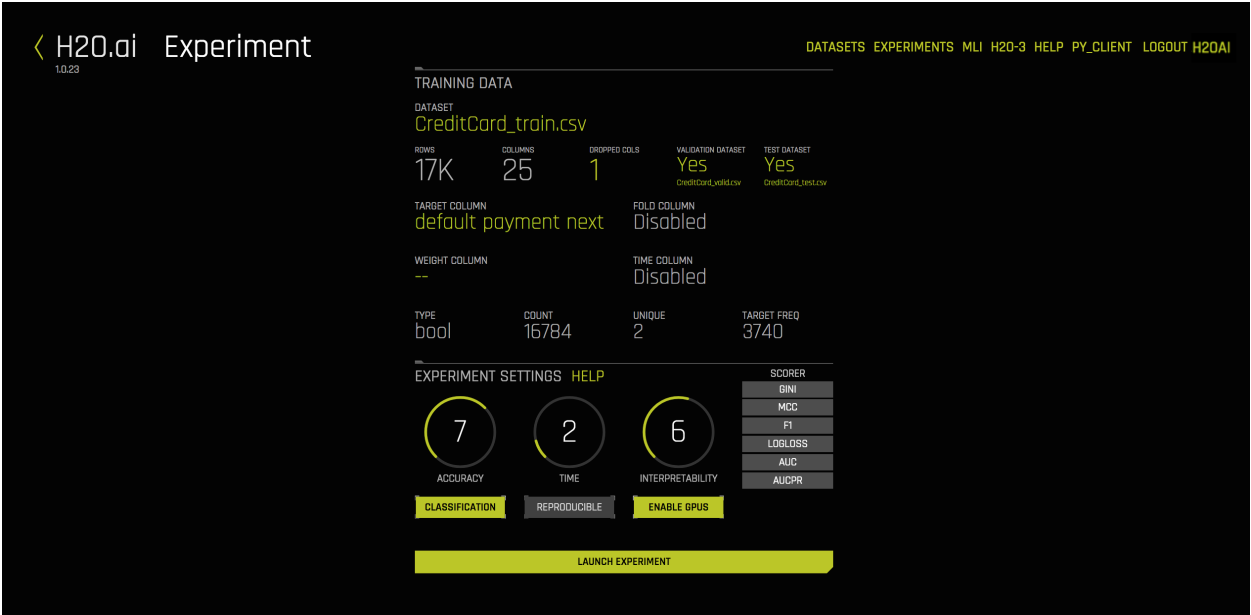


### 10.1 Rerunning Experiments

To rerun an experiment, or run a new experiment using an existing experiment's settings, hover over the experiment that you want to use. A “rerun” icon displays. Clicking this icon opens the selected experiment's settings.

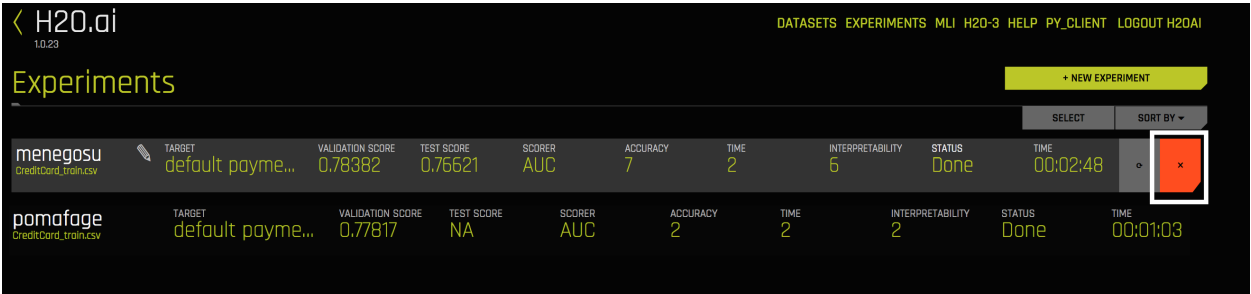


From the settings page, you can rerun the experiment using the original settings, or you can specify to use new data and/or specify different experiment settings.



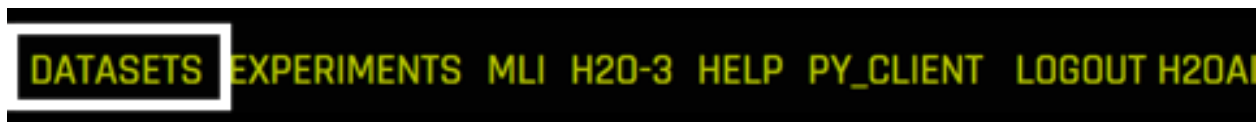
## 10.2 Deleting Experiments

To delete an experiment, hover over the experiment that you want to delete. An “X” option displays. Click this to delete the experiment. 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.

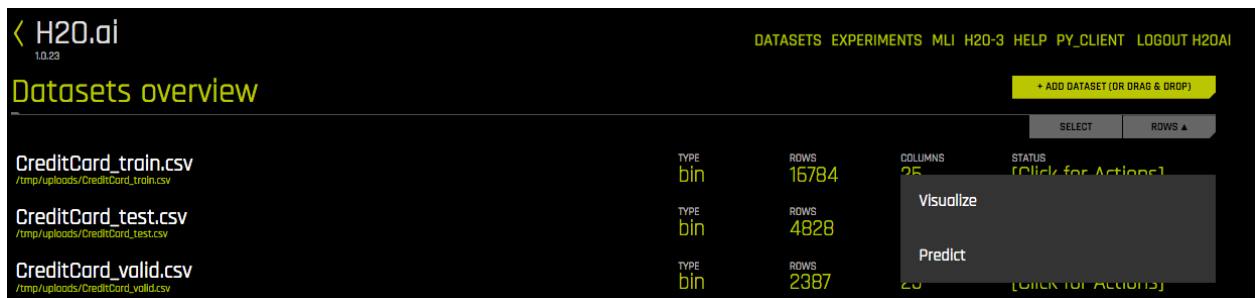


## VISUALIZING DATASETS

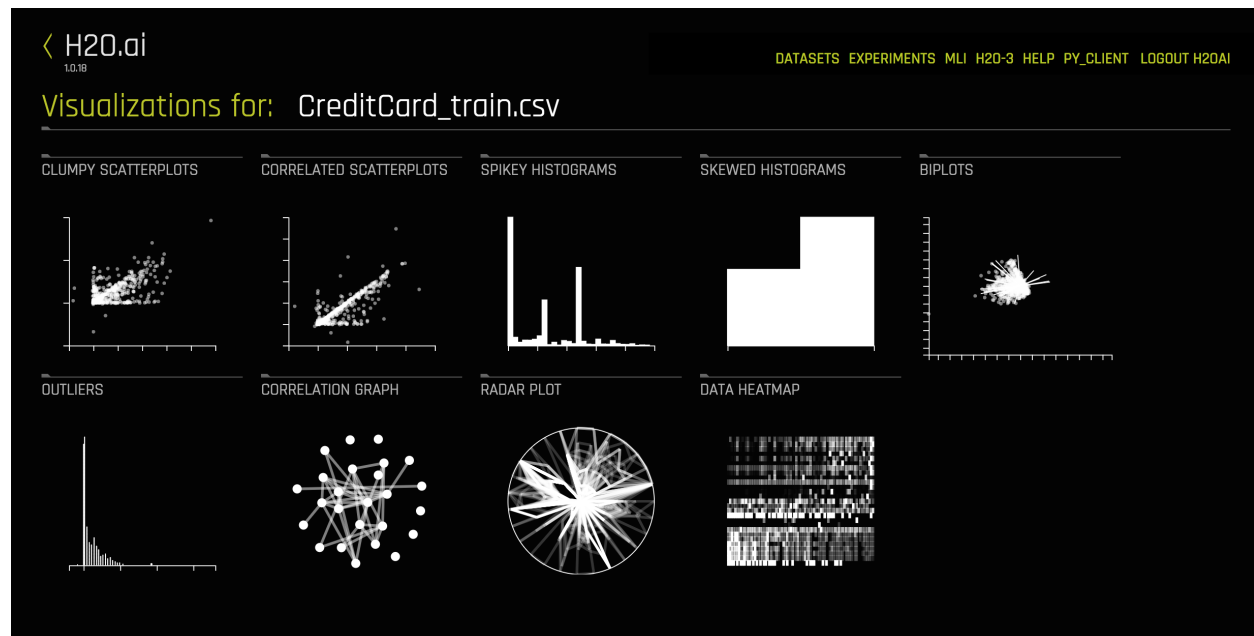
While viewing experiments, click the **Datasets** link in the upper-right corner.



The Datasets Overview page shows a list of the datasets that you've imported. Select the **[Click for Actions]** button beside the dataset that you want to view and then click **Visualize** from the submenu that appears.



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.



The following is a complete list of available graphs.

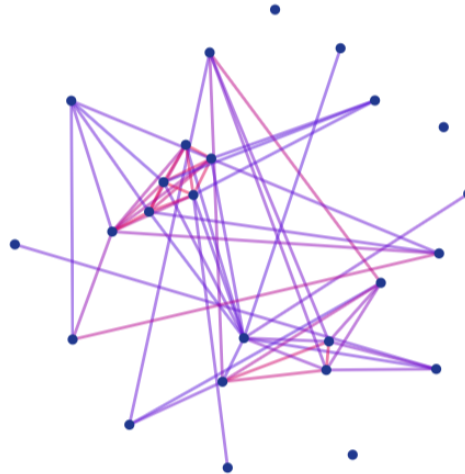
- Clumpy Scatterplots:** Clumpy scatterplots are 2D plots with evident clusters. These clusters are regions of high point density separated from other regions of points. The clusters can have many different shapes and are not necessarily circular. All possible scatterplots based on pairs of features (variables) are examined for clumpiness. The displayed plots are ranked according to the RUNT statistic. Note that the test for clumpiness is described in Hartigan, J. A. and Mohanty, S. (1992), “The RUNT test for multimodality,” *Journal of Classification*, 9, 63–70. The algorithm implemented here is described in Wilkinson, L., Anand, A., and Grossman, R. (2005), “Graph-theoretic Scagnostics,” in *Proceedings of the IEEE Information Visualization 2005*, pp. 157–164.
- 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 Pearson’s  $r$ . When modeling with these variables, you may want to leave out variables that are perfectly correlated with others.
- Unusual Scatterplots:** Unusual scatterplots are 2D plots with features not found in other 2D plots of the data. The algorithm implemented here is described in Wilkinson, L., Anand, A., and Grossman, R. (2005), “Graph-theoretic Scagnostics,” in *Proceedings of the IEEE Information Visualization 2005*, pp. 157–164. Nine scagnostics (“Outlying”, “Skewed”, “Clumpy”, “Sparse”, “Striated”, “Convex”, “Skinny”, “Stringy”, “Correlated”) are computed for all pairs of features. The scagnostics are then examined for outlying values of these scagnostics and the corresponding scatterplots are displayed.
- 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 spikey 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  $k_1$  categories and the second variable has  $k_2$  categories, then a  $k_1 \times k_2$  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  $(\text{chi-square} / n) / \min(k_1, k_2)$ , 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.
- **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 blue-ish and large values are red-ish. 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.
- **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.


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.

[DOWNLOAD](#)

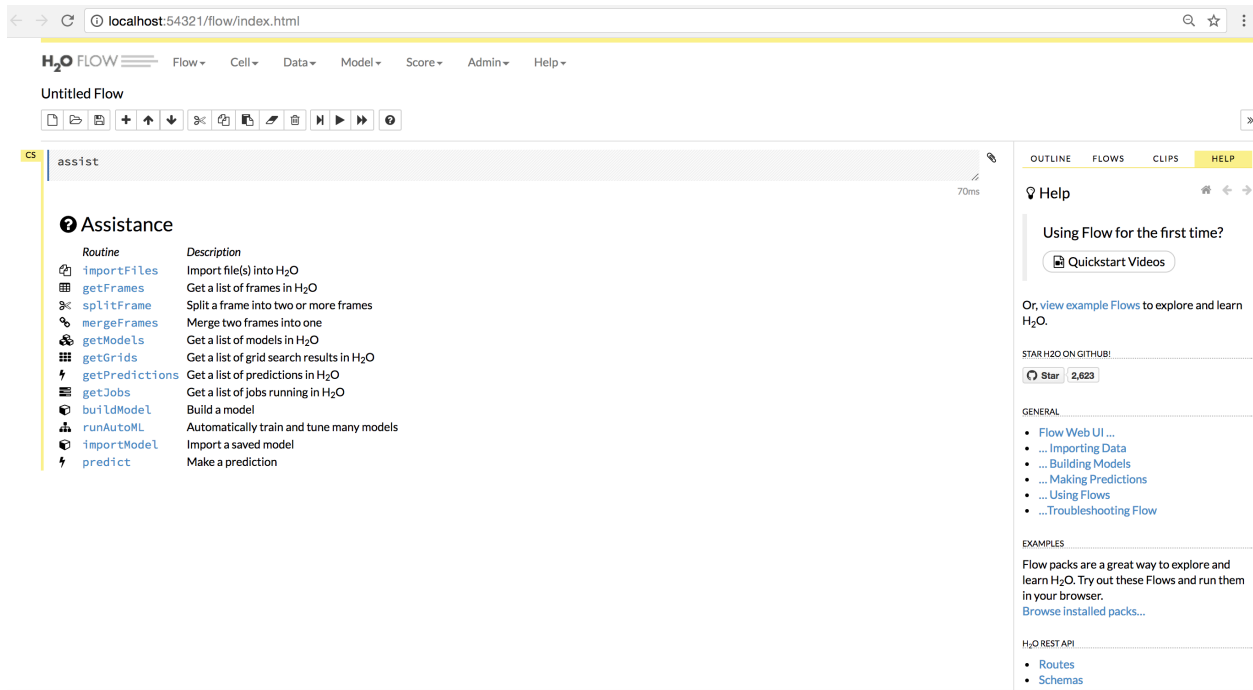
The correlation network graph is constructed from all pairwise squared correlations between variables (features). For continuous-continuous variable pairs, the statistic used is the squared Pearson correlation. For continuous-categorical variable pairs, the statistic is based on the squared intraclass 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  $k_1$  categories and the second variable has  $k_2$  categories, then a  $k_1 \times k_2$  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  $(\chi^2 \text{-square} / n) / \min(k_1 k_2)$ , 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.

## LAUNCHING H2O FLOW

If you opened port 54321 when starting Driverless AI, then you can launch H2O Flow from within Driverless AI. Click the H2O-3 link in the top menu.



This launches Flow on port 54321.







## DATA CONNECTORS

Driverless AI provides various data connectors for external data sources. Data sources are exposed in the form of the file systems. Each file system is prefixed by a unique prefix. For example, to reference data on HDFS, use the prefix `hdfs://`, or to reference data on S3, use `s3://`

Available file systems can be configured via the `enabled_file_systems` property. Note that each property must be prepended with `DRIVERLESS_AI_`. For example:

```
DRIVERLESS_AI_ENABLED_FILE_SYSTEMS="file,hdfs"
```

### 13.1 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.

#### 13.1.1 Description of Configuration Attributes

- `hdfs_auth_type`: Selects HDFS authentication. Available values are:
  - `principal`
  - `keytab`
  - `keytabimpersonation`
  - `noauth`
- `hdfs_core_site_xml_path`: The location of `core-site.xml` configuration file.

#### 13.1.2 HDFS Setup in Docker

The following examples demonstrate how to configure the HDFS connector when Driverless AI is running inside Docker.

##### HDFS with No Authentication

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 \
--add-host name.node:172.16.2.186 \
-e DRIVERLESS_AI_ENABLED_FILE_SYSTEMS="file,hdfs" \
-e DRIVERLESS_AI_HDFS_AUTH_TYPE='noauth' \
-p 12345:12345 \
--init -it --rm \
-v /tmp/dtmp:/tmp \
-v /tmp/dlog:/log \
-u $(id -u):$(id -g) \
opsh2oai/h2oai-runtime
```

## HDFS with Keytab-Based Authentication

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`).

```
# Docker instructions
nvidia-docker run \
-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>>' \
-p 12345:12345 \
--init -it --rm \
-v /tmp/dtmp:/tmp \
-v /tmp/dlog:/log \
-u $(id -u):$(id -g) \
opsh2oai/h2oai-runtime
```

## HDFS with Keytab-Based Impersonation

The example:

- 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`).
- Configures the DRIVERLESS\_AI\_HDFS\_APP\_LOGIN\_USER variable, which references a user who is being impersonated (usually in the form of `user@realm`).

```
# Docker instructions
nvidia-docker run \
-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='<<appuser@kerberosrealm>>' \
-e DRIVERLESS_AI_HDFS_APP_LOGIN_USER='<<thisuser@kerberosrealm>>' \
-p 12345:12345 \
--init -it --rm \
-v /tmp/dtmp:/tmp \
-v /tmp/dlog:/log \
```

(continues on next page)

(continued from previous page)

```
-u $(id -u):$(id -g) \
opsh2oai/h2oai-runtime
```

## 13.2 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.

### 13.2.1 Description of Configuration Attributes

- `aws_access_key_id`: The S3 access key ID
- `aws_secret_access_key`: The S3 access key

### 13.2.2 S3 Setup in Docker

The following examples demonstrate how to configure the S3 connector when Driverless AI is running inside Docker.

#### S3 with No Authentication

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 name node address, for example: `s3://name.node/datasets/iris.csv`.

```
nvidia-docker run \
--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 \
-u $(id -u):$(id -g) \
opsh2oai/h2oai-runtime
```

#### S3 with Authentication

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 name node address, for example: `s3://name.node/datasets/iris.csv`.

```
nvidia-docker run \
--add-host name.node:172.16.2.186 \
-e DRIVERLESS_AI_ENABLED_FILE_SYSTEMS="file,s3" \
-e DRIVERLESS_AI_AWS_ACCESS_KEY_ID="<access_key_id>" \
-e DRIVERLESS_AI_AWS_SECRET_ACCESS_KEY="<access_key>" \
-p 12345:12345 \
--init -it --rm \
-v /tmp/dtmp:/tmp \
-v /tmp/dlog:/log \
```

(continues on next page)

(continued from previous page)

```
-u $(id -u):$(id -g) \  
opsh2oai/h2oai-runtime
```

## THE CONFIG.TOML FILE

Rather than passing individual parameters when starting Driverless AI, admins can instead reference a config.toml file. This file includes all possible configuration options that would otherwise be specified in the `nvidia-docker run` command. Simply place this file in a folder on the container (for example, in /tmp), then set the desired environment variables.

After all of the environment variables are set, start Driverless AI using the following command:

```
nvidia-docker run \  
  --rm \  
  -u `id -u`:`id -g` \  
  -e DRIVERLESS_AI_CONFIG_FILE_PATH="tmp/config.toml" \  
  -v `pwd`/data:/data \  
  -v `pwd`/log:/log \  
  -v `pwd`/license:/license \  
  -v `pwd`/tmp:/tmp \  
  opsh2oai/h2oai-runtime
```

### 14.1 Sample config.toml File

```
# -----  
#                               DRIVERLESS AI CONFIGURATION FILE  
#  
#  
# This file is authored in TOML (see https://github.com/toml-lang/toml)  
#  
# The variables in this file can be overridden by corresponding environment  
#   variables named DRIVERLESS_AI_* (e.g. "max_cores" can be overridden by  
#   the environment variable "DRIVERLESS_AI_MAX_CORES").  
#  
# -----  
  
# IP address and port for Driverless AI HTTP server.  
ip = "127.0.0.1"  
port = 12345  
  
# Max number of CPU cores to use per experiment. Set to <= 0 to use all cores.  
max_cores = 0  
  
# Number of GPUs to use per model training task. Set to -1 for all GPUs.  
# Currently n_gpus!=1 disables GPU locking, so is only recommended for single-  
→ experiments and single users.
```

(continues on next page)

(continued from previous page)

```

# Ignored if GPUs disabled or no GPUs on system.
num_gpus = 1

# Which gpu_id to start with
# If use CUDA_VISIBLE_DEVICES=... to control GPUs, gpu_id=0 is still the first in
↳that list of devices.
# E.g. if CUDA_VISIBLE_DEVICES="4,5" then gpu_id_start=0 will refer to the device #4.
gpu_id_start = 0

# Maximum number of workers for DriverlessAI server pool (only 1 needed currently)
max_workers = 1

# Minimum amount of disk space in GB needed to run experiments.
# Experiments will fail if this limit is crossed.
disk_limit_gb = 5

# Minimum amount of system memory in GB needed to start experiments
memory_limit_gb = 5

# IP address and port of process proxy.
process_server_ip = "127.0.0.1"
process_server_port = 8080

# IP address and port of H2O instance.
h2o_ip = "127.0.0.1"
h2o_port = 54321

# Data directory. All application data and files related datasets and experiments
# are stored in this directory.
data_directory = "./tmp"

# Start HTTP server in debug mode (DO NOT enable in production).
debug = false

# Whether to run quick performance benchmark at start of application and each
↳experiment
enable_benchmark = false

# Minimum number of rows needed to run experiments (values lower than 100 might not
↳work)
min_num_rows = 100

# Internal threshold for number of rows to trigger certain statistical techniques to
↳increase statistical fidelity
statistical_threshold_num_rows_small = 10000

# Internal threshold for number of rows to trigger certain statistical techniques
↳that can speed up modeling
statistical_threshold_num_rows_large = 1000000

# Maximum number of columns
max_cols = 10000

# Threshold of rows * columns for which GPUs are disabled for speed purposes
gpu_small_data_size = 100000

# Maximum number of uniques allowed in fold column

```

(continues on next page)

(continued from previous page)

```

max_fold_uniques = 100000

# Maximum number of classes
max_num_classes = 100

# Minimum allowed seconds for time column
min_time_value = 5e8 # ~ > 1986

# Minimum number of rows above which try to detect time series
min_rows_detected_time = 10000

# relative standard deviation of hold-out score below which early stopping is
↳ triggered for accuracy~5
stop_early_rel_std = 1e-3

# Variable importance below which feature is dropped (with possible replacement found
↳ that is better)
# This also sets overall scale for lower interpretability thresholds
varimp_threshold_at_interpretability_10 = 0.05

# Maximum number of GBM trees (early-stopping usually chooses much less)
max_ntrees = 2000

# Authentication
#   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 against user against the operating
↳ system
#   ldap : Accepts user id and password, Validates against an ldap server, look for
↳ additional settings under LDAP settings
authentication_method = "unvalidated"

# LDAP Settings
ldap_server = ""
ldap_port = ""
ldap_dc = ""

# Supported file formats (file name endings must match for files to show up in file
↳ browser): a comma separated list
supported_file_types = "csv, tsv, txt, dat, tgz, zip, xz, xls, xlsx"

# File System Support
# Format: "file_system_1, file_system_2, file_system_3"
# Allowed file systems:
#   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

enabled_file_systems = "file, hdfs, s3"

# Configurations for a HDFS data source
# Path of hdfs coresite.xml
core_site_xml_path = ""
# Path of the principal key tab file
key_tab_path = ""

```

(continues on next page)

(continued from previous page)

```
# 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 = "noauth"

# Kerberos app principal user (recommended)
hdfs_app_principal_user = ""
# Specify the user id of the current user here as user@realm
hdfs_app_login_user = ""
#
hdfs_app_jvm_args = ""

# AWS authentication settings
#   True : Authenticated connection
#   False : Unverified connection
aws_auth = "False"

# S3 Connector credentials
aws_access_key_id = ""
aws_secret_access_key = ""
```



### How can I change my username and password?

The username and password is 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.

### How can I upgrade to a newer version of Driverless AI?

Driverless AI provides the following set of commands that can you can run on the Linux machine that's running Driverless AI. Note that these commands only work on Linux environments. These commands are not available for Mac and Windows installations.

```
h2oai stop      (Stop all instances)
h2oai start     (Start an instance)
h2oai restart   (Restart instance)
h2oai clean     (Removes old containers)
h2oai purge     (Removes old containers and purges tmp and log)
h2oai upgrade   (Upgrades image to Developer latest *unstable*)
h2oai update    (Upgrades image to latest Release)
h2oai ssh       (Attaches to running docker)
h2oai log       (Tails the running server log)
h2oai jupyter   (Fetch the Jupyter URL with token)
```

To upgrade to the latest version, stop Driverless AI, then run the `h2oai update` command.

### What kind of authentication is supported in Driverless AI?

Driverless AI supports LDAP and PAM authentication. This 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 [Data Connectors](#) and [The Config.toml File](#) for more information.) Examples for enabling PAM and LDAP authentication will be documented in the next release.

```
# Authentication
# 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 against user against the
↳operating system
# ldap : Accepts user id and password, Validates against an ldap server,
↳look for additional settings under LDAP settings
authentication_method = "unvalidated"

#Ldap Settings
ldap_server = ""
```

(continues on next page)

(continued from previous page)

```

ldap_port = ""
ldap_dc = ""

# Configurations for a HDFS data source
# Path of hdfs coresite.xml
core_site_xml_path = ""
# Path of the dai principal key tab file
key_tab_path = ""

# HDFS connector
# Auth type can be Principal/keytab/keytabPrincipal
# Specify HDFS Auth Type, allowed options are described below
#   Principal : Authenticate with HDFS with a principal user
#   Keytab : Authenticate with a Key tab, preferably the keytab is created
#             ↪for the dai application
#   Impersonate with Keytab : Login with Impersonation using a dai keytab
hdfs_auth_type = "keytab"
# DAI recommends a user to be created in kerberos for the driverless ai
#             ↪application
# specify the kerberos app principal user below
hdfs_app_principal_user = ""
# Specify the user id of the current user here as user@realm
hdfs_app_login_user = ""
#
hdfs_app_jvm_args = ""

#S3 Connector credentials
aws_access_key_id = ""
aws_secret_access_key = ""

```

### Can I set up SSL on Driverless AI?

Yes, you can set up HTTPS/SSL on Driverless AI running in an AWS environment. HTTPS/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 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;
}

```

(continues on next page)

(continued from previous page)

```

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>.

### Is there a file size limit for datasets?

The file size for datasets is limited by GPU memory, but we continue to make optimizations for getting more data into an experiment.

### How does Driverless AI detect the ID column?

The ID column logic is that the column is named 'id', 'Id', 'ID' or 'iD' exactly. (It does not check the number of unique values.) For now, if you want to ensure that your ID column is downloaded with the predictions, then you would want to name it one of those names.

### 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 clustering, we impute missing values; for frequency encoding, we count the number of rows that have a certain missing feature.

### 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 dataset, Driverless AI will do the same on the test dataset.

### Which algorithms are used in Driverless AI?

Currently we only use XGBoost GPU for building models and testing the engineered features. Additional GPU algorithms will be added at a later date.

### 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 close to iid

- Internal holdouts are used for parameter tuning
- 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 drift in the data
  - No training data wasted during training since training data not used for parameter tuning
  - Entire validation set used for parameter tuning
  - No CV possible, since we explicitly do not want to overfit on the training data

**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, ensuring that the model build will stop when it ceases to improve. And additional steps to prevent overfitting include checking for IID and avoiding leakage during feature engineering.

A blog post describing Driverless AI overfitting protection in greater detail is currently in development.

### What can I do if my training and validation data points are not identical?

If you feel that training and validation data points are not identical, you can optionally provide an observation weights column (such as exponential weighting in time, different weights for valid vs train, etc.). All of our algorithms and metrics in Driverless AI support observation weights.

### 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, since we do up to 5-fold CV.)

### Where can I get details of the various transformations performed in an experiment?

Inside the `/tmp` folder, you will see a folder with your experiment ID, and within that folder is a `*logs_<experiment>.zip` file. This zip file includes summary information, log information, and a `gene_summary.txt` file with details of the transformations used in the experiment.

### How can I download the predictions onto the machine where Driverless AI is running?

When you select **Score on Another Dataset**, the predictions will be automatically downloaded to the machine where Driverless AI is running. It will be saved in the following locations:

- Training Data Predictions: `tmp/experiment_name/train_preds.csv`
- Testing Data Predictions: `tmp/experiment_name/test_preds.csv`
- New Data Predictions: `tmp/experiment_name/automatically_generated_name`. Note that the automatically generated name will match the name of the file downloaded to your local computer.

## APPENDIX A: DRIVERLESS AI TRANSFORMATIONS

Transformations in Driverless AI are applied to columns in the data. The transformers create the engineered features. Driverless AI provides the following transformers:

- Filter Transformer

The Filter Transformer counts each numeric value in the dataset.

- Frequent Transformer

The Frequent Transformer counts each categorical value in the dataset. This count can be either the raw count or the normalized count.

- Bulk Interactions Transformer

The Bulk Interactions Transformer will add, divide, multiply, and subtract two columns in the data.

- Truncated SVD Numeric Transformer

Truncated SVD trains on a selected numeric of columns in the data. The components of the truncated SVD will be new features.

- Cross Validation Target Encoding

Cross validation target encoding is done on a categorical column.

- Cross Validation Categorical to Numeric Encoding

This transformer converts a categorical column to a numeric column. Cross validation target encoding is done on the categorical column.

- Dates Transformer

The Dates Transformer retrieves any date values, including:

- Year
- Quarter
- Month
- Day
- Day of year
- Week
- Week day
- Hour
- Minute
- Second

- Date Polar Transformer

The Date Polar Transformer expands the date using polar coordinates. The Date Transformer will only expand the date into different units, for example month. This does not capture the similarity between the months December and January (12 and 1) or the hours 23 and 0. The polar coordinates capture the similarities between these cases by representing the unit of the date as a point in a cycle. For example, the polar coordinates of: get minute in hour, would be the minute hand position on a clock.

- Text Transformer

The Text Transform transforms a text column using TFIDF (term frequency-inverse document frequency) or count (count of the word). This may be followed by dimensionality reduction using truncated SVD.

- Numeric to Categorical Target Encoding Transformer

This transformer converts a numeric column to categorical by binning. Cross validation target encoding is done on the binned column.

- Cluster Target Encoding Transformer

Selected columns in the data are clustered, and target encoding is done on the cluster ID.

- Cluster Distance Transformer

Selected columns in the data are clustered, and the distance to a chosen cluster center is calculated.

- Weight of Evidence

Creates likelihood type of features using the Weights Of Evidence (WOE) transformation method. The weight of evidence tells the predictive power of an independent variable in relation to the dependent variable, for example, the measurement of good customers in relations to bad customers.

$$WOE = \ln \left( \frac{\text{Distribution of Goods}}{\text{Distribution of Bads}} \right)$$

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/>.

- Numeric To Categorical Weight of Evidence Transformer

This transformer converts a numeric column to categorical by binning and then creates the likelihood type of features using the WOE transformation method.

## 16.1 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

### 16.1.1 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.

### 16.1.2 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.

### 16.1.3 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.

### 16.1.4 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.

### 16.1.5 Date Polar Transformer

- get hour in day, get minute in hour, get day in month, get day in year, get quarter in year, get month in year, get week in year, get week day in week

Date Built	Square Footage	Num Beds	Num Baths	State	Price	Date-Built_MonthInYear_x	Date-Built_MonthInYear_y
01/01/1920	1700	3	2	NY	700,000	0.5	1

The polar coordinates of the month January in year is (0.5, 1). This allows the model to catch the similarities between January and December. This information was not captured in the simple Date Transformer.

### 16.1.6 Text Transformer

- transform text column using methods: TFIDF or count (count of the word)
- this may be followed by dimensionality reduction using truncated SVD

### 16.1.7 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.

### 16.1.8 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.

### 16.1.9 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.

### 16.1.10 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.

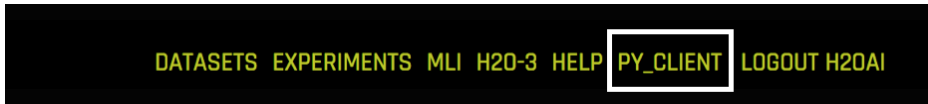


## APPENDIX B: USING THE DRIVERLESS AI PYTHON CLIENT

This section describes how to run Driverless AI using the Python client.

**Notes:**

- This is an early release of the Driverless AI Python client.
- Python 3.6 is the only supported version.
- You must install the `h2oai_client` wheel to your local Python. This is available from the **PY\_CLIENT** link in the top menu of the UI.



### 17.1 Running an Experiment

1. After the `h2oai_client` wheel is installed, import the required modules and log in.

```
import h2oai_client
import numpy as np
import pandas as pd
import requests
import math
from h2oai_client import Client, ModelParameters, InterpretParameters

address = 'http://ip_where_driverless_is_running:12345'
username = 'username'
password = 'password'
h2oai = Client(address = address, username = username, password = password)
# Be sure to use the same credentials that you use when signing in through
↳ the GUI
```

2. Upload training, testing, and validation datasets from the Driverless AI **/data** folder. The validation and testing dataset are optional. This example shows how to add a validation dataset, but the experiment will only specify training and testing datasets.

```
train_path = '/data/CreditCard/CreditCard-train.csv'
test_path = '/data/CreditCard/CreditCard-test.csv'
valid_path = '/data/CreditCard/CreditCard-valid.csv'

train = h2oai.create_dataset_sync(train_path)
```

(continues on next page)

(continued from previous page)

```
test = h2oai.create_dataset_sync(test_path)
valid = h2oai.create_dataset_sync(valid_path)
```

### 3. Set experiment parameters:

The next step is to set parameters for the 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.
- **Time Column:** The column that provides a time order, if applicable. If “AUTO”, Driverless AI will auto-detect a potential time order. If “OFF”, auto-detection is disabled.

For this example, we will be predicting **default payment next month**. We can set the parameters by hand or let Driverless AI infer the parameters and override any we disagree with. In this case, we will let Driverless AI suggest the best parameters for our experiment.

```
# let Driverless suggest parameters for experiment
target="default payment next month"
params = h2oai.get_experiment_tuning_suggestion(dataset_key = train.key,
→target_col = target)

params.dump()
{'accuracy': 7,
 'cols_to_drop': [],
 'dataset_key': 'corulege',
 'enable_gpus': True,
 'fold_col': '',
 'interpretability': 7,
 'is_classification': True,
 'scorer': '',
 'seed': False,
 'target_col': 'default payment next month',
 'testset_key': '',
 'time': 3,
 'time_col': '',
 'validset_key': '',
 'weight_col': ''}
```

Driverless AI has found that the best parameters are to set accuracy = 7, time = 3, and interpretability = 7. We will add our test data to the parameters, set the scorer to AUC, and add a seed to make the experiment reproducible.

```
params.testset_key = test.key
params.scorer = "AUC"
params.seed = 1234
```

### 4. Launch the experiment to run feature engineering and final model training. In addition to the settings previously defined, be sure to also specify the imported training dataset. Adding a test dataset and a validation dataset is optional. This example will not use a validation dataset.

```
experiment = h2oai.start_experiment_sync(params)
```

5. Examine 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.

```
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: 0.802
```

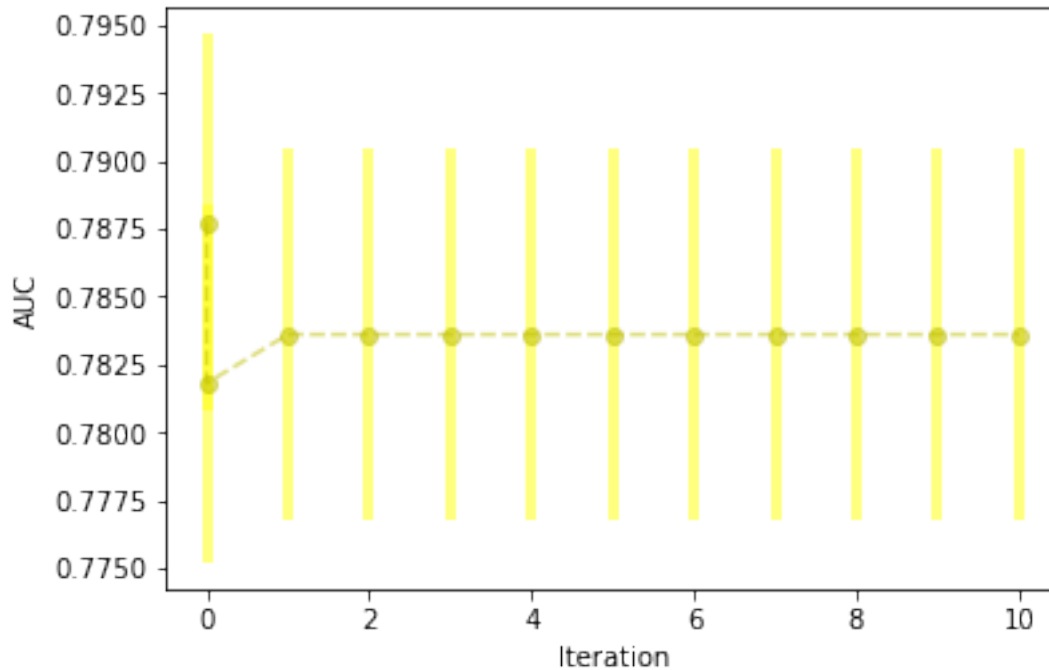
The experiment object also contains the scores calculated for each iteration on bootstrapped samples on the validation data. In the iteration graph in the UI, we can see the mean performance for the best model (yellow dot) and +/- 1 standard deviation of the best model performance (yellow bar).

This information is saved in the experiment object.

```
import matplotlib.pyplot as plt

iterations = list(map(lambda iteration: iteration.iteration, experiment.
↪iteration_data))
scores_mean = list(map(lambda iteration: iteration.score_mean, experiment.
↪iteration_data))
scores_sd = list(map(lambda iteration: iteration.score_sd, experiment.
↪iteration_data))

plt.figure()
plt.errorbar(iterations, scores_mean, yerr=scores_sd, color = "y",
             ecolord='yellow', fmt = '--o', elinewidth = 4, alpha = 0.5)
plt.xlabel("Iteration")
plt.ylabel(scorer_str)
plt.show();
```



6. 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.

```
h2oai.download(src_path=experiment.test_predictions_path, dest_dir=".")
'./test_preds.csv'

test_preds = pd.read_csv("./test_preds.csv")
test_preds.head()
```

	ID	default payment next month.1
0	24001	0.629740
1	24002	0.147111
2	24003	0.058483
3	24004	0.608169
4	24005	0.128982

We can also download and examine the summary of the experiment and feature importance for the final model.

```
# Download Summary
import subprocess
summary_path = h2oai.download(src_path=experiment.summary_path, dest_dir=".")
dir_path = "./h2oai_experiment_summary_" + experiment.key
subprocess.call(['unzip', '-o', summary_path, '-d', dir_path], shell=False)
```

The table below shows the feature name, its relative importance, and a description. Some features will be engineered by Driverless AI and some can be the original feature.

```
# View Features
features = pd.read_table(dir_path + "/features.txt", sep=',',
↳ skipinitialspace=True)
features.head(n = 10)
```

	Relative Importance	Feature	Description
0	1.000000	21_PAY_0	PAY_0 (original)
1	0.279240	22_PAY_2	PAY_2 (original)
2	0.114620	27_PAY_AMT1	PAY_AMT1 (original)
3	0.113840	19_LIMIT_BAL	LIMIT_BAL (original)
4	0.100850	12_BILL_AMT1	BILL_AMT1 (original)
5	0.095101	4_Freq:PAY_0	Encoding of categorical levels of feature(s) [...
6	0.094240	30_PAY_AMT4	PAY_AMT4 (original)
7	0.092937	28_PAY_AMT2	PAY_AMT2 (original)
8	0.087147	29_PAY_AMT3	PAY_AMT3 (original)
9	0.077035	8_Freq:PAY_5	Encoding of categorical levels of feature(s) [...

## 17.2 Access an Experiment Object that was Run through the Web UI

It is also possible to use the Python API to examine an experiment that was started through the Web UI using the experiment key.

You can get a pointer to the experiment by referencing the experiment key in the Web UI.

```
# Get a list of experiments
experiment_list = list(map(lambda x: x.key, h2oai.list_models(offset=0, limit=100)))
experiment_list
['fapudage', 'cunegacu', 'hamasida']

# Get pointer to experiment
experiment = h2oai.get_model_job(experiment_list[0]).entity
```

## 17.3 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. Its path can be found with `experiment.test_predictions_path`.

The following shows the predicted probability of default for each record in the test.

```
prediction = h2oai.make_prediction_sync(experiment.key, test_path, output_margin = False, pred_contribs = False)
pred_path = h2oai.download(prediction.predictions_csv_path, '.')
pred_table = pd.read_csv(pred_path)
pred_table.head()
```

	ID	default payment next month.1
0	24001	0.629740
1	24002	0.147111
2	24003	0.058483
3	24004	0.608169
4	24005	0.128982

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.

```
prediction_contributions = h2oai.make_prediction_sync(experiment.key, test_path, 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()
```

	ID	contrib_1_Freq:AGE	contrib_2_Freq:EDUCATION	contrib_3_Freq:LIMIT_BAL	contrib_4_Freq:PAY_0	contrib_5_Freq:
0	24001	0.001003	0.021604	-0.015781	0.180856	-0.022679
1	24002	-0.029895	0.021343	0.013078	-0.094722	-0.027017
2	24003	-0.022149	0.017400	-0.007056	-0.142536	-0.032671
3	24004	-0.036485	-0.525060	-0.026318	0.140338	0.104220
4	24005	-0.043551	-0.003614	-0.017633	-0.123395	-0.020288

5 rows x 29 columns



We will examine the contributions for our first record more closely.

```
contrib = pd.DataFrame(pred_contributions_table.iloc[0][1:], columns = ["contribution", "abs_contribution"])
contrib["abs_contribution"] = contrib.contribution.abs()
contrib.sort_values(by="abs_contribution", ascending=False)[["contribution"]].head()
```

	contribution
contrib_17_PAY_0	1.385679
contrib_bias	-1.325654
contrib_18_PAY_2	0.311258
contrib_4_Freq:PAY_0	0.180856
contrib_16_LIMIT_BAL	0.120779

This customer's PAY\_0 value had the greatest impact on their prediction. Because the contribution is positive, we know that it increases the prediction that they will default.

## 17.4 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. You can run model interpretation on raw data and on external model predictions.

### 17.4.1 Run Model Interpretation on 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. If you set `use_raw_features` to `False`, the model will be interpreted using the features used in the final model (raw and engineered).

Note that you can also specify the number of cross-validation folds to use in K-LIME. This defaults to 0.

```
mli_experiment = h2oai.run_interpretation_sync(InterpretParameters
    (dai_model_key=experiment.key,
     dataset_key=train.key,
     target_col=target,
     prediction_col = '',
     use_raw_features=True, # show interpretation based on the_
     ↪original columns
     klime_cluster_col='',
     nfolds = 0 # number of folds used for k-lime
    ))
```

You can also see the list of interpretations using the Python client.

```
# Get list of interpretations
mli_list = list(map(lambda x: x.key, h2oai.list_interpretations(offset=0, limit=100)))
mli_list

['wekature', 'bawamoci']
```

## 17.4.2 Run Model Interpretation on External Model Predictions

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. 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.

```
# Dataset must be located where python client is running
train_pd = pd.read_csv(train_path)

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])
gbm_model = 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,
                                       presort='auto', random_state=10, subsample=1.0, verbose=0,
                                       warm_start=False)

predictions = gbm_model.predict_proba(train_pd[predictors])
predictions[0:5]
array([[ 0.38111179,  0.61888821],
       [ 0.44396186,  0.55603814],
       [ 0.91738328,  0.08261672],
       [ 0.88780536,  0.11219464],
       [ 0.80028008,  0.19971992]])
```

### Interpret on External Predictions

Now that we have the predictions from our scikit-learn GBM model, we can call Driverless AI's `h2o_ai.run_interpretation_sync` to create the interpretation screen.

```
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)
```

(continues on next page)

(continued from previous page)

```
train_gbm_pred = h2oai.upload_dataset(train_gbm_path)

mli_external = h2oai.run_interpretation_sync(
    InterpretParameters(dai_model_key="",
                       dataset_key=train_gbm_pred.strip("\\"),
                       target_col=target,
                       prediction_col = "p1",
                       use_raw_features=True, # not relevant since we are
↪ interpreting our external model
                       klime_cluster_col='',
                       nfolds = 0 # number of folds used for k-lime
    ))
```



## 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.
- 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.