H2O Steam

Preview Release

Table of Contents

Introduction

Steam is an "instant on" platform that streamlines the entire process of building and deploying applications. It is the industry's first data science hub that lets data scientists and developers collaboratively build, deploy, and refine predictive applications across large scale datasets. Data scientists can publish Python and R code as REST APIs and easily integrate with production applications.

This document describes how to start and use Steam and the Steam Scoring Service. Note that this document assumes that an admin has successfully installed and started Steam on a YARN edge node using the instructions provided in the *Steam Installation and Setup* guide.

> **Note**: Before you begin using Steam, be sure that your minimum version of H2O is 3.10.0.3. Earlier versions are not supported. If necessary, follow the instructions on the H2O Download page for your platform to upgrade H2O.

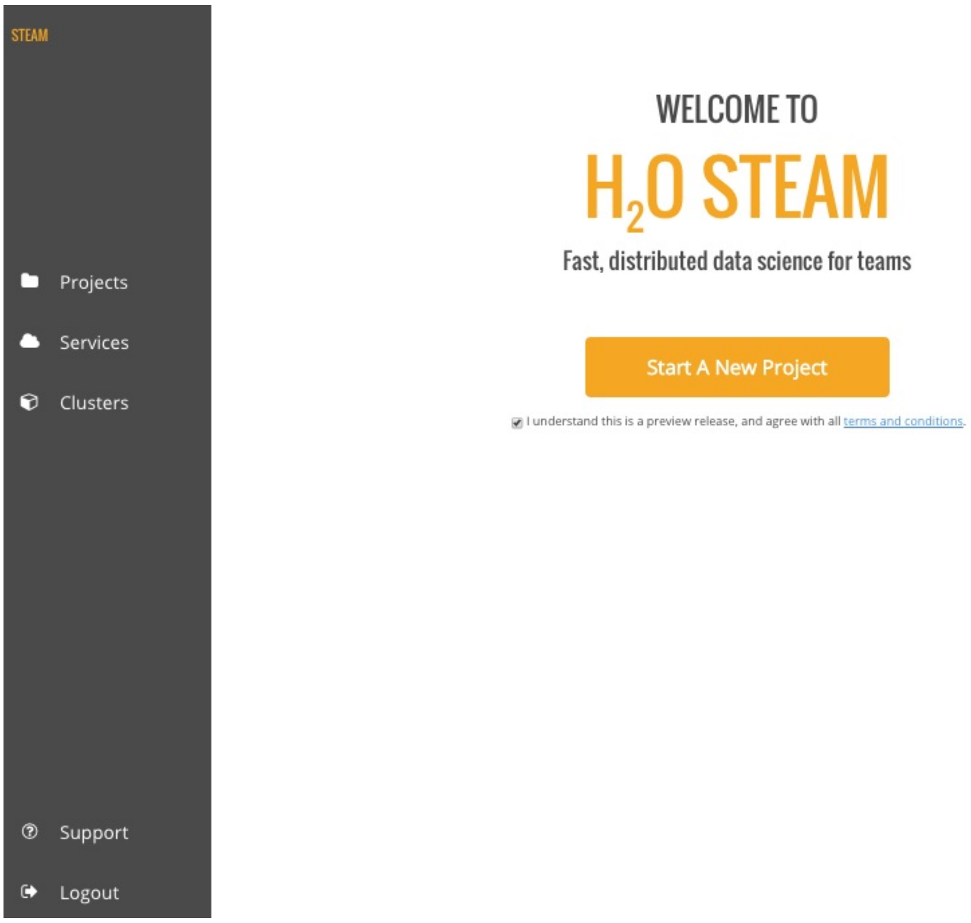
Refer to the following sections:

Logging in to Steam

In a Chrome web browser, navigate to the Steam web server using the login credentials provided by your admin and/or Steam superuser. This Steam web server is the server on which an admin has installed Steam (for example, http://192.16.2.182:9000). Contact your admin for the IP address and for credentials.

The Steam UI

The first time you log in to Steam, an empty Steam page will display, prompting you to start a project. Be sure to accept the terms and conditions in order to continue.


STEAM
Projects
Services
Clusters
Support
Logout
WELCOME TO
H2O STEAM
Fast, distributed data science for teams
Start A New Project
I understand this is a preview release, and agree with all terms and conditions.

The left navigation provides quick links for all the following:

- All projects available on the Steam cluster
- Deployed services, including pre-processing packages
- Cluster details
- E-mail link to Steam support at H2O
- Logging out

> **Note**: When Steam is started for the first time, no projects, services, or clusters will appear in the UI.

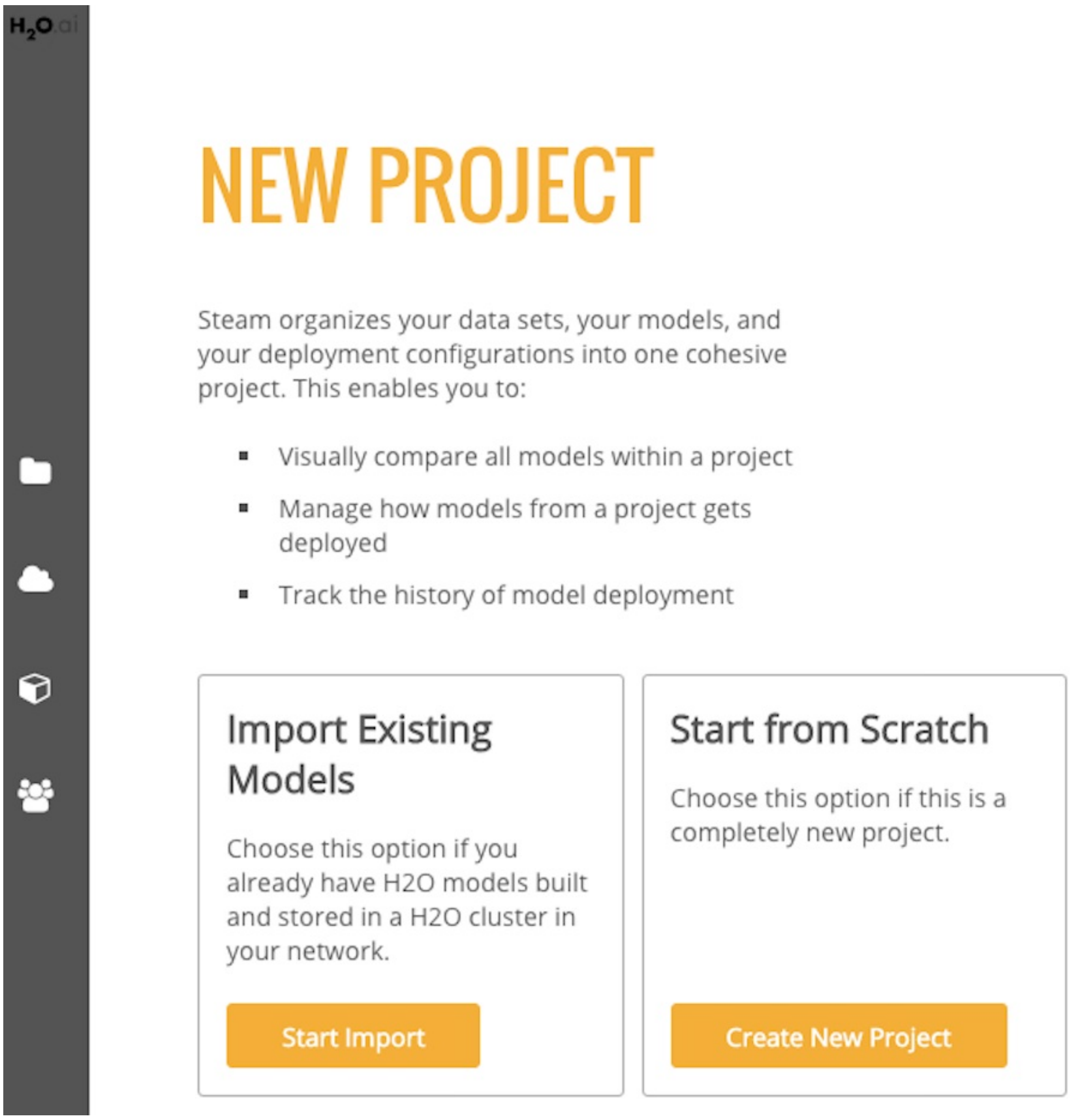
Accept the terms of this preview release, then click **Start a New Project**. This opens a page allowing you to start a new project from scratch or to begin importing models into your Steam environment. Refer to the next section for information about Steam projects.

Projects

Steam makes use of project-based machine learning. Whether you are trying to detect fraud or predict user retention, the datasets, models, and test results are stored and saved in the individual projects. All Steam users within your environment can access these projects and the files within them.

Creating a Project

1. To start a new project from scratch, click **Create New Project**.



NEW PROJECT

Steam organizes your data sets, your models, and your deployment configurations into one cohesive project. This enables you to:

- Visually compare all models within a project
- Manage how models from a project gets deployed
- Track the history of model deployment

Import Existing Models

Choose this option if you already have H2O models built and stored in a H2O cluster in your network.

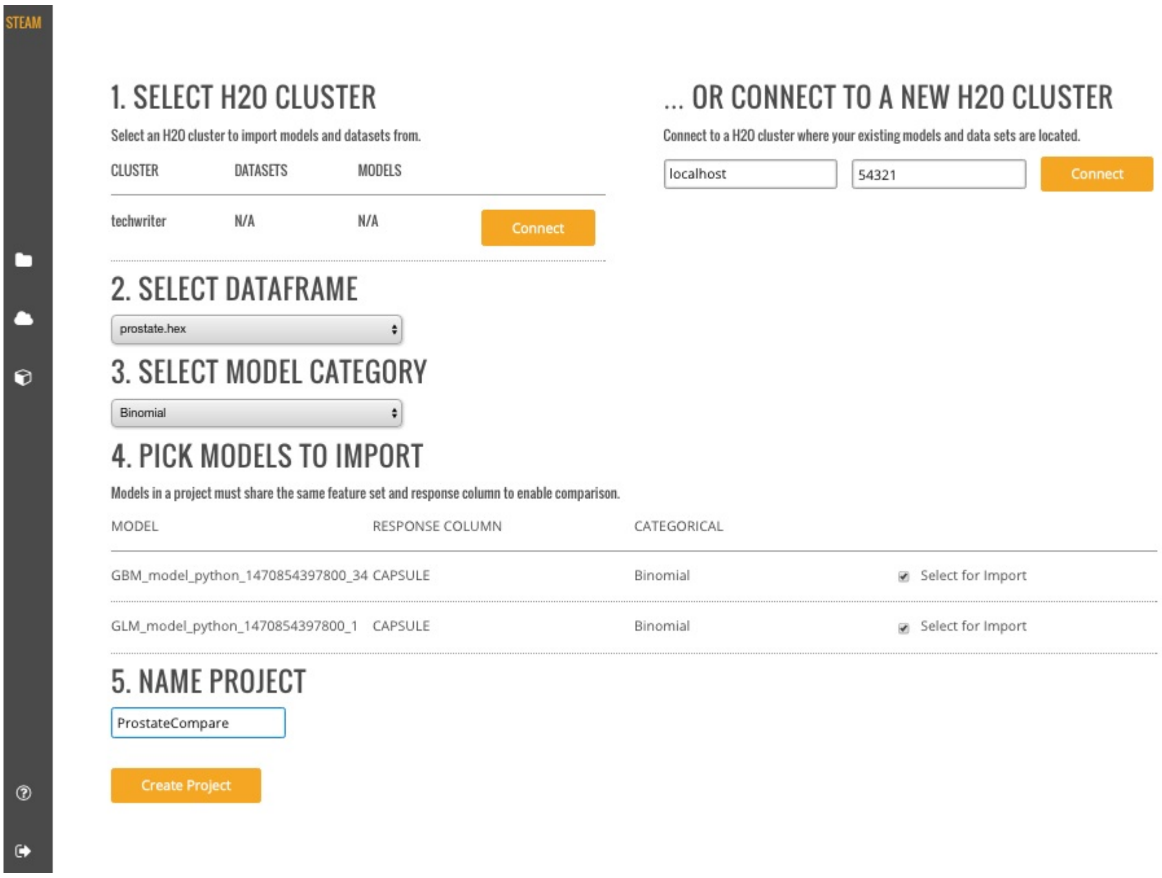
Start from Scratch

Choose this option if this is a completely new project.

2. When you first log in to Steam, the list of clusters will be empty. Enter the IP address of the cluster that is running H2O, then click **Connect**. Once connected, the current list of clusters will immediately populate with this cluster. Connect to this cluster to continue.

3. Select an available H2O frame from the Datasets dropdown, then select the Category. Note that these dropdowns are automatically populated with information from datasets that are available on the selected cluster. If no datasets are available, the the dataset list will be empty. For clusters that contain datasets, after a dataset is selected, a list of corresponding models will display.

4. Select the checkbox beside the model(s) to import into the Steam project. In this example, two models are available on the H2O cluster: one model built using GBM and one model built using GLM.
5. Specify a name for the project.


STEAM
1. SELECT H2O CLUSTER
Select an H2O cluster to import models and datasets from.
... OR CONNECT TO A NEW H2O CLUSTER
Connect to a H2O cluster where your existing models and data sets are located.
Connect
2. SELECT DATAFRAME
3. SELECT MODEL CATEGORY
4. PICK MODELS TO IMPORT
Models in a project must share the same feature set and response column to enable comparison.
5. NAME PROJECT
Create Project

CLUSTER	DATASETS	MODELS	
techwriter	N/A	N/A	Connect

localhost | 54321

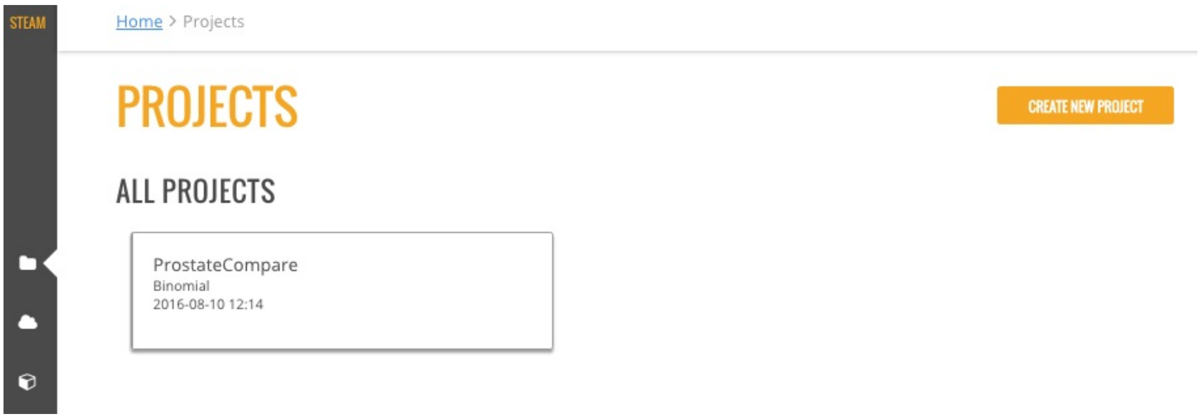
prostate.hex

Binomial

MODEL	RESPONSE COLUMN	CATEGORICAL	
GBM_model_python_1470854397800_34	CAPSULE	Binomial	✓ Select for Import
GLM_model_python_1470854397800_1	CAPSULE	Binomial	✓ Select for Import

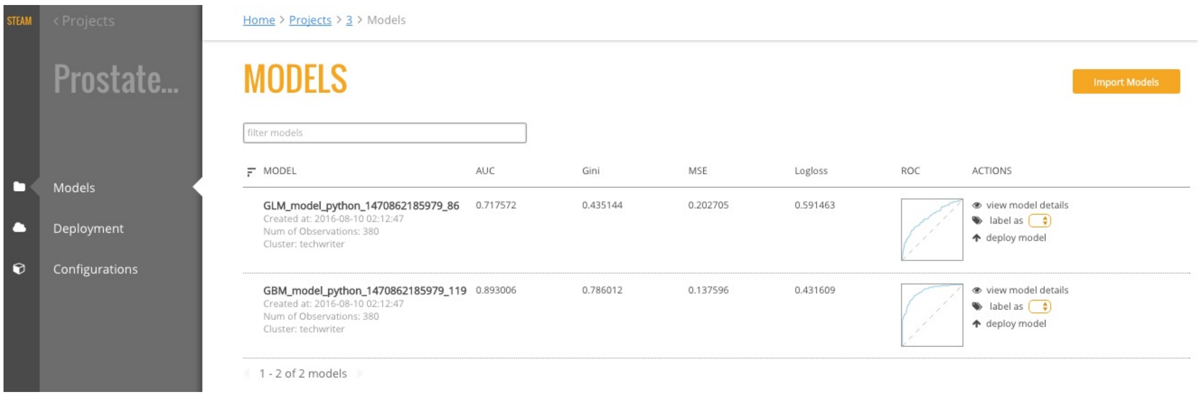
ProstateCompare

6. Click **Create Project** when you are done. Upon successful completion, the Models page will be populated with the model(s) that you added to your project, and the new project will be available on the **Projects** page.



ProstateCompare
Binomial
2016-08-10 12:14

7. On the **Projects** page, click on the newly created project. This opens a submenu allowing you to view the imported models, deployed models, and configurations specific to that project. Information about these topics is available in the sections that follow.



MODEL	AUC	Gini	MSE	Logloss	ROC	ACTIONS
GLM_model_python_1470862185979_86 Created at: 2016-08-10 02:12:47 Num of Observations: 380 Cluster: techwriter	0.717572	0.435144	0.202705	0.591463		view model details label as deploy model
GBM_model_python_1470862185979_119 Created at: 2016-08-10 02:12:47 Num of Observations: 380 Cluster: techwriter	0.893006	0.786012	0.137596	0.431609		view model details label as deploy model

1 - 2 of 2 models

Models

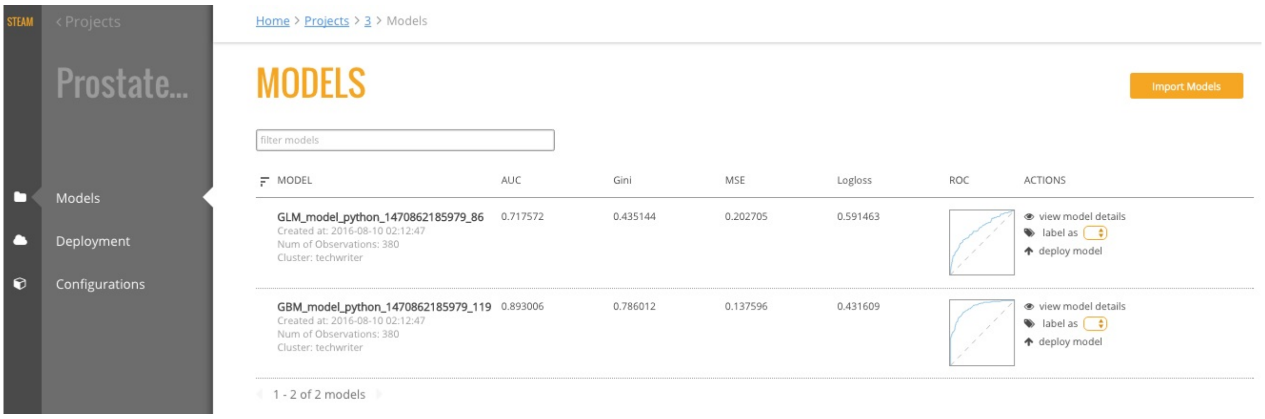
The **Models** page shows a list of all models included in a selected Project. This list also includes summary information for each model. This information varies based on whether the model is binomial or regresssion.

For binomial models, the following values will display on the Models page.

- AUC
- Gini
- MSE
- Logloss
- ROC

For regression models, the following values will display on the Models page.

- MRD
- MSE
- R^2



MODEL	AUC	Gini	MSE	Logloss	ROC	ACTIONS
GLM_model_python_1470862185979_86 Created at: 2016-08-10 02:12:47 Num of Observations: 380 Cluster: techwriter	0.717572	0.435144	0.202705	0.591463		view model details label as deploy model
GBM_model_python_1470862185979_119 Created at: 2016-08-10 02:12:47 Num of Observations: 380 Cluster: techwriter	0.893006	0.786012	0.137596	0.431609		view model details label as deploy model

1 - 2 of 2 models

You can perform the following actions directly from this page:

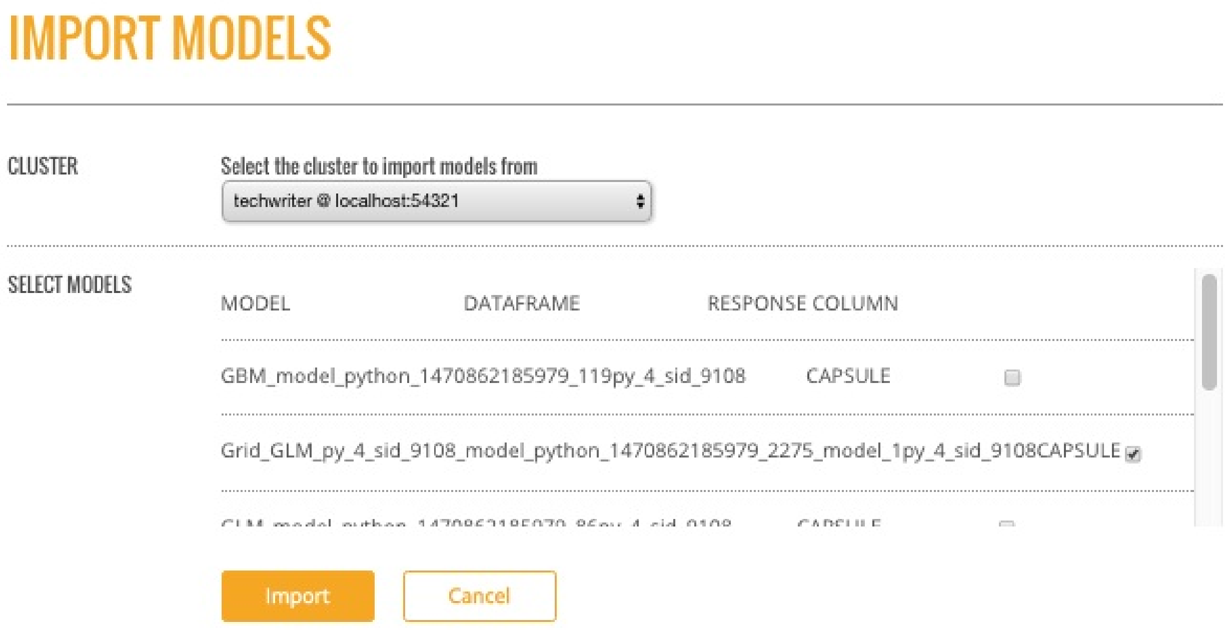
- Import a new model
- View model details and export the model as a java, jar, or war file
- Label a model (refer to Configurations for information on how to create labels)
- Deploy the model

> **Note**: The Models page lists models in alphabetical order and shows five models per page. If your project includes more than five models, use the forward and back arrows at the bottom of the page to view more models.

Importing Models

After models are added to an H2O cluster, they can be imported into an existing Steam project. In the upper-right corner of the Models page, click the **Import Models** button. This opens an Import Models popup form.

The Cluster dropdown automatically populates with a list H2O clusters. Specify the H2O cluster that has the models you want to import, then select the additional model or models that you want to add to the project.



IMPORT MODELS

CLUSTER

Select the cluster to import models from

techwriter @ localhost:54321

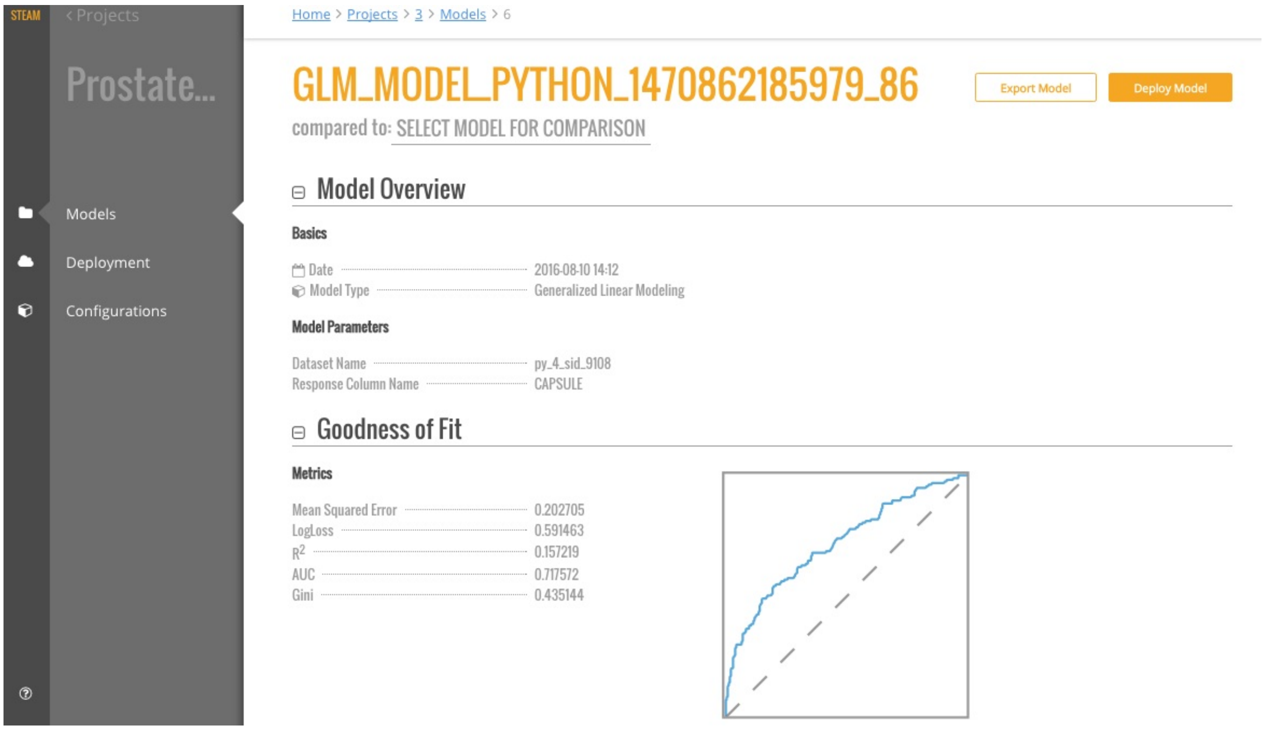
SELECT MODELS

MODEL	DATAFRAME	RESPONSE COLUMN	
GBM_model_python_1470862185979_119py_4_sid_9108		CAPSULE	☐
Grid_GLM_py_4_sid_9108_model_python_1470862185979_2275_model_1py_4_sid_9108		CAPSULE	☑
GLM_model_python_1470862185979_86py_4_sid_9108		CAPSULE	☐

Click **Import** when you are done. The newly added models will then appear on the Models page.

Viewing Model Details

On the **Models** page, click the **view model details** link under the Action column for the model that you want to view.


STEAM < Projects

Prostate...

Models
Deployment
Configurations

Home > Projects > 3 > Models > 6

GLM_MODEL_PYTHON_1470862185979_86

Export Model
Deploy Model

compared to: SELECT MODEL FOR COMPARISON

Model Overview

Basics

Date	2016-08-10 14:12
Model Type	Generalized Linear Modeling

Model Parameters

Dataset Name	py_4_sid_9108
Response Column Name	CAPSULE

Goodness of Fit

Metrics

Mean Squared Error	0.202705
LogLoss	0.591463
R^2	0.157219
AUC	0.717572
Gini	0.435144

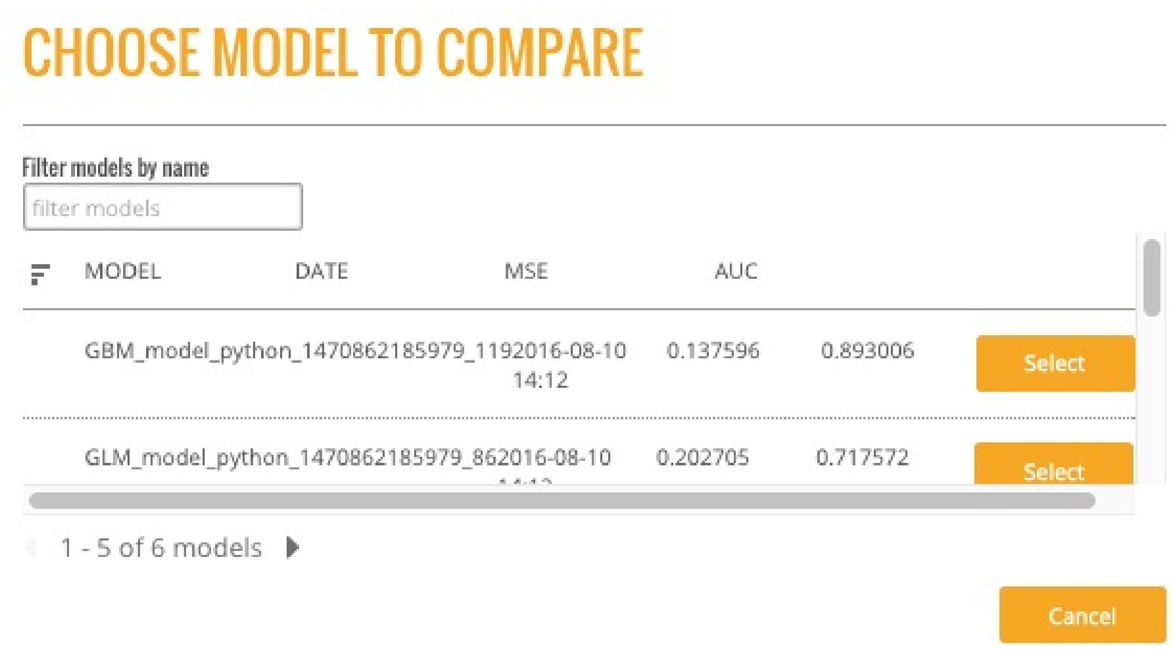
This page provides information about when the model was created, the algorithm and dataset used to create the model, and the response column specified when the model was built. The Goodness of Fit section provides value information for the model, including the Mean Squared Error, LogLoss, R^2, AUC, and Gini score. An ROC curve is available for binomial models.

From this page, you can perform the following actions:

- Compare two models
- Deploy the model
- Export the model

Comparing Models

1. While viewing model details, click the **Compared To** field. This opens a popup showing all models available in the current project.



CHOOSE MODEL TO COMPARE

Filter models by name

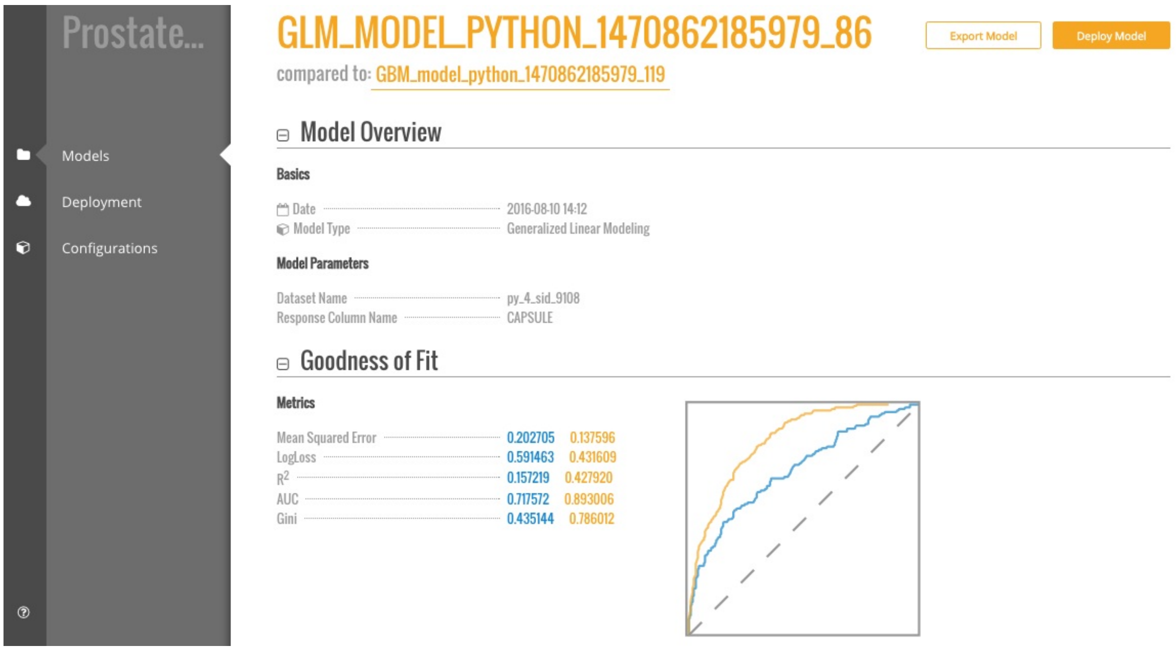
filter models

MODEL	DATE	MSE	AUC	
GBM_model_python_1470862185979_119	2016-08-10 14:12	0.137596	0.893006	Select
GLM_model_python_1470862185979_86	2016-08-10	0.202705	0.717572	Select

1 - 5 of 6 models

Cancel

2. Select to compare the current model with any available model. This exampel compares a GLM model with a GBM model. Once a model is selected, the Model Details page immediately populates with the comparison information. The current model values are displayed in blue, and the selected comparison model displays in orange.



GLM_MODEL_PYTHON_1470862185979_86

compared to: GBM_model_python_1470862185979_119

Model Overview

Basics

Date	2016-08-10 14:12
Model Type	Generalized Linear Modeling

Model Parameters

Dataset Name	py_4_sid_9108
Response Column Name	CAPSULE

Goodness of Fit

Metrics

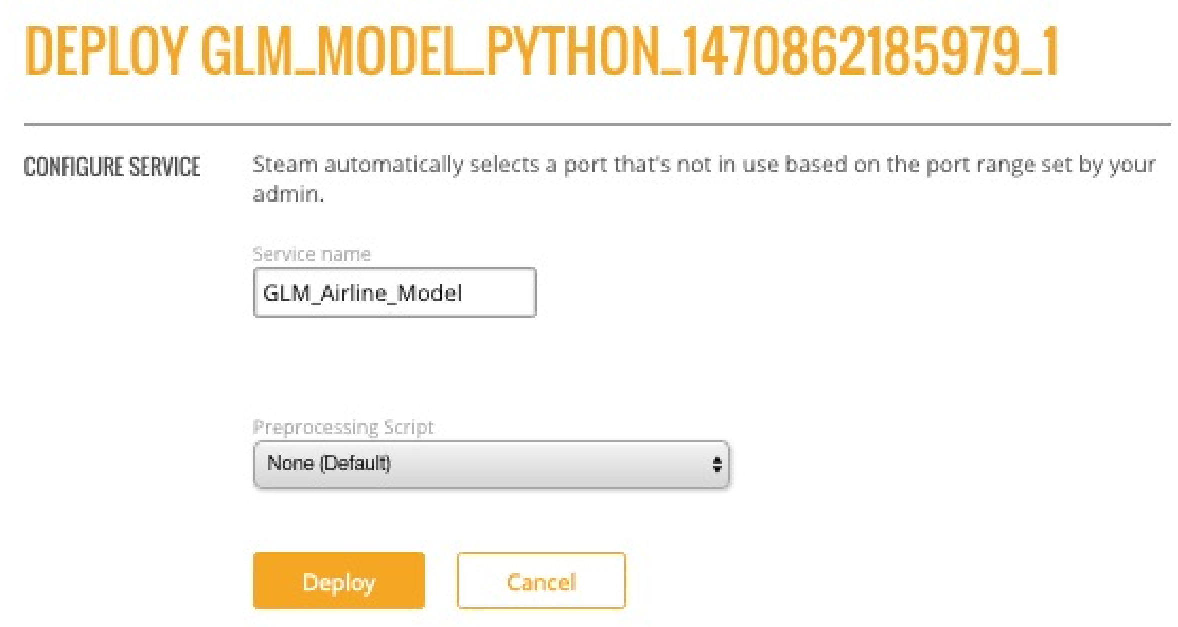
Mean Squared Error	0.202705	0.137596
LogLoss	0.591463	0.431609
R^2	0.157219	0.427920
AUC	0.717572	0.893006
Gini	0.435144	0.786012

Deploying a Model

After comparing models, you might decide to deploy one or more of the best models. Perform the steps below to deploy a model.

1. While viewing the model details, click the **Deploy Model** button. (Note that this can

also be done directly from the **Models** page by selecting the **deploy model** link in the Action column.)
2. Specify a service name for the deployment.
3. To perform pre-processing on the model, specify a Preprocessing Script. Note that this dropdown is populated with scripts that are added to the project. Information about adding preprocessing scripts is available in the Deployment section.
4. Click **Deploy** when you are done.


DEPLOY GLM_MODEL_PYTHON_1470862185979_1

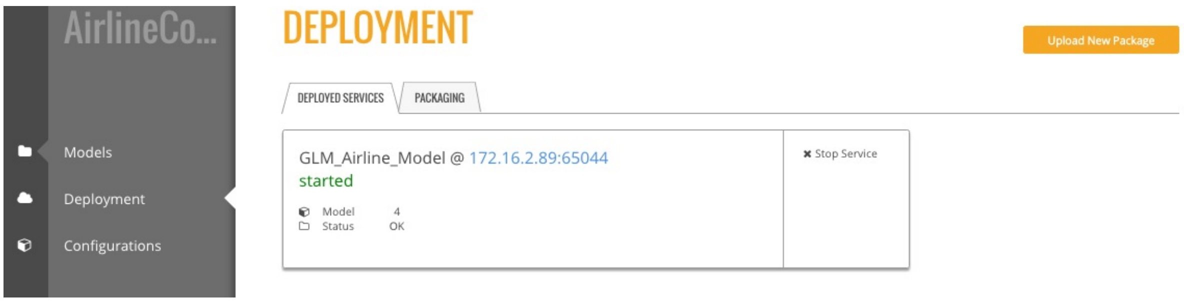
CONFIGURE SERVICE

Steam automatically selects a port that's not in use based on the port range set by your admin.

Service name: GLM_Airline_Model

Preprocessing Script: None (Default)

5. Upon successful completion, a scoring service will be created for this deployed model. Click the **Deployment** menu option on the left navigation to go to the Deployment page. Refer to the Deployment Page section for more information.



GLM_Airline_Model @ 172.16.2.89:65044
started
Model 4
Status OK

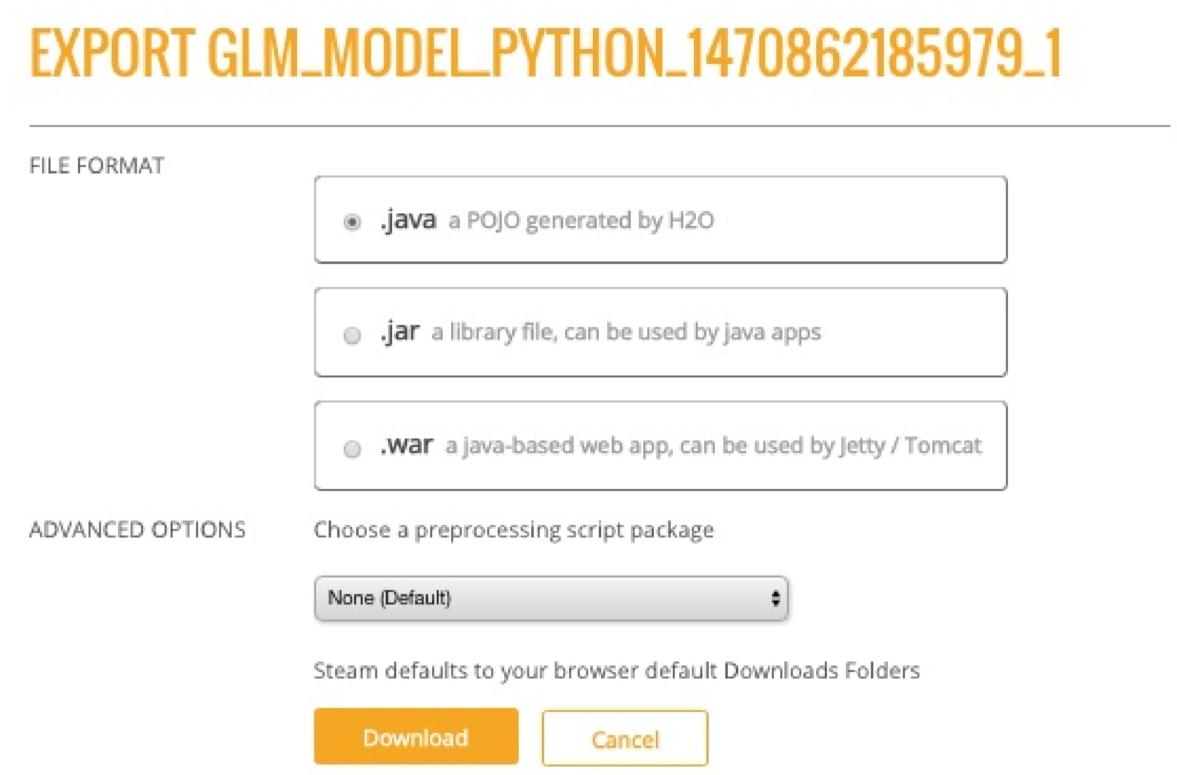
Exporting a Model

Steam allows you to export models to your local machine.

1. While viewing the model details, click the **Export Model** button.
2. Specify whether to export the model as a .java, .jar, or .war file.
3. To perform pre-processing on the model during the export, specify a Preprocessing Script. Note that this dropdown is populated with scripts that are added to the

project. Information about adding preprocessing scripts is available in the Deployment section.

4. Click **Download** when you are done.


EXPORT GLM_MODEL_PYTHON_1470862185979_1

FILE FORMAT

(●) .java a POJO generated by H2O

( ) .jar a library file, can be used by java apps

( ) .war a java-based web app, can be used by Jetty / Tomcat

ADVANCED OPTIONS

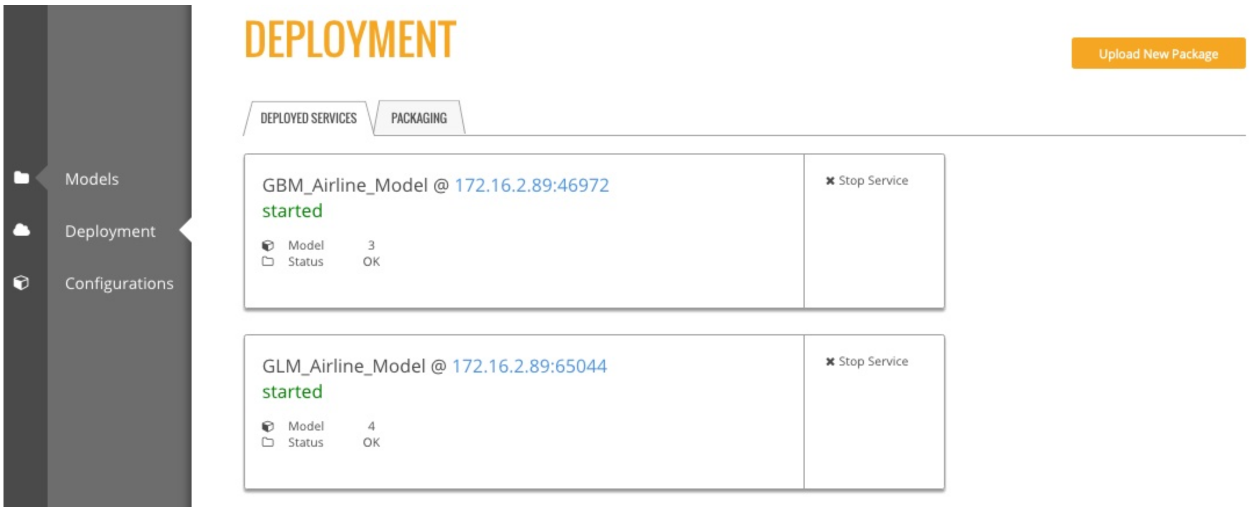
Choose a preprocessing script package

None (Default)

Steam defaults to your browser default Downloads Folders

Deployment

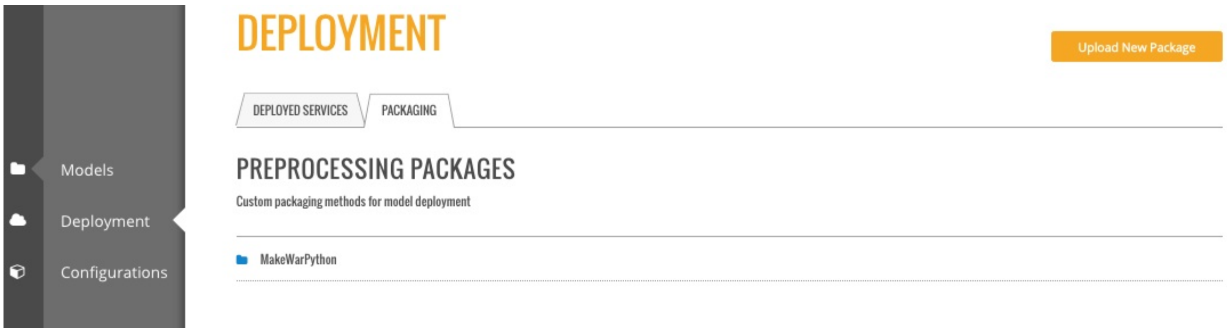
The **Deployment** page lists all available deployed services. For each deployed service, this page shows the model name, model ID, and the status. You can stop a running service by clicking the **Stop Service** button



GBM_Airline_Model @ 172.16.2.89:46972
started
Model 3
Status OK

GLM_Airline_Model @ 172.16.2.89:65044
started
Model 4
Status OK

In addition to showing deployed services, a Packaging tab is available showing the preprocessing packages used in the deployment.



PREPROCESSING PACKAGES
Custom packaging methods for model deployment

MakeWarPython

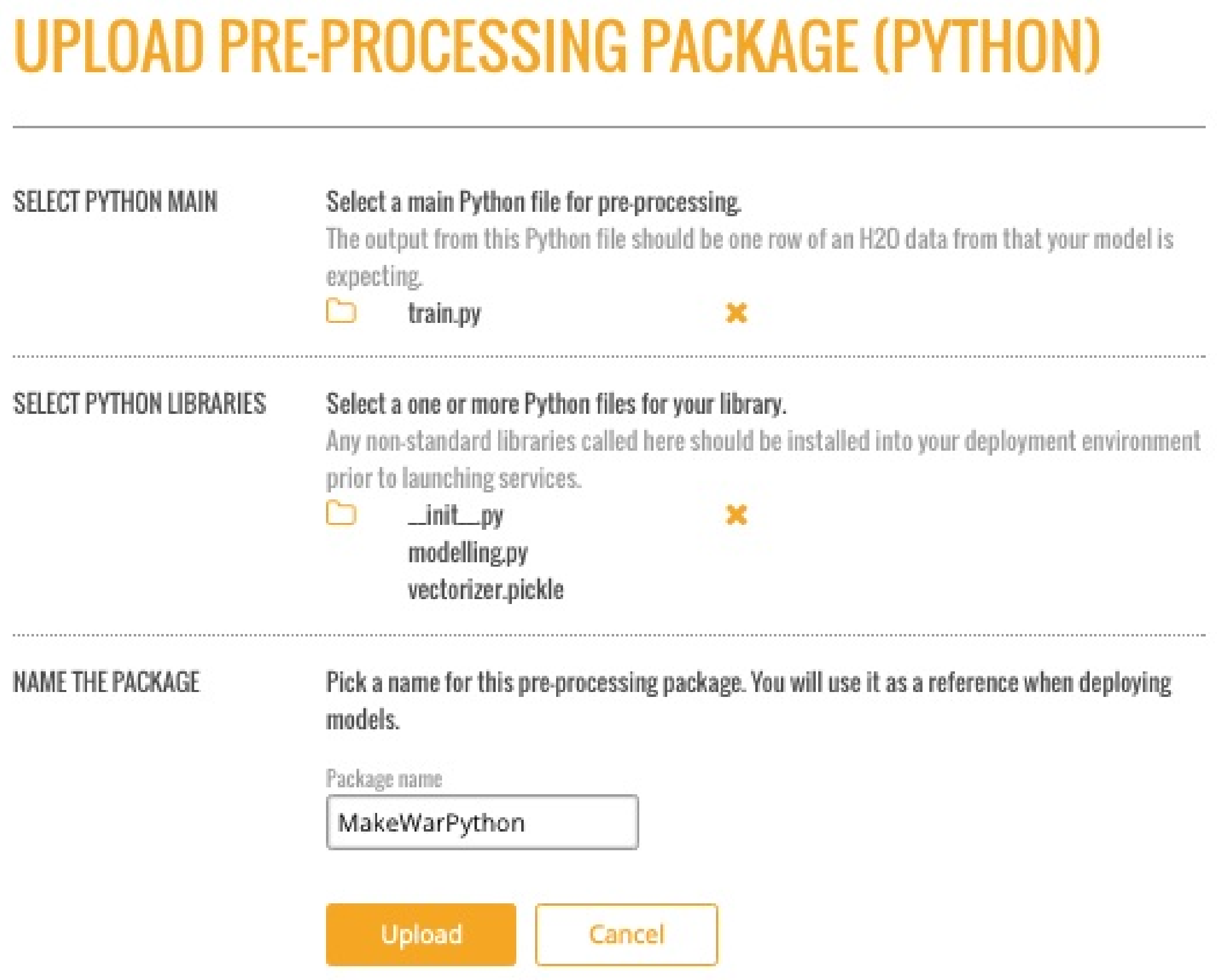
Uploading a New Package

Preprocessing packages can be used to perform additional data munging on an existing model.

1. To upload a new preprocessing package, click the **Upload New Package** button in the upper-right corner of the Deployment page.
2. Specify the main Python file that will be used for preprocessing. Click on the folder link to browse for this file.
3. Specify additional files that may be dependencies of the main Python preprocessing file.

4. Enter a name for this new package.
5. Click **Upload** when you are finished.

Upon successful completion, the new preprocessing package will display on the Packages tab of the Deployment page. This file can then be specified when deploying or exporting models. (Refer to Deploying a Model or Exporting a Model.)



UPLOAD PRE-PROCESSING PACKAGE (PYTHON)

SELECT PYTHON MAIN

Select a main Python file for pre-processing.
The output from this Python file should be one row of an H2O data from that your model is expecting.
train.py

SELECT PYTHON LIBRARIES

Select a one or more Python files for your library.
Any non-standard libraries called here should be installed into your deployment environment prior to launching services.
__init__.py
modelling.py
vectorizer.pickle

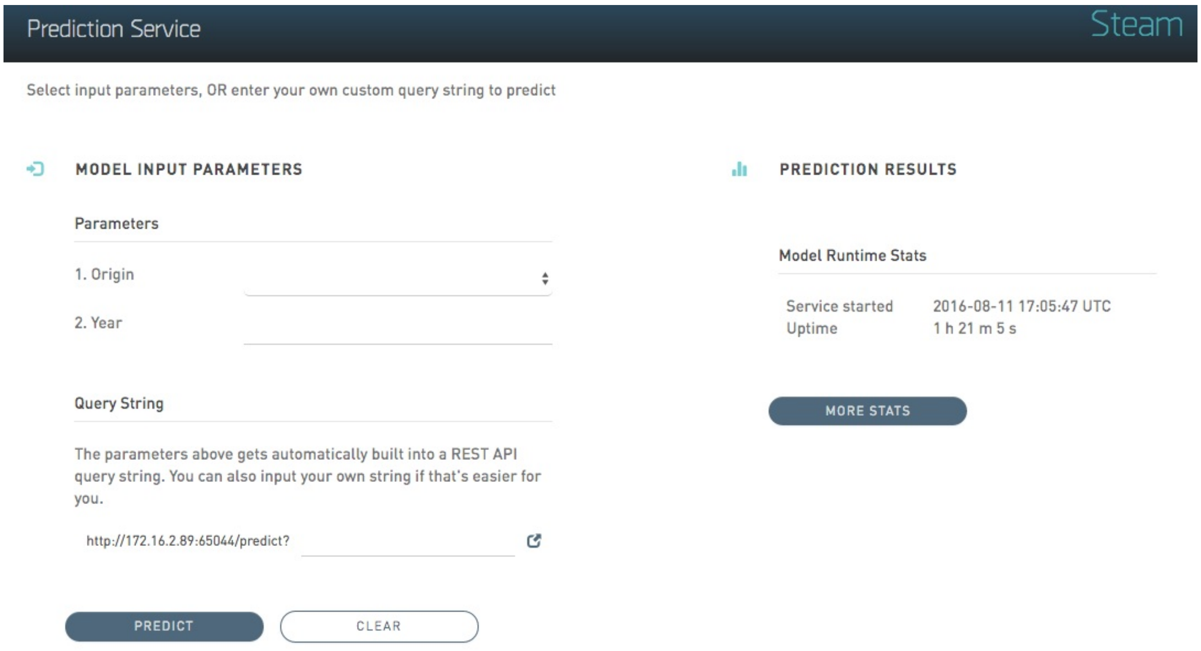
NAME THE PACKAGE

Pick a name for this pre-processing package. You will use it as a reference when deploying models.

Package name: MakeWarPython

Making Predictions

1. To reach the Steam Prediction Service, click the IP address link listed under the Deployed Services for the deployed model that you want to score. This opens Steam Prediction Service tool. The fields that display on the Prediction Service tool are automatically populated with field information from the deployed model.



Prediction Service

Steam

Select input parameters, OR enter your own custom query string to predict

MODEL INPUT PARAMETERS

Parameters

1. Origin

2. Year

Query String

The parameters above gets automatically built into a REST API query string. You can also input your own string if that's easier for you.

http://172.16.2.89:65044/predict?

PREDICTION RESULTS

Model Runtime Stats

Service started	2016-08-11 17:05:47 UTC
Uptime	1 h 21 m 5 s

2. Make predictions by specifying input values based on column data from the original dataset. This automatically populates the fields in the query string. (Note that you can optionally include input parameters directly in the query string instead of specifying parameters.)

3. Click **Predict** when you are done.

> **Note**: Use the **Clear** button to clear all entries and begin a new prediction. Use the **More Stats** button to view additional statistics about the scoring service results.

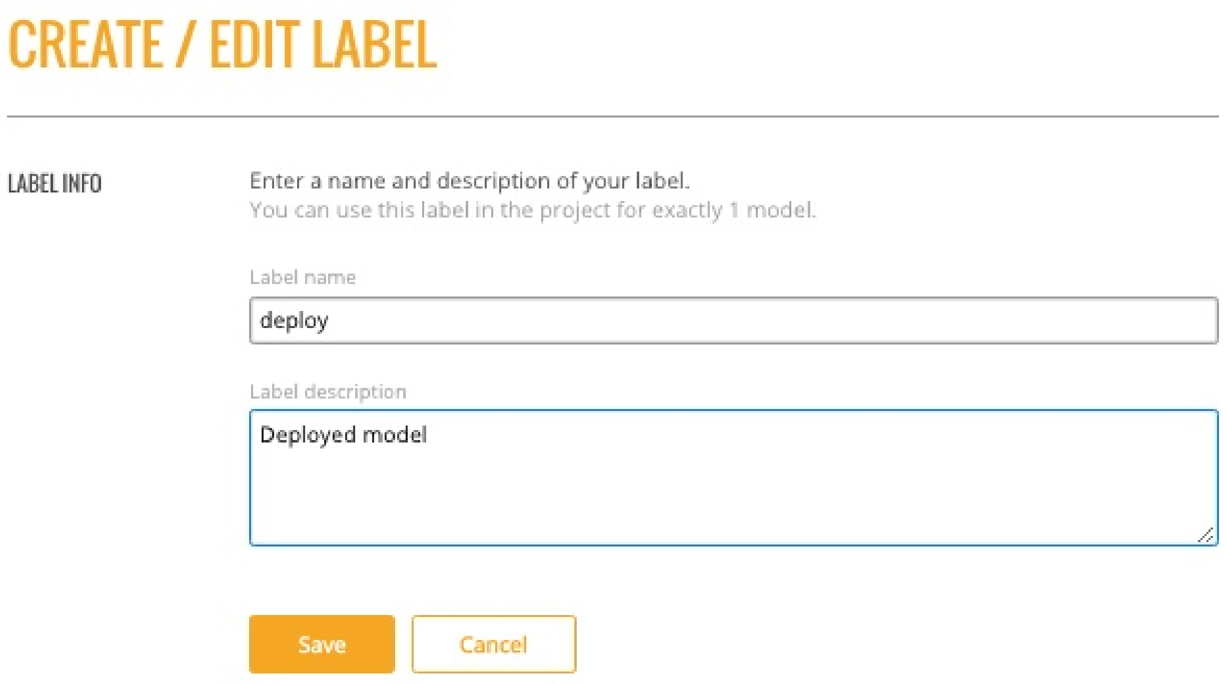
Configurations

Steam allows you to set labels for models (such as Production, Test, etc.) and apply permissions for using the labels. The Steam admin/superuser is responsible for creating new Steam users and setting roles and workgroups for those users. When setting Steam project configurations, labels can be created that allow, for example, only users in a Production workgroup to label a model as a production model.

When a label is applied to a model, the Project Configurations page will show all models associated with a label.

Creating a New Label

1. On the Configurations page, click the **Create New Label** button.
2. Enter a unique name for the label, the provide a description.
3. Click **Save** when you are done.



CREATE / EDIT LABEL

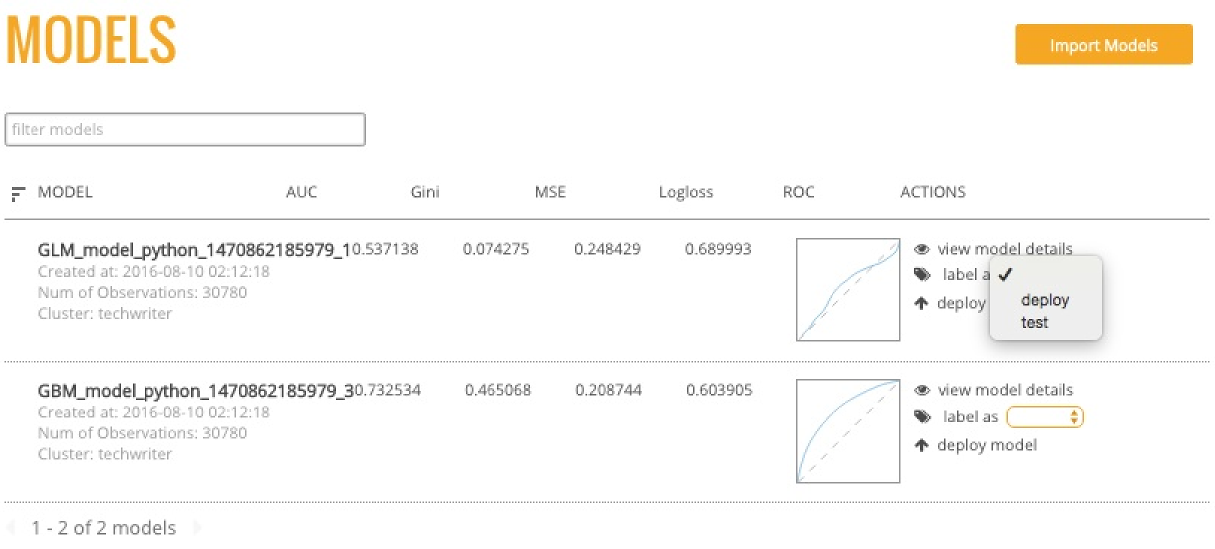
LABEL INFO

Enter a name and description of your label.
You can use this label in the project for exactly 1 model.

Label name: deploy

Label description: Deployed model

Upon successful completion, the new label will display on the Project Configurations page and can be edited or deleted. This label will also be available on the Models page in the **label as** dropdown. The following image shows two labels in the **label as** dropdown: deploy and test.

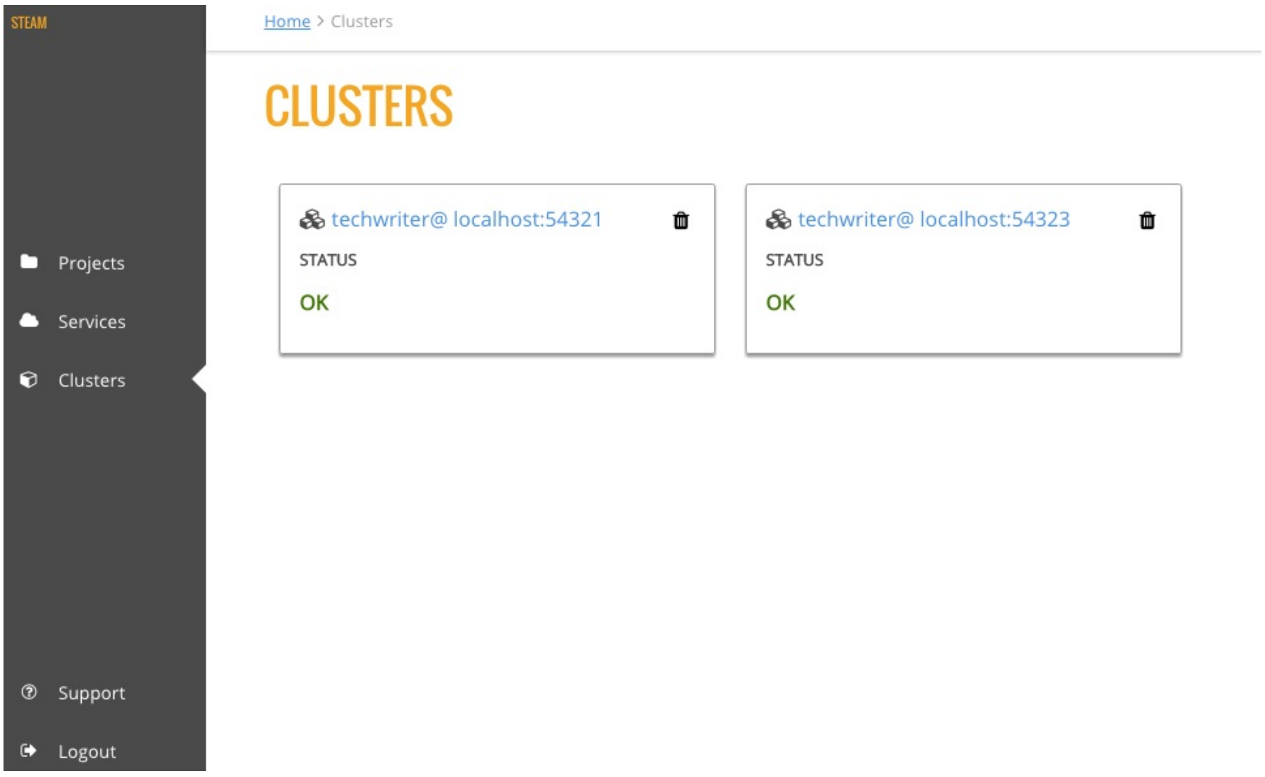


MODEL	AUC	Gini	MSE	Logloss	ROC	ACTIONS
GLM_model_python_1470862185979_1 Created at: 2016-08-10 02:12:18 Num of Observations: 30780 Cluster: techwriter	0.537138	0.074275	0.248429	0.689993		view model details label a ✓ deploy test deploy
GBM_model_python_1470862185979_3 Created at: 2016-08-10 02:12:18 Num of Observations: 30780 Cluster: techwriter	0.732534	0.465068	0.208744	0.603905		view model details label as deploy model

1 - 2 of 2 models

Clusters

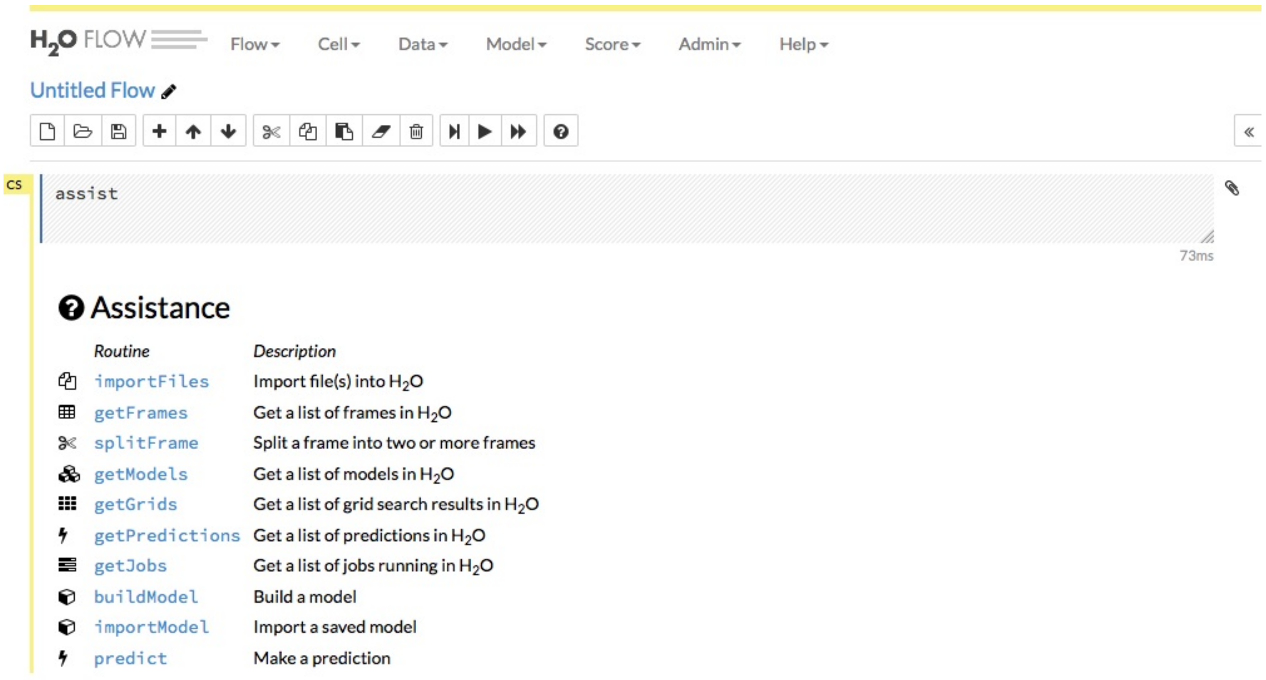
The **Clusters** page shows all H2O clusters that Steam is connected to along with the status of the cluster. From this page, you can click the link to access H2O Flow (see next section), or delete a cluster using the trashcan icon.



Using Steam with H2O Flow

As with other H2O products, Flow can be used alongside Steam when performing machine learning tasks. On the **Clusters** page, click the link for the H2O cluster that you want to open.

This opens H2O Flow in a new tab.



```
assist
```

Assistance

Routine	Description
importFiles	Import file(s) into H_2O
getFrames	Get a list of frames in H_2O
splitFrame	Split a frame into two or more frames
getModels	Get a list of models in H_2O
getGrids	Get a list of grid search results in H_2O
getPredictions	Get a list of predictions in H_2O
getJobs	Get a list of jobs running in H_2O
buildModel	Build a model
importModel	Import a saved model
predict	Make a prediction

> **Note**: Refer to the H2O Flow documentation for information on how to use Flow.

Stopping Steam

When you are finished using Steam, press Ctrl+C in each of the Steam, Compilation Service, and postgres terminal windows to stop the services end your session.

Using the Steam CLI

The CLI is an optional utility that can be used to maintain a Steam environment and to create new roles, workgroups, and users. The CLI will primarily be used by admins and/or Steam superusers. The steps below describe how to start the Steam CLI.

Perform the following steps to start the Steam CLI.

1. Open a terminal window and ssh to the machine running Steam. Be sure to provide the correct password for the node when prompted.

   `ssh <user>@<yarn_edge_node>`

2. Change directories to the Steam folder. From within this folder, log in to the machine running Steam. Use the password that you provided when you created superuser. The exmaple below logs in a user named **Bob**.

   `cd steam-0`

   `./steam login 192.168.2.182:8080 --username=bob --password=bobSpassword`

3. Run the following to verify that the CLI is working correctly.

   ```
   ./steam help
   ```

Refer to the CLI Command Reference Appendix for information on the commands available in the CLI.

CLI Command Reference Appendix

- `add engine`
- `build model`
- `create dataset`
- `create datasource`
- `create identity`
- `create project`
- `create role`
- `create workgroup`
- `deactivate identity`
- `delete cluster`
- `delete dataset`
- `delete engine`
- `delete model`
- `delete role`
- `delete service`
- `delete workgroup`
- `deploy engine`
- `get all permissions`
- `get all cluster-types`
- `get cluster`
- `get clusters`
- `get datasource`
- `get datasources`
- `get engine`
- `get engines`
- `get entities`
- `get history`
- `get identities`
- `get identity`
- `get model`
- `get models`
- `get project`
- `get projects`
- `get role`

- `get roles`
- `get service`
- `get services`
- `get workgroup`
- `get workgroups`
- `import model`
- `link identity`
- `link role`
- `login`
- `register cluster`
- `reset`
- `start cluster`
- `stop cluster`
- `stop service`
- `unlink identity`
- `unregister cluster`
- `update role`
- `update workgroup`

`add engine`

Description

Adds a new engine.

Usage

```
./steam add engine --engine-name="[name]" --engine-path="[path]"
```

Parameters

- `--engine-name="[name]"` : Enter the name of the engine
- `--engine-path="[path]"` : Enter the path for the engine

Example

The following example adds **h2o-genmodel.jar** to the list of available engines.

```
./steam add engine --engine-name="h2o-genmodel.jar" --engine-path="../Desktop/engi
nes"
```

`build model`

Description

Builds a model using either a specified algorithm or through AutoML.

Usage

```
./steam build model --cluster-id="[cluster]" --dataset-id="[dataset]" --algorithm=
"[algorithm]"

./steam build model --auto --cluster-id="[cluster]" --dataset-id="[dataset]" --tar
get-name="[model_name]" --max-run-time="[seconds]"
```

Parameters

- `--cluster-id="[cluster]"` : Specify the ID of the cluster that contains the dataset and will contain this model
- `--dataset-id="[dataset]"` : Specify the ID of the dataset to use to build the model
- `--algorithm="[algorithm]"` : Specify the algorithm to use for the model. This option cannot be used with `--auto` . Options include:
    - `gbm` : Build a Gradient Boosting Machine model
    - `glm` : Build a Generalized Linear model
    - `glrm` : Build a Generalized Low Rank model
    - `rf` : Build a Random Forest model
    - `svm` : Build a Support Vector Machine model
    - `dl` : Build a Deep Learning model
    - `nb` : Build a Naive Bayes model
- `--auto` : Specify to use AutoML to build the model

- `--target-name=[model_name]` : Specify a name for the AutoML model
- `--max_run_time` : When building an AutoML model, speicfy the maximum runtime in seconds to allow for the model to build.

Example

The following example builds a gbm model from the airlines dataset. This dataset was added using `create dataset` and has an ID of 1.

```
./steam build model --cluster-id="1" --dataset-id="1" --algorithm="gbm"
```

`create dataset`

Description

Creates a dataset from an available source file. Once created, the dataset can be used to build a model.

Usage

```
./steam create dataset --cluster-id=[cluster] --datasource-id=[source] --name="[da
tasetname]" --description="[description]" --response-column-name="[column]"
```

Parameters

- `--cluster-id=[cluster]` : Specify the ID of the cluster running H2O that will contain this dataset
- `--datasource-id=[source]` : Specify the ID of the datasource that will be used to create this dataset
- `--name="[datasetname]"` : Optionally enter a name for this dataset
- `--description="[description]"` : Optionally provide a description for this dataset
- `--response-column-name="[column"]` : Specify the column that will be used when making predictions

Example

The following example creates a dataset from a source file that was added using `create datasource`. In this example, Steam will generate a name for the dataset.

```
./steam build model --cluster-id="1" --datasource-id="1" --response-column-name="O
rigin"
```

`create datasource`

Description

Adds a datasource to the Steam database. Once added, this source file can be used to create a dataset.

Usage

```
./steam create datasource --name="[sourcename]" --description="[description]" --pa
th="[path]" --project-id=[id]
```

Parameters

- `--name="[datasetname]"` : Optionally enter a name for this dataset
- `--description="[description]"` : Optionally provide a description for this dataset
- `--path="[path]"` : Enter the path for the source file. This path is relative to the H2O cluster.
- `--project-id=[id]` : Specify the ID of the project that will contain this source file

Example

The following example adds the allyears2k.csv file to the Steam database.

```
./steam create datasource --name="allyears2k.csv" --description="airline data" --p
ath="../../Desktop/allyears2k.csv" --project-id=1
```

`create identity`

Description

Creates a new user.

Usage

```
./steam create identity [username] [password]
```

Parameters

- `[username]` : Enter a unique string for the new user name
- `[password]` : Enter a string for the new user's password

Example

The following example creates a new user with a username of "minsky" and a password of "m1n5kypassword".

```
./steam create identity minsky m1n5kypassword
Created user minsky ID: 2
```

`create project`

Description

Creates a project in the Steam database. Once created, datasources can be added to the project, ensuring that allo associated datasets and models are contained in this single location.

Usage

```
./steam create project --name="[projectname]" --description="[description]"
```

Parameters

- `--name="[projectname]"` : Enter a unique name for the project
- `--description="[description]"` : Enter a description for the project

Example

The following example creates a Prediction project.

```
./steam create project --name="Prediction" --description="Prediction project"
```

`create role`

Description

Creates a new role.

Usage

```
./steam create role [rolename] --desc="[description]"
```

Parameters

- `[rolename]` : Enter a unique string for the new role
- `--desc="[description]"` : Optionally enter a string that describes the new role

Example

The following example creates an engineer role.

```
./steam create role engineer --desc="a default engineer role"
Created role engineer ID: 2
```

`create workgroup`

Description

Creates a new workgroup.

Usage

```
./steam create workgroup [workgroupname] --desc="[description]"
```

Parameters

- `[workgroupname]` : Enter a unique string for the new workgroup
- `--desc="[description]"` : Optionally enter a string that describes the new workgroup

Example

The following example creates a data preparation workgroup.

```
./steam create workgroup preparation --desc="data prep group"
Created workgroup preparation ID: 1
```

`deactivate identity`

Description

Deactivates an identity based on the specified username.

Usage

```
./steam deactivate identity [username]
```

Parameters

- `[username]` : Specify the username of the identity that you want to deactivate.

Example

The following example deactivates user "minsky".

```
./steam deactivate minsky
```

`delete cluster`

Description

Deletes the specified YARN cluster from the database. Note that this command can only be used with YARN clusters (i.e., those started using `start cluster`.) This command will not work with local clusters. In addition, this commmand will only work on cluster that have been stopped using `stop cluster`.

Usage

```
./steam delete cluster [id]
```

Parameters

- `[id]` : Specify the ID of the cluster that you want to delete.

Example

The following example deletes cluster 1.

```
./steam get clusters
NAME        ID     ADDRESS            STATE     TYPE        AGE
user         1     localhost:54321    started    external    2016-07-01 11:45:58 -0
700 PDT    Cluster deleted: 1
```

`delete engine`

Description

Deletes the specified engine from the database.

Usage

```
./steam delete engine [id]
```

Parameters

- `[id]` : Specify the ID of the engine that you want to delete.

Example

The following example retrieves a list of engines currently added to the database. It then specifies to delete that automodel-hdp2.2.jar engine.

```
./steam get engines
NAME             ID    AGE
automl-hdp2.2.jar    1    2016-07-14 11:48:42 -0700 PDT
h2o-genmodel.jar    2    2016-07-14 11:49:47 -0700 PDT
./steam delete engine 1
Engine deleted: 1
```

`delete model`

Description

Deletes a model from the database based on the model's ID.

Usage

```
./steam delete model [modelId]
```

Parameters

- `[modelId]` : Specify the ID of the model that you want to delete.

Example

The following example deletes model 3 from the database. Note that you can use `get models` to retrieve a list of models.

```
./steam delete model 3
```

`delete role`

Description

Deletes a role from the database based on its ID.

Usage

```
./steam delete role [roleId]
```

Parameters

- `[roleId]` : Specify the ID of the role that you want to delete.

Example

The following example deletes role 3 from the database. Note that you can use [`get roles`]'(#get roles) to retrieve a list of roles. In the case below, this role corresponds to the default data science role.

```
./steam delete role 3
```

`delete service`

Description

A service represents a successfully deployed model on the Steam Scoring Service. This command deletes a service from the database based on its ID. Note that you must first stop a service before it can be deleted. (See `stop service` .)

Usage

```
./steam delete service [serviceId]
```

Parameters

- `[serviceId]` : Specify the ID of the service that you want to delete. Note that you can use `get services` to retrieve a list of services.

Example

The following example stops and then deletes service 2. This service will no longer be available on the database.

```
./steam stop service 2
./steam delete service 2
```

`delete workgroup`

Description

Deletes a workgroup from the database based on its ID.

Usage

```
./steam delete workgroup [workgroupId]
```

Parameters

- `[workgroupId]` : Specify the ID of the role that you want to delete.

Example

The following example deletes workgroup 3 from the database. Note that you can use `get workgroups` to retrieve a list of workgroups.

```
./steam delete workgroup 3
```

`deploy engine`

Description

Deploys an H2O engine. After an engine is successfully deployed, it can be specified when starting a cluster. (See `start cluster`.)

Usage

```
./steam deploy engine [path/to/engine]
```

Parameters

- `[path/to/engine]` : Specify the location of the engine that you want to deploy.

Example

The following specifies to deploy the H2O AutoML engine.

```
./steam deploy engine ../engines/automl-hdp2.2.jar
```

`get all permissions`

Description

Retrieves a list of permissions available in Steam along with the corresponding code. These permissions are currently hard coded into Steam.

Usage

```
./steam get all --permissions
```

Parameters

None

Example

The following example retrieves a list of Steam permissions.

```
./steam get all --permissions
Id    Code                Description
9    ManageCluster        Manage clusters
15    ManageDataset        Manage datasets
13    ManageDatasource    Manage datasources
7    ManageEngine        Manage engines
5    ManageIdentity    Manage identities
19    ManageLabel        Manage labels
17    ManageModel        Manage models
11    ManageProject        Manage projects
1    ManageRole        Manage roles
21    ManageService        Manage services
3    ManageWorkgroup    Manage workgroups
10    ViewCluster        View clusters
16    ViewDataset        View datasets
14    ViewDatasource    View datasources
8    ViewEngine        View engines
6    ViewIdentity        View identities
20    ViewLabel            View labels
18    ViewModel            View models
12    ViewProject        View projects
2    ViewRole            View roles
22    ViewService        View services
4    ViewWorkgroup        View workgroups
```

`get all cluster-types`

Description

Returns a list of all cluster types that Steam is connected to.

Usage

```
./steam get all --cluster-types
```

Parameters

None

Example

The following example retrieves all of the current Steam cluster types.

```
./steam get all --cluster-types
Id    Name
1    external
2    yarn
```

`get cluster`

Description

Retrieves detailed information for a specific cluster based on its ID.

Usage

```
./steam get cluster --cluster-id=[clusterId]
```

Parameters

- `--cluster-id=[clusterId]` : Specify the ID of the cluster that you want to retrieve

Example

The following example retrieves information for cluster ID 1.

```
./steam get cluster --cluster-id=1
Attribute        Value
Id:                1
Name:            H2O_from_python_techwriter_hh4m3i
TypeId:        1
DetailId:        0
Address:        localhost:54321
State:            started
CreatedAt:    1473883790
```

`get clusters`

Description

Retrieves a list of clusters.

Usage

```
./steam get clusters
```

Parameters

None

Example

The following example retrieves a list of clusters that are running H2O and are registered in Steam. (See `register cluster`.)

```
./steam get clusters
NAME        ID    ADDRESS            STATE    TYPE        AGE
user         1    localhost:54321    started    external    2016-07-01 11:45:58 -0
700 PDT
```

`get datasource`

Description

Retrieves information about a specific source file based on its ID.

Usage

```
./steam get datasource --datasource-id=[id]
```

Parameters

- `datasource-id=[id]`: Specify the ID of the datasource that you want to retrieve.

Example

The following example retrieves information about a datasource whose ID is 1. Note that you can use `get datasources` to retrieve a list of all datasources.

```
./steam get datasource --datasource-id=1
Attribute            Value
Id:                    1
ProjectId:        1
Name:                allyears2k.csv
Description:        airline data
Kind:                CSV
Configuration:    {"path":"../Desktop"}
CreatedAt:        1473879765
```

`get datasources`

Description

Retrieves a list of all datasources available in the database.

Usage

```
./steam get datasources
```

Parameters

None

Example

The following example retrieves a list of all datasources.

```
./steam get datasources

Id    ProjectId    Name            Description        Kind    Configuration
     CreatedAt
1    1            allyears2k.csv    airline data    CSV        {"path":"../Desktop
"}    1473879765
2    1            prostate.csv    prostate data    CSV        {"path":"../Desktop"
}    1473880195
```

`get engine`

Description

Retrieves information for a specific engine based on its ID.

Usage

```
./steam get engine --engine-id=[engineId]
```

Parameters

- `--engine-id=[engineId]` : Specify the ID of the engine that you want to retrieve

Example

The following example retrieves information about engine 1.

```
./steam get engine --engine-id=1
Attribute        Value
ID:                1
Name:            h2o-genmodel.jar
Location:        ../Desktop/engines
CreatedAt:     1473874219
```

`get engines`

Description

Retrieves a list of deployed engines.

Usage

```
./steam get engines
```

Parameters

None

Example

The following example retrieves a list of engines that have been deployed. (Refer to `deploy engine`.)

```
./steam get engines
Id     Name                Location            CreatedAt
1    h2o-genmodel.jar    ../Desktop/engines    1473874219
```

`get entities`

Description

Retrieves a list of supported Steam entity types.

Usage

```
./steam get entities
```

Parameters

None

Example

The following example retrieves a list of the supported Steam entity types.

```
./steam get entities
NAME        ID
role        1
workgroup     2
identity    3
engine        4
cluster        5
project        6
model        7
service        8
```

`get history`

Description

Retrieves recent activity information related to a specific user or for a specific cluster.

Usage

```
./steam get history [identity [identityName] | cluster [clusterId]]
```

Parameters

- `identity [identityName]`: Specifies to retrieve activity information related to a specific user

- `cluster [clusterId]` : Specifies to retrieve a activity information related to a specific cluster

Example

The following example retrieves information for user "bob".

```
./steam get history identity bob
USER    ACTION    DESCRITPION                        TIME
1        link    {"id":"2","name":"preparation","type":"workgroup"}    2016-07-15
09:32:55 -0700 PDT
1        link    {"id":"2","name":"engineer","type":"role"}        2016-07-15 09:3
2:44 -0700 PDT
1        create    {"name":"bob"}                        2016-07-15 09:32:32 -0700
 PDT
```

`get identities`

Description

Retrieves a list of users.

Usage

```
./steam get identities
```

Parameters

None

Example

The following example retrieves a list of users that are available on the database.

```
./steam get identities
NAME        ID    LAST LOGIN            AGE
bob            2    0000-12-31 16:00:00 -0800 PST    2016-07-15 09:32:32 -0700 PDT
jim            3    0000-12-31 16:00:00 -0800 PST    2016-07-15 09:32:38 -0700 PDT
superuser    1    0000-12-31 16:00:00 -0800 PST    2016-07-15 09:21:58 -0700 PDT
```

`get identity`

Description

Retrieve information about a specific user.

Usage

```
./steam get identity [identityId]
```

Parameters

- `[identityId]` : Specify the ID of the user you want to retrieve

Example

The following example retrieves information about user Jim.

```
./steam get identity jim
            jim
STATUS:        Active
LAST LOGIN:    0000-12-31 16:00:00 -0800 PST
ID:        3
AGE:        2016-07-15 09:32:38 -0700 PDT

WORKGROUP    DESCRIPTION
production    production group

ROLE        DESCRIPTION
datascience    a default data scientist role

PERMISSIONS
Manage models
View clusters
Manage projects
```

`get model`

Description

Retrieves detailed information for a specific model.

Usage

```
./steam get model [modelId]
```

Parameters

- `[modelId]` : Specify the ID of the model that you want to retrieve

Example

The following example retrieves information for model 2.

```
./steam get model 2
```

`get models`

Description

Retrieves a list of models.

Usage

```
./steam get models
```

Parameters

None

Example

The following example retrieves a list of models that are available on the database.

```
./steam get models
```

`get project`

Description

Retrieves detailed information for a specific project based on its ID.

Usage

```
./steam get project --project-id=[id]
```

Parameters

- `--project-id=[id]`: Specify the ID of the project that you want to retrieve

Examples

The following example retrieves information about a project whose ID is 1. Note that you can use `get projects` to retrieve a list of all projects and IDs.

```
./steam get project --project-id=1
Attribute        Value
Id:                1
Name:            Prediction
Description:    Prediction project
ModelCategory:
CreatedAt:    1473878624
```

`get projects`

Description

Retrieves a list of all projects in the Steam database.

Usage

```
./steam get projects
```

Parameters

None

Example

The following example retrieves a list of projects that are available on the database.

```
./steam get projects
Id     Name        Description            ModelCategory    CreatedAt
1    Prediction    Prediction project                      1473878624
2    Churn        Customer churn project                  1473879033
```

`get role`

Description

Retrieves detailed information for a specific role based on its name.

Usage

```
./steam get role --role-id=[id]
```

Parameters

- `--role-id=[id]` : Specify the ID of the role that you want to retrieve

Example

The following example retrieves information about the datascience role.

```
./steam get role --role-id=2
Attribute        Value
Id:                2
Name:            datascience
Description:    a default data science role
Created:        1473874053
```

`get roles`

Description

Retrieves a list of roles.

Usage

```
./steam get roles
```

Parameters

None

Example

The following example retrieves a list of roles that are available on the database.

```
./steam get roles
NAME        ID    DESCRIPTION                    CREATED
Superuser    1    Superuser                    1473874053
datascience    2    a default data science role    1473893347
```

`get service`

Description

A service represents a successfully deployed model on the Steam Scoring Service. This command retrieves detailed information about a specific service based on its ID.

Usage

```
./steam get service [serviceId]
```

Parameters

- `[serviceId]` : Specify the ID of the service that you want to retrieve

Example

The following example retrieve information about service 2.

```
./steam get service 2
```

`get services`

Description

A service represents a successfully deployed model on the Steam Scoring Service. This command retrieves a list of services available on the database.

Usage

```
./steam get services
```

Parameters

None

Example

The following example retrieves a list of services that are available on the database.

```
./steam get services
```

`get workgroup`

Description

Retrieves information for a specific workgroup based on its name.

Usage

```
./steam get workgroup [workgroupName]
```

Parameters

- `[workgroupName]` : Specify the name of the workgroup that you want to retrieve

Example

The following example retrieves information about the production workgroup

```
./steam get workgroup production
                production
DESCRIPTION:    production group
ID:        3
AGE:    2016-07-15 09:32:27 -0700 PDT

IDENTITIES: 1
NAME    STATUS    LAST LOGIN
jim        Active    0000-12-31 16:00:00 -0800 PST
```

`get workgroups`

Description

Retrieves a list of workgroups currently available on the database.

Usage

```
./steam get workgroups --identity=[identityName]
```

Parameters

- `--identity=[identityName]` : Optionally specify to view all workgroups associated with a specific user name

Example

The following example retrieves a list of workgroups that are available on the database.

```
./steam get workgroups
NAME        ID    DESCRIPTION        AGE
preparation    2    data prep group        2016-07-15 09:32:21 -0700 PDT
production    3    production group    2016-07-15 09:32:27 -0700 PDT
```

`import model`

Description

Imports a model from H2O based on its ID.

Usage

```
./steam import model [clusterId] [modelName]
```

Parameters

- `[clusterId`]: Specify the H2O cluster that contains the model you want to import
- `[modelName]` : Specify the name of the that you want to import into steam.

Example

The following example specifies to import the GBM_model_python_1468599779202_1 model from Cluster 1.

```
./steam import model 1 GBM_model_python_1468599779202_1
```

`link identity`

Description

Links a user to a specific role or workgroup.

Usage

```
./steam link identity [identityName] [role [roleId] | workgroup [workgroupId]]
```

Parameters

- `[identityName]` : Specify the user that will be linked to a role or workgroup.
- `role [roleId]` : Specify the role that the user will be linked to.
- `workgroup [workgroupId]` : Specify the workgroup that the the user will be linked to.

Example

The following example links user Jim to datascience role and then to the production workgroup.

```
./steam link identity jim role datascience
./steam link identity jim workgroup production
```

`link role`

Description

Links a role to a certain set of permissions.

Usage

```
./steam link role [roleId] [permissionId1 permissionId2 ...]
```

Parameters

- `[roleId]` : Specify the role that the user will be linked to.
- `[permissionId]` : Specify a single permission or a list of permissions to assign to this role.

Example

The following example links the datascience role to the ManageProject, ManageModel, and ViewCluster permissions.

```
./steam link role datascience ManageProject ManageModel ViewCluster
```

`login`

Description

Logs a user in to Steam

Usage

```
./steam login [address:port] --username=[userName] --password=[password]
```

Parameters

- `[address:port]` : Specify the address and port of the Steam server.
- `--username=[userName]` : Specify the username.
- `--password=[password]` : Specify the user's password.

Example

The following example logs user Bob into a Steam instance running on localhost:9000.

```
./steam login localhost:9000 --username=bob --password=bobSpassword
Login credentials saved for server localhost:9000
```

`register cluster`

Description

Registers a cluster that is currently running H2O (typically a local cluster). Once registered, the cluster can be used to perform machine learning tasks through Python, R, and Flow. The cluster will also be visible in the Steam web UI.

Note that clusters that are started using this command can be stopped from within the web UI or using `unregister cluster` . You will receive an error if you attemt to stop registered clusters using the `stop cluster` command.

Usage

```
./steam register cluster [address]
```

Parameters

- `[address]` : Specify the IP address and port of the cluster that you want to register.

Example

The following example registers Steam on localhost:54323. Note that this will only be successful if H2O is already running on this cluster.

```
./steam register cluster localhost:54323
Successfully connected to cluster 2 at address localhost:54323
```

`reset`

Description

Resets the current Steam cluster instance. This removes the current authentication from Steam. You will have to re-authenticate in order to continue to use Steam.

Usage

```
./steam reset
```

Parameters

None

Examples

The following example resets the current Steam instance.

```
./steam reset
Configuration reset successfully. Use 'steam login <server-address>' to re-authent
icate to Steam
```

`start cluster`

Description

After you have deployed engine, you can use this command to start a new cluster through YARN using a specified engine. Note that this command is only valid when starting Steam on a YARN cluster. To start Steam on a local cluster, use `register cluster` instead.

Usage

```
./steam start cluster [id] [engineId] --size=[numNodes] --memory=[string]
```

Parameters

- `[id]` : Enter an ID for this new cluster.
- `[engineId]` : Specify the ID of the engine that this cluster will use. If necessary, use `get engines` to retrieve a list of all available engines.
- `--size=[numNodes]` : Specify an integer for the number of nodes in this cluster.
- `--memory=[string]` : Enter a string specifying the amount of memory available to Steam in each node (for example, "1024m", "2g", etc.)

Example

The following example retrieves a list of engines, then starts a cluster through YARN using one from the list. The cluster is configured with 2 nodes that are 2 gigabytes each.

```
./steam get engines
NAME                ID     AGE
h2o-genmodel.jar    1     2016-07-01 13:30:50 -0700 PDT
h2o.jar            2     2016-07-01 13:32:10 -0700 PDT
./steam start cluster 9 1 --size=2 --memory=2g
```

`stop cluster`

Description

Stops a YARN cluster that was started through the CLI or web UI. (See `start cluster`.) Note that you will receive an error if you attempt to stop a cluster that was started using `register cluster`.

Usage

```
./steam stop cluster [id]
```

Parameters

- `[id]` : Specify the ID of the cluster that you want to stop. If necessary, use `get clusters` to retrieve a list of clusters.

Example

The following example stops a cluster that has an ID of 9.

```
./steam stop cluster 9
```

`stop service`

Description

A service represents a successfully deployed model on the Steam Scoring Service. Use this command to stop a service.

Usage

```
./steam stop service [serviceId]
```

Parameters

- `[serviceId]` : Specify the ID of the scoring service that you want to stop. If necessary, use `get services` to retrieve a list of running services.

Example

The following example stops a service that has an ID of 2.

```
./steam stop service 2
```

`unlink identity`

Description

Removes a user's permissions from a specific role or workgroup.

Usage

```
./steam unlink identity [identityName] [role [roleId] | workgroup [workgroupId]]
```

Parameters

- `[identityName]` : Specify the user that will be unlinked from a role or workgroup
- `role [roleId]` : Specify the role that the user will be unlinked from
- `workgroup [workgroupId]` : Specify the workgroup that the the user will be unlinked from

Example

The following example removes user Jim from the datascience role and then from the production workgroup.

```
./steam unlink identity jim role datascience
./steam unlink identity jim workgroup production
```

`unregister cluster`

Description

Stops a cluster that was registered through the CLI or the web UI. (See `register cluster`.) Note that this does not delete the cluster. Also note that you will receive an error if you attempt to unregister a cluster that was started using `start cluster`.

Usage

```
./steam unregister cluster [id]
```

Parameters

- `[id]` : Specify the ID of the cluster that you want to stop. If necessary, use `get clusters` to retrieve a list of clusters.

Example

The following example stops a cluster that has an ID of 9.

```
./steam unregister cluster 2
Successfully unregisted cluster %d 2
```

`update role`

Description

Edits the description and/or name of an existing role. When a role is edited, the edit will automatically propagate to all identities that are associated with this role.

Usage

```
./steam update role [rolename] --desc="[description]" --name="[newRoleName]
```

Parameters

- `[rolename]` : Enter the role name that you want to edit
- `desc="[description]"` : Optionally enter a string that describes the new role
- `name="[newRoleName]"` : Enter a unique string for the new role name

Example

The following example changes the name of the engineer role to be "science engineer".

```
./steam update role engineer --desc="A better engineer" --name="science engineer"
Successfully updated role: engineer
```

`create workgroup`

Description

Edits the description and/or name of an existing workgroup. When a workgroup is edited, the edit will automatically propagate to all identities that are associated with this workgroup.

Usage

```
./steam update workgroup [workgroupname] --desc="[description]" --name="[newWorkgr
oupName]
```

Parameters

- `[workgroup]` : Enter the workgroup name that you want to edit
- `desc="[description]"` : Optionally enter a string that describes the new workgroup
- `name="[newWorkgroupName]"` : Enter a unique string for the new workgroup name

Example

The following example changes the name of the production workgroup to be "deploy".

```
./steam update workgroup production --desc="A deploy workgroup" --name="deploy"
Successfully updated workgroup: production
```