Saving, Loading, Downloading, and Uploading Models

This section describes how to save, load, download, and upload binary and MOJO models using R, Python, and Flow.

Binary Models

When saving an H2O binary model with `h2o.saveModel` (R), `h2o.save_model` (Python), or in Flow, you will only be able to load and use that saved binary model with the same version of H2O that you used to train your model. H2O binary models are not compatible across H2O versions. If you update your H2O version, then you will need to retrain your model. For production, you can save your model as a POJO/MOJO. These artifacts are not tied to a particular version of H2O because they are just plain Java code and do not require an H2O cluster to be running.

In R and Python

In R and Python, you can save a model locally or to HDFS using the `h2o.saveModel` (R) or `h2o.save_model` (Python) function . This function accepts the model object and the file path. If no path is specified, then the model will be saved to the current working directory. After the model is saved, you can load it using the `h2o.loadModel` (R) or `h2o.load_model` (Python) function. You can also upload a model from a local path to your H2O cluster.

Notes:

- When saving a file, the owner of the file saved is the user by which H2O cluster or Python/R session was executed.
- When downloading a file, the owner of the file saved is the user by which the Python/R session was executed.

R Python

```
# build the model
model <- h2o.deeplearning(params)

# save the model
model_path <- h2o.saveModel(object = model, path = getwd(), force = TRUE)
print(model_path)
/tmp/mymodel/DeepLearning_model_R_1441838096933

# load the model
saved_model <- h2o.loadModel(model_path)

# download the model built above to your local machine
my_local_model <- h2o.download_model(model, path = "/Users/UserName/Desktop")

# upload the model that you just downloaded above
# to the H2O cluster
uploaded_model <- h2o.upload_model(my_local_model)
```

Note: When saving to HDFS, you must prepend the save directory with `hdfs://`. For example:

R   Python

```
# build the model
model <- h2o.glm(model params)

# save the model to HDFS
hdfs_name_node <- "node-1"
hdfs_tmp_dir <- "/tmp/runit"
model_path <- sprintf("hdfs://%s%s", hdfs_name_node, hdfs_tmp_dir)
h2o.saveModel(model, path = model_path, name = "mymodel")
```

In Flow

The steps for saving and loading models in Flow are described in the **Using Flow - H2O's Web UI** section. Specifically, refer to Exporting and Importing Models for information about exporting and importing binary models in Flow.

MOJO Models

Introduction

The MOJO import functionality provides a means to use external, pre-trained models in H2O - mainly for the purpose of scoring. Depending on each external model, metrics and other model information might be obtained as well. Currently, only selected H2O MOJOs are supported. (See the MOJO Quick Start section for information about creating MOJOs.)

Supported MOJOs

Only a subset of H2O MOJO models is supported in this version.

- GBM (Gradient Boosting Machines)
- DRF (Distributed Random Forest)
- IRF (Isolation Random Forest)
- GLM (Generalized Linear Model)
- XGBoost

Saving and Importing MOJOs

Importing a MOJO can be done from Python, R, and Flow. H2O imports the model and embraces it for the purpose of scoring. Information output about the model may be limited.

Saving and Importing in R or Python

R Python

```
data <- h2o.importFile(path = 'training_dataset.csv')
cols <- c("Some column", "Another column")
original_model <- h2o.glm(x = cols, y = "response", training_frame = data)

path <- "/path/to/model/directory"
mojo_destination <- h2o.save_mojo(original_model, path = path)
imported_model <- h2o.import_mojo(mojo_destination)

new_observations <- h2o.importFile(path = 'new_observations.csv')
h2o.predict(imported_model, new_observations)
```

Importing a MOJO Model in Flow

To import a MOJO model in Flow:

1. Import or upload the MOJO as a Generic model into H2O. To do this, click on **Data** in the top menu and select either **Import Files** or **Upload File**.
2. Retrieve the imported MOJO by clicking **Models** in the top menu and selecting **Import MOJO Model**.

Downloading and Uploading MOJOs

Alternatively, the `download_mojo()` and `h2o.upload_mojo()` R and Python functions can be used when downloading/uploading MOJOs from a client computer standing outside of an H2O cluster.

Downloading and Uploading in R and Python

R Python

```
# train a GBM model
library(h2o)
h2o.init()
fr <- as.h2o(iris)
my_model <- h2o.gbm(x = 1:4, y = 5, training_frame = fr)

# save to the current working directory
my_mojo <- h2o.download_mojo(my_model, "/Users/UserName/Desktop")

# upload the MOJO
mojo_model <- h2o.upload_mojo(my_mojo)
```

Advanced MOJO Model Initialization

It is also possible to import a MOJO from already uploaded MOJO bytes using Generic model. Generic model is the underlying mechanism behind MOJO import. In this case, there is no need to re-upload the MOJO every time a new MOJO imported model is created. The upload can occur only once.

Defining a Generic Model

The following options can be specified when using a Generic model:

- model_id: Specify a custom name for the model to use as a reference.
- **model_key**: Specify a key for the self-contained model archive.
- **path**: Specify a path to the file with the self-contained model archive.

Examples

R Python

```r
data <- h2o.importFile(path = 'training_dataset.csv')
cols <- c("Some column", "Another column")
original_model <- h2o.glm(x = cols, y = "response", training_frame = data)

path <- "/path/to/model/directory"
mojo_destination <- h2o.download_mojo(model = original_model, path = path)

# Only import or upload MOJO model data, do not initialize the generic model yet
imported_mojo_key <- h2o.importFile(mojo_destination, parse = FALSE)
# Build the generic model later, when needed
generic_model <- h2o.generic(model_key = imported_mojo_key)

new_observations <- h2o.importFile(path = 'new_observations.csv')
h2o.predict(generic_model, new_observations)
```