[Hands - On] Introduction to R, Python, and Flow

Amy Wang

amy@h2o.ai

H2O Prerequisites

To Launch H2O and Flow the only prerequisite is:

- 64 bit Java 6+

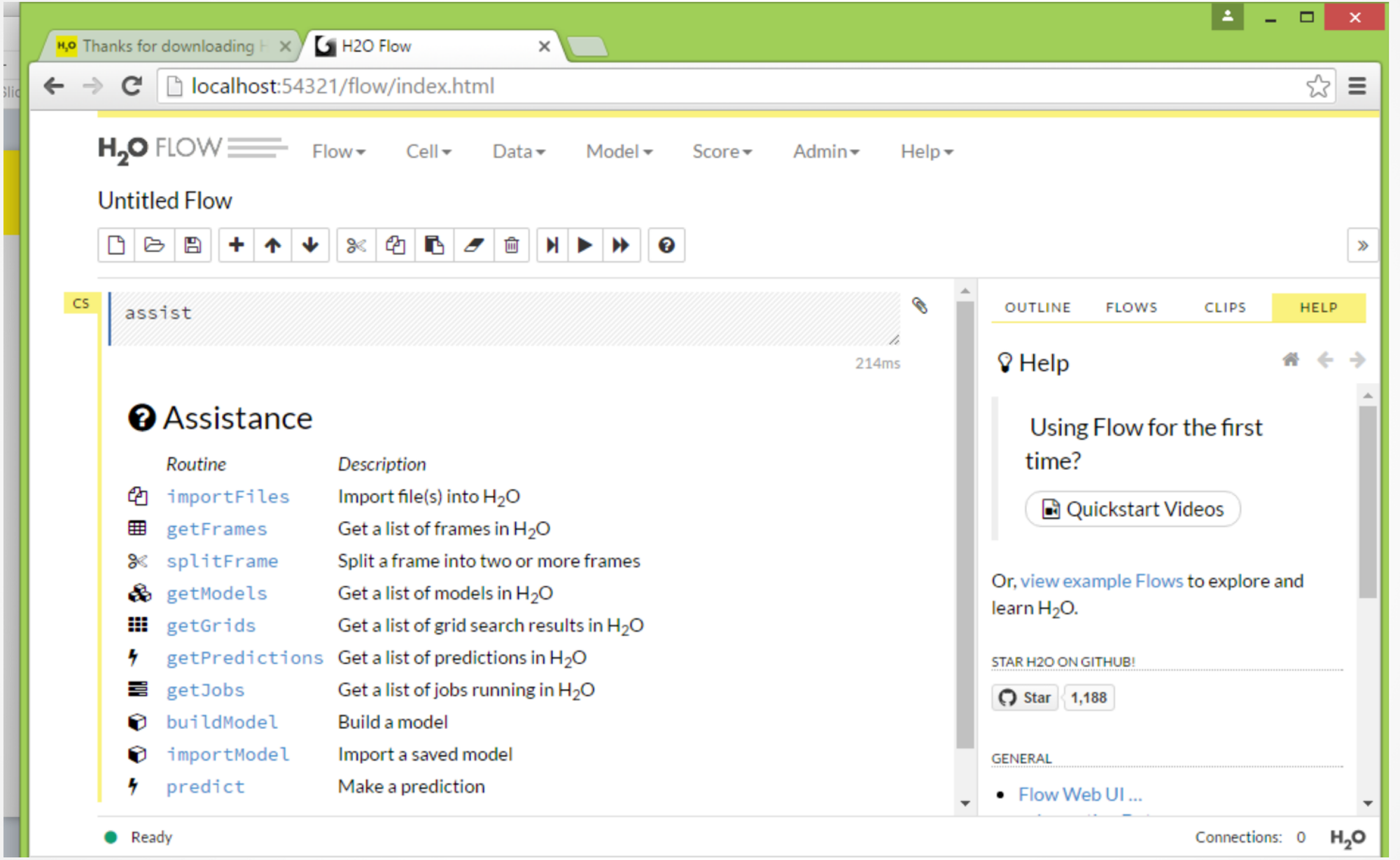
Consequences of not having 64 bit version:

- You will not be able to launch a cluster larger than 1 GB

Please install it from USB drive from the appropriate Windows or Mac Directory!

Flow Users

- Web-based UI that comes prepackaged in H2O
- To launch Flow you simply have to launch H2O either through the command line, R, or python
- To launch via the command line:
  - Unpack the h2o-3.8.3.3.zip file in Windows/ or OSX/
  - In h2o-3.8.3.3 directory execute from a terminal or command prompt: java -Xmx4g -jar h2o.jar
  - Access flow at http://localhost:54321


Thanks for downloading
H2O Flow
localhost:54321/flow/index.html
H2O FLOW
Flow
Cell
Data
Model
Score
Admin
Help
Untitled Flow
CS
assist
214ms
Assistance
Routine
Description
importFiles
Import file(s) into H2O
getFrames
Get a list of frames in H2O
splitFrame
Split a frame into two or more frames
getModels
Get a list of models in H2O
getGrids
Get a list of grid search results in H2O
getPredictions
Get a list of predictions in H2O
getJobs
Get a list of jobs running in H2O
buildModel
Build a model
importModel
Import a saved model
predict
Make a prediction
OUTLINE
FLOWS
CLIPS
HELP
Help
Using Flow for the first time?
Quickstart Videos
Or, view example Flows to explore and learn H2O.
STAR H2O ON GITHUB!
Star
1,188
GENERAL
Flow Web UI ...
Ready
Connections: 0
H2O

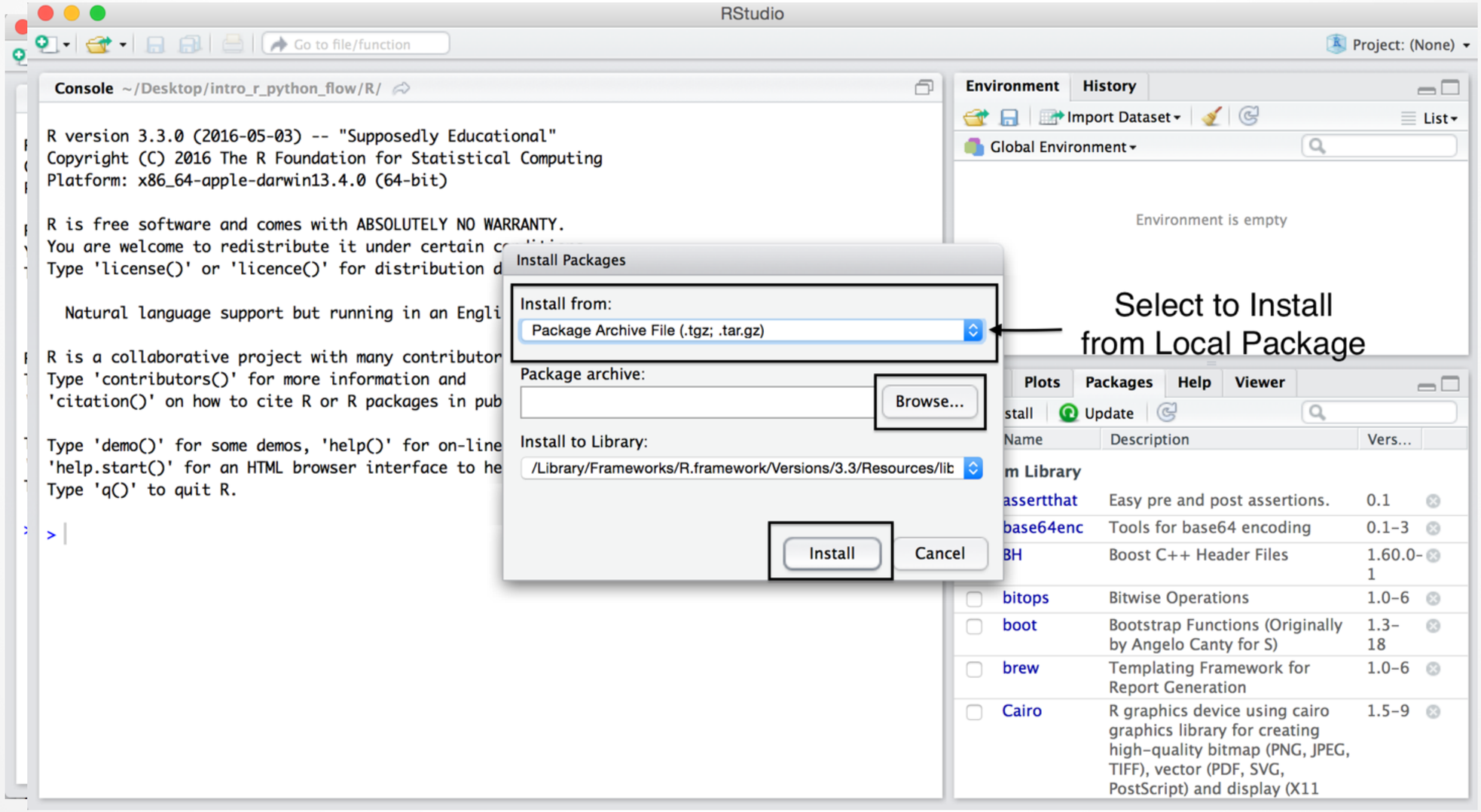
R Users

- Prerequisite: R installed (for Windows Users please installed in a folder with no spaces e.g. Program Files)
- Add R to your PATH
- To quickly Install Dependent Packages:
  - R CMD INSTALL Windows/RPackages/*
- Install H2O package
  - R CMD INSTALL h2o-3.8.3.3/R/*

R Users

- To install from cran, please run:
  - install.packages(“statmod”)
  - install.packages(“RCurl”)
  - install.packages(“jsonlite”)
  - install.packages(“h2o”)
- To check that it is properly installed run:
  - library(h2o)
  - h2o.init(nthreads = -1)

If you’re having difficulties installing from the terminal or command prompt, you can just as easily install from a R console or R Studio windows:



Python Users

- Prerequisite: Python 2.7 or 3.3 installed
- To Install Dependent modules:
  - pip install Windows/PythonModules/requests-2.8.1.tar.gz
  - pip install Windows/PythonModules/tabulate-0.7.5.tar.gz
  - pip install Windows/PythonModules/futures-3.0.5-py2-none-any.whl
- Install H2O wheel
  - pip install h2o-3.8.3.3-py2.py3-none-any.whl

Python Users

- Prerequisite: Python 2.7 or 3.3 installed
- To Install Dependent modules:
  - pip install requests
  - pip install tabulate
  - pip install futures
  - pip install six
- Install H2O wheel
  - pip install h2o-3.8.3.3-py2.py3-none-any.whl

Debugging Installation Issues

- To check to make sure you have the necessary modules installed, open a python interrupter and try to import them:
  - import requests
  - import tabulate
  - import futures
  - import six
- Check to make sure h2o module is installed
  - import h2o
  - h2o.init(nthreads = -1)

Getting Started with H2O

Objective

- Learn how R, Flow, and Python sends commands to compute in H2O
- FAQ on writing R, Flow, and Python expressions
- Hands on introduction into data science
- Understanding model outputs
- Note the limitations of the basic workflow to improve upon later

Reading Data into H2O with R

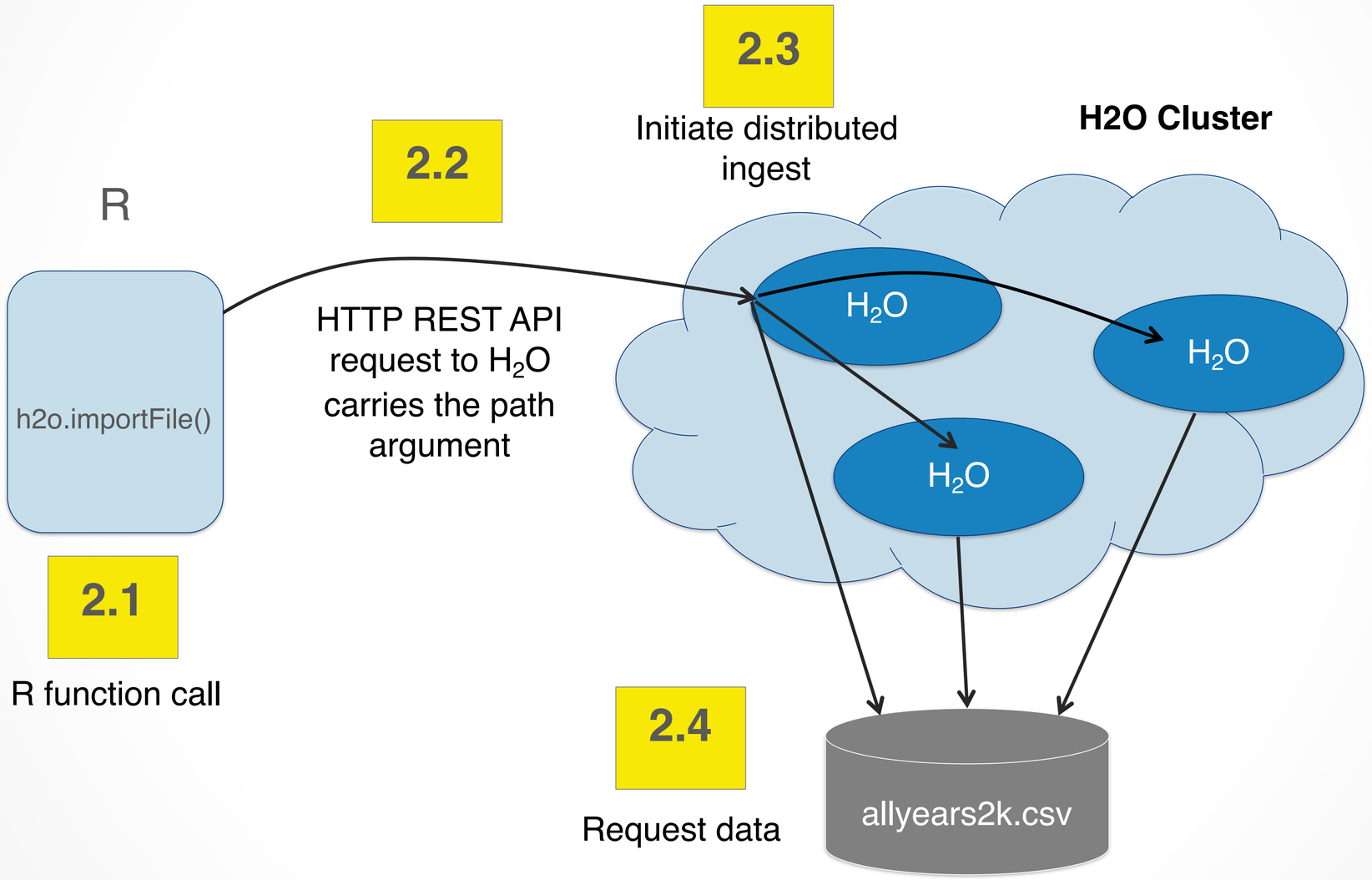
STEP 1

R User

```
h2o_df = h2o.importFile("../data/allyears2k.csv")
```

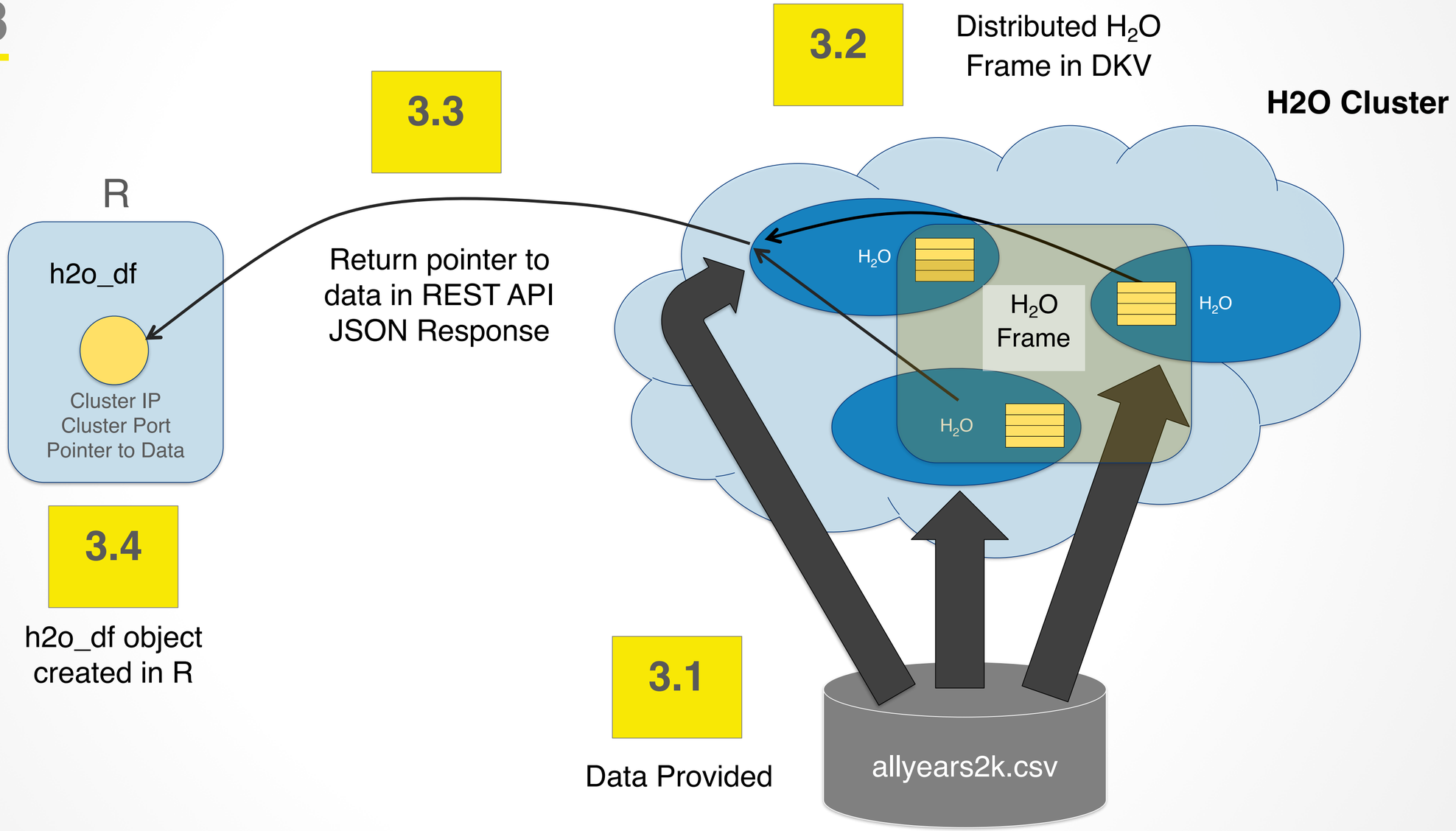
Reading Data into H2O with R

STEP 2



Reading Data into H2O with R

STEP 3



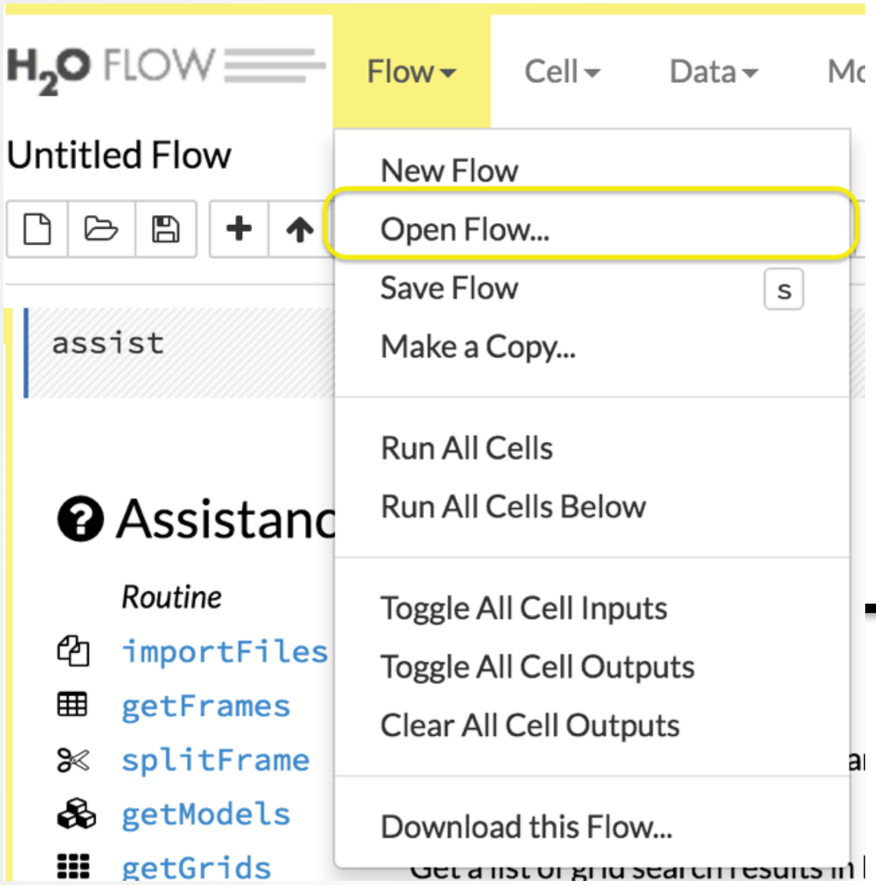
R Parity

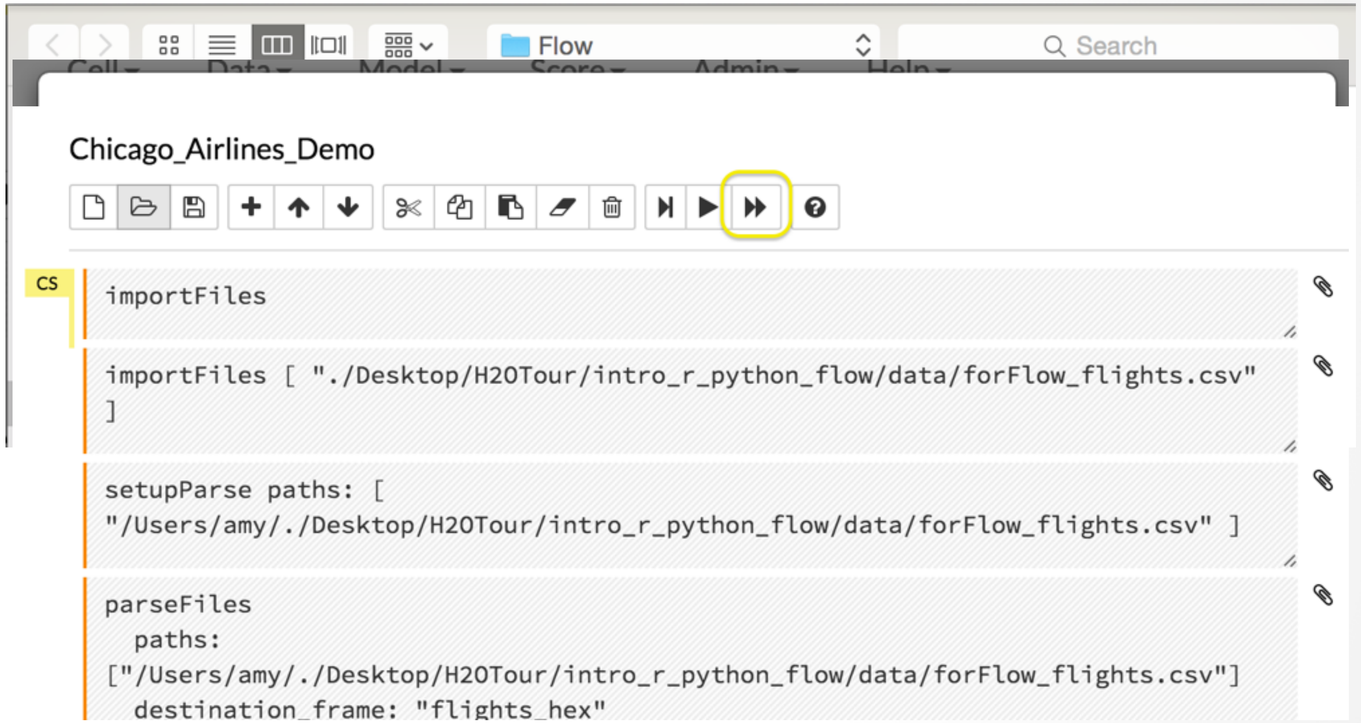
	Standard R	R on H2O
Reading in data	read.csv, read.table, etc	h2o.importFile
Summarizing data	summary	summary, h2o.summary
Combining rows or columns	cbind, rbind	h2o.cbind, h2o.rbind
Unary or Binary Operations	+, -, *, /, ^, %%, %/%, etc	+, -, *, /, ^, %%, %/%, etc
Building GLM model	glm, glmnet	h2o.glm
Predict with Model	predict	h2o.predict
Obtaining Metrics	auc, mse, logLoss, etc	h2o.auc, h2o.mse, h2o.logloss
Unsupported Functions	smooth	UNSUPPORTED

Hands on Introduction to Running H2O

- Import & parse a small row airlines dataset
- Run a logistic regression
- Build a gradient boosting machine model
- Review the model outputs
- Join the flight data with weather data
- Rebuild logistic regression and GBM model
- Review new model outputs

Load up an example Flow from Disk





Starting up H2O and Preloaded Workbook

Flow Users

From terminal, change the directory to where h2o.jar file is sitting and run:

```
> java -jar h2o.jar
```

Then access the Flow UI at:

https://localhost:54321

R Users

Open the intro-to-r.md.R file and run from R (Native R or Rstudio):

```
library(h2o)
h2o.init(nthreads = -1)
```

Python Users

Open either intro-to-python.ipynb or intro-to-python.py with python:

```
import h2o
import …
h2o.init()
```

Import Airlines Data into H2O

Flow Users

```
importFiles [ "./Desktop/H2OTour/intro_r_python_flow/data/forFlow_flights.csv" ]

setupParse paths: ["./Desktop/H2OTour/intro_r_python_flow/data/
forFlow_flights.csv" ]
…
```

R Users

```
flights_hex = h2o.importFile(path = flights_path, destination_frame =
"flights_hex")
weather_hex = h2o.importFile(path = weather_path, destination_frame
= "weather_hex")
```

Python Users

```
flights_hex = h2o.import_file(path = flights_path, destination_frame =
"flights_hex")
weather_hex = h2o.import_file(path = weather_path,
destination_frame = "weather_hex")
```

The Airlines Data

- Goal: To predict departure delays using historical airlines data with Destination to Chicago O'Hare Airport or ORD
- Enumerator columns: Origin, and Unique Carrier have a cardinality of 132 and 10 respectively
- Numeric Columns: DayOfMonth, Year, DayOfWeek, Month, and Distance
- Binary Response Column: IsArrDelayed

Build a Logistic Regression Model

Flow Users

```
buildModel 'glm', {"model_id":"glm_model","training_frame":"allyears2k.hex","ignored_columns":
["DayofMonth","DepTime","CRSDepTime","ArrTime","CRSArrTime","TailNum","ActualElapsedTime","CRSElapsedTime"
,"AirTime","ArrDelay","DepDelay","TaxiIn","TaxiOut","Cancelled","CancellationCode","Diverted","CarrierDela
y","WeatherDelay","NASDelay","SecurityDelay","LateAircraftDelay","IsArrDelayed"],"ignore_const_cols":true,
"response_column":"IsDepDelayed","family":"binomial","solver":"IRLSM","alpha":[0.5],"lambda":
[0.00001],"lambda_search":false,"standardize":true,"non_negative":false,"score_each_iteration":false,"max_
iterations":-1,"link":"family_default","intercept":true,"objective_epsilon":0.00001,"beta_epsilon":0.0001,
"gradient_epsilon":0.0001,"prior":-1,"max_active_predictors":-1}
```

4.1s

R Users

```
y <- "IsDepDelayed"
x <- c("Dest", "Origin", "DayofMonth", "Year", "UniqueCarrier", "DayOfWeek", "Month",
"Distance")
arr_delay_glm = h2o.glm(x = myX,y = myY, training_frame = train, validation_frame = valid,
family = "binomial", alpha = 0.5,lambda_search = T)
```

Python Users

```
myY = "IsDepDelayed"
myX = ["Dest", "Origin", "DayofMonth", "Year", "UniqueCarrier", "DayOfWeek", "Month", "Distance"]
glm_model = H2OGeneralizedLinearEstimator(family = "binomial",standardize = True, alpha = 0.5,
lambda_search = True )
glm_model.train( x = myX, y = myY, training_frame  = train, validation_frame = valid)
```

GLM Model Output

Model

Model ID: glm-b59c111a-cf92-48d1-b920-79047c3d659f

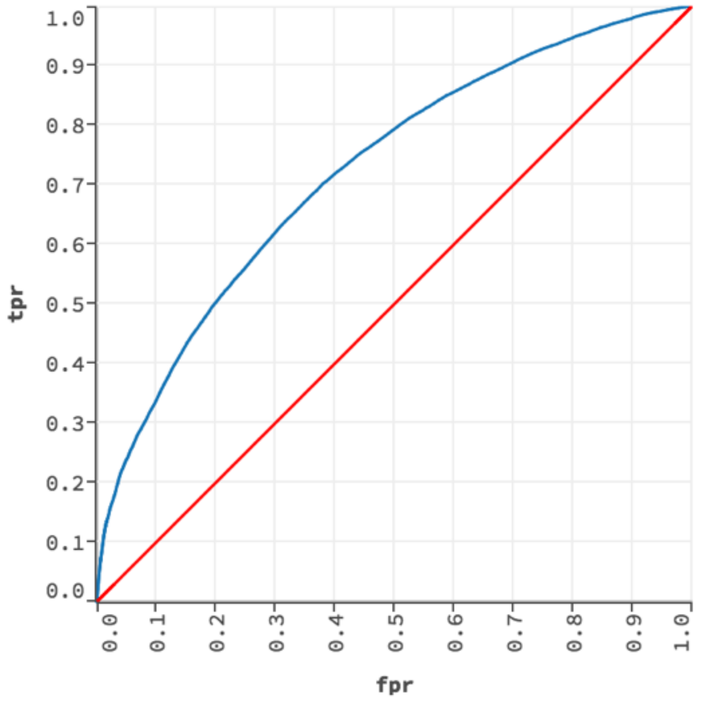
Algorithm: Generalized Linear Modeling

Actions: Refresh | Predict... | Download POJO | Export | Inspect | Delete

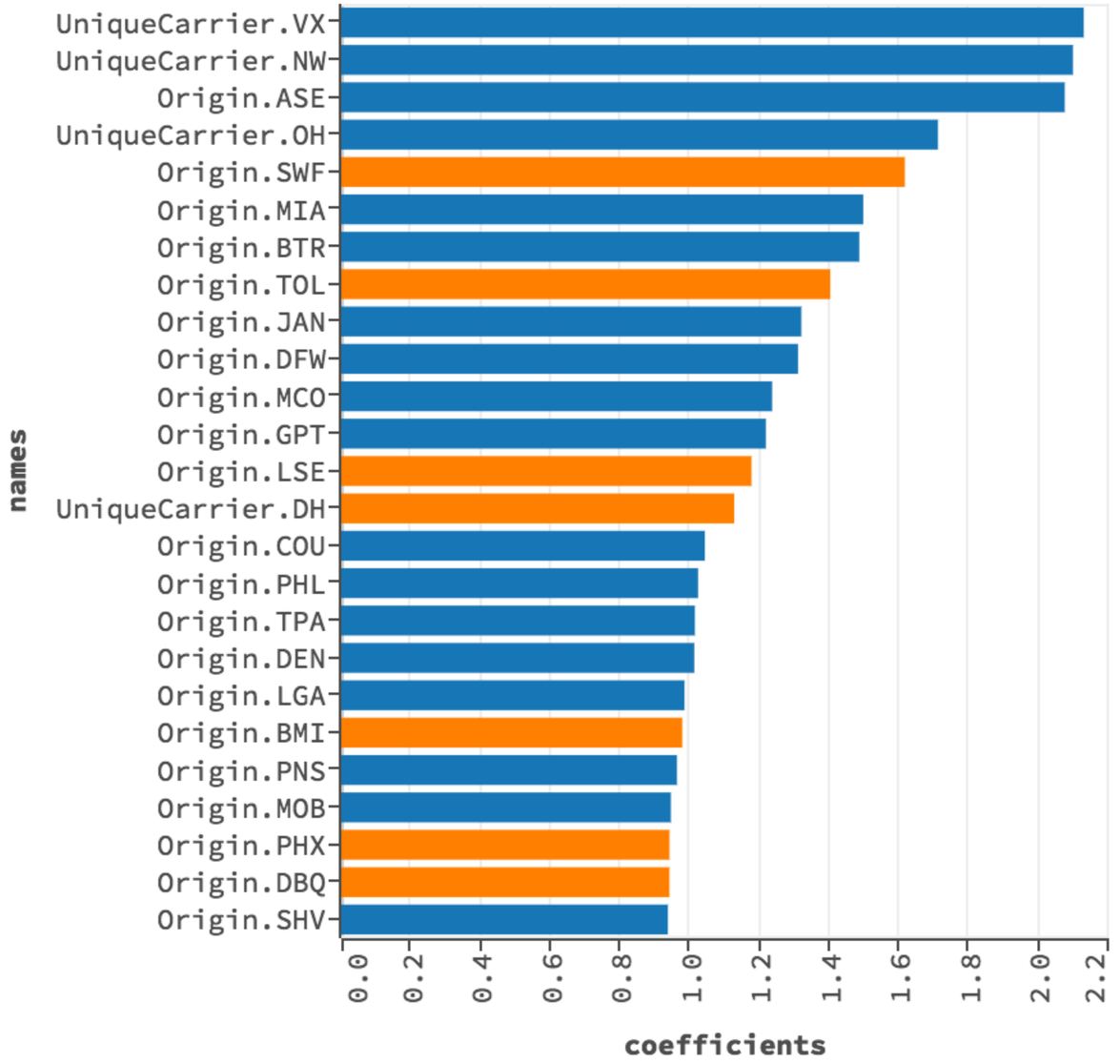
▸ MODEL PARAMETERS

▸ SCORING HISTORY

▾ ROC CURVE - TRAINING METRICS , AUC = 0.722933



▾ STANDARDIZED COEFFICIENT MAGNITUDES



Build a GBM Model

Flow Users

```
buildModel 'gbm', {"model_id":"gbm-ec9ed500-382b-4766-8af4-
dc040ab0f6a5","training_frame":"train","validation_frame":"valid","nfolds":0,"response_column":"IsArrDelayed","igno
red_columns":
["DepTime","CRSDepTime","ArrTime","CRSArrTime","FlightNum","TailNum","ActualElapsedTime","CRSElapsedTime","AirTime"
,"ArrDelay","DepDelay","Dest","TaxiIn","TaxiOut","Cancelled","CancellationCode","Diverted","CarrierDelay","WeatherD
elay","NASDelay","SecurityDelay","LateAircraftDelay","IsDepDelayed", "IsWeatherDelayedNumeric",
"IsArrDelayedNumeric"],"ignore_const_cols":true,"ntrees":50,"max_depth":5,"min_rows":10,"nbins":20,"seed":-1,"learn
_rate":0.1,"distribution":"AUTO","sample_rate":1,"col_sample_rate":1,"score_each_iteration":false,"score_tree_inter
val":0,"balance_classes":false,"nbins_top_level":1024,"nbins_cats":1024,"r2_stopping":0.999999,"stopping_rounds":0,
"stopping_metric":"AUTO","stopping_tolerance":0.001,"max_runtime_secs":0,"learn_rate_annealing":1,"checkpoint":"","
col_sample_rate_per_tree":1,"min_split_improvement":0.00001,"histogram_type":"AUTO","build_tree_one_node":false,"sa
mple_rate_per_class":[],"col_sample_rate_change_per_level":1,"max_abs_leafnode_pred":1.7976931348623157e+308}
```

R Users

```
arr_delay_gbm = h2o.gbm(x = myX,
                        y = myY,
                        training_frame = train,
                        validation_frame = valid,
                        distribution = "bernoulli",
                        ntrees = 50)
```

Python Users

```
arr_delay_gbm = H2OGradientBoostingEstimator(distribution   ="bernoulli",  ntrees =50)
arr_delay_gbm.train(x=myX,y=myY,training_frame  =train,validation_frame=valid)
```

GBM Model Output

Model

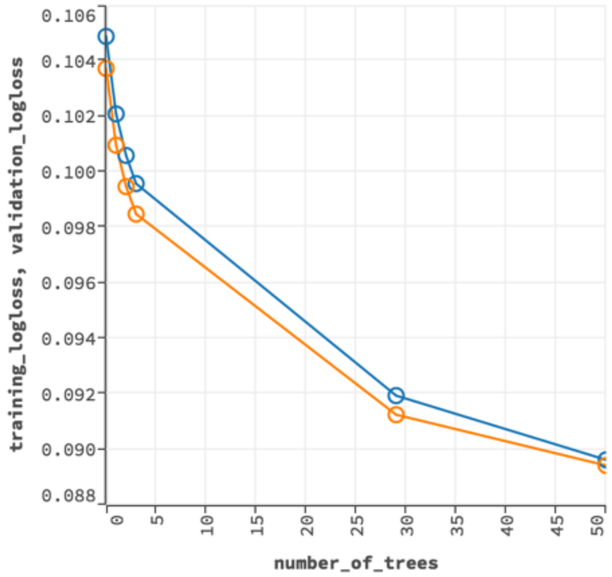
Model ID: gbm-ec9ed500-382b-4766-8af4-dc040ab0f6a5

Algorithm: Gradient Boosting Method

Actions: Refresh | Predict... | Download POJO | Export | Inspect | Delete

▸ MODEL PARAMETERS

▾ SCORING HISTORY - LOGLOSS



Model

Model ID: gbm-ec9ed500-382b-4766-8af4-dc040ab0f6a5

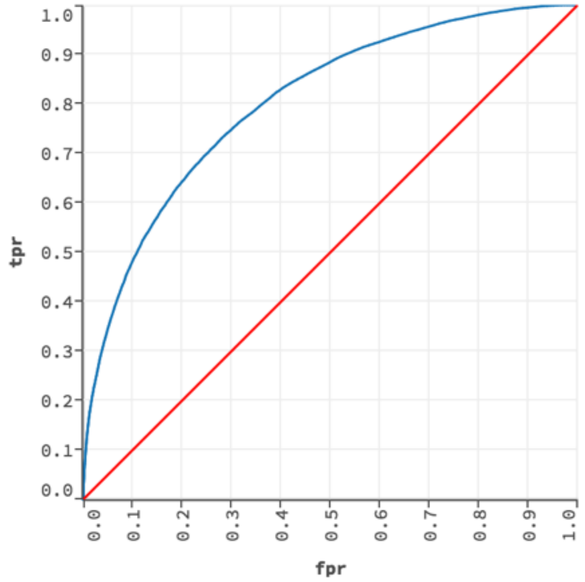
Algorithm: Gradient Boosting Method

Actions: Refresh | Predict... | Download POJO | Export | Inspect | Delete

▸ MODEL PARAMETERS

▸ SCORING HISTORY - LOGLOSS

▾ ROC CURVE - TRAINING METRICS , AUC = 0.804336



Join with Weather Data

- Join Chicago weather data with flights data
- Many numeric columns featuring weather information including min, max, mean for temperature, humidity, visibility, windspeed, and precipitation
- Categorical column ‘Event’ with 18 levels indicating presence of fog, rain, snow, or thunderstorm

GLM Model With Weather Output

Model

Model ID: arr_delay_weather_glm

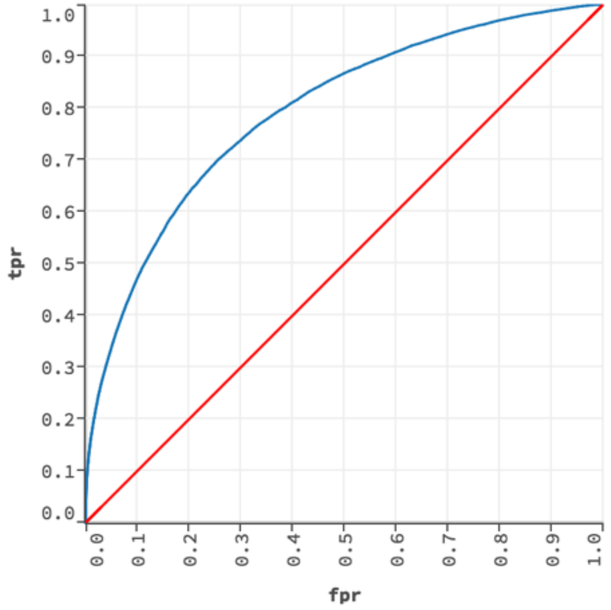
Algorithm: Generalized Linear Modeling

Actions: Refresh | Predict... | Download POJO | Export | Inspect | Delete

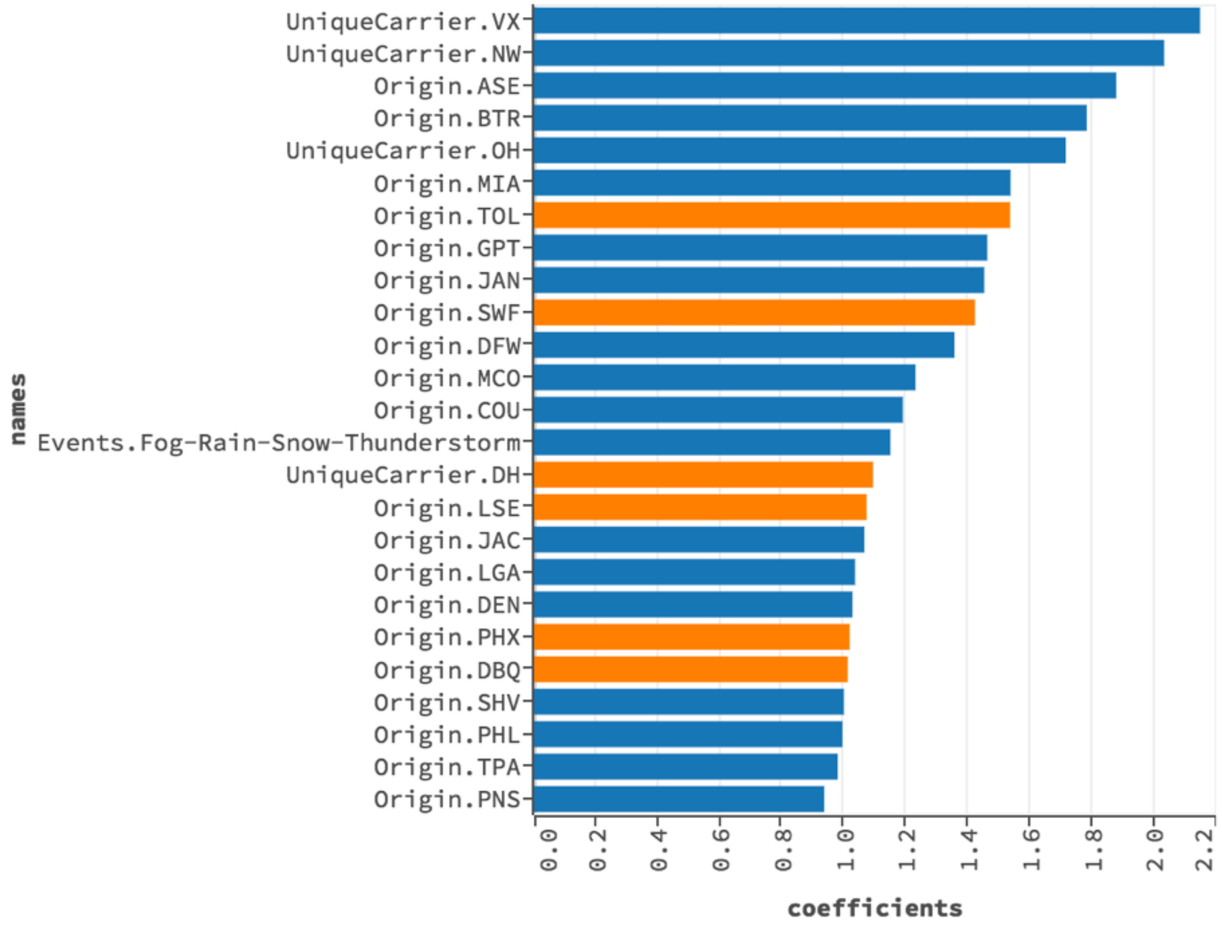
▸ MODEL PARAMETERS

▾ SCORING HISTORY

▾ ROC CURVE - TRAINING METRICS , AUC = 0.793394



▾ STANDARDIZED COEFFICIENT MAGNITUDES



GBM Model With Weather Output

Model

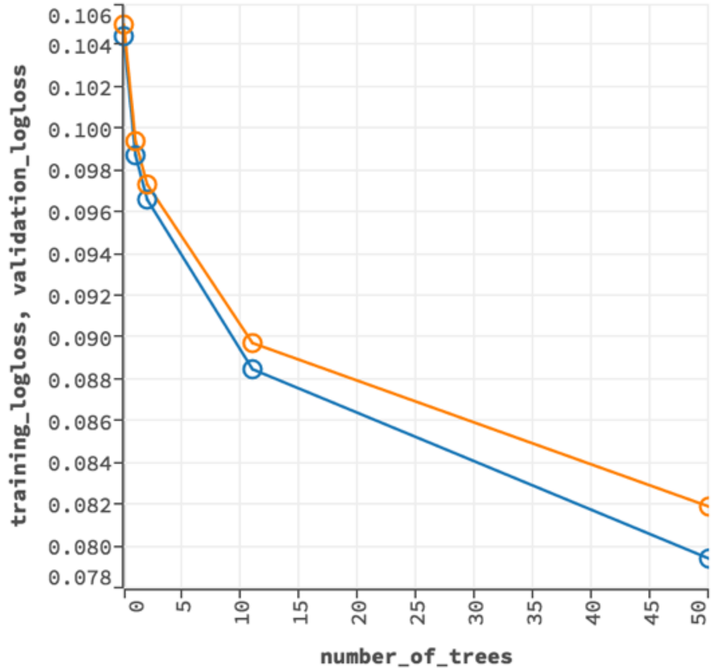
Model ID: arr_delay_weather_gbm

Algorithm: Gradient Boosting Method

Actions: Refresh | Predict... | Download POJO | Export | Inspect | Delete

▸ MODEL PARAMETERS

▾ SCORING HISTORY - LOGLOSS



Model

Model ID: arr_delay_weather_gbm

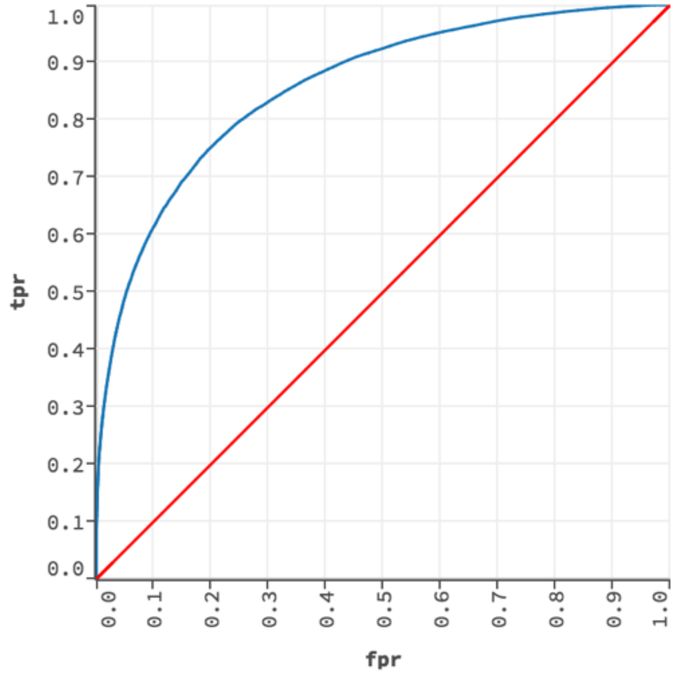
Algorithm: Gradient Boosting Method

Actions: Refresh | Predict... | Download POJO | Export | Inspect | Delete

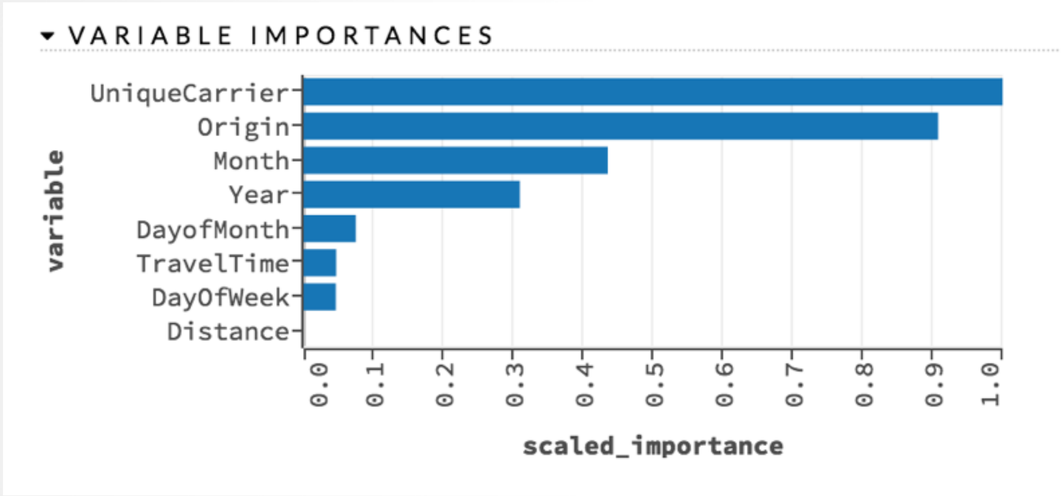
▸ MODEL PARAMETERS

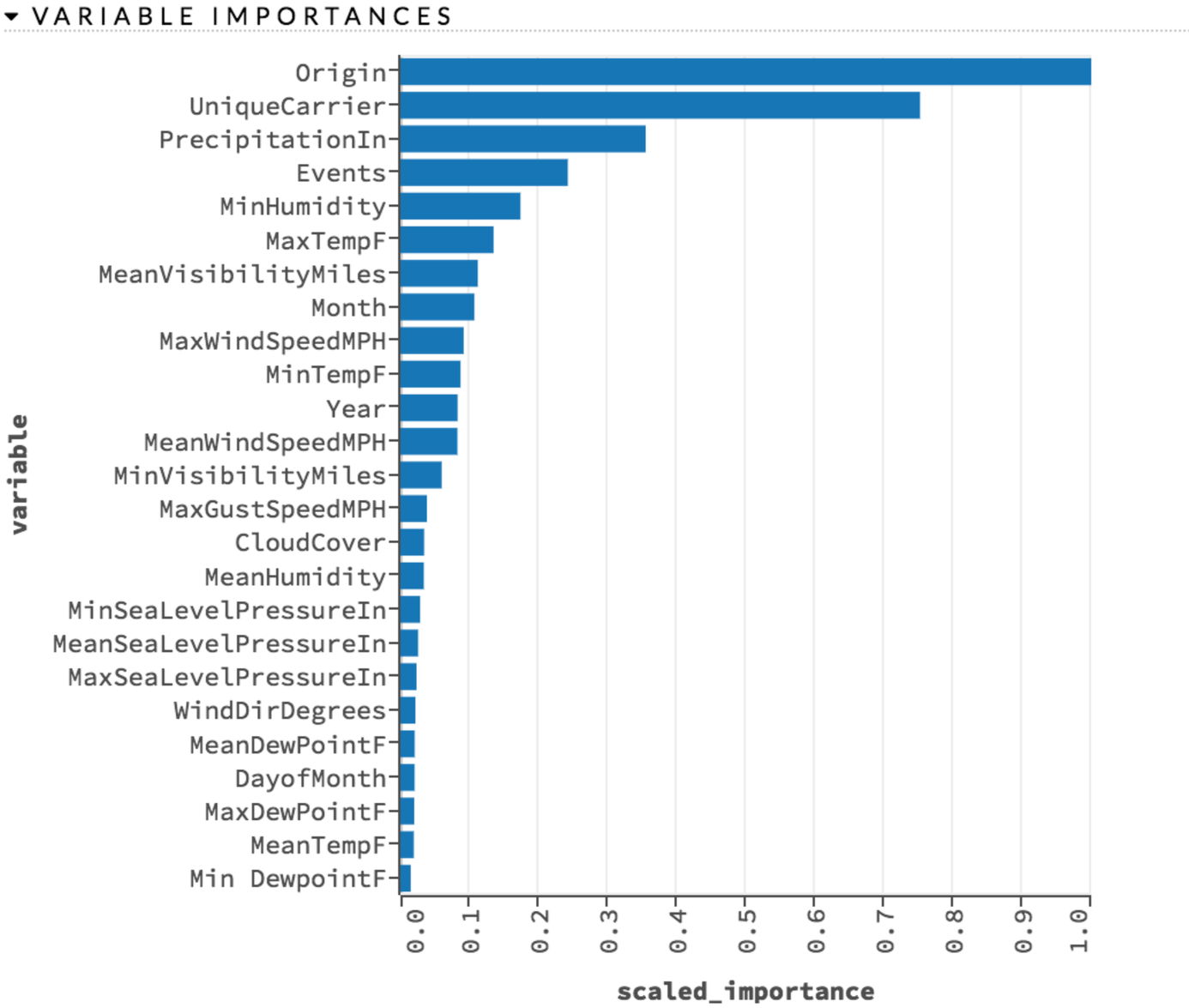
▸ SCORING HISTORY - LOGLOSS

▾ ROC CURVE - TRAINING METRICS , AUC = 0.857926



GBM Model Output - Variable Importance





Overview

- GBM models outperforms GLM with the same feature sets
- GLM models are more interpretable with beta values that relate each feature including expanded categorical levels to the response variable
- However you can combine what you know from a GBM model into a logistic regression model but creating interaction features using leaves from the GBM model
- There was a improvement when adding weather data for Chicago
- However the models can be further improved by grabbing weather data for all airports and doing a join with Origin in addition to weather of the destination

Overview

- Write R and Python expression to clean and munge the data in a parallelized and distributed fashion in H2O.
- Automate model builds by writing R and Python code and because all frames and models are generate in H2O, it is also accessible from the Flow UI.
- From the Web UI, the user can readily access the POJO which is the H2O independent Java representation of the models.