Introduction to R, Python, and Flow

Amy Wang

amy@h2o.ai

H2O WORLD 2015

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

I have H2O Installed

I have R installed

I have Python installed

I have the H2O World data sets

Pick up stickers or get install help at the information booth

H2O WORLD

Reading Data into H2O with R

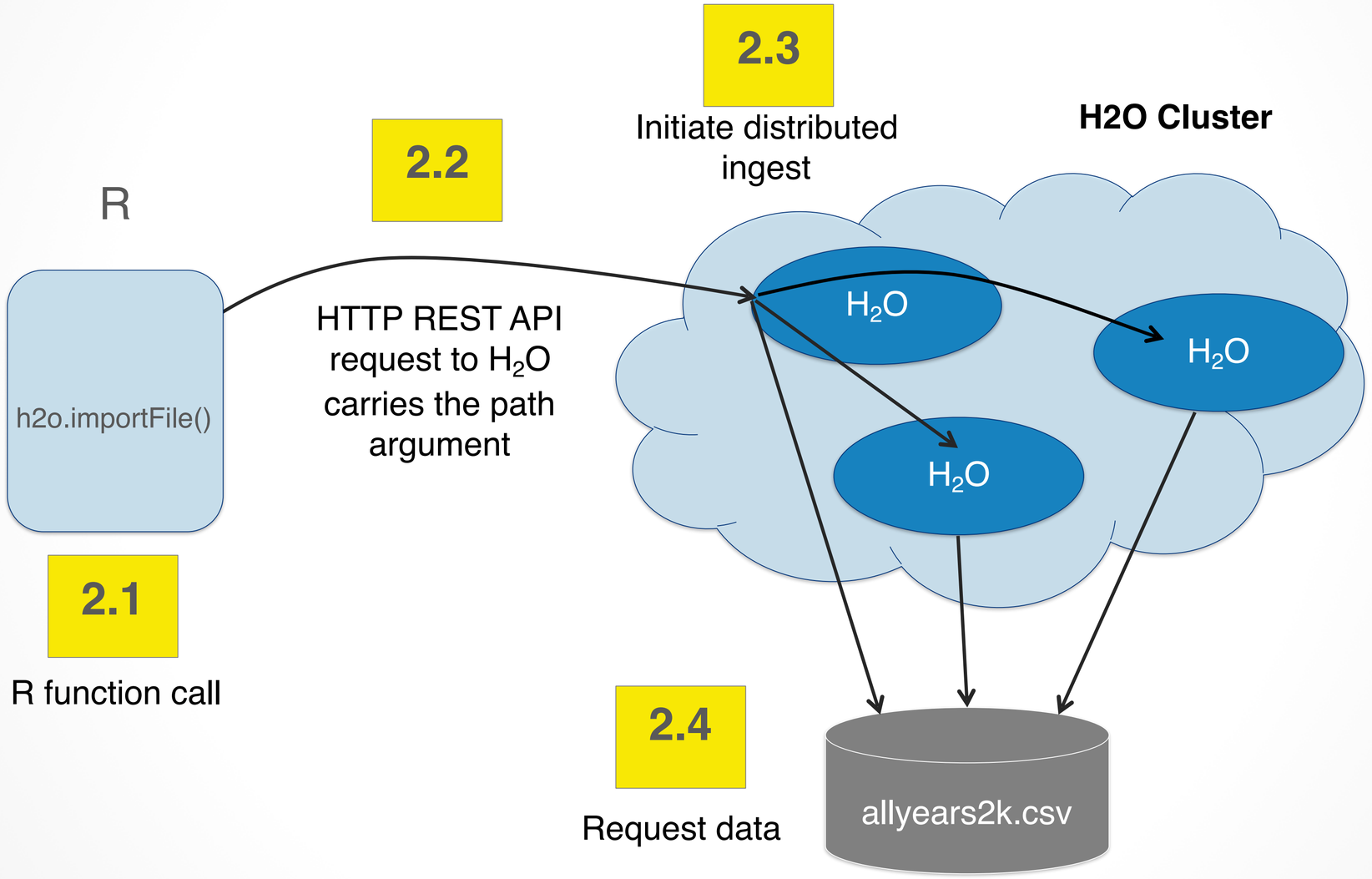
STEP 1

R User

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

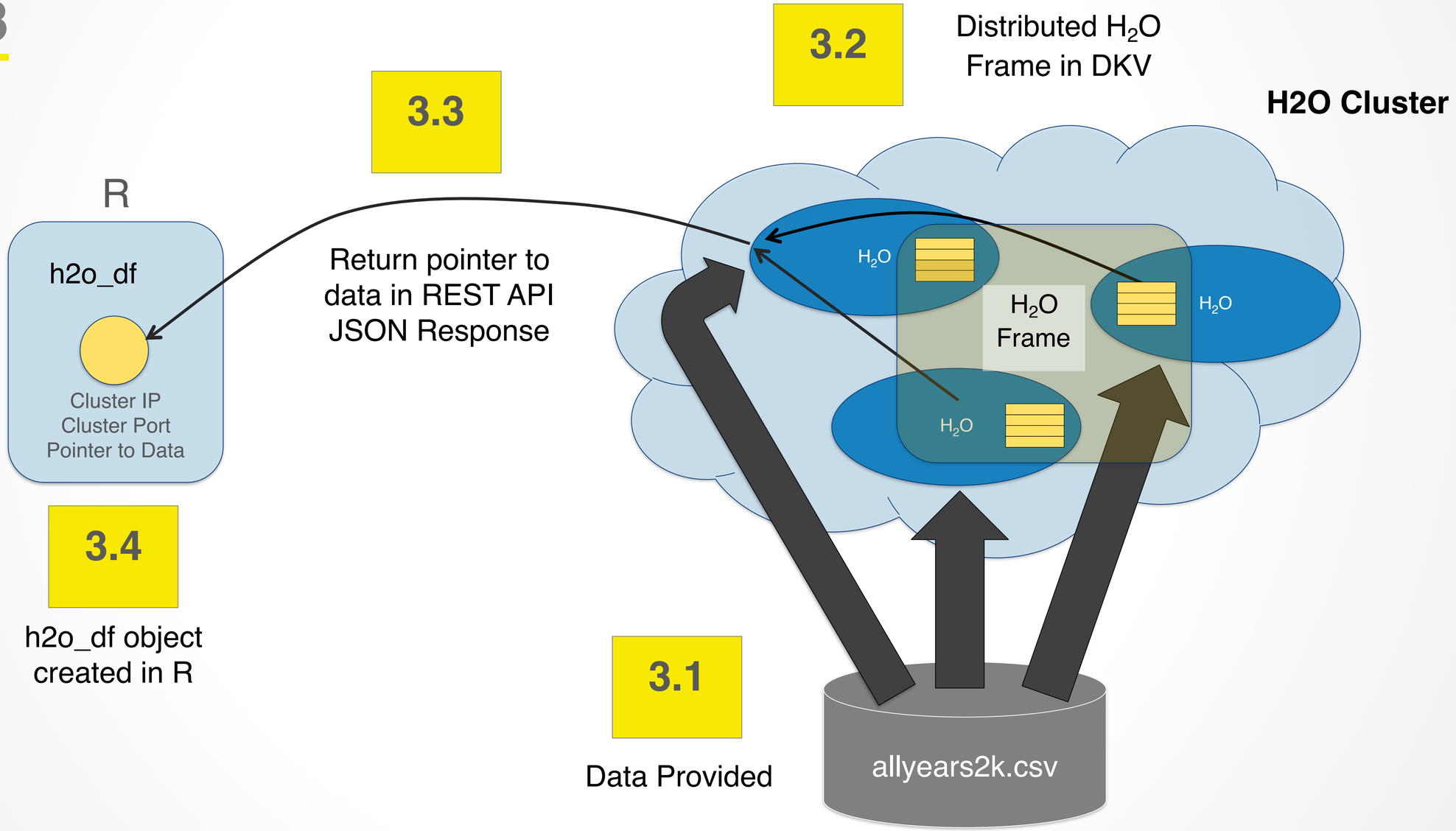
Reading Data into H2O with R

STEP 2

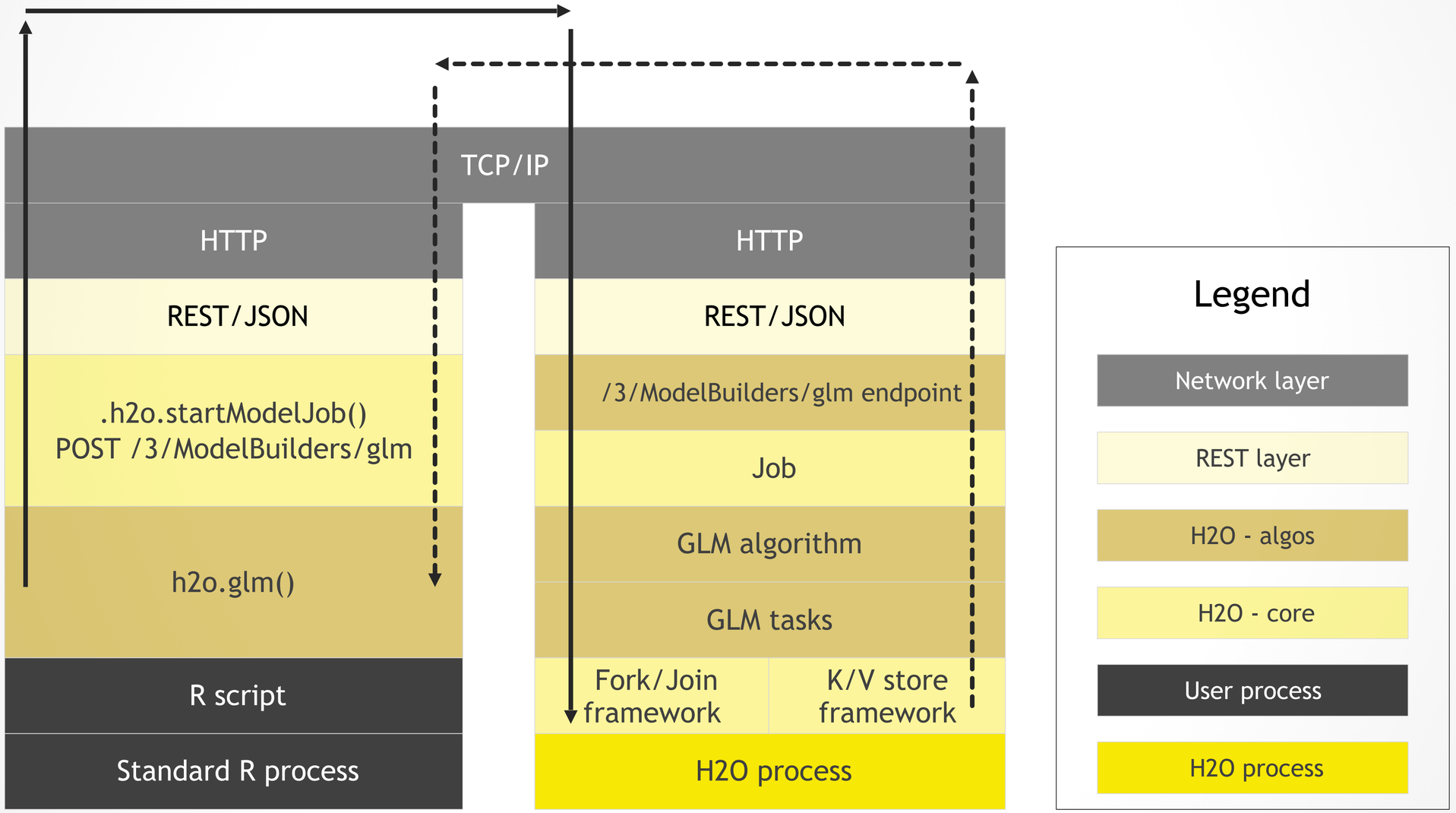


Reading Data into H2O with R

STEP 3



R Script Starting H2O GLM



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 44000 row airlines dataset
- Run a Logistic Regression
- Build a Deep Learning model
- Review the Model Outputs

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

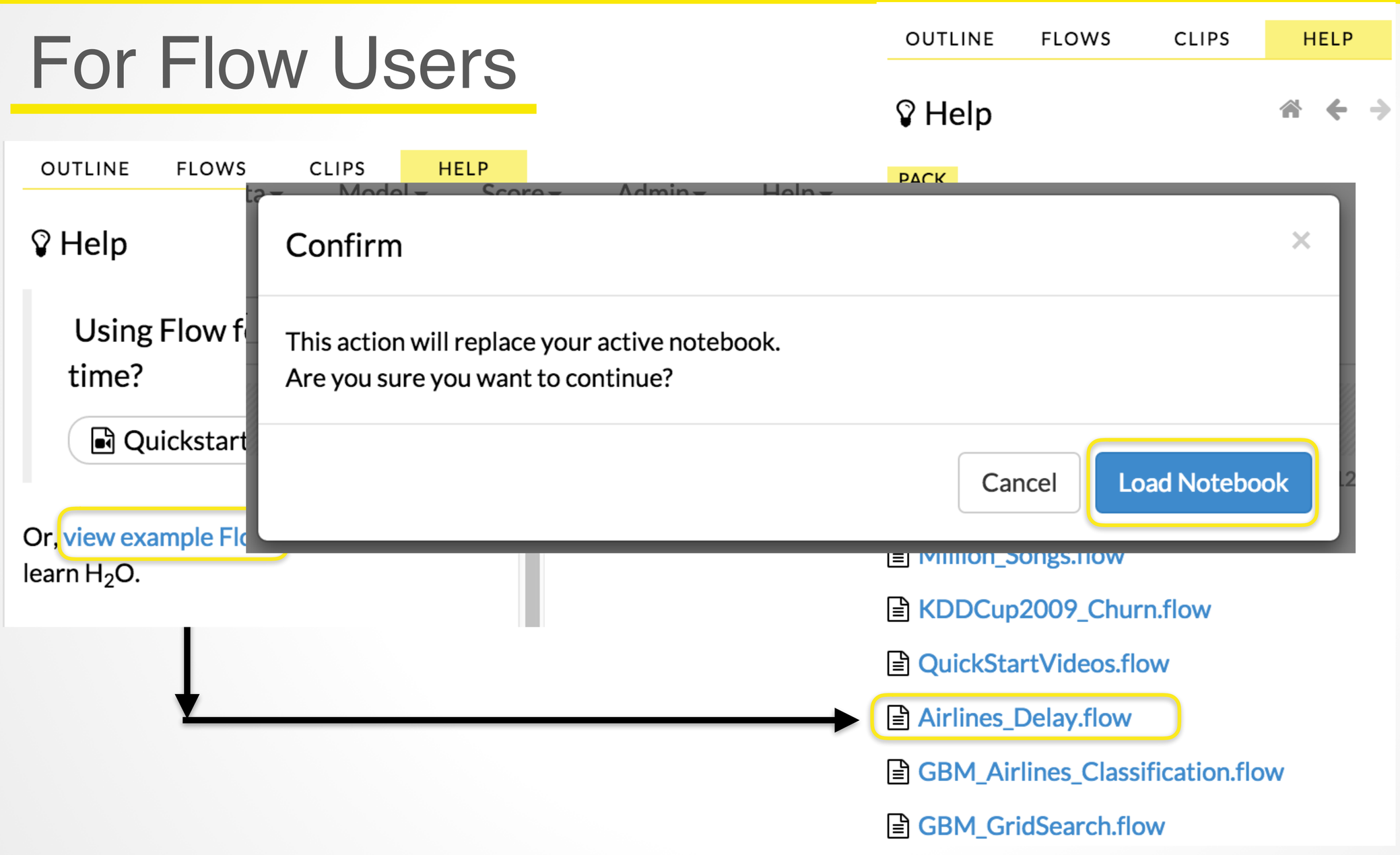
Open either intro-to-python.ipynb or intro-to-python.py with python:

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

H2O WORLD

Load up Preinstalled Flow Pack

For Flow Users



Import Airlines Data into H2O

Flow Users

```
importFiles [ "https://s3.amazonaws.com/h2o-airlines-unpacked/allyears2k.csv" ]

setupParse paths: [ "https://s3.amazonaws.com/h2o-airlines-unpacked/
allyears2k.csv" ]
…
```

R Users

```
airlines.hex <- h2o.importFile(path = normalizePath("../data/allyears2k.csv"),
destionation_frame = "allyearks2k.hex")
```

** Note: Make sure that the working directory is set to where your intro-to-r.md.R file is located otherwise the path to the data will be wrong.

Python Users

```
airlines.hex = h2o.import_file(path = os.path.realpath("../data/allyears2k.csv"),
destionation_frame = "allyearks2k.hex")
airlines.hex.describe()
```

H2O WORLD

The Airlines Data

- Goal: To predict departure delays using historical airlines data.
- Enumerator columns: Dest, Origin, and Unique Carrier have a cardinality of 134, 132, and 10 respectively
- Numeric Columns: DayOfMonth, Year, DayOfWeek, Month, and Distance
- Binary Response Column: IsDepDelayed

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")
glm_model <- h2o.glm(x = x, y = y, training_frame = airlines.hex, model_id = "glm_model_from_R",
                     solver = "IRLSM", standardize = T, link = "logit",
                     family = "binomial", alpha = 0.5, lambda = 1e-05)
```

Python Users

```
myY = “IsDepDelayed”
myX = ["Dest", "Origin", "DayofMonth", "Year", "UniqueCarrier", "DayOfWeek", "Month", “Distance"]
glm_model = H2OGeneralizedLinearEstimator(family = "binomial",standardize = True, solver = “IRLSM", link =
"logit", alpha = 0.5, model_id = "glm_model_from_python" )
glm_model.train( x = myX, y = myY, training_frame  = airlines_hex)
```

H2O WORLD

GLM Model Output

```
getModel "glm_model"
```

Model

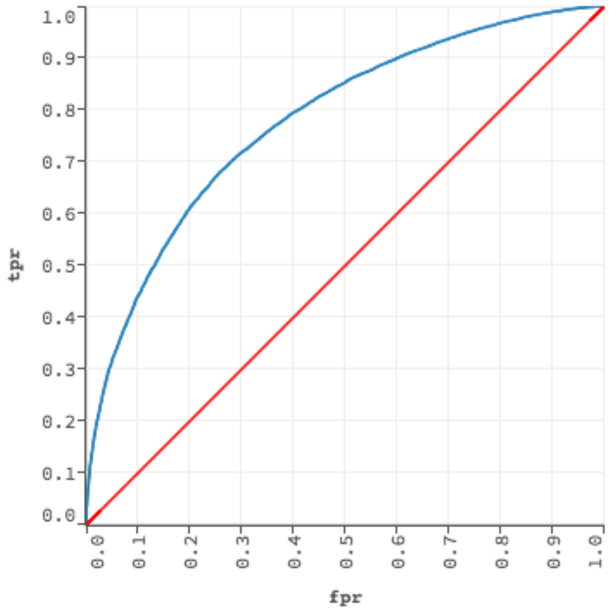
Model ID: glm_model

Algorithm: Generalized Linear Model

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

▸ MODEL PARAMETERS

▾ ROC CURVE - TRAINING METRICS , AUC = 0.779666



Threshold: Choose...

Criterion: Choose...

▾ OUTPUT - TRAINING_METRICS

model	glm_model
model_checksum	-6445631968972164096
frame	allyears2k.hex
frame_checksum	-2320848401324430336
description	·
model_category	Binomial
scoring_time	1447036418363
predictions	·
MSE	0.191083
r2	0.233752
logloss	0.562583
AUC	0.779666
Gini	0.559332
residual_deviance	49443.172387
null_deviance	60808.044803
AIC	53991.172387
null_degrees_of_freedom	43942
residual_degrees_of_freedom	41669

H2O WORLD

To Learn More About GLM

- Ramanujan Stage @ 10:45 AM
- Erdos Stage @ 2:15 PM

Build a Deep Learning Model

Flow Users

```
buildModel 'deeplearning',
{"model_id":"deeplearning_model","training_frame":"allyears2k.hex","ignored_columns":
["DepTime","CRSDepTime","ArrTime","CRSArrTime","FlightNum","TailNum","ActualElapsedTime","CRSElapsedTime","
AirTime","ArrDelay","DepDelay","TaxiIn","TaxiOut","Cancelled","CancellationCode","Diverted","CarrierDelay",
"WeatherDelay","NASDelay","SecurityDelay","LateAircraftDelay","IsArrDelayed"],"ignore_const_cols":true,"res
ponse_column":"IsDepDelayed","activation":"Rectifier","hidden":
[200,200],"epochs":"100","variable_importances":false,"balance_classes":false,"checkpoint":"","use_all_fact
```

R Users

```
dl_model <- h2o.deeplearning(x = x, y = y, training_frame = airlines.hex,
  distribution = “bernoulli”, model_id = "deeplearning_model_from_R", epochs = 100,
  hidden = c(200,200), target_ratio_comm_to_comp = 0.02, seed =
  6765686131094811000, variable_importances = T)
```

Python Users

```
deeplearning_model = H2ODeepLearningEstimator( distribution = "bernoulli",
  model_id = “deeplearning_model_from_python”, epochs = 100, hidden = [200,200], seed =
  6765686131094811000, variable_importances = True)
deeplearning_model.train(x = myX, y = myY, training_frame  = airlines_hex)
```

H2O WORLD

Deep Learning Model Output

Model

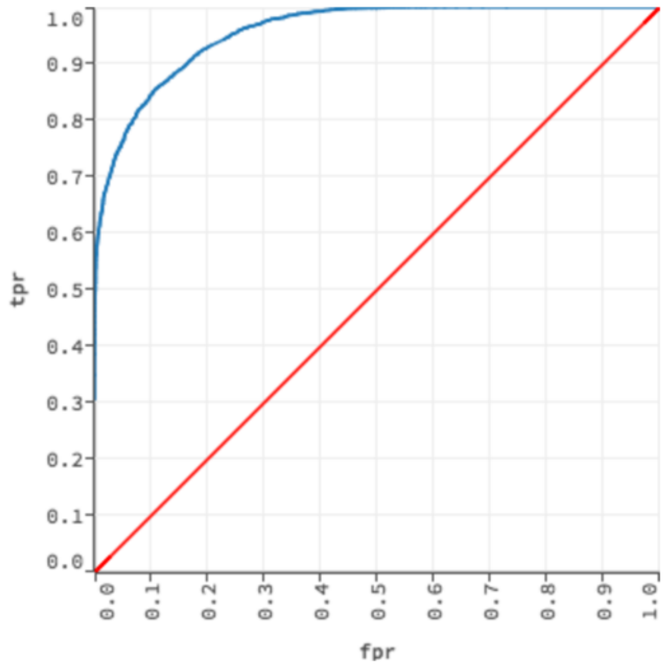
Model ID: deeplearning_model

Algorithm: Deep Learning

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

▸ MODEL PARAMETERS

▾ ROC CURVE - TRAINING METRICS , AUC = 0.957062



▾ OUTPUT - TRAINING_METRICS

model	deeplearning_model
model_checksum	-2685954226480812032
frame	·
frame_checksum	0
description	Metrics reported on temporary training frame with 10127 samples
model_category	Binomial
scoring_time	1447036547656
predictions	·
MSE	0.085315
r2	0.658471
logloss	0.257966
AUC	0.957062
Gini	0.914124

To Learn More About Deep Learning

- Ramanujan Stage @ 5:45 PM
- Erdos Stage @ 1:30 PM

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.