

H2O Training



H2O Machine Learning Methods

Supervised Learning

Statistical Analysis

- **Penalized Linear Models:** Super-fast, super-scalable, and interpretable
- **Naïve Bayes:** Straightforward linear classifier

Decision Tree Ensembles

- **Distributed Random Forest:** Easy-to-use tree-bagging ensembles
- **Gradient Boosting Machine:** Highly tunable tree-boosting ensembles
- **eXtreme Gradient Boosting:** Popular XGBoost algorithm in H2O

Stacking

- **Stacked Ensemble:** Combine multiple types of models for better predictions

AutoML

- **Automatic Machine Learning:** Automated exploration of supervised learning approaches

Unsupervised Learning

Clustering

- **K-means:** Partitions observations into similar groups; automatically detects number of groups

Dimensionality Reduction

- **Principal Component Analysis:** Transforms correlated variables to independent components
- **Generalized Low Rank Models:** Extends the idea of PCA to handle arbitrary data consisting of numerical, Boolean, categorical, and missing data

Aggregator

- **Aggregator:** Efficient, advanced sampling that creates smaller data sets from larger data sets

Neural Networks

Multilayer Perceptron

- **Deep neural networks:** Multi-layer feed-forward neural networks for standard data mining tasks

Deep Learning

- **Convolutional neural networks:** Sophisticated architectures for pattern recognition in images, sound, and text

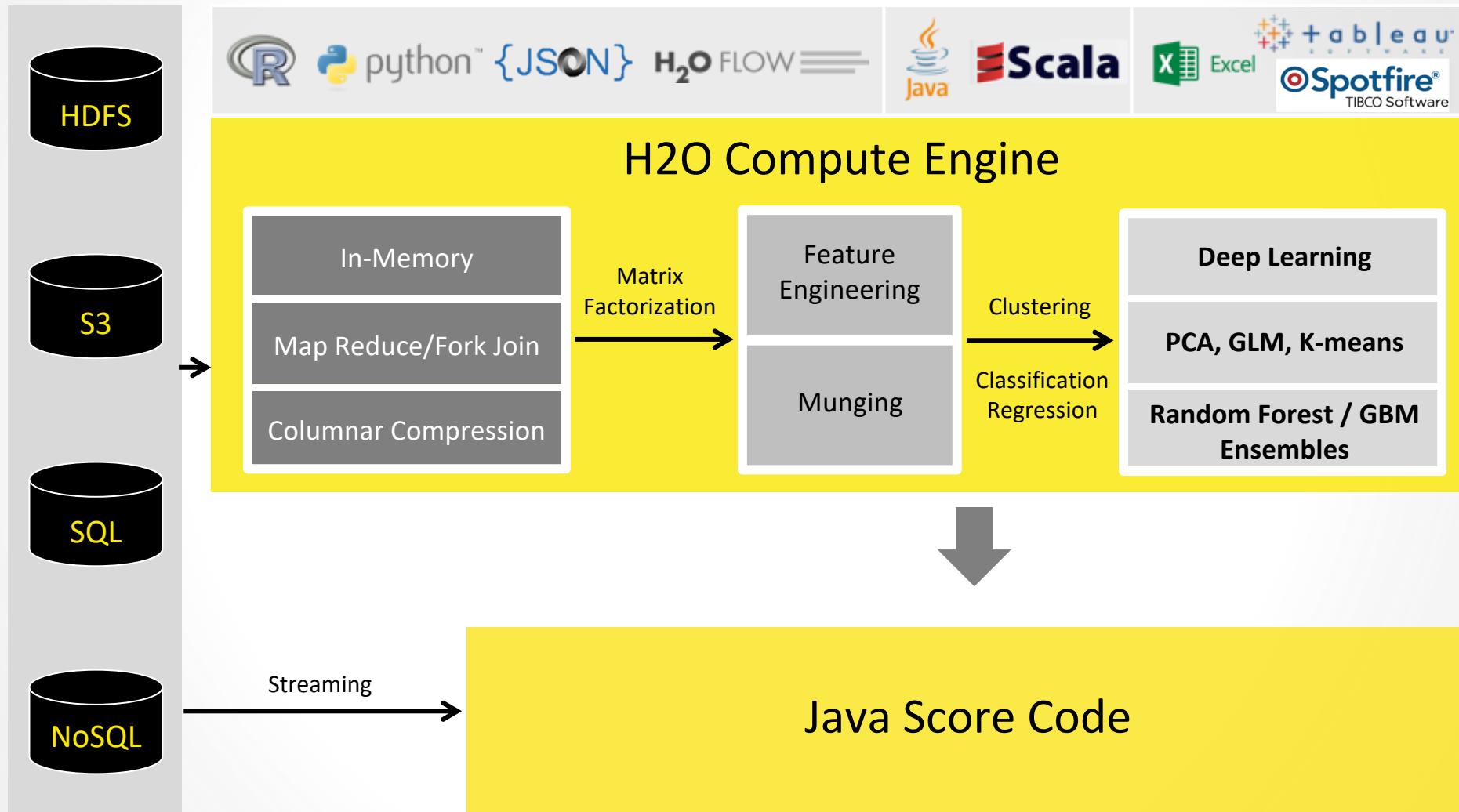
Anomaly Detection

- **Autoencoders:** Find outliers using a nonlinear dimensionality reduction technique

Term Embeddings

- **Word2vec:** Generate context-sensitive numerical representations of a large text corpus

Data and Client Agnostic



R, Python and Flow

RStudio

```
60 # Run GLM with variable importance, lambda search and using all factor levels
61 my.glm <- h2o.glm(x = myx, y = myy, training_frame = data.hex, family = "binomial", standardize = TRUE,
62                      lambda_search = TRUE)
63
64 # Select the best model picked by glm
65 # best_model <- my.glm$best_model
66
67 # Get the normalized coefficients of the best model
68 n_coeff <- abs(my.glm$models[[best_model]]$model$normalized_coefficients)
69 glm.vi <- my.glm$models[[best_model]]$model$variable_importance
70 print("Variable Importance from GLM")
71 print(glm.vi)
72 print(glm.vi)
73
74 # Plot variable importance from glm
75 barplot(glm.vi[1:20], coefficients, names.arg = glm.vi[1:20], names, las = 2, main = "VI from GLM")
76
77 # Run deeplearning with variable importance
78 my.dl <- h2o.deepLearning(x = myx, y = myy, training_frame = data.hex,
79                           activation = "Tanh", hidden = c(10, 10, 10),
80                           epochs = 12, variable_importances = TRUE)
81
82 # Access variable importance from the built model
83 print("Variable Importance from Deep Learning")
84 print(my.dl.varimp())
85
86 # Print variable importance from H2O
87 print(h2o.varimp())
88
89 # Print variable importance from GBM
90 print(gbm.vi)
91
92 # Print variable importance from RF
93 print(my.rf.varimp())
94
```

Environment History

Values

Files Plots Packages Help Viewer

VI from GBM VI from RF

Jupyter H2O_chicago_crimes (1) Last Checkpoint: 20 hours ago (unsaved changes)

In [7]:

```
deeplearning Model Build Progress: [########################################] 100%
In [7]: # GBM performance on train/test data
train_auc_gbm = data_gbm.model_performance(train).auc()
test_auc_gbm = data_gbm.model_performance(test).auc()

# Deep Learning performance on train/test data
# train_auc_dl = data_dl.model_performance(train).auc()
# test_auc_dl = data_dl.model_performance(test).auc()

# Make a pretty HTML table printout of the results
header = ["Model", "AUC Train", "AUC Test"]
table = [
    ["GBM", train_auc_gbm, test_auc_gbm],
    # ["DL", train_auc_dl, test_auc_dl]
]
h2o.display.H2ODisplay(table, header)
```

Out[7]:

Model	AUC Train	AUC Test
GBM	0.9568221	0.9307979
DL	0.8956055	0.8841564

In [8]:

```
# Create new H2OFrame of crime observations
examples = {
    "Date": ["02/08/2015 11:43:58 PM", "02/08/2015 11:00:39 PM"],
    "IUCR": [1811, 1150],
    "Primary.Type": ["NARCOTICS", "DECEPTIVE PRACTICE"],
```

H2O FLOW

K-Means_Example

Setup Parse

PARSE CONFIGURATION

Sources http://s3.amazonaws.com/h2o-public-test-data/smalldata/flow_examples/seeds_dataset.txt

ID Key_Frame__http__s3.amazonaws_com_h2o_public_test_data_smalldata_flow_examples_seeds_dataset.hex

Parser CSV

Separator HT '\t' (horizontal tab): '09'

Column Headers Auto First row contains column names First row contains data

Options Enable single quotes as a field quotation character Delete on done

EDIT COLUMN NAMES AND TYPES

Search by column name...

1	Numeric	15.26	14.88	14.29	13.84	16.14	14.38
2	Numeric	14.84	14.57	14.09	13.94	14.99	14.21
3	Numeric	0.871	0.8811	0.905	0.8955	0.9034	0.8951
4	Numeric	5.763	5.554	5.291	5.324	5.658	5.386
5	Numeric	3.312	3.333	3.337	3.379	3.562	3.312
6	Numeric	2.221	1.018	2.699	2.259	1.355	2.462
7	Numeric	5.22	4.956	4.825	4.805	5.175	4.956
8	Numeric	1	1	1	1	1	1

Previous page Next page

Ready

Python Interface Overview

Action	Pandas or scikit-learn	H2O
Reading data	<code>pandas.read_csv(data_path)</code>	<code>h2o.import_file(data_path)</code>
Summarizing data	<code>pandas_frame.describe()</code>	<code>h2o_frame.describe()</code>
Summary statistics	<code>pandas_frame.mean()</code>	<code>h2o_frame.mean()</code>
Combining rows	<code>pandas.concat(list[frame1,frame2])</code>	<code>h2o_frame.rbind(h2o_frame2)</code>
Combining columns	<code>pandas.concat(list[frame1,frame2],axis = 1)</code>	<code>h2o_frame.cbind(h2o_frame2)</code>
Data selection	<code>pandas_frame[:, :]</code>	<code>h2o_frame[:, :]</code>
Transforming columns	<code>np.log(pandas_frame[x])</code> <code>np.sqrt(pandas_frame[x])</code>	<code>h2o_frame[x].log()</code> <code>h2o_frame[x].sqrt()</code>
Building Random Forest	<code>model = RandomForestClassifier(n_estimators = 100)</code> <code>model = model.fit(x_frame, y_frame)</code>	<code>model = H2ORandomForestClassifier(n_trees = 100)</code> <code>model = model.train(x, y, train_frame)</code>
Model Prediction	<code>model.predict</code>	<code>model.predict</code>
Model Metrics	<code>metrics.auc</code>	<code>metrics = model.model_performance(frame)</code> <code>metrics.auc()</code>

Reading Data into H2O with Python

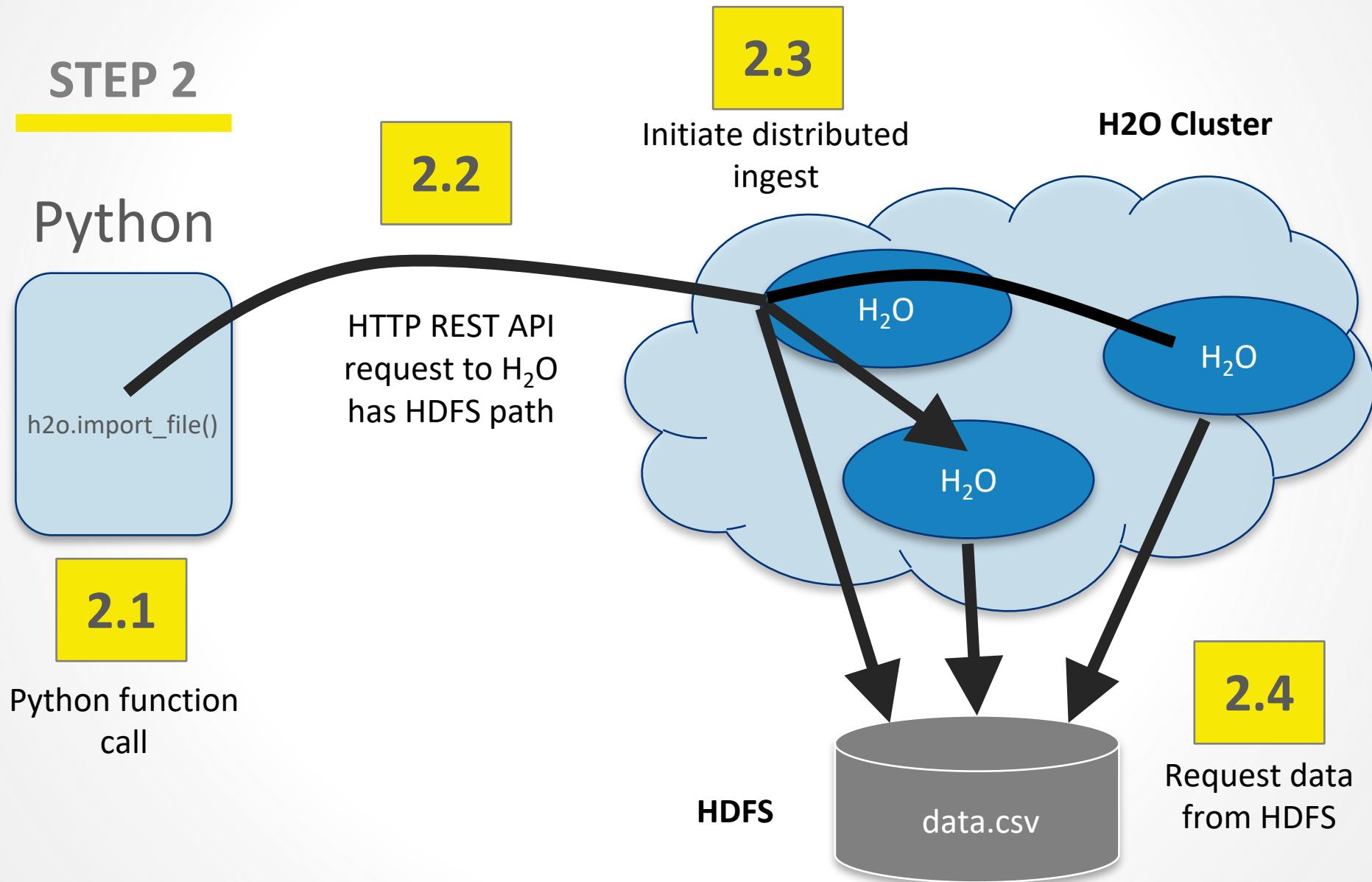
STEP 1



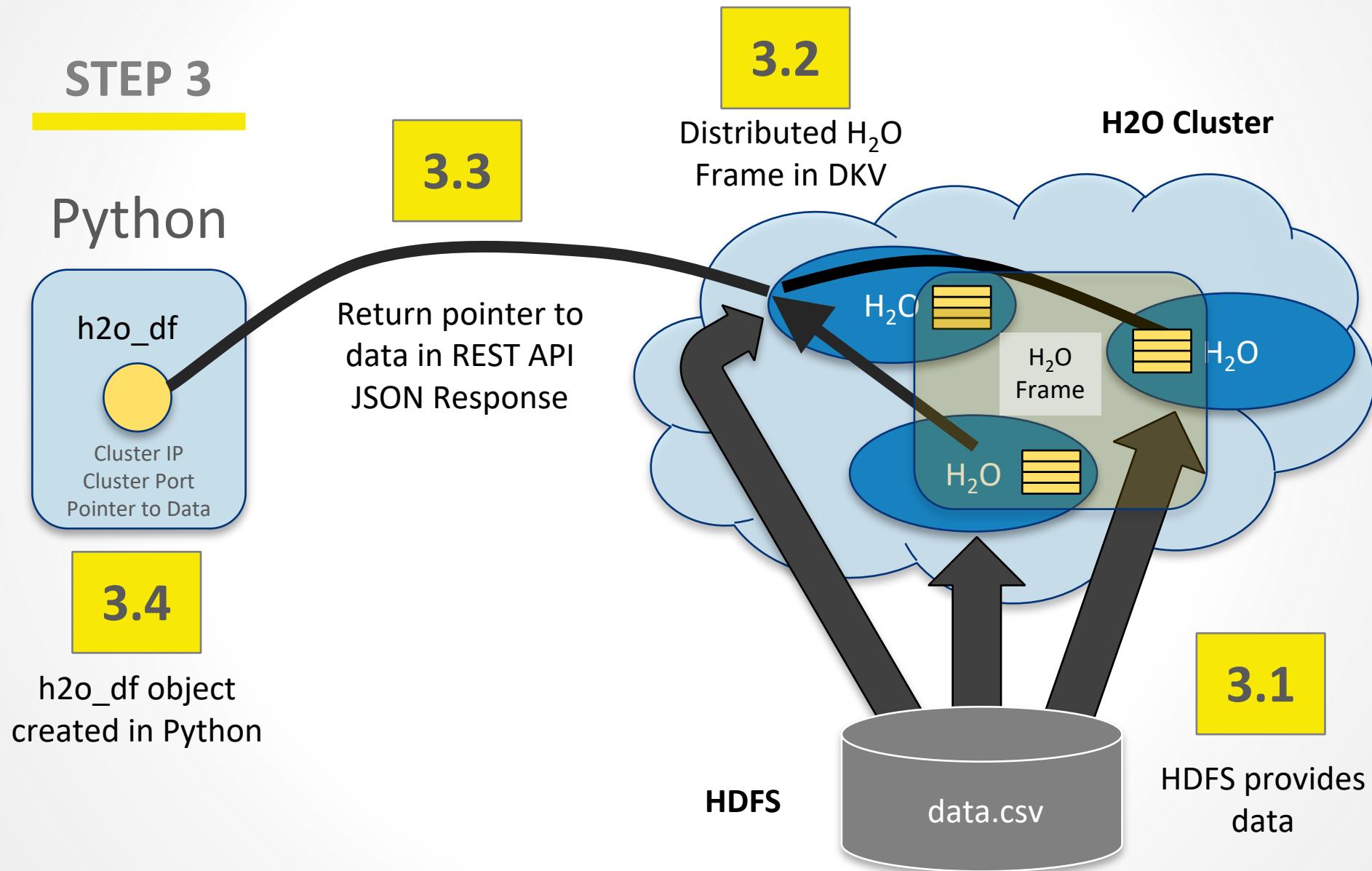
```
h2o_df = h2o.import_file("../data/allyears2k.csv")
```

Python
user

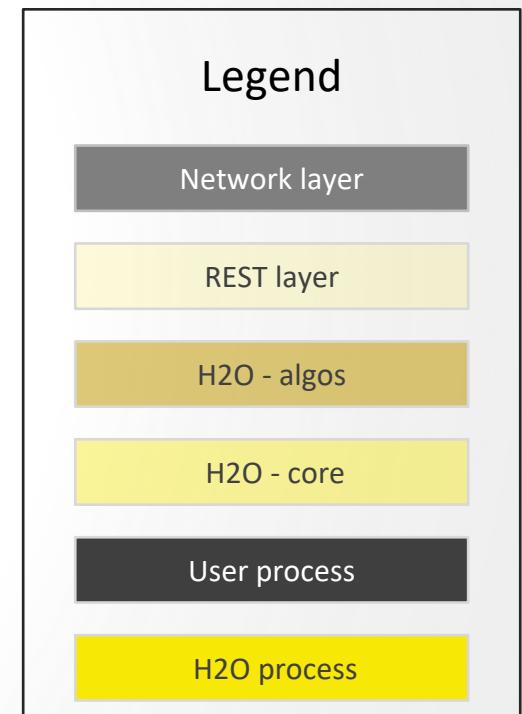
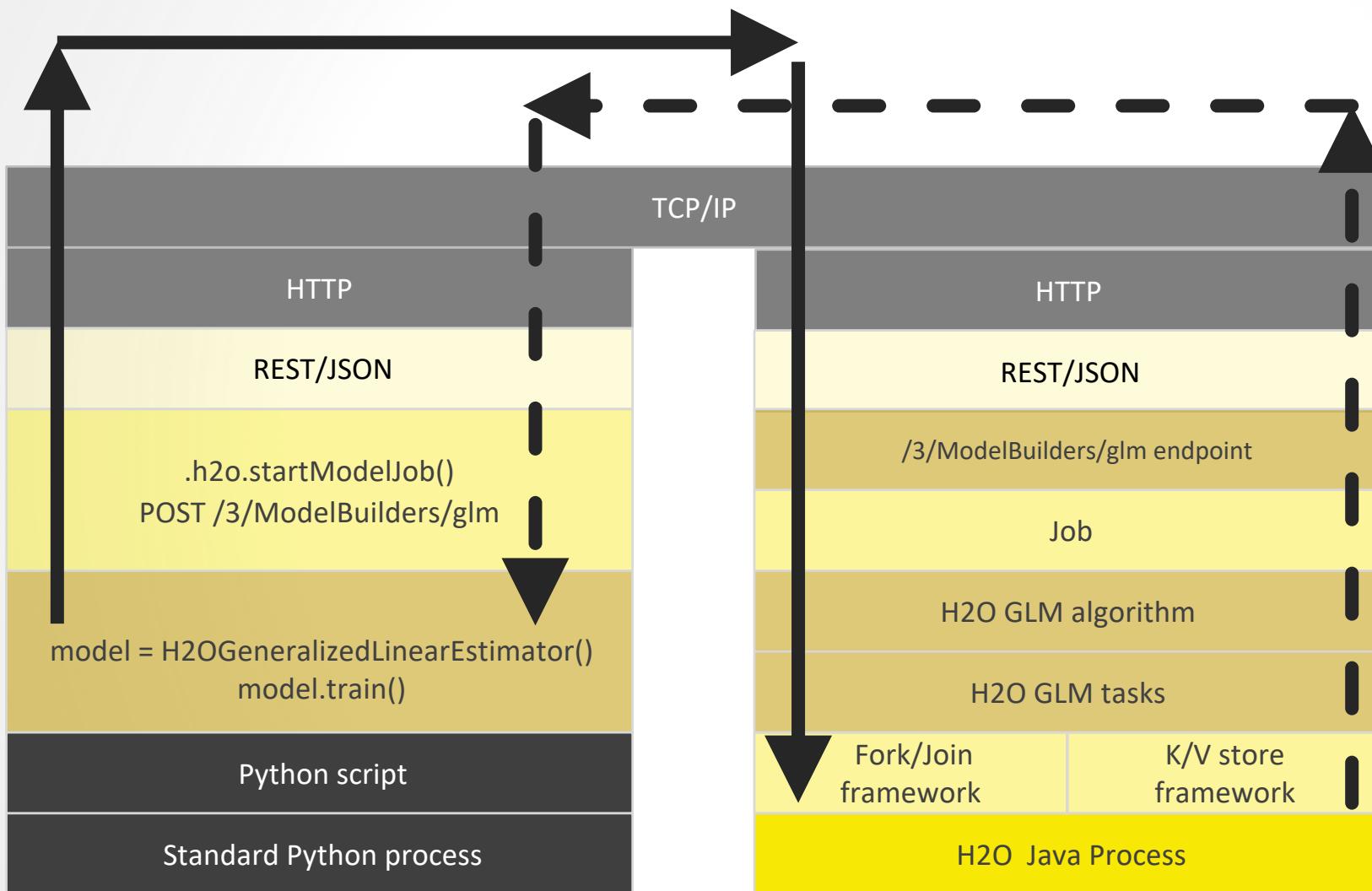
Reading Data into H2O with Python



Reading Data into H2O with Python



Training GLM from Python



Retrieving GLM Result from Python

