Instructions: Comparing models

In this exercise you will use H2O Flow to create and compare two predictive models built using our bank customer dataset.

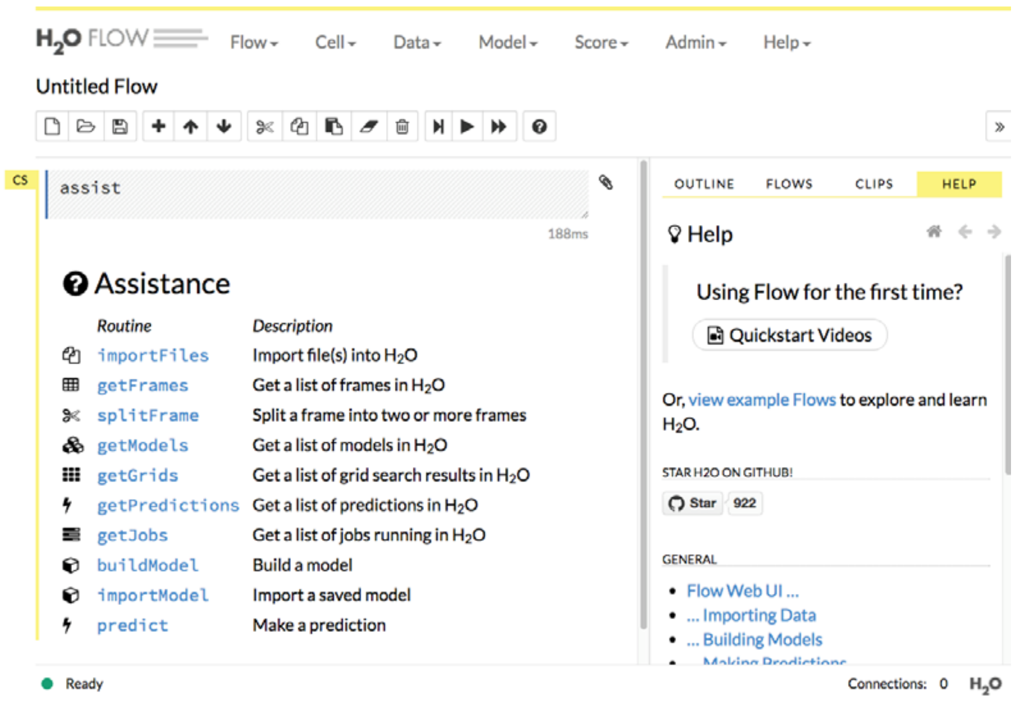
Note

This exercise continues on from the previous exercise: Which customers will accept offers? Refer to the instructions for this exercise if you have not initialised your local H2O server.

1) Open H2O Flow:

   a) Open your browser.

   b) Go to http://localhost:54321/flow/index.html.



```
assist
```

188ms

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

Using Flow for the first time?

Or, view example Flows to explore and learn H_2O.

STAR H2O ON GITHUB!

GENERAL

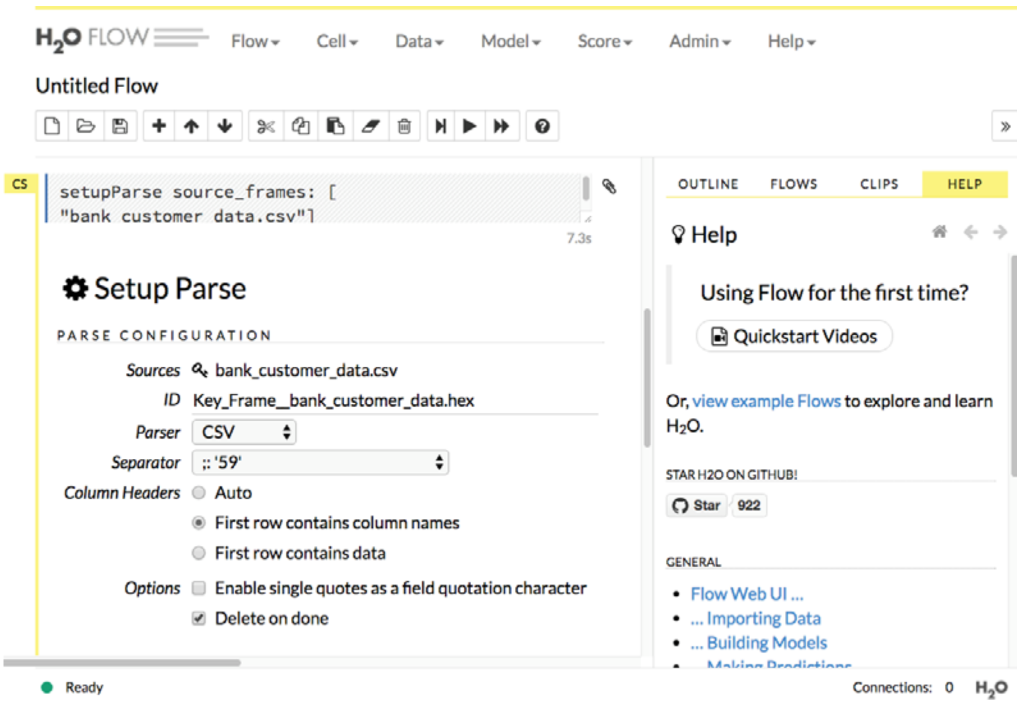
- Flow Web UI ...
- ... Importing Data
- ... Building Models

2) Upload our bank customer dataset to H2O Flow:

   a) From the **Data** menu, select **Upload File**

   b) Select **Choose File**, browse to your FLbigdataStats folder, choose the bank_customer_data.csv file and select **Upload**.

   This will automatically load the Setup Parse block.



```
setupParse source_frames: [ "bank_customer_data.csv"]
```

Setup Parse

PARSE CONFIGURATION

- Sources: bank_customer_data.csv
- ID: Key_Frame__bank_customer_data.hex
- Parser: CSV
- Separator: ; '59'
- Column Headers: Auto / [✓ First row contains column names] / First row contains data
- Options: Enable single quotes as a field quotation character; [✓ Delete on done]

3) Parse the banking dataset.

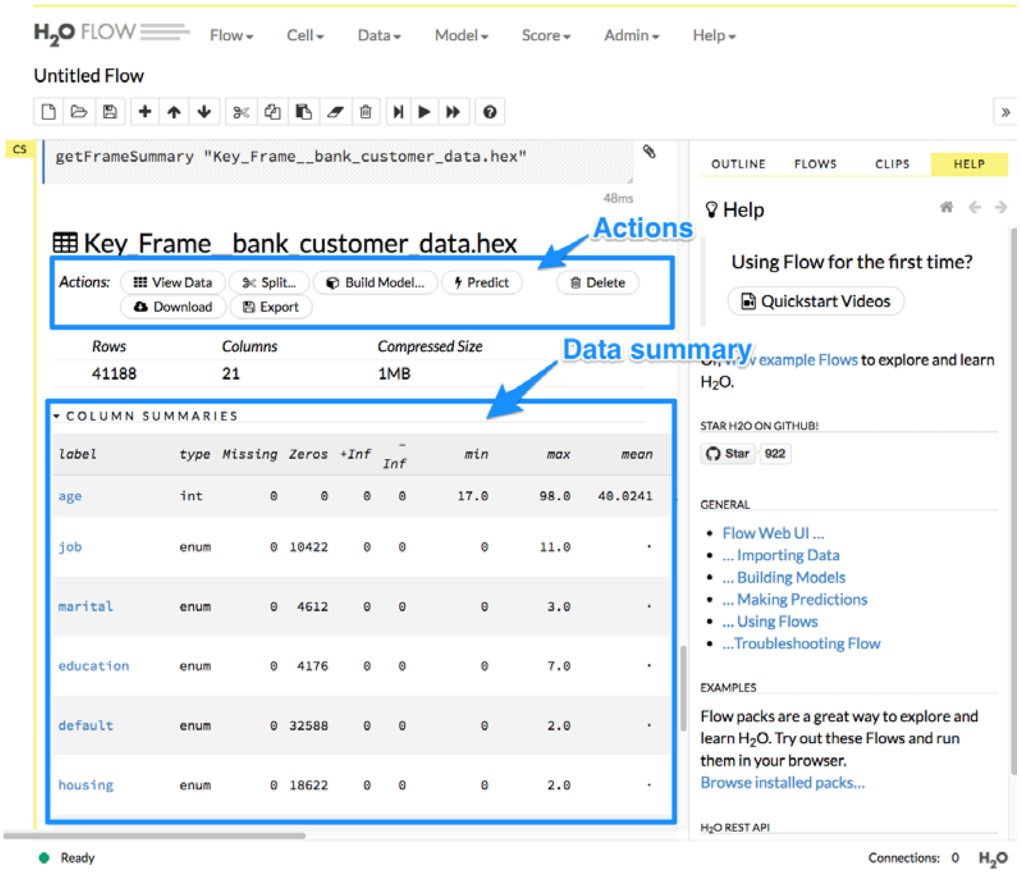
   H20 Flow will automatically guess an appropriate data type for each column such as numeric or enum (enum, for enumerable, is the data type for categorical data). These can be manually changed if needed but in this case we will leave the default collumn types and parse configuration.

   a) Select **Parse** (at the bottom of the Parse block) to parse this file with these settings.

      H2O Flow will display the Job block, showing the progress of the parse task.

   b) Once the job is complete, select **View**.

H2O Flow will display the Key Frame block showing a summary of the parsed data.



```
getFrameSummary "Key_Frame__bank_customer_data.hex"
```

Key_Frame__bank_customer_data.hex

Actions: View Data | Split... | Build Model... | Predict | Delete | Download | Export

Rows	Columns	Compressed Size
41188	21	1MB

COLUMN SUMMARIES

label	type	Missing	Zeros	+Inf	-Inf	min	max	mean
age	int	0	0	0	0	17.0	98.0	40.0241
job	enum	0	10422	0	0	0	11.0	.
marital	enum	0	4612	0	0	0	3.0	.
education	enum	0	4176	0	0	0	7.0	.
default	enum	0	32588	0	0	0	2.0	.
housing	enum	0	18622	0	0	0	2.0	.

...view example Flows to explore and learn H2O.

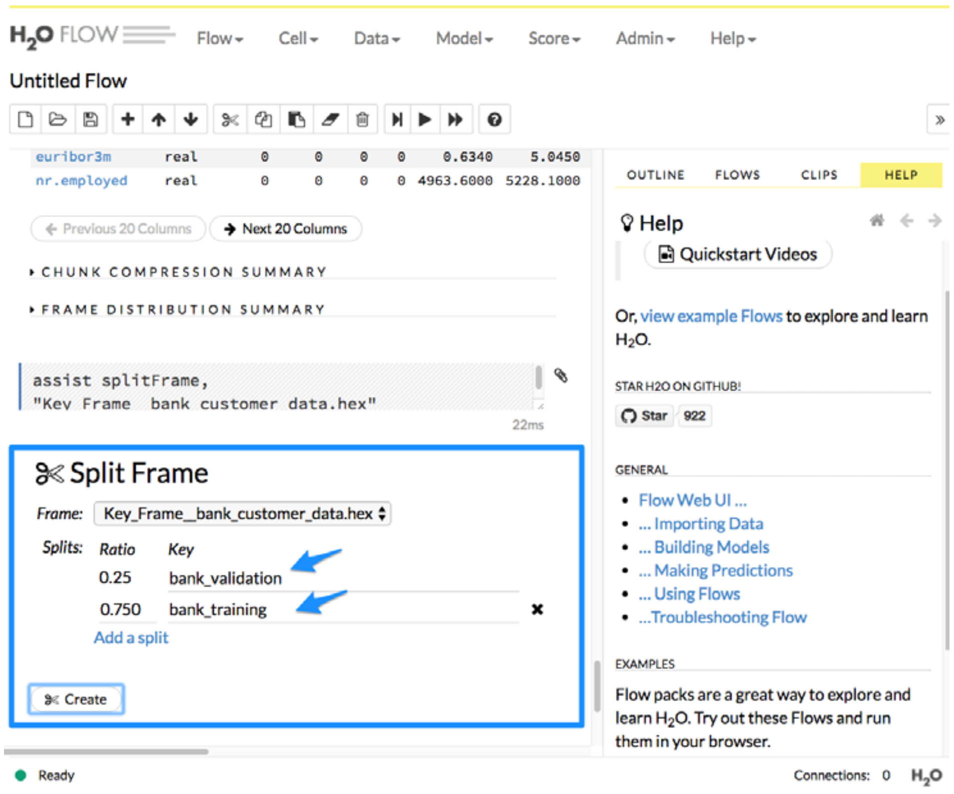
Flow packs are a great way to explore and learn H_2O. Try out these Flows and run them in your browser.

4) Split the data frame into Training and Validation sets:

a) In the Key Frame block, select **Split**… from the Actions available.

b) Label the key values to *bank_validation* for the 0.25 split and *bank_training* for the 0.75 split.

c) Select **Create**. The data is now ready to be modelled.



```
euribor3m    real    0    0    0    0    0.6340    5.0450
nr.employed  real    0    0    0    0 4963.6000 5228.1000
```

```
assist splitFrame,
"Key_Frame__bank_customer_data.hex"
```

22ms

Split Frame

Frame: Key_Frame__bank_customer_data.hex

Splits:	Ratio	Key
	0.25	bank_validation
	0.750	bank_training

Add a split

Or, view example Flows to explore and learn H_2O.

- Flow Web UI ...
- ... Importing Data
- ... Building Models
- ... Making Predictions
- ... Using Flows
- ...Troubleshooting Flow

Flow packs are a great way to explore and learn H_2O. Try out these Flows and run them in your browser.

5) Create a Neural Net model.

a) Select **Deep learning** from the **Model** menu.

b) Set the training frame to be 'bank_training' and the validation frame to be 'bank_validation'.

c) Set the response_column to 'y' (the final variable on the list).

d) Observe that the main parameters of importance are:

- activation: the type of activation function used in the neural network
- hidden: the number of hidden nodes arranges in there columns
- epoch: the number of full iterations of the training data used for training.

e) For these main variables choose:

- activation = rectifier
- hidden = 100, 100 (this means 2 hidden layers of 100 nodes each)
- epoch = 10

f) We wish to see how different variables impacted the overall results so tick the box next to variable_importances.

g) Observe the extensive 'Advanced' and 'Expert' parameters available. These are beyond the scope of this tutorial however they should give you a feeling for the complexity available in deep learning. In the H20 framework it's easy to experiment with these parameters and other datasets as an extension to this exercise.

h) Select **Build Model** at the end of the parameters list, the remainder of which we will leave to the default for now.

6) View the Neural Net model.

a) Select **View** once the job is finished.

b) The two ROC curve two graphs are for the training data and validation data effectively. The area under the curve AUC is the main indicator of performance.

c) The next graph shows the overall error for different epochs for both the training set and the validation set. We see the error has decreased from 1 to 10 epochs.

d) From the Variable Importance bar graph we get an idea of which variables were most important to the overall result. This can only be taken as an indicator rather than an exact measure due to the weights being initialised randomly, as well as inherent noise in the data. Most likely the *length* will appear near the top of this list. Why would this be such an important variable? Why should we exclude it from our model?

7) Modify the Neural Net model to ignore Call Length.

a) Build a new model from the Model menu or scroll up to the existing Build a Model block and adjusting the settings.

b) Use the same parameters, except this time choose to ignore the length column by selecting the appropriate box under ignored_columns.

c) Observe the degraded performance on both the training and validation set. This can be seen in both the changed shape of the curve as well as the AUC value.

8) Overfitting the Neural Net model.

a) Again, build a new model from the Model menu, or scroll up to the existing Build a Model block and adjusting the settings.

b) Use the same parameters, still choosing to ignore the call length, however now choose hidden = 100, epochs = 80. What do you expect to occur?

c) Observe in the scoring history graph that the validation score, in yellow, hits a minimum before beginning to climb again. This is a typical example of overfitting. By focusing the predictions too heavily on the training data the model does not generalise well to the validation data.

d) Notice the final reported score of MSE for training and validation is a discontinuous jump. This occurs because H2O automatically selects the model which minimises the error on the validation set. This can be avoided by unchecking the box overwrite_with_best_model (the first Expert parameter).

9) Create a Gradient Boosting Tree model.

We can compare the results of a neural Network trained on this data to a Gradient Boosted Tree model trained on this data.

a) Select **Gradient Boosting Machine** from the **Model** menu.

b) Set the training frame to be 'bank_training' and the validation frame to be 'bank_validation'.

c) Set the response variable to be 'y' and ignore the length column.

d) Note the following parameters of importance for a GBM:
- Ntrees: the number of trees created
- Max_depth: maximum depth of each tree
- min_rows: the minimum number of observations allowed in a leaf node
- Learn_rate: add only a fraction of the value given by each new tree to the accumulative model.

e) Leave these parameters to the default values and select **Build Model**.

f) Select **View** once the job is complete.FHow does this model compare to a neural network?

10) Overfitting with Gradient Boosting methods:

GBM models can be regularised in a number of ways. This includes limiting the allowed depth of each tree created, ensuring leaf nodes don't become too localised by requiring a minimum number of observations in each leaf, and adding only a small fraction of each new tree to the accumulative model by using a small learning rate.

a) Build a new model from the Model menu, or scroll up to the existing Build a Model block and adjusting the following settings:
- Max_depth = 10
- Min_rows = 4
- Learn_rate = 0.5

b) Observe the overfitting that occurs. AUC is now very near one for the training set, degraded for the validation set and the divergence should be illustrated in the scoring history.