



R, Python and Flow

[Feedback](#)

Environment **Water**

 [Google Scholar](#)

12 pts. of a variable

12. obj. of 1 variable

12. 403. 48 4 507 503 44

walkers.

clear here

[Home](#)
[About](#)
[Contact](#)

Comp. all normal class relationships

```

@my_gis      format class >= 20100101

```

normal class characterisation

©myIT Format - Class - 2008 model of Recd

File Edit Package Help Window



W. H. Green, G. B. Hill

1100



43-

100

1134

1000

1

100

100



2134

1000

100

© 2004 Blackwell Publishing Ltd

**W. Gregory Fry**

We train QLM



Written By: [Derek Leung](#)



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___c2_amazonaws_com_h2o_public_test_data_smalldata_flow_examples_seeds_dataset.txt

Parser CSV ▼

Separator HT ^E (horizontal tab: 09 ▼

Column Headers ☐ Auto☐ First row contains column names☒ First row contains dataOptions ☐ Enable single quotes as a field quotation character☒ Delete on done

EDIT COLUMN NAMES AND TYPES

Search by column name...

1	<input type="text"/>	Numeric ▼	15.26	14.88	14.26	13.84	16.14	14.38
2	<input type="text"/>	Numeric ▼	14.04	14.57	14.00	13.94	14.09	14.21
3	<input type="text"/>	Numeric ▼	0.071	0.0811	0.905	0.8955	0.9034	0.8951
4	<input type="text"/>	Numeric ▼	0.703	0.534	0.291	0.314	0.038	0.388
5	<input type="text"/>	Numeric ▼	3.312	3.333	3.337	3.379	3.502	3.312
6	<input type="text"/>	Numeric ▼	2.221	1.018	2.699	2.259	1.335	2.462
7	<input type="text"/>	Numeric ▼	5.22	4.936	4.815	4.885	5.175	4.936
8	<input type="text"/>	Numeric ▼	1	1	1	1	1	1

Previous page

Next page

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.HTML(h2o.display(table, header))
```

Model	AUC Train	AUC Test
GBM	0.9568221	0.9307979
DL	0.8956955	0.8841064

```
Out[7]:
```

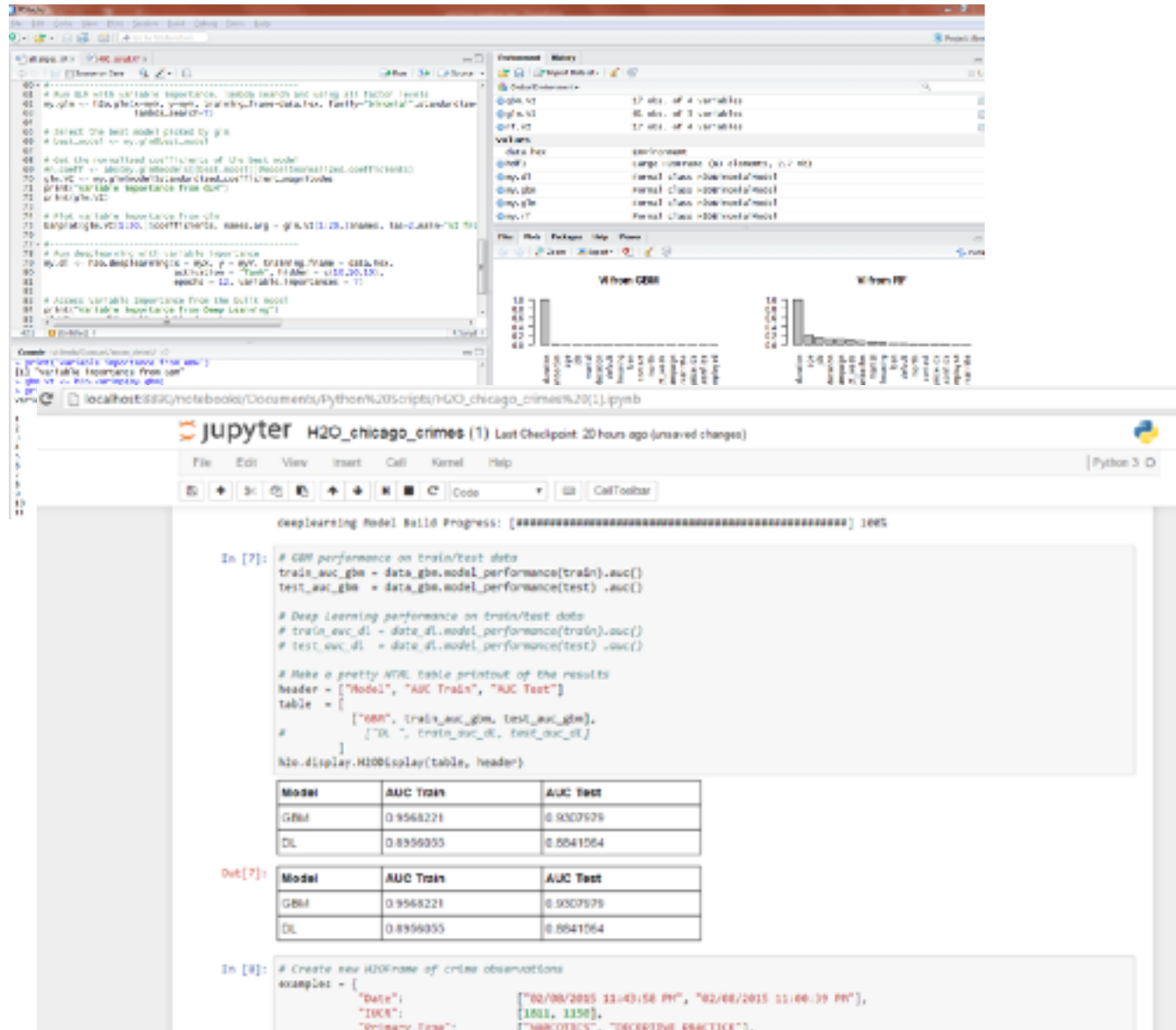
Model	AUC Train	AUC Test
GBM	0.9568221	0.9307979
DL	0.8956955	0.8841064

```
In [8]: # Create new H2OFrame of crime observations
examples = [
    {"Date": ["02/08/2015 11:43:58 PM", "02/08/2015 11:44:39 PM"],
     "IUCR": [1801, 1150],
     "Primary.Type": ["NARCOTICS", "DECEPTIVE PRACTICE"]}
```

R Interface Overview

Action	R	H2O
Reading data	<code>read_csv(data_path)</code>	<code>h2o.importFile(data_path)</code>
Summarizing data	<code>summary(data_frame)</code>	<code>h2o.summary(h2o_frame)</code>
Summary statistics	<code>mean(data_frame[["x"]])</code>	<code>h2o.mean(h2o_frame)</code>
Combining rows	<code>rbind(data_frame1, data_frame2)</code>	<code>h2o.rbind(h2o_frame1, h2o_frame2)</code>
Combining columns	<code>cbind(data_frame1, data_frame2)</code>	<code>h2o.cbind(h2o_frame1, h2o_frame2)</code>
Data selection	<code>data_frame[,]</code>	<code>h2o_frame[,]</code>
Transforming columns	<code>log(data_frame[, "x"])</code> <code>sqrt(data_frame[, "x"])</code>	<code>log(h2o_frame[, "x"])</code> <code>sqrt(h2o_frame[, "x"])</code>
Building Random Forest	<code>model = randomForest(y ~ x, data_frame)</code>	<code>model = h2o.randomForest(x, y, train_frame)</code>
Model Prediction	<code>predict(model, data_frame)</code>	<code>h2o.predict(model, h2o_frame)</code>
Model Metrics	<code>performance(model)</code> <code>auc(model)</code>	<code>metrics = model.model_performance(frame)</code> <code>h2o.auc(model)</code>

R, Python and Flow



The screenshot shows a Jupyter Notebook titled "H2O_chicago_crimes (1)". The code in the notebook is as follows:

```
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()

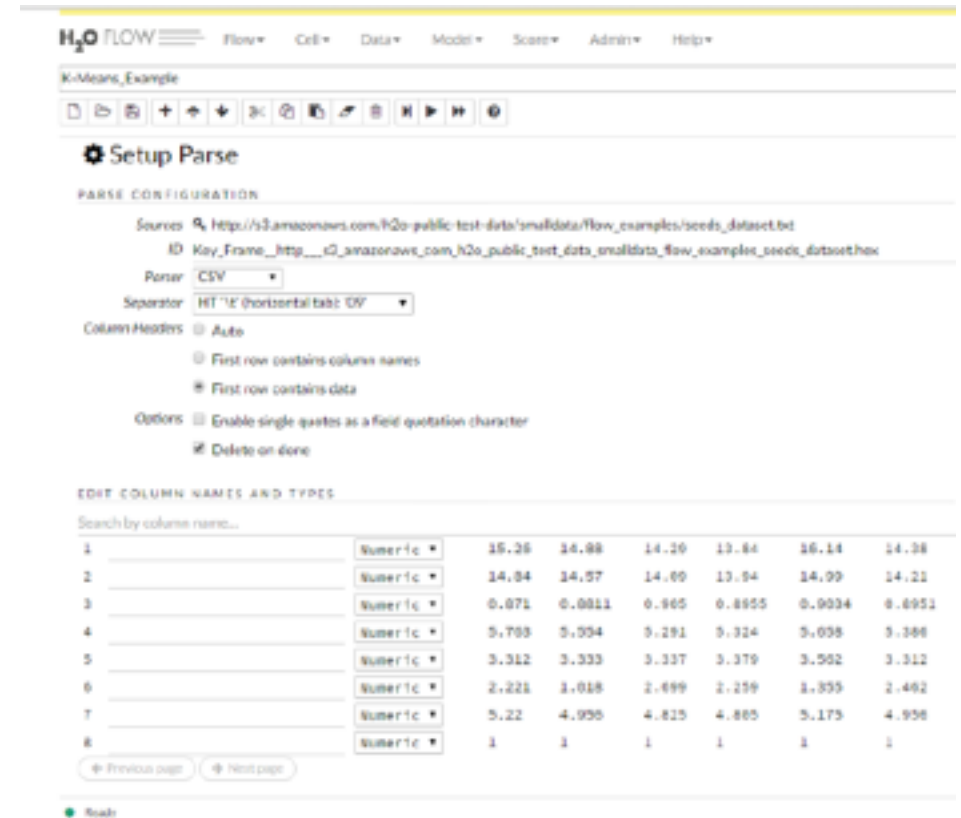
# Make a pretty HTML table printout of the results
header = ["Model", "AUC Train", "AUC Test"]
table = [
    ["GBM", train_auc_gbm, test_auc_gbm],
    ["DL", train_auc_dl, test_auc_dl]
]
h2o.display.HTMLdisplay(table, header)
```

The output of the code is a table showing the AUC Train and AUC Test for the GBM and DL models:

Model	AUC Train	AUC Test
GBM	0.9568221	0.9307979
DL	0.8960355	0.8641564

The notebook also includes a code cell for creating a new H2O name of crime observations:

```
In [8]: # Create new H2O name of crime observations
sample1 = [
    "Date": ["02/08/2015 11:43:58 PM", "02/08/2015 11:46:39 PM"],
    "IDCR": [1801, 1150],
    "Primary.Type": ["NARCOTICS", "DECEPTIVE PRACTICE"],
]
```



The screenshot shows the H2O Flow interface for a "K-Means Example". The "Setup Parse" configuration is as follows:

- Source: 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.txt
- Parser: CSV
- Separator: HT '\t' (horizontal tab) 'CSV'
- Column Headers: ☐ Auto
- ☐ First row contains column names
- ☐ First row contains data
- Options: ☐ Enable single quotes as a field quotation character
- ☒ Delete on done

The "EDIT COLUMN NAMES AND TYPES" section shows a table with 8 columns and 8 rows. The first column is labeled "Search by column name...". The other columns are labeled "Numeric", "15.38", "14.88", "14.20", "13.84", "16.14", and "14.38".

Search by column name...	Numeric	15.38	14.88	14.20	13.84	16.14	14.38
1	Numeric	14.84	14.57	14.00	13.94	14.99	14.21
2	Numeric	0.071	0.0811	0.965	0.8955	0.9034	0.6951
3	Numeric	5.703	5.594	5.291	5.324	5.038	5.386
4	Numeric	3.312	3.333	3.337	3.379	3.562	3.312
5	Numeric	2.221	1.018	2.699	2.259	1.855	2.462
6	Numeric	5.22	4.958	4.825	4.885	5.175	4.958
7	Numeric	1	1	1	1	1	1
8	Numeric	1	1	1	1	1	1