Kaggle Competition

```
h2o.init(nthreads = -1, max_mem_size = 4)
```

```
Checking whether there is an H2O instance running at http://localhost:54321..... not found.
Attempting to start a local H2O server...
  Java Version: openjdk version "11.0.4" 2019-07-16; OpenJDK Runtime Environment (build 11.0.4+11-post-
Ubuntu-1ubuntu218.04.3); OpenJDK 64-Bit Server VM (build 11.0.4+11-post-Ubuntu-1ubuntu218.04.3, mixed mode,
sharing)
  Starting server from /home/dfernandez/.local/lib/python3.6/site-packages/h2o/backend/bin/h2o.jar
  Ice root: /tmp/tmp3ikam17r
  JVM stdout: /tmp/tmp3ikam17r/h2o_dfernandez_started_from_python.out
  JVM stderr: /tmp/tmp3ikam17r/h2o_dfernandez_started_from_python.err
  Server is running at http://127.0.0.1:54321
Connecting to H2O server at http://127.0.0.1:54321 ... successful.
          H2O cluster uptime:                       03 secs
         H2O cluster timezone:                  Europe/Madrid
     H2O data parsing timezone:                            UTC
          H2O cluster version:                     3.26.0.11
      H2O cluster version age:                        4 days
             H2O cluster name:  H2O_from_python_dfernandez_cuy78o
      H2O cluster total nodes:                              1
      H2O cluster free memory:                          4 Gb
      H2O cluster total cores:                              4
```

```
import pandas as pd
df_pandas=pd.read_csv('Input/data.csv')
df=h2o.H2OFrame(df_pandas)
```

```
Parse progress: |[progress bar]| 100%
```

```
df.types
```

```
{'carat': 'real',
 'cut': 'enum',
 'color': 'enum',
 'clarity': 'enum',
 'depth': 'real',
 'table': 'real',
 'x': 'real',
 'y': 'real',
 'z': 'real',
 'price': 'int'}
```

```
df = df.asnumeric()
```

```
df.types
```

```
{'carat': 'real',
 'cut': 'real',
 'color': 'real',
 'clarity': 'real',
 'depth': 'real',
 'table': 'real',
 'x': 'real',
 'y': 'real',
 'z': 'real',
 'price': 'real'}
```

```
y_columns = "price"
x_columns = ["carat","cut","color", "clarity","depth","table","x","y","z"]
```

```
train, test = df.split_frame(ratios = [.8])
X_train=train[x_columns]
y_train=train[y_columns]
X_test=test[x_columns]
y_test=test[y_columns]
```

```
from h2o.automl import H2OAutoML
aml_ti = H2OAutoML(max_runtime_secs= 180,max_models= 15, seed= 1, nfolds=0)
aml_ti.train(x = x_columns, y = y_columns, training_frame = train, validation_frame=test)
```

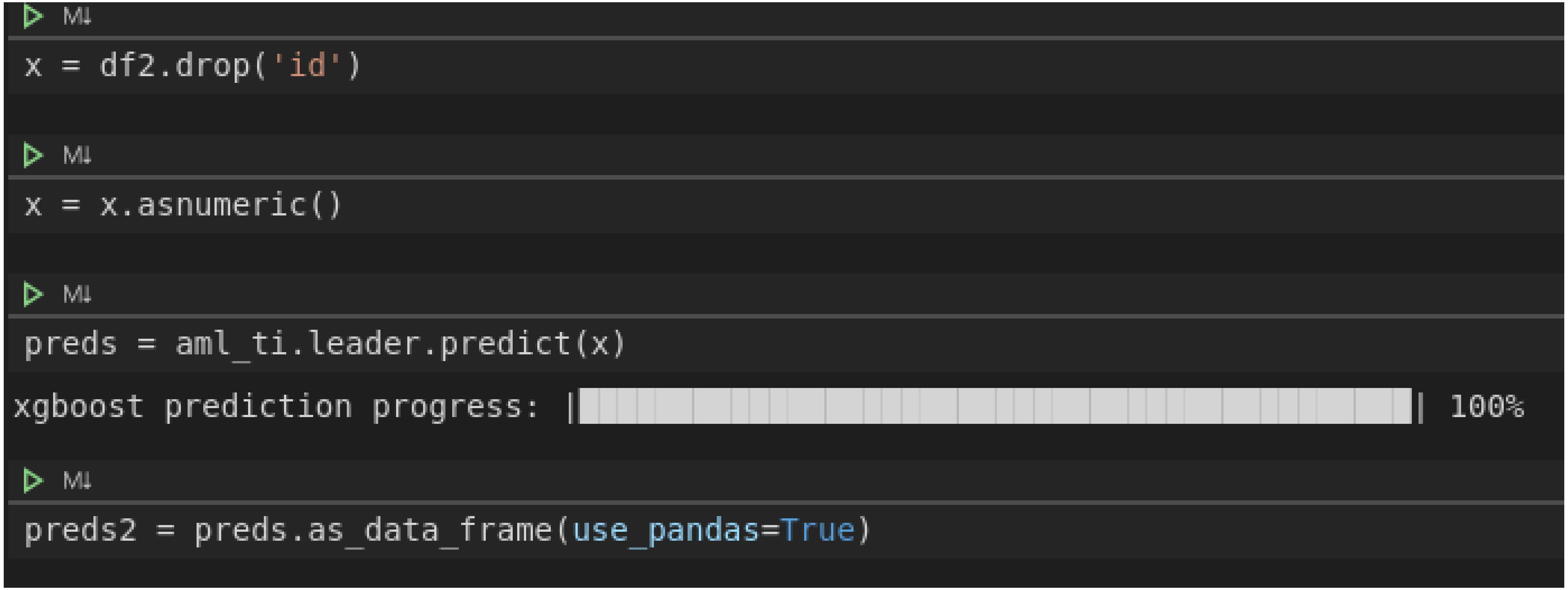
```
AutoML progress: |
| 100%
```

```
lb_ti = aml_ti.leaderboard
lb_ti
```

model_id	mean_residual_deviance	rmse	mse	mae	rmsle
XGBoost_1_AutoML_20191210_054101	283993	532.91	283993	263.043	0.0934656
GBM_3_AutoML_20191210_054101	288162	536.807	288162	271.852	0.0991364
GBM_4_AutoML_20191210_054101	290641	539.111	290641	270.271	0.102248
GBM_1_AutoML_20191210_054101	308100	555.068	308100	278.136	0.104674
GBM_2_AutoML_20191210_054101	308354	555.297	308354	277.162	0.103929
XGBoost_2_AutoML_20191210_054101	310856	557.545	310856	266.065	0.0915488
GBM_5_AutoML_20191210_054101	323207	568.513	323207	273.814	0.103403
XGBoost_3_AutoML_20191210_054101	323597	568.856	323597	284.363	0.111431
XRT_1_AutoML_20191210_054101	338206	581.555	338206	286.121	0.0986776
DRF_1_AutoML_20191210_054101	346568	588.701	346568	289.14	0.0998354

```
df2 = pd.read_csv('Input/test.csv')
df2 = h2o.H2OFrame(df2)
```

```
Parse progress: |[redacted]| 100%
```



```python
x = df2.drop('id')
```

```python
x = x.asnumeric()
```

```python
preds = aml_ti.leader.predict(x)
```

```
xgboost prediction progress: |█████████████████████████████████████████████| 100%
```

```python
preds2 = preds.as_data_frame(use_pandas=True)
```

```
test = df2.as_data_frame(use_pandas=True)
```

```
results = pd.DataFrame()
results['id'] = test['id']
results['price'] = preds2['predict']
```

```
# Create & upload a file.
results.to_csv('./Output/results2.csv',index=False)
```