Training and scoring within a SQL Big Data Cluster

In this notebook you will train a model, use it to score data that has been uploaded to HDFS, and save the scored result to an external table.

Begin by running the following cell. You can run any code cell by placing your cursor within its region and then selecting the play icon (a triangle within a circle) that appears on the left.

In [3]:
```
# Import the standard modules we need
import numpy as np
import pandas as pd
```

Next, you will load the training data from HDFS. Run the following cell.

In [16]:
```
# access the training data from HDFS by reading into a Spark DataFrame
df = (spark.read.option("inferSchema", "true").option("header", "true").csv('/
data/training-formatted.csv'))

# convert the Spark DataFrame to a Pandas DataFrame so we can use Scikit-Learn
data = df.toPandas()
```

Now, you will pick out the features and labels from the training data. Run the following cell.

In [20]:
```
# Select the features used for predicting battery life
X = data.iloc[:,1:74]
X = X.iloc[:,np.r_[2:7, 9:73]]
X = X.interpolate()

# Select the labels only (the measured battery life)
Y = data.iloc[:,0].values.flatten()
```

Run the following cell to view the features that will be used to train the model.

In [19]:
```
# Examine the features selected
X.info()
```

In the following cell, you train a model using a GradientBoostingRegressor, providing it the features (X) and the label values (Y). Run the following cell.

In [21]:
```
# Train a regression model
from sklearn.ensemble import GradientBoostingRegressor
model = GradientBoostingRegressor()
model.fit(X,Y)
```

Now try making a single prediction with the trained model. Run the following cell.

In [22]:
```
# Try making a single prediction and observe the result
model.predict(X.iloc[0:1])
```

With a trained model in hand, you are now ready to score battery life predictions against a new set of vehicle telemetry data. The output of the cell will be predicted battery life for each vehicle. Run the following cell.

In [27]:
```
# access the test data from HDFS by reading into a Spark DataFrame
df_test = (spark.read.option("inferSchema", "true").option("header", "true").c
sv('/data/fleet-formatted.csv'))
test_data = df_test.toPandas()

# prepare the test data (dropping unused columns)
test_data = test_data.drop(columns=["Car_ID", "Battery_Age"])
test_data = test_data.iloc[:,np.r_[2:7, 9:73]]
test_data.rename(columns={'Twelve_hourly_temperature_forecast_for_next_31_days
_reversed': 'Twelve_hourly_temperature_history_for_last_31_days_before_death_l
ast_recording_first'}, inplace=True)

# make the battery life predictions for each of the vehicles in the test data
battery_life_predictions = model.predict(test_data)

# examine the prediction
battery_life_predictions
```

Now you can package up the predictions along with the vehicle telemetry into a single DataFrame so that you can export it back out to HDFS as a CSV. In the last line below, replace `YOUR_UNIQUE_IDENTIFIER` with your assigned identifier and then run the following cell.

In [35]:
```
# prepare one data frame that includes predictions for each vehicle
scored_data = test_data
scored_data["Estimated_Battery_Life"] = battery_life_predictions

df_scored = spark.createDataFrame(scored_data)

# Replace YOUR_UNIQUE_IDENTIFIER with your ID value in the below:
df_scored.coalesce(1).write.option("header", "true").csv("/data/battery-life-Y
OUR_UNIQUE_IDENTIFIER.csv")
```

The above command creates a folder called `battery-life-YOUR_UNIQUE_IDENTIFIER.csv`, which contains one CSV file that you can create an external table from, which will enable you to query the predictions for each vehicle from SQL. Return to the lab instructions to learn how to create an external table you can use for querying this data using SQL.