**M8 Lab: Building Regression Models in Spark**

Steps:

1. What is the RMSE of the regression model that the code built? Is it high or low? Research online for the meaning of RMSE, compare it to the SSE that was discussed in the lesson slides.

As the value of RMSE is lower , better the fit of the model.

Root mean square error is 2.514112.

As the per the observation which can be observed in residuals, we can say that

RMSE is low.

SSE is also known as residual sum of squares. This is a measure of how far from each point the best fitting regression line (plane, set) was squared. If you take the average of that you get MSE. Now MSE is mean squared error. What if you do not want the square of measures but are concerned more with how far on average your predicted values were to your actual values. When you take the square root of MSE and that gives you the root mean squared error.

from pyspark.ml.regression import LinearRegression

# Load training data

training = spark.read.format("libsvm")\

.load("dbfs:/FileStore/tables/abalone-1.txt")

lr = LinearRegression(maxIter=50, regParam=0.3, elasticNetParam=0.8)

# Fit the model

lrModel = lr.fit(training)

# Print the coefficients and intercept for linear regression

print("Coefficients: %s" % str(lrModel.coefficients))

print("Intercept: %s" % str(lrModel.intercept))

# Summarize the model over the training set and print out some metrics

trainingSummary = lrModel.summary

print("numIterations: %d" % trainingSummary.totalIterations)

print("objectiveHistory: %s" % str(trainingSummary.objectiveHistory))

trainingSummary.residuals.show()

print("RMSE: %f" % trainingSummary.rootMeanSquaredError)

print("r2: %f" % trainingSummary.r2)

Graphical user interface, text, application, email

Description automatically generated

Graphical user interface, text, application

Description automatically generated

1. Change the number of iterations from 10 to 20. What is the resulting RMSE? Change it again to 50, what is the resulting RMSE?

Graphical user interface, text, application, email

Description automatically generated

Graphical user interface, application, Word

Description automatically generated

Number of iterations changed from 10 to 20

RMSE value is 2.505995.

Graphical user interface, text, application, email

Description automatically generated

Graphical user interface, text, application

Description automatically generated

Number of Iteration changed from 20 to 50.

RMSE value is 2.504.

1. On the top menu, click on your cluster name -> View Spark UI option. This will show you details about how Spark ran your code.
2. Check all the top tabs in the Spark UI. These are Jobs, Stages, Storage, etc. How many jobs did Spark submit to run your code? How many blocks was the RDD holding the dataset divided into? Take screenshots for this information once you find it in the various tabs.

Spark ran 24 Jobs Complete the execution

the RDD holding the dataset divided into 0 blocks.

Graphical user interface, application, Word

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

Graphical user interface, application, table

Description automatically generated

Graphical user interface, application

Description automatically generated

A computer screen capture

Description automatically generated with medium confidence

A computer screen capture

Description automatically generated with medium confidence

Graphical user interface, application, Word

Description automatically generated

### **Part II: Regression Model using Decision Trees in Spark’s ML Library**

from pyspark.ml import Pipeline

from pyspark.ml.regression import DecisionTreeRegressor

from pyspark.ml.feature import VectorIndexer

from pyspark.ml.evaluation import RegressionEvaluator

# Load the data stored in LIBSVM format as a DataFrame.

data = spark.read.format("libsvm").load("dbfs:/FileStore/tables/abalone-1.txt")

# Automatically identify categorical features, and index them.

# We specify maxCategories so features with > 4 distinct values are treated as continuous.

featureIndexer =\

VectorIndexer(inputCol="features", outputCol="indexedFeatures", maxCategories=4).fit(data)

# Split the data into training and test sets (30% held out for testing)

(trainingData, testData) = data.randomSplit([0.7, 0.3])

# Train a DecisionTree model.

dt = DecisionTreeRegressor(featuresCol="indexedFeatures")

# Chain indexer and tree in a Pipeline

pipeline = Pipeline(stages=[featureIndexer, dt])

# Train model. This also runs the indexer.

model = pipeline.fit(trainingData)

# Make predictions.

predictions = model.transform(testData)

# Select example rows to display.

#predictions.select("prediction", "label", "features").show(5)

rmse =0

# Select (prediction, true label) and compute test error

for i in range(10):

evaluator = RegressionEvaluator(

labelCol="label", predictionCol="prediction", metricName="rmse")

rmse = rmse + evaluator.evaluate(predictions)

rmse = rmse/10

print("Root Mean Squared Error (RMSE) on test data = %g" % rmse)

#treeModel = model.stages[1]

# summary only

#print(treeModel)

1. What is the obtained RMSE?
2. Run your code again multiple times. What are the obtained RMSE? Do the RMSE number change? Why?

RMSE make a great loss metric for a model to optimize because it can be computed quickly.

Graphical user interface, text, application

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

1. On way to find the actual generalization error (RMSE) is by running the code multiple times then computing the average over all obtained RMSE's.  Modify your code so that it includes a for loop that runs the regression model learning 10 times, and prints out the average over the 10 obtained RMSE's. What is the average RMSE?

The average RMSE varies between 2.29 sometimes to 2.33 .

1. Now when you run your modified code multiple times, you will notice that the RMSE does not change much among the different runs. That's because running multiple times and average over all runs is an approximation for the k-fold cross validation concepts that allows for better estimation of a model's generalization error.

Graphical user interface, text, application

Description automatically generated

Graphical user interface, text, application

Description automatically generated

Graphical user interface, text, application

Description automatically generated

Graphical user interface, text, application

Description automatically generated