Tecnológico nacional de México
Instituto tecnológico de Tijuana

Subdirección Académica
Departamento de sistemas y computación

SEMESTRE:
Febrero - agosto 2022

CARRERA:
Ingeniería en Sistemas Computacionales

MATERIA:
Datos masivos

NOMBRE TRABAJO:
Practice 3 Random Forest Classifier

UNIDAD:
2

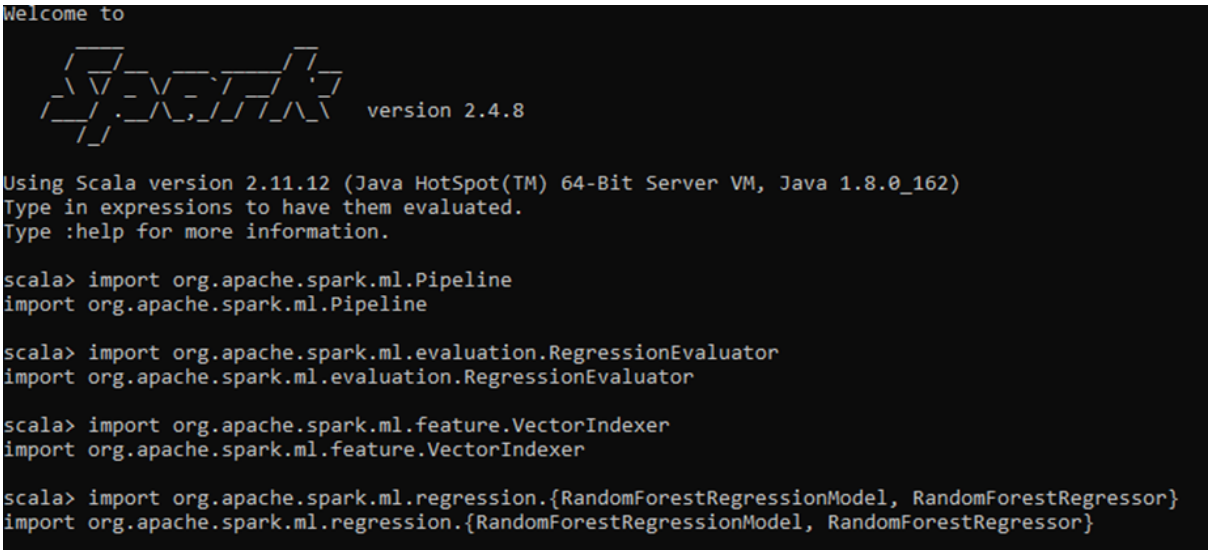
NOMBRE Y NÚMERO DE CONTROL
Perez Mora Ana Ivonne #18212074
Madrigal Ramos Ulises Omar #18210496

NOMBRE DEL MAESTRO:
Jose Christian Romero Hernandez

FECHA:
Tijuana Baja California 5 de mayo del 2022

// Import libraries

```
import org.apache.spark.ml.Pipeline
import org.apache.spark.ml.classification.{RandomForestClassificationModel, RandomForestClassifier}
import org.apache.spark.ml.evaluation.MulticlassClassificationEvaluator
import org.apache.spark.ml.feature.{IndexToString, StringIndexer, VectorIndexer}
```



// Load and parse the data file, converting it to a DataFrame.

```
val data = spark.read.format("libsvm").load("C:\Spark\spark-2.4.8-bin-hadoop2.7\data\mllib\sample_libsvm_data.txt)
```

// Index labels, adding metadata to the label column.
// Fit on whole dataset to include all labels in index.

```
val labelIndexer = new StringIndexer()
  .setInputCol("label")
  .setOutputCol("indexedLabel")
  .fit(data)
```

// Automatically identify categorical features, and index them.
// Set maxCategories so features with > 4 distinct values are treated as continuous.

```
val featureIndexer = new VectorIndexer()
  .setInputCol("features")
  .setOutputCol("indexedFeatures")
  .setMaxCategories(4)
```

.fit(data)

```
scala> val featureIndexer = new VectorIndexer() .setInputCol("features") .setOutputCol("indexedFeatures") .setMaxCategories(4) .fit(data)
featureIndexer: org.apache.spark.ml.feature.VectorIndexerModel = vecIdx_e5c576e34dc8
```

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

val Array(trainingData, testData) = data.randomSplit(Array(0.7, 0.3))

```
scala> val Array(trainingData, testData) = data.randomSplit(Array(0.7, 0.3))
trainingData: org.apache.spark.sql.Dataset[org.apache.spark.sql.Row] = [label: double, features: vector]
testData: org.apache.spark.sql.Dataset[org.apache.spark.sql.Row] = [label: double, features: vector]
```

// Train a RandomForest model.

val rf = new RandomForestClassifier()
  .setLabelCol("indexedLabel")
  .setFeaturesCol("indexedFeatures")
  .setNumTrees(10)

```
scala> val rf = new RandomForestRegressor() .setLabelCol("label") .setFeaturesCol("indexedFeatures")
rf: org.apache.spark.ml.regression.RandomForestRegressor = rfr_7dd95ee99dc0
```

// Convert indexed labels back to original labels.

val labelConverter = new IndexToString()
  .setInputCol("prediction")
  .setOutputCol("predictedLabel")
  .setLabels(labelIndexer.labelsArray(0))

// Chain indexers and forest in a Pipeline.

val pipeline = new Pipeline()
  .setStages(Array(labelIndexer, featureIndexer, rf, labelConverter))

```
scala> val pipeline = new Pipeline() .setStages(Array(featureIndexer, rf))
pipeline: org.apache.spark.ml.Pipeline = pipeline_6bad1b60306c
```

// Train model. This also runs the indexers.

val model = pipeline.fit(trainingData)

```
scala> val model = pipeline.fit(trainingData)
model: org.apache.spark.ml.PipelineModel = pipeline_6bad1b60306c
```

// Make predictions.

val predictions = model.transform(testData)

```
scala> val predictions = model.transform(testData)
predictions: org.apache.spark.sql.DataFrame = [label: double, features: vector ... 2 more fields]
```

// Select example rows to display.

predictions.select("predictedLabel", "label", "features").show(5)

```
scala> predictions.select("prediction", "label", "features").show(5)
+----------+-----+--------------------+
|prediction|label|            features|
+----------+-----+--------------------+
|       0.0|  0.0|(692,[95,96,97,12...|
|       0.0|  0.0|(692,[122,123,124...|
|       0.0|  0.0|(692,[122,123,148...|
|       0.0|  0.0|(692,[124,125,126...|
|       0.0|  0.0|(692,[126,127,128...|
+----------+-----+--------------------+
only showing top 5 rows
```

// Select (prediction, true label) and compute test error.

```
val evaluator = new MulticlassClassificationEvaluator()
  .setLabelCol("indexedLabel")
  .setPredictionCol("prediction")
  .setMetricName("accuracy")
val accuracy = evaluator.evaluate(predictions)
println(s"Test Error = ${(1.0 - accuracy)}")

val rfModel =
model.stages(2).asInstanceOf[RandomForestClassificationModel]
println(s"Learned classification forest model:\n ${rfModel.toDebugString}")
```

```
scala> val evaluator = new RegressionEvaluator() .setLabelCol("label") .setPredictionCol("prediction") .setMetricName("rmse")
evaluator: org.apache.spark.ml.evaluation.RegressionEvaluator = regEval_98db33ea9ecd
```

```
scala> println(s"Root Mean Squared Error (RMSE) on test data = $rmse")
Root Mean Squared Error (RMSE) on test data = 0.16422453217986943

scala> val rfModel = model.stages(1).asInstanceOf[RandomForestRegressionModel]
rfModel: org.apache.spark.ml.regression.RandomForestRegressionModel = RandomForestRegressionModel (uid=rfr_7dd95ee99dc0) with 20 trees
```

```
scala> val rfModel = model.stages(1).asInstanceOf[RandomForestRegressionModel]
rfModel: org.apache.spark.ml.regression.RandomForestRegressionModel = RandomForestRegressionModel (uid=rfr_7dd95ee99dc0) with 20 trees

scala> println(s"Learned regression forest model:\n ${rfModel.toDebugString}")
Learned regression forest model:
 RandomForestRegressionModel (uid=rfr_7dd95ee99dc0) with 20 trees
  Tree 0 (weight 1.0):
    If (feature 433 <= 52.5)
     Predict: 0.0
    Else (feature 433 > 52.5)
     Predict: 1.0
  Tree 1 (weight 1.0):
    If (feature 490 <= 44.5)
     Predict: 0.0
    Else (feature 490 > 44.5)
     Predict: 1.0
  Tree 2 (weight 1.0):
    If (feature 462 <= 62.5)
     Predict: 0.0
    Else (feature 462 > 62.5)
     Predict: 1.0
  Tree 3 (weight 1.0):
    If (feature 461 <= 46.5)
     Predict: 0.0
    Else (feature 461 > 46.5)
     Predict: 1.0
  Tree 4 (weight 1.0):
    If (feature 405 <= 21.0)
     Predict: 0.0
    Else (feature 405 > 21.0)
     Predict: 1.0
  Tree 5 (weight 1.0):
    If (feature 323 <= 57.5)
     Predict: 0.0
    Else (feature 323 > 57.5)
     If (feature 455 <= 24.5)
      Predict: 1.0
     Else (feature 455 > 24.5)
      Predict: 0.0
  Tree 6 (weight 1.0):
    If (feature 490 <= 44.5)
     Predict: 0.0
    Else (feature 490 > 44.5)
```