TECNOLÓGICO NACIONAL DE MÉXICO

INSTITUTO TECNOLÓGICO DE TIJUANA
SUBDIRECCIÓN ACADÉMICA
DEPARTAMENTO DE SISTEMAS Y COMPUTACIÓN

SEMESTRE:

Enero - Junio 2022

CARRERA:

Ing. en Sistemas Computacionales

MATERIA:

Datos Masivos

TÍTULO ACTIVIDAD:

Practica 3

NOMBRE Y NÚMERO DE CONTROL DEL ALUMNO:

Hernandez Pablo Anahi Del Carmen – 18210486
Real Castro Manuel Angel – 18212253

Here were the libraries that we would use

```
Welcome to
      ____              __
     / __/__  ___ _____/ /__
    _\ \/ _ \/ _ `/ __/  '_/
   /___/ .__/\_,_/_/ /_/\_\   version 2.4.8
      /_/

Using Scala version 2.11.12 (Java HotSpot(TM) 64-Bit Server VM, Java 1.8.0_162)
Type in expressions to have them evaluated.
Type :help for more information.

scala> import org.apache.spark.ml.Pipeline
import org.apache.spark.ml.Pipeline

scala> import org.apache.spark.ml.evaluation.RegressionEvaluator
import org.apache.spark.ml.evaluation.RegressionEvaluator

scala> import org.apache.spark.ml.feature.VectorIndexer
import org.apache.spark.ml.feature.VectorIndexer

scala> import org.apache.spark.ml.regression.{RandomForestRegressionModel, RandomForestRegressor}
import org.apache.spark.ml.regression.{RandomForestRegressionModel, RandomForestRegressor}
```

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

```
scala> val data = spark.read.format("libsvm").load("C:/Spark/spark-2.4.8-bin-hadoop2.7/data/mllib/sample_libsvm_data.txt")
22/05/04 21:52:44 WARN LibSVMFileFormat: 'numFeatures' option not specified, determining the number of features by going though the input. If you know the number in adv
ance, please specify it via 'numFeatures' option to avoid the extra scan.
data: org.apache.spark.sql.DataFrame = [label: double, features: vector]
```

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

```
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).

```
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.

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

Chain indexer and forest in a Pipeline.

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

Train model. This also runs the indexer.

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

Make predictions.

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

Select example rows to display.

```
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.

```
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)
```

EDUCACIÓN
SECRETARÍA DE EDUCACIÓN PÚBLICA

TECNOLÓGICO NACIONAL DE MÉXICO®