TECNOLÓGICO NACIONAL DE MÉXICO

INSTITUTO TECNOLÓGICO DE TIJUANA

SUBDIRECCIÓN ACADÉMICA

DEPARTAMENTO DE SISTEMAS Y COMPUTACIÓN

NOMBRE DE LOS ALUMNOS:

RAYMUNDO HIRALES LAZARENO (N. CONTROL: 17212339)

PAULA ANDREA RAMOS VERDIN (N. CONTROL: 18210721)

Carrera: Ingeniería Informática

MATERIA: Datos Masivos

PROFESOR: JOSE CHRISTIAN ROMERO HERNANDEZ

Practica 3

FECHA: 06/05/22

//Ejemplo Clasificacion

import org.apache.spark.mllib.tree.RandomForest import org.apache.spark.mllib.tree.model.RandomForestModel import org.apache.spark.mllib.util.MLUtils

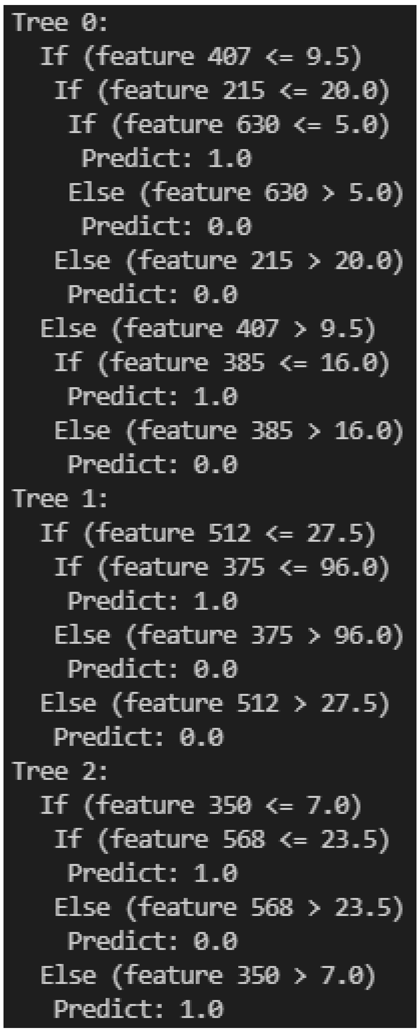
// Load and parse the data file. val data = MLUtils.loadLibSVMFile(sc, "sample_libsvm_data.txt") // Split the data into training and test sets (30% held out for testing) val splits = data.randomSplit(Array(0.7, 0.3)) val (trainingData, testData) = (splits(0), splits(1))

// Train a RandomForest model. // Empty categoricalFeaturesInfo indicates all features are continuous. val numClasses = 2 val categoricalFeaturesInfo = MapInt, Int val numTrees = 3 // Use more in practice. val featureSubsetStrategy = "auto" // Let the algorithm choose. val impurity = "gini" val maxDepth = 4 val maxBins = 32

val model = RandomForest.trainClassifier(trainingData, numClasses, categoricalFeaturesInfo, numTrees, featureSubsetStrategy, impurity, maxDepth, maxBins)

// Evaluate model on test instances and compute test error val labelAndPreds = testData.map { point => val prediction = model.predict(point.features) (point.label, prediction) } val testErr = labelAndPreds.filter(r => r._1 != r._2).count.toDouble / testData.count() println(s"Test Error = $testErr") println(s"Learned classification forest model:\n ${model.toDebugString}")

// Save and load model model.save(sc, "target/tmp/myRandomForestClassificationModel") val sameModel = RandomForestModel.load(sc, "target/tmp/myRandomForestClassificationModel")


```
Tree 0:
  If (feature 407 <= 9.5)
   If (feature 215 <= 20.0)
    If (feature 630 <= 5.0)
     Predict: 1.0
    Else (feature 630 > 5.0)
     Predict: 0.0
   Else (feature 215 > 20.0)
    Predict: 0.0
  Else (feature 407 > 9.5)
   If (feature 385 <= 16.0)
    Predict: 1.0
   Else (feature 385 > 16.0)
    Predict: 0.0
Tree 1:
  If (feature 512 <= 27.5)
   If (feature 375 <= 96.0)
    Predict: 1.0
   Else (feature 375 > 96.0)
    Predict: 0.0
  Else (feature 512 > 27.5)
   Predict: 0.0
Tree 2:
  If (feature 350 <= 7.0)
   If (feature 568 <= 23.5)
    Predict: 1.0
   Else (feature 568 > 23.5)
    Predict: 0.0
  Else (feature 350 > 7.0)
   Predict: 1.0
```

//Ejemplo Regresion

import org.apache.spark.mllib.tree.RandomForest import org.apache.spark.mllib.tree.model.RandomForestModel import org.apache.spark.mllib.util.MLUtils

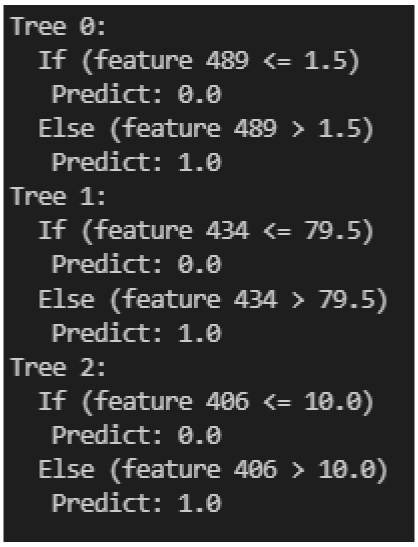
// Load and parse the data file. val data = MLUtils.loadLibSVMFile(sc, "sample_libsvm_data.txt") // Split the data into training and test sets (30% held out for testing) val splits = data.randomSplit(Array(0.7, 0.3)) val (trainingData, testData) = (splits(0), splits(1))

// Train a RandomForest model. // Empty categoricalFeaturesInfo indicates all features are continuous. val numClasses = 2 val categoricalFeaturesInfo = MapInt, Int val numTrees = 3 // Use more in practice. val featureSubsetStrategy = "auto" // Let the algorithm choose. val impurity = "variance" val maxDepth = 4 val maxBins = 32

val model = RandomForest.trainRegressor(trainingData, categoricalFeaturesInfo, numTrees, featureSubsetStrategy, impurity, maxDepth, maxBins)

// Evaluate model on test instances and compute test error val labelsAndPredictions = testData.map { point => val prediction = model.predict(point.features) (point.label, prediction) } val testMSE = labelsAndPredictions.map{ case(v, p) => math.pow((v - p), 2)}.mean() println(s"Test Mean Squared Error = $testMSE") println(s"Learned regression forest model:\n ${model.toDebugString}")

// Save and load model model.save(sc, "target/tmp/myRandomForestRegressionModel") val sameModel = RandomForestModel.load(sc, "target/tmp/myRandomForestRegressionModel")



```
Tree 0:
  If (feature 489 <= 1.5)
   Predict: 0.0
  Else (feature 489 > 1.5)
   Predict: 1.0
Tree 1:
  If (feature 434 <= 79.5)
   Predict: 0.0
  Else (feature 434 > 79.5)
   Predict: 1.0
Tree 2:
  If (feature 406 <= 10.0)
   Predict: 0.0
  Else (feature 406 > 10.0)
   Predict: 1.0
```