TECNOLÓGICO NACIONAL DE MEXICO

INSTITUTO TECNOLOGICO DE TIJUANA

SUBDIRECCIÓN ACADÉMICA

DEPARTAMENTO DE INGENIERÍA EN SISTEMAS COMPUTACIONALES

SEMESTRE FEBRERO-JUNIO 2022

MATERIA:

Datos masivos.

UNIDAD 2

Practica 3

DOCENTE:

JOSE CHRISTIAN ROMERO HERNANDEZ

INTEGRANTES:

López Higuera Saúl Alfredo #18210493

Ramos Rivera Manuel Isaí #17212931

Tijuana BC 06 de mayo del 2022

```
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")
```

```
Learned classification forest model:
 TreeEnsembleModel classifier with 3 trees

  Tree 0:
    If (feature 272 <= 43.0)
     If (feature 378 <= 18.0)
      Predict: 0.0
     Else (feature 378 > 18.0)
      Predict: 1.0
    Else (feature 272 > 43.0)
     Predict: 0.0
  Tree 1:
    If (feature 518 <= 6.0)
     If (feature 323 <= 251.5)
      Predict: 0.0
     Else (feature 323 > 251.5)
      Predict: 1.0
    Else (feature 518 > 6.0)
     If (feature 387 <= 10.5)
      Predict: 1.0
     Else (feature 387 > 10.5)
      Predict: 0.0
  Tree 2:
    If (feature 435 <= 32.5)
     Predict: 0.0
    Else (feature 435 > 32.5)
     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")

```
Test Mean Squared Error = 0.02287581699346406

scala> println(s"Learned regression forest model:\n ${model.toDebugString}")
Learned regression forest model:
 TreeEnsembleModel regressor with 3 trees

  Tree 0:
    If (feature 407 <= 26.0)
     Predict: 0.0
    Else (feature 407 > 26.0)
     Predict: 1.0
  Tree 1:
    If (feature 433 <= 52.5)
     If (feature 295 <= 253.5)
      Predict: 0.0
     Else (feature 295 > 253.5)
      Predict: 1.0
    Else (feature 433 > 52.5)
     Predict: 1.0
  Tree 2:
    If (feature 406 <= 147.5)
     Predict: 0.0
    Else (feature 406 > 147.5)
     Predict: 1.0
```