**Tecnológico Nacional De México**

**Instituto Tecnológico De Tijuana**

**Subdirección Académica**

**Departamento de Sistemas y Computación**

**Semestre Enero - Junio 2022**

**Ingeniería Informática**

**Datos Masivos**

**Práctica 4 - Gradient-boosted tree classifier**

**Unidad 2**

**Perez Ortega Victoria Valeria No.18210718**

**Israel López Pablo No.17210585**

**JOSE CHRISTIAN ROMERO HERNANDEZ**

**Tijuana, B.C. a 18 de Mayo de 2022.**

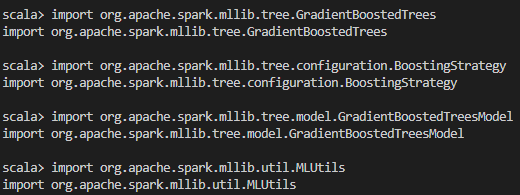
**Importación de librerías**

import org.apache.spark.mllib.tree.GradientBoostedTrees

import org.apache.spark.mllib.tree.configuration.BoostingStrategy

import org.apache.spark.mllib.tree.model.GradientBoostedTreesModel

import org.apache.spark.mllib.util.MLUtils



**Se carga y analiza el archivo de datos**

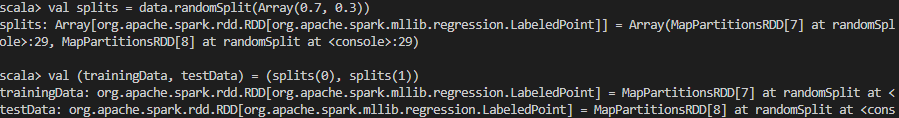
val data = MLUtils.loadLibSVMFile(sc, "data/mllib/sample\_libsvm\_data.txt")



**Se dividen los datos en conjuntos de entrenamiento y prueba (30% retenido para prueba)**

val splits = data.randomSplit(Array(0.7, 0.3))

val (trainingData, testData) = (splits(0), splits(1))



**Entrena un modelo GradientBoostedTrees.**

**Los parámetros predeterminados para la clasificación utilizan LogLoss de forma predeterminada.**

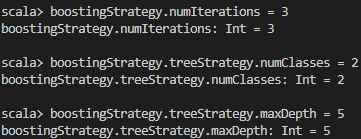
val boostingStrategy = BoostingStrategy.defaultParams("Classification")



boostingStrategy.numIterations = 3

boostingStrategy.treeStrategy.numClasses = 2

boostingStrategy.treeStrategy.maxDepth = 5



**Vacío categoricalFeaturesInfo indica que todas las características son continuas.**

boostingStrategy.treeStrategy.categoricalFeaturesInfo = Map[Int, Int]()



val model = GradientBoostedTrees.train(trainingData, boostingStrategy)



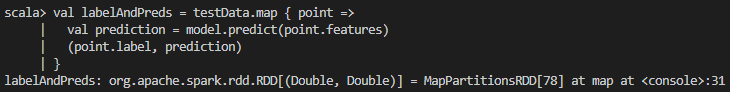
**Evaluar el modelo en instancias de prueba y calcular el error de prueba**

val labelAndPreds = testData.map { point =>

val prediction = model.predict(point.features)

(point.label, prediction)

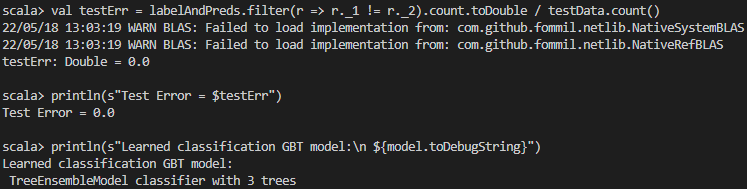
}

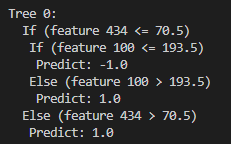


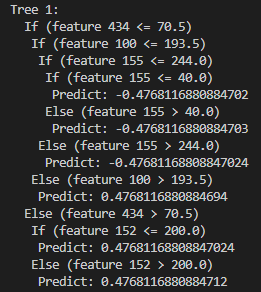
val testErr = labelAndPreds.filter(r => r.\_1 != r.\_2).count.toDouble / testData.count()

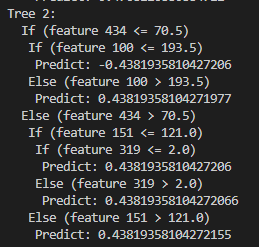
println(s"Test Error = $testErr")

println(s"Learned classification GBT model:\n ${model.toDebugString}")









**Guardar y cargar modelo**

model.save(sc, "target/tmp/myGradientBoostingClassificationModel")



val sameModel = GradientBoostedTreesModel.load(sc,

"target/tmp/myGradientBoostingClassificationModel")

