Tecnológico Nacional De México

Instituto Tecnológico De Tijuana

Subdirección Académica

Departamento de Sistemas y Computación

Semestre Enero - Junio 2022

Ingeniería Informática

Datos Masivos

Práctica 3 - Random Forest Classifier

Unidad 2

Perez Ortega Victoria Valeria **No.18210718**

Israel López Pablo **No.17210585**

JOSE CHRISTIAN ROMERO HERNANDEZ

Tijuana, B.C. a 06 de Mayo de 2022.

Documentar y ejecutar el ejemplo de la documentación de spark del **Random Forest classifier**, en su branch correspondiente.

https://spark.apache.org/docs/latest/ml-classification-regression.html

Pratice 3

el primer paso es que necesitamos agregar las liberias necesarias

```
package org.apache.spark.examples.ml
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}
import org.apache.spark.sql.SparkSession
```

Nosotros comenzamos la sesión con chispa

```
val spark = SparkSession
.builder
.appName("RandomForestClassifierExample")
.getOrCreate()
```

después de eso, necesitamos agregar los datos para el análisis.

```
   val data =
spark.read.format("libsvm").load("c:/Spark/data/mllib/sample_libsvm_data.txt"
)
```

ahora creamos el stringindexer para nuestro ejemplo

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

este paso no está contemplado en el documento de chispa pero lo necesitamos para otras próximas configuraciones

```
val labelindexed = labelIndexer.fit(data).transform(data)
```

aquí creamos nuestro vectorindex

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

este código crea 2 matrices en la primera matriz para datos de entrenamiento y la otra es para testdaf

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

ahora es necesario crear tu clasificador de bosque aleatorio

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

ahora necesitamos crear su convertidor de etiquetas, esto es generado por indextostring

```
   val labelConverter = new IndexToString()
      labelConverter.setInputCol("prediction")
      labelConverter.setOutputCol("predictedLabel")

labelConverter.setLabels(labelindexed.schema("indexedLabel").metadata.getMeta
data("ml_attr").getStringArray("vals"))
```

ahora creamos una pipeline y configuramos nuestros parámetros

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

ahora creamos nuestro modelo con nuestro entrenamiento de datos

```
val model = pipeline.fit(trainingData)
```

ahora este código es para nuestras predicciones, así que creamos nuestro valor

```
val predictions = model.transform(testData)
```

en esta parte del código mostramos 5 resultados para nuestro valor de predicción

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

Resultado

```
scala>     predictions.select("predictedLabel", "label", "features").sh
+--------------+-----+--------------------+
|predictedLabel|label|            features|
+--------------+-----+--------------------+
|           0.0|  0.0|(692,[124,125,126...|
|           0.0|  0.0|(692,[127,128,129...|
|           1.0|  1.0|(692,[123,124,125...|
|           1.0|  1.0|(692,[124,125,126...|
|           1.0|  1.0|(692,[125,126,127...|
+--------------+-----+--------------------+
only showing top 5 rows
```

por ahora, cree nuestro evaluador de clasificación multiclase y muestre si presenta un error en nuestro resultado.

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

Resultado

```
scala>  val evaluator = new MulticlassClassificationEvaluator()
evaluator: org.apache.spark.ml.evaluation.MulticlassClassificationEvalu

scala>       evaluator.setLabelCol("indexedLabel")
res17: evaluator.type = mcEval_93c2148c3e04

scala>       evaluator.setPredictionCol("prediction")
res18: evaluator.type = mcEval_93c2148c3e04

scala>       evaluator.setMetricName("accuracy")
res19: evaluator.type = mcEval_93c2148c3e04

scala>     val accuracy = evaluator.evaluate(predictions)
accuracy: Double = 1.0

scala>     println(s"Test Error = ${(1.0 - accuracy)}")
Test Error = 0.0
```

el último código muestra su resultado sobre cómo se crea la interacción en el modelo

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

Resultado

```
scala>     println(s"Learned classification forest model:\n ${rfModel.t
Learned classification forest model:
 RandomForestClassificationModel (uid=rfc_b9a6d1df4bd7) with 10 trees
  Tree 0 (weight 1.0):
    If (feature 412 <= 8.0)
     If (feature 454 <= 12.5)
      Predict: 0.0
     Else (feature 454 > 12.5)
      Predict: 1.0
    Else (feature 412 > 8.0)
     Predict: 1.0
  Tree 1 (weight 1.0):
    If (feature 463 <= 2.0)
     If (feature 380 <= 6.5)
      Predict: 1.0
     Else (feature 380 > 6.5)
      Predict: 0.0
    Else (feature 463 > 2.0)
     Predict: 0.0
  Tree 2 (weight 1.0):
    If (feature 540 <= 87.0)
     If (feature 510 <= 6.5)
      Predict: 0.0
     Else (feature 510 > 6.5)
      Predict: 1.0
    Else (feature 540 > 87.0)
     Predict: 1.0
  Tree 3 (weight 1.0):
    If (feature 328 <= 25.5)
     If (feature 261 <= 1.0)
      Predict: 0.0
     Else (feature 261 > 1.0)
      Predict: 1.0
    Else (feature 328 > 25.5)
     Predict: 1.0
  Tree 4 (weight 1.0):
    If (feature 429 <= 23.5)
     If (feature 358 <= 10.5)
      If (feature 481 <= 14.5)
       Predict: 0.0
      Else (feature 481 > 14.5)
       Predict: 1.0
     Else (feature 358 > 10.5)
      Predict: 1.0
    Else (feature 429 > 23.5)
     Predict: 1.0
```