**Tecnológico Nacional De México**

**Instituto Tecnológico De Tijuana**

**Subdirección Académica**

**Departamento de Sistemas y Computación**

**Semestre Enero - Junio 2022**

**Ingeniería Informática**

**Datos Masivos**

**Práctica 2 - Decision Tree Classifier**

**Unidad 2**

**Perez Ortega Victoria Valeria No.18210718**

**Israel López Pablo No.17210585**

**JOSE CHRISTIAN ROMERO HERNANDEZ**

**Tijuana, B.C. a 04 de Mayo de 2022.**

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

<https://spark.apache.org/docs/latest/mllib-decision-tree.html>

## **Practica 2**

En el primer paso, necesitamos importar la biblioteca necesaria para ejecutar el código.

import org.apache.spark.ml.Pipeline

import org.apache.spark.ml.classification.DecisionTreeClassificationModel

import org.apache.spark.ml.classification.DecisionTreeClassifier

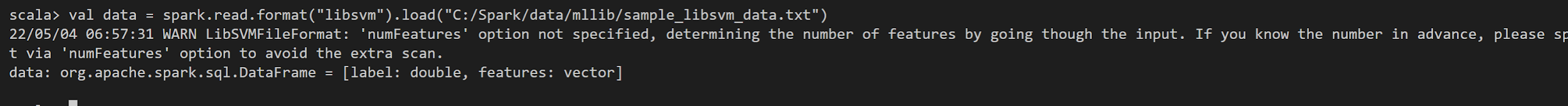
import org.apache.spark.ml.evaluation.MulticlassClassificationEvaluator

import org.apache.spark.ml.feature.{IndexToString, StringIndexer, VectorIndexer}

En este paso es necesario cargar los datos para este paso es necesario tomar el archivo "sample\_libsvm\_data.txt"

val data = spark.read.format("libsvm").load("C:/Spark/data/mllib/sample\_libsvm\_data.txt")

#### **Result**

****

necesitamos crear labelindexer y featureIndexer, esto es necesario para agregar metadatos y ajustar el conjunto de datos

val labelIndexer = new StringIndexer()

labelIndexer.setInputCol("label")

labelIndexer.setOutputCol("indexedLabel")

labelIndexer.fit(data)

val featureIndexer = new VectorIndexer()

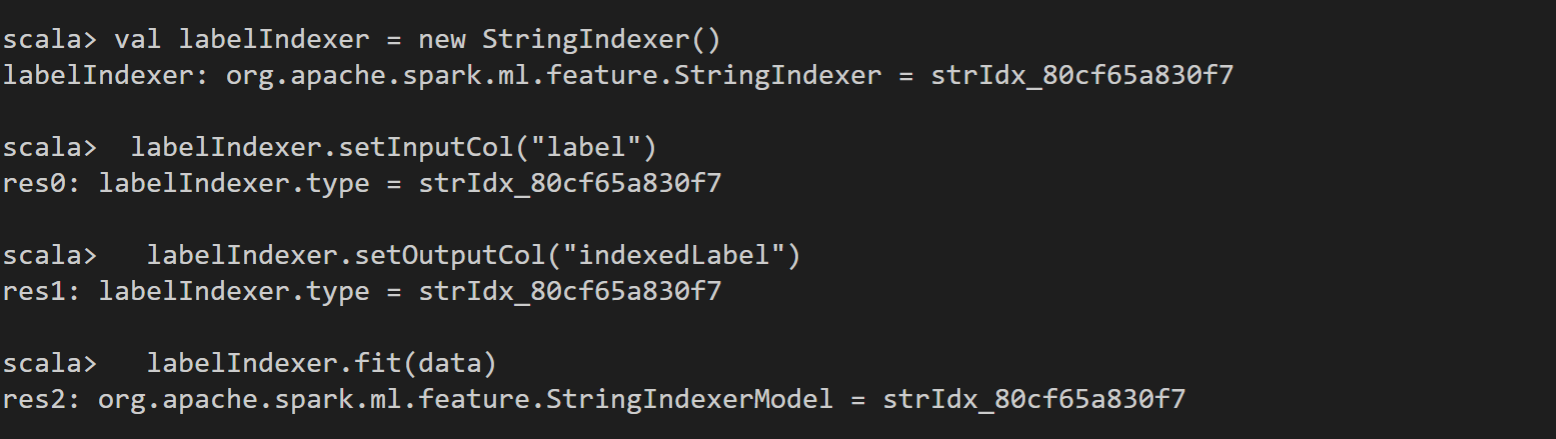
featureIndexer.setInputCol("features")

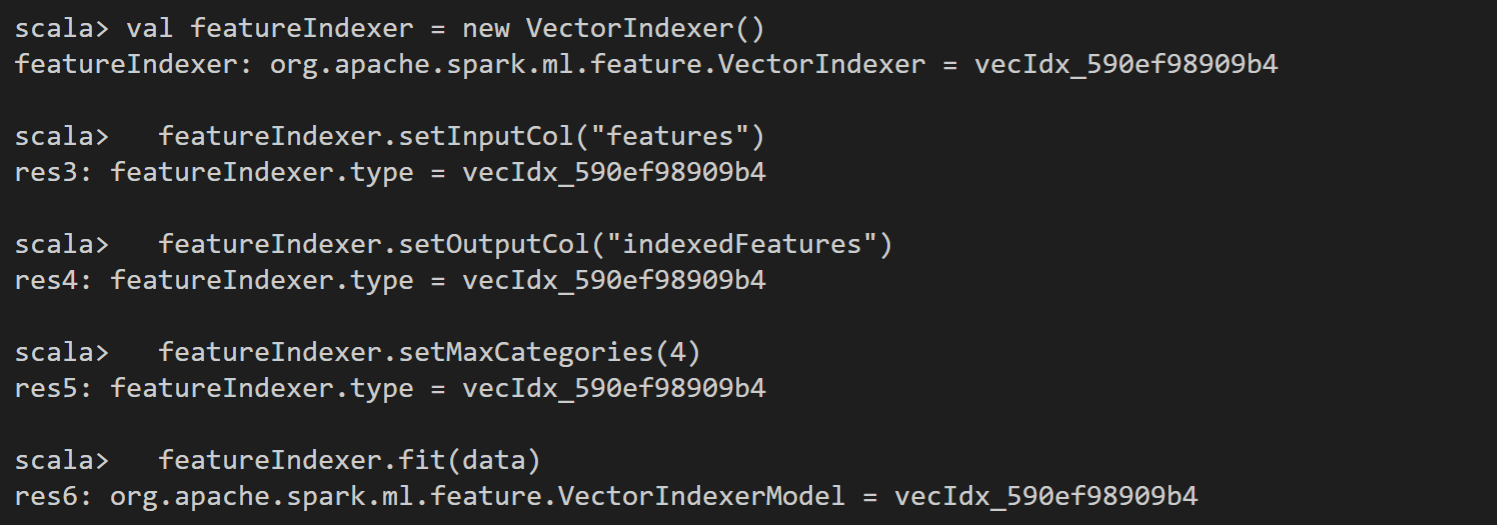
featureIndexer.setOutputCol("indexedFeatures")

featureIndexer.setMaxCategories(4)

featureIndexer.fit(data)

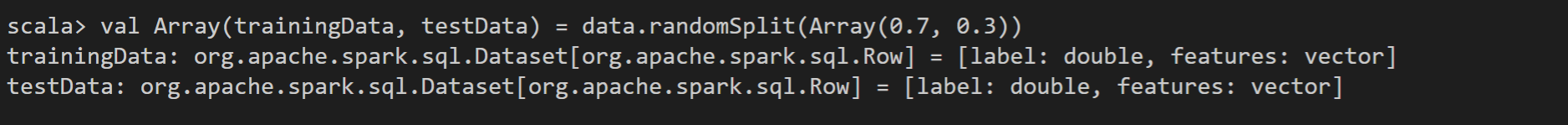
#### **Result**

****

****

Agregue el siguiente código para separar los datos para probar y entrenar

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



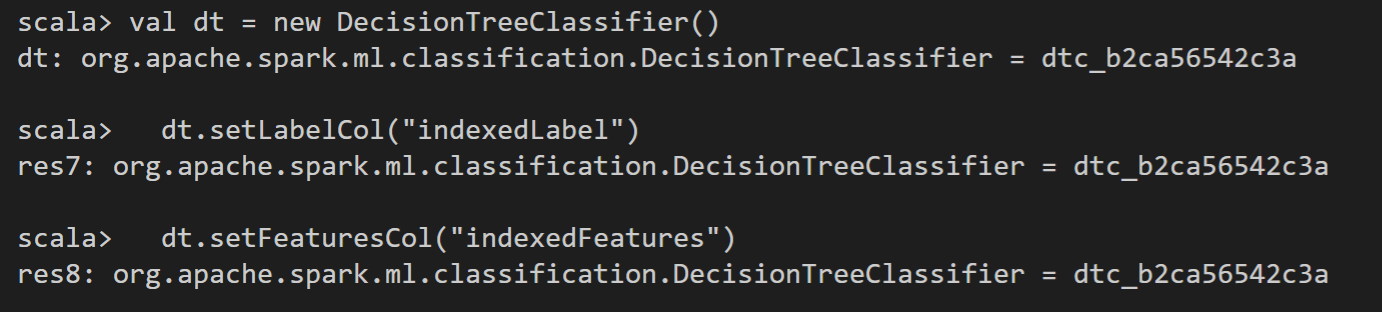
en esta sección hacemos todas las características necesarias para el modelo de tren

val dt = new DecisionTreeClassifier()

dt.setLabelCol("indexedLabel")

dt.setFeaturesCol("indexedFeatures")

#### **Result**

****

ahora volvemos a convertir etiquetas indexadas a originales

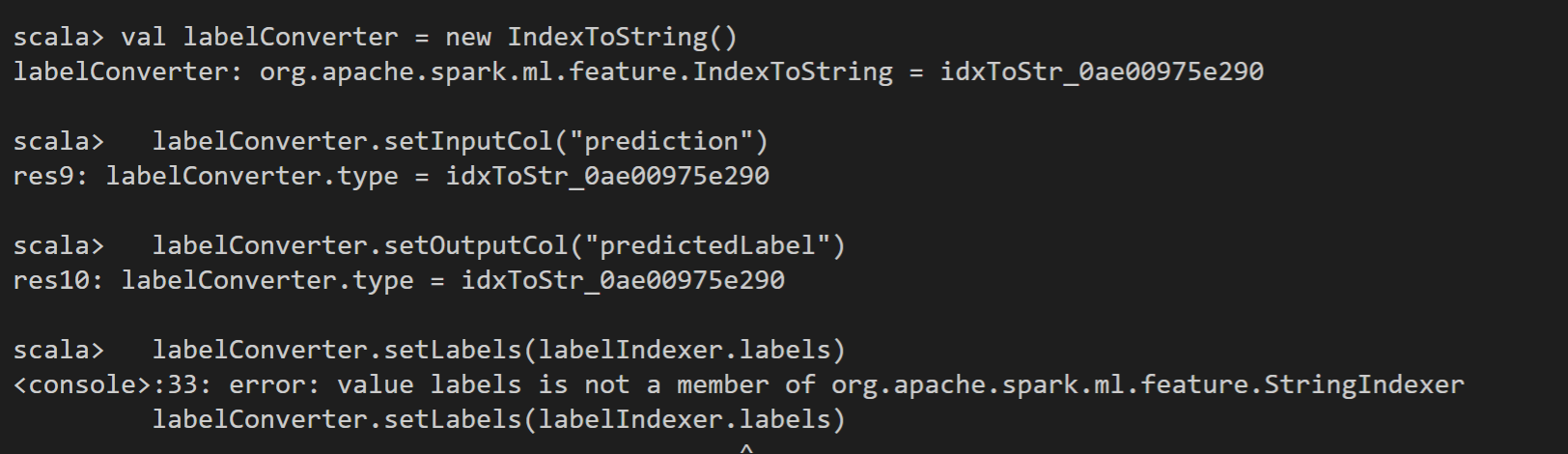
val labelConverter = new IndexToString()

labelConverter.setInputCol("prediction")

labelConverter.setOutputCol("predictedLabel")

labelConverter.setLabels(labelIndexer.labels)

#### **Result**

****

ahora cambiamos los indexadores y el árbol a la canalización

val pipeline = new Pipeline()

pipeline.setStages(Array(labelIndexer, featureIndexer, dt, labelConverter))

#### **Result**

Ahora es necesario entrenar al modelo.

val model = pipeline.fit(trainingData)

#### **Result**

Nosotros hacemos predicciones(masculino)

Buscar esto en

val predictions = model.transform(testData)

#### **Result**

En esta sección mostramos 5 filas

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

#### **Result**

este último código es seleccionar un error de prueba de cálculo

val evaluator = new MulticlassClassificationEvaluator()

.setLabelCol("indexedLabel")

.setPredictionCol("prediction")

.setMetricName("accuracy")

val accuracy = evaluator.evaluate(predictions)

println(s"Test Error = ${(1.0 - accuracy)}")

val treeModel = model.stages(2).asInstanceOf[DecisionTreeClassificationModel]

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