TECNOLÓGICO NACIONAL DE MÉXICO
INSTITUTO TECNOLÓGICO DE TIJUANA
SUBDIRECCIÓN ACADÉMICA

DEPARTAMENTO DE SISTEMAS Y COMPUTACIÓN

SEMESTRE FEBRERO-JUNIO 2022

CARRERA

Ingeniería en informática e Ingeniería en sistemas computacionales.

MATERIA
Datos masivos
TÍTULO
Práctica#2

Integrantes:

Munguía Silva Edgar Geovanny #17212344
Pérez López Alicia Guadalupe #18210514

NOMBRE DEL MAESTRO
José Christian Romero Hernández

Tijuana Baja California 03 de Mayo del 2022

```
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}

// Load the data stored in LIBSVM format as a DataFrame.
val data =
spark.read.format("libsvm").load("data/mllib/sample_libsvm_data.txt")
```

```
scala> val data = spark.read.format("libsvm").load("C:/Spark/data/mllib/sample_libsvm_data.txt")
22/05/04 06:57:31 WARN LibSVMFileFormat: 'numFeatures' option not specified, determining the number of features by going though the input. If you know the number in advance, please sp
t via 'numFeatures' option to avoid the extra scan.
data: org.apache.spark.sql.DataFrame = [label: double, features: vector]
```

```
// Index labels, adding metadata to the label column.
// Fit on whole dataset to include all labels in index.
val labelIndexer = new StringIndexer()
  .setInputCol("label")
  .setOutputCol("indexedLabel")
  .fit(data)
// Automatically identify categorical features, and index them.
val featureIndexer = new VectorIndexer()
  .setInputCol("features")
  .setOutputCol("indexedFeatures")
  .setMaxCategories(4) // features with > 4 distinct values are treated as
continuous.
  .fit(data)

// Split the data into training and test sets (30% held out for testing).
val Array(trainingData, testData) = data.randomSplit(Array(0.7, 0.3))

// Train a DecisionTree model.
val dt = new DecisionTreeClassifier()
  .setLabelCol("indexedLabel")
  .setFeaturesCol("indexedFeatures")
```

```
scala> val dt = new DecisionTreeClassifier()
dt: org.apache.spark.ml.classification.DecisionTreeClassifier = dtc_b2ca56542c3a

scala>   dt.setLabelCol("indexedLabel")
res7: org.apache.spark.ml.classification.DecisionTreeClassifier = dtc_b2ca56542c3a

scala>   dt.setFeaturesCol("indexedFeatures")
res8: org.apache.spark.ml.classification.DecisionTreeClassifier = dtc_b2ca56542c3a
```

```
// Convert indexed labels back to original labels.
val labelConverter = new IndexToString()
  .setInputCol("prediction")
  .setOutputCol("predictedLabel")
  .setLabels(labelIndexer.labels)
```

```
scala> val labelConverter = new IndexToString()
labelConverter: org.apache.spark.ml.feature.IndexToString = idxToStr_0ae00975e290

scala>   labelConverter.setInputCol("prediction")
res9: labelConverter.type = idxToStr_0ae00975e290

scala>   labelConverter.setOutputCol("predictedLabel")
res10: labelConverter.type = idxToStr_0ae00975e290

scala>   labelConverter.setLabels(labelIndexer.labels)
<console>:33: error: value labels is not a member of org.apache.spark.ml.feature.StringIndexer
         labelConverter.setLabels(labelIndexer.labels)
```

```
// Chain indexers and tree in a Pipeline.
val pipeline = new Pipeline()
  .setStages(Array(labelIndexer, featureIndexer, dt, labelConverter))

// Train model. This also runs the indexers.
val model = pipeline.fit(trainingData)

// Make predictions.
val predictions = model.transform(testData)

// Select example rows to display.
predictions.select("predictedLabel", "label", "features").show(5)

// Select (prediction, true label) and compute test error.
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}")
```