TECNOLÓGICO NACIONAL DE MÉXICO

INSTITUTO TECNOLÓGICO DE TIJUANA

SUBDIRECCIÓN ACADÉMICA

DEPARTAMENTO DE SISTEMAS Y COMPUTACIÓN

NOMBRE DE LOS ALUMNOS:

RAYMUNDO HIRALES LAZARENO (N. CONTROL: 17212339)

PAULA ANDREA RAMOS VERDIN (N.CONTROL: 18210721)

Carrera: Ingeniería Informática

Semestre: 10mo

MATERIA: Datos masivos

PROFESOR: JOSE CHRISTIAN ROMERO HERNANDEZ

TRABAJOS: Practica 2

FECHA: 05/05/22

```
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")

// 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")

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