TECNOLÓGICO NACIONAL DE MÉXICO

INSTITUTO TECNOLÓGICO DE TIJUANA

SUBDIRECCIÓN ACADÉMICA

DEPARTAMENTO DE SISTEMAS Y COMPUTACIÓN

NOMBRE DE LOS ALUMNOS:

GALAVIZ LONA OSCAR EDUARDO (N.CONTROL: 17212993)

MARQUEZ MILLAN SEASHELL VANESSA (N.CONTROL:)

Carrera: Ingeniería Informática

Semestre: 9no

MATERIA: Datos Masivos

PROFESOR: JOSE CHRISTIAN ROMERO HERNANDEZ

Practica 3

Unidad 2

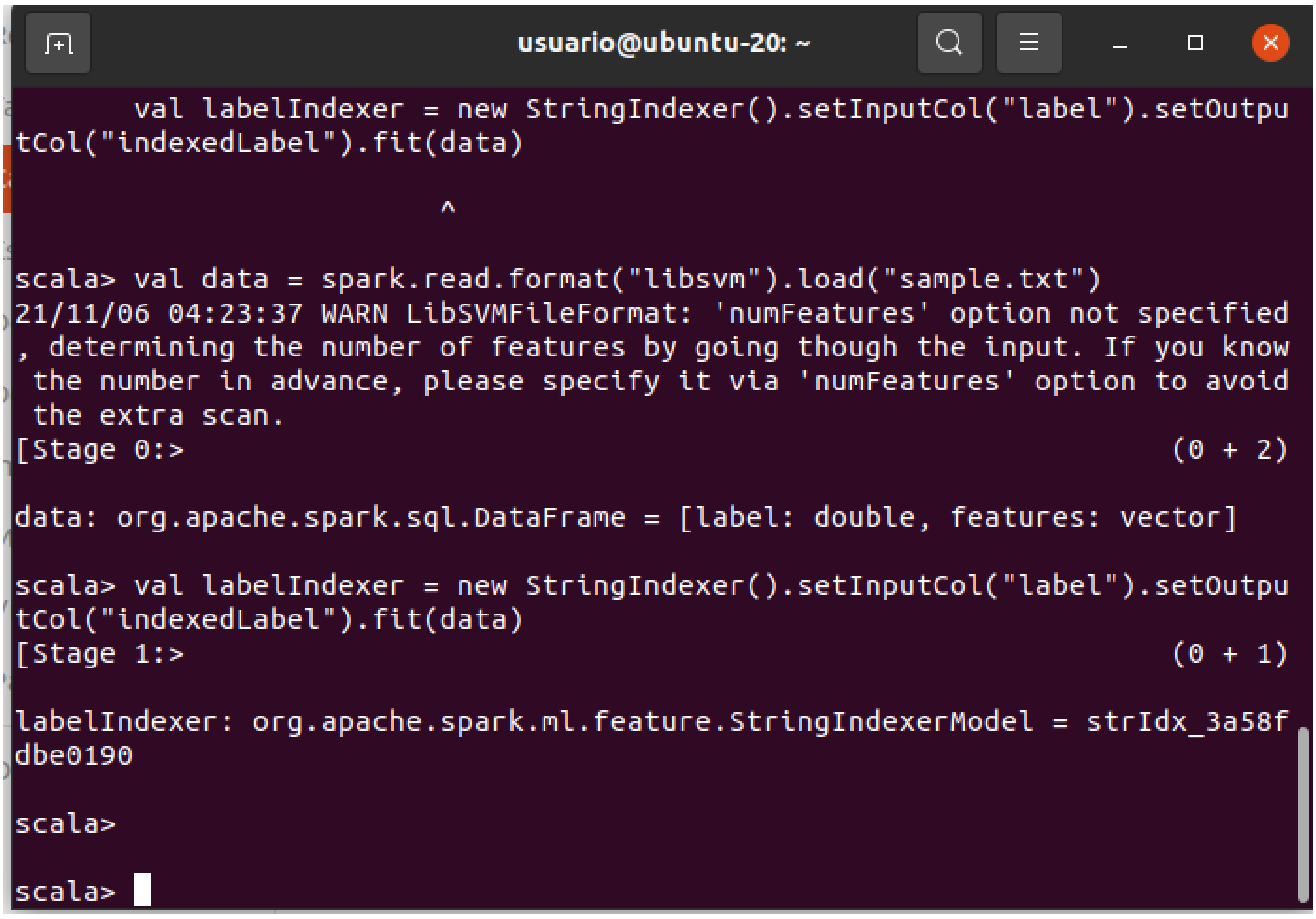
Development

we import the libraries that practice requires

```
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}
```

here we load the data from a text file to be the dataframe

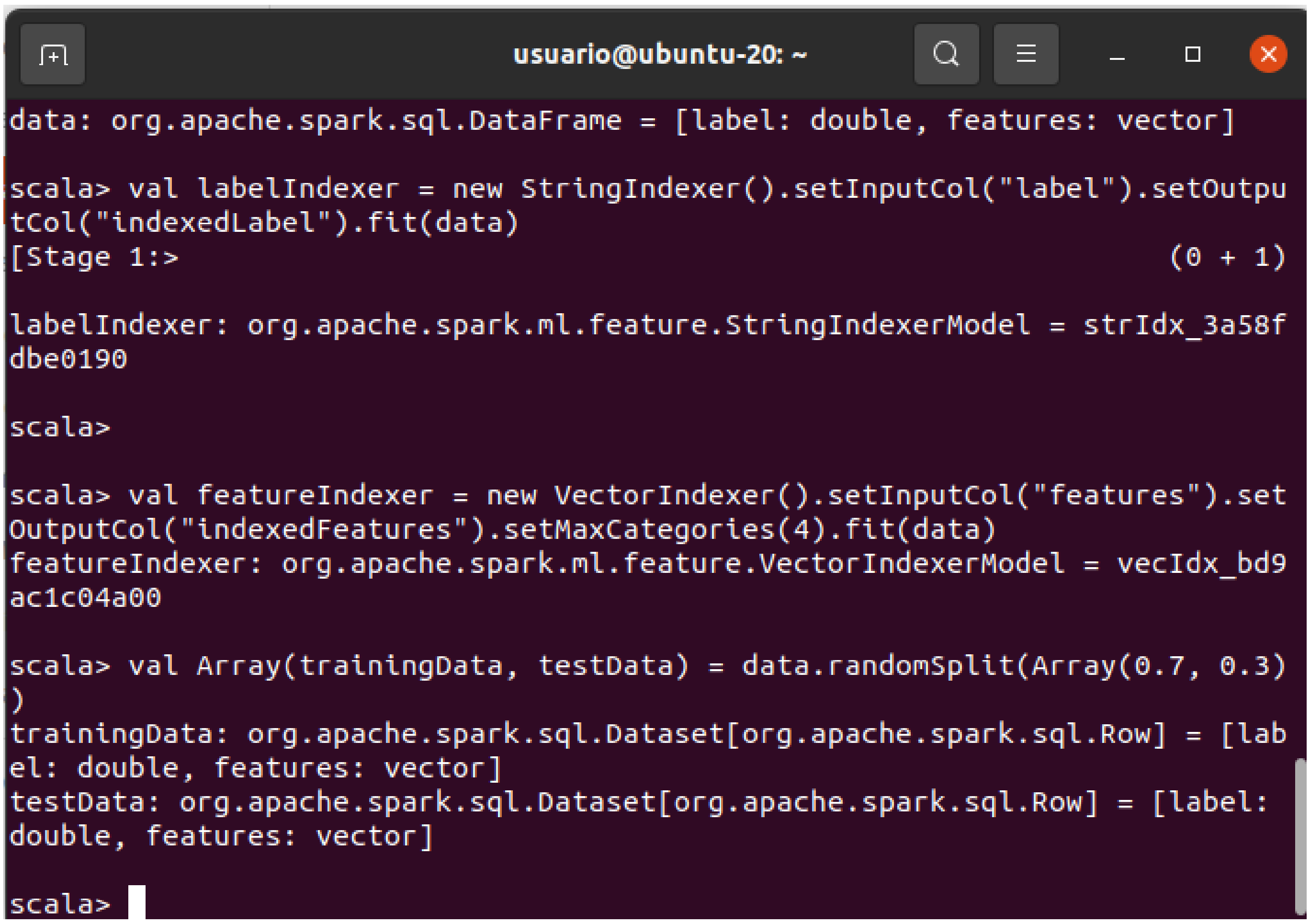
```


val data = spark.read.format("libsvm").load("sample.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)
```



```
       val labelIndexer = new StringIndexer().setInputCol("label").setOutpu
tCol("indexedLabel").fit(data)

                         ^

scala> val data = spark.read.format("libsvm").load("sample.txt")
21/11/06 04:23:37 WARN LibSVMFileFormat: 'numFeatures' option not specified
, determining the number of features by going though the input. If you know
 the number in advance, please specify it via 'numFeatures' option to avoid
 the extra scan.
[Stage 0:>                                                          (0 + 2)

data: org.apache.spark.sql.DataFrame = [label: double, features: vector]

scala> val labelIndexer = new StringIndexer().setInputCol("label").setOutpu
tCol("indexedLabel").fit(data)
[Stage 1:>                                                          (0 + 1)

labelIndexer: org.apache.spark.ml.feature.StringIndexerModel = strIdx_3a58f
dbe0190

scala>

scala> 
```

```
// Automatically identify categorical features, and index them.
// Set maxCategories so features with > 4 distinct values are treated as
```

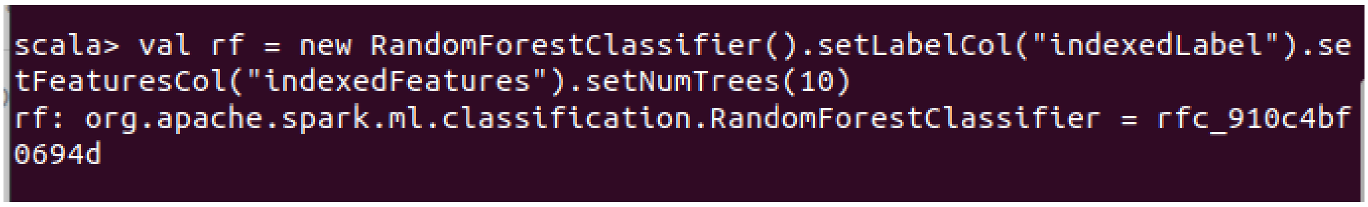
```
continuous.
val featureIndexer = new
VectorIndexer().setInputCol("features").setOutputCol("indexedFeatures").setMaxCate
gories(4).fit(data)

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



```
usuario@ubuntu-20: ~
data: org.apache.spark.sql.DataFrame = [label: double, features: vector]

scala> val labelIndexer = new StringIndexer().setInputCol("label").setOutpu
tCol("indexedLabel").fit(data)
[Stage 1:>                                                          (0 + 1)

labelIndexer: org.apache.spark.ml.feature.StringIndexerModel = strIdx_3a58f
dbe0190

scala>

scala> val featureIndexer = new VectorIndexer().setInputCol("features").set
OutputCol("indexedFeatures").setMaxCategories(4).fit(data)
featureIndexer: org.apache.spark.ml.feature.VectorIndexerModel = vecIdx_bd9
ac1c04a00

scala> val Array(trainingData, testData) = data.randomSplit(Array(0.7, 0.3)
)
trainingData: org.apache.spark.sql.Dataset[org.apache.spark.sql.Row] = [lab
el: double, features: vector]
testData: org.apache.spark.sql.Dataset[org.apache.spark.sql.Row] = [label:
double, features: vector]

scala>
```

Train a RandomForest model.

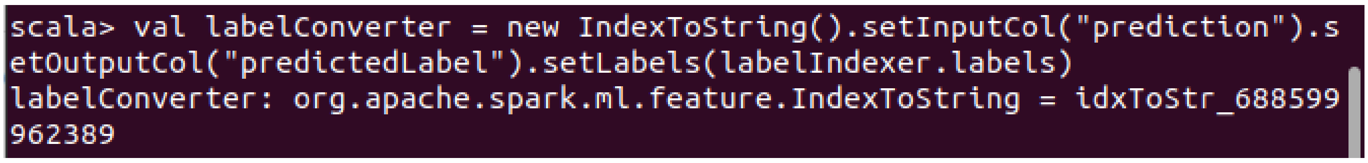
```
val rf = new
RandomForestClassifier().setLabelCol("indexedLabel").setFeaturesCol("indexedFeatur
es").setNumTrees(10)
```



```
scala> val rf = new RandomForestClassifier().setLabelCol("indexedLabel").se
tFeaturesCol("indexedFeatures").setNumTrees(10)
rf: org.apache.spark.ml.classification.RandomForestClassifier = rfc_910c4bf
0694d
```

Convert indexed labels back to original labels.

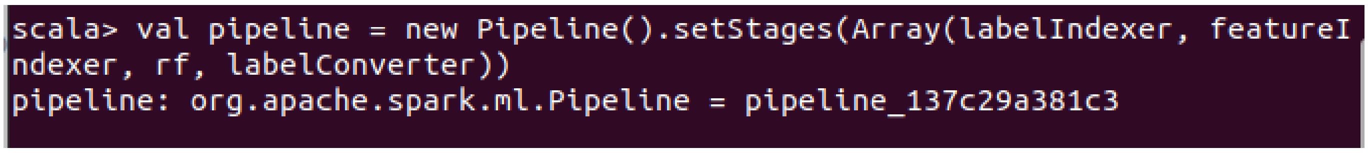
```
val labelConverter = new
IndexToString().setInputCol("prediction").setOutputCol("predictedLabel").setLabels
(labelIndexer.labels)
```



```
scala> val labelConverter = new IndexToString().setInputCol("prediction").s
etOutputCol("predictedLabel").setLabels(labelIndexer.labels)
labelConverter: org.apache.spark.ml.feature.IndexToString = idxToStr_688599
962389
```

Chain indexers and forest in a Pipeline.

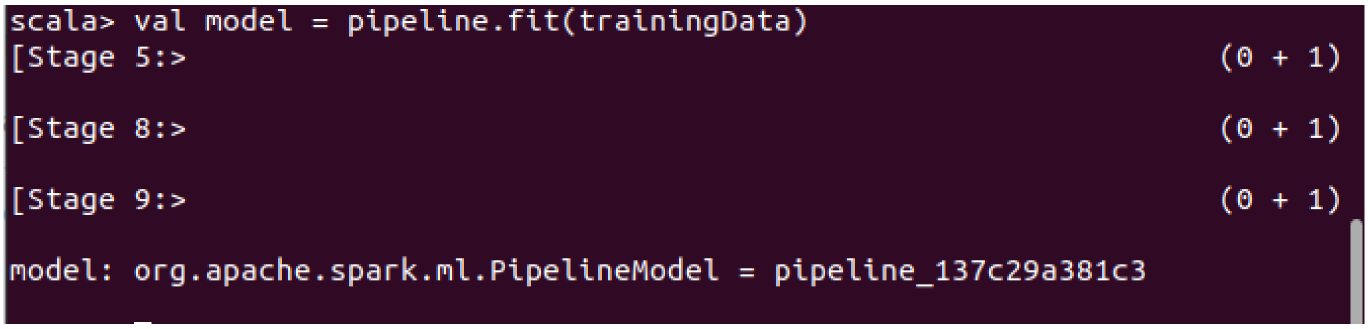
```
val pipeline = new Pipeline().setStages(Array(labelIndexer, featureIndexer, rf,
labelConverter))
```



```
scala> val pipeline = new Pipeline().setStages(Array(labelIndexer, featureI
ndexer, rf, labelConverter))
pipeline: org.apache.spark.ml.Pipeline = pipeline_137c29a381c3
```

Train model. This also runs the indexers.

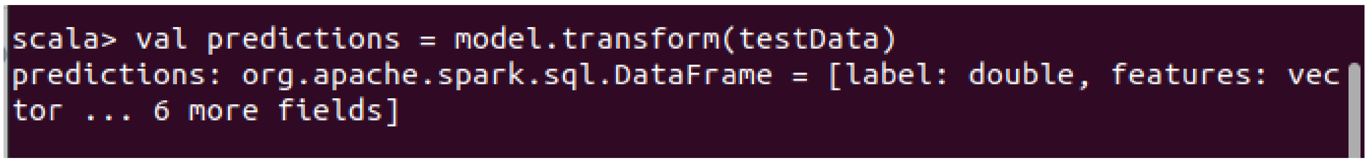
```
val model = pipeline.fit(trainingData)
```



```
scala> val model = pipeline.fit(trainingData)
[Stage 5:>                                                          (0 + 1)

[Stage 8:>                                                          (0 + 1)

[Stage 9:>                                                          (0 + 1)

model: org.apache.spark.ml.PipelineModel = pipeline_137c29a381c3
```

Make predictions.

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



```
scala> val predictions = model.transform(testData)
predictions: org.apache.spark.sql.DataFrame = [label: double, features: vec
tor ... 6 more fields]
```

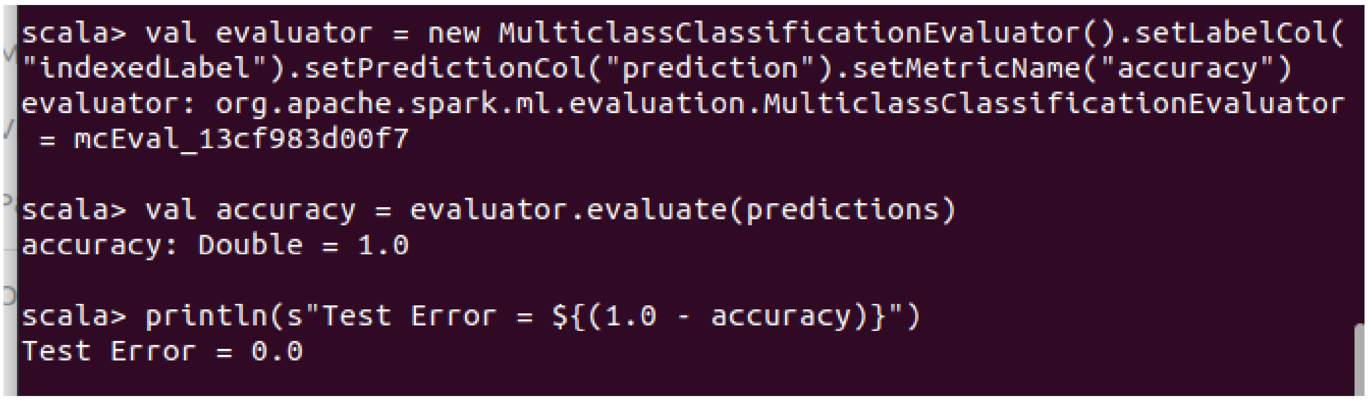
Select example rows to display.

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

Logo

Select (prediction, true label) and compute test error.

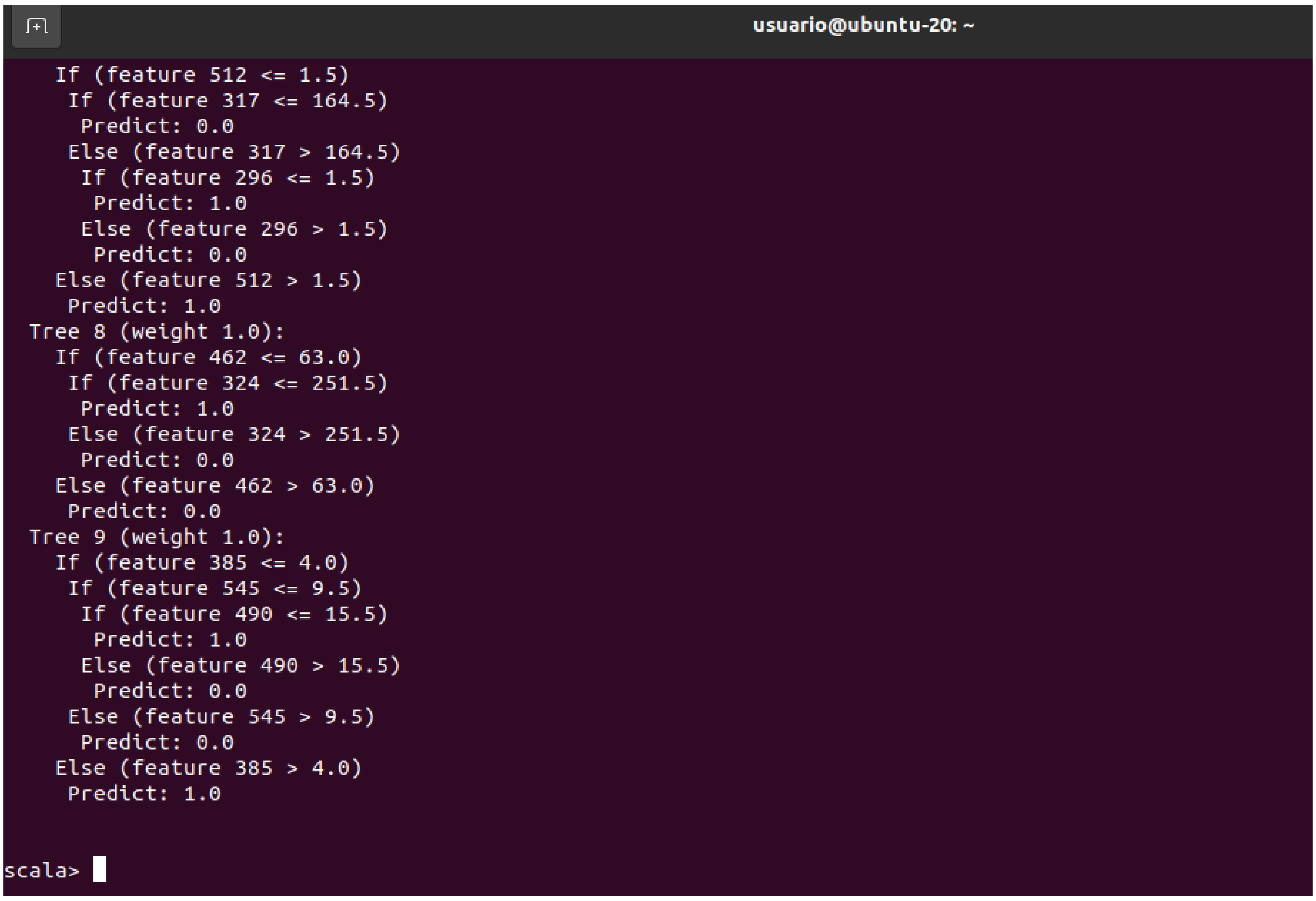
```
val evaluator = new
MulticlassClassificationEvaluator().setLabelCol("indexedLabel").setPredictionCol("
prediction").setMetricName("accuracy")
val accuracy = evaluator.evaluate(predictions)
println(s"Test Error = ${(1.0 - accuracy)}")
```



```
scala> val evaluator = new MulticlassClassificationEvaluator().setLabelCol(
"indexedLabel").setPredictionCol("prediction").setMetricName("accuracy")
evaluator: org.apache.spark.ml.evaluation.MulticlassClassificationEvaluator
 = mcEval_13cf983d00f7

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

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

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

Logo


usuario@ubuntu-20: ~

```
    If (feature 512 <= 1.5)
     If (feature 317 <= 164.5)
      Predict: 0.0
     Else (feature 317 > 164.5)
      If (feature 296 <= 1.5)
       Predict: 1.0
      Else (feature 296 > 1.5)
       Predict: 0.0
    Else (feature 512 > 1.5)
     Predict: 1.0
  Tree 8 (weight 1.0):
    If (feature 462 <= 63.0)
     If (feature 324 <= 251.5)
      Predict: 1.0
     Else (feature 324 > 251.5)
      Predict: 0.0
    Else (feature 462 > 63.0)
     Predict: 0.0
  Tree 9 (weight 1.0):
    If (feature 385 <= 4.0)
     If (feature 545 <= 9.5)
      If (feature 490 <= 15.5)
       Predict: 1.0
      Else (feature 490 > 15.5)
       Predict: 0.0
     Else (feature 545 > 9.5)
      Predict: 0.0
    Else (feature 385 > 4.0)
     Predict: 1.0


scala>
```