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

Practice 4

Unidad 2

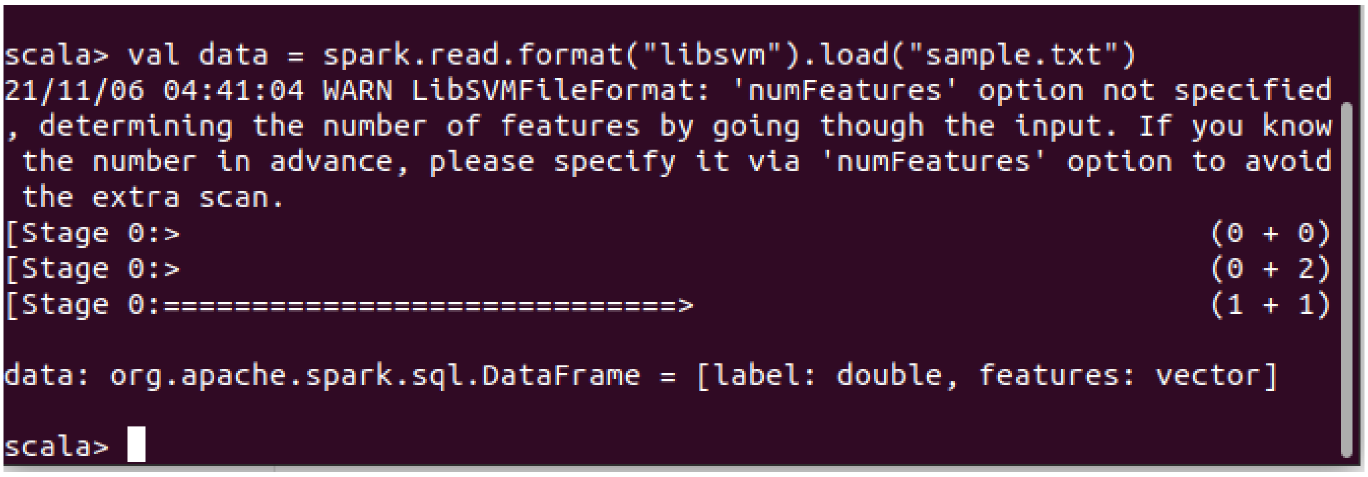
Development

we have to import every library we need for the practice

```
import org.apache.spark.ml.Pipeline
import org.apache.spark.ml.classification.{GBTClassificationModel, GBTClassifier}
import org.apache.spark.ml.evaluation.MulticlassClassificationEvaluator
import org.apache.spark.ml.feature.{IndexToString, StringIndexer, VectorIndexer}
```

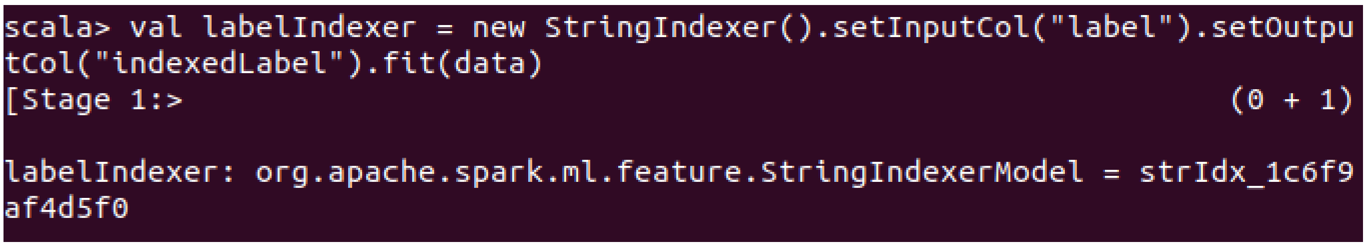
here load a data file, we made use to a dataframe

```
// Load and parse the data file, converting it to a DataFrame.
val data = spark.read.format("libsvm").load("sample.txt")
```



```
scala> val data = spark.read.format("libsvm").load("sample.txt")
21/11/06 04:41:04 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 + 0)
[Stage 0:>                                                          (0 + 2)
[Stage 0:==============================>                            (1 + 1)

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

scala> 
```

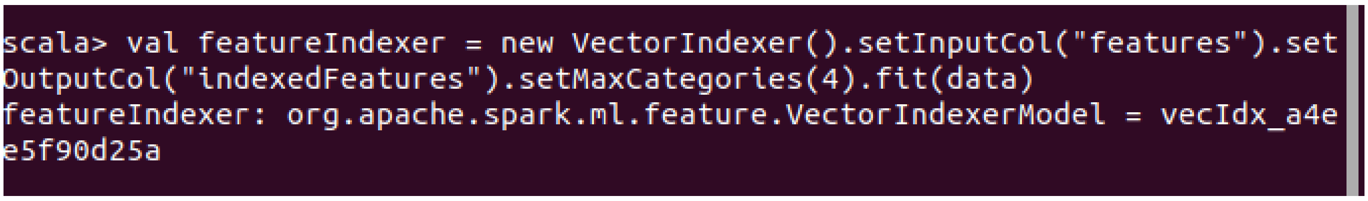
```
// Index labels, adding metadata to the label column.
// Fit on whole dataset to include all labels in index.
val labelIndexer = new
StringIndexer().setInputCol("label").setOutputCol("indexedLabel").fit(data)
// Automatically identify categorical features, and index them.
// Set maxCategories so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



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

labelIndexer: org.apache.spark.ml.feature.StringIndexerModel = strIdx_1c6f9
af4d5f0
```

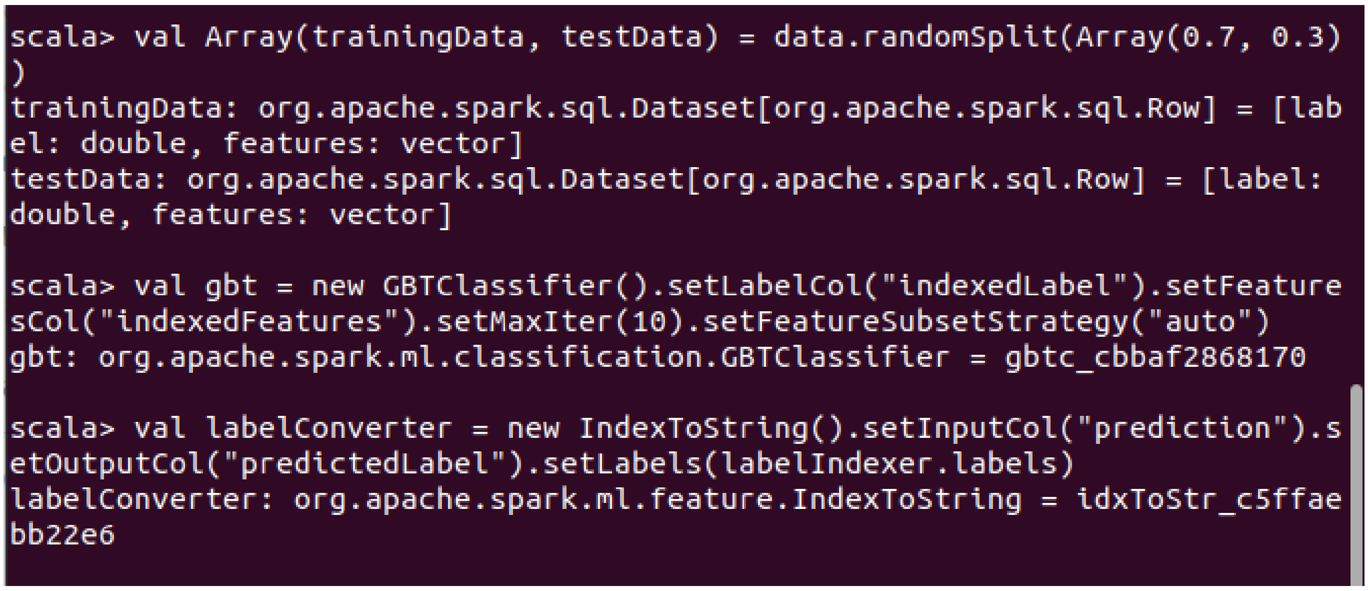
```
val featureIndexer = new
VectorIndexer().setInputCol("features").setOutputCol("indexedFeatures").setMaxCate
```

```
gories(4).fit(data)
```



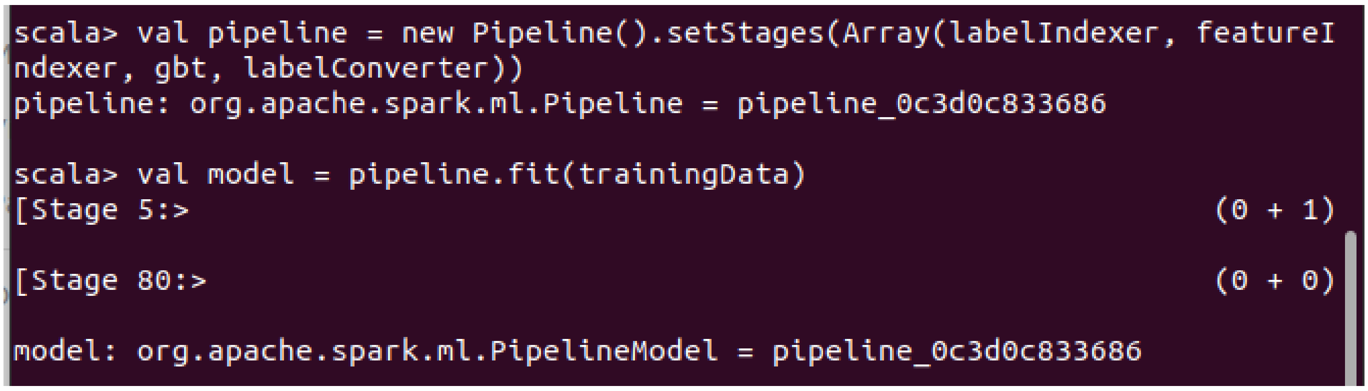
```
scala> val featureIndexer = new VectorIndexer().setInputCol("features").set
OutputCol("indexedFeatures").setMaxCategories(4).fit(data)
featureIndexer: org.apache.spark.ml.feature.VectorIndexerModel = vecIdx_a4e
e5f90d25a
```

```
// 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 GBT model.
val gbt = new
GBTClassifier().setLabelCol("indexedLabel").setFeaturesCol("indexedFeatures").setM
axIter(10).setFeatureSubsetStrategy("auto")

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



```
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> val gbt = new GBTClassifier().setLabelCol("indexedLabel").setFeature
sCol("indexedFeatures").setMaxIter(10).setFeatureSubsetStrategy("auto")
gbt: org.apache.spark.ml.classification.GBTClassifier = gbtc_cbbaf2868170

scala> val labelConverter = new IndexToString().setInputCol("prediction").s
etOutputCol("predictedLabel").setLabels(labelIndexer.labels)
labelConverter: org.apache.spark.ml.feature.IndexToString = idxToStr_c5ffae
bb22e6
```

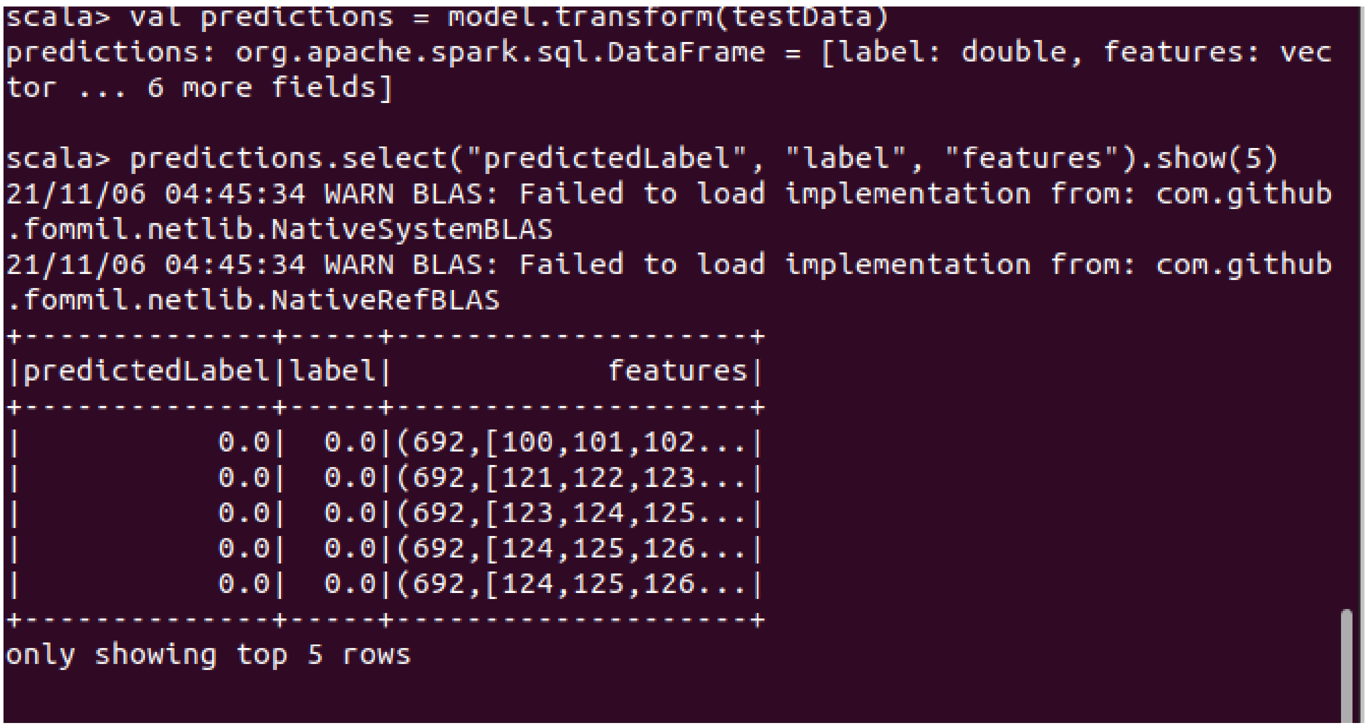
```
// Chain indexers and GBT in a Pipeline.
val pipeline = new Pipeline().setStages(Array(labelIndexer, featureIndexer, gbt,
labelConverter))

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



```
scala> val pipeline = new Pipeline().setStages(Array(labelIndexer, featureI
ndexer, gbt, labelConverter))
pipeline: org.apache.spark.ml.Pipeline = pipeline_0c3d0c833686

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

[Stage 80:>                                                         (0 + 0)

model: org.apache.spark.ml.PipelineModel = pipeline_0c3d0c833686
```

Make predictions.

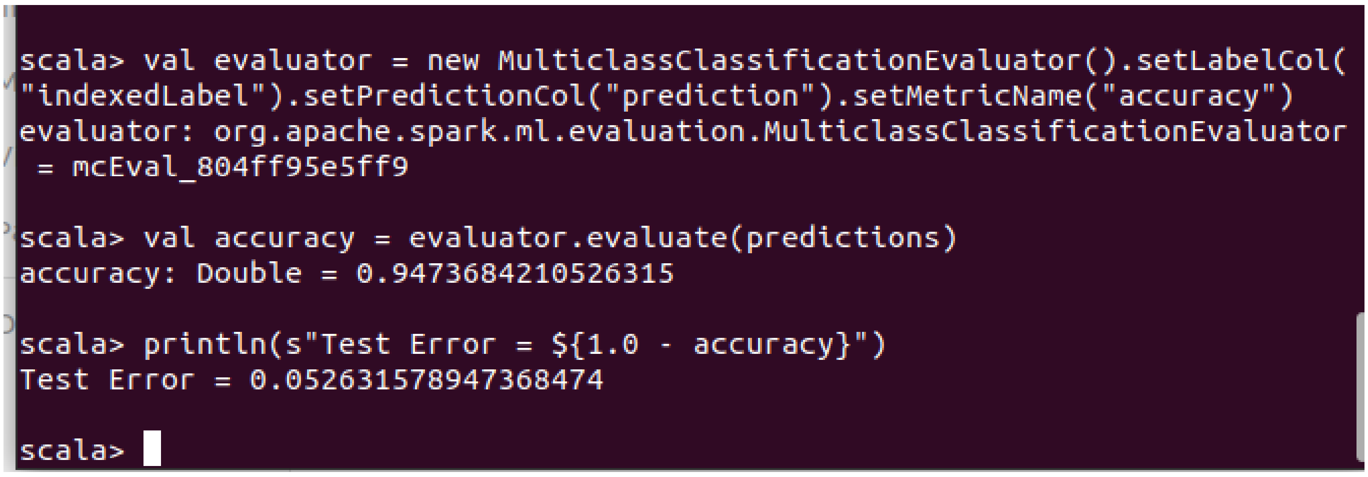
```
val predictions = model.transform(testData)

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



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

scala> predictions.select("predictedLabel", "label", "features").show(5)
21/11/06 04:45:34 WARN BLAS: Failed to load implementation from: com.github
.fommil.netlib.NativeSystemBLAS
21/11/06 04:45:34 WARN BLAS: Failed to load implementation from: com.github
.fommil.netlib.NativeRefBLAS
+--------------+-----+--------------------+
|predictedLabel|label|            features|
+--------------+-----+--------------------+
|           0.0|  0.0|(692,[100,101,102...|
|           0.0|  0.0|(692,[121,122,123...|
|           0.0|  0.0|(692,[123,124,125...|
|           0.0|  0.0|(692,[124,125,126...|
|           0.0|  0.0|(692,[124,125,126...|
+--------------+-----+--------------------+
only showing top 5 rows
```

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

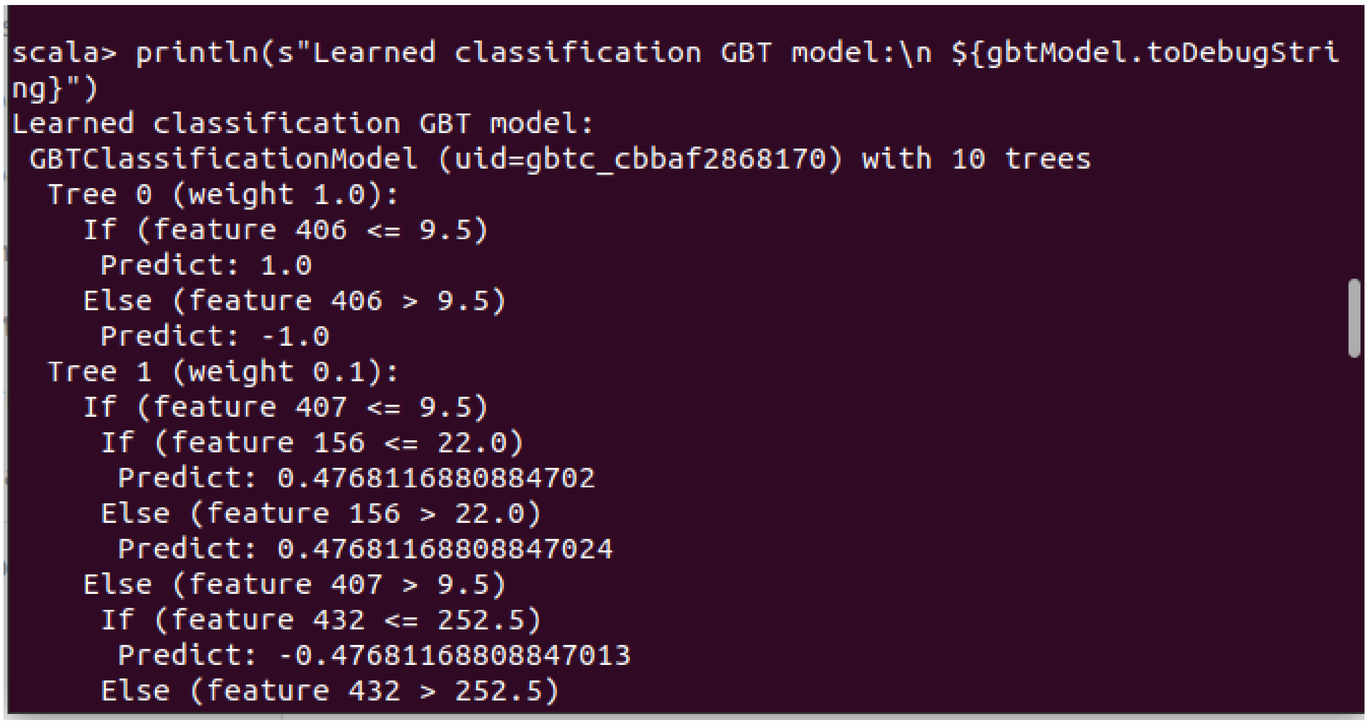
```
val evaluator = new
MulticlassClassificationEvaluator().setLabelCol("indexedLabel").setPredictionCol("
prediction").setMetricName("accuracy")
val accuracy = evaluator.evaluate(predictions)
println(s"Test Error = ${1.0 - accuracy}")
```



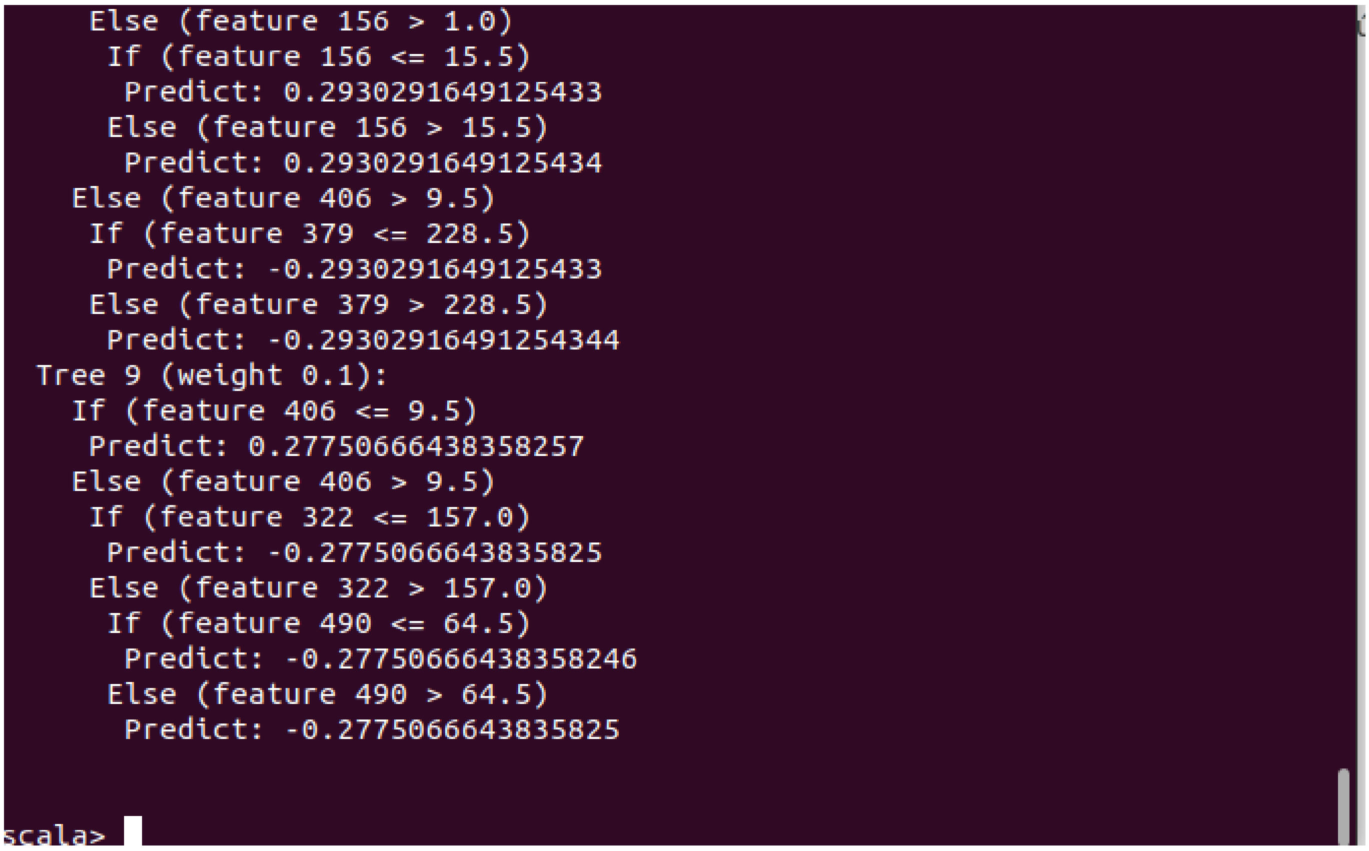
```
scala> val evaluator = new MulticlassClassificationEvaluator().setLabelCol(
"indexedLabel").setPredictionCol("prediction").setMetricName("accuracy")
evaluator: org.apache.spark.ml.evaluation.MulticlassClassificationEvaluator
 = mcEval_804ff95e5ff9

scala> val accuracy = evaluator.evaluate(predictions)
accuracy: Double = 0.9473684210526315

scala> println(s"Test Error = ${1.0 - accuracy}")
Test Error = 0.052631578947368474

scala> 
```

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

finally as a result we have several trees that each has branches inside



```
scala> println(s"Learned classification GBT model:\n ${gbtModel.toDebugStri
ng}")
Learned classification GBT model:
 GBTClassificationModel (uid=gbtc_cbbaf2868170) with 10 trees
  Tree 0 (weight 1.0):
    If (feature 406 <= 9.5)
     Predict: 1.0
    Else (feature 406 > 9.5)
     Predict: -1.0
  Tree 1 (weight 0.1):
    If (feature 407 <= 9.5)
     If (feature 156 <= 22.0)
      Predict: 0.4768116880884702
     Else (feature 156 > 22.0)
      Predict: 0.47681168808847024
    Else (feature 407 > 9.5)
     If (feature 432 <= 252.5)
      Predict: -0.47681168808847013
     Else (feature 432 > 252.5)
```



```
    Else (feature 156 > 1.0)
     If (feature 156 <= 15.5)
      Predict: 0.2930291649125433
     Else (feature 156 > 15.5)
      Predict: 0.2930291649125434
   Else (feature 406 > 9.5)
    If (feature 379 <= 228.5)
     Predict: -0.2930291649125433
    Else (feature 379 > 228.5)
     Predict: -0.29302916491254344
 Tree 9 (weight 0.1):
   If (feature 406 <= 9.5)
    Predict: 0.27750666438358257
   Else (feature 406 > 9.5)
    If (feature 322 <= 157.0)
     Predict: -0.2775066643835825
    Else (feature 322 > 157.0)
     If (feature 490 <= 64.5)
      Predict: -0.27750666438358246
     Else (feature 490 > 64.5)
      Predict: -0.2775066643835825


scala>
```