**NATIONAL TECHNOLOGICAL INSTITUTE OF MEXICO**
**TECHNOLOGICAL INSTITUTE OF TIJUANA**

ACADEMIC SUBDIRECTION
Systems and Computing Department

SEMESTER
August – December 2021

ACADEMIC CAREER
Eng. Information and Comunications Technologies

SUBJECT AND KEY:
Big Data BDD – 1704TI9A

STUDENT'S NAME AND REGISTRATION:
Velázquez Farrera César Alejandro

NAME OF WORK:
Practice #1 – Decision Tree Classifier

UNIT:
Unit II

NAME OF TEACHER:
M.C. José Christian Romero Hernández

**Tijuana Baja California** **Due Date:** November 5th, 2021

## Introduction

In this present document, the topic of Decision Tree Classifier, how it works and examples will be explained. The example in this document is done with the documentation provided by Apache Spark in the following URL: [https://spark.apache.org/docs/2.4.7/ml-classification-regression.html#decision-trees]

Decision trees and their ensembles are popular methods for machine learning tasks of classification and regression. Decision trees are widely used, since they are easy to interpret, handle categorical features, extend to multi-class classification settings, do not require feature scaling, and they are able to capture non—linearity and feature interactions.

The decision trees use the CART algorithm (Classification and Regression Trees). In both cases, decision trees are based on conditions on any of the features. The internal nodes represent the decision based on conditions.

On each step or node of a decision tree, used for classification, we try to form a condition on the features to separate all the labels or classes contained in the dataset to the fullest purity.

We can continue comparing our record's attribute with other internal nodes of the tree until we reach a leaf node with a predicted class value. As we know the modeled decision tree can be used to predict the target class or the value.

## Development

The example uses a Spark-Shell environment, powered by Java Runtime and as a input language Scala is used. Spark is available in MacOS, Windows and Linux Distros.

The example below demostrates how to load a LibSVM data file, parse it as an RDD of the column Labeled Point and then perform classifications using decision tree with GiniImpurity as a mesure, and lastly add a maximum tree depth of 5.

**Step 1:** Invoke the following libraries that will be used throughout the example. The libraries are:

- tree.DecisionTree
- tree.Model.DecisionTreeModel
- util.MLUtils

```
import org.apache.spark.mllib.tree.DecisionTree
import org.apache.spark.mllib.tree.model.DecisionTreeModel

import org.apache.spark.mllib.util.MLUtils
```

**Step 2:** Load and split the data file into two parts (training_set, test_set) in 70% and 30% of the data respectively.

```
val data = MLUtils.loadLibSVMFile(sc, "data/mllib/sample_libsvm_data.txt")

val splits = data.randomSplit(Array(0.7, 0.3))

val (trainingData, testData) = (splits(0), splits(1))
```

**Step 3:** Set the environment values and train the decision tree model.

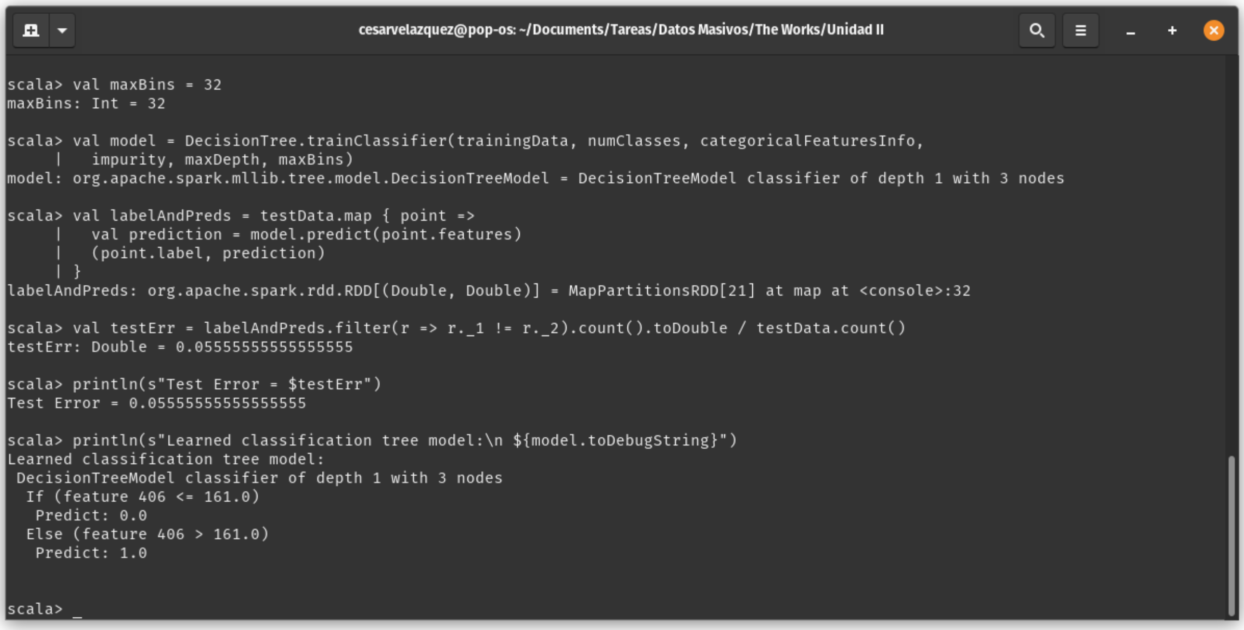
```
val numClasses = 2

val categoricalFeaturesInfo = Map[Int, Int]()

val impurity = "gini"

val maxDepth = 5

val maxBins = 32

val model = DecisionTree.trainClassifier(trainingData, numClasses, categoricalFeaturesInfo,

  impurity, maxDepth, maxBins)
```

**Step 4:** Evaluate the model with the remaining data set (test_set) and compute the test error.

```
val labelAndPreds = testData.map { point =>

    val prediction = model.predict(point.features)

    (point.label, prediction)

}

val testErr = labelAndPreds.filter(r => r._1 != r._2).count().toDouble / testData.count()

println(s"Test Error = $testErr")

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

## Conclusions

Decision trees are one of the easiest machine learning models to understand (can be represented as a diagram) and cover the aspects of regression and classification.


cesarvelazquez@pop-os: ~/Documents/Tareas/Datos Masivos/The Works/Unidad II

```
scala> val maxBins = 32
maxBins: Int = 32

scala> val model = DecisionTree.trainClassifier(trainingData, numClasses, categoricalFeaturesInfo,
     |   impurity, maxDepth, maxBins)
model: org.apache.spark.mllib.tree.model.DecisionTreeModel = DecisionTreeModel classifier of depth 1 with 3 nodes

scala> val labelAndPreds = testData.map { point =>
     |   val prediction = model.predict(point.features)
     |   (point.label, prediction)
     | }
labelAndPreds: org.apache.spark.rdd.RDD[(Double, Double)] = MapPartitionsRDD[21] at map at <console>:32

scala> val testErr = labelAndPreds.filter(r => r._1 != r._2).count().toDouble / testData.count()
testErr: Double = 0.05555555555555555

scala> println(s"Test Error = $testErr")
Test Error = 0.05555555555555555

scala> println(s"Learned classification tree model:\n ${model.toDebugString}")
Learned classification tree model:
 DecisionTreeModel classifier of depth 1 with 3 nodes
  If (feature 406 <= 161.0)
   Predict: 0.0
  Else (feature 406 > 161.0)
   Predict: 1.0


scala> _
```

As we can see in the image above, we can see that the testing error is around 5%. This means that the model predicts the data 95% correctly.