**Tecnológico Nacional de México**
**Instituto Tecnológico de Tijuana**

**Subdirección Académica**
**Departamento de Sistemas y Computación**

**Semestre:**
Agosto – Diciembre 2021

**Carrera:**
Ingeniería en Tecnologías de la Información y Comunicaciones
Ingeniería en Sistemas Computacionales

**Materia y serie:**
Datos Masivos BDD-1704TI9A

**Unidad a evaluar:** Unidad II

**Nombre de la Tarea:**
Práctica 4

**Nombre del Alumno:**
Flores Gonzalez Luis Diego C16211486
Sifuentes Martinez Manuel Javier 17212934

**Nombre del docente:**
José Christian Romero Hernández

## Practice 4

Gradient-boosted tree maintains the same structure for this classification, only changing when the model is created. First you will find the import of all the libraries to use.

```
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}
```

After importing the dataset, specifying the format to be used and the directory where the file is located.

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

Already having the data in the data variable, the labelIndexer and VectorIndexer variables are declared, the first one is for the data that is trying to get and the second for the characteristics that will be used to obtain said data, the columns are indexed so they are reassigned with a temporary name specified in the method setOutputCol

```
val labelIndexer = new StringIndexer() .setInputCol ("label")
.setOutputCol ("indexedLabel") .fit (data)
val featureIndexer = new VectorIndexer() .setInputCol ("features")
.setOutputCol ("indexedFeatures") .setMaxCategories (4) .fit (data)
```

To perform the classification, two data vectors are needed, one with which the model will be trained (trainingData ) taking 70% of the data and another with which the predictions will be made and the effectiveness of the model will be verified (testData) with 30% of the remaining data.

```
val Array(trainingData, testData) = data.randomSplit (Array(0.7, 0.3))
```

Then a GBTClassifier model must be created, specifying which will be the label column, the features column and the maximum number of iterations to perform.

```
val gbt = new GBTClassifier() .setLabelCol ("indexedLabel")
.setFeaturesCol ("indexedFeatures") .setMaxIter (10)
.setFeatureSubsetStrategy ("auto")
```

Here the indexedLabel column is converted to label.

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

The pipeline is created, which is responsible for creating the process through which the training and test data will pass. After the model is created, which is entered with the training data, this data goes through the pipeline and the result is stored in the model variable.

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


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

Once the model has been trained, the predictions are made, they pass through the pipeline again and the results are stored in the variable predictions.

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

The results of the predictions can be observed with the help of the select and show method, in the first the columns are specified and in the second the number of rows to be displayed.

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

**Result:**

```
+ -------------- + ----- + ----- --------------- +
|predictedLabel|label|            features|
+ -------------- + ----- + ------------------- +
|           0.0|  0.0| (692, [98,99,100,1...|
|           0.0|  0.0| (692, [100,101,102...|
|           0.0|  0.0| (692, [124,125,126. ..|
|           0.0|  0.0| (692, [125,126,127...|
|           0.0|  0.0| (692, [126,127,128...|
+ ---------- ---- + ----- + ------------------- +
only showing top 5 rows
```

The evaluator stores the precision with which the data was predicted.

```
val evaluator = new MulticlassClassificationEvaluator() .setLabelCol
("indexedLabel") .setPredictionCol ("prediction") .setMetricName
("accuracy")
```

And by having the precision, the error can also be obtained, subtracting entirely the percentage of precision obtained.

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

**Result:**

```
accuracy: Double = 1.0
Test Error = 0.0
```

At the end, all the procedures performed in Gradient-boosted tree are displayed to get the results seen previously.

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

**Result:**

```
Learned classification GBT model:
 GBTClassificationModel (uid=gbtc_b5eac4259fd7) with 10 trees
  Tree 0 (weight 1.0):
    If (feature 434 <= 88.5)
     If (feature 99 in {2.0})
      Predict: -1.0
     Else (feature 99 not in {2.0})
      Predict: 1.0
    Else (feature 434 > 88.5)
     Predict: -1.0
  Tree 1 (weight 0.1):
    If (feature 490 <= 29.0)
     If (feature 568 <= 253.5)
      If (feature 154 <= 57.5)
       Predict: 0.4768116880884702
      Else (feature 154 > 57.5)
       Predict: 0.47681168808847024
     Else (feature 568 > 253.5)
      Predict: -0.4768116880884694
    Else (feature 490 > 29.0)
     If (feature 323 <= 112.5)
      Predict: -0.47681168808847024
     Else (feature 323 > 112.5)
      Predict: -0.47681168808847035
  Tree 2 (weight 0.1):
    If (feature 434 <= 88.5)
     If (feature 626 <= 2.5)
      Predict: -0.4381935810427206
     Else (feature 626 > 2.5)
      Predict: 0.43819358104272055
    Else (feature 434 > 88.5)
     If (feature 323 <= 13.0)
      Predict: -0.4381935810427206
     Else (feature 323 > 13.0)
      Predict: -0.43819358104272066
  Tree 3 (weight 0.1):
    If (feature 462 <= 62.5)
```

```
    If (feature 213 <= 85.5)
     Predict: -0.4051496802845983
    Else (feature 213 > 85.5)
     Predict: 0.40514968028459825
  Else (feature 462 > 62.5)
   If (feature 432 <= 4.0)
    Predict: -0.40514968028459825
   Else (feature 432 > 4.0)
    Predict: -0.4051496802845983
Tree 4 (weight 0.1):
  If (feature 490 <= 29.0)
   If (feature 548 <= 253.5)
    If (feature 578 <= 56.5)
     Predict: 0.3765841318352991
    Else (feature 578 > 56.5)
     Predict: 0.3765841318352993
   Else (feature 548 > 253.5)
    Predict: -0.3765841318352994
  Else (feature 490 > 29.0)
   If (feature 432 <= 58.5)
    If (feature 377 <= 88.5)
     Predict: -0.3765841318352991
    Else (feature 377 > 88.5)
     Predict: -0.37658413183529926
   Else (feature 432 > 58.5)
    Predict: -0.3765841318352994
Tree 5 (weight 0.1):
  If (feature 434 <= 88.5)
   If (feature 99 in {2.0})
    Predict: -0.35166478958101005
   Else (feature 99 not in {2.0})
    If (feature 156 <= 9.0)
     Predict: 0.35166478958101005
    Else (feature 156 > 9.0)
     Predict: 0.3516647895810101
  Else (feature 434 > 88.5)
   If (feature 351 <= 176.5)
    If (feature 181 <= 2.0)
     Predict: -0.35166478958101005
    Else (feature 181 > 2.0)
     Predict: -0.3516647895810101
   Else (feature 351 > 176.5)
```

```
    Predict: -0.35166478958101005
Tree 6 (weight 0.1):
  If (feature 490 <= 29.0)
   If (feature 548 <= 253.5)
    If (feature 233 <= 198.5)
     Predict: 0.32974984655529926
    Else (feature 233 > 198.5)
     Predict: 0.3297498465552994
   Else (feature 548 > 253.5)
    Predict: -0.32974984655530015
  Else (feature 490 > 29.0)
   Predict: -0.32974984655529915
Tree 7 (weight 0.1):
  If (feature 434 <= 88.5)
   If (feature 239 <= 253.5)
    If (feature 211 <= 160.0)
     Predict: 0.3103372455197956
    Else (feature 211 > 160.0)
     If (feature 456 <= 31.5)
      Predict: 0.3103372455197956
     Else (feature 456 > 31.5)
      Predict: 0.3103372455197957
   Else (feature 239 > 253.5)
    Predict: -0.31033724551979525
  Else (feature 434 > 88.5)
   If (feature 294 <= 99.5)
    If (feature 463 <= 53.5)
     Predict: -0.3103372455197956
    Else (feature 463 > 53.5)
     Predict: -0.3103372455197957
   Else (feature 294 > 99.5)
    If (feature 323 <= 112.5)
     Predict: -0.3103372455197956
    Else (feature 323 > 112.5)
     If (feature 211 <= 72.0)
      Predict: -0.3103372455197956
     Else (feature 211 > 72.0)
      Predict: -0.3103372455197957
Tree 8 (weight 0.1):
  If (feature 434 <= 88.5)
   If (feature 243 <= 4.0)
    Predict: -0.2930291649125432
```

```
    Else (feature 243 > 4.0)
     If (feature 210 <= 221.0)
      Predict: 0.2930291649125433
     Else (feature 210 > 221.0)
      Predict: 0.2930291649125434
   Else (feature 434 > 88.5)
    If (feature 462 <= 136.0)
     Predict: -0.2930291649125433
    Else (feature 462 > 136.0)
     Predict: -0.2930291649125434
Tree 9 (weight 0.1):
   If (feature 490 <= 29.0)
    If (feature 239 <= 253.5)
     If (feature 577 <= 10.0)
      Predict: 0.27750666438358246
     Else (feature 577 > 10.0)
      Predict: 0.27750666438358257
    Else (feature 239 > 253.5)
     Predict: -0.2775066643835826
   Else (feature 490 > 29.0)
    If (feature 377 <= 132.5)
     Predict: -0.2775066643835825
    Else (feature 377 > 132.5)
     Predict: -0.27750666438358257
```