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 7

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

Nombre del docente:
José Christian Romero Hernández

Practice 7

First, the libraries that will make it possible to create the NaiveBayes object must be loaded , and consequently being able to perform the classification by this method.

```
import org.apache.spark.ml.classification.NaiveBayes
import
org.apache.spark.ml.evaluation.MulticlassClassificationEvaluator
```

The data set is loaded into the data variable thanks to the read method, which together with format (specifies the format in which the data comes) and load (the directory where the file is located is passed as a parameter) allows you to take the dataset to use.

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

To perform the classification you need two data sets, one for training and one for testing, the first one helps to adjust the model to the data, allowing the NaiveBayes method to better yield predictions with the test data set.

```
val Array(trainingData, testData) = data.randomSplit (Array(0.7,
0.3), seed = 1234L)
```

This is where the model is trained, adjusting it to the data provided through the data variable, this is saved in the model variable , in this the test data will be entered so that, already having reference data classified by NaiveBayes, it can give the results of the predictions.

```
val model = new NaiveBayes() .fit (trainingData)
```

With the help of the transform method, the test data set is passed to the model already saved in the model variable, this so that predictions can be made, which are shown with the help of the show method. In this way you can view the results obtained after going through the NaiveBayes classification.

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

Result:

```
+ ----- + -------------------- + ------ -------------- + ----------- +
---------- +
|label|            features|       rawPrediction|probability|prediction|
+ ----- + -------------------- + -------------------- + - ---------- +
---------- +
|  0.0| (692, [95,96,97,12...| [-173678.60946628...| [1.0,0.0] |       0.0|
|  0.0| (692, [98,99,100,1...| [-178107.24302988...| [1.0,0.0] |       0.0|
|  0.0| (692, [100,101,102...| [-100020.80519087...| [1.0,0.0] |       0.0|
|  0.0| (692, [124,125,126...| [-183521.85526462...| [1.0,0.0] |       0.0|
|  0.0| (692, [127,128,129...| [-183004.12461660...| [1.0,0.0] |       0.0|
|  0.0| (692, [128,129,130...| [-246722.96394714...| [1.0,0.0] |       0.0|
|  0.0| (692, [152,153,154...| [-208696.01108598...| [1.0,0.0] |       0.0|
|  0.0| (692, [153,154,155...| [-261509.59951302...| [1.0,0.0] |       0.0|
|  0.0| (692, [154,155,156...| [-217654.71748256...| [1.0,0.0] |       0.0|
|  0.0| (692, [181,182,183...| [-155287.07585335...| [1.0,0.0] |       0.0|
|  1.0| (692, [99,100,101,...| [-145981.83877498...| [0.0,1.0] |       1.0|
|  1.0| (692, [100,101,102...| [-147685.13694275...| [0.0,1.0] |       1.0|
|  1.0| (692, [123,124,125...| [-139521.98499849...| [0.0,1.0] |       1.0|
|  1.0| (692, [124,125,126...| [-129375.46702012. ..| [0.0,1.0] |       1.0|
|  1.0| (692, [126,127,128...| [-145809.08230799...| [0.0,1.0] |       1.0|
|  1.0| (692, [127,128,129...| [-132670.15737290...| [0.0,1.0] |       1.0|
|  1.0| (692, [128,129,130...| [-100206.72054749...| [0.0,1.0] |       1.0|
|  1.0| (692, [129,130,131...| [-129639.09694930...| [0.0,1.0] |       1.0|
|  1.0| (692, [129,130,131...| [-143628.65574273...| [0.0,1.0] |       1.0|
|  1.0| (692, [129,130,131...| [-129238.74023248...| [0.0,1.0] |       1.0|
+ - --- + -------------------- + -------------------- + ---- ------- +
---------- +
only showing top twenty rows
```

Once the results have been obtained, it is valuable to know the rate of precision obtained from the process performed, for this the MulticlassClassificationEvaluator object is created, which will have the task of obtaining the precision obtained after passing the predicted data set as a parameter.

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

At the end, with the help of the evaluate method, the set of predicted data is passed to evaluate the precision obtained by NaiveBayes, it is saved in the accuracy variable and is printed with the println method.

```
val accuracy = evaluator.evaluate (predictions)
println (s "Test set accuracy = $ accuracy")
```

Result:

```
accuracy: Double = 1.0
Test set accuracy = 1.0
```