EDUCACIÓN
SECRETARÍA DE EDUCACIÓN PÚBLICA

TECNOLÓGICO NACIONAL DE MÉXICO

Tecnológico Nacional de México
Instituto Tecnológico de Tijuana

ACADEMIC SUBDIRECTION
Systems and Computing Department

SEMESTER
August - December 2021

CAREER
Information and Communication Technologies Engineer

SUBJECT AND KEY:
Big Data BDD-1704TI9A

STUDENT'S NAME AND REGISTRATION:

Castillo Ramirez Guadalupe 17213043
Velázquez Farrera César Alejandro 17212937

NAME OF THE JOB:
Practice # 7 - Naive Bayes

UNIT TO BE EVALUATED
Unit II

TEACHER'S NAME:
M.C. Jose Christian Romero Hernandez

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

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

Load the data stored in LIBSVM format as a DataFrame. 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 taking the dataset to use.

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

Split the data into training and test sets (30% held out for testing). To perform the classification, two data sets are needed, one for training and one for testing, the first helps to fit the model to the data, allowing the NaiveBayes method to better yield the predictions with the test data set.

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

Train a NaiveBayes model. 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 which the test data will be entered so that, already having reference data classified by NaiveBayes, it can return the results of the predictions.

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

Select example rows to display. 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 the 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()
```

Select (prediction, true label) and compute test error. Once the results have been obtained, it is valuable to know the rate of precision obtained from the process carried out, 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("predictio
n").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 it 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
```