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 5

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

Nombre del docente:
José Christian Romero Hernández

Practice 5

In order to carry out this type of classification, the respective libraries must be imported for creation of the object, so the MultilayerPerceptronClassifier and MulticlassClassificationEvaluator libraries are imported.

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

The data set must be loaded, and this is done with the read method, first specifying the format in which the data comes, and then, with the load method, the directory where the file is located is placed, the data is stored in the variable data.

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

Having already loaded the data set in memory, they must be divided randomly to be able, first, to train the model, and later do the tests to see what results it yields having carried out the classification procedures. The random split method is used, specifying that 60% will go to training and the rest to test. The seed parameter is used to indicate a pseudo-randomization pattern so that, each time the code is executed, different results are obtained.

```
val splits = data.randomSplit (Array(0.6, 0.4), seed = 1234L)
val train = splits (0)
val test = splits (1)
```

Here the fields for the neural network are specified, which are the input fields, output and intermediate ones, so they are declared in this way, with the help of a vector.

```
val layers = Array[Int] (4, 5, 4, 3)
```

Here you create the MultilayerPerceptronClassifier object, and with the help of different methods you specify the characteristics you want it to have. The layers declared in the previous step are passed as a parameter with the help of the setLayers method, with setBlockSize the number of bits to be used is defined, setSeed to grant randomness and finally setMaxIter is found, which is the maximum number of iterations. To make.

```
val trainer = new MultilayerPerceptronClassifier() .setLayers (layers)
.setBlockSize (128) .setSeed (1234L) .setMaxIter (100)
```

Here the model is adjusted to the data, it is trained to give better results for the subsequent prediction, what is saved in the trainer variable It is what will be executed on the dataset that is passed as a parameter, which results in the MultilayerPerceptronClassifier model.

```
val model = trainer.fit (train)
```

Already being the final part of the process, the data set saved for the test is passed as a parameter in the transform method, these will be executed on the model and will return the result obtained after going through the MultilayerPerceptronClassifier process. Once the data has been saved in the result variable, it is accessed with the help of the select method, specifying the columns to be displayed. In the evaluator variable, the percentage of precision that the classification method had will be saved.

```
val result = model.transform (test)
val predictionAndLabels = result.select ("prediction", "label")
val evaluator = new MulticlassClassificationEvaluator() .setMetricName
("accuracy")
```

At the end, the precision obtained by the MultilayerPerceptron classification process is displayed.

```
println (s "Test set accuracy = $ {evaluator.evaluate
(predictionAndLabels)}")
```

Result:

```
Test set accuracy = 0.975
```