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 2

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

Nombre del docente:
José Christian Romero Hernández

Practice 2

In order to create the linear regression object, the respective library must first be imported, for this is where said library is imported.

```
import org.apache.spark.ml.regression.LinearRegression
```

The following library is called Log4j and it is used to write log messages, whose purpose is to record a certain action carried out during the execution of the code. The next line shows the registry declaration without doing any configuration.

```
import org.apache.log4j._
Logger.getLogger ("org"). SetLevel (Level.ERROR)
```

The library is imported to be able to establish a session in spark, and then it is declared saving in the variable named spark.

```
import org.apache.spark.sql.SparkSession
var spark = SparkSession.builder (). GetOrCreate ()
```

After the initial previous configuration, the data to be used is loaded, this is done with the read method, adding more parameters through the option method, specifying that the headers are read, the type of format that will be csv and the path where the data file will be read.

```
val data = spark.read.option ("header","true") .option
("inferSchema","true") .format ("csv") .load ("Clean-Ecommerce.csv")
```

The schema allows to view the type of data that each column of the CSV has, and what is done next is to print the first line of the dataset, as well as the name of its columns, going through it with the help of a for loop.

```
data.printSchema ()
val colnames = data.columns
val firstrow = data.head (1) (0)
println ("\ n")
println ("Example Data Row")
for(ind <- Range(1, colnames.length )) {
  println (colnames (ind))
  println (firstrow (ind))
  println ("\ n")
}
```

Result:

```
data: org.apache.spark.sql.DataFrame = [Email: string, Avatar: string ... 5
more fields]
root
 | - Email: string (nullable = true)
 | - Avatar: string (nullable = true)
 | - Avg Session Length: double (nullable = true)
 | - Time on App: double (nullable = true)
 | - Time on Website: double (nullable = true)
 | - Length of Membership: double (nullable = true)
 | - Yearly Amount Spent: double (nullable = true)

colnames: Array[String] = Array(Email, Avatar, Avg Session Length, Time on
App, Time on Website, Length of Membership, Yearly Amount Spent)
firstrow:                      org.apache.spark.sql.Row                      =
[mstephenson@fernandez.com,Violet,34.49726772511229,12.65565114916675,39.57766
801952616,4.0826206329529615,587.9510539684005]

Example Data Row
Avatar
Violet

Avg Session Length
34.49726772511229

Time on App
12.65565114916675

Time on Website
39.57766801952616

Length of Membership
4.0826206329529615

Yearly Amount Spent
587.9510539684005
```

In order to create a VectorAssembler object, the library must first be imported, so the libraries are imported here to be able to perform this action.

```
import org.apache.spark.ml.feature.VectorAssembler
import org.apache.spark.ml.linalg.Vectors
```

With the help of the select method, it is specified that the column Yearly Amount Spent will be taken and renamed by the name label, also including the rest of the columns and saving it in a variable called df.

```
val df = data.select (data ("Yearly Amount Spent") .as ("label"),
$"Avg Session Length", $"Time on App", $"Time on Website", $"Length of
Membership")
```

A new VectorAssembler object is declared, converting all the columns of the dataset to a single output column called features, all of this will be saved in the assembler variable. The newly created variable is used in conjunction with the transform method, in this the set of data saved in df is passed as a parameter to transform it to have two columns, which are label and features.

```
val assembler = new VectorAssembler() .setInputCols (Array("Avg Session
Length","Time on App","Time on Website","Length of Membership")).
SetOutputCol ("features")

val output = assembler.transform (df). select ($"label", $"features")
```

What has been done so far is the cleaning and transformation of the data, now it is time to create the LinearRegression object, and this is saved in the lr variable.

```
val lr = new LinearRegression()
```

Once the LinearRegression object has been created, the model must be adjusted to the data contained in the dataset stored in the output variable. This is where the diagonal is generated, which would be what would be seen graphically in the linear regression.

```
val lrModel = lr.fit (output)
```

With the help of println, the coefficients contained in the variable lrModel are printed.

```
println (s "Coefficients: $ {lrModel.coefficients} Intercept: $
{lrModel.intercept}")
```

Here is the summary of the trained model, using the variable lrModel, this with the help of the summary method.

```
val trainingSummary = lrModel.summary
```

Finally, the results obtained using the linear regression method are shown. The residuals is the difference between the predicted value and what is contained in the current label column, the RMSE is the distance between the points and the r2 is how variant the model is

```
trainingSummary.residuals.show()
println (s "RMSE: $ {trainingSummary.rootMeanSquaredError}")
println (s "MSE: $ {trainingSummary.meanSquaredError}")
println (s "r2: $ {trainingSummary.r2}")
```

Result:

```
Coefficients:
[25.73427108467073638.734271084670716,,38.73427108467071638.734271084670716,,3
8.73427108467073638.709355864,61.57732375487594] Intercept:
-1051.5942552990748
trainingSummary:
org.apache.spark.ml.regression.LinearRegressionTrainingSummary =
org.apache.spark.ml.regression.LinearRegressionTrainingSummary@39dc0a19
+ ------------------- +
|          residuals|
+ ------------------- +
| -6.788234090018818|
| 11.841128565326073|
| -17.65262700858966|
| 11.454889631178617|
| 7.7833824373080915|
| -1.8347332184773677|
|  4.620232401352382|
| -8.526545950978175|
| 11.012210896516763|
| -13.828032682158891|
| -16.04456458615175|
|  8.786634365463442|
| 10.425717191807507|
| 12.161293785003522|
|  9.989313714461446|
| 10.626662732649379|
|  20.15641408428496|
| -3.7708446586326545|
| -4.129505481591934|
|  9.206694655890487|
+ ------------------- +
only showing top twenty rows

RMSE: 9.923256785022229
MSE: 98.47102522148971
r2: 0.9843155370226727
```