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

Materia y serie:
Datos Masivos BDD-1704TI9A

Unidad a evaluar: Unidad III

Nombre de la Tarea:
Práctica Evaluatoria - Unidad 3

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

Nombre del docente:
José Christian Romero Hernández

Develop the following instructions in Spark with the Scala programming language .

The goal of this practical test is to try to group customers from specific regions of a wholesaler. This based on the sales of some product categories.

1. Import a simple Spark session

```
import org.apache.spark.sql.SparkSession
```

2. Use the lines of code to minimize errors
The necessary library is imported to later use the line of code that helps to minimize errors in the code.

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

3. Create an instance of the session Spark

```
val spark = SparkSession.builder (). getOrCreate ()
```

4. Import the library from Kmeans for the clustering algorithm

```
import org.apache.spark.ml.clustering.KMeans
```

5. Load the Wholesale Customers Data dataset
It is specified, through the option method, that the first line is taken as the header of the columns, and the place where the data file is located is put in load.

```
val dataset = spark.read.option ("header","true") .option
("inferSchema","true") .format ("csv") .load ("Wholesale customers
data.csv")
```

6. Select the following columns: Fresh, Milk, Grocery, Frozen, Detergents_Paper, Delicassen and call this set feature_data We
select the data we require from the dataset so we use the select function and specifying the fields we want a new dataset called "feature_data" is created.

```
val feature_data = (dataset.select ($"Fresh", $"Milk", $"Grocery",
$"Frozen", $"Detergents_Paper", $"Delicassen"))
```

7. Import Vector Assembler and Vector

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

8. Create a new Vector Assembler object for the feature columns as an input set, remembering that there are no labels.
We create a new “assembler” variable using the “VectorAssembler” library in order to create a vector that contains all the data from the other fields.

```
val assembler = (new VectorAssembler() .setInputCols (Array("Fresh",
"Milk", "Grocery", "Frozen", "Detergents_Paper", "Delicassen")).
SetOutputCol ("features"))
```

9. Use the assembler object to transform feature_data
Once having the “assembler” variable, the transformation of the data of the “features_data” dataset is carried out, which is stored in the training_data variable where only the new “features” field is selected, this dataset will be used to define the k means model .

```
val training_data = assembler.transform (feature_data) .select
($"features")
```

10.Create a Kmeans model with K = 3
A new instance of the K Means library is declared, specifying the number of clusters to create (k) that represents the number of possible groups to find and the random seed for cluster initialization.

```
val kmeans = new KMeans() .setK (3) .setSeed (1L)
```

10.1 Fit that model to training_data
We create the kmeans model by grabbing the fit using the dataset “training_data”

```
val model = kmeans.fit (training_data)
```

11.Evaluate the groups using Within Set Sum of Squared Errors WSSSE and print the centroids
We show the sum of squares of the residuals with a result of 8,095 from the “Training_data” dataset. The sum of the squares of the residual error is the variation attributed to the error. By comparing the sum of the squares of the model with the total sum of the squares, the proportion of the total variation is determined, the larger this value, the better the relationship for each centroid.

```
val WSSSE = model.computeCost (training_data)
println (s "Within Set Sum of Squared Errors = $ WSSSE")

WSSSE: Double = 8.095172370767671E10
Within Set Sum of squared Errors = 8.095172370767671E10
```

11.1 Show the result

Finally we have the “Cluster Centers” which are the arithmetic mean of all the points that belong to the group where each point is closer to its own group center than to other group centers.

```
println ("Cluster Centers:")
model.clusterCenters.foreach (println)

Cluster Centers:
[7993.574780058651,4196.803519061584,5837.4926686217,2546.624633431085,2
016.2873900293255,1151.4193548387098]
[9928.18918918919,21513.081081081084,30993.486486486487,2960.43243243243
25,13996.594594594595,3772.3243243243246]
[35273.854838709674,5213.919354838709,5826.096774193548,6027.66129032258
05,1006.9193548387096,2237.6290322580644]
```

YouTube Link: https: // www. youtube.com/watch?v=pc0oNentXYc