



EDUCACIÓN
SECRETARÍA DE EDUCACIÓN PÚBLICA



TECNOLÓGICO
NACIONAL DE MÉXICO®

TECNOLÓGICO NACIONAL DE MÉXICO

INSTITUTO TECNOLÓGICO DE TIJUANA

SUBDIRECCIÓN ACADÉMICA

DEPARTAMENTO DE SISTEMAS Y COMPUTACIÓN

NOMBRE DE LOS ALUMNOS:

GALAVIZ LONA OSCAR EDUARDO (N.CONTROL: 17212993)

MARQUEZ MILLAN SEASHELL VANESSA (N.CONTROL:)

Carrera: Ingeniería Informática

Semestre: 9no

MATERIA: Datos Masivos

PROFESOR: JOSE CHRISTIAN ROMERO HERNANDEZ

Practice evaluatoria 3

Unidad 3

Development

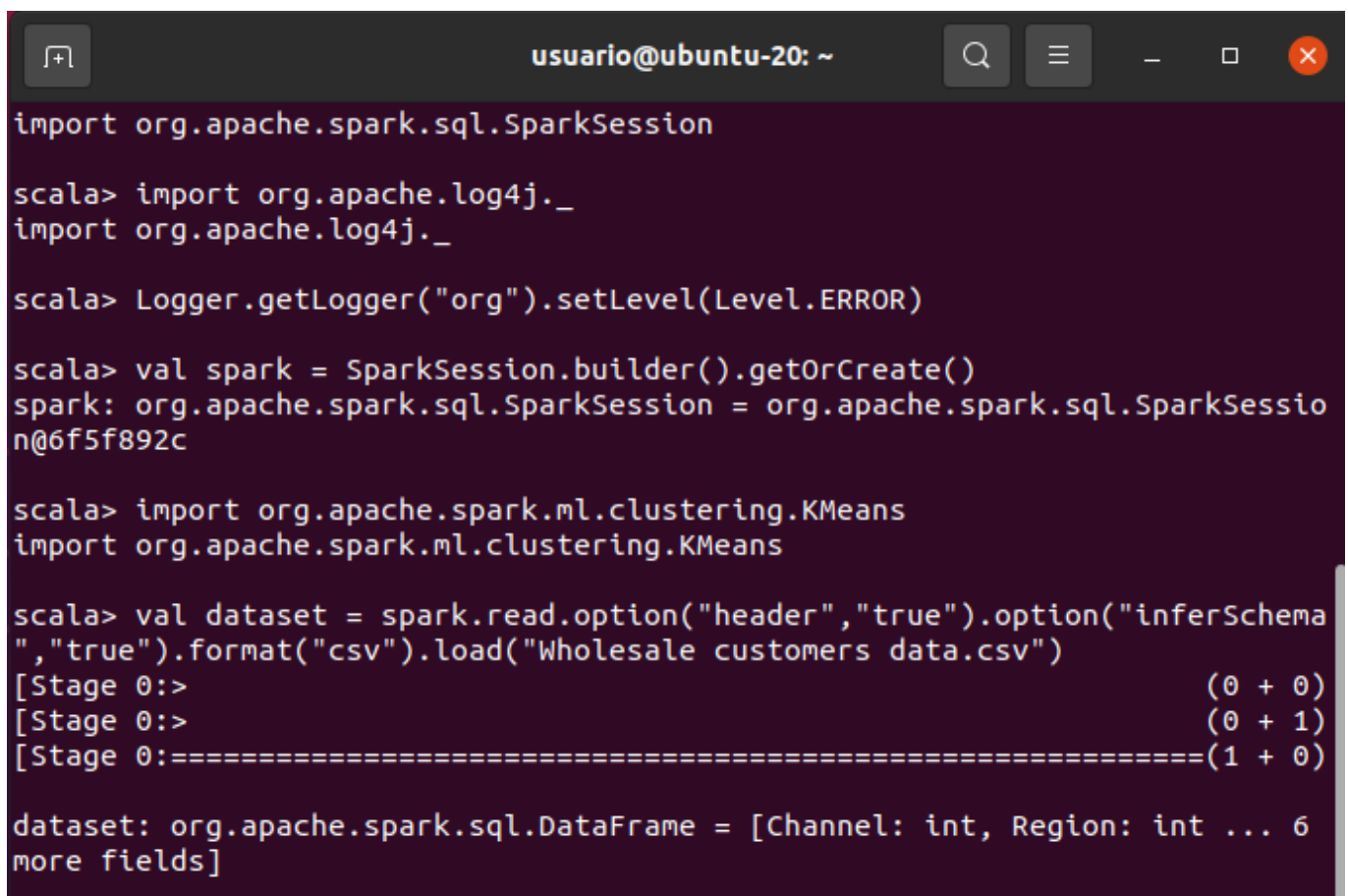
```
// 1-. Importar una simple sesión Spark.
import org.apache.spark.sql.SparkSession

// 2-. Utilice las líneas de código para minimizar errores
import org.apache.log4j._
Logger.getLogger("org").setLevel(Level.ERROR)

// 3-. Cree una instancia de la sesión Spark
val spark = SparkSession.builder().getOrCreate()

// 4-. Importar la librería de Kmeans para el algoritmo de agrupamiento.
import org.apache.spark.ml.clustering.KMeans

// 5-. Carga el dataset de Wholesale Customers Data
val dataset =
spark.read.option("header","true").option("inferSchema","true").format("csv").load
("Wholesale customers data.csv")
```



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

scala> import org.apache.log4j._
import org.apache.log4j._

scala> Logger.getLogger("org").setLevel(Level.ERROR)

scala> val spark = SparkSession.builder().getOrCreate()
spark: org.apache.spark.sql.SparkSession = org.apache.spark.sql.SparkSession@6f5f892c

scala> import org.apache.spark.ml.clustering.KMeans
import org.apache.spark.ml.clustering.KMeans

scala> val dataset = spark.read.option("header","true").option("inferSchema",
"true").format("csv").load("Wholesale customers data.csv")
[Stage 0:> (0 + 0)
[Stage 0:> (0 + 1)
[Stage 0:===== (1 + 0)

dataset: org.apache.spark.sql.DataFrame = [Channel: int, Region: int ... 6
more fields]
```

```
// 6-. Seleccione las siguientes columnas: Fresh, Milk, Grocery, Frozen,
Detergents_Paper, Delicassen y llamar a este conjunto feature_data
val feature_data = (dataset.select($"Fresh", $"Milk", $"Grocery", $"Frozen",
$"Detergents_Paper", $"Delicassen"))
```

```
scala> val feature_data = (dataset.select($"Fresh", $"Milk", $"Grocery", $"Frozen", $"Detergents_Paper", $"Delicassen"))
feature_data: org.apache.spark.sql.DataFrame = [Fresh: int, Milk: int ... 4 more fields]
```

```
// 7-. Importar Vector Assembler y Vector
import org.apache.spark.ml.feature.VectorAssembler
import org.apache.spark.ml.linalg.Vectors

//8-.Crea un nuevo objeto Vector Assembler para las columnas de características como un conjunto de entrada, recordando que no hay etiquetas
val assembler = new
VectorAssembler().setInputCols(Array("Fresh","Milk","Grocery","Frozen","Detergents_Paper","Delicassen")).setOutputCol("features")
```

```
scala> val assembler = new VectorAssembler().setInputCols(Array("Fresh","Milk","Grocery","Frozen","Detergents_Paper","Delicassen")).setOutputCol("features")
assembler: org.apache.spark.ml.feature.VectorAssembler = vecAssembler_a07f838b94b9
```

```
//9-.Utilice el objeto assembler para transformar feature_data
val features = assembler.transform(feature_data)

//10-.Crear un modelo Kmeans con K=3
val kmeans = new KMeans().setK(3).setSeed(1L)
val model = kmeans.fit(features)
```

```
scala> val model = kmeans.fit(features)
[Stage 2:> (0 + 1)

21/12/08 22:50:06 WARN BLAS: Failed to load implementation from: com.github
.fommil.netlib.NativeSystemBLAS
21/12/08 22:50:06 WARN BLAS: Failed to load implementation from: com.github
.fommil.netlib.NativeRefBLAS
[Stage 32:> (0 + 0)
[Stage 32:> (0 + 1)
[Stage 33:=====> (29 + 3) /
[Stage 33:=====> (39 + 2) /
[Stage 33:=====> (49 + 2) /
[Stage 33:=====> (55 + 2) /
[Stage 33:=====> (67 + 2) /
[Stage 33:=====> (77 + 2) /
[Stage 33:=====> (87 + 2) /
[Stage 33:=====> (99 + 2) /
[Stage 33:=====> (111 + 2) /
[Stage 33:=====> (120 + 2) /
[Stage 33:=====> (135 + 2) /
[Stage 33:=====> (146 + 2) /
[Stage 33:=====> (160 + 2) /
[Stage 33:=====> (174 + 2) /
[Stage 33:=====> (188 + 2) /
[Stage 33:=====> (197 + 2) /

model: org.apache.spark.ml.clustering.KMeansModel = kmeans_33fde5f9d5cc
```

//11-.Evalúe los grupos utilizando Within Set Sum of Squared Errors WSSSE e imprima los centroides.

```
val WSSSE = model.computeCost(features)
println(s"Within set sum of Squared Errors = $WSSSE")
```

```
scala> val WSSSE = model.computeCost(features)
warning: there was one deprecation warning; re-run with -deprecation for de
tails
WSSSE: Double = 8.095172370767671E10

scala> println(s"Within set sum of Squared Errors = $WSSSE")
Within set sum of Squared Errors = 8.095172370767671E10
```

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

```
scala> println("Cluster Centers: ")
Cluster Centers:

scala> model.clusterCenters.foreach(println)
[7993.574780058651,4196.803519061584,5837.4926686217,2546.624633431085,2016
.2873900293255,1151.4193548387098]
[9928.18918918919,21513.081081081084,30993.486486486487,2960.4324324324325,
13996.594594594595,3772.3243243243246]
[35273.854838709674,5213.919354838709,5826.096774193548,6027.6612903225805,
1006.9193548387096,2237.6290322580644]
```