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

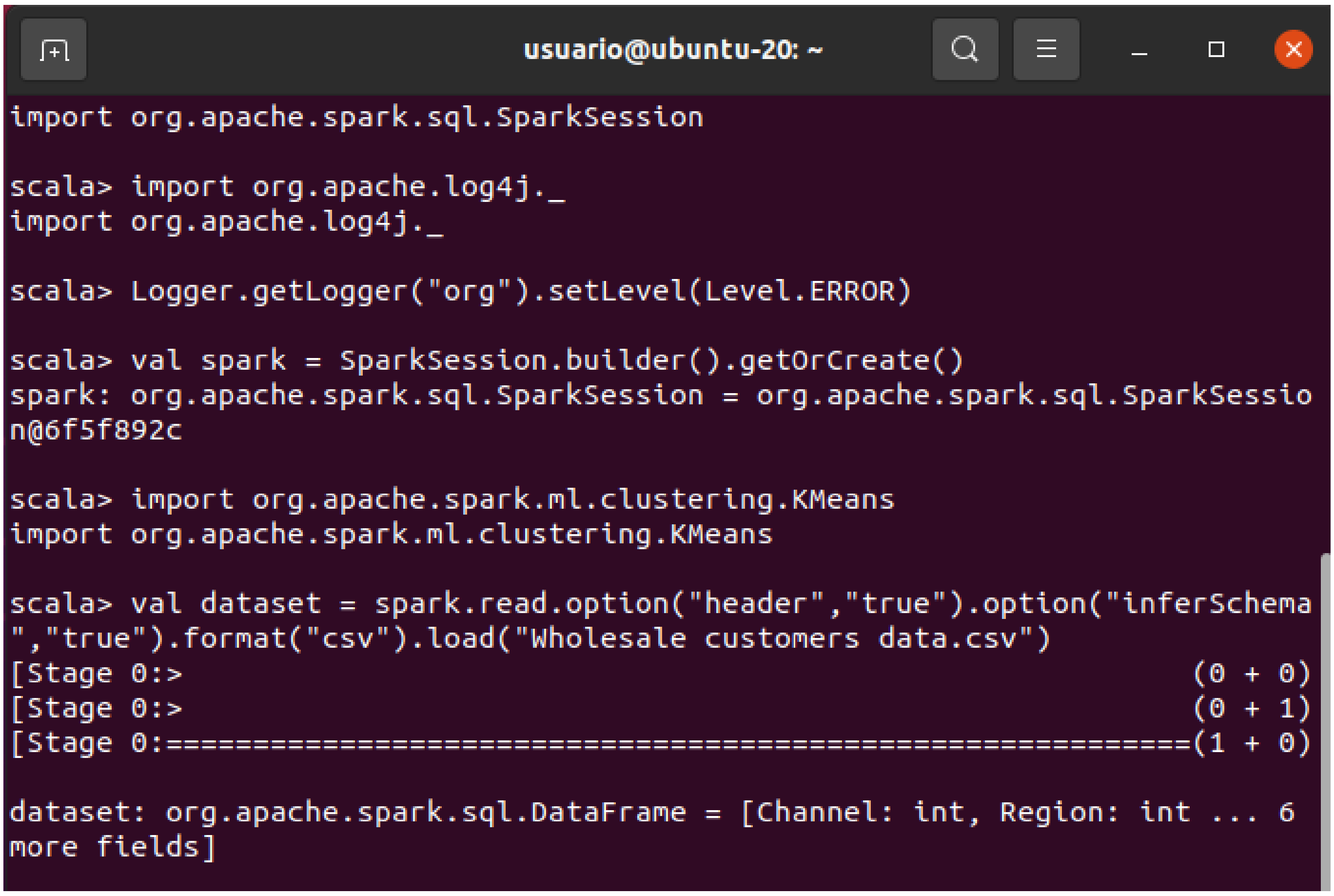
Unidad 3

Introduction

This proyect is a form to can manipulare the data frame, well the data and can get results and to do comparations and can to do probability in a future, we can see the information on orderly manner so important for can see best the results and the easily way.

The first part is import the library you need in this case is soark session for the sesion spark like befoore we see, aftter log for to do more small the wrong, and import the library for kmeans for the model, the last thing is load the dataset.

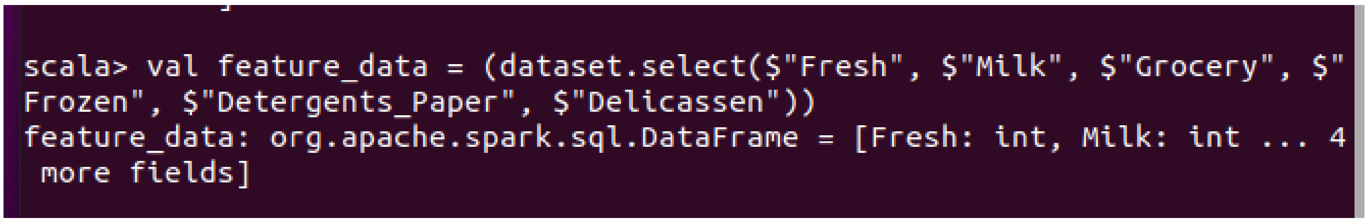
```
// 1-. Importar una simple sesión Spark.
import org.apache.spark.sql.SparkSession

// 2-. Utilice las lineas 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.SparkSessio
n@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]
```

For these practice we need to select some columuns in spesific

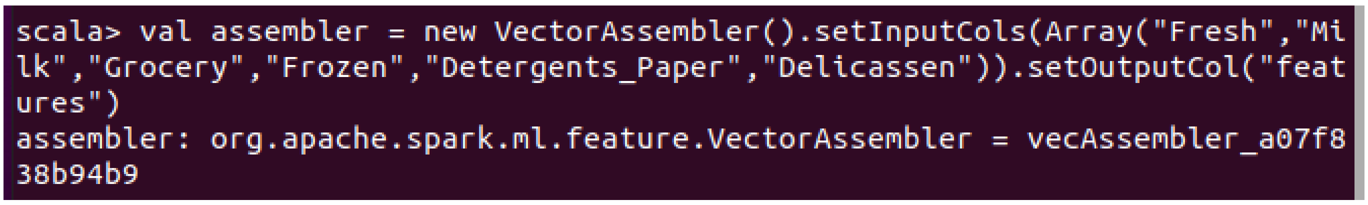
```
// 6-. Seleccione las siguientes columnas: Fresh, Milk, Grocery, Frozen,
Detergents_Paper, Delicassen y llamar a este conjunto feature_data
val feature_data = (dataset.select($"Fresh", $"Milk", $"Grocery",
$"Frozen", $"Detergents_Paper", $"Delicassen"))
```



```
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]
```

Well here only import the library for to manipulate vectors and assembly, and used them

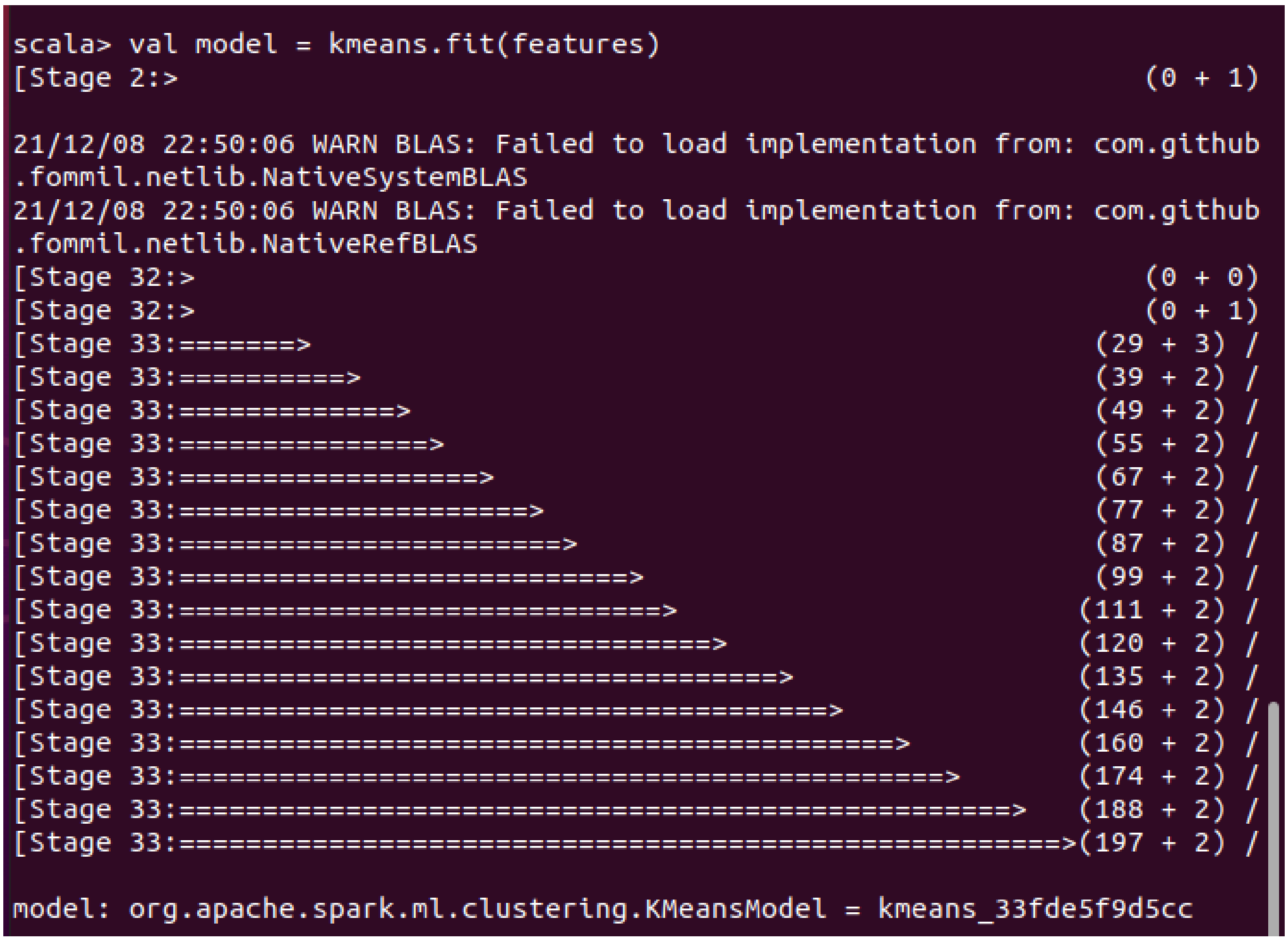
```
// 7-. Importar Vector Assembler y Vector
import org.apache.spark.ml.feature.VectorAssembler
import org.apache.spark.ml.linalg.Vectors

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



```
scala> val assembler = new VectorAssembler().setInputCols(Array("Fresh","Mi
lk","Grocery","Frozen","Detergents_Paper","Delicassen")).setOutputCol("feat
ures")
assembler: org.apache.spark.ml.feature.VectorAssembler = vecAssembler_a07f8
38b94b9
```

Withe the object create we go to transform, used transform and save in features, the other thing is used the model kmeans

```
//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)
```

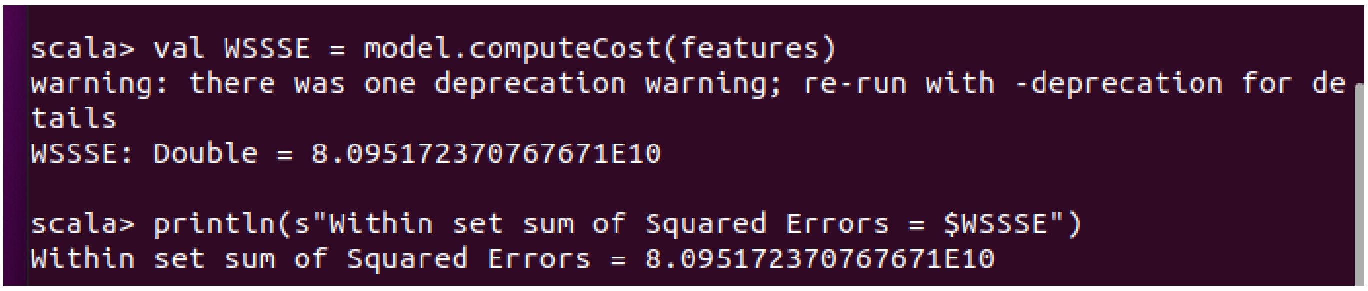


```
scala> val model = kmeans.fit(features)
[Stage 2:>                                                          (0 + 1)

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
```

Here only need to evalute the model and print the results

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

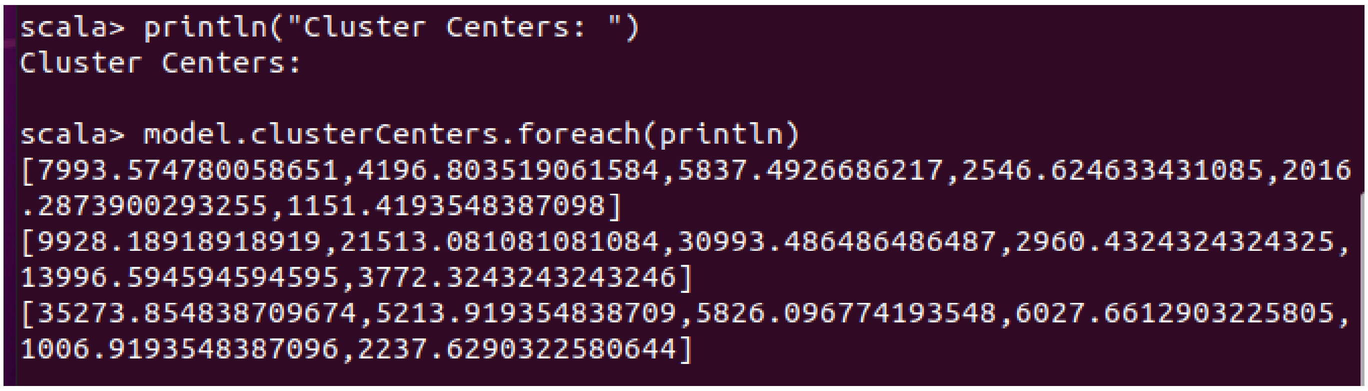
```
val WSSSE = model.computeCost(features)
println(s"Within set sum of Squared Errors = $WSSSE")
```



```
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
```

And the last thing is only print the centers

```
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]
```

Conclusion

After to create the model and work with the data, is clear the things than plain sight we cant se all information more clear withe thigs than really important you, and to do comparate withe the other results.

When you to do this type of analics you can learn more about the data, you can see the information, because is different the data and the result, and if you made this model is more easy to read the important information.