Big Data Project

LoudAcre Mobile Kmeans Clustering

```
import org.apache.spark.mllib.linalg.Vectors import
org.apache.spark.mllib.clustering.KMeans import
org.apache.spark.sql.functions._
```

1. Loading the Dataset

```
val filename = "C:/spark/loudacre_BigData/loudacre_Dataset" val
loudacre_data = sc.textFile(filename)
```

- Create a new dataframe having only latitude and longitude

```
val loudacre_data_split = loudacre_data.map(x => x.split(",")).map(s => (s(3),s(4))).toDF
loudacre_data_split.show(10)
```

```
+-------------+--------------+
|           _1|            _2|
+-------------+--------------+
|33.6894754264|-117.543308253|
|37.4321088904|-121.485029632|
|39.4378908349|-120.938978486|
|39.3635186767|-119.400334708|
|33.1913581092|-116.448242643|
|33.8343543748|-117.330000857|
|37.3803954321|-121.840756755|
|34.1841062345|  -117.9435329|
|32.2850556785|-111.819583734|
|45.2400522984|-122.377467861|
+-------------+--------------+
only showing top 10 rows
```

2. Cleaning

- **Many of the values are zero in both the columns, customer might have not provided their information about their location. we will remove these rows.**

```
loudacre_data_split.filter($"_1"==="0").show(10)
```

```
+---+---+
| _1| _2|
+---+---+
|  0|  0|
|  0|  0|
|  0|  0|
|  0|  0|
|  0|  0|
|  0|  0|
|  0|  0|
|  0|  0|
|  0|  0|
|  0|  0|
+---+---+
only showing top 10 rows
```

loudacre_data_split.filter($"_1" === "0").count val

la_df = loudacre_data_split.filter($"_1" =!= "0")

la_df.filter($"_1" === "0").count

- **Convert dataframe back to RDD**

val la_rdd = la_df.rdd.map(row => List(row.getString(0),row.getString(1)))

- **Create a vector RDD**

val vectors = la_rdd.map(s => Vectors.dense(s(0).toDouble,s(1).toDouble)).cache()

3. Train the model

val numClusters = 3 val numIterations = 20 val kmeansmodel =

KMeans.train(vectors,numClusters,numIterations)

- **Display the center points of each cluster**

kmeansmodel.clusterCenters.foreach(println)

```
[34.54436372366799,-118.05513586535038]
[39.92266331983432,-121.38464957351843]
[35.08592000544937,-112.57643826547802]
```

- **Create a broadcast variable for the kmeans model**

  kmeansmodel.computeCost(vectors)

```
val kmeansModel_BC = sc.broadcast(kmeansmodel)

val cluster_df = kmeansModel_BC.value.predict(vectors).toDF

cluster_df.show
```

```
+-----+
|value|
+-----+
|    0|
|    1|
|    1|
|    1|
|    0|
|    0|
|    1|
|    0|
|    2|
|    1|
|    1|
|    1|
|    0|
|    0|
|    2|
|    2|
|    1|
|    0|
|    1|
|    2|
+-----+
only showing top 20 rows
```

- **Create a new column called "id", which will be used for joining the two dataframes**

```
val df1 = cluster_df.withColumn("id",monotonically_increasing_id())

 val df2 = la_df.withColumn("id",monotonically_increasing_id())

df1.show(5)
```

```
+-----+---+
|value| id|
+-----+---+
|    0|  0|
|    1|  1|
|    1|  2|
|    1|  3|
|    0|  4|
+-----+---+
only showing top 5 rows
```

```
df2.show(5)
```

```
+-------------+--------------+---+
|           _1|            _2| id|
+-------------+--------------+---+
|33.6894754264|-117.543308253|  0|
|37.4321088904|-121.485029632|  1|
|39.4378908349|-120.938978486|  2|
|39.3635186767|-119.400334708|  3|
|33.1913581092|-116.448242643|  4|
+-------------+--------------+---+
only showing top 5 rows
```

- **Join the two dataframes based on “id” column**

val df3 = df2.join(df1,"id")

df3.show

```
+----+-------------+--------------+-----+
|  id|           _1|            _2|value|
+----+-------------+--------------+-----+
|  26|39.4708861702|-119.659926097|    1|
|  29|33.8514465953|-117.787423338|    0|
| 474|39.5548840745|-121.020788448|    1|
| 964|42.8588445306|-122.490145742|    1|
|1677|42.0521070411|-123.772498927|    1|
|1697| 34.206121509|-118.248366666|    0|
|1806|36.5039511635|  -121.0370419|    1|
|1950|34.1840832076| -118.13214751|    0|
|2040|39.4458400243|-119.337989148|    1|
|2214|37.9311384736|-121.511788497|    1|
|2250|34.1648878746| -117.67549234|    0|
|2453|36.1947181592|-115.053517304|    2|
|2509|38.3183374986|-120.975396248|    1|
|2529|38.1666641293|-121.845985179|    1|
|2927|36.3195030742|-120.722388463|    0|
|3091| 37.801310103| -122.29490043|    1|
|3506|38.3391887988|-122.403638455|    1|
|3764|33.5908087287|-109.124901699|    2|
|4590|37.1075594103| -121.72570178|    1|
|4823|34.0175895081| -111.98383465|    2|
+----+-------------+--------------+-----+
only showing top 20 rows
```

- **The number of point for each cluster in the dataset**

df3.groupBy("value").count().show()

```
+-----+------+
|value| count|
+-----+------+
|    1|177812|
|    2| 65686|
|    0|188359|
+-----+------+
```

- **We remove the “id” column and name the other three columns**

```
val newNames = Seq("latitude","longitude","cluster") val

finaldf = df3.drop("id").toDF(newNames:_*) finaldf.show
```

```
+--------------+---------------+-------+
|      latitude|      longitude|cluster|
+--------------+---------------+-------+
|39.4708861702|-119.659926097|       1|
|33.8514465953|-117.787423338|       0|
|39.5548840745|-121.020788448|       1|
|42.8588445306|-122.490145742|       1|
|42.0521070411|-123.772498927|       1|
| 34.206121509|-118.248366666|       0|
|36.5039511635|   -121.0370419|       1|
|34.1840832076| -118.13214751|       0|
|39.4458400243|-119.337989148|       1|
|37.9311384736|-121.511788497|       1|
|34.1648878746| -117.67549234|       0|
|36.1947181592|-115.053517304|       2|
|38.3183374986|-120.975396248|       1|
|38.1666641293|-121.845985179|       1|
|36.3195030742|-120.722388463|       0|
| 37.801310103| -122.29490043|       1|
|38.3391887988|-122.403638455|       1|
|33.5908087287|-109.124901699|       2|
|37.1075594103| -121.72570178|       1|
|34.0175895081| -111.98383465|       2|
+--------------+---------------+-------+
only showing top 20 rows
```

• Display the points for each cluster

```
finaldf.filter($"cluster" === "0").show
```

```
+--------------+---------------+-------+
|      latitude|      longitude|cluster|
+--------------+---------------+-------+
|33.8514465953|-117.787423338|       0|
| 34.206121509|-118.248366666|       0|
|34.1840832076| -118.13214751|       0|
|34.1648878746| -117.67549234|       0|
|36.3195030742|-120.722388463|       0|
|34.2298811925|-117.877102556|       0|
|34.0669923073|-118.192176947|       0|
|33.0262077564|-116.812861708|       0|
|34.1521333194|-116.638723297|       0|
|32.9596188021|-116.805661125|       0|
|33.9249343611|-117.881397679|       0|
|33.0239999005|-116.674524717|       0|
|33.0339156527|-116.837511787|       0|
| 37.038360427|-119.588745345|       0|
|32.9965920016|-116.685453769|       0|
|35.6625077365|-120.285702621|       0|
|34.2486240248|   -118.2386197|       0|
|33.1863998188|-116.724525124|       0|
|32.9127253094|-116.307005506|       0|
|33.8233479881|-117.770690361|       0|
+--------------+---------------+-------+
only showing top 20 rows
```

finaldf.filter($"cluster" === "1").show

```
+-------------+--------------+-------+
|     latitude|     longitude|cluster|
+-------------+--------------+-------+
|39.4708861702|-119.659926097|      1|
|39.5548840745|-121.020788448|      1|
|42.8588445306|-122.490145742|      1|
|42.0521070411|-123.772498927|      1|
|36.5039511635|  -121.0370419|      1|
|39.4458400243|-119.337989148|      1|
|37.9311384736|-121.511788497|      1|
|38.3183374986|-120.975396248|      1|
|38.1666641293|-121.845985179|      1|
| 37.801310103| -122.29490043|      1|
|38.3391887988|-122.403638455|      1|
|37.1075594103| -121.72570178|      1|
|38.4745414188|-122.134143962|      1|
|37.6543322066|-121.588940213|      1|
|45.3409177163|-117.542333377|      1|
|45.1781132987|-117.661882259|      1|
|42.6990076523|-122.637552833|      1|
|38.7301629267|-121.408276173|      1|
|45.4371672468|-117.700782699|      1|
|37.5033221854|-121.525720878|      1|
+-------------+--------------+-------+
only showing top 20 rows
```

finaldf.filter($"cluster" === "2").show

```
+-------------+--------------+-------+
|     latitude|     longitude|cluster|
+-------------+--------------+-------+
|36.1947181592|-115.053517304|      2|
|33.5908087287|-109.124901699|      2|
|34.0175895081| -111.98383465|      2|
|39.6628949081| -114.56289137|      2|
|33.6230375795|-111.542615553|      2|
|35.0789202973|-111.408292531|      2|
|36.0838304774|-114.731425185|      2|
|36.2981261276|-113.613971565|      2|
|32.2734814339|-111.641629449|      2|
|33.5206745813|-112.003312523|      2|
| 36.661607919|-115.006223963|      2|
|33.3561047647|-111.484732167|      2|
|33.3954391971|-111.861572714|      2|
|32.2866036613|-111.791915607|      2|
|32.3938041755|-111.506443379|      2|
|33.6603116416|-111.513103129|      2|
|35.0455801515|-111.316399941|      2|
|33.7285608333|-111.825495401|      2|
|32.5350707596|-110.614014808|      2|
|33.6029085259|-111.304704398|      2|
+-------------+--------------+-------+
only showing top 20 rows
```

4. Conclusion

- **Succesfully divided the user's location data into three clusters with centers [34.544,-118.055],[39.922,-121.384],[35.085,-112.576]. This will further help the company improve their service by maximising the coverage for its users.**