5/7/2018

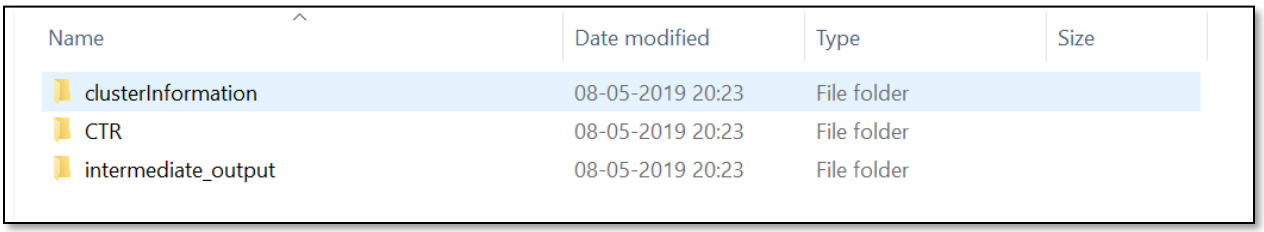
Saavn Analytics Project

Case Study on saavn data.

Name:Ruksana Bhanu

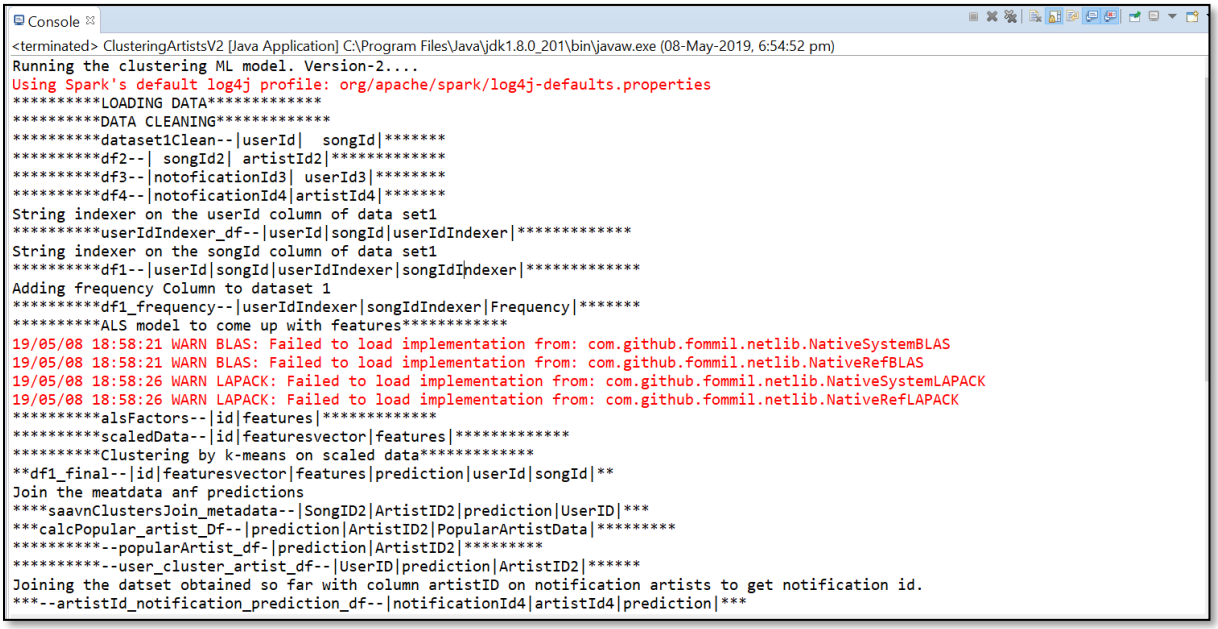
READ ME-PRE-REQUISITES:

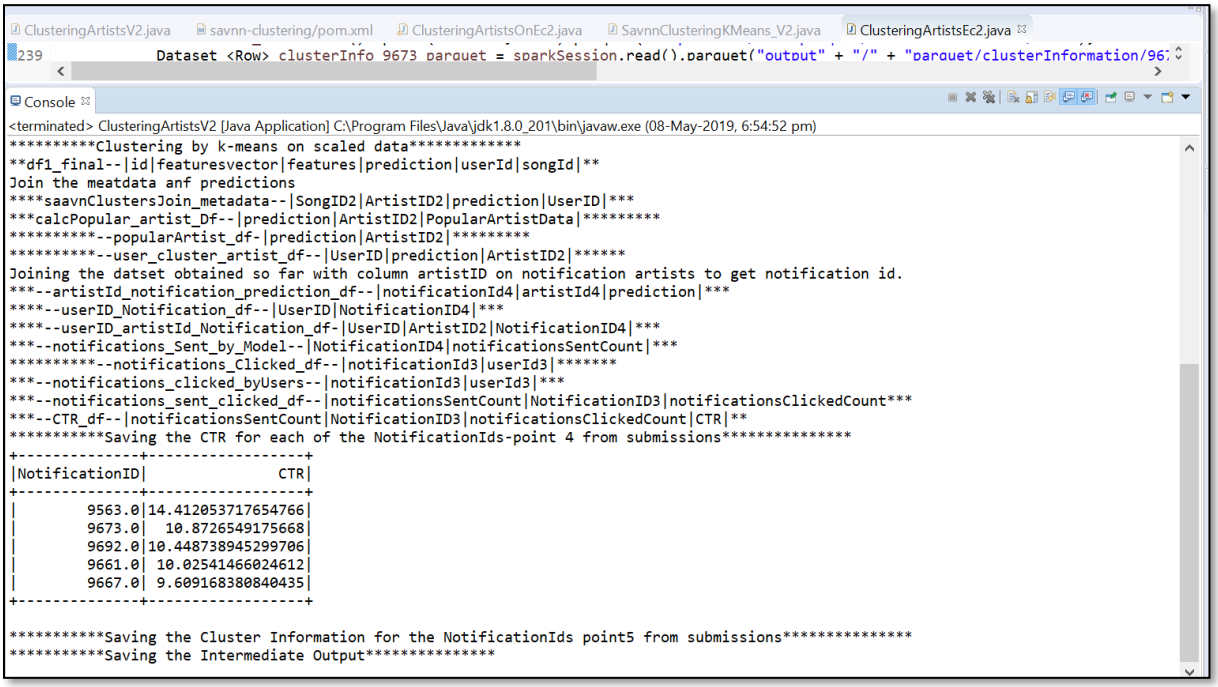
- Before executing the project, Please ensure the output directory doesn't exists. Otherwise, program will throw error. As part of submission, in the project zip, I have renamed the output folder to output1.
- Please check for the winutils configuration before executing the program.
- The Required csv files are generated by first writing on parquet file and then read from parquet and write to csv.
- CTR is given for the top 5 which has got maximum percentage. Also, cluster information is also given for those files only.
- This project is run locally not on ec2. I have downloaded the s3 files on local and accessed them from the local path (project path) and developed the model. Hence the model is run as a java application.
- **Before running the application, Please ensure the data files are present on the path specified in the program. Deleting them in zip file submission due to size issues and taking long time to upload on drive.**
- The files generated are in the folder csv. Folder names for the submissions are as follows
-



-
The files generated are in the folder csv. Folder names for the submissions are as follows
- CTR contains the csv asked in point 4 submission. CTR contains the csv asked in point 4 submission.
- Clusterinformation CTR contains the csv asked in point 5 submission of submission guidelines
- Intermediate output contains the csv for point 6 submission.

SCREEN SHOTS:





STEPS INVOLVED IN CALCULATING THE CTR

1. LOAD THE DATASET

Load all the 4 given data sets in to the program.

Eg : Dataset<Row> rawDataset = sparkSession.read().option("header", false).option("inferschema", true).csv("spark-warehouse/data/sample100mb.csv");

Output column of these data set is as below.

Data set-1:

```
|userId|  songId|
```

Data set-2:

```
| songId2| artistId2|
```

Data set-3:

```
|notoficationId3| userId3|
```

Data set-4:

```
|notoficationId4|artistId4|
```

2. SEPARATE NECESSARY COLUMNS

For each of the datasets, select / get the only required columns for the model building.

Eg : Dataset<Row> dataset1 = rawDataset.select(rawDataset.col("_c0").as("userId"), rawDataset.col("_c2").as("songId"));

3. IGNORE ROWS HAVING NULL VALUE

Dropping the rows which have null values. This is done in order to improve the efficiency of the model and to predict the clusters accurately.

Eg : Dataset<Row> dataset1Clean = dataset1.na().drop();

4. CONVERTING THE STRING BASED COLUMS TO NUMERIC

As the analytics models accepts the parameters as double or vector, we need to convert the string based columns such as userId and songid from the data set1. i.e, sample 100mb data.

Eg : StringIndexer Indexer= new StringIndexer().setInputCol("userId").setOutputCol("userIdIndexer");

Output of string indexer on above datasets looks below.

```
|userId|songId|userIdIndexer|songIdIndexer|
```

5. CALCULATE THE FREQUENCY

For the model to perform well, calculate the frequency by group on songid and userid and use this column to set on setRatingColumn

Eg : Dataset<Row> df1_frequency = df1.groupBy("userIdIndexer", "songIdIndexer").agg(functions.count("*").alias("Frequency"));

Output of string indexer on above datasets looks below.

```
|userIdIndexer|songIdIndexer|Frequency|
```

6. BUILD ALS MODEL

Using ALS model to come up with features by setting the usercol as userid, itemcol as songid and the rating col as frequency.

Eg : ALS als = new ALS().setRank(10).setMaxIter(5).setImplicitPrefs(true).setUserCol("userIdIndexer").setItemCol("songIdIndexer").setRatingCol("Frequency");

Output of ALS model will contain the column id and features.

|id|features|

7. UDF TO CONVERT THE FEATURES TO VECTOR

The Output from the above ALS model contains the features of type array. But for the K-means algorithm, the features must be of type vector. Hence a UDF is written to convert the array of floats to vector of double.

sparkSession.udf().register("udfConvertArrayToVector", udfConvertArrayToVector, new VectorUDT());

Dataset<Row> alsFactorsAsVector = sparkSession.sql("SELECT id,udfConvertArrayToVector(features) as featuresvector FROM SavnnFeatures");

Output of the dataset after applying the UDF is

|id|featuresvector|features|

8. SCALE THE VARIABLES

Scaling and normalizing the features so obtained from the above data set.

StandardScaler scaler = new StandardScaler().setInputCol("featuresvector").setOutputCol("features").setWithStd(true).setWithMean(true);Output after applying the k-means algorithm on the above data set looks as per below format.

|id|featuresvector|features|

9. CLUSTERING BY K-MEANS ALGORITHM

For clustering the given data set in order to send the notifications, I have used the k-means algorithm. After so many trial and error and my measuring the WSSE and silhouette, no of clusters is set as 300 for this entire data set.

KMeans kmeans = new KMeans().setK(300).setSeed(1L);

Output after applying the k-means algorithm on the above data set looks as per below format.

|id|featuresvector|features|prediction|userId|songId|

10. JOINING THE PREDICTION DATA SET WITH METADATA

Now, join the prediction data set with the metadata on the column songId

Dataset<Row> saavnClustersJoin_metadata = df2.join(df1_final,
df2.col("SongID2").equalTo(df1_final.col("SongID")),
"inner").drop(df1_final.col("SongID")).select("SongID2", "ArtistID2", "prediction", "UserID");

Output after joining these 2 data sets is as belows.

```
|SongID2|ArtistID2|prediction|UserID|
```

11. CALCULATE POPULAR ARTIST

Calculate the popular artist by partitioning on the prediction column

WindowSpec w =
org.apache.spark.sql.expressions.Window.partitionBy("prediction").orderBy(functions.desc("PopularArtistData"));

Dataset<Row> popularArtist_df= calcPopular_artist_Df.withColumn("rn",
row_number().over(w)).where("rn = 1")

.select(calcPopular_artist_Df.col("prediction"), calcPopular_artist_Df.col("ArtistID2"));Output after joining these 2 data sets is as belows.

```
|prediction|ArtistID2|
```

12. GET THE USER CLUSTER AND ARTIST ID

Join the available datasets in such a way that the data set gives the output of user id and its respective cluster and artistID

Dataset<Row> user_cluster_artist_df = df1_final.join(popularArtist_df,df1_final.col("prediction")

.equalTo(popularArtist_df.col("prediction")), "inner").drop(popularArtist_df.col("prediction"))

.select("UserID", "prediction","ArtistID2");.select(calcPopular_artist_Df.col("prediction"), calcPopular_artist_Df.col("ArtistID2"));

|prediction|ArtistID2|

13. JOIN THE DATASET OF NOTIFICATION ARTIST AND CLUSTER

Join the data set3 to get the notification id corresponding to the artistid and cluster. Join on the column artist ID .

Dataset<Row> userID_artistId_Notification_df = artistId_notification_prediction_df.join(saavnClustersJoin_metadata,

artistId_notification_prediction_df.col("artistId4").equalTo(saavnClustersJoin_metadata.col("ArtistID2"))

.and(artistId_notification_prediction_df.col("prediction").equalTo(saavnClustersJoin_metadata.col("prediction"))),

"inner").drop(artistId_notification_prediction_df.col("prediction"))

.drop(artistId_notification_prediction_df.col("ArtistID4"));

The outcome of above operation looks as per below format

|UserID|ArtistID2|NotificationID4|

14. CALCULATE THE NOTIFICATIOS SENT COUNT BY THE MODEL

To get the CTR, we need to calculate the notification sent by the model and the notifications clicked by the users.

CTR is calculated as per below

Numerator is = no of users who clicked notification n1.

This Can be obtained from the notifications data (dataset 3 &$).

Group by the notification ID and get count of users where notification id is 9553. Consider the count as 8

Denominator is = no of users to whom notification n1 (9553) is sent.

To arrive at this, get the artist ID of this notification id 9553 from datset3&4.

Consider that the artist id for this notification id 9553 is artist1234.

From the predicted dataset of our model which has clusters, group by the artist ID and get count of users where artist id is artist1234.

Below is the data frame to get the notifications sent count by the model for each of the notification Ids.

Dataset<Row> notifications_Sent_by_Model = userID_Notification_df.groupBy("NotificationID4").agg(functions.count("*").alias("notificationsSentCount"));

The outcome of above operation looks as per below format

```
|NotificationID4|notificationsSentCount|
```

15. CALCULATE THE NOTIFICATIONS CLICKED BY THE USERS

As per the logic defined above for the CTR calculation, notifications clicked count by the users for each of the notification ids is calculated.

```
Dataset<Row> notifications_Clicked_df = userID_Notification_df.join(df3,

userID_Notification_df.col("UserID").equalTo(df3.col("UserID3"))

.and(userID_Notification_df.col("NotificationID4").equalTo(df3.col("NotificationID3"))),"inner").drop(userID_Notification_df.col("UserID")).drop(userID_Notification_df.col("NotificationID4"));

Dataset<Row> notifications_Sent_by_Model = userID_Notification_df.groupBy("NotificationID4").agg(functions.count("*").alias("notificationsSentCount"));

The outcome of the above operation look as per below format

|notificationsSentCount|NotificationID3|
```

16. CALCULATE THE NOTIFICATIONS CLICKED BY THE USERS

As per the logic defined above for the CTR calculation, notifications clicked count by the users for each of the notification ids is calculated.

```
Dataset<Row> notifications_Clicked_df = userID_Notification_df.join(df3,
```

```
userID_Notification_df.col("UserID").equalTo(df3.col("UserID3"))

.and(userID_Notification_df.col("NotificationID4").equalTo(df3.col("NotificationI
D3"))),"inner").drop(userID_Notification_df.col("UserID")).drop(userID_Notificati
on_df.col("NotificationID4"));

Dataset<Row> notifications_clicked_byUsers =
notifications_Clicked_df.groupBy("NotificationID3").agg(functions.count("*").alia
s("notificationsClickedCount"));

The outcome of the above operation look as per below format

|notificationsSentCount|NotificationID3|notificationsClickedCount
```

17. CALCULATE THE CTR

Divide the columns notificationsentby model and the notification clicked by users from the above data set.

Dataset<Row> CTR_in_percentage = sparkSession.sql("SELECT NotificationID3, CTR*100 as CTR FROM CTR_df ");

The output gives the notification id and its respective CTR

```
|notificationId |CTR
```

Hence the top 5 CTR obtained from the model is as follows.

NotificationID	CTR
9563	14.41205
9673	10.87265
9692	10.44874
9661	10.02541
9667	9.609168