

1. I used the dataset directly from S3 and was able to successfully read all values from single as well as multiple files.
2. I used ALS model to extract features. For that I converted the user\_id and song\_id to numbers using StringIndexer.
3. I calculated the frequency of each song listened by a user and passed this frequency value as an argument to .setRatingCol in ALS.
4. After getting the output from ALS, the features were an array so I converted them to Vector using a UDF called "*toVector*".
5. This feature vector was then passed as an input to K-means model.
6. From this model, I obtained predictions. I randomly chose k=10 for my model hence I got 10 user clusters who like similar songs.
7. Using **IndexToString** I converted the numeric user\_id back to their String equivalents and store the final result in **finalPredictedData** dataset.
8. Next I joined this dataset with the original dataset which had user\_id and song\_id to fetch song\_id listened by each user\_id in a cluster and named this dataset **joinedUserWithMetaData**
9. The below screenshot shows the result of above step.

```
[ec2-user@ip-172-31-85-40 ~]$ spark2-submit --class Test.Clustering.App --master yarn --deploy-mode client --name test /home/ec2-user/Clustering-0.0.1-SNAPSHOT.jar
19/05/15 14:33:59 INFO hive.metastore: Trying to connect to metastore with URI thrift://ip-172-31-85-40.ec2.internal:9083
19/05/15 14:33:59 INFO hive.metastore: Opened a connection to metastore, current connections: 1
19/05/15 14:33:59 INFO hive.metastore: Connected to metastore.
19/05/15 14:38:02 WARN netlib.BLAS: Failed to load implementation from: com.github.fommil.netlib.NativeSystemBLAS
19/05/15 14:38:02 WARN netlib.BLAS: Failed to load implementation from: com.github.fommil.netlib.NativeRefBLAS
+-----+
|prediction|      user_id|      song_id|
+-----+
|0df0d31ffddfa7bb70...|QUNHFVVV|
|0df0d31ffddfa7bb70...|4RmbCw4|
|0df0d31ffddfa7bb70...|wpe4NbSF|
|0df0d31ffddfa7bb70...|tkxMCrvC|
|0df0d31ffddfa7bb70...|adfg102|
|0df0d31ffddfa7bb70...|j1w_tYRU|
|0df0d31ffddfa7bb70...|NvK1OEIR|
|0df0d31ffddfa7bb70...|MzuOp7da|
|0df0d31ffddfa7bb70...|hxb78FRH|
|0df0d31ffddfa7bb70...|VwFBDSer|
|0df0d31ffddfa7bb70...|v0ZDlgeM|
|0df0d31ffddfa7bb70...|rxg2WbDn|
|0df0d31ffddfa7bb70...|QW2M-Miy|
|0df0d31ffddfa7bb70...|mt_v41-o|
|0df0d31ffddfa7bb70...|idoLwJICu|
|0df0d31ffddfa7bb70...|FH9dQInl|
|0df0d31ffddfa7bb70...|cRw2N9Se|
|0df0d31ffddfa7bb70...|qRwkvWw|
|0df0d31ffddfa7bb70...|yaoIKlrO|
|0df0d31ffddfa7bb70...|pr4cSVVx|
+-----+
only showing top 20 rows
[ec2-user@ip-172-31-85-40 ~]$
```

10. After this, I wanted to join the **joinedUserWithMetaData** dataset with the actual metadata having corresponding artist\_id with each song but I repeatedly got exceptions and hence ultimately commented that part from my code.
11. My approach was to use the above joined dataset having user\_id, song\_id, artist\_id, prediction columns, partition by prediction and save each prediction file as a csv.

12. From each csv dataset, I would have tried to find out the artist\_id having max count and make that cluster correspond to that artist.

13. I could not reach upto the part of calculating CTR and notification clicks, etc. hence also not providing any details about my further approach.

**Command used to run jar file:**

```
spark2-submit --class Test.Clustering.App --master yarn --deploy-mode client --name test  
/home/ec2-user/Clustering-0.0.1-SNAPSHOT.jar
```