Readme

I have submitted below artifacts as per submission guidelines:

1. Jar file name is SaavnAnalyticsSparkML-0.0.1-SNAPSHOT.jar. Main program is **SaavnAnalyticsKMeans.java**.

2. SaavnAnalyticsSparkML.zip contains entire project.

3. 2017CBDE639_AmitGoel_C2BDE_ SaavnAnalyticsSparkML_Report.pdf is the entire project report.

4. Output csv files are also provided. I have produced CSV files in 5 different folders namely: **NotificationNumber1, UserClusterArtist, NotificationNumber2, CTRData and CTRSpecificNotificationData**. I have provided output zip file namely **saavnanalytics_kmeans_hdfs.zip** which contains above folders along with files which were produced as output of program. For **NotificationNumber1, UserClusterArtist and NotificationNumber2**, I could not save down data in a single file for each of these, through Spark program as it was going out of memory again and again and I tried a lot. So, I let program generate files in parts. But I have ensured that each part has header so evaluator won't have a problem in understanding the content in part files.

    a. **NotificationNumber1**
    This folder has csv files which contain UserID, ClusterID, ArtistID and NotificationID for specific notification ids as per problem statement before finding popular artist and new combined cluster.

    b. **UserClusterArtist**
    This folder has csv files which contain UserID, NewCommonClusterID and PopularArtistID. Popular artist in each cluster was found and if same artist was popular in multiple clusters then those clusters were combined and new common cluster was formed so we have 1 artist mapped to 1 cluster.

    c. **NotificationNumber2**
    This folder has a csv file which contains UserID, NewCommonClusterID, PopularArtistID and NotificationID for specific notification ids as per problem statement after finding popular artist and new combined cluster.

    d. **CTRData**
    This folder has a csv file which contains NotificationID and CTR for all the notification ids for which notifications were pushed by this program.

    e. **CTRSpecificNotificationData**
    This folder has a csv file which contains NotificationID and CTR for specific notifications ids as per problem statement.

5. Commands to execute jar, are shown next in this document.

6. Screenshots of successful runs are attached in this document at the end.

7. In order to maximize CTR, I had to increase number of clusters due to which K-Means algorithm started taking bit more time but I still managed whole program within reasonable amount of time for such massive data and getting good overlap. In-fact saving data to files was also quite time consuming. If I would have just saved data at the end only once then my program would have got over in just 30-35 minutes. That's how I had tested and tuned my algorithm. At the end, I added steps to save more data in files at different intervals which added more time in program execution.

Steps to execute jar file:

1. I have used Cloudera VM for this project in order to save cost associated with use of EC2 instance. But I have cross verified so that same jar gets executed on EC2 instance as well. It is just that someone will have to tune EC2 instance. I have tuned my Cloudera VM instance which is explained in the submitted pdf report. I have attached a screenshot at the end, just to show that same jar gets executed on EC2 instance well. **It takes lot of time in reading files from S3 on Cloudera VM so I had to download files and had copied onto Hadoop file system in Cloudera VM.**

2. Put jar file on Cloudera VM / EC2 instance.

3. Run below commands from the same directory where jar file is present:

   a. Command to run jar file when input files are present in Hadoop file system on Cloudera VM:

```
spark2-submit \
--class com.ml.upgrad.saavnanalytics.SaavnAnalyticsKMeans \
--master yarn \
--deploy-mode client \
--driver-memory 18g \
--executor-memory 18g \
--executor-cores 3 \
--num-executors 3 \
--conf "spark.executor.memoryOverhead=2048" \
--conf  "spark.executor.extraJavaOptions=-XX:+PrintGCDetails -XX:+PrintGCTimeStamps" \
SaavnAnalyticsSparkML-0.0.1-SNAPSHOT.jar \
/user/cloudera/checkpoint_dir \
/user/cloudera/saavnanalytics/activity/sample100mb.csv \
/user/cloudera/saavnanalytics/newmetadata/* \
/user/cloudera/saavnanalytics/notification_clicks/* \
/user/cloudera/saavnanalytics/notification_actor/* \
/user/cloudera/saavnanalytics_kmeans_hdfs
```

   b. Command to run jar file when input files are present in S3 and jar is being executed on EC2 instance:

```
spark2-submit \
--class com.ml.upgrad.saavnanalytics.SaavnAnalyticsKMeans \
--master yarn \
--deploy-mode client \
--driver-memory 18g \
--executor-memory 18g \
--executor-cores 3 \
--num-executors 3 \
--conf "spark.executor.memoryOverhead=2048" \
--conf  "spark.executor.extraJavaOptions=-XX:+PrintGCDetails -XX:+PrintGCTimeStamps" \
SaavnAnalyticsSparkML-0.0.1-SNAPSHOT.jar \
/user/ec2-user/checkpoint_dir \
s3a://bigdataanalyticsupgrad/activity/sample100mb.csv \
s3a://bigdataanalyticsupgrad/newmetadata/* \
s3a://bigdataanalyticsupgrad/notification_clicks/* \
s3a://bigdataanalyticsupgrad/notification_actor/* \
/user/ec2-user/saavnanalytics_kmeans_s3
```

Screenshot of jar execution on Cloudera VM when input data files are present in HDFS:

Output on screen from jar execution on Cloudera VM:

```
[cloudera@quickstart target]$ spark2-submit \
> --class com.ml.upgrad.saavnanalytics.SaavnAnalyticsKMeans \
> --master yarn \
> --deploy-mode client \
> --driver-memory 18g \
> --executor-memory 18g \
> --executor-cores 3 \
> --num-executors 3 \
> --conf "spark.executor.memoryOverhead=2048" \
> --conf  "spark.executor.extraJavaOptions=-XX:+PrintGCDetails -XX:+PrintGCTimeStamps" \
> SaavnAnalyticsSparkML-0.0.1-SNAPSHOT.jar \
> /user/cloudera/checkpoint_dir \
> /user/cloudera/saavnanalytics/activity/sample100mb.csv \
> /user/cloudera/saavnanalytics/newmetadata/* \
> /user/cloudera/saavnanalytics/notification_clicks/* \
> /user/cloudera/saavnanalytics/notification_actor/* \
> /user/cloudera/saavnanalytics_kmeans_hdfs

Start Time : 2019-05-04 02:58:54


Loading user activity data....

19/05/04 02:58:57 INFO hive.metastore: Trying to connect to metastore with URI thrift://quickstart.cloudera:9083
19/05/04 02:58:57 INFO hive.metastore: Opened a connection to metastore, current connections: 1
19/05/04 02:58:57 INFO hive.metastore: Connected to metastore.

Loading new metadata....


Loading notification clicks data....


Loading notification artists data....

Changing UserID to numeric...

Changing SongID to numeric...

Starting the ALS algorithm to get features from implicit learning....

19/05/04 03:03:51 WARN netlib.BLAS: Failed to load implementation from: com.github.fommil.netlib.NativeSystemBLAS
19/05/04 03:03:51 WARN netlib.BLAS: Failed to load implementation from: com.github.fommil.netlib.NativeRefBLAS

Recieved implicit factors from ALS. However, it is in array format. Working to change it into Vector now...

Starting K-means algorithm to form clusters....

Silhouette with squared euclidean distance = 0.5220915434707392


Saving NotificationNumber1 data for specific notification ids as per problem statement, to a file


Saving UserClusterArtist data to a file
```

Saving NotificationNumber2 data for specific notification ids as per problem statement, to a file

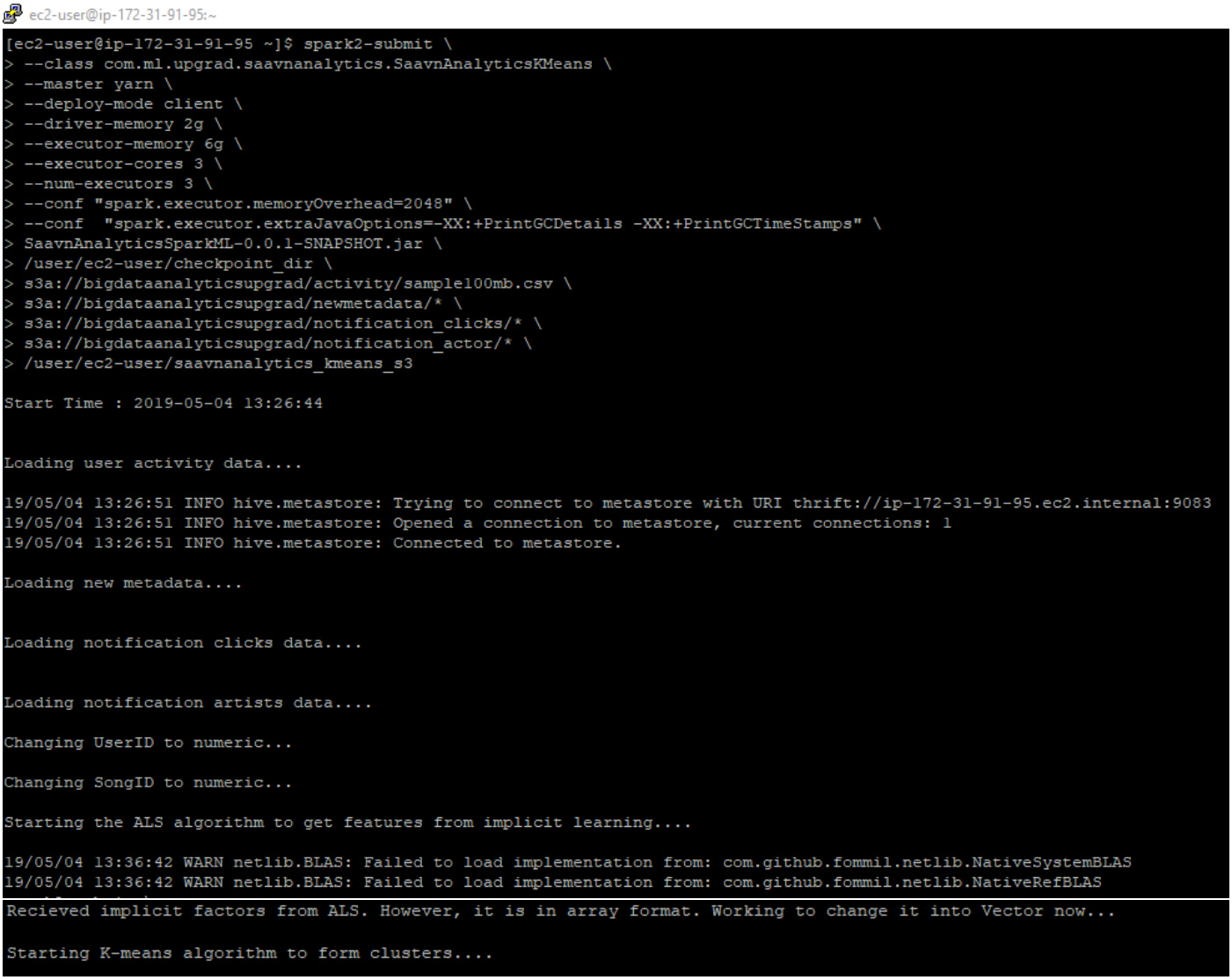
Saving CTR data to a file

Saving CTR data for specific notification ids as per problem statement, to a file

End Time : 2019-05-04 04:32:38

[cloudera@quickstart target]$

Screenshot just to show that same jar gets on executed on EC2 instance as well. I had stopped the instance before 1 hour itself to save further incurrence of charges.



==========THE END==========