VIT®
Vellore Institute of Technology
(Deemed to be University under section 3 of UGC Act, 1956)

School of Computer Science and Engineering

J Component report

Programme : M.Tech integrated computer science engineering with specialization in business analytics.

Course title :Big Data Frameworks

Course code :CSE3120

Slot : F1

TOPIC: COMPARATIVE STUDY BETWEEN ALS AND SGD IN MUSIC RECOMMENDATION SYSTEM.

By,

MADASU DEEPIKA 19MIA1066

HARINISRI G 19MIA1069

Faculty:

Sign:

Date:

INTRODUCTION:

Music is an essential part of human life. Music is the pleasant sound that leads us to experience harmony and higher happiness. With the advancement in technology, music has significantly progressed and increased in terms of quality and volume.

The type of music people create and listen differs according to place and culture. The taste of music even differs from person to person and even in moods of same person. So, it is very useful if we could determine some method to find what kind of music a person might be interested in listening and use this finding to recommend music to him/her. Collaborative filtering is one the most popular filtering techniques today and with this project we aim to enhance it.

For this, We have assumed that the personality of the user might be one of the key factor in his/her music listening habit. Hence, via this project, we are to see if the personality of the user might have any impact on the collaborative filtering enhancement assuming the correlation between the personality and music listening habit exists and also use of alternating least square algorithm to build our model.

LITERATURE SURVEY:

The Literature survey is to focus on creating a music recommendation system that will suggest artists to user based on their listening history. The project uses collaborative filtering technique and uses Apache Spark and Python for implementation.
Bertin Mahieux T et al. proposed a Million Song Dataset Challenge: a large scale, personalized music recommendation challenge, where the goal is to predict the songs that a user will listen to, given both the users listening history and full information (including metadata and content analysis) for all songs. They describes the three algorithms used to produce baseline results: a global popularity based recommender with no personalization, a simple recommender which predicts songs by artists already present in the users taste profile, and finally a latent factor model. All results are based on a train-test split that is similar, but may differ from the split to be used in the contest. The training set consists of the full taste profiles for approximately 1Million users, and partial taste profiles for the 10K test users.

Parmar Darsna proposed a song recommendation system for user to get particular item of his/her interest based on 2 popular algorithms, Content Based Filtering and Collaborative Based Filtering. Content-Based method recommends music based on user data. Content based method music subjective features are Speechness, Loudness and Acousticness etc. These features are stored in database using k-mean clustering algorithm. Collaborative Based method recommends on the user

rating and content sharing between different users. In this method, rating given by user to particular music is considered and find cosine similarity between users. Cold-Start is solved by recommending most popular tracks to new user. The dataset is downloaded from MovieLens Website. It contains 100004 rating and 1296 tag application across 9125 movies. In this model, Spotify API is used to get the songs. In this, any of artist name, if information is available on Spotify should be given and then it will fetch the data related to it.

DATASET:

The dataset is a public song dataset from Audioscrobbler. This data set contains profiles for around 150,000 real people. The dataset lists the artists each person listens to, and a counter indicating how many times each user played each artist. The files are present in the data_raw folder. A more detailed description about the dataset is given in the readme file of data_raw folder.

This data set contains profiles for around 150,000 real people. The dataset lists the artists each person listens to, and a counter indicating how many times each user played each artist

The dataset is continually growing; at the time of writing Audioscrobbler is receiving around 2 million song submissions per day. The dataset included here is a trimmed version of the original dataset containing information about the 50 most active users. The original dataset contains information about 141K users, and 1.6 million artists.

This data is made available under the following Creative Commons license:

http://creativecommons.org/licenses/by-nc-sa/1.0/

user_artist_data.txt:

3 columns: userid artistid playcount

artist_data.txt:

2 columns: artistid artist_name

artist_alias.txt:

2 columns: badid, goodid

We have later increased the dataset in each text file by 500 counts and compared the larger size dataset with smaller dataset . These both dataset were used to perform ALS matrix factorization in Pyspark .

We have took another dataset which comes from last.fm, hosted by GroupLens. It contains the following:

user_artists.dat - userID, artistID, weight - plays of artist by user.

artists.dat - id, name, url, pictureURL

tags.dat - tagID, tagValue

user_taggedartists.dat - userID, artistID, tagID, day, month, year

user_taggedartists-timestamps.dat - userID, artistID, tagID, timestamp

user_friends.dat - userID, friendID. User/friend relationships.

We'll focus on user_artists.dat and artists.dat as they contain all data required to make recommendations for new music artists to a user. Using this dataset we will be using Stochastic gradient descent in python.

PROBLEM STATEMENT:

With the advancement in technology, music has significantly progressed and increased in terms of quality and volume. The type of music people create and listen differs according to place and culture and many artists are not also been recognised very easily. The taste of music even differs from person to person and even in moods of same person. So, it is very useful if we could determine some method to find what kind of music a person might be interested in listening and use this finding to recommend music to him/her.

PROPOSED SYSTEM:

In this proposed system, the motive is to build an efficient recommendation system for music and add a friendly user interface for the benefit for the listeners. The goal of a music recommendation system is to help consumers and the music industry with the discovery and delivery of music. In order to realize the personalized distribution of music, it may be beneficial for recommender designers to understand the music listening behaviors and know about the state of music consumption in the industry.

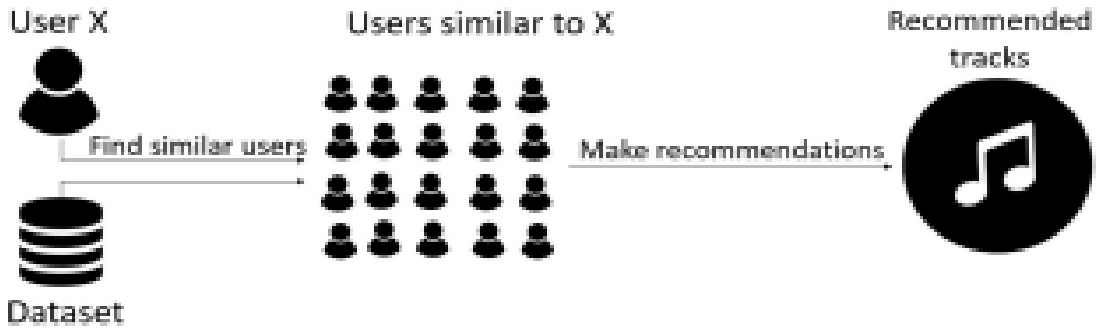
Understanding user preference and behavior can help to propose a reasonable

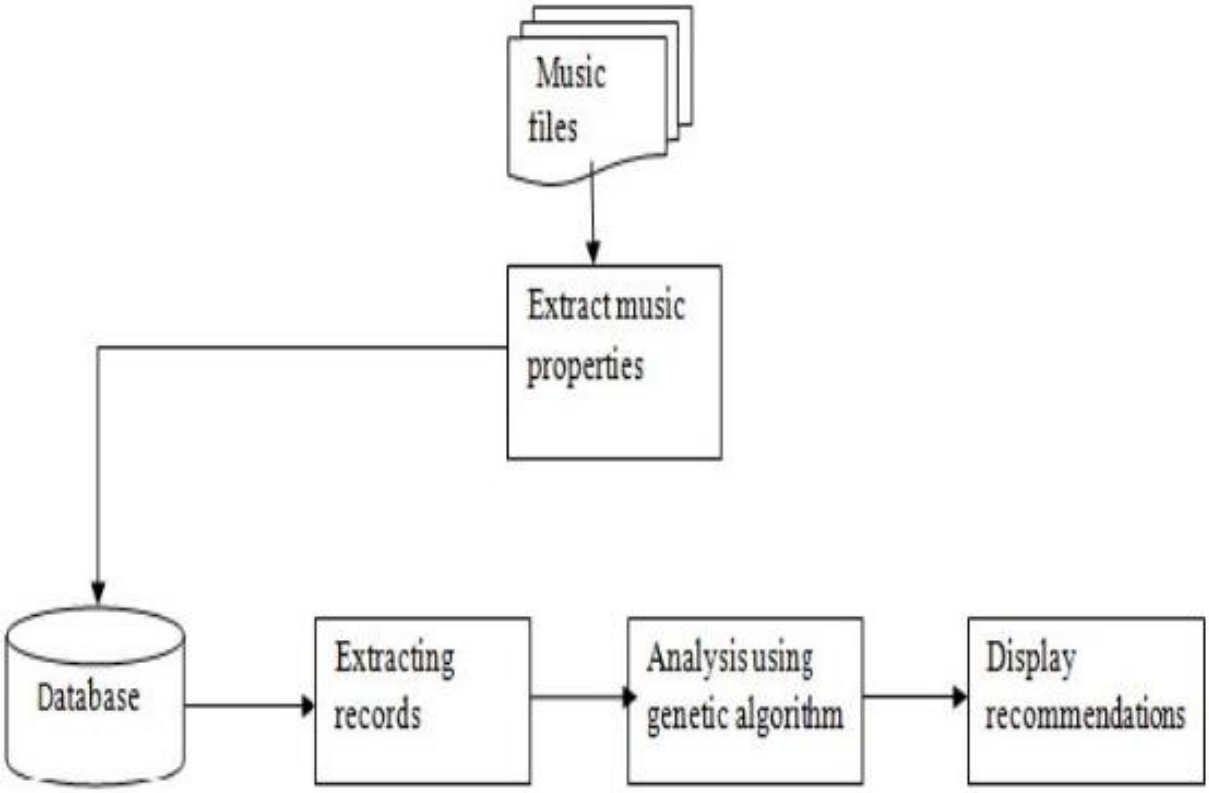
recommendation to a specific user.

- Inspect the data using Spark SQL, and build some basic, but very valuable knowledge about the information we have at hand
- Formally define what is a sensible algorithm to achieve our goal: given the "history" of user taste for music, recommend new music to discover. Essentially, we want to build a statistical model of user preferences such that we can use it to "predict" which additional music the user could like
- With our formal definition at hand, we will learn different ways to implement such an algorithm. Our goal here is to illustrate what are the difficulties to overcome when implementing a (parallel) algorithm
- Finally, we will focus on an existing implementation, available in the Apache Spark MLLib, which we will use out of the box to build a reliable statistical model.
- We should cut out outliers (top 5 artists). With this method, recommendation will be more diverse and accurate.

ARCHITECTURE:

Alternating Least Squares (ALS) matrix factorisation attempts to estimate the ratings matrix R as the product of two lower-rank matrices, X and Y, i.e. X * Yt = R. Typically these approximations are called ‘factor’ matrices. The general approach is iterative. During each iteration, one of the factor matrices is held constant, while the other is solved for using least squares. The newly-solved factor matrix is then held constant while solving for the other factor matrix.





USE CASE DIAGRAM:

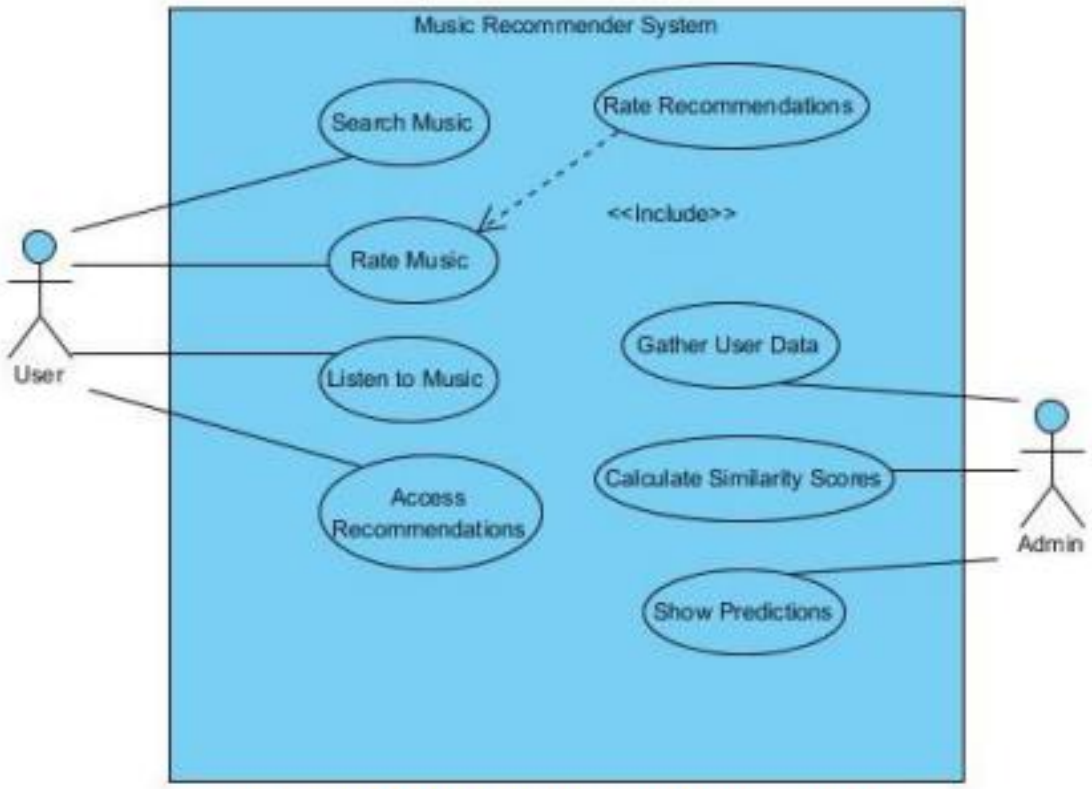


Fig1: Use case diagram for Music Recommender System

MODULES:

- **Data Preprocessing:** The data is checked for null values and the missing values are handled using the suitable method. The incorrect artist names are corrected based on the artist_alias.txt file.

- **Visualizations:** The data is visualized using graphs, plots etc. in order to better interpret and analyze the data. The methods of visualization include word cloud, bar graphs, tree maps etc.

- **Information extraction:** Extraction of information about users like artist playcounts, user-mean playcount, etc by filtering, manipulating and transforming the RDDs. The RDDs are further sorted to find the most active user.

- **Splitting of data:** Data is split into training, validation and test set in the ratio of 4:4:2.
- **Model Selection:** For recommendation systems, models are chosen based on content-based or collaborative-based filtering methods or a hybrid of the two.

- **Model Building and Evaluation:** The accuracy of the model can be evaluated based on the validation set and a suitable error function.

- **Prediction and Recommendation:** Artist recommendations are made based on the predictions made by the model.

IMPLEMENTATION:

Collaborative filtering is one of the most popular recommender system techniques and is being extensively used in many applications such as in music and movies streaming applications, in ecommerce industry for product recommendations, used in restaurant and place recommendations etc.

In this this application, we have used spark and the collaborative filtering technique to build a artist recommender system, which suggests users, artists based on his/her and other's listening history. We have used the Spark mllib's Alternating least square algorithm library to build our model.

To get recommendations, following steps were performed:

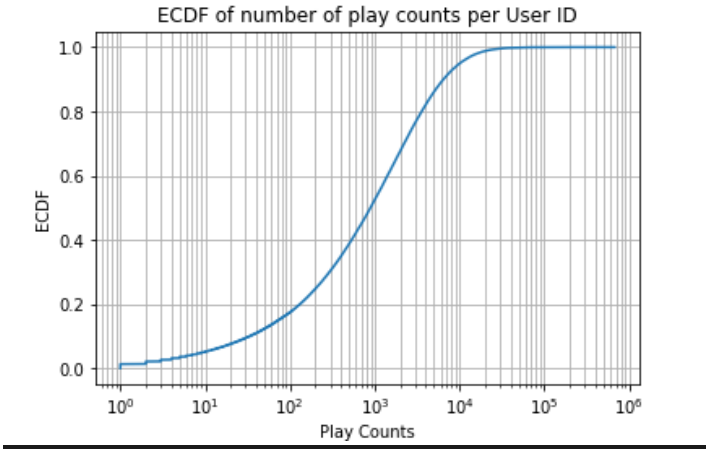
- Perform Data Exploration to understand the dataset.
- Split the data into training, validation and testing set.
- Train the model using implicit feedback.

- Perform parameter sweep and choose the model that perform best on validation data.
- Predict the top 5 artists.

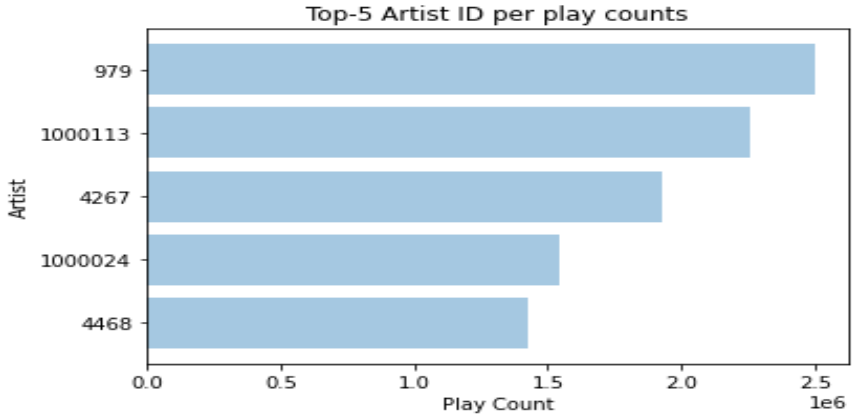
RESULT:

→ **SMALLER DATASET :**

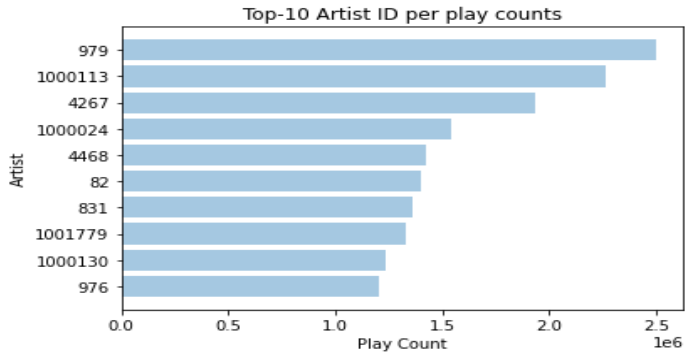
1) **ECDF OF NUMBER OF PLAY COUNTS PER USER ID FOR SMALLER DATASET:**

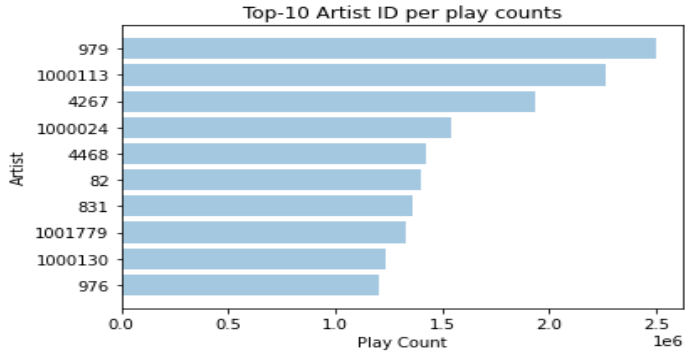


2) **Plot a bar chart to show top 5 artists In terms of absolute play counts.**

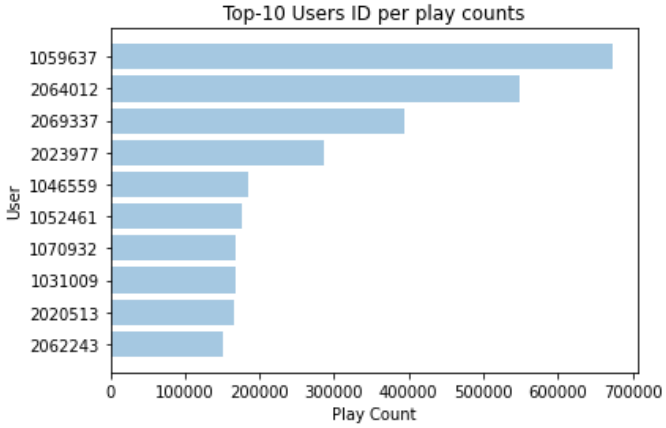


3) **calculate top-10 artists in term of play counts and plotting them**

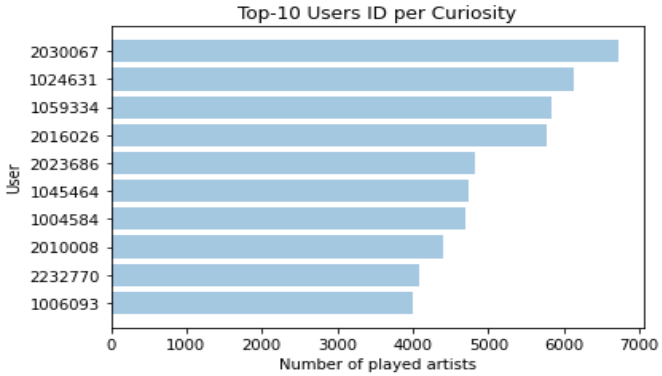




4) Calculate top 10 users in term of play counts



5) Calculate top 10 users ID per curiosity :



6) ALS MATRIX FACTORIZATION FOR SMALLER DATASET AND RECOMMENDATION:

In [47]:
```
rankList = [2,10,20]
for rank in rankList:
     model = ALS.trainImplicit(trainData, rank , seed=345)
     modelEval(model,validationData)
```

```
The model score for rank 2 is ~0.07535599799693796
The model score for rank 10 is ~0.09765418032407164
The model score for rank 20 is ~0.08667584027612182
```

We choose the model which yielded the highest accuracy as our final model to make recommendations.

In [48]:
```
bestModel = ALS.trainImplicit(trainData, rank=10, seed=345)
modelEval(bestModel, testData)
```

```
The model score for rank 10 is ~0.06445912100969664
```

Find the top 5 artists for a particular user and list their names

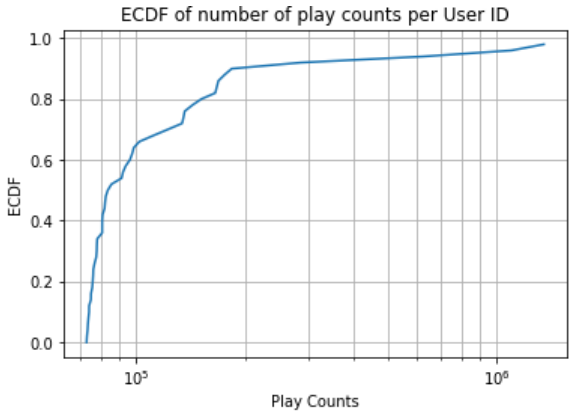
Our model is ready to make recommendations. We now make top 5 artist recommendations for user 1059637. The function recommendProducts takes userId as the first input parameter and an integer n, while returning the top n highest ranked recommendations.

In [49]:
```
TopFive = bestModel.recommendProducts(1059637,5)
for item in range(0,5):
    print("Artist "+str(item)+": "+artistData.filter(lambda x:x[0] == TopFive[item][1]).collect()[0][1])
```

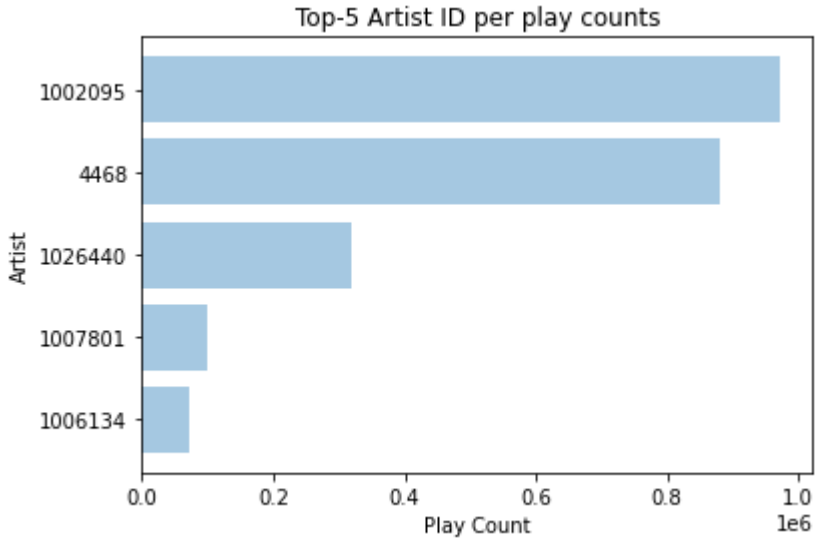
```
Artist 0: Something Corporate
Artist 1: My Chemical Romance
Artist 2: Rage Against the Machine
Artist 3: blink-182
Artist 4: Evanescence
```

→LARGER DATASET:

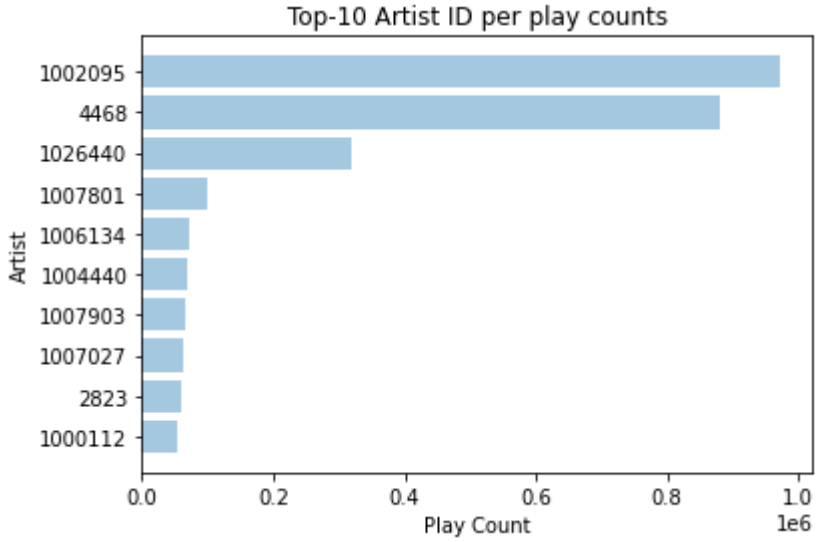
1) ECDF OF NUMBER OF PLAY COUNTS PER USER ID



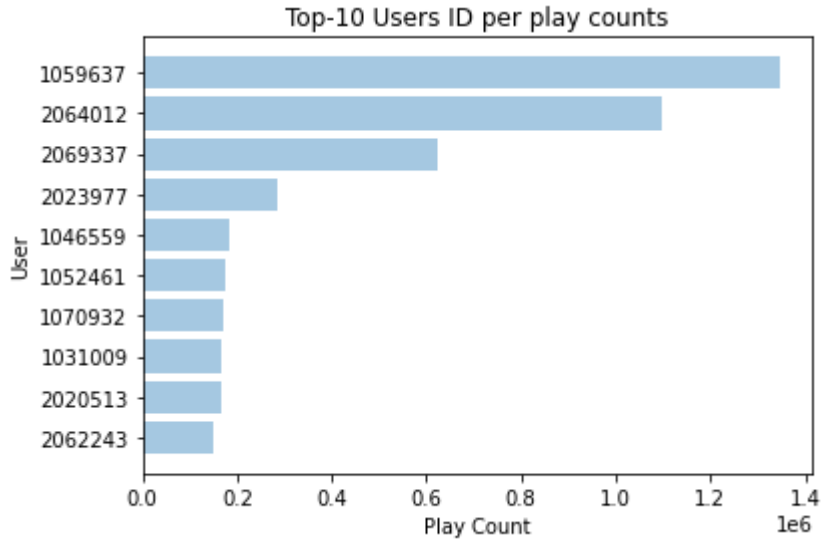
2) TOP 5 ARTIST ID PER PLAY COUNTS



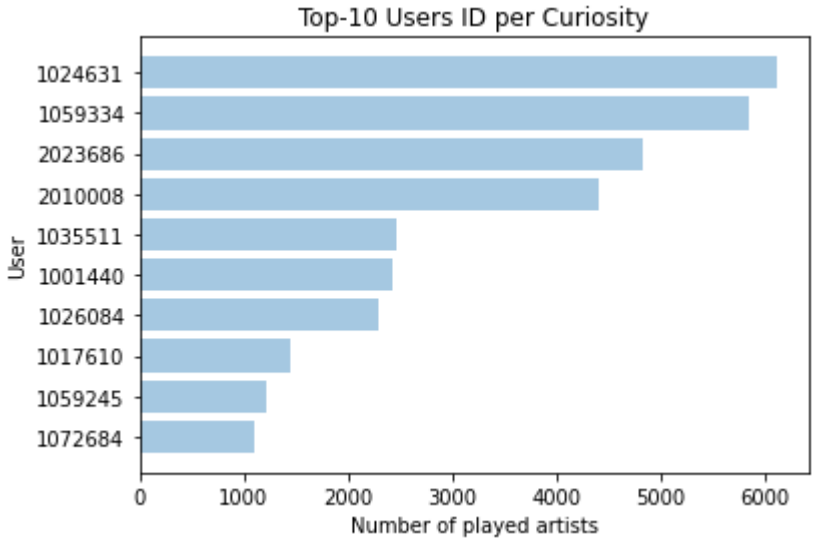
3) TOP 10 ARTIST ID PER PLAY COUNTS



4) TOP 10 USERS ID PER PLAY COUNTS



5) TOP 10 USERS ID PER CURIOSITY:



6) ALS MATRIX FACTORIZATION FOR LARGER DATASET AND RECOMMENDATION:

In [44]:
```
rankList = [2,10,20]
for rank in rankList:
     model = ALS.trainImplicit(trainData, rank , seed=345)
     modelEval(model,validationData)
```

```
The model score for rank 2 is ~0.07478965804909071
The model score for rank 10 is ~0.09949125799664481
The model score for rank 20 is ~0.09020351262501673
```

In [45]:
```
bestModel = ALS.trainImplicit(trainData, rank=10, seed=345)
modelEval(bestModel, testData)
```

```
The model score for rank 10 is ~0.06814203309788236
```

In [46]:
```
TopFive = bestModel.recommendProducts(1059637,5)
for item in range(0,5):
    print("Artist "+str(item)+": "+artistData.filter(lambda x:x[0] == TopFive[item][1]).collect()[0][1])
```

```
Artist 0: Thrice
Artist 1: The Used
Artist 2: The Blood Brothers
Artist 3: System of a Down
Artist 4: Rage Against the Machine
```

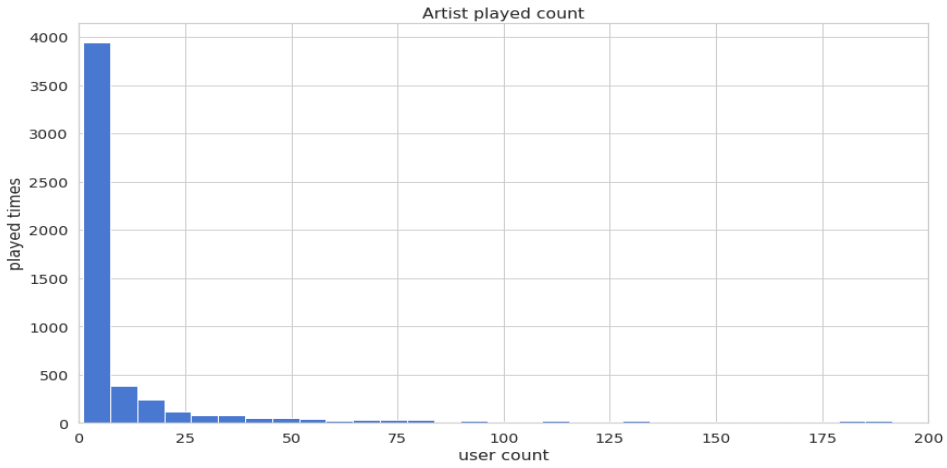
SCRIPT TIME FOR :

(A)SMALLER DATASET : The script takes 0.638038 seconds

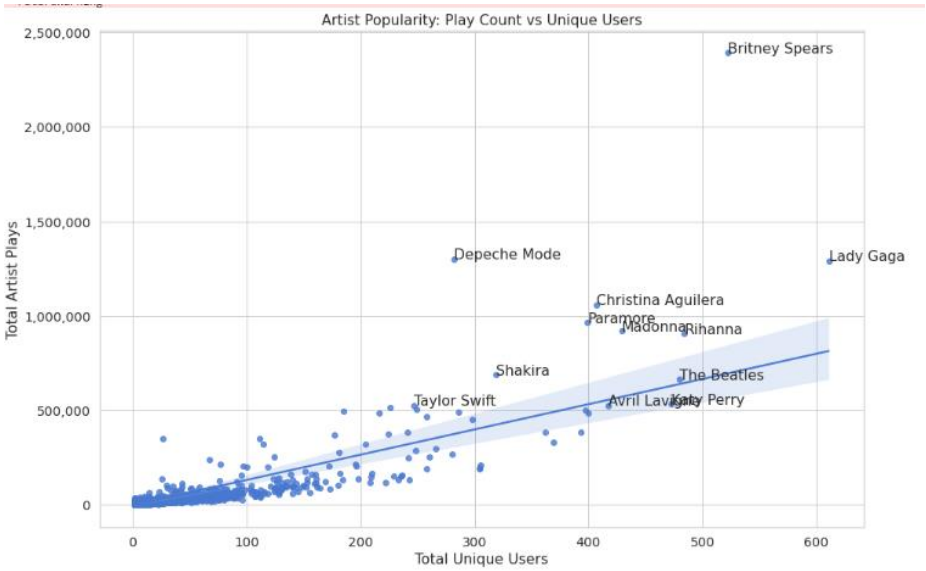
(B)LARGER DATASET : The script takes 1.059970 seconds.

WORKING ON PYTHON FOR LAST.FM DATASET USING SGD MATRIX FACTORIZATION:

1) Artist played count:



2) Artist popularity play count vs unique users



3) Recommendation using SGD model

Out[104…

	id	name	rating
0	3117	Pete Yorn	0.104792
1	6403	The Highwaymen	0.103575
2	7547	Los Paranoias	0.103198
3	13111	Votchi	0.102901
4	13198	Raindancer	0.101285
5	13830	Phil Ochs	0.101205
6	15742	Nelstar*	0.100503
7	16190	Heroin	0.100498
8	17805	Huski	0.100484
9	18300	Ken Laszlo	0.100358

CONCLUSION:

By doing the analysis and recommendation using python and spark we conclude that each user is recommended with a uniqueplaylist based on their interest. This also provides the information for the user like what are the most popular songs, who are all the artist, which artist is liked by the people through which he gets to know about the current trend. Music recommender system plays a significant role in identifying a set of music for users based on user interest. Although many move recommendation systems are available for users, these systems have the limitation of not recommending the music efficiently to the existing users. This paper presented a music recommender system based on ALS matrix factorization in collaborative filtering using Apache Spark and SGD matrix factorization in python. From the results, the selection of parameters of ALS and SGD algorithms can affect the performance of building of a music recommender engine. System evaluation is done using various metrics like RMSE.

FUTURE WORK:

In the future, we will try to add a greater number of artists and languages which will make the recommendation stronger giving even better playlists for the users. We can try the system with other machine learning models as well to compare the results and look for better results. When there are millions of songs out there, our motive was to give the users their preference of songs which they want to listen to and we are satisfied after getting one step closer to it. For future applications, an emotional detector system that will recommend the songs by recognizing our facial emotion can be developed.

REFERENCE:

- McFee, B., Bertin Mahieux,T., Ellis, D. P., Lanckriet, G. R. The million song dataset challenge, In Proceedings of the 21st international conference companion on World Wide Web (pp. 909916) ACM. April 2012.

- Parmar Darsna "Music Recommendation Based on Content and Collaborative Approach & Reducing Cold Start Problem", IEEE 2nd International Conference on Intensive Systems and Control, 2018.

- Sun, J. (2022). Personalized Music Recommendation Algorithm Based on Spark Platform. Computational Intelligence and Neuroscience, 2022, 1–9. https://doi.org/10.1155/2022/7157075

- *Barracuda Networks*. What is Content Filtering? | Barracuda Networks. (n.d.). Retrieved February 25, 2022, from https://www.barracuda.com/glossary/content-

filtering#:~:text=Content%20filtering%20is%20the%20use,also%20by%20home%20computer%20owners

- Luo, S. (2019, February 6). Intro to Recommender System: Collaborative filtering. Medium. Retrieved February 25, 2022, from https://towardsdatascience.com/intro-to-recommender-system-collaborative-filtering-64a238194a26
- https://towardsdatascience.com/music-artist-recommender-system-using-stochastic-gradient-descent-machine-learning-from-scratch-5f2f1aae972c
- Simple Matrix Factorization example on the Millionsong dataset using Pyspark. (2018). Medium.Retrieved22November2018,from ht tps://medium.com/@connectwithghosh/simp le-matrix-factorization-example-on-themusiclens-dataset-using-pyspark9b7e3f567536\