Million Song Dataset Recommendation Project Report

Aravind Reddy Dandu (NetID: ad1452) **Sai Mounica Pothuru** (NetID: sp1912)
Chaitanya Sharma Domudala (NetID: cd817) **Sri Kaavya Toodi** (NetID: st938)

May 3, 2021

Abstract

In this project, we adopt to explore, plan, and implement a song recommendation system. Introducing the data, Million Song Dataset [1] (MSDS), a freely-available collection of audio features and metadata for a million contemporary popular music tracks, we will find the correlation between users and songs to predict and recommend multiple songs to users based on their preference. The focus of the project is to explore the dataset, show various implementations on the dataset and discuss the issues faced along with the methods used.

1 Introduction

Recommendations are pretty popular in domains like products, movies, or hotels, and even music. But the music domain has certain specific characteristics that should be taken into account when creating a recommendation system. The multiplex semantics involved in the music objects makes their representation critical for the recommendation systems. A few of them worth mentioning would be duration, number of items in a music catalog with millions of tracks, content-based features. While music is categorized into various genres such as classical, pop, rock, jazz, folk, etc., listening to them is much easier due to various music applications available both online and offline. Sometimes repeated recommendations are appreciated by the user, unlike in other domains like movies or products where users usually disfavor the same recommendations. Hence, developing a music recommender system that can search music libraries automatically and suggest songs based on user preference is much more welcomed. This project will be implementing the memory-based and item-based recommendations while the content-based can be a future scope.

2 Data

Million Song Dataset is a huge data-set of total size 280GB. To take the complete dataset and do the analysis it might require days or even years. A subset of the dataset named triplets is available which would give the information on the users, the songs they listen to and the number of plays. This subset of data("train_triplets.txt") is used which is of 3GB size along with two other files and PySpark is used to analyze the data in a streaming way. The "song_to_tracks.txt" has information of ids of songs and their tracks. The "track_metadata.csv" contains the metadata of the dataset. Below are the important features in the dataset that are being used:

- UserID
- SongID
- TrackID
- Plays
- Artist
- Title

3 Data Pre-Processing

In this project, we are using Spark to parallely process the data for faster results. Below are few steps followed in preparing the data

1. Create new schemas to accomodate the 3 files train_triplets.txt, song_to_tracks.txt and track_metadata.csv
2. Load the data from TXT and CSV files to Spark schemas by using SQLContext class
3. Convert the datatypes to suit the data appropriately
4. Create new dataframes by joining users with songs and songs with tracks to get the number of plays

4 Data Exploration

Let us understand more about the data from the explorations below.

As we can see from the below plot, the percentage of songs that have been played 10 times is not even 1%. The maximum percentage songs in the dataset are played only once.

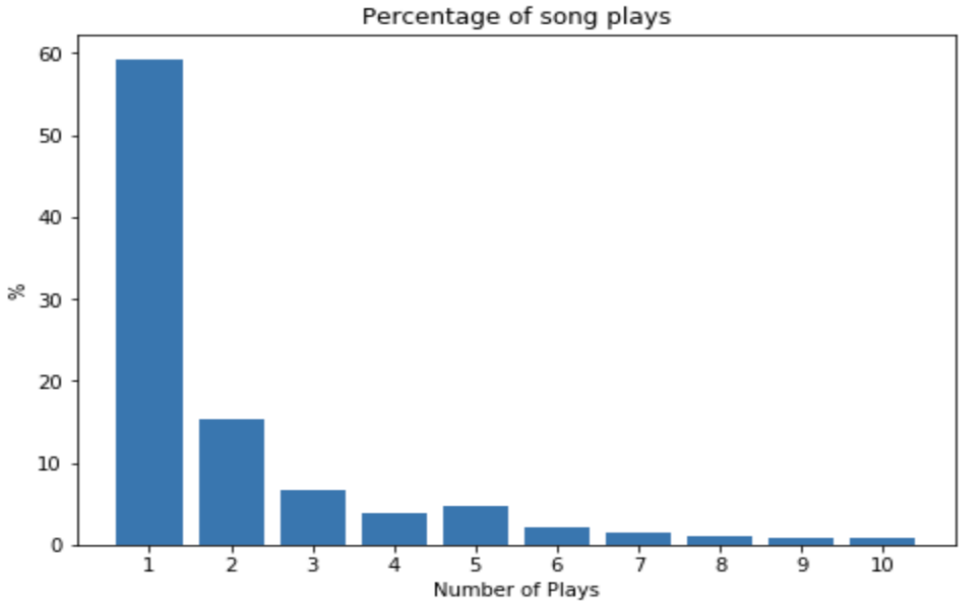

Figure 1: Number of plays of song vs % of total plays

The below three plots shows the trend for Artist Popularity with Plays, Atist Familiarity with the number of plays and the correlation between the Artist's Popularity and Familiarity

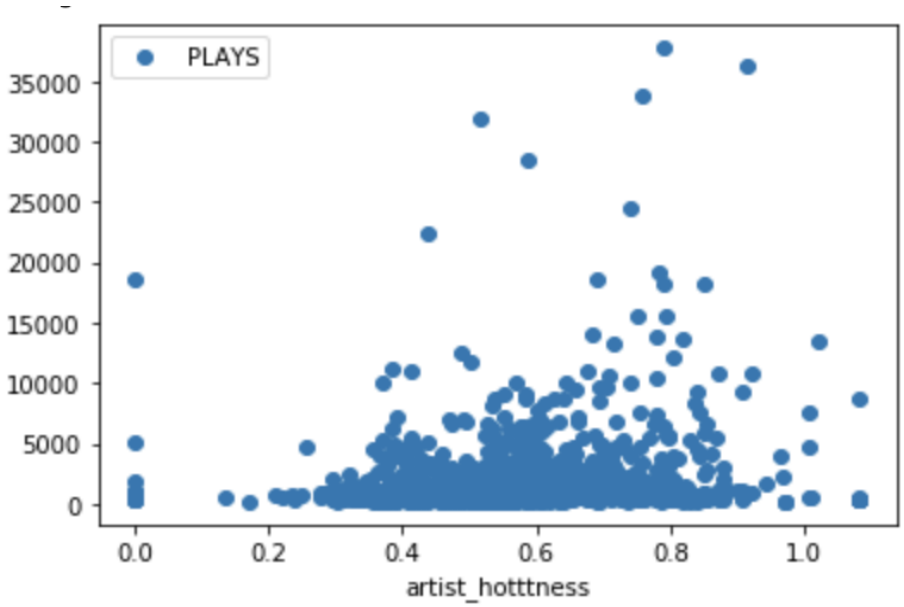

Figure 2: Trend for Total Plays w.r.t. Artiste Hotness

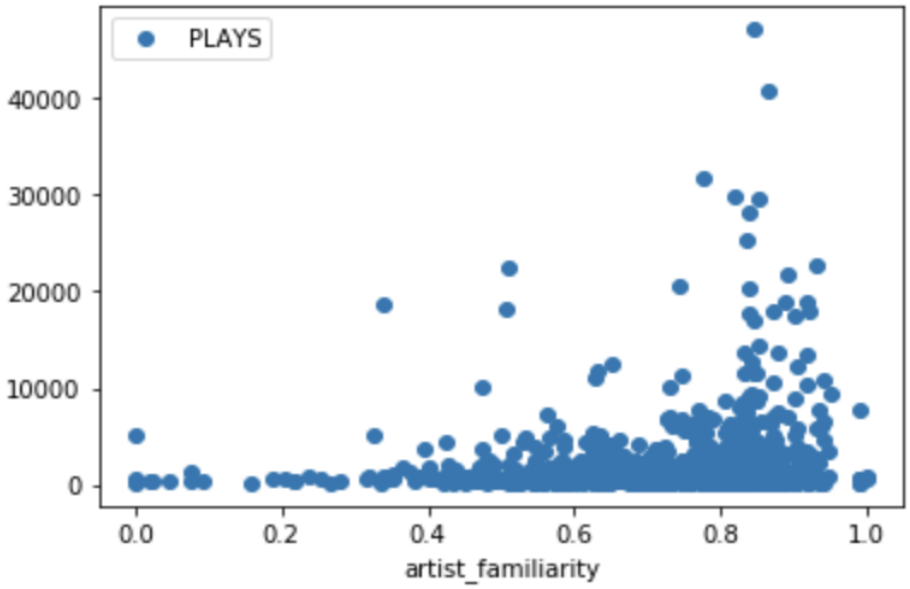

Figure 3: Trend for Total Plays w.r.t. Artiste Familiarity

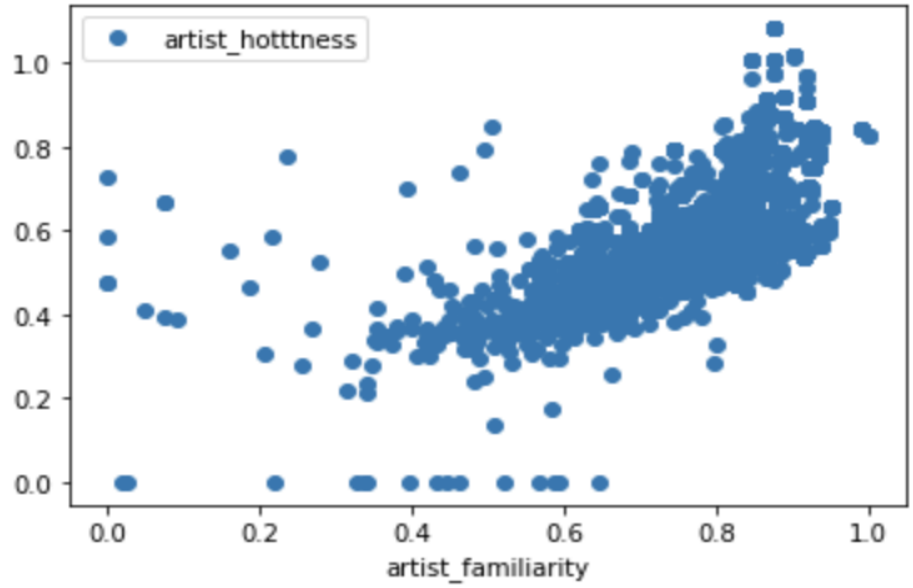

Figure 4: Correlation b/w Artist Trend and Familiarity

5 Data Modelling

Recommendation of music is done in three levels.

1. Basic Recommendations
2. Memory based Collaborative Filtering
3. Collaborative Filtering -ALS (Alternating Least Squares)

5.1 Basic Recommendations:

For the basic level of recommendations, we will use the number of times the song has been listened to. We are going to use the triplets data here and get the details of the songs, the number of times they are played and the number of users using Spark. Using the groupBy function on songIds and sum them using aggregate function to get user count of individual songs. For better recommendations, we are using the distinct number of users and filter it by a count of 200.

5.2 Collaborative Filtering (CF):

Collaborative Filtering - A traditional and powerful tool for recommender systems that produces automatic predictions by collecting multiple user preferences or taste information. It only needs the information of users' historical preference on a set of items and is based on the assumption that past tend to also agree in the future. User preference can be categorized based on direct or indirect feedback. While ratings, reviews come under direct preference, the page views, clicks, number of times song played etc comes under indirect feedback. In this project, we will be using the implicit rating or the indirect feedback on MSDS dataset to recommend songs to users.

5.2.1 Memory based Collaborative Filtering:

The first method that we implemented in Collaborative Filtering is memory based. The main idea behind this method is to compare one half of a user's preference with others and predict the other half of the songs with precision.Let us discuss more on how this is implemented.

First, let's split the triplets data into training and testing data. Now, get the test set and divide it into two halves. The first half of the data is compared to the users in training set. A similarity score is generated by the formula

Let's say user A has listened to song a 2 times and song b 4 times. Total play count is 6. Now song a is given a score of 0.33 and song b 0.66. This is calculated for all users. If we have to calculate similarity of this user with user B who has listened to song a 1 time and song b 1 time, below is procedure. For user B scores will be 0.5 for each song with total play count 2.

For song a,

$$x(a) = 0.33, y(a) = 0.5 \text{ with } x(a) \text{ smaller}$$

$$S_1(a) = x(a) + (\frac{x(a)}{y(a)} \times (y(a) - x(a))) = 0.33 + (\frac{0.33}{0.5} \times (0.5 - 0.33)) = 0.4422$$

$$x^{'} = 0.33 * 6 = 2, y^{'} = 0.5 * 2 = 1$$

Calculate using above formula again gives 1.5

This is done in similar fashion for all users other than user A. Now, all the similarities and re-scaled and summed up to get the final score.

The higher the score, the more similar the user preferences are.Now based on the similarity score, let us say if user A has highest similarity with User D, then a set of recommendations from User D(songs that user A hasn't listened but user D did) is sent as response to User A querying. These results are compared with the other half of User A and an accuracy is calculated based on the matched songs.

5.2.2 Item based Collaborative Filtering -ALS (Alternating Least Squares):

The second method of implementation is most frequently used ALS abbreviated as Alternating Least Squares.

For song recommendations, we start with a matrix whose values are the number of song plays by users . Since not all users have listened to all songs, we don't know all the entries in this matrix, which is precisely why collaborative filtering is needed. For each user, we have the number of song plays for only a subset of the songs. With collaborative filtering, the idea behind is to approximate the matrix by factorizing it as the product of two matrices: one that describes the properties of each user and one that represents the properties of each song.

We want to choose these two matrices such that the error for the users/song pairs where we know the correct number of plays is limited. The Alternating Least Squares algorithm does this by first arbitrarily filling the user's matrix with values and then optimizing the value of the songs such that more accurate results are produced. Then, it holds the songs matrix constant and optimizes the value of the user's matrix. This alternation between which matrix to optimize is the reason for the "alternating" in the name.

Given a fixed set of user factors (entries in the user's matrix), we utilize the known number of plays to track down the best values for the songs factors using the least-squares optimization. Then we modify and pick the best user factors given fixed songs factors.

In the model generated, the similarity is based on items rather than users. The triplets dataset is split into train, validation and test sets. The ALS learner method of Pyspark ML pipeline is used to generate a ALS model with items as songs, plays as implicit rating and userids as users.

Initially we start with building a matrix with rows as songs and columns as users. The values are set when the data is trained with training set. Before that we need to set an important hyper-parameter which is Rank. The Rank of the matrix determines whether the model we fit is going to be under fitting or over fitting. A lower rank will under fit while a higher rank will over fit. For this project, we are setting the rank to be 4,8,12,16. The ALS fit will generate a new column prediction containing predicted value. A prediction is run on these models against validation set. The models with lowest error rate is chosen. The best model is tested against test dataset and RMSE is calculated.

An additional analysis is done by considering only the songs that had been listened to atleast twice to improve the RMSE. The same procedure is followed with this dataset and have seen that the results have been improved by

6 Results

Let us look at the results for all three models generated.

- Basic recommendations gives the top songs according to the number of times the songs has been played
- Memory based Collaborative filtering has a score of 10-12 out of 100 which means that 10-12 plays are accurate out of 100 which is almost 5 times better than the random generation
- ALS generated a RMSE of 7.51 with average of 3 songs against all songs considered. But with the filtered dataset of atleast 2 songs repeated, we had a better RMSE of 8.64 with an average of 6 songs. Even though the RMSE looks higher, the average is also higher assuring that it is a better model.

7 Conclusion

To conclude, we have generated three different types of models that would predict the songs listened based on user's preference. We have modified the datasets for the ALS to give a better model with higher accuracy. In comparison, ALS is a better model to the memory based model and the randomly generated basic recommendations. But the accuracy is quite low for ALS in number due to the NaNs that come due to lack of data. Since, we are using Spark, this analysis can be done for much larger data and more training can give better results.

8 Future work

In the future, we would like experiment on Content based including memory based on the complete dataset and use various Collaborative Filtering methods that include Deep learning to get the better models and hence better accurate recommendation system.

References

[1] Thierry Bertin-Mahieux, Daniel P.W. Ellis, Brian Whitman, and Paul Lamere. The Million Song Dataset. In Proceedings of the 12th International Society for Music Information Retrieval Conference (ISMIR 2011), 2011.

[2] Fabio Aiolli.: Preliminary Study on a recommender system for the Million Songs Dataset challenge. (2011)

[3] Bertin-Mahieux, T ., Eck, D., Mandel, M.: Automatic tagging of audio: The state-of-the-art. In Wang, W., ed.: Machine Audition: Principles, Algorithms and Systems. IGI Publishing (2010) In press.

[4] https://www.ee.columbia.edu/ dpwe/pubs/BertEWL11-msd.pdf

[5] Musicbrainz: a community music metadatabase, Feb. 2011. MusicBrainz is a project of The MetaBrainz Foundation, http://metabrainz.org/.