CAPSTONE PROJECT
MUSIC RECOMMENDATION SYSTEMS
MILESTONE 1 REPORT

Problem definition

1. **Context** - Streaming platforms like Spotify, Soundcloud, and YouTube use different recommendation models to provide personalized suggestions to each user. For example, Spotify uses a machine learning tool called the approximate nearest-neighbour search algorithm to group songs and users together based on shared attributes or qualities.

2. **Objective -** Build a recommendation system to propose the top 10 songs for a user based on the likelihood of listening to those songs.

3. **Key Questions –**
   - Is there enough data/user interaction to build a recommendation system?
   - How can we explore that data and obtain insights?
   - Which tools can we use to recommend songs to users?
   - Which models will give us the best results?
   - How do we measure the results of our models?

4. **Problem Formulation** - Given the following dataset of user-song interactions, how do we use our knowledge of recommendation systems to suggest songs to users to maximise user satisfaction and their engagement with the platform, and hence maximise our profits?

Data Exploration

1. **Data description** - The core dataset is the Taste Profile Subset released by The Echo Nest as part of the Million Song Dataset. There are two files in this dataset. One contains the details about the song id, titles, release, artist name, and the year of release. The second file contains the user id, song id, and the play count of users. Data Source - http://millionsongdataset.com/

   - **song_data**
     - song_id: A unique id given to every song
     - title: Title of the song
     - Release: Name of the released album
     - Artist_name: Name of the artist
     - year: Year of release
   - **count_data**
     - user _id: A unique id given to the user
     - song_id: A unique id given to the song
     - play_count: Number of times the song was played.

2. **Observations and Insights** –
   I. **Data info, data types and missing values** -
      i. count_df
         1. There are 2,000,000 observations and 4 columns in the dataset. One of the columns doesn't have any information and seems to be there by error so we have taken it out.
         2. All columns have 2,000,000 non-null values, i.e., there are no missing values.

3. user_id and song_id are of object data type and and unnamed: 0 and play_count are of int64 data type.

ii. song_df

1. There are 1,000,000 observations and 5 columns in the dataset.
2. A few columns have null values. The column 'year' has the most null values. song_id and artist_name have no null_values.
3. All columns have object data types except 'year' which has a float64 data type.

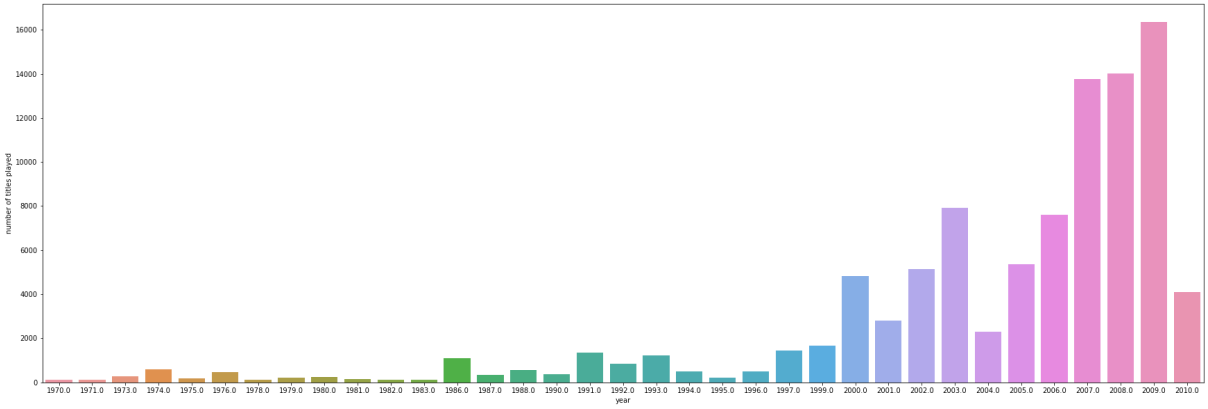
We have enough data to support our recommendations.

II. **Size of the dataset after filtering** – After filtering our dataset such that all users have listened to at least 90 songs, the songs have been played by at least 120 users, and no user has listened to any song more than 5 times, we have 117,876 observations.

III. **The total number of users, songs, and artists in the dataset -** There are 3156 users, 9999 songs and 3374 artists in this data set. The total number of user-song interactions is 3156 times 9999 = 31,556,884. We can see that the number of observations is nowhere near the number of possible observations in either case. We can build a recommendation system to recommend songs to users that they have not interacted with and likely increase user interaction by a huge margin.

IV. **Most interacted items -** The most interacted songs are 'Dog Days Are Over (Radio Edit)', 'Sehr kosmisch', 'Use Somebody', 'Secrets', and 'Fireflies'. The most interacted users are 3237, 15733, 62759, 43041, 27401.

V. **Histogram depicting play_count of songs from each year –** We notice that songs released in recent years are played more.



Proposed approach

1. **Potential Techniques** –
    1. Popularity-based recommendation systems
    2. Collaborative Filtering
        - User-user similarity-based system
        - Item-item similarity-based system
    3. Model-Based Collaborative Filtering - Matrix Factorization
    4. Cluster-Based Recommendation System
    5. Content-Based Recommendation Systems
2. **Overall solution design:** Build each of the models listed above, optimize parameters when possible, and compare them.
3. **Measures of success:** Precision@k, Recall@k, and F1-score@k

- Precision@k - It is the fraction of recommended items that are relevant in top k predictions. The value of k is the number of recommendations to be provided to the user. One can choose a variable number of recommendations to be given to a unique user.

- Recall@k - It is the fraction of relevant items that are recommended to the user in top k predictions.

- F1-score@k - It is the harmonic mean of Precision@k and Recall@k. When precision@k and recall@k both seem to be important, it is useful to use this metric because it is representative of both of them.

APPENDIX

Q. As the user_id and song_id are encrypted. Can they be encoded to numeric features?

A. Label encoding is a method to convert categorical variables to numerical ones. In the current project, as the user_id and song_id are given in the text format, it becomes necessary to convert them to numbers so that we can pass them to the algorithm.

Q. As the data also contains users who have listened to very few songs and vice versa, is it required to filter the data so that it contains users who have listened to a good count of songs and vice versa?

A. We want our recommendations to be supported by enough data. So we can reduce the dataset by applying certain restrictions, while also making our model computationally less expensive.

Here, we will be taking users who have listened to at least 90 songs, and the songs that have been played by at least 120 users. Moreover, the number of songs with play_counts more than 5 is extremely low, which will result in an extremely sparse user-item interaction matrix. This will also increase the size of the matrix, which will make building a recommendation system model computationally expensive. Hence, all such songs are removed from the data.

Q. What other insights can be drawn using exploratory data analysis?

A. We can get top releases and artists. We can get the total number of releases per year and in total. We can create a new data frame of users-artists, and users-releases with play count.