Milestone 2.

Music Recommendation System

1. Refined Insights

The most meaningful insights from the data relevant to the problem

Before going into applying the algorithms to build recommendation systems, let us first interpret the data for our problem. Originally, we were given two datasets count_df with 2 000 000 records and song_df with 1 000 000 records. To utilize the data, we combined them and made appropriate changes to work with only relevant columns and rows for our problem. The merged data would have 2 000 000 records which makes it hard to apply for our problem due to the large dataset. Hence our final data is reduced, and it consisted of 117 876 records and 7 columns. Even though small dataset is also helpful in making decisions, it does not aim to impact business to a great extent, rather for a short amount of time. However, for the sake of time, to be able to draw useful insights and get general idea we will be dealing with small data for our problem.

Key variables that are necessary for our problem are user_id, song_id and play_count. That makes it clearer to solve the problem and get the desired outputs. So, we would be using the final data to do a large, personalized music recommendation system with the goal of predicting the songs that a user is going to listen. We learn from users' listening history and full information of all songs. Our goal is to make our system largescale and more personal to the users.

2. Comparison of Techniques and their Performances

Next, we will perform the following popular recommendation algorithms to our dataset. Consequently, we will compare them according to chosen criteria and try to find out the best algorithm for our problem. From that we will be able to recommend relevant songs for the user.

Popularity-based recommendation system

The most trivial recommendation algorithm is to present each song in decreasing order of its popularity skipping those songs already consumed by the user, regardless of the user's taste profile. So, we found the top 10 songs for a recommendation based on the average play count of song. Pros of this method are it is easy to implement and no need for the user's historical data. Cons of the method are the output is not personalized (users and songs' information are not considered) and some songs may never be listened.

User-based similarity engine

User-based and item-based similarity engines are subtypes of collaborative filtering recommendation system. So now we will attempt each of the methods separately.
In user-based recommendation system, users who listen to the same songs in the past tend to have similar interests and will probably listen to the same songs in future. In this type of recommendation system, we did not need any information about the users.

Item-based similarity engine

In the item-based recommendation method, songs that are often listened by the same user tend to be similar and are more likely to be listened together in the future by some other user. In this type of recommendation system, we did not need any information about the songs.

Advantages of collaborative filtering methods are it does not need any domain knowledge, personalized to users and adaptive to the preferences changing over time. Disadvantages are this method cannot handle new data, unable to estimate play counts if there are a very small number of user preferences.

Matrix factorization-based engine

This method can be used to find the relationship between songs' and users' entities. The association between users and songs data is determined to find similarity and make a prediction based on both song and user entities. The associations are artist_name or release in our case. We can infer these latent features through this method. The advantage of the method is it allows us to get more personalized recommendations.

Cluster-based recommendation system

The idea behind cluster-based recommendation engine is it creates a group of users and cluster them based on preferences. We explored the similarities and differences in people's tastes in songs based on how they rate different songs. We clustered similar users together and recommended songs to a user based on play_counts from other users in the same cluster. Using clustering techniques in recommendation systems helped us to identify groups of users with similar tastes, and it supposed to greatly improve the performance.

Content-based recommendation system

Based on song's description and user's preference profile we are now able to construct content-based recommendation system which requires information about the users and songs as title, release name and artist name. The idea was to make recommendations by looking for music whose features are very similar to the tastes of the user. Pros of this method is it doesn't require a lot of user data and it does not suffer from cold start. Cons of the method is it requires a lot of domain knowledge, and it has limited ability to expand on the users' existing preferences. Content-based recommenders work solely with the past interactions of a given user and do not take other users into consideration. As an illustration we recommended 10 songs similar to 'Learn to Fly'.

We described each of the algorithms and now we can compare their effectiveness and performance.

Below is the comparison table of different methods based on F-1 score metric. Each of the methods mentioned above were improved by optimizing hyperparameters in that method. They are now called optimized versions.

	User-based	Optimized User-based	Item-based	Optimized item-based
F-1 Score (%)	50.4	52.5	39.7	50.6
	SVD method	Optimized SVD	Cluster-based	Optimized cluster-based
F-1 Score (%)	49.8	50.2	47.2	46.5

As we can see relatively better technique is *optimized user-based method* based on F-1 score metric. It further can be improved by considering more possibilities for the hyperparameters.

3. Proposal for the Final Solution Design

What model do you propose to be adopted?

Based on the comparison, relatively good model is optimized user-based collaborative filtering method. We are considering F-1 score metric to decide the model performance. However, there are other relevant metrics such as precision and recall. Precision is the fraction of recommended song that are relevant in top 10 predictions; recall is the fraction of relevant songs that are recommended to the user in top 10 predictions. F-1 score can be considered as the average of them. The main idea behind precision and recall is to consider some threshold value. So, song that is predicted play count is higher than the threshold = 1.5 is a recommended song, if the predicted play count is below the threshold, then that song will not be recommended to the user. In the present case, precision and recall both need to be optimized as we would like to minimize the losses and errors. Hence, the correct performance measure was the F-1 score.

Why is this the best solution to adopt?

Generally, collaborative filtering method has strong predictive power among recommendation systems. It does not require any song information, rather considers user and item preferences. People usually tend to get the recommendations from someone that share similar tastes. So, user-based collaborative filtering engine would be suitable for our problem. Even though optimized user-based method had the highest F-1 score among all methods. The value for F-1 score is considered as not good, so model performance is not well. The reason is we are using small data (117 876 rows) comparing to the original data (2 000 000 rows). It is only 5% of the whole dataset. So, the huge lack of information lead to the bad performance of the methods.