SPOTIFY MILLION PLAYLIST DATASET AND RECOMMENDER SYSTEM

BY

Tram Thi Bich Ngo

March 2025

Table of Content

Chapter 1

INTRODUCTION

Recommender systems have become an essential component of modern industries, changing the way firms communicate with their customers and improving user experiences across multiple industries. In the entertainment and content streaming businesses, such as big music streaming platforms like Spotify and video streaming services like Netflix, recommender systems assist users in discovering content based on their tastes and preferences. These systems assess user behavior, including as listening habits, watching histories, and platform interactions, to recommend music, playlists, movies, or TV series, to help the business increase their costumer's satisfications. The personalized recommendations not only enhance the experience, but also keep users coming back, improving platform stickiness and time spent on the service. In e-commerce, recommender systems play an important role in boosting sales by proposing products that are relevant to a user's interests, resulting in improved conversion rates, higher average order values, and more effective cross- and upselling. Finally, recommender systems not only influence the future of digital platforms, but they also play an important part in generating success and competitive advantage in today's data-driven economy. Whether it's to boost user satisfaction, improve sales performance, or acquire a better understanding of consumer demands, recommender systems are critical tools for businesses looking to succeed in a quickly changing digital market.

Chapter 2

LITERATURE REVIEW

In this research, I will do a big analyze on a huge dataset of 'Spotify Million Playlists Dataset', and create a Spotify-like recommender systems using Machine Learning models and algorithms. I will apply all types of Recocommender System like: Content-base, Collaborative Filtering, and Hybrid to represent how they bring different results from different approachs.

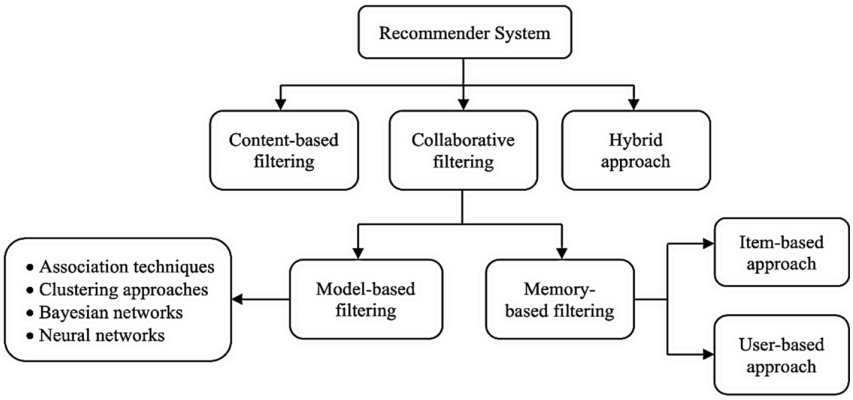


Figure 2.1: type of Recommender System

- **Collaborative Filtering (CF)**: CF-based approaches recommend songs based on user interaction patterns, utilizing techniques like matrix factorization (Singular Value Decomposition, Alternating Least Squares) and nearest neighbor models. In this reaerch, I will focus on Collaborative Filtering using user-based (playlist-based) which will predict the items (songs) based on the other users (playlists) who have the same interaction with songs.

- **Content-Based Filtering (CBF):** This research will focus on the genres of artists and

song lyrics to predict the songs for a specific playlist by looking at its song's content.

- **Hybrid Approaches:** A hybrid approach in recommendation systems combines multiple recommendation techniques to improve accuracy and overcome the limitations of individual methods. It typically blends content-based and collaborative filtering. In this research, we will focus on the combination of Content-base, Collaborative Filtering, and Natural Language Processing.

For training models, I will utilize the concept of K-Means Clusterings for Content-base and Alternating Least Square (ALS) in implicit Python library for my Collaborative Filtering system.

Chapter 3

DATA AND ANALYSIS

One of the most important steps in Data analysis is choosing the right data resource to analyze. Choosing the right data for analysis is critical since it ensures that your findings are correct and relevant. Good data enables precise decisions, saves time, and allows for meaningful comparisons. It also assures compliance with legal and ethical requirements. The dataset we choose to start our project is "Spotify Million Playlists Dataset". This dataset was released as part of the RecSys Challenge 2018, organized by Spotify. The key contributors are Noam Koenigstein, Uri Paquet, and the Spotify Research team.

About the dataset, it consits total 1,000,000 Spotify playlists with 6,634,6428 tracks and 2,262,292 artists. The variables at playlist level are: pid (playlist IDs), playlist name, collaborative, etc. The variables track level are: track's name, artist's name, track's uri, artist's uri, album name, etc.

In this chapter, we will focus on analyzing and explore the dataset insights, patterns, and trends. Additionaly, we will provide some of data visualizations about how each variables in the dataset behaves.

3.1 Data Cleaning

Data cleaning helps to make the data consistent and ready to analyze. However, this MPD dataset is completly cleaned, which means there is no missing values or duplicates. The only missing values are the 11 missing playlist's name, and 981,242 missing playlist's desciption. These 2 variable are not useful in our analyzing, so we don't have to clean them.

3.2 Exploratory Data Anlysis (EDA)

EDA was performed to understand the characteristics of the dataset and extract key patterns. The analysis included:

- Approxiamtely there are 66 tracks in each playlist. The biggest playlist contain 376 tracks, and the smallest playlist only contain 5 tracks
- The average number of artists contain in each playlist is 39. There is 1 playlist contains up to 238 artists, but there is 1 playlists only contains 3 artists in total.
- A right-skewed histogram of track numbers per playlist suggests most playlists are shorter, with a few playlists having a significantly larger number of tracks. This indicates that while most users create smaller playlists, a few generate much larger ones.

Here are some Data Visualization:

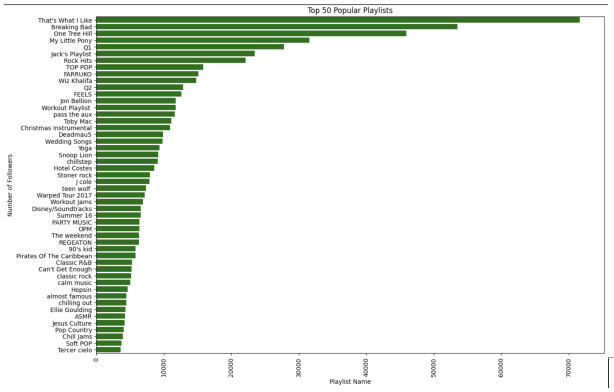


Figure 3.1: Top 50 Popular Playlists

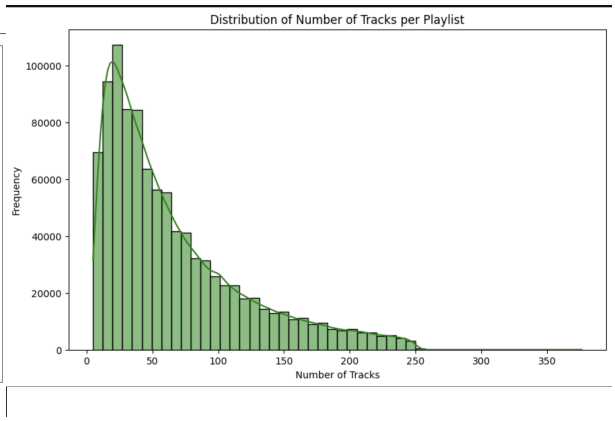


Figure 3.2: Distribution Of Number Of Tracks Per Playlist

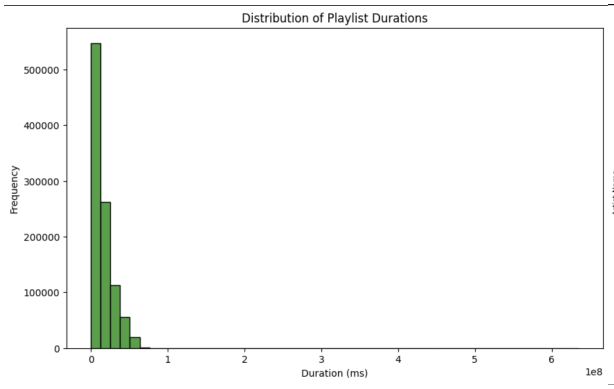


Figure 3.3: Distribution Of Playlist Durations

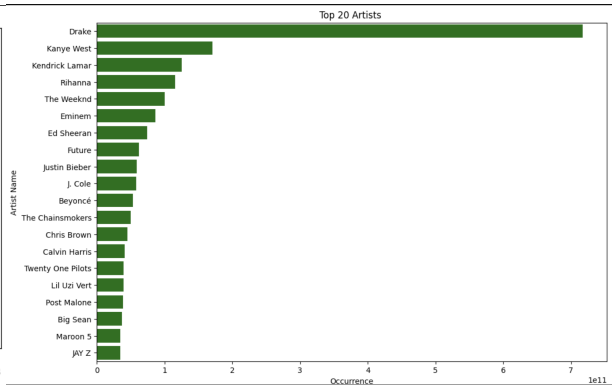


Figure 3.4: Top 20 Popular Artists

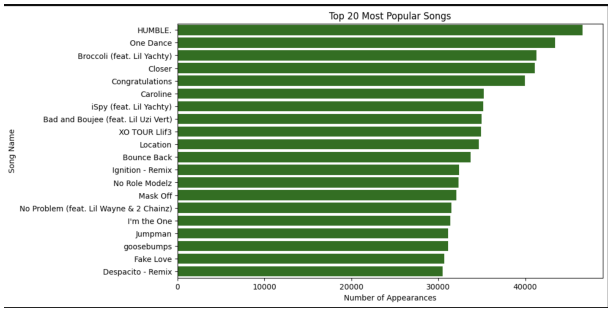


Figure 3.5: Top 50 Popular Playlists

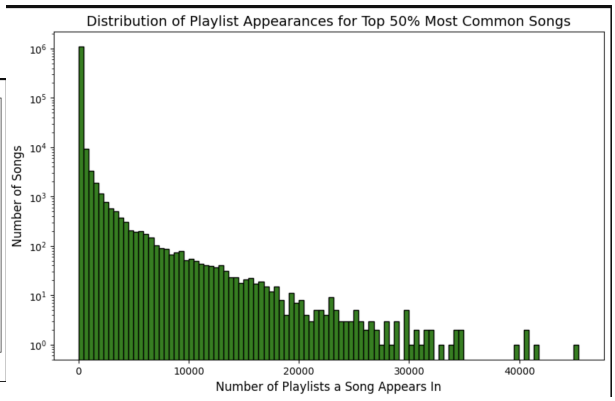


Figure 3.6: Distribution Of Number Of Tracks Per Playlist

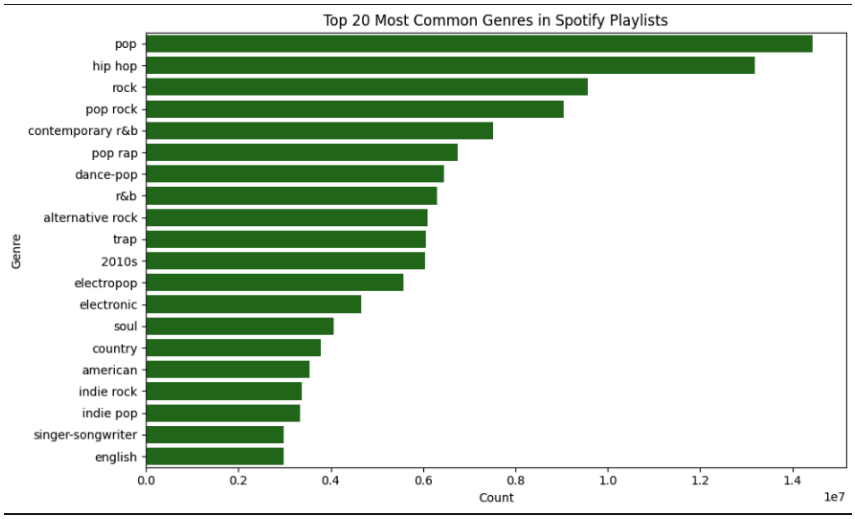


Figure 3.7: Top 20 Popular Genres In Spotify Playlists

We could see that most of Top popular artists, songs, or playlists are all about Pop, Hiphop, Roc,.... These genres are remain popular because of their catchy melodies, broad appeal, and emotional connection. These genres blend various styles, making them versatile and relatable. Their cultural influence, global reach, and constant evolution ensure they stay relevant and resonate with diverse audiences worldwide.

Chapter 4

RECOMMENDER SYSTEM

4.1 Content-Based Recommender System

A content-based recommender system analyzes the properties of items (like songs) with which a user has engaged, such as genre, artist, and audio feature. It then recommends further things with comparable characteristics.

In this research, we will focus on the algorithm which analyzes information from user's listening history—such as genres, lyrics to recommend songs with similar attributes. For example, if a playlist mostly contains Hiphop, Trap, Rap songs, the recommender system will predict songs which this playlists likely to listen from another playlists. Furthermore, this Content-based system also focus on the lyrics of the songs, by using NLP (Natural Language Processing), to create a system which predict songs with similar contents.

4.1.1 System 2: Cosine Similarity On Percentage Of Genres

This method recommends songs based on a playlist's genre distribution. Instead of doing a basic genre match, it computes the cosine similarity between the percentage distribution of genres in a playlist and the genre distribution of artists. The main idea of this system is to take a playlists and calculate the percentage of each genres to that playlists, and calculate the percentage of each genres to a specific artists.

The Distribution of Genres in a playlist is calculated by using this formula:

$$\frac{\text{Number of genre1}}{\text{Total numer of genres in a playlist}} \times 100$$

The Distribution of Genres in a playlist is calculated by using this formula:

$$\frac{\text{Number of genre1}}{\text{Total number of genres of an artist}} \times 100$$

For example: We need to create a list which is a intersection of top 100 popular genres amongs playlists and top 100 popular genres among artists.

Here is 53 popular genres.
alternative metal', 'disco', 'indie folk', 'punk', 'electronic', 'blues rock', 'british', 'indie pop', 'indie rock', 'house', 'synth-pop', 'jazz', 'rap', 'hip hop', 'edm', 'country', 'metal', 'reggae', 'latin pop', 'latin', 'usa', 'trap', 'blues', 'alternative', 'contemporary r&b', 'punk rock', 'hard rock', 'art rock', 'pop', 'folk rock', 'pop rap', 'funk', 'rock and indie', 'english', 'folk', 'dance', 'dream pop', 'soul', 'dance-pop', 'rock', 'singer-songwriter', 'electropop', 'r&b', 'american', 'indie', 'pop rock', 'alternative rock', 'synthpop', 'psychedelic rock', '2008 universal fire victim', 'uk', 'new wave', 'pop punk'

Here is the list of genres and their number of occurences in playlist Brasileiras (110):
'mpb': 24, 'bossa nova': 15, 'brazilian': 15, 'latin': 13, 'singer-songwriter': 6, 'pop': 5, 'jazz': 5.

Here is an exmaple of artist genres:
J Balvin: ['hip hop', 'latin', 'latin pop', 'pop', 'reggaeton']

The total genres in plauylist Brasileiras (110) is 163, so we can compute the dictribution of genres for each genres in common genres list, if the genres not present in this playlist, set it as 0. For example:

$$\frac{\text{Pop} = 5}{163} \times 100 = 3.07\%$$

, and so one for the rest opf the genre in common genres lists.
J Balvin has total 5 genres, now let's calcuate the percentage of genres in comon genres list to J Balvin's genre lists. For example:

$$\frac{\text{alternative metal} = 0}{5} \times 100 = 0$$

$$\frac{\text{hiphop} = 1}{5} \times 100 = 20\%$$

, and so on for the rest of the genres in the common genres list.

4.1.2 System 3: K-Means Clustering On Percentage Of Genres

This system using the same calcuate from the System 2, which will calculate the **Distribution of Genres** among playlists and artists. However, to extend the options for recommending songs, we will train the model of K-Means Clustering. the reason is because which System 2, we always get the songs from the same top 5 similar artists. We need more artists who are also similar not just the same 5 of them. In other word, the model groups artists who has the same ”vibe” into clusters. Given a playlist, the system chooses the closest cluters (group of artists has similar ”vibe” to that playlist), then randomly choose 5 artists in that clusters, and recommend random songs from those 5 artists.

To determine the best number of cluster, we use the **Elbow Method** is a technique used in clustering, specifically in K-Means clustering, to determine the optimal number of clusters (k) for the data. Let's check the graph below:

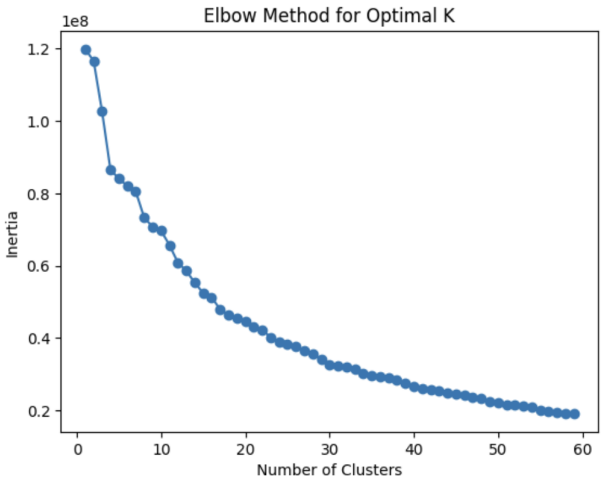
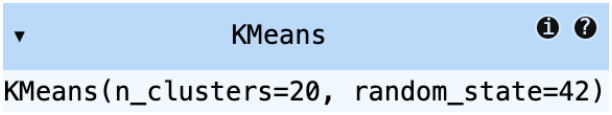


Figure 4.1: Elbow Method For Optimal K Graph

Although the graph doesn't clearly show the **Elbow** point, but we can see there is a dramatically decreasing on x-axis from 0 to 20, then the points start decreases slightly from $x = 20$. Therefore, the best number of clusters will be 20. Since we know what would be the number of clusters, let train the **K-Means Clustering** model.



The model take the input of artist's vector (artist's genres distribution), then divides 50,647 artists 20 groups.

4.2 Collaborative-Filtering Recommender System

Collaborative filtering is a method used in recommender systems that makes predictions based on past user behavior, such as interactions, ratings, or preferences. It assumes that if users have agreed on certain items in the past, they will likely agree on similar items in the future. There are two main types of collaborative filtering:

- User-based collaborative filtering: Recommends songs based on what songs does similar playlists contain.
- Item-based collaborative filtering: Recommends songs similar to those the playlist has liked in the past.

We will be only focus on User-based collaborative filtering which will analyze over a million playlists and their interaction with songs. To build a collaborative filtering system, we first create an interaction dataset between playlists and songs, and then convert it to a sparse matrix. Here is what the interaction sparse matrix look like.

4.2.1 Train-Test Split

For the implicit rating like interaction between playlists and songs, we can't just use a normal train/test split for recommender systems because it might leave out important interactions from individual users, making it hard to test the model's ability to recommend new items. Leave-One-Out (LOO) ensures each user's preferences are properly tested by predicting only one missing interaction at a time, simulating real-world recommendations.

The ideas of LOO techniquies is that we will be iterate over each playlist and its song (go through its rows in the Ultized Sparse Interaction Matrix), then we will random choose some of the interaction and hide them (set them 1 as 0). Note that, we can set the number of value that

being left out is 20%. here is the example of how it look.

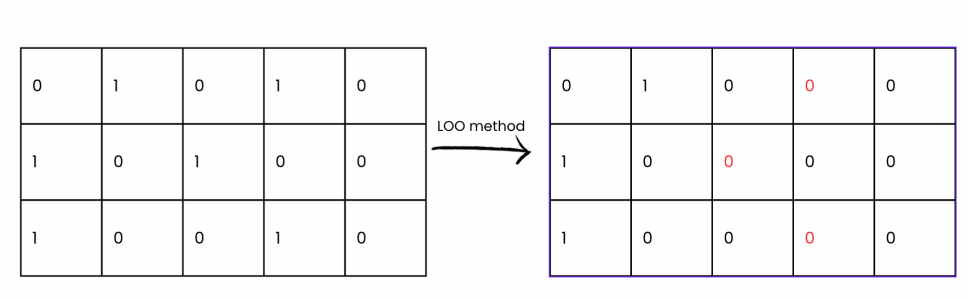


Figure 4.2: Leave-One-Out Method

4.2.2 Alternating Least Square Model

The ALS (Alternating Least Squares) model is a popular algorithm used for collaborative filtering in recommendation systems. Its goal is to predict user-item interactions, such as ratings or clicks, by learning the relationships between users and items. We will use the trai set we just created using LOO method to fit into ALS model, here is how the model trained.

1. Matrix factorization

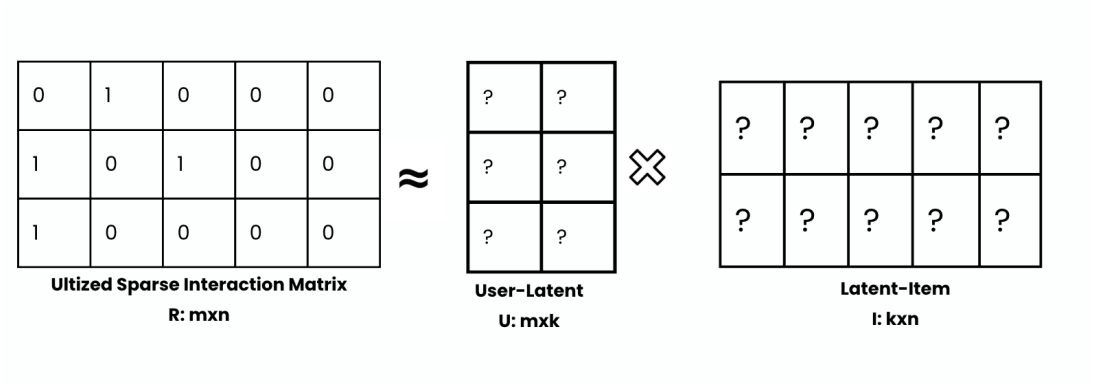


Figure 4.3: Matrix Factorization

The goal of matrix factorization method is to separate the utility matrix into the user-latent matrix and the latent-item matrix, such that

$$R \sim U \times I$$

In other words, we will initilize the values for the original User-Latent and Latent-item Matrix. Then, we will optimized so that we find the best 2 matrices. Here is the formular of User-Latent

and Latent-Item matrix optimization. The main goal is to minimize the Optimization Objective fucntion:

$$\mathscr{L}(U,V) = \sum_{(i,j)\in\mathscr{K}} \left(R_{ij} - U_i \cdot V_j^T\right)^2 + \lambda \left(\|U\|^2 + \|V\|^2\right)$$

Where:

- $\mathscr{K}$ is the set of indices for the observed entries in R.
- R_{ij} is the observed rating (or interaction) for user i and item j.
- $U_i \cdot V_j^T$ is the predicted rating based on the latent features of user i and item j.
- λ is the regularization parameter.

We continue updating the U and I matrix by using these formula:

$$U_i \leftarrow U_i + \eta \left[2 \sum_{j\in\mathscr{K}_i} (R_{ij} - U_i \cdot V_j^T)V_j + \lambda U_i\right]$$

Where:

- η is the learning rate.
- $\mathscr{K}_i$ is the set of items that user i has interacted with.
- V_j is the latent feature vector for item j.

$$V_j \leftarrow V_j + \eta \left[2 \sum_{i\in\mathscr{K}_j} (R_{ij} - U_i \cdot V_j^T)U_i + \lambda V_j\right]$$

Where:

- $\mathscr{K}_j$ is the set of users that have interacted with item j.
- U_i is the latent feature vector for user i.

The matrix factorization will stop after n times of updating U and I method and the Objective Fucction being minimize and converged. The output of the model will be the predicting interation value matrix, for example:

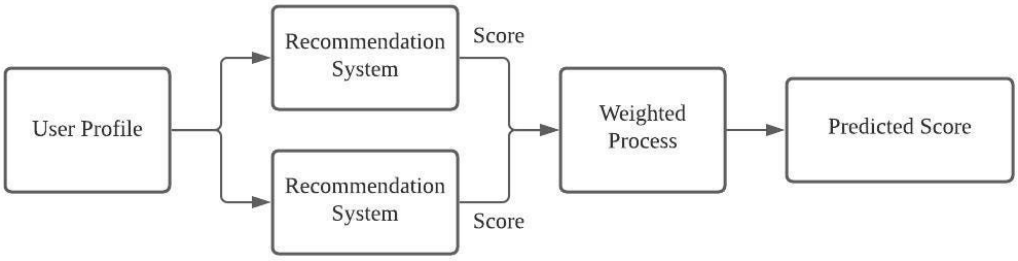
In the library of implicit.AlternatingLeastSquare, there is a method called **recommend()** which will exact the row correspond to the given playlist, then it sorts the value of interactions, and give out top N highest scores. The scores will be corresponding to each Track's ID.

0.8	0.2	0.4	0.6	0.1
0.3	0.5	0.7	0.04	0.2
0.4	0.67	0.76	0.3	0.5

Figure 4.4: Predicted Interaction Matrix

4.3 Hybrid Recommender System

In this approach, we will apply the idea of Weighted Hybrid Recommender System. A **weighted hybrid recommender system** combines multiple recommendation methods (e.g., content-based filtering (CBF) and collaborative filtering (CF)) by assigning a weight to each method's score. The final score is computed as a weighted sum of both scores.



However, since our Content-Based and Collaborative Filtering systems have different way to calcuate the predicted score. While Content-Based focuses on calcuate the Cosine Similarity scores, the Collaborative Filtering system focuses on minizing the rating scores by using Matrix factorization technique. To make sure both of the scores are on the same scale, so that there may dominate the other, leading to biased recommendations. One of the popular scaling method is **Min-Max Scaling**, which is calcuated by:

$$X_{\text{normalized}} = \frac{X - X_{\text{min}}}{X_{\text{max}} - X_{\text{min}}}$$

By normalizing score, we make sure that both of the score predicted bby both system are in the same scale, which is [0,1]. From this, the weighted are assign to both score. Here is the formula

of final score after assigning weighted.

$$S_{\text{final}} = w_{\text{CBF}} \times S_{\text{CBF}} + (1 - w_{\text{CF}}) \times S_{\text{CF}}$$

- S_{final}: The final score of a track based on the weighted combination of content-based filtering (CBF) and collaborative filtering (CF) scores.
- w_{CBF}: The weight assigned to the content-based filtering score.
- S_{CBF}: The score generated from content-based filtering.
- w_{CF}: The weight assigned to the collaborative filtering score.
- S_{CF}: The score generated from collaborative filtering.

The algorithm then sort those final score in the descending order, and select the top N highest scores (N is the number of desired recommended songs).

Chapter 5

RESULTS AND DISCUSSION

5.1 Limitations

While the recommender systems performs well on predicting songs, there are still some limitations. First, the system can't recommend songs to a new user (new playlist with no songs added yet). Secondly, since we don't have access to the Spotify API (policy changed), we can't get audio's feature, which will be a good variable for Content-based recommender system because it provide a deep meaning of a song (sad, upbeat, danceability,...). Thirdly, since this is a big dataset, it's very hard to handle all the genres, so we need to do a dimensional reduction on top popular genres vector, which limits the diversity of genres to recommend. Furthermore, since we recommend songs from this MPD dataset, there would be no new artists because this dataset comes from 2018, so that it contain contains old songs from old artists. Finally, there are a lot of playlists which contain very few songs, making it hard to find similar playlists or songs.

5.2 Application

In this application, I will use my own playlist on Spotify. Like what I states in the Limitations section, my playlist contains some new songs after the year 2018, so the recommender system might not very accurate. Here is the link to my Spotify playlist: This playlist is mostly Hip Hop, Trap, Rap, Pop,... and the popular artists would be Kendrick Lamar, Future, Travis Scott. Here are the songs each system recommended to my playlist:

Chapter 6

CONCLUSION

In this study, we created a Spotify playlist recommender system with a hybrid method that incorporated content-based and collaborative filtering. The method recommended songs based on genre distributions in playlists and cosine similarity. This methodology produced a balanced solution by integrating the strengths of both methods: content-based filtering identified songs with comparable genres, while collaborative filtering took into consideration user behavior and preferences.

Despite its success, the system encountered issues such as the cold start problem, in which new users or playlists with little interaction data received fewer accurate recommendations. Data sparsity was another concern, as many users only interacted with a tiny fraction of music, lowering the efficiency of collaborative filtering. Furthermore, the system demonstrated popularity bias, with popular songs dominating the recommendations, limiting diversity.

For the future goal and improvement, this study want to extend the diversity of the dataset by getting lyrics dataset, audio's feature dataset, and more user's interaction with songs like (play counts, likes, clicks, ...) to boost the accuracy of the recommender system. Additionally, incorporating methods to enhance diversity and reduce bias would provide more varied and personalized recommendations. Overall, this research demonstrate a potential system for any entertainment platform like Spotify, Apple Music, Netflix, etc.

Bibliography

[1] hen, C.W., Lamere, P., Schedl, M., and Zamani, H. “Recsys Challenge 2018: Automatic Music Playlist Continuation.” Proceedings of the 12th ACM Conference on Recommender Systems (RecSys ’18), 2018 [Online]. Available at: https://doi.org/10.1145/3240323.3240354

[2] entao Xu, Qianqian Xie, Shuo Yang, Jiangxia Cao, Shuchao Pang, “Enhancing Content-based Recommendation via Large Language Model” Cornell University, 2024. [Online]. Available at: https://arxiv.org/abs/2404.00236

[3] iangnan He, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu, Tat-Seng Chua, “Neural Collaborative Filtering” Cornell University, 2017. [Online]. Available at: https://arxiv.org/abs/2404.00236

[4] onule, Avinash and Jagtap, Hemant and Mendhe, Vikas, ”Weighted Hybrid Recommendation System” INTERNATIONAL JOURNAL OF RESEARCH AND ANALYTICAL REVIEWS, Available at: https://www.researchgate.net/publication/378846446_Weighted_Hybrid_Recommendation_System