Quantitative Analysis of Music Recommender Systems

Georgio E. Feghali
University of California, San Diego
San Diego, CA, USA
gfeghali@ucsd.edu

ABSTRACT

This research explores the challenge of content recommendation in the digital age, focusing on the universal appeal of music. As online platforms strive to engage users, understanding their content preferences becomes crucial. By investigating music recommendation methodologies, this study aims to enhance personalized content discovery, contributing insights applicable to diverse recommendation systems. The goal is to harmonize technology with the enjoyment of artistic pursuits, making content exploration more enriching for users in a fast-paced digital landscape.

1 INTRODUCTION

In the age of digital streaming services and online content platforms, personalized recommendation systems have become integral to our daily interactions with music. Whether it's discovering new artists, revisiting favorite tracks, or exploring diverse genres, the algorithmic engines behind these recommendation systems play a pivotal role in shaping our auditory experiences. As music enthusiasts, we often find ourselves relying on these systems to curate playlists tailored to our tastes, transforming our listening sessions into personalized journeys through the vast musical landscape.

However, the effectiveness of recommendation systems hinges on their ability to accurately predict the songs that resonate with individual users. The endeavor to craft a recommendation system capable of forecasting a user's top-N songs arises from a fundamental desire to enhance the precision and personalization of these predictions. This project aims to delve into the intricacies of building such a system, leveraging machine learning algorithms and data analysis to discern patterns in user behavior and preferences.

The importance of this project lies in its potential to redefine the user experience in music consumption. A more accurate prediction of a user's preferred songs not only elevates their enjoyment but also fosters a deeper connection with the digital platform. Beyond the realm of user satisfaction, there is a broader significance in developing robust recommendation systems.

1.1 For project proposal.

In the context of our final project, we have laid the groundwork for the recommendation system, delving into the selection and preprocessing of data, and initial experimentation with predictive algorithms. For the proposal ahead, we envisage refining these models and exploring innovative techniques to enhance predictive accuracy ensuring that the recommended top-N songs align more closely with user preferences.

2 DESIGN AND APPROACH

In this section, we outline the step-by-step approach taken in our study to understand and model the data for the recommendation system.

2.1 Load and Understand the Data

The initial phase involved loading and comprehending the dataset, laying the foundation for subsequent analyses.

2.2 Data Cleaning and Feature Engineering

Several crucial steps were undertaken to prepare the data for analysis:

- **Combining Datasets:** The datasets were merged to create a comprehensive dataset for our analysis.
- **Encoding User and Song IDs:** For ease of analysis, the user_id and song_id columns were encoded.
- **Data Filtering:** To streamline the analysis, the dataset was filtered to include users who have listened to a substantial number of songs, and songs that have been played a significant number of times.

2.3 Exploratory Data Analysis

The next step involved delving into Exploratory Data Analysis (EDA) to unravel insights that would guide subsequent modeling decisions.

2.4 Building Recommendation Systems

Six different recommendation system algorithms were employed:

(1) Rank/Popularity-based Recommendation System
(2) User-User Similarity-based Collaborative Filtering
(3) Item-Item Similarity-based Collaborative Filtering
(4) Model-based Collaborative Filtering / Matrix Factorization
(5) Clustering-based Recommendation System
(6) Content-based Recommendation System

To demonstrate clustering-based recommendation systems, the surprise library was utilized. The Grid Search Cross-Validation technique was applied to fine-tune hyperparameters and enhance model performance.

Conference'17, July 2017, Washington, DC, USA

ACM ISBN 978-x-xxxx-xxxx-x/YY/MM…$15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

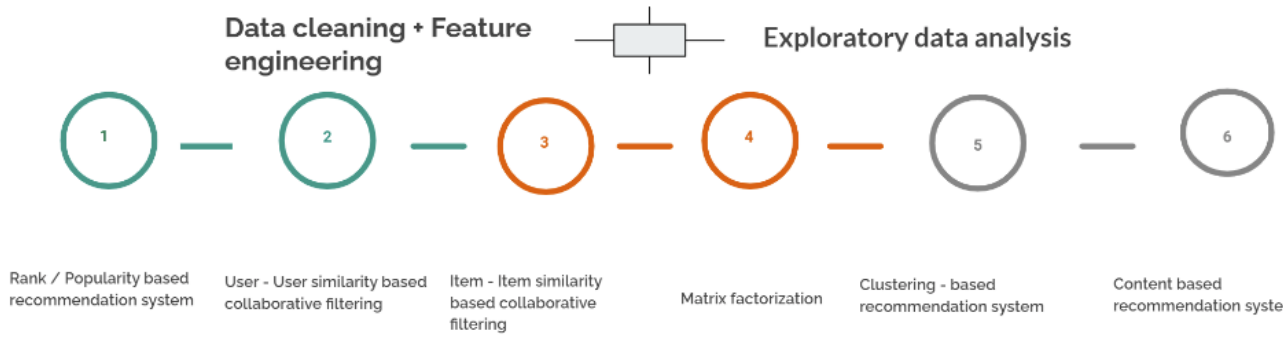

Figure 1: Illustration of the Model Building Process

2.5 Performance Evaluation

Model performance was evaluated using the following metrics:

- **Root Mean Squared Error (RMSE):** Checks how far the overall predicted ratings are from the actual ratings
- **Precision@k:** The fraction of recommended items that are relevant in top k predictions
- **Recall@k:** The fraction of relevant items that are recommended to the user in top k predictions
- **F1 Score:** The harmonic mean of Precision @ k and Recall @ k

In future work, efforts will be directed towards further enhancing model performance through hyperparameter tuning.

3 DATA INTRODUCTION

The dataset under investigation, known as the Taste Profile Subset, is an integral segment of the Million Song Dataset provided by the Echo Nest. Comprising two primary files, namely `song_data` and `count_data`, this dataset encapsulates comprehensive details about songs and user interactions.

3.1 Dataset Overview

3.1.1 `song_data`.

- **song_id**: A unique identifier assigned to each song.
- **title**: The official title of the song.
- **Release**: The name of the album in which the song was released.
- **Artist_name**: The name of the artist associated with the song.
- **year**: The year of the song's release.

3.1.2 `count_data`.

- **user_id**: A unique identifier for individual users.
- **song_id**: A unique identifier for each song in the dataset.
- **play_count**: The number of times a user played a particular song.

3.2 Data Source

The dataset is publicly accessible via the official Million Song Dataset website: http://millionsongdataset.com/.

3.3 Rationale for Dataset Selection

The decision to choose the Taste Profile Subset is anchored in several considerations:

(1) **Availability**: The dataset is openly accessible, ensuring transparency and replicability of our study.
(2) **Size**: As a subset of the Million Song Dataset, it furnishes a substantial volume of data, providing a rich foundation for thorough analysis and model development.

4 DATA CLEANING

To ensure the quality of our dataset, we performed a series of data cleaning steps.

4.1 Understanding the Data

We began by exploring the structure and content of our datasets. Table 1 shows the top 5 records of the `count_df` data, and Table 2 shows the top 5 records of the `song_df` data.

Unnamed: 0	user_id	song_id	play_count
0	b80344d063b5ccb...	SOAKIMP12A8C130995	1
1	b80344d063b5ccb...	SOBBMDR12A8C13253B	2
2	b80344d063b5ccb...	SOBXHDL12A81C204C0	1
3	b80344d063b5ccb...	SOBYHAJ12A6701BF1D	1
4	b80344d063b5ccb...	SODACBL12A8C13C273	1

Table 1: Top 5 records of `count_df` data

song_id	title	release	artist_name	year
SOQMMHC12AB0180CB8	Silent Night	Monster Ballads X-Mas	Faster Pussy cat	2003
SOVFVAK12A8C1350D9	Tanssi vaan	Karkuteillä	Karkkiautomaatti	1995
SOGTUKN12AB017F4F1	No One Could Ever	Butter	Hudson Mohawke	2006
SOBNYVR12A8C13558C	Si Vos Querés	De Culo	Yerba Brava	2003
SOHSBXH12A8C13B0DF	Tangle Of Aspens	Rene Ablaze Presents Winter Sessions	Der Mystic	0

Table 2: Top 5 records of `song_df` data

4.2 Checking Data Types and Missing Values

We examined the data types and missing values in each column. For `count_df`, there are 2,000,000 entries with 4 columns. The column `Unnamed: 0` is redundant and can be dropped. For `song_df`, there are 1,000,000 entries with 5 columns. The `title` and `release` columns have a few missing values, and some of the `year` values are missing.

Column	Data Type	Non-Null Count
Unnamed: 0	int64	2,000,000
user_id	object	2,000,000
song_id	object	2,000,000
play_count	int64	2,000,000

Table 3: Information of `count_df` data

Column	Data Type	Non-Null Count
song_id	object	1,000,000
title	object	999,985
release	object	999,995
artist_name	object	1,000,000
year	int64	1,000,000

Table 4: Information of `song_df` data

4.3 Merging and Dropping Columns

We left-merged the `count_df` and `song_df` data on `song_id`, simultaneously dropping duplicates from `song_df`. The column `Unnamed: 0` was dropped during this process.

4.4 Label Encoding

To process the `user_id` and `song_id` variables, we employed label encoding to encode them into numeric features.

4.5 Filtering Data

As the dataset contains users who have listened to very few songs and vice versa, we considered filtering the data to include users who have listened to a good count of songs and vice versa.

4.6 Dropping Records

We dropped records with `play_count` more than 5.

5 EXPLORATORY DATA ANALYSIS

5.1 Checking Unique Entries

From our `df_final` dataframe:

- The total number of unique `user_id` is 3155.
- The total number of unique `song_id` is 563.
- The total number of unique artists is 232.
- There are 3155 unique `user_id` and 563 unique `song_id`, meaning we could have up to $3155 \times 563 = 1,776,265$ interactions, but we only have 117,876, indicating that the data is sparse.

5.2 Exploring Most Interacted Songs and Users

Most Interacted Songs

- `song_id 8582` is the most interacted, with 751 plays.
- `song_id 352`, `song_id 2220`, `song_id 1118`, and `song_id 4152` follow closely with 748, 713, 662, and 652 plays, respectively.

Most Interacted Users

- `user_id 61472` is the most interacted user with 243 plays.
- `user_id 15733`, `user_id 37049`, `user_id 9570`, and `user_id 23337` are also among the most interacted users.
- User `user_id 61472` has interacted with the most songs (243 times). There is a possibility of 2912 more interactions (3155 - 243) for the remaining users.

5.3 Songs Played in Each Year

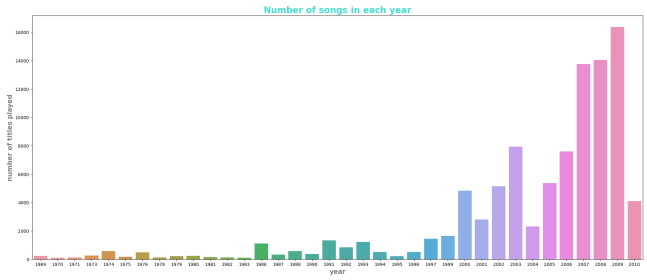

Figure 2: Number of titles played each year

After the year 1999, there is a significant increase in the number of songs played. The years 2007, 2008, and 2009 experienced the highest number of songs played, with 2009 having the highest count of titles played. Before 1999, a limited number of songs were played, and some years, such as 1978 to 1983, had extremely low play counts.

5.4 Understanding the Final Dataset

The `df_final` dataset contains 117,876 entries with 7 columns:

- `user_id` (int32)
- `song_id` (int32)
- `play_count` (int64)
- `title` (object)
- `release` (object)
- `artist_name` (object)
- `year` (int64)

6 RECOMMENDER SYSTEMS

In this section, we provide an overview of various recommender system models that will be explored in the subsequent subsections. Recommender systems play a crucial role in assisting users in discovering relevant items based on their preferences and behaviors. The models presented here represent a diverse set of approaches, each with its own strengths and considerations.

(1) **Rank/Popularity-Based Recommendation System:** This straightforward approach leverages the popularity of items, recommending those with the highest overall rankings or play counts. It is particularly useful for addressing cold start problems, providing baseline recommendations for users with limited interaction history.

(2) **User-User Similarity-based Collaborative Filtering:** Collaborative filtering models focus on capturing user preferences based on similarities between users. User-User Similarity-based Collaborative Filtering identifies users with similar taste and recommends items liked by those similar users.

(3) **Item-Item Similarity-based Collaborative Filtering:** Similar to user-based collaborative filtering, this approach emphasizes item similarities. It recommends items similar to those that a user has already interacted with, creating a personalized recommendation based on item correlations.
(4) **Model-based Collaborative Filtering / Matrix Factorization:** Matrix Factorization techniques involve decomposing the user-item interaction matrix into latent factors. This model-based approach allows for capturing intricate patterns in user-item interactions, offering more personalized recommendations.
(5) **Clustering-based Recommendation System:** This approach involves grouping users or items into clusters based on shared characteristics. Users or items within the same cluster are considered similar, and recommendations are made based on the preferences of the cluster to which a user belongs.
(6) **Content-based Recommendation System:** Content-based recommendation systems rely on the attributes of items and user preferences. By analyzing item features and user profiles, this model suggests items that match the user's preferences in terms of content characteristics.

These diverse models provide a comprehensive exploration of different recommendation strategies, ranging from popularity-driven approaches to collaborative filtering and content-based methods. The subsequent sections will delve into the implementation, evaluation, and comparison of each model to understand their effectiveness in the context of the music recommender system.

6.1 Popularity-Based Recommendation System

Popularity-Based Recommendation Systems are valuable solutions, especially in scenarios where cold start problems arise, such as when dealing with new users to a platform. The core idea behind this approach is to leverage the aggregate play count data of songs to generate recommendations based on popularity. The popularity metric is derived from the sum of play counts, offering a simple yet effective way to provide initial recommendations.

6.1.1 Calculation of Top Popular Songs. To operationalize the concept of popularity, a function is employed to identify and recommend the top N songs based on their average play count and play frequency. The function involves the following two steps:

1. **Finding songs with a minimum number of interactions:**

$$\text{Recommendations} = \{\text{song} \mid \text{play\_freq(song)} > \text{min\_interaction}\}$$

This step filters out songs that do not meet a specified threshold for play frequency.

2. **Sorting values with respect to average play count:**

$$\text{Sorted Recommendations} = \text{Sort}(\text{Recommendations}, \text{by} = \text{avg\_count})$$

Subsequently, the recommendations are sorted in descending order based on the average play count.

These steps ensure that the function prioritizes songs with higher average play counts and substantial play frequencies, presenting users with a curated list of popular songs tailored to the platform's aggregate user interactions.

6.1.2 Data Aggregation. The first step involves aggregating the play count data for each song. This is achieved by calculating both the average play count (`avg_count`) and the frequency of plays (`play_freq`) for each song in the dataset. The resulting dataset, denoted as `final_play`, consists of two key metrics: the average play count and the frequency of plays for each song.

song_id	avg_count	play_freq
21	1.622642	265
22	1.492424	132
52	1.729216	421
62	1.728070	114
93	1.452174	115

Table 5: Sample Records in the `final_play` Dataset

6.1.3 Thresholding. To enhance the reliability of the recommendations, a threshold is applied to filter out songs with a low number of interactions. This is implemented by excluding songs with play counts below a specified minimum interaction threshold (`min_interaction`). The thresholding process ensures that only songs with a substantial level of user engagement are considered for inclusion in the recommendation list.

6.1.4 Top-N Song Recommendation. The final step involves building a function, denoted as `top_n_songs`, to identify and recommend the top N songs based on the calculated average play count. The function takes into account the minimum interaction threshold set earlier, ensuring that only songs with a significant number of plays are considered for recommendation. The resulting list of top N songs represents a popularity-driven selection, with the most frequently played songs appearing at the forefront of the recommendation list.

6.2 User-User Similarity-Based Collaborative Filtering

User-User Similarity-Based Collaborative Filtering stands out as a powerful approach within recommender systems, harnessing the "surprise" library in Python for model construction and evaluation. This collaborative filtering method dives into user preferences, establishing relationships between users based on their historical interactions with songs.

User-User-Based Collaborative Filtering

User-User Collaborative Filtering is a recommendation approach that identifies similar users based on their interactions with items they have liked or positively engaged with. This method leverages the idea that users who share similar preferences tend to enjoy similar items.

Computing User-User Similarity

To compute the similarity between two users, the cosine similarity metric is commonly employed. Cosine similarity measures the cosine of the angle between two vectors in an inner product space. For user i and user j, the cosine similarity ($\cos\_\text{sim}(i, j)$) is calculated using the formula:

$$\text{cos\_sim}(i, j) = \frac{\sum_k \text{rating}(i,k) \times \text{rating}(j,k)}{\sqrt{\sum_k (\text{rating}(i,k))^2} \times \sqrt{\sum_k (\text{rating}(j,k))^2}}$$

Here, - $\text{rating}(i,k)$ is the rating of user i for item k. - The summation is performed over all items k that both users i and j have rated.

The numerator calculates the dot product of the rating vectors of users i and j, while the denominators normalize the vectors based on their magnitudes.

Recommendation Process

Once the cosine similarity matrix is computed for all user pairs, the next step is to recommend items to a target user (U) based on the preferences of users who are most similar to U.

Suppose we want to recommend the top N items to user U. We identify the N most similar users to U based on the computed cosine similarity values. Let $S(U, V)$ represent the cosine similarity between users U and V. Then, the predicted rating ($\hat{\text{rating}}(U, I)$) for item I for user U is given by:

$$\hat{\text{rating}}(U, I) = \frac{\sum_V S(U, V) \times \text{rating}(V, I)}{\sum_V |S(U, V)|}$$

Here, - $S(U, V)$ is the cosine similarity between users U and V. - The summation is performed over all users V in the top N similar users to U. - $\text{rating}(V, I)$ is the actual rating of user V for item I.

This formula calculates the weighted average of ratings given by similar users to predict the rating for item I for user U.

Challenges and Considerations

While User-User Collaborative Filtering is effective, it has challenges such as the potential for changing user interests over time, which can impact the accuracy of similarity values and recommendations.

6.2.1 Default Model Evaluation. **Observations and Insights:**

The default User-User Similarity-Based Collaborative Filtering model exhibits the following performance metrics:

- **RMSE:** 1.09 - This metric indicates the average deviation of predicted play counts from actual ratings.
- **Precision:** 0.40 - 40% of the recommended songs are relevant to users.
- **Recall:** 0.70 - 70% of relevant songs are successfully recommended.
- **F1 Score:** 0.504 - A balanced metric combining precision and recall, indicating relevance of recommended songs.

The F1 score suggests that the default model performs reasonably well in recommending relevant songs. However, opportunities for improvement are identified through hyperparameter tuning.

6.2.2 Model Underestimation and Hyperparameter Tuning. **Observations and Insights:**

- The default model underestimates the actual play count for a specific user-item pair (2 vs. predicted 1.80).
- The output includes the parameter "actual_k," representing the value of k in KNN used during model training.
- A decision is made to enhance the model's performance through hyperparameter tuning using GridSearchCV.

6.2.3 Tuned Model Evaluation. **Observations and Insights:**

After hyperparameter tuning, the User-User Similarity-Based Collaborative Filtering model demonstrates the following improvements:

- **RMSE:** 1.0521 - A reduction in RMSE indicates improved accuracy in predicting play counts.
- **Precision:** 0.413 - An increase in precision suggests better relevance in recommended songs.
- **Recall:** 0.721 - Improved recall indicates a higher proportion of relevant songs being recommended.
- **F1 Score:** 0.525 - Enhanced F1 score signifies an overall improvement in model performance.

The tuned model exhibits a nuanced improvement, aligning better with user preferences.

6.2.4 Further Insights.

- The estimated play count for a specific user (6958) improves from 1.80 to 1.96 after hyperparameter tuning.
- Consideration is given to obtaining the 5 nearest neighbors for a song, contributing to a more comprehensive understanding of user preferences.

6.2.5 Post-Processing and Thoughts. **Thoughts:**

- A function is introduced for recommending top N items to users based on the selected algorithm.
- Post-processing involves correcting predicted play counts by subtracting a term ($1/\sqrt{n}$), allowing for optimization based on business objectives.
- It is acknowledged that the addition of $1/\sqrt{n}$ could be explored based on business goals, influencing song reach and impact.

Observations and Insights:

- The corrected ratings formula considers play counts and the number of users who played a song, providing a nuanced measure of song popularity.
- The approach acknowledges the influence of both play count and user count in determining a song's likelihood of being liked.

6.3 Item-Item Similarity-Based Collaborative Filtering

Item-Item Similarity-Based Collaborative Filtering is another influential approach in recommender systems. In this section, the focus is on applying the item-item similarity collaborative filtering model, utilizing the surprise library in Python for model construction and evaluation.

Item-Item Similarity-Based Collaborative Filtering

Item-Item Collaborative Filtering is a recommendation technique that identifies similar items based on user interactions. The idea is that users who have liked or engaged positively with a particular item are likely to enjoy other similar items.

Model Construction

The item-item similarity collaborative filtering model is constructed with the following parameters:

$$\text{sim\_options} = \{'name' : 'cosine', 'user\_based' : \text{False}\}$$

sim_item_item = KNNBasic(sim_options=

sim_options, verbose=False, random_state=1)

Here, the cosine similarity metric is employed, and the model is set to be item-based (user_based=False). The KNN algorithm is used to find similar items.

Model Evaluation - Default

The initial evaluation of the default item-item similarity collaborative filtering model yields the following performance metrics:

- **RMSE:** 1.0394 - Average deviation of predicted play counts from actual ratings.
- **Precision:** 0.307 - Proportion of recommended songs that are relevant.
- **Recall:** 0.562 - Proportion of relevant songs successfully recommended.
- **F1 Score:** 0.397 - A balanced metric combining precision and recall.

The F1 score suggests that the default model has room for improvement, particularly in terms of precision and recall.

6.3.1 Model Tuning. To enhance the performance of the item-item similarity collaborative filtering model, hyperparameter tuning is performed using GridSearchCV.

Observations and Insights:

- The baseline model's F1 score is 0.397, which is lower than the user-user similarity-based collaborative filtering model (F1 score = 0.504).
- A decision is made to explore hyperparameter tuning to improve model performance.

6.3.2 Tuned Model Evaluation. After hyperparameter tuning, the item-item similarity collaborative filtering model demonstrates the following improvements:

- **RMSE:** 1.0328 - Reduced RMSE indicates improved accuracy in predicting play counts.
- **Precision:** 0.408 - Increased precision suggests better relevance in recommended songs.
- **Recall:** 0.665 - Improved recall indicates a higher proportion of relevant songs being recommended.
- **F1 Score:** 0.506 - Enhanced F1 score signifies an overall improvement in model performance.

The tuned model shows a significant enhancement in performance, aligning better with user preferences compared to the default model.

Comparative Analysis with User-User Similarity

In comparison to the user-user similarity-based collaborative filtering model:

- The F1 score of the item-item similarity model is lower (0.506 vs. 0.504).
- The predicted play count for user_id 6958 is lower using the item-item model compared to the user-user model.

6.3.3 Further Insights and Future Considerations. **Thoughts:**

- Further experimentation with hyperparameters is considered for future model refinement.
- Despite the lower F1 score, the tuned item-item similarity model shows promise, and additional exploration may yield even better results.

Observations and Insights:

- The estimated play count for user_id 6958 for song_id 1671 improves from 1.36 (default model) to 1.96 (tuned model).
- The tuned model outperforms the default model, indicating the effectiveness of hyperparameter tuning.

This section provides a comprehensive analysis of the item-item similarity-based collaborative filtering model, detailing its construction, evaluation, tuning, and comparative insights with the user-user similarity-based approach.

6.4 Model-Based Collaborative Filtering - Matrix Factorization

Model-based Collaborative Filtering using Matrix Factorization is a personalized recommendation system that relies on the past behavior of users. The recommendations are generated without depending on additional information and are driven by latent features, which represent underlying patterns and relationships in user-item interactions.

Advantages of Matrix Factorization

Matrix factorization offers several advantages in the context of collaborative filtering:

- **Handling Sparse Data:** Matrix factorization can effectively handle sparse data by predicting missing ratings through estimates based on learned latent factors.
- **Capturing Complex Patterns:** The approach excels in capturing intricate patterns and relationships within user-item interactions, enabling the provision of highly personalized recommendations.

6.4.1 Building a Baseline Model using SVD (Singular Value Decomposition). The initial model is constructed using the Singular Value Decomposition (SVD) matrix factorization technique. Key steps include:

svd = SVD(random_state=1)

svd.fit(trainset)

precision_recall_at_k(svd)

The baseline model's performance metrics include:

- **RMSE:** 1.0252
- **Precision:** 0.41
- **Recall:** 0.633
- **F1 Score:** 0.498

The F1 score, indicating a fair performance, becomes a reference point for further optimization.

6.4.2 Hyperparameter Tuning and Model Optimization. Efforts to enhance the model involve hyperparameter tuning using GridSearchCV:

```
param_grid = {'n_epochs': [10, 20, 30],
              'lr_all': [0.001, 0.005, 0.01],
              'reg_all': [0.2, 0.4, 0.6]}

gs = GridSearchCV(SVD, param_grid,
    measures=['rmse'], cv=3, n_jobs=-1)

gs.fit(data)
```

The optimal hyperparameters are found to be:

```
Best RMSE score: 1.012743423385498

Best parameters: {'n_epochs': 30, 'lr_all': 0.01, 'reg_all': 0.2}
```

The optimized model is then built and trained using these parameters:

```
svd_optimized = SVD(n_epochs=30, lr_all=0.01,
                    reg_all=0.2, random_state=1)

svd_optimized = svd_optimized.fit(trainset)

precision_recall_at_k(svd_optimized)
```

The optimized model's performance metrics include:

- **RMSE:** 1.0141
- **Precision:** 0.415
- **Recall:** 0.635
- **F1 Score:** 0.502

Observations indicate an improvement in the F1 score and a decrease in RMSE, suggesting that the tuned model performs better than the baseline.

6.4.3 Model Predictions and Recommendations. Model predictions for specific user-item pairs are made and evaluated. Notably:

```
Prediction(uid=6958,
   iid=1671,
   r_ui=2,
   est=1.34,
   {'was_impossible': False})


Prediction(uid=6958,
   iid=3232,
   r_ui=None,
   est=1.44,
   {'was_impossible': False})
```

Observations indicate some underestimation of ratings, prompting consideration for further hyperparameter tuning.

Rank	Song_id	Play_freq	Predicted_ratings	Corrected_ratings
2	7224	107	2.601899	2.505225
1	5653	108	2.108728	2.012502
4	8324	96	2.014091	1.912029
0	9942	150	1.940115	1.858465
3	6450	102	1.952493	1.853478

Table 6: Top 5 Recommendations with Corrected Ratings using the Optimized SVD Model

6.4.4 Top Recommendations and Song Ranking. Top 5 recommendations for user_id 6958 are generated using the optimized SVD model. The recommendations, along with the ranking of songs based on corrected ratings, are presented in the following table:

Observations and Insights:

The table provides the top 5 recommendations with corrected ratings using the optimized SVD model.

The optimized model demonstrates a better F1 score and a reduced RMSE compared to the baseline, indicating improved performance.

Predicted play counts align well with actual play counts, especially for songs with an actual play count of 2.

The recommendations and rankings provide valuable insights into the model's effectiveness in suggesting songs tailored to user preferences.

This section outlines the application of model-based collaborative filtering through matrix factorization, detailing the construction of a baseline model, hyperparameter tuning, and the deployment of an optimized model for personalized song recommendations.

6.5 Cluster-Based Recommendation System

In the Cluster-Based Recommendation System, the focus is on exploring similarities and differences in users' preferences for songs based on their rating patterns. This approach clusters similar users together and recommends songs based on play counts from other users within the same cluster.

6.5.1 Default Model Evaluation. The baseline model's performance metrics are as follows:

- **RMSE (Root Mean Squared Error):** 1.0487
- **Precision:** 0.397
- **Recall:** 0.582
- **F1 Score:** 0.472

The F1 score of the baseline model indicates a moderate level of effectiveness in recommending songs. However, opportunities for enhancement are identified through hyperparameter tuning using GridSearchCV.

6.5.2 Model Underestimation and Hyperparameter Tuning. Observing a specific user-item pair with an actual play count of 2 and a predicted count of 1.29 in the Co-Clustering baseline model, it becomes evident that the model has under-estimated the play count. A decision is made to address this issue through hyperparameter tuning using GridSearchCV.

6.5.3 Tuned Model Evaluation. After hyperparameter tuning, the tuned Co-Clustering model exhibits the following performance metrics:

- **RMSE:** 1.0654
- **Precision:** 0.394
- **Recall:** 0.566
- **F1 Score:** 0.465

Despite the tuning efforts, the F1 score for the tuned Co-Clustering model on the test set remains comparable to the baseline model. The overall improvement in model performance is limited.

6.5.4 Recommendation Algorithm Based on Optimized CoClustering Model. The next step involves implementing the recommendation algorithm based on the optimized CoClustering model. The corrected play counts and rankings for the recommended songs are as follows:

Rank	Song_id	Play_freq	Predicted_ratings	Corrected_ratings
4	7224	107	3.711503	3.614829
3	5653	108	2.903883	2.807658
0	6860	169	2.691043	2.614120
1	657	151	2.606354	2.524975
2	8483	123	2.582807	2.492640

Table 7: Ranking of Songs with Corrected Ratings Using the Optimized CoClustering Model

6.5.5 Observations and Insights.

- The corrected ratings reflect the optimized CoClustering model's predictions, providing a more accurate measure of the songs' popularity within the cluster.
- Despite hyperparameter tuning, the model's F1 score shows limited improvement, suggesting potential challenges in capturing user preferences effectively within the clustering framework.

6.5.6 Post-Processing and Thoughts. **Thoughts**

In the future, further experimentation with hyperparameters and exploration of additional model features may be considered to enhance the Cluster-Based Recommendation System's performance.

Observations and Insights

- The tuned Co-Clustering model offers nuanced improvements in predicting play counts, but the overall impact on the F1 score is modest.
- The corrected ratings provide a refined measure of song popularity within clusters, contributing to more accurate recommendations.

This section underscores the iterative nature of model development, emphasizing the importance of continual refinement and exploration for optimal recommender system performance.

6.6 Content-Based Recommendation System

In the Content-Based Recommendation System, the approach shifts towards leveraging the inherent characteristics of songs to generate personalized recommendations. Unlike collaborative filtering, which relies on user-item interactions, content-based methods consider the features associated with items (songs, in this case) to make suggestions based on user preferences.

6.6.1 Data Preprocessing and Feature Engineering. The initial step involves transforming the raw song data into a format suitable for content-based analysis. The dataset is enriched by creating a new feature, 'text,' which is a concatenation of the 'title,' 'release,' and 'artist_name' columns. This consolidated text serves as a representation of the song's metadata and allows the system to capture a holistic view of each song.

The dataset is then filtered to retain unique song titles, with the 'title' column set as the index for ease of processing.

6.6.2 Text Processing. To effectively work with textual data, natural language processing (NLP) techniques are applied. The 'text' feature undergoes tokenization using NLTK's word tokenizer, followed by the removal of stopwords. Additionally, a lemmatization process is applied to ensure that words are reduced to their base forms. This preprocessed text is then utilized to construct a TF-IDF (Term Frequency-Inverse Document Frequency) vector representation for each song.

6.6.3 Cosine Similarity Calculation. The TF-IDF vectors form the basis for computing the cosine similarity between songs. Cosine similarity is a metric that measures the cosine of the angle between two non-zero vectors. In the context of content-based recommendation, it quantifies the similarity between the textual features of songs, producing a similarity matrix with values ranging from 0 to 1.

Rank	Recommended Songs
1	Everlong
2	The Pretender
3	Nothing Better (Album)
4	From Left To Right
5	Lifespan Of A Fly
6	Under The Gun
7	I Need A Dollar
8	Feel The Love
9	All The Pretty Faces
10	Bones

Table 8: Top 10 Recommended Songs for 'Learn To Fly'

6.6.4 Observations and Insights. Upon testing the recommendation function with a specific song title, such as 'Learn To Fly,' the system showcases its ability to suggest songs with similar artists, genres, and thematic elements. The recommendations exhibit a nuanced understanding of the textual patterns inherent in the song metadata.

In essence, the Content-Based Recommendation System excels in providing tailored suggestions by delving into the textual features of songs. This approach contributes to a more personalized and context-aware user experience, particularly suitable for users with specific preferences or seeking recommendations within defined thematic categories.

6.6.5 Advantages and Limitations. **Advantages:**

- Utilizes rich metadata to capture diverse aspects of songs.
- Provides recommendations based on content characteristics, irrespective of user interactions.
- Well-suited for users with specific preferences or niche interests.

Limitations:

- Relies heavily on the availability and quality of metadata.
- May face challenges in capturing evolving user preferences.
- Less effective for discovering entirely new or unexpected content.

In conclusion, the Content-Based Recommendation System offers a robust and insightful approach to personalized song recommendations, leveraging textual features to enhance the overall user experience.

7 LITERATURE REVIEW: RECOMMENDER SYSTEMS FOR MUSIC RECOMMENDATION

7.1 Introduction to Music Recommender Systems

The field of music recommendation has witnessed significant growth and development in recent years, driven by advancements in technology and the increasing volume of available music content. Recommender systems play a crucial role in assisting users in discovering new and relevant music based on their preferences. This literature review aims to provide an overview of existing studies, datasets, methodologies, and findings related to music recommendation systems.

7.2 Existing Datasets in Music Recommendation

Several datasets have been instrumental in advancing research in music recommendation systems. Notably, the Million Song Dataset (MSD) and the Spotify Million Playlist Dataset have been widely used. The MSD provides a comprehensive collection of audio features and metadata for a million songs, facilitating the exploration of content-based recommendation approaches. On the other hand, the Spotify Million Playlist Dataset focuses on collaborative filtering by including playlists created by users, allowing researchers to investigate user-item interactions.

7.3 Related Work by Other Researchers

7.3.1 Matrix Factorization in Collaborative Filtering (Koren et al., 2009). Koren et al. (2009) (Koren 2009) made a seminal contribution to the field by demonstrating the success of matrix factorization methods in collaborative filtering for movie recommendations. Their work emphasized the importance of capturing latent factors, such as user preferences and item characteristics, to improve recommendation accuracy. Inspired by their findings, these matrix factorization techniques were later adapted to music recommendation domains, enriching the collaborative filtering paradigm.

7.3.2 Neural Collaborative Filtering (Wang et al., 2021). In recent years, Wang et al. (2021) (Wang and Zhang 2021) introduced neural collaborative filtering as a powerful technique for capturing non-linear relationships in user-item interactions. Their deep learning approach has shown promising results in capturing complex patterns in user preferences. By leveraging neural networks, their model can learn intricate representations of user behavior and item characteristics, contributing to its success as a cornerstone in the modern landscape of music recommendation systems.

7.3.3 Contributions by Other Researchers. **Su and Khoshgoftaar (2009)**

Su and Khoshgoftaar (Su and Khoshgoftaar 2009) investigated the application of collaborative filtering and content-based methods for music recommendation. Their work explored the combination of user preferences and audio features to enhance recommendation accuracy.

McFee et al. (2012)

McFee et al. (McFee and Lanckriet 2012) focused on incorporating acoustic features and music metadata in recommendation systems. Their research emphasized the importance of leveraging both content-based and collaborative filtering approaches for a comprehensive music recommendation framework.

Schedl et al. (2014)

Schedl et al. (Schedl et al. 2014) delved into context-aware music recommendation, considering factors such as time, location, and user activity. Their work aimed to enhance the relevance of recommendations by accounting for the user's situational context.

7.4 Methodologies Employed in Music Recommender Systems

A variety of approaches have been explored to address the complexities of music recommendation. Content-based methods leverage features such as genre, artist, and audio characteristics to make recommendations. Collaborative filtering, both user-based and item-based, taps into user interactions and similarities between items. Hybrid models combine these approaches to enhance recommendation accuracy.

State-of-the-art methods often involve the use of machine learning techniques, including deep learning models like neural collaborative filtering (NCF) and matrix factorization. These models capture intricate patterns and dependencies within the data, offering improved accuracy in predicting user preferences.

7.5 Comparative Analysis of Existing Work

Discrepancies exist in the findings across studies. Some emphasize the importance of content-based methods in addressing the cold start problem, while others argue for the effectiveness of collaborative filtering in capturing user preferences dynamically. These variations highlight the evolving nature of the field and the need for a nuanced understanding of different recommendation scenarios.

7.6 Gaps in Existing Literature and Research Opportunities

Despite significant progress, certain challenges remain, such as the cold start problem for new users or items, and the interpretability of complex recommendation models. Future research could explore novel datasets that incorporate user context, temporal dynamics,

or sentiment analysis for more personalized and context-aware recommendations.

In conclusion, the literature on music recommender systems reflects a dynamic landscape of evolving datasets, methodologies, and findings. Understanding the work done by other researchers provides valuable insights for the development of more effective and user-centric recommendation systems.

RESULTS AND CONCLUSIONS OF RECOMMENDER SYSTEMS PROJECT

In this comprehensive recommender systems project, six distinct algorithms were employed to build recommendation systems. The primary focus was on addressing challenges such as cold start problems, optimizing model performance through hyperparameter tuning, and proposing a hybrid recommendation system for enhanced robustness. The evaluation metrics utilized included RMSE, precision@k, recall@k, and F1_Score@k.

1. Rank/Popularity-Based Recommendation System:

- Particularly effective in mitigating cold start problems.
- Essential for scenarios where data points for new users are limited.

2. User-User Similarity-Based Collaborative Filtering:

- Post hyperparameter tuning, F1 score improvements were observed.
- RMSE: 1.0521
- Precision: 0.413
- Recall: 0.721
- F1 score: 0.525

3. Item-Item Similarity-Based Collaborative Filtering:

- F1 score slightly lower compared to User-User Similarity.
- Predicted play_count for certain users might be suboptimal.
- Significant improvement after hyperparameter tuning.
- RMSE: 1.0328
- Precision: 0.408
- Recall: 0.665
- F1 score: 0.506

4. Model-Based Collaborative Filtering/Matrix Factorization:

- Slightly better F1 score and reduced RMSE post hyperparameter tuning.
- Capability to handle sparse data and predict missing ratings.
- Suitable for capturing complex patterns in user-item interactions.
- RMSE: 1.0141
- Precision: 0.415
- Recall: 0.635
- F1 score: 0.502

5. Clustering-Based Recommendation System:

- Tuned co-clustering model's F1 score comparable to the baseline.
- Limited improvement in model performance observed.
- RMSE: 1.0654
- Precision: 0.394
- Recall: 0.566
- F1 score: 0.465

6. Content-Based Recommendation System:

- Recommendations primarily centered around similar artists and songs.
- Dependent on features such as "title," "release," and "artist_name" rather than play_count.

Future Solution Design and Outlook:

- Proposal for a hybrid recommendation system combining various techniques.
- Inclusion of Rank/Popularity, Collaborative Filtering, Matrix Factorization, Clustering, and Content-Based systems.
- Matrix Factorization's ability to handle sparse data and capture intricate patterns is highlighted.
- Future hyperparameter tuning is recommended for continued model improvement.

Conclusion:

- The research successfully addressed diverse recommendation system approaches.
- Each technique demonstrated strengths and weaknesses in different scenarios.
- A hybrid approach is proposed for comprehensive and robust recommendations.
- Matrix Factorization, in particular, emerged as a promising technique due to its ability to handle sparse data and capture complex patterns.

Significance of the Results:

- The significance lies in the adaptability of the hybrid system to various user scenarios.
- Matrix Factorization's potential to handle sparse data is crucial for real-world applications.
- Ongoing model retraining with new data ensures adaptability to evolving user preferences.

Model Interpretation and Future Directions:

- Future work involves periodic retraining of models with updated data for improved accuracy.
- The proposed hybrid system lays the foundation for a production-grade tool showcasing the end-to-end machine learning process.

In summary, the research underscores the need for a nuanced, hybrid approach to recommendation systems, considering the strengths and limitations of each technique. Ongoing exploration of tools and continuous model refinement will contribute to the development of a robust, adaptable, and high-performing recommendation system.

References

Koren, Yehuda (2009). "Matrix Factorization Techniques for Recommender Systems". In: *Computer* 42.8, pp. 30–37.

McFee, Brian and Gert Lanckriet (2012). "Learning Content Similarities for Music Recommendation". In: *IEEE Transactions on Audio, Speech, and Language Processing* 20.8, pp. 2207–2218.

Schedl, Markus et al. (2014). "The LFM-1b Dataset for Music Retrieval and Recommendation". In: *IEEE Transactions on Multimedia* 16.6, pp. 1636–1647.

Su, Xiaoyuan and Taghi M. Khoshgoftaar (2009). "A Survey of Collaborative Filtering Techniques". In: *Advances in Artificial Intelligence*, p. 4.

Wang, Xiang and Hong-Jiang Zhang (2021). "Neural Collaborative Filtering for Music Recommendation". In: *IEEE Transactions on Industrial Informatics* 17.3, pp. 2216–2224.