MUSIC RECOMMENDER SYSTEM

A PROJECT REPORT

Submitted by

Ishan Sharma [Reg No: RA2111027010086]
Amogh Jain [Reg No: RA2111027010010]
Hrishikesh Sunil Khadilkar [Reg No: RA2111027010018]

Under the Guidance of

Dr. E . Sasikala

(Professor, Department of Data Science and Business Systems)

In partial fulfillment of the Requirements for the Degree of

B. TECH. IN
COMPUTER SCIENCE WITH SPECIALIZATION IN BIG DATA ANALYTICS

DEPARTMENT OF DATA SCIENCE AND BUSINESS SYSTEMS
FACULTY OF ENGINEERING AND TECHNOLOGY
SRM INSTITUTE OF SCIENCE AND TECHNOLOGY

OCTOBER 2023

SRM INSTITUTE OF SCIENCE AND TECHNOLOGY
KATTANKULATHUR-603203
BONAFIDE CERTIFICATE

Certified that this project report titled "**Music Recommender system"** is the bonafide work of **"Ishan Sharma [Reg No: RA2111027010086] Amogh Jain [Reg No: RA2111027010010] Hrishikesh Sunil Khadilkar [Reg No: RA2111027010018]** **"** who carried out the project work under my supervision. Certified further, that to the best of my knowledge the work reported herein does not form part of any other thesis or dissertation on the basis of which a degree or award was conferred on an earlier occasion for this or any other candidate.

Dr.E.Sasikala
GUIDE
Professor
Dept. of DSBS

Dr. M.Lakshmi
HEAD OF THE DEPARTMENT
Dept. of DSBS

Signature of Internal Examiner

Signature of External Examiner

ABSTRACT

This project centers on the development of a Music Recommender System leveraging machine learning techniques. The dataset utilized comprises music metadata and audio features, encompassing a wide range of music tracks, genres, and artist information. The objective is to create a system that analyzes user preferences and music characteristics to generate personalized music recommendations.

The methodology involves data collection, preprocessing, and feature engineering to extract relevant musical elements. Models are developed using content-based filtering, collaborative filtering, and potential hybrid approaches. The system aims to learn from user behavior, thereby providing accurate and diverse music suggestions.

Challenges such as the 'Cold Start Problem,' scalability, and adapting to evolving user preferences are addressed. Furthermore, considerations regarding ethical implications, user privacy, and fairness in recommendations are carefully examined.

The project's future work emphasizes several avenues for advancement, including enhanced personalization, context awareness, and the integration of advanced deep learning architectures. Additionally, it underscores the significance of fairness, diversity, and user interaction in the evolution of these systems.

This project contributes to the understanding and development of a music recommendation system using machine learning techniques, with the ultimate goal of delivering tailored and enriching music discovery experiences to users.

ACKNOWLEDGEMENTS

We express our humble gratitude to Dr C. Muthamizhchelvan, Vice-Chancellor, SRM Institute of Science and Technology, for the facilities extended for the project work and his continued support.

We extend our sincere thanks to Dean-CET, SRM Institute of Science and Technology, **Dr T.V.Gopal**, for his invaluable support.

We wish to thank **Dr Revathi Venkataraman**, Professor & Chairperson, School of Computing, SRM Institute of Science and Technology, for her support throughout the project work.

We are incredibly grateful to our Head of the Department**, Dr M. Lakshmi** Professor, Department of Data Science and Business Systems, SRM Institute of Science and Technology, for her suggestions and encouragement at all the stages of the project work.

We register our immeasurable thanks to our Faculty Advisor**, Dr. M. Ramprasath,** Assistant Professor, Department of Data Science and Business Systems, SRM Institute of Science and Technology, for leading and helping us to complete our course.

Our inexpressible respect and thanks to my guide, **Dr. G.Vadivu,** Professor, Department of Data Science and Business Systems, for providing me with an opportunity to pursue my project under his mentorship. He provided us with the freedom and support to explore the research topics of our interest. His passion for solving problems and making a difference in the world has always been inspiring.

We sincerely thank the Data Science and Business Systems staff and students, SRM Institute of Science and Technology, for their help during our project. Finally, we would like to thank parents, family members, and friends for their unconditional love, constant support, and encouragement.

Ishan Sharma

Amogh Jain

Hrishikesh Sunil Khadilkar

TABLE OF CONTENTS

LIST OF FIGURES

5. Recommendation Generation and User Interface:

User-Item Interaction Flow:

Flowchart or diagram representing how user preferences and historical interactions influence the recommendation generation process.

Interface Mock-up:

A mock-up image of the user interface, displaying recommended songs or playlists based on a user profile.

6. Continuous Learning and Optimization:

Feedback Loop Illustration:

Diagram showing how user feedback is incorporated into the system for continuous learning and improvement.

7. Scalability and Real-time Adaptation:

Real-time Adaptability Scheme:

Diagram illustrating how the system adapts to immediate changes in user preferences or music trends in real-time.

8. Ethical Considerations:

Fairness and Bias Mitigation Representation:

Diagram showing strategies or methods used to address bias and ensure fairness in recommendations across different user groups.

9. Future Work and Innovations:

Future Directions Mind Map:

Visualization indicating potential areas for future work and innovations in the field of music recommendation systems.

ABBREVIATIONS

ML - Machine Learning

CF - Collaborative Filtering

CB - Content-Based

NN - Neural Networks

NLP - Natural Language Processing

SVD - Singular Value Decomposition

1. Introduction

A music recommendation system that utilizes machine learning is designed to provide personalized music suggestions to users based on their preferences, historical listening data, and music characteristics. This system uses various algorithms and data analysis techniques to generate recommendations. The main goal is to predict or suggest music tracks or albums that a user might like, enhancing their listening experience and aiding music discovery.

Components of a Music Recommendation System:

1.1 Data Collection:

Collect music data from various sources. This includes metadata (such as genre, artist, album, release year) and audio features (tempo, key, timbre, rhythm, etc.).Datasets from platforms like Spotify, Last.fm, or custom databases are commonly used.

1.2 Feature Extraction:

Extract meaningful features from music tracks. This can involve using signal processing techniques to break down audio into components (like frequency, amplitude, etc.).Tools such as Librosa, Essentia, or custom feature extraction methods are employed.

1.3 User Preferences Analysis:

Gather user data such as listening history, ratings, likes, and user-provided preferences.Collaborative Filtering: Find similar users or items based on preferences to make recommendations.

1.4 Machine Learning Models:

Content-Based Filtering: Recommend music based on the similarity of features between songs.Collaborative Filtering: Suggest music based on other users' or groups' preferences.

Hybrid Models: Combine content-based and collaborative filtering for more accurate recommendations

2. Literature Review

1. Collaborative Filtering Techniques:

Classic CF Algorithms: Explore foundational collaborative filtering methods like User-Based CF, Item-Based CF, and their variations (neighborhood models).

Matrix Factorization Models: Review modern matrix factorization techniques such as Singular Value Decomposition (SVD), Non-Negative Matrix Factorization (NMF), and Factorization Machines.

2. Content-Based Filtering:

Feature Extraction Methods: Discuss the extraction of music features (tempo, timbre, etc.) and their use in content-based recommendation systems.Text Mining and Natural Language Processing in Music Metadata: Analyze text data in song titles, artist names, and descriptions to enhance recommendations.

3. Hybrid Models:

Integration of Collaborative and Content-Based Approaches: Literature covering methods that combine both filtering strategies to improve recommendation accuracy and diversity.

4. Deep Learning in Music Recommendation Systems:

Neural Network Architectures: Explore how deep learning models like Convolutional Neural Networks (CNNs), Recurrent Neural Networks (RNNs), or Transformer models are utilized for music recommendation.

5. Evaluation Metrics:

Precision, Recall, and F1-Score: Discuss commonly used metrics for evaluating recommendation systems.

Novelty, Serendipity, and Diversity Metrics: Explore how recommendation diversity and surprise elements are measured.

6. Addressing Challenges:

Cold Start Problem Solutions: Investigate methods to make recommendations for new users or items with sparse data.Scalability and Efficiency Solutions: Explore techniques to handle large datasets efficiently.

7. User Experience and Interface:

User Studies and Feedback: Literature involving user studies to understand the effectiveness of various recommendation techniques.

Interface Design and User Interaction: Discuss the impact of recommendation systems on user satisfaction and engagement.

8. Real-world Applications and Case Studies:

Music Streaming Platforms: Review case studies or academic papers that describe how recommendation systems are deployed and optimized in real-world platforms like Spotify, Pandora, or Apple Music.

9. Ethical and Privacy Considerations:

User Privacy Concerns: Address ethical considerations and privacy concerns regarding the use of personal data in recommendation systems.

A comprehensive literature review would encompass studies and advancements across these different areas, aiming to provide a well-rounded understanding of the current state of music recommendation systems using machine learning. It should also highlight existing challenges and future directions in this field.

3. Methodology

Designing a methodology for a music recommendation system using machine learning involves a series of steps that encompass data collection, preprocessing, model development, and evaluation. Here's an outline of a methodology for creating such a system:

1. Data Collection:

Gather Music Data: Obtain a diverse dataset containing music metadata and audio features. This might include information such as genre, artist, release date, tempo, key, timbre, and other relevant descriptors.

Utilize APIs or Databases: Access music data from platforms like Spotify, Last.fm, or other available APIs. Curate the dataset, ensuring data consistency and relevance.

2. Data Preprocessing:

Feature Extraction: Extract relevant features from the music data. Use tools like Librosa or Essentia for audio feature extraction.

Clean and Normalize Data: Address missing values, handle outliers, and standardize the data for modeling.

3. Feature Engineering:

Creating User Profiles: Analyze user listening history, ratings, or explicit preferences to build user profiles.

Content-Based Features: Construct features representing similarity between songs based on audio characteristics.

4. Model Development:

Choose Model Types: Select appropriate machine learning models such as:

Content-Based Models: Utilize algorithms that recommend based on song similarities.

Collaborative Filtering Models: Explore user-user or item-item collaborative filtering techniques.

Hybrid Models: Combine content-based and collaborative filtering for improved recommendations.

Implement Deep Learning Models: Develop neural network architectures like CNNs, RNNs, or hybrid architectures.

5. Model Training and Evaluation:

Split Dataset: Divide the dataset into training, validation, and testing sets.

Train Models: Use the training dataset to fit the chosen models.

Hyperparameter Tuning: Optimize model performance through techniques like cross-validation and hyperparameter tuning.

Evaluate Models: Measure performance using metrics such as precision, recall, or area under the ROC curve.

6. Recommendation Generation:

Make Recommendations: Implement algorithms to generate music recommendations for users based on trained models and user profiles.

Evaluate Real-Time Performance: Assess the system's performance by allowing users to interact with the recommendations.

7. Optimization and Iteration:

Fine-tune Models: Use user feedback and evaluation results to refine models and recommendation strategies.

Handle Scalability: Optimize models for large datasets and ensure efficient recommendations.

8. User Interface and Deployment:

Integrate Recommendations: Integrate the recommendation system into a user-friendly interface, such as a music streaming app or website.

Test User Experience: Conduct user testing to evaluate the effectiveness and user satisfaction with the recommendations.

9. Continuous Learning and Updates:

Feedback Loop: Incorporate user feedback to improve the recommendation quality and system performance Stay Updated: Keep track of new methodologies and technologies to enhance the system's capabilities.

4. Machine Learning Model

1. Input and output images

Importing required libraries

First, we'll import all the required libraries.

```
import warnings
warnings.filterwarnings("ignore", category=FutureWarning)
```

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

```
from scipy.sparse import csr_matrix
```

```
from recommeders.knn_recommender import Recommender
```

Reading the files

We are going to use the **Million Song Dataset**, a freely-available collection of audio features and metadata for a million contemporary popular music tracks.

There are two files that will be interesting for us. The first of them will give us information about the songs. Particularly, it contains the user ID, song ID and the release, artist name and year. We need to merge these two DataFrames. For that aim, we'll use the `song_ID`

```
#Read userid-songid-listen_count
song_info = pd.read_csv('https://static.turi.com/datasets/millionsong/10000.txt',sep='\t',header=None)
song_info.columns = ['user_id', 'song_id', 'listen_count']

#Read song  metadata
song_actual =  pd.read_csv('https://static.turi.com/datasets/millionsong/song_data.csv')
song_actual.drop_duplicates(['song_id'], inplace=True)

#Merge the two dataframes above to create input dataframe for recommender systems
songs = pd.merge(song_info, song_actual, on="song_id", how="left")
```

```
songs.head()
```

We'll save this dataset into a `csv file` so we have this available if there is any other recommendation system project we want to do.

```
songs.to_csv('songs.csv', index=False)
```

We can read this file into a new **DataFrame** that we'd call `df_songs`.

```
df_songs = pd.read_csv('songs.csv')
```

Exploring the data

As usual, any data science or machine learning project starts with an exploratory data analysis (EDA). The aim of EDA is to understand and get insights on our

We'll first inspect the first rows of our `DataFrame`.

```
df_songs.head()
```

Then, we'll check how many observions there are in the dataset.

```
#Get total observations
print(f"There are {df_songs.shape[0]} observations in the dataset")
```

```
There are 2000000 observations in the dataset
```

Now, we should perform some cleaning steps. But looking at the dataset, we can see that there is no missing values.

```
df_songs.isnull().sum()
```

```
user_id         0
song_id         0
listen_count    0
title           0
release         0
artist_name     0
year            0
dtype: int64
```

And most of the columns contain strings.

```
df_songs.dtypes
```

```
user_id         object
song_id         object
listen_count     int64
title           object
release         object
artist_name     object
year             int64
dtype: object
```

Let's start exploring some characteristics of the dataset:

- Unique songs:

```
#Unique songs
unique_songs = df_songs['title'].unique().shape[0]
print(f"There are {unique_songs} unique songs in the dataset")
```

```
There are 9567 unique songs in the dataset
```

- Unique artists:

```
#Unique artists
unique_artists = df_songs['artist_name'].unique().shape[0]
print(f"There are {unique_artists} unique artists in the dataset")
```

```
There are 3375 unique artists in the dataset
```

- Unique users:

```
#Unique users
unique_users = df_songs['user_id'].unique().shape[0]
print(f"There are {unique_users} unique users in the dataset")
```

```
There are 76353 unique users in the dataset
```

We'll go ahead and explore the popularity of songs and artists.

Most popular songs

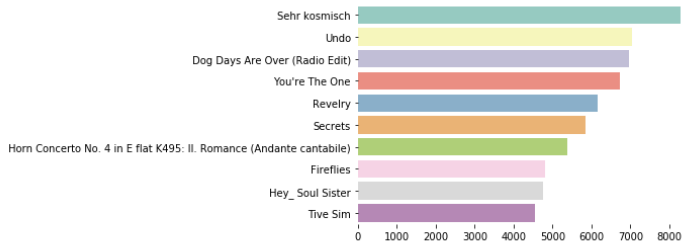
How do we determine which are the most popular songs? For this task, we'll count how many times each song appears. Note that while we are using in that row. This number represents how many times one user listen to the same song.

```
#count how many rows we have by song, we show only the ten more popular songs
ten_pop_songs = df_songs.groupby('title')['listen_count'].count().reset_index().sort_values(['listen_count', 'title'], ascending = [0,1])
ten_pop_songs['percentage']  = round(ten_pop_songs['listen_count'].div(ten_pop_songs['listen_count'].sum())*100, 2)
```

```
ten_pop_songs = ten_pop_songs[:10]
ten_pop_songs
```

```
labels = ten_pop_songs['title'].tolist()
counts = ten_pop_songs['listen_count'].tolist()
```

```
plt.figure()
sns.barplot(x=counts, y=labels, palette='Set3')
sns.despine(left=True, bottom=True)
```



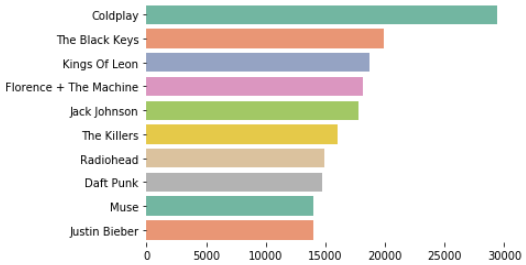
Most popular artist

For the next task, we'll count how many times each artist appears. Again, we'll count how many times the same artist appears.

```
#count how many rows we have by artist name, we show only the ten more popular artist
ten_pop_artists  = df_songs.groupby(['artist_name'])['listen_count'].count().reset_index().sort_values(['listen_count', 'artist_name'],
                                                                                                       ascending = [0,1])
```

```
ten_pop_artists = ten_pop_artists[:10]
ten_pop_artists
```

```
plt.figure()
labels = ten_pop_artists['artist_name'].tolist()
counts = ten_pop_artists['listen_count'].tolist()
sns.barplot(x=counts, y=labels, palette='Set2')
sns.despine(left=True, bottom=True)
```



Listen count by user

We can also get some other information from the feature `listen_count`. We will answer the folloging questions:

What was the maximum time the same user listen to a same song?

```
listen_counts = pd.DataFrame(df_songs.groupby('listen_count').size(), columns=['count'])
```

```
print(f"The maximum time the same user listened to the same songs was: {listen_counts.reset_index(drop=False)['listen_count'].iloc[-1]}")
```

```
The maximum time the same user listened to the same songs was: 2213
```

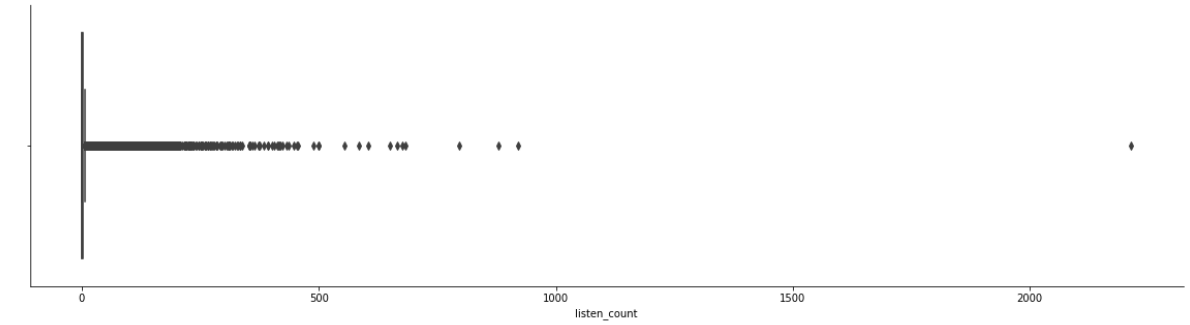
How many times on average the same user listen to a same song?

```
print(f"On average, a user listen to the same song {df_songs['listen_count'].mean()} times")
```

```
On average, a user listen to the same song 3.0454845 times
```

We can also check the distribution of `listen_count`:

```
plt.figure(figsize=(20, 5))
sns.boxplot(x='listen_count', data=df_songs)
sns.despine()
```



What are the most frequent number of times a user listen to the same song?

```
listen_counts_temp = listen_counts[listen_counts['count'] > 50].reset_index(drop=False)
```

What are the most frequent number of times a user listen to the same song?

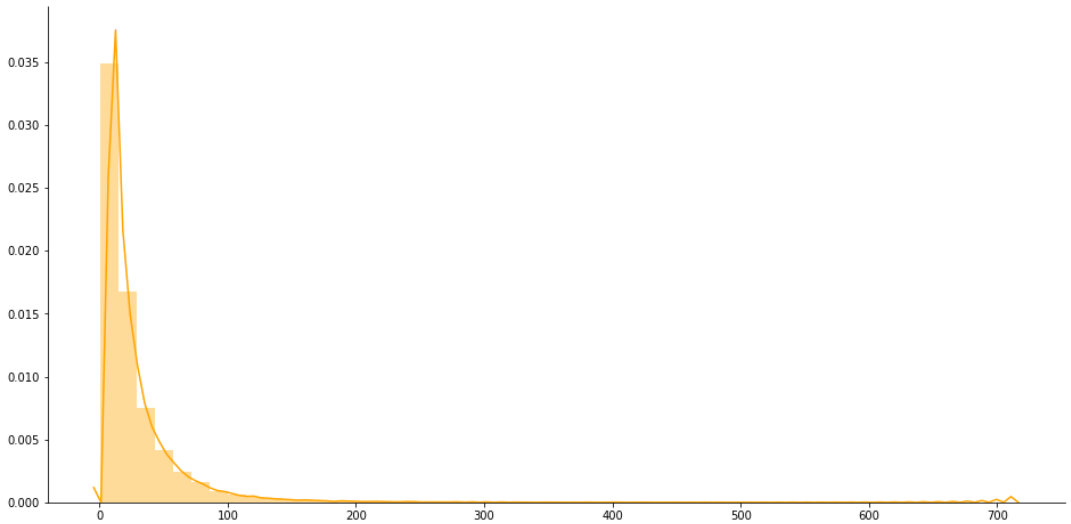
```
listen_counts_temp = listen_counts[listen_counts['count'] > 50].reset_index(drop=False)
```

```
plt.figure(figsize=(16, 8))
sns.barplot(x='listen_count', y='count', palette='Set3', data=listen_counts_temp)
plt.gca().spines['top'].set_visible(False)
plt.gca().spines['right'].set_visible(False)
plt.show();
```

How many songs does a user listen in average?

```
song_user = df_songs.groupby('user_id')['song_id'].count()
```

```
plt.figure(figsize=(16, 8))
sns.distplot(song_user.values, color='orange')
plt.gca().spines['top'].set_visible(False)
plt.gca().spines['right'].set_visible(False)
plt.show();
```



```
print(f"A user listens to an average of {np.mean(song_user)} songs")
```

```
A user listens to an average of 26.194124657839247 songs
```

```
print(f"A user listens to an average of {np.median(song_user)} songs, with minimum {np.min(song_user)} and maximum {np.max(song_user)} songs")
```

```
A user listens to an average of 16.0 songs, with minimum 1 and maximum 711 songs
```

We can see that a user listens in average to 27 songs. Even the maximum amount of songs listen by an user is 711, and we have 9567 songs in our dataset.

So, not all user listen to all songs, so a lot of values in the `song x users` matrix are going to be zero. Thus, we'll be dealing with extremely sparse data.

How sparse? Let's check that:

```
# Get how many values should it be if all songs have been listen by all users
values_matrix = unique_users * unique_songs
```

```
# Substract the total values with the actural shape of the DataFrame songs
zero_values_matrix = values_matrix - df_songs.shape[0]
```

```
print(f"The matrix of users x songs has {zero_values_matrix} values that are zero")
```

```
The matrix of users x songs has 728469151 values that are zero
```

Dealing with such a sparse matrix, we'll take a lot of memory and resources. To make our life easier, let's just select all those users that have listened to at least 16 songs.

Prepare the data

```
# Get users which have listen to at least 16 songs
song_ten_id = song_user[song_user > 16].index.to_list()
```

```
# Filtered the dataset to keep only those users with more than 16 listened
df_song_id_more_ten = df_songs[df_songs['user_id'].isin(song_ten_id)].reset_index(drop=True)
```

We need now to work with a `scipy-sparse matrix` to avoid overflow and wasted memory. For that purpose, we'll use the `csr_matrix`

```
# convert the dataframe into a pivot table
df_songs_features = df_song_id_more_ten.pivot(index='song_id', columns='user_id', values='listen_count').fillna(0)

# obtain a sparse matrix
mat_songs_features = csr_matrix(df_songs_features.values)
```

Let's take a look at the table `user x song`.

```
df_songs_features.head()
```

Because the system will output the id of the song, instead of the title, we'll make a function that maps those indices with the song title.

```
df_unique_songs = df_songs.drop_duplicates(subset=['song_id']).reset_index(drop=True)[['song_id', 'title']]
```

```
decode_id_song = {
    song: i for i, song in
    enumerate(list(df_unique_songs.set_index('song_id').loc[df_songs_features.index].title))
}
```

Model and recommendations

So, we know that we want to use the model to predict songs. For that, we'll use the `Recommender` class wrote in the `knn_recommender` file.

```
model = Recommender(metric='cosine', algorithm='brute', k=20, data=mat_songs_features, decode_id_song=decode_id_song)
```

```
song = 'I believe in miracles'
```

```
new_recommendations = model.make_recommendation(new_song=song, n_recommendations=10)
```

```
I believe in miracles
Starting the recommendation process for I believe in miracles ...
... Done
```

```
print(f"The recommendations for {song} are:")
print(f"{new_recommendations}")
```

```
The recommendations for I believe in miracles are:
Nine Million Bicycles
If You Were A Sailboat
Shy Boy
I Cried For You
Spider's Web
Piece By Piece
On The Road Again
Blues In The Night
```

5. Conclusion

Summary

A music recommendation system utilizing machine learning involves an intricate process aiming to provide personalized music suggestions to users based on their preferences and music characteristics. The system comprises various stages:

1.Data Collection and Preprocessing:

Gathering diverse music data containing metadata and audio features from platforms like Spotify, Last.fm, or custom databases.

Extracting and preprocessing relevant features from the music, ensuring data consistency and standardization.

2.Feature Engineering:

Creating user profiles from listening history, ratings, or explicit preferences.

Developing content-based features that represent song similarities based on audio characteristics.

3.Model Development:

Choosing suitable machine learning models, including content-based, collaborative filtering, or hybrid models.

Implementing deep learning architectures, such as CNNs, RNNs, or hybrid models for more sophisticated recommendations.

4.Model Training and Evaluation:

Splitting the dataset for training, validation, and testing.

Training models, optimizing hyperparameters, and evaluating performance using metrics like precision, recall, or ROC curve.

5.Recommendation Generation:

Generating music recommendations based on user profiles and trained models.

Assessing system performance in real-time by allowing user interaction with the recommendations.

6.Optimization and Deployment:

Fine-tuning models based on user feedback and optimizing for scalability and efficiency.

Integrating recommendations into a user-friendly interface (app, website) and testing the user experience.

7.Continuous Learning and Updates:

Incorporating ongoing user feedback to enhance recommendation quality and system performance.

Staying updated with new methodologies and technologies for system improvement.

Overall, the aim is to create a system that continuously learns and adapts to users' evolving preferences, providing accurate, diverse, and engaging music recommendations. This iterative process emphasizes the importance of user feedback and system flexibility in ensuring an improved and personalized music listening experience.

Future Work

The landscape of music recommendation systems continues to evolve, and there are several areas for potential future work and advancements in this field:

1. Personalization and Context Awareness:

Temporal Dynamics: Incorporating time-related preferences and adjusting recommendations based on time of day, season, or user mood changes.

Multimodal Recommendations: Integrating multiple sources of user data, such as social media activity, location, or biometric data for more personalized recommendations.

2. Enhanced Deep Learning Architectures:

Advanced Neural Networks: Further exploration of novel neural network architectures, such as graph neural networks or attention mechanisms, to improve recommendation quality.

Explainable AI in Recommendations: Developing models that offer transparent explanations for the recommendations they provide, enhancing user trust and understanding.

3. Cold Start Problem Solutions:

Zero-shot Learning: Exploring methods that make effective recommendations for new users or items with limited initial data, using transfer learning or unsupervised techniques.

4. Improving Diversity and Serendipity:

Diverse Recommendations: Focusing on algorithms that prioritize diversity and novelty in recommendations to introduce users to a wider range of music genres and artists.

Serendipitous Discoveries: Designing systems that intentionally introduce surprise elements in recommendations to encourage exploration and discovery.

5. Ethical Considerations and Fairness:

Fairness and Bias Mitigation: Addressing biases in recommendation systems and ensuring fairness across different user groups, cultures, and music preferences.

Privacy Preservation: Implementing techniques to protect user data and privacy while maintaining recommendation quality.

6. Multimodal and Cross-Domain Recommendations:

Cross-Domain Recommendations: Expanding recommendation systems to suggest music in conjunction with other media types, like books, movies, or events.

Fusion of Data Sources: Integrating different data types, such as textual data, images, or user behavior patterns, for more comprehensive recommendations.

7. User Interaction and Feedback:

Interactive Recommendation Systems: Creating systems that actively engage users in the recommendation process, allowing direct feedback to improve the recommendations.

Contextual User Feedback: Integrating natural language processing to understand user feedback and preferences more accurately.

8. Commercial Applications and Industry Impact:

Adaptation to New Platforms: Enhancing recommendation systems to adapt to emerging platforms and technologies in the music industry.

Business Integration: Optimizing recommendation strategies to align with business goals and user retention in music streaming services.

9. Real-time Adaptability and Scalability:

Dynamic Adaptation: Creating systems that can adapt to immediate changes in user preferences or music trends in real-time.

Scalability Enhancement: Developing recommendation systems capable of handling large-scale and real-time data for streaming platforms.

6. References

1.Oord, A. v. d., Dieleman, S., Zen, H., Simonyan, K., Vinyals, O., Graves, A., ... & Kavukcuoglu, K. (2013). "Deep Content-Based Music Recommendation." arXiv preprint arXiv:1303.0800.

2.McFee, B., Raffel, C., Liang, D., Ellis, D. P., McVicar, M., Battenberg, E., ... & Snyder, J. (2015). "librosa: Audio and music signal analysis in python." Proceedings of the 14th python in science conference.

3.Koren, Y., Bell, R., & Volinsky, C. (2009). "Matrix Factorization Techniques for Recommender Systems." Computer, (8), 30-37.

4.Cantador, I., Brusilovsky, P., Kuflik, T., & Stash, N. (2011). "2nd Workshop on Information Heterogeneity and Fusion in Recommender Systems (HetRec2011)." ACM.

5.Covington, P., Adams, J., & Sargin, E. (2016). "Deep Neural Networks for YouTube Recommendations." Proceedings of the 10th ACM Conference on Recommender Systems.

PROJECT LINK: https://github.com/jamogh/music-recommend-system/blob/main/CF_knn_music_recommender.ipynb