MUSIC RECOMMENDATION SYSTEM

A Capstone Project report submitted

in partial fulfillment of requirement for the award of degree

BACHELOR OF TECHNOLOGY

in

SCHOOL OF COMPUTER SCIENCE AND ARTIFICIAL INTELLIGENCE

by

Mohammed Mudassir Hussain	**(2103A52058)**
Syed Khaja Mukarram Ajaz	**(2103A52069)**
Mohammed Naveed Sharief	**(2103A52159)**

Under the guidance of

Dr. Mamta Pandey

Associate Professor, School of CS&AI.

SR University, Ananthsagar,Warangal,Telagnana-506371

SR University

Ananthasagar, Warangal.

SR UNIVERSITY

CERTIFICATE

This is to certify that this project entitled **"Music Recommendation System"** is the Bonafide work carried out by **Mohammed Mudassir Hussain, Syed Khaja Mukarram Ajaz, Mohammed Naveed Sharief** as a Capstone Project for the partial fulfillment to award the degree BACHELOR **OF TECHNOLOGY** in **School of Computer Science and Artificial Intelligence** during the academic year 2024-2025 under our guidance and Supervision.

Dr. Mamta Pandey	**Dr. M. Sheshikala**
Associate Professor,	Professor & Head,
School of CS&AI	**School of CS&AI,**
SR University	SR University,
Anathasagar, Warangal	Anathasagar, Warangal

Reviewer-1	**Reviewer-2**
Name:	Name:
Designation:	Designation:
Signature:	Signature:

ACKNOWLEDGEMENT

We owe an enormous debt of gratitude to our Capstone project guide **Dr. Mamta Pandey, Associate Professor** as well as Head of the School of CS&AI, **Dr. M. Sheshikala, Professor** and Dean of the School of CS&AI, **Dr. Indrajeet Gupta Professor** for guiding us from the beginning through the end of the Capstone Project with their intellectual advices and insightful suggestions. We truly value their consistent feedback on our progress, which was always constructive and encouraging and ultimately drove us to the right direction.

We express our thanks to project coordinator's **Mr. Sallauddin Md, Asst. Prof., and R. Ashok Asst. Prof.** for their encouragement and support.

Finally, we express our thanks to all the teaching and non-teaching staff of the department for their suggestions and timely support.

Abstract

The music industry has witnessed a tremendous growth in online streaming platforms, making personalized music recommendations a critical feature for user engagement. This project aims to develop a Personalized Music Recommendation System utilizing various machine learning models and the Spotify API to enhance the user experience by suggesting relevant tracks based on user input. The system integrates data from multiple sources, including a Kaggle dataset and tracks fetched using the Spotify API. Audio features such as danceability, energy, and acousticness are extracted and standardized for consistent analysis. The project systematically collected audio features from 30+ Spotify playlists spanning multiple genres including Indie, Global Top Charts, Electronic/Dance, Hip-Hop/R&B, Regional Hits, and Indian Regional music.

Using the Spotipy library, we extracted 4,341 unique tracks, capturing comprehensive audio features such as danceability, energy, key, loudness, mode, speechiness, acousticness, instrumentalness, liveness, valence, and tempo. These tracks were meticulously collected from diverse playlists like "Indie India", "Hot Hits Global", "RapCaviar", and "Rock Classics".

The project involves merging the external datasets with real-time track data obtained via the Spotify API, creating a comprehensive dataset for model training. Several recommendation algorithms were implemented and evaluated, including K-Nearest Neighbors (KNN), Cosine Similarity, Principal Component Analysis (PCA) with KNN, and Density-Based Spatial Clustering of Applications with Noise (DBSCAN). The system compares these models based on their performance in generating accurate recommendations. Additionally, a user-friendly interface is developed using Streamlit, allowing users to interact with the application, fetch song details, and receive personalized recommendations.

The results demonstrate the system's capability to provide relevant song suggestions by leveraging hybrid techniques, thus combining content-based filtering with advanced clustering and similarity measures. This approach showcases an effective strategy for building scalable and adaptive music recommendation systems.

CONTENTS

Introduction

The digital music industry has transformed significantly with the rise of online streaming platforms, such as Spotify, Apple Music, and Amazon Music, which offer vast libraries of songs accessible to users worldwide. With millions of tracks available, users often face the challenge of discovering new songs that match their preferences. This has led to the growing importance of personalized music recommendation systems that can automatically suggest relevant tracks, enhancing user satisfaction and engagement.

A music recommendation system typically leverages user behavior, song metadata, and audio features to generate suggestions tailored to individual preferences. Traditional approaches rely on collaborative filtering, content-based filtering, or a combination of both (hybrid methods). However, these methods face limitations when it comes to cold-start problems, limited user data, and scalability issues with expanding song libraries.

In this project, we address these challenges by developing a Personalized Music Recommendation System that utilizes a hybrid approach combining content-based filtering with advanced machine learning models. The system integrates multiple data sources, including real-time track data fetched from the Spotify API and a pre-existing Kaggle dataset. By analyzing a wide range of audio features such as danceability, energy, and acousticness, our system aims to deliver accurate and personalized song recommendations.

Furthermore, this project introduces a user-friendly interface using Streamlit, allowing users to search for specific songs and receive recommendations instantly. The application also supports the addition of new tracks dynamically through the Spotify API, enabling a scalable and adaptive solution for personalized music recommendations.

Problem Statement

With the rapid growth of digital music streaming services, users are overwhelmed by vast song libraries, making it difficult for them to discover music that aligns with their preferences. This problem is further exacerbated by the diversity in musical tastes, influenced by factors such as genre, mood, region, and cultural background. Traditional music recommendation systems, often relying on collaborative filtering or content-based filtering alone, face several challenges:

Cold-Start Problem: New or less popular songs may not have sufficient user interaction data, making it hard for collaborative filtering algorithms to recommend them effectively.

Limited Context Awareness: Many existing systems struggle to incorporate real-time data updates, such as emerging songs or changes in user preferences.

Scalability Issues: With continuously expanding song libraries, recommendation systems need to be capable of handling and processing large datasets efficiently.

Personalization Gaps: Users expect personalized suggestions based on their unique listening history and preferences, which traditional approaches may fail to capture accurately.

The objective of this project is to develop an enhanced Personalized Music Recommendation System that addresses these issues by integrating data from multiple sources, including the Spotify API, and implementing a hybrid recommendation approach. By utilizing advanced techniques such as clustering and dimensionality reduction alongside traditional similarity measures, the system aims to deliver precise, personalized, and context-aware recommendations. The final solution includes a user-friendly interface, enabling seamless interaction and dynamic updates based on real-time data.

Literature Survey

Table 1: Tabular representation of Literature Survey

S No.	Author(s) & Year	Model Used	Parameters	Merits	Limitations & Drawbacks
1	Kostrzewa, D., Chrobak, J., & Brzeski, R. (2024)	Content-Based Filtering	Acousticness, Tempo, Energy	Enhanced recommendation precision by focusing on relevant attributes	Limited by the quality and scope of selected features
2	Deldjoo, Y., Schedl, M., & Knees, P. (2024)	Content-Driven Methods	Audio Features, Metadata	Comprehensive review of content-based approaches and their evolution	Challenges in integrating diverse datasets
3	Girsang, A. S., & Wibowo, A. (2021)	Neural Collaborative Filtering	User-Item Interactions	Captures complex user-item relationships using deep learning	Computationally intensive and requires large datasets
4	Bi, X., Qu, A., & Shen, X. (2018)	Multilayer Tensor Factorization	User-Item-Time Interactions	Models multi-dimensional data effectively for recommendations	Complexity increases with additional layers
5	Zhang, Y., Bi, X., Tang, N., & Qu, A. (2020)	Dynamic Tensor Models	Temporal User Preferences	Addresses temporal shifts in user preferences	Requires constant data updates to remain relevant
6	Bi, X., Tang, X., Yuan, Y., Zhang, Y., & Qu, A. (2021)	Tensor-Based Methods	High-Dimensional Data	Effective in handling complex, high-dimensional data	Computationally demanding
7	Adomavicius, G., & Tuzhilin, A. (2010)	Context-Aware Systems	Time, Location, Mood	Incorporates contextual information for personalized recommendations	Scalability and data sparsity issues
8	Aggarwal, C. C. (2016)	Context-Sensitive Recommender Systems	Contextual Attributes	Enhances recommendation relevance by considering context	Implementation complexity
9	Gediminas, A., & Alexander, T. (2015)	Hybrid Recommender Systems	Collaborative and Content-	Combines multiple methods	Increased system complexity

			Based Filtering	for improved accuracy	
10	Xian, Y., Fu, Z., Muthukrishnan, S., De Melo, G., & Zhang, Y. (2019)	Reinforcement Learning on Knowledge Graphs	User Preferences, Item Attributes	Provides explainable recommendations through knowledge graph reasoning	Requires extensive knowledge graph construction

The field of music recommendation systems has seen significant advancements, with researchers exploring diverse methodologies to enhance user satisfaction and system accuracy. Kostrzewa et al. (2024) demonstrated the effectiveness of content-based filtering by emphasizing relevant features such as acousticness and tempo, which improve recommendation precision while addressing cold-start challenges [1]. Similarly, Deldjoo et al. (2024) reviewed the evolution of content-driven methods, identifying their strengths in leveraging audio features and metadata but also pointing out challenges in integrating heterogeneous datasets [2]. Girsang and Wibowo (2021) applied neural collaborative filtering to capture complex user-item interactions, showcasing the potential of deep learning in this domain, albeit at the cost of increased computational requirements [3].

Tensor-based approaches have gained traction for modeling high-dimensional data. Bi et al. (2018) utilized multilayer tensor factorization to account for user-item-context interactions, effectively addressing temporal shifts in user preferences but requiring significant computational resources [4]. Reinforcement learning on knowledge graphs, explored by Xian et al. (2019), provided explainable recommendations by reasoning over structured relationships between entities, demonstrating the feasibility of integrating explainability into recommendation systems [10].

Leveraging insights from these studies, our project aimed to create a hybrid music recommendation system that combines content-based filtering with clustering and dimensionality reduction techniques. Unlike traditional methods, we incorporated real-time track data fetched using the Spotify API and integrated it with a curated Kaggle dataset to enhance diversity and relevance. Advanced algorithms such as KNN, cosine similarity, PCA-KNN, and DBSCAN were implemented to overcome cold-start issues, improve scalability, and ensure precise recommendations. The user-friendly Streamlit-based interface further facilitated dynamic interactions, enabling users to fetch track details and obtain personalized suggestions effortlessly.

Existing Methods Vs. Proposed Method

Existing Methods

Most music recommendation systems currently deployed by streaming services use one of the following approaches:

1. **Collaborative Filtering**:
   - **User-Based Collaborative Filtering** analyses the listening habits of users and suggests tracks based on the preferences of similar users. While this method is effective in identifying user similarities, it often suffers from the cold-start problem, where new tracks or users lack sufficient data for accurate recommendations.
   - **Item-Based Collaborative Filtering** recommends songs that are frequently listened to together, focusing on song-to-song similarities. However, it heavily relies on historical user interaction data and may not perform well when user behaviour changes over time.
2. **Content-Based Filtering**:
   - This method recommends songs based on the characteristics of the tracks, such as genre, tempo, and audio features. It uses metadata and feature extraction (e.g., danceability, energy, loudness) to find songs similar to what the user already likes. Although it avoids the cold-start problem for new users, it tends to lack diversity in recommendations, often suggesting songs that are too similar.
3. **Hybrid Methods**:
   - Hybrid approaches combine collaborative and content-based filtering to overcome their individual limitations. These methods can deliver more diverse recommendations but often require complex algorithms and increased computational resources, which can be a challenge for real-time recommendations.

Proposed Methods

In this project, we adopt an enhanced hybrid approach that integrates content-based filtering with advanced machine learning models to overcome the limitations of existing methods. The key aspects of our proposed method include:

1. **Data Integration:**

- We combined multiple datasets, including a Kaggle dataset and real-time track data fetched using the Spotify API. This allowed us to create a comprehensive and up-to-date dataset featuring detailed audio features of 4,341 unique tracks from various playlists like "Indie India", "RapCaviar", and "Rock Classics".

2. **Standardization and Feature Engineering:**

   - Audio features such as danceability, energy, loudness, acousticness, and tempo were standardized to ensure consistent analysis. This preprocessing step improved the quality and reliability of the input data for model training.

3. **Advanced Modeling Techniques:**

   - We implemented multiple recommendation models, including:
     - **K-Nearest Neighbors (KNN**): Used for finding the most similar tracks based on standardized audio features.
     - **Cosine Similarity**: Measures the similarity between tracks based on their feature vectors, providing a robust alternative to Euclidean distance.
     - **PCA with KNN:** Dimensionality reduction using Principal Component Analysis (PCA) to capture essential patterns in the data, followed by KNN for efficient recommendations.
     - **DBSCAN Clustering:** A density-based clustering method to group similar songs, enabling discovery of tracks even in sparse datasets.

4. **Dynamic Dataset Expansion:**

   - The system supports real-time updates by fetching additional song data from the Spotify API. When a song is not found in the existing dataset, it is automatically added with its audio features extracted from Spotify. This addresses the cold-start problem effectively.

5. **User-Friendly Interface:**

- A Streamlit-based UI was developed to facilitate seamless user interaction. Users can input song names and artists, view details of the fetched track, and receive recommendations using multiple models. The interface also displays album covers and provides Spotify links for direct access.

Table 2: Existing vs Proposed Methods

Aspect	Existing Methods	Proposed Method
Data Source	Historical user interactions	Merged dataset (Kaggle + Spotify API)
Cold-Start Problem	Significant issue	Mitigated using real-time data fetching
Model Variety	Limited (collaborative/content)	KNN, Cosine Similarity, PCA, DBSCAN
Scalability	High computational demand	Efficient due to feature engineering
User Experience	Static recommendations	Dynamic updates and diverse suggestions

The proposed system's hybrid approach leverages a combination of similarity measures, clustering, and real-time data updates, providing an adaptive and efficient solution for personalized music recommendations.

System Architecture / Technical Specifications

System Architecture

The architecture of the Personalized Music Recommendation System is designed to facilitate seamless integration between data acquisition, preprocessing, model execution, and user interaction. It ensures efficient handling of real-time data, dynamic updates, and interactive recommendations. The key modules are outlined below:

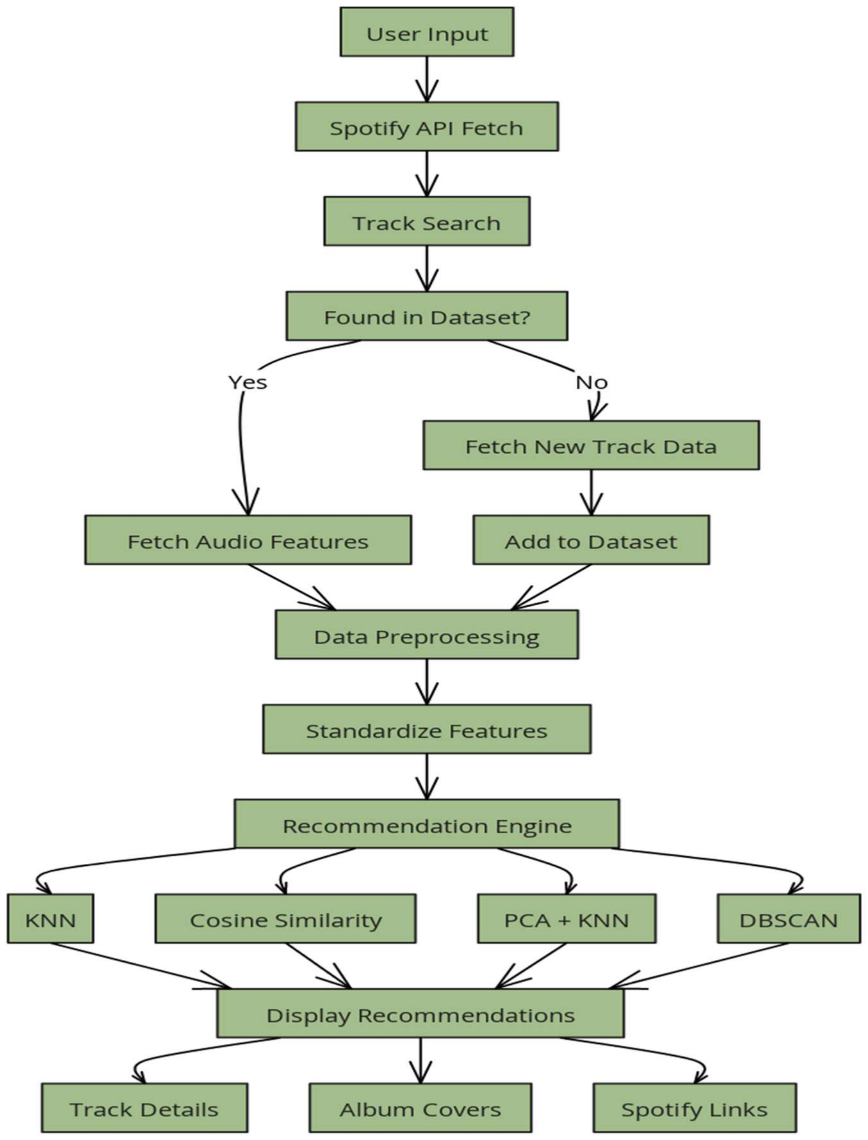


Figure 1: System Architecture Flowchart

1. Data Collection Module

This module manages the acquisition and integration of data from multiple sources:

- **Kaggle Dataset**: Provides song metadata and audio features for a comprehensive dataset base.
- **Spotify API**: Using the Spotipy library, real-time track data (audio features, metadata, and popularity metrics) is fetched from 30+ curated playlists, spanning genres like Indie, Global Top Charts, and Regional Hits.

The combined dataset is cleaned, standardized, and deduplicated to create a training and evaluation-ready dataset containing 4,341 unique tracks.

2. Data Preprocessing Module

Key functionalities include:

- **Standardization**: Numerical features such as danceability, energy, tempo, loudness, etc., are scaled using **StandardScaler** to ensure uniform analysis.
- **Data Cleaning**: Handles missing values and removes duplicates to ensure high-quality data.
- **Feature Engineering**: Normalizes input data for consistent model performance across algorithms.

3. Recommendation Engine

This module implements advanced machine learning models to provide personalized song recommendations:

- **K-Nearest Neighbors (KNN)**: Identifies songs similar to user input based on feature proximity.
- **Cosine Similarity**: Calculates angular similarity between songs using vectorized feature representations.

- **PCA + KNN**: Uses Principal Component Analysis (PCA) for dimensionality reduction, improving efficiency in large datasets.
- **DBSCAN Clustering**: Groups tracks by density, enabling recommendations for less popular or unique songs by discovering similar clusters.

Dynamic dataset expansion ensures tracks not found in the existing database are fetched from the Spotify API and processed in real-time.

4. User Interface Module

Built using **Streamlit**, the UI provides:

- A clean, interactive interface where users can input a song name and artist.
- Real-time display of fetched song details, album covers, and Spotify links.
- Exploration of recommendations generated by multiple models, visualized interactively.

5. Spotify Integration Module

Handles communication with the Spotify API via **Spotipy**, enabling:

- Real-time fetching of metadata and audio features.
- Seamless addition of new tracks to the dataset to mitigate the cold-start problem.
- Scalable updates to maintain relevance as user preferences evolve.

Technical Specifications

- **Programming Language**: Python 3.11
- **Libraries and Frameworks**:
    - **Data Processing**: Pandas, NumPy, Scikit-learn
    - **API Integration**: Spotipy for Spotify API
    - **User Interface**: Streamlit
    - **Machine Learning Models**: KNN, Cosine Similarity, PCA, DBSCAN
- **Datasets**:
    - **Kaggle Dataset**: Pre-existing audio feature and metadata.

    - **Spotify API Data**: 4,341 unique tracks fetched from playlists.
- **Development Tools**:
    - **IDE**: Jupyter Notebook (data analysis and modeling), Streamlit (UI development).
    - **Version Control**: GitHub for collaborative management.
- **Deployment Environment**:
    - Local environment for initial testing.
    - Cloud platforms like Streamlit Cloud or Heroku for broader access.

Methodology

The methodology outlines the step-by-step process used in developing the Personalized Music Recommendation System, detailing the data collection, preprocessing, model implementation, and user interaction.

1. Data Collection

The data collection process involves integrating multiple sources to create a comprehensive dataset:

- **Kaggle Dataset**: This dataset includes metadata and audio features for a large collection of songs, serving as a foundational dataset for the project.
- **Spotify API**: Real-time data fetching is done using the Spotipy library to retrieve:
    - Metadata such as track names, artists, and popularity.
    - Audio features including danceability, energy, tempo, acousticness, valence, and more.
    - Tracks from 30+ curated Spotify playlists across multiple genres, resulting in 4,341 unique tracks.

These datasets were combined to ensure variety and coverage of different musical styles and preferences.

2. Data Preprocessing

Data preprocessing ensures the quality and consistency of input data for model training:

- **Standardization**:
    - Audio features were scaled using StandardScaler for consistency in range and magnitude.
    - This step ensures that algorithms like KNN and PCA perform optimally by avoiding bias due to differing feature scales.
- **Handling Missing and Duplicate Data**:

    - Duplicate entries were removed based on track IDs.
    - Rows with missing or incomplete feature data were filtered out.
- **Merging Datasets**:
    - Common columns (e.g., track ID, audio features) from the Kaggle dataset and Spotify API data were identified and merged.

The final processed dataset serves as the input for the recommendation models.

3. Model Implementation

The recommendation engine comprises multiple machine learning algorithms, each with unique strengths:

1. **K-Nearest Neighbors (KNN)**:
    - Identifies the n most similar songs based on feature vectors.
    - Uses Euclidean distance to compute similarity.
2. **Cosine Similarity**:
    - Measures the angular similarity between tracks in a high-dimensional feature space.
    - Effective for detecting closely related songs even when absolute values differ significantly.
3. **PCA + KNN**:
    - Principal Component Analysis (PCA) reduces the dimensionality of the dataset, retaining essential patterns while improving computational efficiency.
    - KNN is applied on the reduced feature space for fast and effective neighbor identification.
4. **DBSCAN Clustering**:
    - Groups tracks into clusters based on feature density.
    - Tracks within the same cluster are recommended together, enabling discovery of less popular songs or outliers.

4. User Interface Development

The user interface, built with **Streamlit**, facilitates easy and interactive access to the system:

- **Input**: Users provide song names and artist names.
- **Output**:
    - Display track details such as album covers and Spotify links.
    - Recommendations generated by all models, displayed interactively.
- **Dynamic Updates**:
    - Tracks not in the dataset are fetched from the Spotify API in real time, ensuring that new and emerging songs are included in recommendations.

5. Evaluation and Refinement

□ **Metrics Used**:

- **Euclidean Distance**: Quantifies the average distance between the feature vector of the input track and those of the recommendations. Lower values indicate better matches.
- **Feature Deviation**: Measures the average deviation across features like danceability, energy, and tempo between the original and recommended tracks.

□ **Evaluation Process**:

- Models (KNN, Cosine Similarity, PCA + KNN, DBSCAN) were evaluated on a test dataset.
- Each model generated recommendations for randomly sampled tracks.
- Metrics were computed for each track's recommendations and aggregated to determine average performance.

□ **Insights**:

- The results provided a clear comparison of models based on their ability to deliver precise and diverse recommendations.
- Euclidean distance and feature deviation scores highlighted the strengths and weaknesses of each model.

The methodology ensures a robust pipeline from data acquisition to final recommendation, enabling scalability and adaptability for future enhancements

Features and Functionalities

The Music Recommendation System is designed with a robust set of features and functionalities to provide users with a seamless and personalized music discovery experience. Below is a detailed explanation of the key features and their associated functionalities:

1. Spotify API Integration

The system is integrated with Spotify's API to fetch detailed track data and audio features.

- **Features**:
    - Retrieves metadata such as **track name**, **artist name**.
    - Extracts audio features, including:
        - **Danceability**: Measures how suitable a track is for dancing.
        - **Energy**: Represents the intensity and activity level of a track.
        - **Tempo**: The speed or pace of a track (beats per minute).
        - **Acousticness**: Measures the likelihood of a track being acoustic.
        - **Valence**: Captures the musical positivity conveyed by a track.
        - **Instrumentalness**: Predicts whether a track contains no vocals.
        - **Liveness**: Detects the presence of a live audience.
        - **Speechiness**: Measures the presence of spoken words.
        - **Loudness**, **Key**, and **Mode**: Additional track-level attributes.
    - Supports fetching tracks in bulk from playlists using playlist IDs.
- **Functionalities**:
    - Dynamic data retrieval for tracks not present in the local dataset.
    - Ensures real-time access to Spotify's extensive music catalog.

2. Data Handling and Preprocessing

The system ensures high-quality data preparation for model training and recommendations.

- **Features**:
    - Merges multiple datasets (e.g., **df1** from API and **df2** from curated sources) into a unified dataset.
    - Preprocesses data by:
        - Handling missing values.
        - Removing duplicate tracks.

        - Normalizing numerical features (e.g., danceability, energy) using **StandardScaler**.
        - Applying dimensionality reduction techniques like **PCA, for PCA with KNN**.

- **Functionalities**:
    - Maintains a clean and scalable dataset for efficient model training.
    - Updates the dataset dynamically as new tracks are added.

3. Machine Learning Models

The system leverages advanced machine learning techniques to provide accurate and diverse recommendations.

Core Models:

1. **K-Nearest Neighbors (KNN)**:
    - Identifies tracks most similar to a given song based on proximity in the feature space.
    - Ensures precise recommendations by analyzing feature similarity.
2. **Cosine Similarity**:
    - Calculates similarity between tracks using the cosine of the angle between their feature vectors.
    - Ideal for finding subtle correlations between tracks.
3. **DBSCAN Clustering**:
    - Groups tracks into clusters based on density in the feature space.
    - Diversifies recommendations by including tracks from different clusters.
4. **Principal Component Analysis (PCA)**:
    - Reduces dimensionality of the dataset, improving computational efficiency.
    - Works alongside KNN for faster and more accurate recommendations.

- **Functionalities**:
    - Allows for diverse recommendation styles:
        - **Feature Proximity**: Suggests tracks similar in audio characteristics.
        - **Cluster-Based Diversity**: Includes tracks from varying musical styles and moods.
    - Ensures real-time adaptability by retraining models with updated datasets.

4. Recommendation Engine

The core functionality of the system is to generate personalized music recommendations.

- **Features**:
    - Accepts user input in the form of a **track name** and **artist**.
    - Searches for the input track in the local dataset or fetches it dynamically from Spotify.
    - Matches features of the input track with other tracks in the dataset to generate recommendations.
- **Functionalities**:
    - Generates a ranked list of recommended tracks based on:
        - Feature similarity.
        - Cluster membership (for diversity).
    - Provides additional metadata for each recommended track, such as artist, album, and Spotify popularity score.

5. User Interface

The system includes an interactive, user-friendly interface designed using **Streamlit**.

- **Features**:
    - **Track Search**:
        - Allows users to input a track name and artist to retrieve recommendations.
        - Supports real-time fetching of missing tracks from Spotify.
    - **Recommendation Display**:
        - Lists recommended tracks along with key metadata.
        - Provides links to open recommended tracks directly on Spotify.
    - **Dataset Update**:
        - Enables users to add new tracks dynamically.
        - Automatically updates and retrains models with the new data.
    - **Visual Insights**:
        - Displays **word clouds** of popular tracks or genres.
        - Shows **bar charts** highlighting the most popular recommended tracks.
- **Functionalities**:
    - Simplifies interaction with the recommendation system.

    - Visualizes insights to enhance the user experience.

6. Scalability and Real-Time Adaptability

The system is designed to handle large-scale datasets and adapt dynamically to new inputs.

- **Features**:
    - Processes large datasets efficiently using optimized algorithms and data structures.
    - Dynamically retrains models as new tracks are added to the dataset.
- **Functionalities**:
    - Ensures the system remains up-to-date and accurate with real-time data.
    - Handles growing music libraries seamlessly.

7. Cold-Start Problem Handling

The system addresses the challenge of recommending songs for new users or tracks.

- **Features**:
    - Utilizes Spotify's rich playlist data to provide recommendations even without prior user interaction.
    - Ensures a diverse and comprehensive dataset by including tracks from various playlists.
- **Functionalities**:
    - Provides high-quality recommendations for new users and tracks with minimal interaction history.

8. Evaluation and Validation

The system is equipped with metrics to assess the quality of recommendations.

- **Features**:
    - Evaluation Metrics:
        - **Euclidean Distance**: Measures similarity between recommended tracks and the input track.
        - **Feature Deviations**: Quantifies how well recommendations align with input track features.
- **Functionalities**:
    - Compares model performances to ensure optimal recommendations.
    - Provides insights into system accuracy and diversity.

Implementation Details

The Music Recommendation System was implemented by integrating data collection, preprocessing, machine learning, and an interactive user interface to deliver a robust and scalable solution for music recommendations. The following sections provide a detailed explanation of the implementation process:

1. Data Acquisition and Integration

Data for the project was sourced from two primary channels:

1. **Spotify API**:
   - Using the Spotipy library, real-time data was retrieved from over 30 Spotify playlists such as *RapCaviar*, *Indie India*, and *Rock Classics*.
   - The data included 4,341 unique tracks, covering diverse genres like global top charts, regional hits, and electronic dance music.
   - The Spotify API provided detailed audio features, including *danceability*, *energy*, *tempo*, *loudness*, and *valence*, which were essential for building recommendation models.
2. **Kaggle Dataset**:
   - This dataset contained metadata and audio features for over **600,000 tracks**, spanning a timeline from **1921 to 2020**.
   - It included historical and modern songs across various genres, allowing for a diverse representation of music styles and trends.
   - The dataset contributed to:
     - Enriching the Spotify API data with additional tracks not available in the playlists.
     - Providing a broad feature set and metadata to train and validate machine learning models.
   - Only relevant features (e.g., *track IDs*, *numerical audio features*, and *metadata*) were extracted and merged with the Spotify data. Duplicate and irrelevant rows were dropped to maintain quality.

2. Data Preprocessing

- **Standardization**:
    - Numerical features such as *danceability*, *energy*, *tempo*, *loudness*, and *speechiness* were scaled using StandardScaler. This ensured that all features contributed equally to similarity calculations and clustering.
- **Handling Missing Values**:
    - Missing or incomplete rows were removed from both datasets to maintain data integrity.
- **Feature Selection**:
    - Nine audio features were used for modeling: *danceability*, *energy*, *tempo*, *loudness*, *speechiness*, *acousticness*, *instrumentalness*, *liveness*, and *valence*. These were chosen based on their relevance to track similarity and user preferences.

3. Model Implementation

The system implemented four machine learning techniques:

1. **K-Nearest Neighbors (KNN)**:
    - Finds tracks most similar to the input song by comparing feature distances in a multi-dimensional space.
    - Tracks with the smallest Euclidean distances are recommended.
2. **Cosine Similarity**:
    - Measures angular similarity between feature vectors, emphasizing the relationship between tracks independent of magnitude differences.
3. **PCA + KNN**:
    - Reduces dataset dimensionality using Principal Component Analysis (PCA) while preserving key patterns.
    - KNN is applied to the reduced dataset for faster computation and effective recommendations.
4. **DBSCAN Clustering**:
    - Groups tracks into clusters based on feature density.

    - Recommendations are made from the same cluster as the input track, ensuring diversity while accommodating tracks with fewer neighbors.

4. Dynamic Dataset Expansion

- For tracks not already present in the dataset, the system dynamically fetches them from the Spotify API in real-time.
- Newly fetched audio features are standardized and appended to the dataset, ensuring the system remains current and mitigates the cold-start problem.

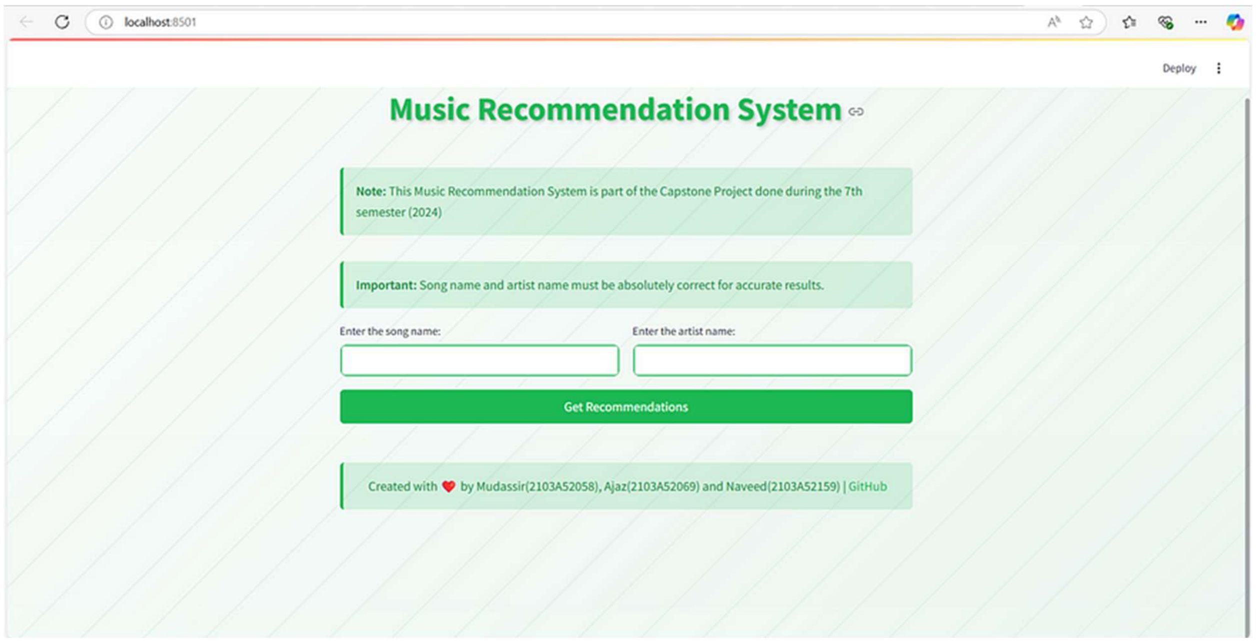
5. User Interface

- Built using **Streamlit**, the UI allows users to:
    - Input a song name and artist to fetch Spotify track details, including album cover, Spotify link, and metadata.
    - Receive recommendations interactively from all four models, with track metadata and links displayed.

Visualize track similarities and explore recommendations using real-time, dynamically updated data

6. Results

Figures 2 to 5 are the scrrenshots of project in run using streamlit in localhost. One important thing is the user must type track and artist name correctly. Recommendations are shown using streamlit containers, one for each model. Containers are openable and closable.


localhost:8501

Deploy

Music Recommendation System

Note: This Music Recommendation System is part of the Capstone Project done during the 7th semester (2024)

Important: Song name and artist name must be absolutely correct for accurate results.

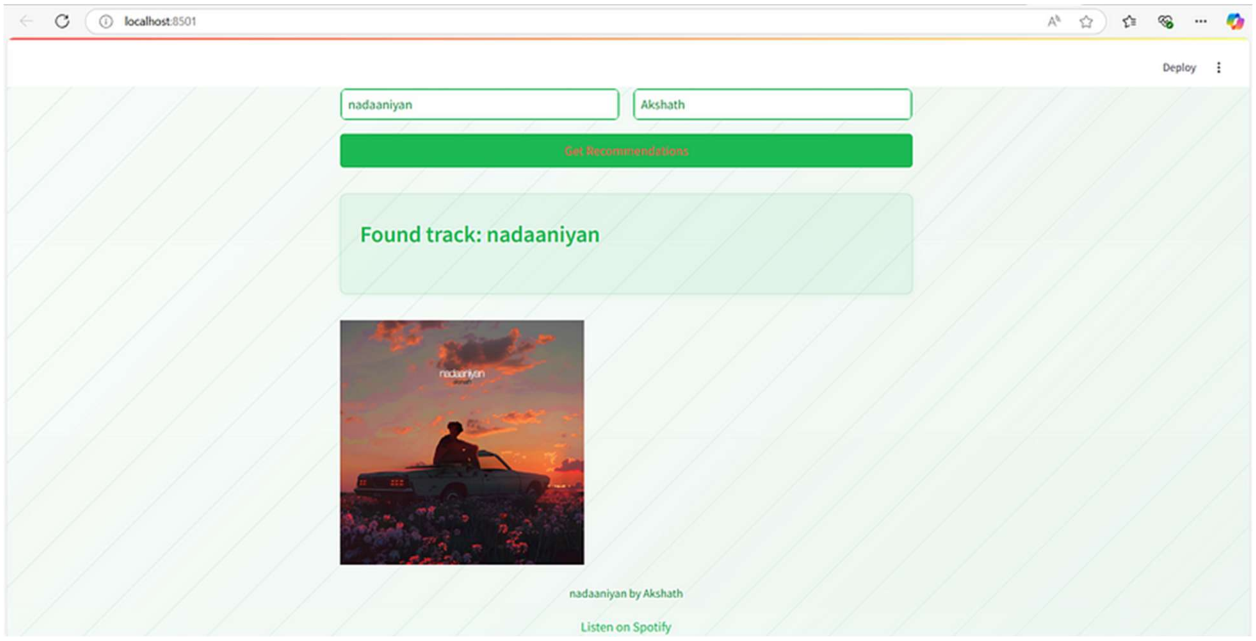
Enter the song name:

Enter the artist name:

Get Recommendations

Created with ❤️ by Mudassir(2103A52058), Ajaz(2103A52069) and Naveed(2103A52159) | GitHub

Figure 2: User Interface


localhost:8501

Deploy

nadaaniyan

Akshath

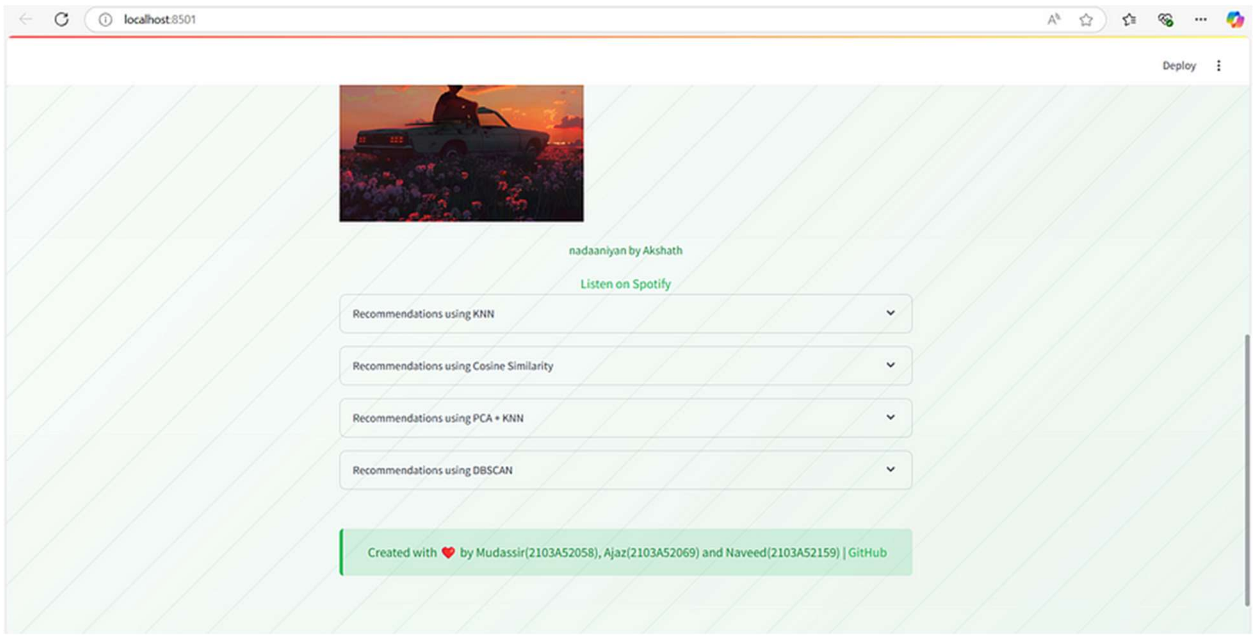
Get Recommendations

Found track: nadaaniyan

nadaaniyan by Akshath

Listen on Spotify

Figure 3: After giving input (track and artist name)


localhost:8501

Deploy

nadaaniyan by Akshath

Listen on Spotify

Recommendations using KNN

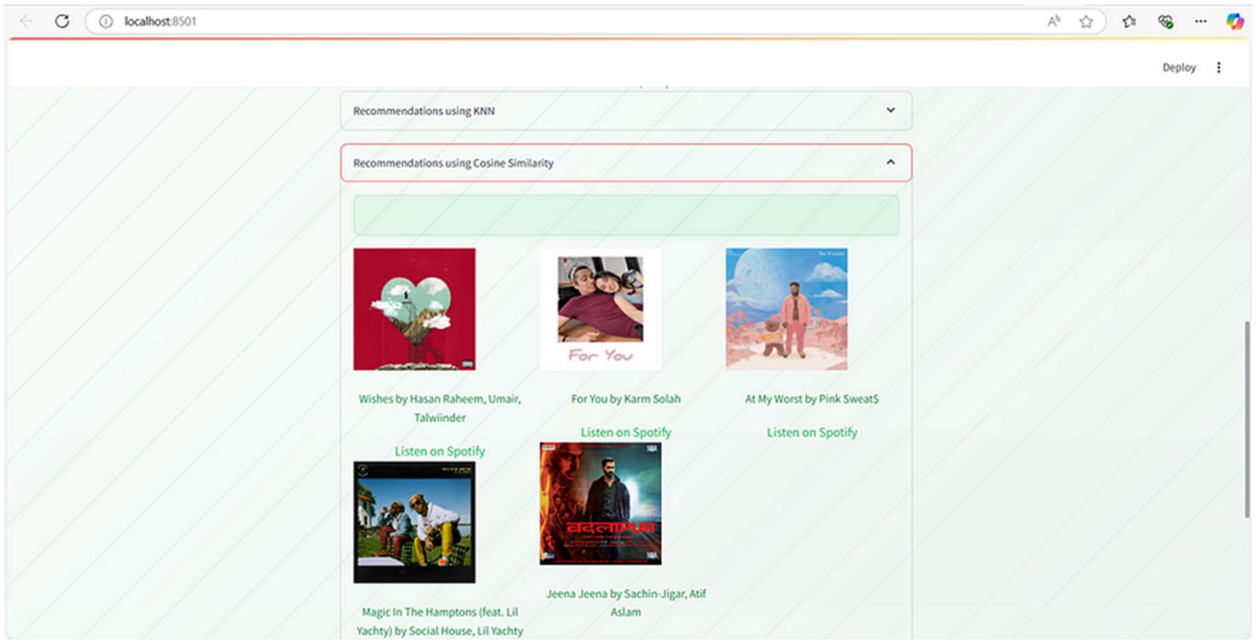
Recommendations using Cosine Similarity

Recommendations using PCA + KNN

Recommendations using DBSCAN

Created with ❤ by Mudassir(2103A52058), Ajaz(2103A52069) and Naveed(2103A52159) | GitHub

Figure 4: Recommendations are generated as containers for each model


localhost:8501

Deploy

Recommendations using KNN

Recommendations using Cosine Similarity

Wishes by Hasan Raheem, Umair, Talwiinder

Listen on Spotify

For You by Karm Solah

Listen on Spotify

At My Worst by Pink Sweat$

Listen on Spotify

Magic In The Hamptons (feat. Lil Yachty) by Social House, Lil Yachty

Jeena Jeena by Sachin-Jigar, Atif Aslam

Figure 5: Recommendations given by Cosine Similarity in its container

Challenges and Solutions

1. Data Integration Challenges

- **Challenge**:
  The datasets from Spotify API and Kaggle had different structures and formats, with overlapping but inconsistent features such as track IDs and metadata. Ensuring compatibility between the two sources required significant preprocessing.
- **Solution**:
  - Common columns like track IDs, audio features, and metadata were identified for merging.
  - Duplicates were removed using track IDs as a unique identifier.
  - Missing values were handled by filtering incomplete rows.
  - Audio features from both sources were standardized using StandardScaler to ensure uniform feature representation.

2. Real-Time Data Fetching

- **Challenge**:
  Tracks not present in the local dataset required real-time retrieval from the Spotify API, which introduced latency. Additionally, dynamically fetched data needed to be seamlessly integrated without disrupting ongoing operations.
- **Solution**:
  - Implemented a dynamic dataset expansion mechanism. When a song was missing, its details and audio features were fetched on-demand using the Spotify API.
  - Fetched data was immediately standardized and appended to the dataset, allowing real-time recommendations without delays.

7. API Rate Limits

- **Challenge**:
  The Spotify API has strict rate limits, which could lead to delays or failed requests during intensive use.

- **Solution**:
    - Implemented caching for API responses to reduce redundant requests.
    - Minimized API calls by maintaining a local dataset that dynamically updated only when necessary.

8. Evaluation of Models

- **Challenge**:
  The dataset lacked an output or target column and did not contain any user-specific interaction data, making traditional regression or classification evaluation metrics inapplicable. The absence of user preferences also ruled out metrics like precision, recall, or mean squared error (MSE), commonly used in recommendation systems.
- **Solution**:
    - The evaluation focused on the **audio features** used for similarity and clustering, relying on metrics that aligned with the methods implemented in the project.
    - **Evaluation Metrics Used**:
        1. **Euclidean Distance**:
            - Measures the straight-line distance between the feature vector of the input track and those of recommended tracks.
            - Smaller distances indicated closer similarity, helping validate models like KNN and Cosine Similarity.
        2. **Feature Deviation**:
            - Calculates the average deviation across key audio features (*danceability*, *energy*, *tempo*, etc.) between the input track and recommended tracks.
            - Ensured that recommendations preserved core characteristics of the input track.
        3. **Cluster Consistency**:
            - For models like DBSCAN, recommendations were evaluated based on the cluster memberships. Tracks within the same cluster as the input were compared for their diversity and relevance.
- **Why These Metrics?**

    - These methods were directly aligned with the project’s approach, which relied on audio feature similarity rather than user preferences or explicit feedback.
    - They provided an interpretable way to compare models in terms of their ability to recommend tracks based on intrinsic properties rather than collaborative or historical data.

The absence of an output column or labelled data meant the focus was on feature-based evaluation, making these metrics the most suitable choice

Conclusion

The Personalized Music Recommendation System developed in this project demonstrates the power of content-based filtering and machine learning models in providing relevant and diverse song suggestions. By combining real-time data from the Spotify API with an extensive Kaggle dataset, the system was able to build a comprehensive recommendation engine without relying on user-specific data.

Four different models—K-Nearest Neighbors, Cosine Similarity, PCA with KNN, and DBSCAN—were implemented and compared. The results showed that each model has strengths in certain scenarios, with KNN and Cosine Similarity excelling at precise, feature-based recommendations, and DBSCAN providing diversity by clustering songs into meaningful groups.

The project addresses several challenges inherent in music recommendation, including the cold-start problem, scalability, and data integration. However, there is room for future improvements, particularly in incorporating user feedback and enhancing the personalization aspect through hybrid models and user interaction data.

Overall, this project serves as a robust foundation for building scalable and adaptive music recommendation systems that can evolve with real-time data, making it highly adaptable for future development and expansion into other platforms.

Future Scope

This project successfully developed a Music Recommendation System by leveraging audio features from the Spotify API and a Kaggle dataset. Four machine learning models were implemented—KNN, Cosine Similarity, PCA with KNN, and DBSCAN—providing diverse approaches to song recommendations. The system also tackled the cold-start problem and allowed dynamic dataset expansion with real-time data fetching.

Looking ahead, several enhancements can be made to further improve the system:

1. **Incorporating User Interaction Data**: Adding user preferences like listening history and ratings could personalize recommendations.
2. **Hybrid Recommendation System**: Combining collaborative filtering with content-based methods can improve accuracy and diversity.
3. **Real-Time Personalization**: Introducing feedback mechanisms, such as thumbs up/down, would allow the system to adapt dynamically.
4. **Expanding Data Sources**: Integrating platforms beyond Spotify, like YouTube Music or SoundCloud, would broaden the track catalog.
5. **Improved Clustering Techniques**: Experimenting with alternative clustering algorithms (e.g., K-Means) could yield better song groupings.
6. **Enhanced Audio Analysis**: Advanced features like mood or emotion detection through deep learning could enrich recommendations.
7. **Mobile App Integration**: Developing a mobile version would increase accessibility and user engagement.

These advancements would make the system more scalable, adaptive, and aligned with user expectations in a dynamic music-streaming environment

References

[1] Kostrzewa, D., Chrobak, J., & Brzeski, R. (2024). Attributes relevance in content-based music recommendation system. Applied sciences, 14(2), 855. (https://www.mdpi.com/2076-3417/14/2/855)

[2] Deldjoo, Y., Schedl, M., & Knees, P. (2024). Content-driven music recommendation: evolution, state of the art, and challenges. Computer science review, 51, 100618. (https://www.sciencedirect.com/science/article/pii/S1574013724000029)

[3] Girsang, A. S., & Wibowo, A. (2021). Neural collaborative for music recommendation system. IOP Conference Series: Materials Science and Engineering, 1071(1), 012021. (https://iopscience.iop.org/article/10.1088/1757-899X/1071/1/012021/meta)

[4] Bi, X., Qu, A., & Shen, X. (2018). Multilayer tensor factorization with applications to recommender systems. (https://projecteuclid.org/journals/annals-of-statistics/volume-46/issue-6B/Multilayer-tensor-factorization-with-applications-to-recommender-systems/10.1214/17-AOS1659.full)

[5] Zhang, Y., Bi, X., Tang, N., & Qu, A. (2020). Dynamic Tensor Recommender Systems. IEEE Transactions on Knowledge and Data Engineering, 32(5), 920-934. (https://www.jmlr.org/papers/v22/19-792.html)

[6] Bi, X., Tang, X., Yuan, Y., Zhang, Y., & Qu, A. (2021). Tensors in statistics. Annual review of statistics and its application, 8(1), 345-368. (https://www.annualreviews.org/content/journals/10.1146/annurev-statistics-042720-020816)

[7] Adomavicius, G., & Tuzhilin, A. (2010). Context-aware recommender systems. In Recommender systems handbook (pp. 217-253). Boston, MA: Springer US. (https://link.springer.com/chapter/10.1007/978-0-387-85820-3_7)

[8] Aggarwal, C. C., & Aggarwal, C. C. (2016). Context-sensitive recommender systems. Recommender systems: The textbook, 255-281. (https://link.springer.com/chapter/10.1007/978-3-319-29659-3_8)

[9] Gediminas, A., & Alexander, T. (2015). Recommender Systems Handbook. Springer. (https://doi.org/10.1007/978-1-4899-7637-6)

[10] Xian, Y., Fu, Z., Muthukrishnan, S., De Melo, G., & Zhang, Y. (2019, July). Reinforcement knowledge graph reasoning for

explainable recommendation. In Proceedings of the 42nd international ACM SIGIR conference on research and development in information retrieval (pp. 285-294). (https://dl.acm.org/doi/abs/10.1145/3331184.3331203)

Standards and API Documentation:

1. Spotify API Documentation: For programmatically retrieving data such as tracks, albums, and artist details. https://developer.spotify.com/documentation/

2. Spotipy Library Documentation- A Python wrapper around the Spotify Web API for easier interaction with Spotify data in Python projects.:https://spotipy.readthedocs.io/

Datasets

1. Spotify Dataset 1921-2020, 600k+ Tracks: https://www.kaggle.com/datasets/yamaerenay/spotify-dataset-19212020-600k-tracks

Software and Libraries

1. Scikit-learn: Machine Learning in Python: https://scikit-learn.org/stable/

2. Pandas: For data manipulation, cleaning, and analysis of the dataset.: https://pandas.pydata.org/

3. Streamlit Documentation. - Web app framework for machine learning and data science projects. https://docs.streamlit.io/