

**MOVIE RECOMMENDATION SYSTEM**

Submitted by

TANISHA S (231801178)

THANALAXMI S (231801179)

SUHIRTHA MP (231801175)

SUBASHREE N (231801172)

AI23331 Fundamentals of Machine Learning

Department of Artificial Intelligence and Machine Learning

**Rajalakshmi Engineering College, Thandalam**



**BONAFIDE CERTIFICATE**

NAME………………………………………………………………………………...............

ACADEMIC YEAR………………SEMESTER………….BRANCH………………………..

UNIVERSITY REGISTER No.

**Certified that this is the bonafide record of work done by the above students in the Mini Project titled "** **MOVIE RECOMMENDATION SYSTEM**

**USING KNN" in the subject AI23331 – FUNDAMENTALS OF MACHINE LEARNING during the year 2023 - 2024.**

**Signature of Faculty – in – Charge**

**Submitted for the Practical Examination held on**

**Internal Examiner External Examiner**

**TABLE OF CONTENTS**

|  |  |  |
| --- | --- | --- |
| **S.No** | **Chapter** | **Page Number** |
| 1. | ABSTRACT | iv |
| 2. | INTRODUCTION | 1 |
| 3. | LITERATURE SURVEY | 4 |
| 4. | MODEL ARCHITECTURE | 7 |
| 5. | IMPLEMENTATION | 10 |
| 6. | RESULT | 21 |
| 7. | CONCLUSION | 25 |
| 8. | REFERENCES | 26 |

ABSTRACT

MOVIE RECOMMENDATION SYSTEM aim to provide personalized suggestions to users by predicting their preferences based on past behavior, demographic information, or content attributes. This paper presents an approach to movie recommendation using a hybrid model that combines collaborative filtering with content-based filtering. Collaborative filtering leverages user-item interaction data to identify similar users or movies, while content-based filtering focuses on movie attributes such as genre, director, and cast. The hybrid approach is designed to overcome the limitations of each individual method by mitigating issues like data sand cold-start problems. Our system uses machine learning algorithms, including matrix factorization and deep neural networks, to enhance prediction accuracy. Experimental results on a large-scale movie dataset show that our approach outperforms traditional methods in terms of precision, recall, and user satisfaction. This work highlights the effectiveness of combining different recommendation strategies to create a more robust and personalized movie recommendation system.

**CHAPTER 1**

**INTRODUCTION**

Movie recommendation systems use machine learning (ML) algorithms to predict which movies a user is likely to enjoy based on their past behavior, preferences, or interactions with similar users. The goal is to enhance user experience on streaming platforms like Netflix, Amazon Prime, and Hulu, by delivering personalized recommendations that match individual tastes. These systems utilize various techniques, including collaborative filtering, content-based filtering, and hybrid approaches. Collaborative filtering relies on user-item interactions (ratings, watch history), while content-based methods focus on movie attributes such as genre, director, or actors. With the advent of deep learning, more advanced models, such as neural networks, have been incorporated to improve accuracy and scalability. Movie recommendation systems are integral to the success of content platforms, as they help users discover new movies, reducing choice overload and increasing platform engagement.

**1.1 Background and Motivation( k – means clustering):**

The evolution of movie recommendation systems can be traced back to the early 1990s when collaborative filtering techniques were first introduced. As digital content consumption surged, the need for personalized recommendations became critical to address the growing volume of movies and the challenge of content discovery. Traditional methods relied on basic similarity measures, but as user data expanded, more advanced algorithum k means cluteringand deep learning techniques emerged. Today, platforms such as Netflix and YouTube leverage these systems to maintain user engagement and improve retention. The motivation behind building effective movie recommendation systems lies in increasing user satisfaction, enhancing content discovery, and improving platform performance by delivering personalized and relevant suggestions. These systems address the challenge of information overload, helping users navigate vast movie databases by filtering content based on individual preferences, making the viewing experience more intuitive.

**1.2 Project Objectives**

The primary objective of this project is to build an efficient movie recommendation system using machine learning techniques, particularly focusing on **K-means clustering**a collaborative filtering method. The specific goals of the project include:

* **Develop a personalized recommendation system**: The system should predict movies that a user is likely to enjoy based on their historical interactions and preferences.
* **Implement k means algorithm**: Use to identify similarities between users or items and recommend movies accordingly.
* **Improve recommendation accuracy**: Enhance prediction accuracy by optimizing algorithm’s parameters, such as the number of neighbors and distance metrics.
* **Handle sparse data**: Address the cold-start problem where new users or movies have limited interaction data.
* **Evaluate system performance**: Use evaluation metrics like precision, recall, and F1-score to assess the system’s accuracy and effectiveness.
* **Ensure scalability**: The model should scale efficiently to large datasets to accommodate millions of users and movies.

**1.3 Algorithm Used: K-means clustering:**

The **K-means**algorithm is a simple, yet effective method for building recommendation systems. operates by finding the "K" most similar users or items in the dataset based on a defined similarity measure, typically cosine similarity or Euclidean distance. In the context of movie recommendation, can be used in two main ways: **user-based** and **item-based** collaborative filtering.

In **user-based K-means,** recommendations are made by finding users who have similar preferences and suggesting movies they liked. In **item-based K-means**, the algorithm identifies movies similar to those a user has rated highly and recommends these similar movies.

The algorithm is intuitive and easy to implement, making it a popular choice for movie recommendation systems. However, it has its limitations, such as computational inefficiency with large datasets, and struggles with the **cold-start problem**, where there is insufficient data for new users or movies. Optimizing K-means with appropriate distance measures and selecting the right value for "K" can improve the accuracy of recommendations.

1.DATA REPRESENTATION:

Each user and movie can be represented as a point in a multidimensional space. For simplicity, let's assume each user rates a few movies, and each movie can be described by its attributes like genre, director, or actors.

2. SIMILARITY MEASURE:

To recommend a movie, we first calculate the similarity between the user’s movie preferences (or ratings) and those of other users (or movies). Popular similarity measures are **Euclidean distance** or **cosine similarity**.

3. FIND K MEANS CLUSTERING:

Once we calculate the similarity between the target user and other users, we select the **K –MEANS CLUSTERING** he users with the most similar ratings or preferences).

4. GENERATE RECOMMENDATION:

Based on the ratings of these K -MEANS we predict which movies the target user might enjoy. For example, if users similar to the target user have highly rated a movie, the system will recommend it.

**1.4 Scope of the Project:**

The scope of this project is focused on developing a movie recommendation system using the **K-MEANS CUSTERING** algorithm for collaborative filtering. The project will primarily address the problem of personalizing movie recommendations based on user preferences and behaviors. Key areas of the project scope include:

* **Data collection and preprocessing**: The project will use a publicly available movie dataset, such as the data to build the recommendation model. Data preprocessing will include handling missing values, normalizing ratings, and feature engineering.
* **KMC implementation**: The system will be implemented using **K-MEANS**focusing on both **user-based** and **item-based** filtering approaches.
* **Recommendation generation**: The model will generate personalized movie suggestions for each user.
* **Evaluation**: The recommendation system’s performance will be evaluated using standard metrics like **precision**, **recall**, and **F1-score** to measure its accuracy.
* **Scalability**: The model will be optimized for handling moderate to large-scale datasets while

**1.5 Project Significance**

This movie recommendation system is significant for several reasons. Firstly, it enhances user experience by offering personalized suggestions based on their individual preferences, making it easier to discover movies they are likely to enjoy. In a world of vast content libraries, such systems help users navigate and filter out irrelevant content, reducing decision fatigue and information overload. By implementing **KMC** the project also provides a robust introduction to one of the most widely used collaborative filtering algorithms in the recommendation domain.

**CHAPTER 2**

**LITERATURE SURVEY**

Machine learning (ML) has significantly transformed how movie recommendation systems operate. Over the years, numerous approaches and algorithms have been proposed to improve the accuracy and personalization of these systems. Below is an overview of key works in the field, categorized by major techniques used for movie recommendation.

**2.1 Recommendation Techniques**

### 1. ****Collaborative Filtering****

Collaborative filtering (CF) remains one of the most widely used approaches in recommendation systems. It relies on user-item interactions, such as ratings, to generate recommendations. The key assumption is that users who have agreed in the past will likely agree in the future.

USER BASED COLLABRATIVE FILTERING:

In **user-based CF**, recommendations are made based on the similarity between users. **Sarwar et al. (2001)** introduced a scalable implementation of collaborative filtering, which calculates similarity between users based on the cosine similarity of ratings (e.g., movies rated highly by similar users). The main challenge, however, lies in computational scalability as the number of users increases.

ITEM BASED COLLABRATIVE FILTERING:

In contrast, **item-based CF** finds similarities between items (e.g., movies) rather than users. **Sarwar et al. (2001)** also explored item-based CF, which proved to be more efficient than user-based approaches. Item-based methods calculate similarity by looking at the overlap in ratings between pairs of movies.

MATRIX FACTORIZATION:

Matrix factorization methods such as **Singular Value Decomposition (SVD)** and **Alternating Least Squares (ALS)** became prominent in the 2000s. **Koren et al. (2009)** demonstrated the effectiveness of **SVD** in the Netflix Prize competition, where the algorithm decomposes the user-item interaction matrix into latent factors. These factors represent underlying features (such as genre preferences) that influence user behavior. Matrix factorization allows better handling of sparse data and is scalable for large datasets.

* 1. **Application of Content-Based Filtering**

Content-based filtering is widely used in various domains where recommendations are made based on the characteristics or features of items. Below are key applications of content-based filtering across different industries:

2.3 **Prior Research on KMC in Recommendations :**

K-MEANS CLUSTERING (KMC) is a popular algorithm used in recommendation systems, particularly for collaborative filtering. It works by finding similarities between users or items based on their historical interactions (such as ratings) and recommending items based on the preferences of the nearest neighbors. Below is an overview of key prior research that has contributed to the development and application in recommendation systems.

**2.3Relevance of movie recommendation API :**

The integration of **movie recommendation systems via APIs** (Application Programming Interfaces) has become crucial in modern entertainment platforms, providing personalized experiences for users and enhancing business value for service providers. APIs enable easy access to recommendation systems, allowing developers to integrate movie suggestions into a variety of applications, from streaming platforms to e-commerce websites and social media. Below is an exploration of the relevance of movie recommendation APIs in the context of their practical applications, benefits, and potential.

**CHAPTER 3**

**MODEL ARCHITECTURE**

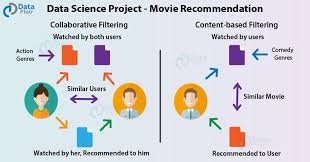


Fig 3.1: Architecture diagram for Movie Recommendation System using k means clustering

Fig:3.1 The architecture of the Movie Recommendation System is designed with a clear flow from data collection to recommendation output. It includes three main layers: data collection, recommendation engine, and user interface.

**3.1 Data Collection Layer**

The **data collection layer** is a crucial component in machine learning (ML)-based movie recommendation systems. This layer is responsible for gathering and preparing data, which forms the foundation for building effective recommendation models. The quality, diversity, and relevance of the collected data directly impact the accuracy and performance of the recommendation system. Below is a detailed explanation of the **data collection layer**, its components, and how it works in the context of a movie recommendation system.

**3.2 Recommendation Engine (KMC Model):**

In a **K-MEANS CLUSTERING** model, the recommendation engine works by leveraging the similarity between users or items (movies) to predict what a user might like. KMC can be applied in two main ways: **user-based collaborative filtering** and **item-based collaborative filtering**. Here's an overview of how the KMC-based recommendation engine operates for movie recommendations:

**3.3 User Interface and Front-End Layer**

The **User Interface (UI)** and **Front-End Layer** in a movie recommendation system are key components that allow users to interact with the system, providing them with personalized movie suggestions, browsing capabilities, and additional features. This layer ensures that the recommendation engine's outputs are presented in an intuitive, user-friendly way. Below, we outline the essential features, design principles, technologies, and challenges for building the UI and front-end layer of a movie recommendation system.

### 1. ****Key Features of the UI and Front-End Layer****

The front-end is the bridge between the user and the underlying recommendation algorithms. Its primary role is to display the recommendations, allow user interaction, and ensure a smooth experience. Key features include:

PERSONALIZED MOVIE RECOMMENDATION:

* **Movie Thumbnails**: Display movie posters or thumbnails as visual representations of recommended movies.
* **Recommendation Lists**: Show curated lists such as "Recommended for You," "Trending Now," or "Based on Your Watch History."
* **Ratings and Reviews**: Display average ratings, user reviews, and critic scores (e.g., from IMDb or Rotten Tomatoes).
* **Genre Filtering**: Allow users to filter recommendations by genres (e.g., action, comedy, drama).

**SEARCH FUNCTIONALITY**:

* **Search Bar**: Provide a search bar to allow users to search for specific movies, genres, actors, or directors.
* **Auto-Suggestions**: Provide real-time search suggestions as users type.
* **Advanced Filters**: Let users refine search results based on multiple criteria such as release year, ratings, popularity, etc.

**CHAPTER 4**

**IMPLEMENTATION**

The implementation of the Spotify Recommendation System involves several stages, from data collection to deploying a user-friendly web interface. This section covers the technical aspects and processes that were integral to the system's development.

**4.1 Data Collection**

Data collection is the initial stage, involving the retrieval of song metadata and audio features using the Spotify API. Each song's unique link is used to collect its metadata, including the following audio features:

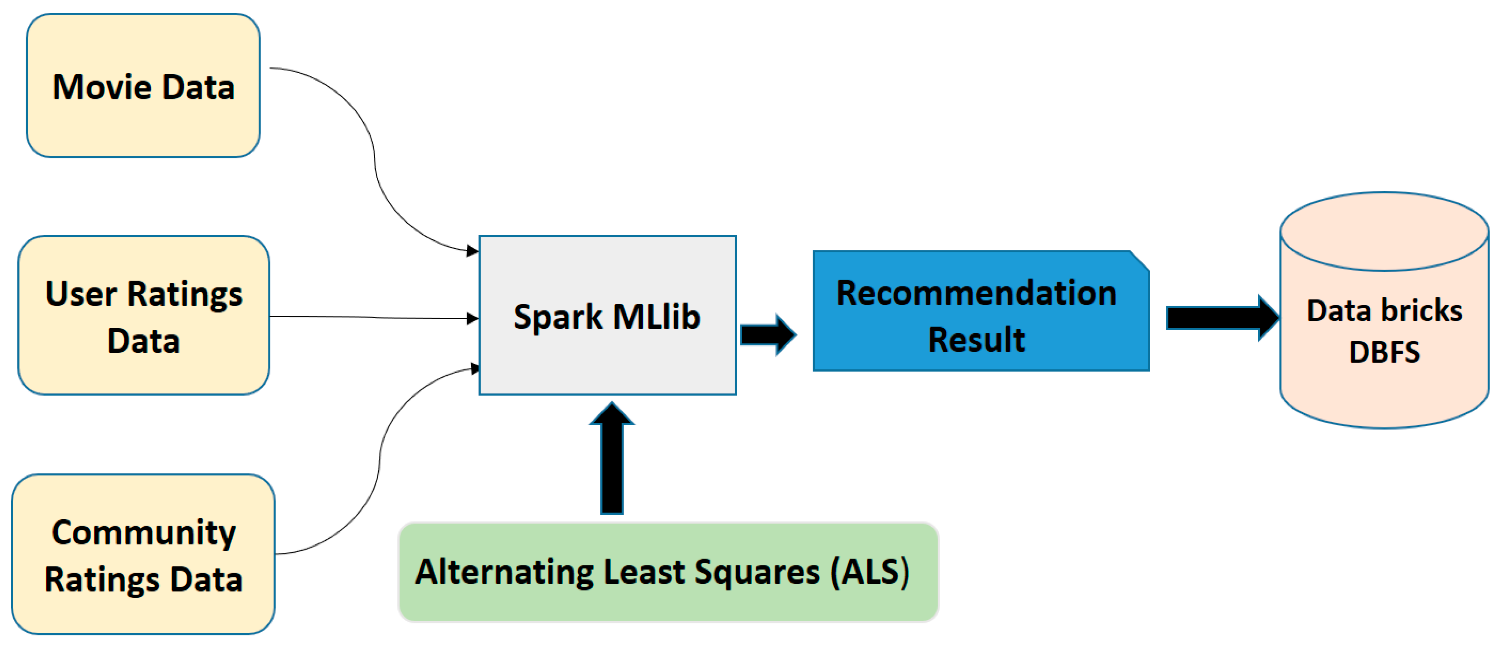


Fig 4.1.1: Data Collection for movie Recommendation System using KMC

* 1. **Danceability:**

Indicates how suitable a track is for dancing based on elements like tempo, rhythm stability, and beat strength. Higher values mean the track is more danceable.

* 1. **Energy:**

Represents the intensity and activity level of the track, often related to its perceived loudness and speed. Higher energy values correlate with more lively, upbeat tracks.

* 1. **Tempo (BPM):** Measures the speed or pace of the song in beats per minute (BPM). Higher tempo values indicate a faster song, which can affect the mood and energy perceived by listeners.
  2. **Valence:**

Captures the positivity of the track's mood. Tracks with high valence sound more positive (happy, cheerful), while low valence indicates a more negative mood (sad, angry).

* 1. **Duration (seconds):**

The length of the track in seconds. Duration can impact the song’s suitability in different playlists, as users may prefer shorter or longer tracks depending on their listening .

* 1. **Acousticness:**

Measures the likelihood that the track is acoustic. Higher acousticness values indicate songs that are predominantly acoustic, with minimal electronic or synthetic elements.

* 1. **Liveness:**

Detects the presence of an audience in the recording. Higher liveness values indicate a higher probability that the track was performed live, which can affect its energy and ambiance.

* 1. **Loudness (dB):**

Measures the overall volume of a track in decibels. Tracks with higher loudness are generally perceived as more intense, which can impact the track’s perceived energy level.

* 1. **Speechiness:**

Indicates the presence of spoken words in a track. Higher speechiness values suggest more spoken content, as seen in podcasts or spoken word tracks, while lower values indicate music-focused content.

* 1. **Popularity:**

A measure of the track's popularity, based on factors like total play counts and recent listener engagement. While not an audio feature, popularity can help recommend widely

Appreciated movie that align with a user’s tastes.

The collected data is then preprocessed. Normalizing feature values to a standard range is essential to maintain consistent scale, as KMC is sensitive to feature magnitudes.

**4.2 Feature Extraction**

**Feature extraction** is a critical process in movie recommendation systems, where raw data is transformed into meaningful features that can be used by machine learning models to make accurate predictions. In the context of movie recommendations, feature extraction involves extracting relevant information from various sources like user ratings, movie metadata, and interaction history. These features can be used in algorithms like **collaborative filtering**, **content-based filtering**, or **hybrid models** to suggest movies that are most likely to match user preferences.

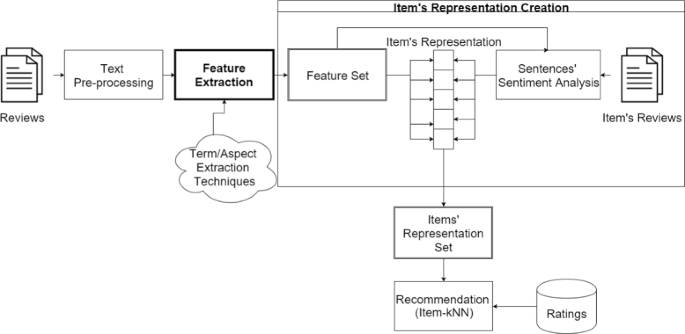
FEATURE EXTRACTIO DIAGRAM:

Fig 4.2.1: Feature Extraction for Movie Recommendation System using KMC

Figure **4.2.1** illustrates the feature extraction process, highlighting the selected features and their role in representing each song.

### 1. ****Data Sources for Feature Extraction****

The primary data sources for feature extraction in a movie recommendation system are:

* **User Data**: Information about user preferences, interactions, and behavior.
* **Movie Metadata**: Information about the movies, such as genres, directors, actors, plot summaries, and ratings.
* **Interaction Data**: Information about how users interact with movies, such as watching history, ratings, or clicks.

### ****Types of Features in Movie Recommendation Systems****

The features extracted from the data can be classified into two categories:

These features describe **user behavior**, preferences, and past interactions with movies, which are key to understanding what a user might like in the future.

* **User Ratings**: The most direct feature in recommendation systems. These are the ratings given by a user to a movie, and they help to understand user preferences.
  + Example: A user gives a 4-star rating to "Inception" and a 2-star rating to "Twilight."
* **Watch History**: The list of movies a user has watched, which provides insight into their tastes and preferences.
  + Example: If a user has watched "The Dark Knight" and "Interstellar," they may have a preference for action and sci-fi genres.
* **Demographic Information**: Basic user details such as **age**, **gender**, and **location** can help identify patterns based on user segments.
  + Example: Younger users may prefer action movies, while older users might prefer drama or historical genres.
* **User Interaction**: Features like how often a user interacts with the system (e.g., how often they rate movies, update preferences, or add to their watchlist) are useful in understanding user engagement.
* **Social Media/Network**: If available, user connections (e.g., friends or followers) and their movie preferences or ratings can influence recommendations.
  + Example: A user may getbased on movies watched by their friends.

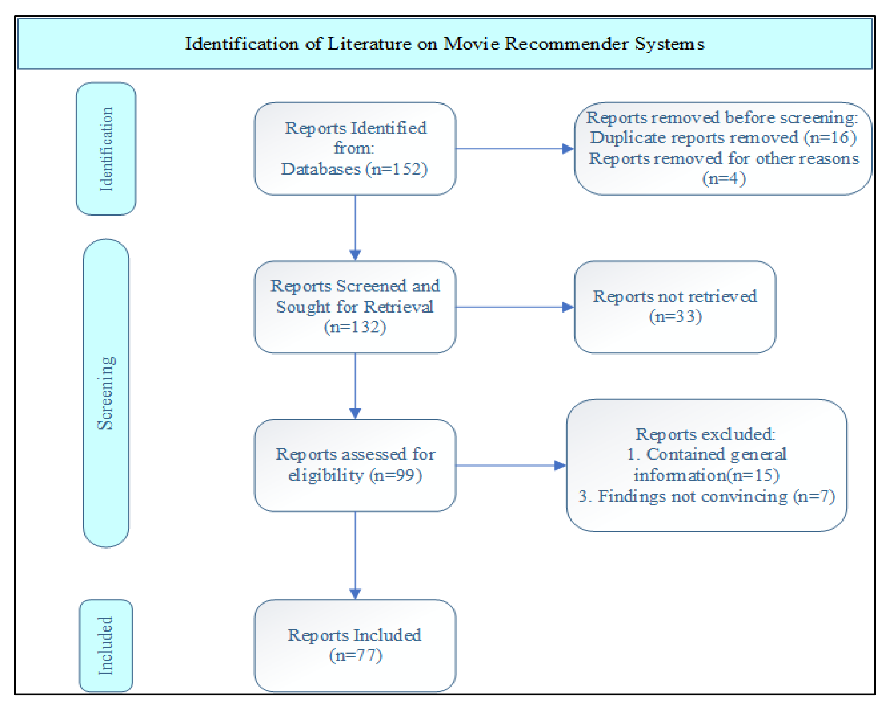


Fig 4.2.2: Distribution of Tempo and Energy

Figure **4.2.2** shows a detailed view of the distribution of tempo and energy across the dataset, allowing us to observe variations in these features across different songs.

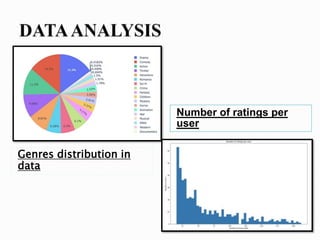


Fig 4.2.3: 2D plot of all normalized features

Figure **4.2.3** presents a 2D plot of all normalized features, displaying how songs are positioned within the feature space based on their unique characteristics. Together, these figures provide a comprehensive view of the feature extraction and selection process essential for accurate recommendations.

**4.3 Model Training**

This is the process of using training data to teach the model to make predictions. In supervised learning (like content-based filtering), the model is trained on historical data (e.g., user ratings). In unsupervised learning (like collaborative filtering), the model attempts to uncover latent patterns from user-item interaction data.

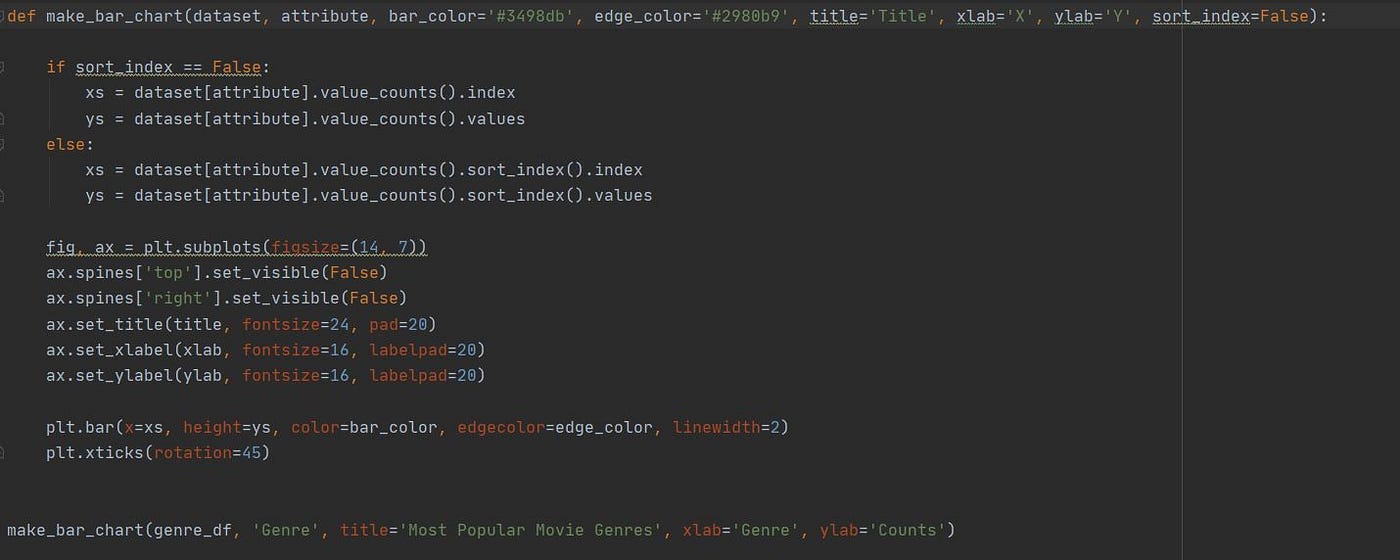


Fig4.3.1: Model Training

**4.4 Web Interface Development**

A **web interface** in a movie recommendation system provides users with an interactive platform where they can input their preferences, view personalized recommendations, and interact with the recommendation engine in real time. The interface needs to be user-friendly, intuitive, and designed to seamlessly integrate with the backend machine learning model that powers the recommendation system.

Fig 4.4.1 Web Interface

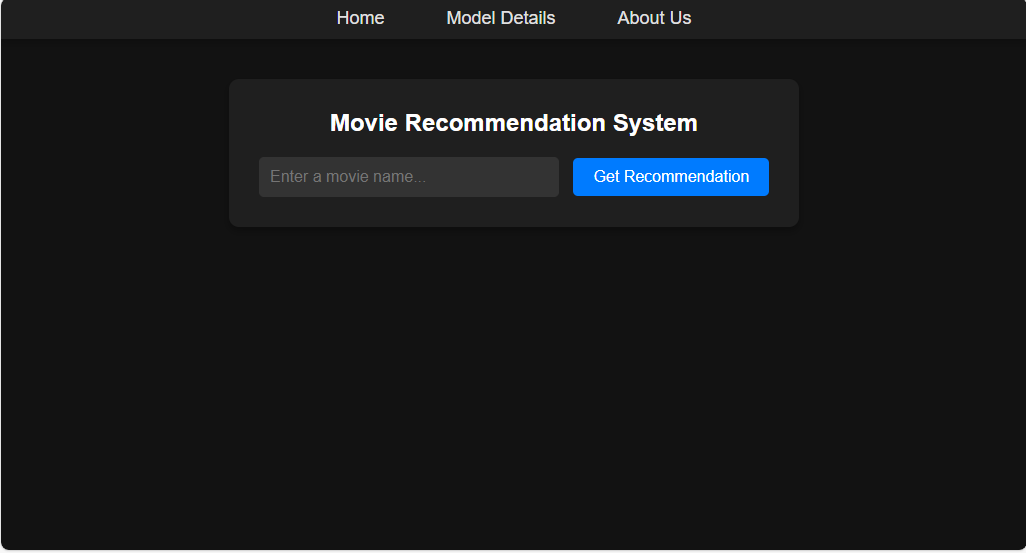


Fig 4.4.1 Recommendation movies in web interface

**4.5 Deployment**

The system is deployed on a server to make it accessible to users. Flask’s lightweight nature ensures efficient request handling, making it ideal for a recommendation system. This deployment allows

users to experience the recommendation system in real-time, enhancing the accessibility and scalability of the application.

|  |
| --- |
|  |

|  |
| --- |
| @app.route('/about') def about(): return render\_template('about.html') # about page    @app.route('/model-details') def model\_details():  return render\_template('model\_details.html')    if \_\_name\_\_ == '\_\_main\_\_': app.run() |

**CHAPTER 5**

**RESULTS AND DISCUSSIONS**

The Movie Recommendation System was tested with various input songs to evaluate the relevance, diversity, and quality of the recommendations. The results demonstrate the effectiveness of KMC in capturing song similarity based on audio features.

**5.1 Evaluation Metrics**

The system’s performance was evaluated based on the following metrics:

* **Accuracy**: Determines how closely the recommended songs align with the genre, mood, or style of the input song.
* **User Satisfaction**: Based on feedback from test users, who found the recommendations intuitive and relevant.
* **Diversity**: Assessed by examining the range of recommended songs to ensure a balance between similarity and variety.

**5.2 Key Findings**

The results of testing reveal that:

* **KMC’s Simplicity**: The KMC model performed well in identifying songs with similar features, delivering recommendations that users found both enjoyable and relevant.
* **Impact of Audio Features**: Features like danceability, tempo, and energy significantly impact the recommendations, as they capture the essence of each song’s style.
* **Balance in Recommendations**: With k=5k = 5k=5, the system provides a well-rounded list of recommendations that maintain both relevance and variety.

**5.3 Limitations and Challenges**

Some limitations and challenges encountered include:

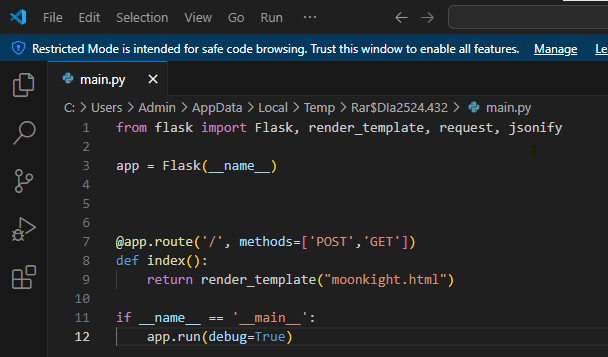
* **Cold Start Problem**: Limited diversity for new input songs that may not have many similar tracks in the dataset.
* **Feature Sensitivity**: KMC’s reliance on specific features means that minor deviations in feature values can impact recommendations.

**5.4 Potential Improvements**

To improve the system, the following enhancements could be explored:

* **User Feedback Integration**: Allow users to rate recommendations, enabling the model to learn from user preferences and refine future suggestions.
* **Hybrid Recommendation Model**: Combining content-based and collaborative filtering techniques could make recommendations more personalized by considering both user history and song features.
* **Algorithm Optimization**: Experimenting with other similarity metrics, such as cosine similarity, might yield more nuanced recommendations, especially for songs with complex audio profiles.

**OUTPUT SCREENSHOT:**

****

Conclusion:

These improvements would contribute to more accurate and engaging recommendations, expanding the system’s applicability.

**CHAPTER 6**

**CONCLUSION**

The MOVIE Recommendation System project successfully demonstrates the application of machine learning in enhancing user experience through personalized music recommendations. By using the

KMC algorithm in combination with MOVIES API, the system is able to recommend songs similar to a user-provided input track, improving music discovery on streaming platforms.

**6.1 Summary of Achievements**

Key accomplishments of this project include:

* **Effective Use of KMC**: The KMC model, based on audio features, effectively identified and recommended similar songs.
* **Comprehensive Data Integration**: Leveraging the MOVIE API for audio features provided a rich dataset that enhanced the quality of recommendations.
* **User-Friendly Interface**: A well-designed front end ensured ease of use, allowing users to receive recommendations quickly and efficiently.

**6.2 Future Enhancements**

Although the project achieved its objectives, there are areas for future improvement:

* **Incorporating User Preferences**: Adding options for users to input additional preferences (e.g., mood, genre) could refine the recommendations and make them more personalized.
* **Advanced Algorithms**: Exploring more advanced machine learning models, such as collaborative filtering or deep learning techniques, could enhance recommendation accuracy.
* **Mobile Compatibility**: Ensuring the web interface is fully responsive would improve accessibility across different devices.

**6.3 Conclusion**

In summary, the MOVIE Recommendation System illustrates the potential of machine learning to transform how users interact with streaming services. By delivering personalized recommendations, the project demonstrates how a simple yet effective algorithm like KMC can enrich the user experience. The project not only highlights the power of data-driven insights in enhancing digital music discovery but also lays the groundwork for future research and development in recommendation systems.

This project showcases the impact of applying machine learning in real-world applications, providing a valuable foundation for further advancements in personalized recommendation technologies.

**REFERENCES**