

angry).

5. 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 .

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

7. 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.

8. 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.

9. 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.

10.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:



9

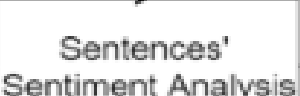
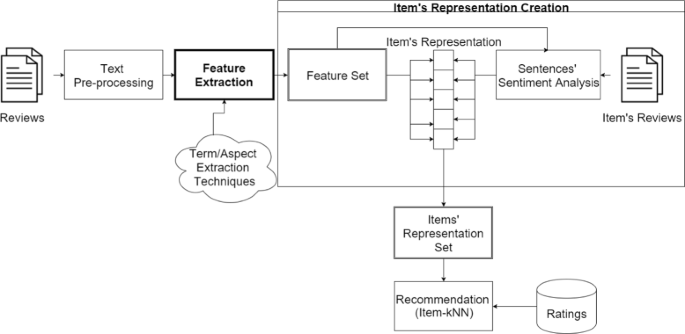
Fig 4.2.1: Feature Extraction for Movie Recommendation System using KMC

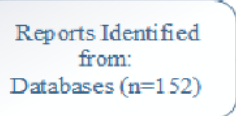
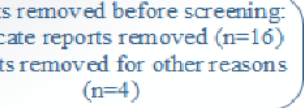
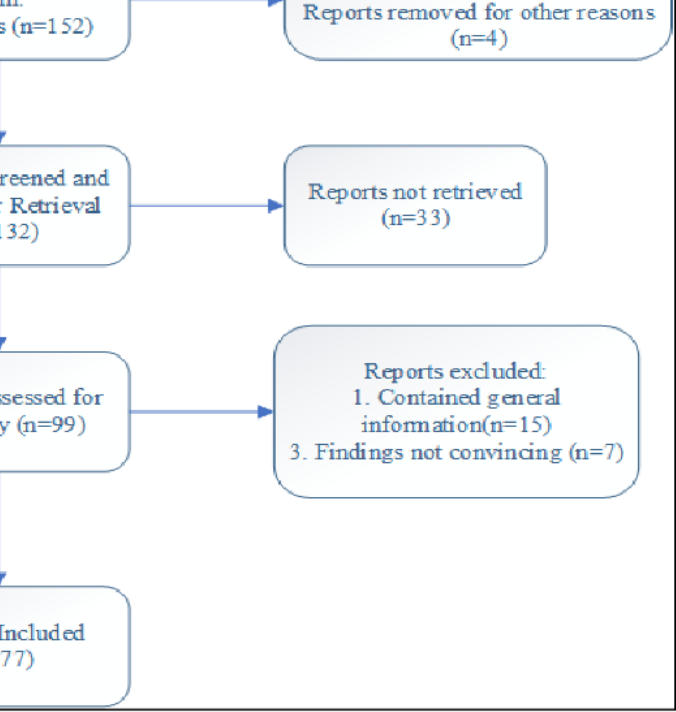
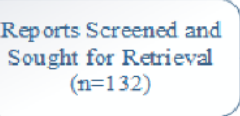
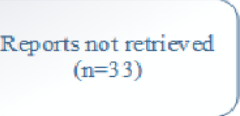
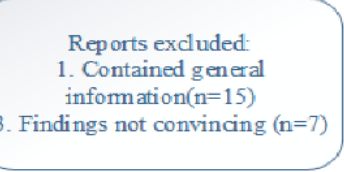
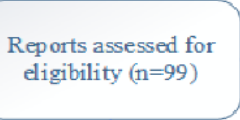
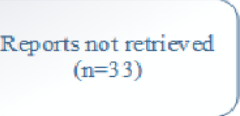
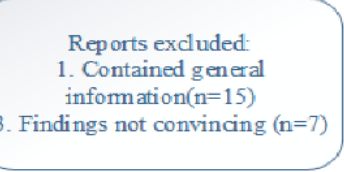
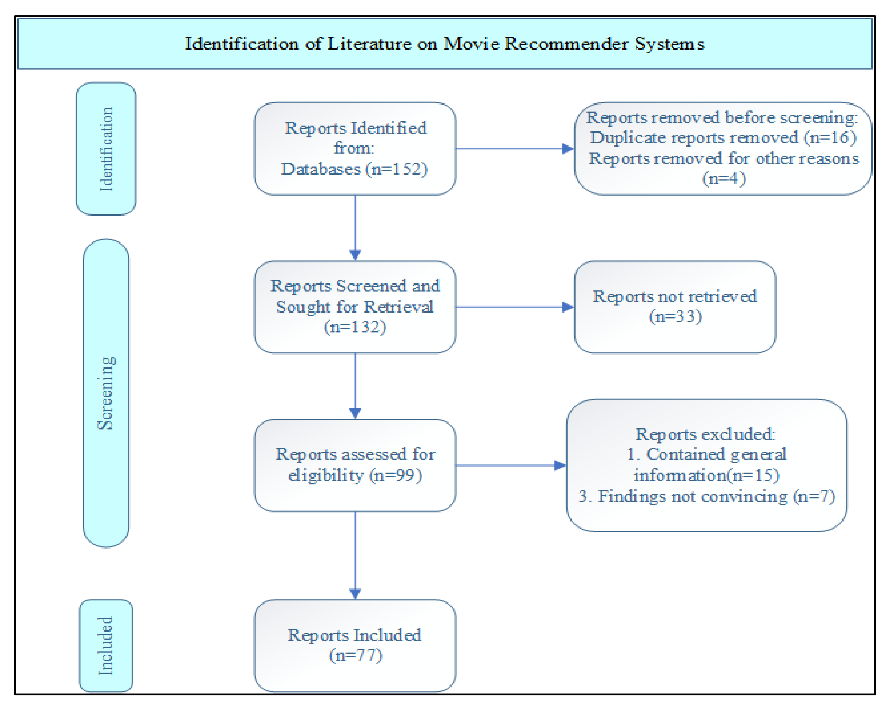
Figure 4.2.1 illustrates the feature extraction process, highlighting the selected features and their role in representing each song.

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. | |
| o | 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. | |
| o | Example: If a user has watched "The Dark Knight" and "Interstellar," they may |
| 10 |  |



|  |  |  |
| --- | --- | --- |
|      | 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. | |
| o | 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. | |
| o | 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.

|  |  |
| --- | --- |
| 11 |  |

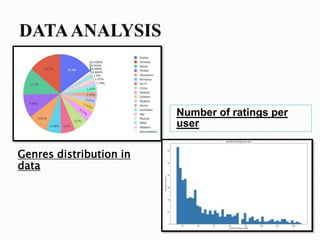
Fig 4.2.3: 2D plot of all normalized features

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.

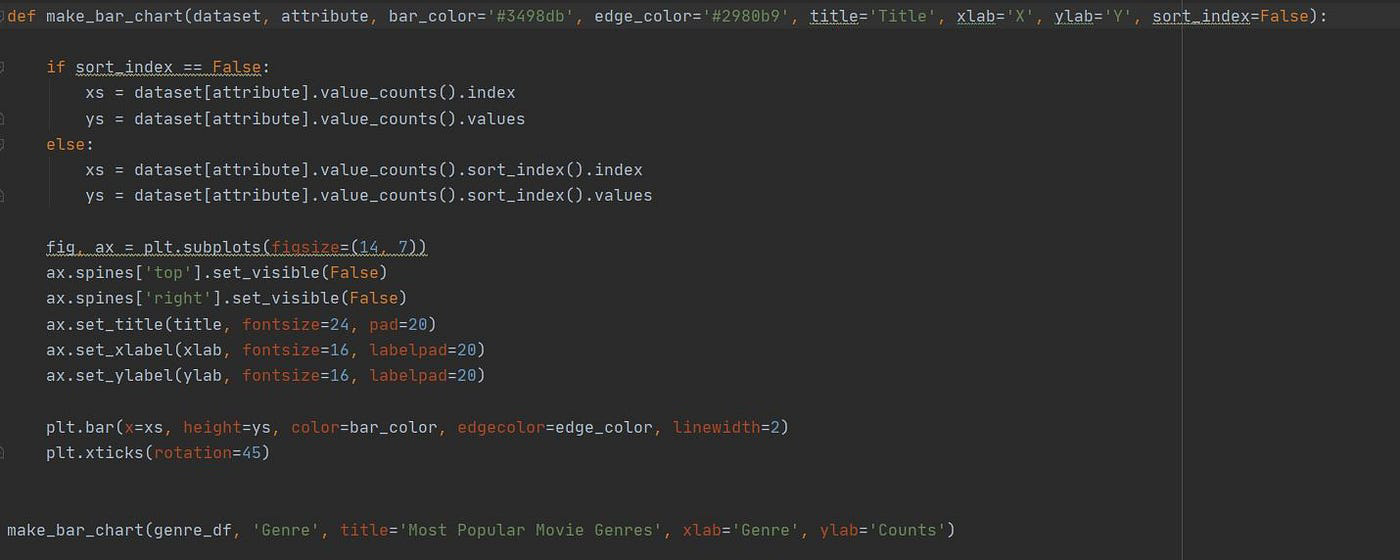


Fig4.3.1: Model Training

4.4 Web Interface Development

|  |  |
| --- | --- |
| 12 |  |



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

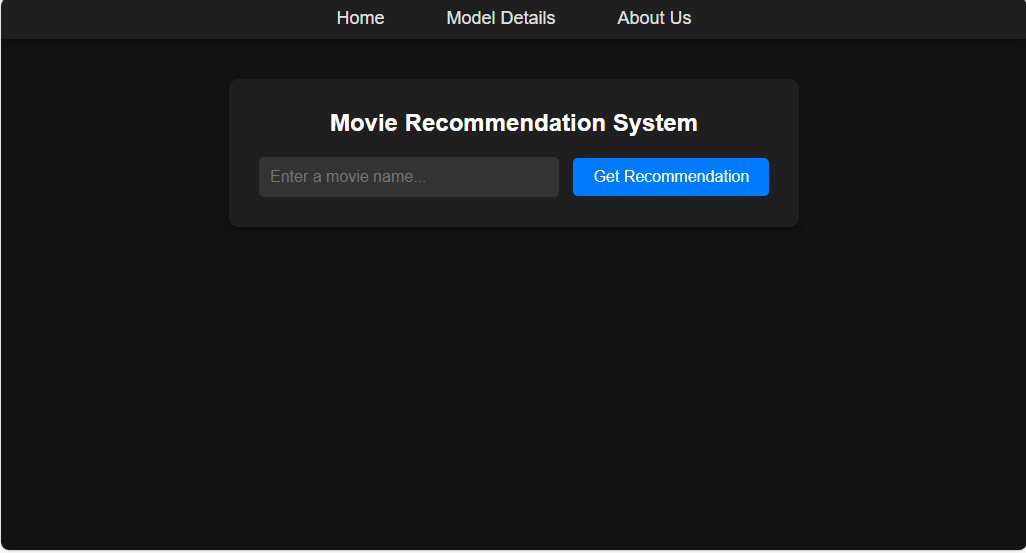


Fig 4.4.1 Recommendation movies in web interface

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.

|  |  |
| --- | --- |
| 13 |  |

|  |  |  |
| --- | --- | --- |
| |  | | --- | |  | | |
| @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\_\_': |

|  |  |
| --- | --- |
| 14 | 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:



15



|  |  |
| --- | --- |
| • • | 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

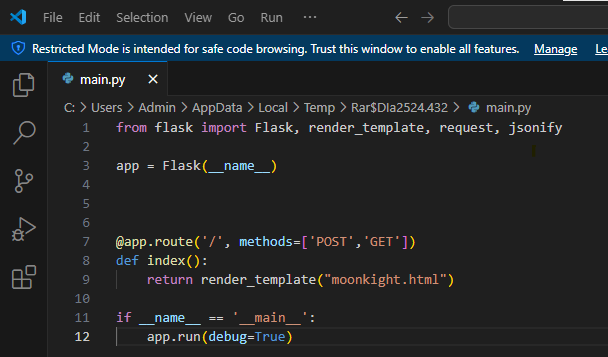
To improve the system, the following enhancements could be explored:

• 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:

|  |  |
| --- | --- |
| 16 |  |



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



17



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 :

Here are some references for building and understanding movie recommendation systems:

1.Matrix Factorisation for recommended System :

A foundational paper by Yehuda Koren, which introduces matrix factorization methods such as Singular Value Decomposition (SVD) and explains their application in recommendation systems.

Link: Matrix Factorization Techniques

2.Deep Rec : An open source toolkit for movie recommendations system :

A toolkit supporting various recommendation models, including matrix factorization and deep learning methods, useful for both research and practice.

Link: DeepRec Toolkit

3.Neural collaborative filtering :

Explains advanced neural network-based recommendation techniques like Generalized Matrix Factorization (GMF) and Neural Matrix Factorization (NeuMF), improving prediction accuracy.

Link: Neural Collaborative Filtering

|  |  |
| --- | --- |
| 18 |  |



4. Optimization techniques :

Discusses optimization techniques for managing large datasets in recommendation systems, such as embedding sharing and cross-tower information exchange.

Link: Large-scale Cardinality

5.Recommendation Management system for Evaluation :

Explores structural complexity and metrics like RMSE for evaluating recommendation systems.

Link: Structural Complexity in Recommenders

These resources provide a mix of theoretical and practical insights into creating robust recommendation systems. Let me know if you'd like detailed explanations about any specific topic.

|  |  |
| --- | --- |
| 19 |  |