**Music Recommender System**

**Submitted for**

**Statistical Machine Learning CSET211**

Submitted by:

**(E23CSEU1896) Shardul Dhekane**

**(E23CSEU1918) Shreya Tripathi**

Submitted to

**Dr. Shakshi Sharma**

**July-Dec 2024**

**SCHOOL OF COMPUTER SCIENCE AND ENGINEERING**

A close-up of a logo

Description automatically generated

**INDEX**

|  |  |  |
| --- | --- | --- |
| Sr.No | Content | Page No |
| 1 | Introduction | 3 |
| 2 | Methodology of the Project | 3 |
| 3 | Minimum Requirements | 4 |
| 4 | Experimental Results and Outcomes | 5 |
| 5 | Challenges Faced | 5 |
| 6 | Conclusion | 6 |
| 7 | Future Scope and Opportunities | 6 |
| 8 | Hyperlinks and References | 7 |

**Introduction**

The Music Recommender System project is designed to enhance user experience by providing personalized music recommendations. In today's digital age, users are inundated with vast amounts of music, making it challenging to discover new and relevant songs. This project addresses this challenge by leveraging hybrid filtering techniques, which combine the strengths of both content-based and collaborative filtering methods. By integrating these approaches, the system aims to deliver more accurate and diverse music suggestions, catering to individual user preferences and enhancing overall satisfaction.

**Methodology of the Project**

The Music Recommender System employs a hybrid filtering approach, integrating both content-based and collaborative filtering methods.

* Content-Based Filtering: Utilizes K-Nearest Neighbors (KNN) with a Cosine Similarity metric to recommend songs like those the user has previously enjoyed. This method analyzes song attributes such as genre, artist, and other metadata to find similarities.
* Collaborative Filtering: Also uses KNN to analyze user behavior and preferences. This method predicts a user's interest in a song based on the preferences of similar users.
* The system is developed using Flask for the web application framework and SQL for database management, ensuring efficient data handling and user interaction.

**Minimum Requirements**

Memory: 4 Gb

Storage: 250 Mb

Compute Power: 0.5 Ghz

Graphical Memory: N/a

**Pre-Requirements**

Software: The system is built using Python, a versatile programming language known for its extensive libraries and frameworks.

Flask is used for web development, while Scikit-learn provides the necessary machine learning algorithms.

An SQL database is employed for efficient data management and retrieval.

Dataset: [Spotify-Dataset with 600K songs](https://www.kaggle.com/datasets/yamaerenay/spotify-dataset-19212020-600k-tracks).

Python Libraries: [Requirement.txt](https://github.com/ProbablyItsSpirit/Music_Recommender_System/blob/main/requirements.txt)

**Experimental Results and Outcomes**

The Music Recommender System was rigorously tested using a dataset of songs and user interactions. The hybrid filtering approach demonstrated significant improvements in recommendation accuracy compared to using either content-based or collaborative filtering alone. Users received more relevant song suggestions, which enhanced their overall experience and satisfaction with the system. The results underscore the effectiveness of combining multiple filtering techniques to address the limitations of individual methods and deliver a more robust recommendation solution.

**Challenges Faced**

Throughout the development and implementation of the Music Recommender System, several challenges were encountered:

* Data Sparsity: The effectiveness of collaborative filtering is often hindered by limited user interaction data. Sparse data can lead to inaccurate recommendations, as the system lacks sufficient information to identify meaningful patterns and relationships.
* Cold Start Problem: New users or songs with no prior data pose significant challenges for generating recommendations. Without historical data, the system struggles to make accurate predictions, necessitating alternative strategies to address this issue.
* Scalability: As the dataset and user base grow, ensuring the system performs efficiently becomes increasingly important. Scalability challenges must be addressed to maintain fast and accurate recommendations, even as the volume of data increases.
* Computation and Performance Issues: Due to many combinations of Matrices and Similarities a lot of times the system lacked compute power and many times the system went out of compute or out of memory.

**Conclusion**

The Music Recommender System successfully integrates content-based and collaborative filtering to provide personalized music recommendations. By leveraging the strengths of both approaches, the system addresses some of the limitations inherent in individual methods, offering a more comprehensive and effective solution for music recommendation. The hybrid approach enhances user satisfaction by delivering more accurate and diverse song suggestions, ultimately improving the overall listening experience.

**Future Scope and Opportunities**

Looking ahead, there are several opportunities to further enhance the Music Recommender System:

* Enhanced Algorithms: Exploring advanced machine learning algorithms, such as deep learning, could improve recommendation accuracy and uncover more complex patterns within the data.
* User Feedback Integration: Incorporating user feedback into the recommendation process can help continuously refine and improve the quality of suggestions, ensuring they remain relevant and personalized.
* Cross-Platform Integration: Expanding the system to support various platforms and devices would provide a seamless user experience, allowing users to access personalized recommendations across different environments.

**Link to my GitHub**

[ProbablyItsSpirit](https://github.com/ProbablyItsSpirit/)

**Dataset**

[Spotify-Dataset with 600K songs](https://www.kaggle.com/datasets/yamaerenay/spotify-dataset-19212020-600k-tracks).

**References and Documentations**

[Flask Documentation](https://flask.palletsprojects.com/en/stable/)

[Scikit-learn Documentation](https://scikit-learn.org/stable/)

[Pairwise Cosine Similarity](https://scikit-learn.org/stable/modules/generated/sklearn.metrics.pairwise.cosine_similarity.html#sklearn.metrics.pairwise.cosine_similarity)

[Kaggle for References](http://www.kaggle.com)

[SQL Documentation](https://dev.mysql.com/doc/)

[KNN Algorithm Overview](https://en.wikipedia.org/wiki/K-nearest_neighbors_algorithm)

[Referenced Project](https://github.com/me-tusharchandra/music-recommendation-system-using-ml)