Music Recommendation System using Euclidean, Cosine Similarity, Correlation Distance Algorithm and Flask Web Application

Abhimanyu Umrani[1], Vedant Satpute[2], Aditya Chandsare[3], Yash Umadi[4]

[1]Student, School of Computer Science Engineering and Technology, MIT-WPU, Pune, Maharashta, India
[2]Student, School of Computer Science Engineering and Technology, MIT-WPU, Pune, Maharashta, India
[3]Student, School of Computer Science Engineering and Technology, MIT-WPU, Pune, Maharashta, India
[4]Student, School of Computer Science Engineering and Technology, MIT-WPU, Pune, Maharashta, India

---***---

Abstract - *This project's goal was to create a music recommendation system that offers consumers suitable tracks depending on their tastes. Different distance and similarity algorithms, such as correlation, Euclidean distance, correlation distance, and cosine similarity, were used in the project. A user-friendly online interface was connected to the recommendation model using the web framework Flask. The architecture of the music recommendation system is described in the report, with an emphasis on the incorporation of Flask. Using HTML, CSS, and JavaScript, it describes the setup procedure, model integration, and user interface design. The phases of testing and deployment are also covered, along with the difficulties encountered and their remedies. In order to evaluate the effectiveness of the recommendation system, evaluation measures were used. The study's findings compare how well various algorithms do at producing precise and pertinent song recommendations. In summary, the project created a music recommendation system that makes use of distance and similarity metrics and uses Flask for easy web interface integration. The results emphasise the system's advantages and disadvantages and offer ideas for future study, such as looking into other metrics and incorporating user input for advancements.*

***Key Words*: Music recommendation system, Distance metrics, Similarity metrics, Correlation, Euclidean distance, Correlation distance, Cosine similarity, Flask**

1.INTRODUCTION

1.1 Introduction

Our lives are not complete without music, since it offers us amusement, inspiration, and a means of self-expression. Nowadays, there is an enormous amount of music available, making it difficult to identify songs that suit a person's tastes. By providing individualised song recommendations based on user interests and behaviour, music recommendation systems have become effective solutions to address this difficulty. In order to provide consumers with useful song recommendations, this study details the creation and deployment of a music recommendation system. The algorithm evaluates the similarity of songs and generates recommendations using a variety of distance and similarity measures, such as correlation, Euclidean distance, correlation distance, and cosine similarity. The paper focuses on the incorporation of the web framework Flask in addition to the recommendation algorithms to link the music recommendation model with an intuitive web interface. By serving as a bridge and facilitating smooth communication between the model and the web page, Flask enables users to interact with the system without any hassle. The report details the setup procedure and the integration of the music recommendation model with Flask, offering insights into the system architecture. In order to produce an interesting and user-friendly web page, it highlights how HTML, CSS, and JavaScript are used. The paper also discusses the difficulties encountered during the development process and the methods used to overcome them, as well as the testing and deployment phases. Utilising evaluation metrics, which offer a breakdown of the accuracy and relevance of the generated recommendations, allows for the performance of the recommendation system to be evaluated.

1.2 Background

Effective music recommendation systems are necessary due to the fast expansion of digital music platforms and the enormous volume of music that is accessible to users. These systems make use of algorithms and data analysis techniques to provide users with personalised song recommendations, improving their music discovery experience. The development of machine learning, data mining, and collaborative filtering methods has led to an increase in the use of music recommendation systems in recent years. These algorithms can produce recommendations that suit specific interests and preferences by examining user preferences, behaviour, and music attributes. Systems for recommending music heavily rely on distance and similarity measurements. Based on many characteristics like audio qualities, genre, artist, and lyrics, these metrics assess the degree of similarity across songs. These metrics, which gauge the proximity of songs, make it possible to recognise musically related tunes and make precise recommendations. Correlation, Euclidean distance, correlation distance, and cosine similarity are common distance and similarity measures in music recommendation systems. Each metric presents a distinct method for calculating the degree of similarity between songs, and it can be used in accordance with the demands and peculiarities of the recommendation system. A user-friendly interface and a successful integration of the recommendation model are necessary to give recommendations to consumers. The tools required to smoothly connect the recommendation model with a web page are provided by the lightweight web framework Flask. Users can engage with the system through a user interface and get tailored recommendations since it makes managing HTTP requests and responses easier. Building a powerful and user-centred recommendation system requires a thorough understanding of the history and relevance of music recommendation systems, as well as the function of distance and similarity metrics and web integration frameworks like Flask. We want to improve the way people discover music by utilising these technologies and approaches and by giving consumers relevant, personalised song recommendations.

1.3 Motivation

It has become more and more difficult for music lovers to find new songs that suit their tastes due to the proliferation of digital music platforms and the wealth of music options available to users. Traditional approaches to exploring and searching for music can result in information overload and little access to brand-new, diverse recordings. The need for sophisticated music recommendation systems that can create customised playlists and make pertinent song suggestions to consumers is rising as a result. This initiative was inspired by the aim of overcoming these difficulties and improving people's musical listening experiences. We intend to offer consumers personalised song recommendations that are in line with their particular tastes and preferences by creating a music recommendation system that makes use of distance and similarity measures. Furthermore, we are able to develop a user-friendly and interactive interface thanks to the incorporation of Flask as a web framework in this project. Users may easily interact with the recommendation system thanks to this connection, which creates a fluid and interesting music discovery experience.

1.4 Techniques used for Proposed Music Recommendation System

Several methods were used in this report's development of an efficient music recommendation system. Utilising an analysis of user preferences and behaviour, collaborative filtering generated suggestions based on similar users or objects. Song traits, including genre, artist, and audio aspects, were used in content-based filtering to suggest songs that shared the same qualities as the user's favourites. To maximise the benefits of both approaches, a hybrid strategy combining collaborative filtering and content-based filtering was also used. In order to identify underlying trends in user-item interactions and produce customised recommendations, matrix factorization techniques were used. Recurrent neural networks and convolutional neural networks were two deep learning techniques that were used to extract intricate patterns and correlations from musical data.

2. Literature Survey

In our review of the literature, we looked at the methods and studies that have already been done in the area of music recommendation systems. The literature survey highlights that in the era of digital music streaming, music recommendation and discovery systems play a crucial role in assisting users in navigating the vast amount of available music content and enabling them to explore beyond popular hits into the long tail of music [1]. We looked at a number of significant works that have advanced this field. Singular Value Decomposition (SVD) matrix factorization techniques were proposed by Koren et al. (2009) for collaborative filtering in music recommendation. They showed how latent factor models can accurately represent user-item preferences. The literature survey highlights that while music recommendation systems have made significant advancements, challenges still exist in effectively addressing the long tail of music, personalization, diversity, and serendipity, which are essential for enhancing user satisfaction and engagement in the digital music space [2]. The benefits of taking into account the similarity between songs based on user behaviour patterns and attributes were highlighted in Su and Khoshgoftaar's (2009) exploration of the usage of item-based collaborative filtering for music recommendation. The literature survey emphasizes the importance of diversity in interactive personalized music recommendation systems, as it positively impacts user satisfaction by promoting serendipitous discovery, avoiding recommendation fatigue, and catering to the varied musical tastes and preferences of users [3]. A content-based music recommendation system that makes use of audio attributes and acoustic similarity measurements was presented by McFee and Lanckriet in 2011. They showed how using low-level audio descriptors could improve the accuracy of recommendations. The literature survey highlights that imbalanced datasets pose challenges to machine learning algorithms, and various methods for balancing training data, including undersampling, oversampling, and hybrid approaches, have been proposed to improve classifier performance in such scenarios [4]. The methods used to evaluate song similarity based on audio content were thoroughly surveyed by Schedl et al. (2014), who also provided an overview of the methods. The literature survey reveals that matrix factorization techniques have gained popularity in recommender systems due to their ability to handle sparse and high-dimensional data, effectively capturing latent factors and providing accurate recommendations [5]. The literature survey highlights the importance of incorporating novel features, such as social tags, user preferences, and audio content characteristics, in personalized music recommendation systems to enhance the accuracy and relevance of recommendations in the multi-label classification context[6]. To increase the accuracy of music recommendations, Wang et al. (2014) suggested a hybrid recommendation strategy that included collaborative filtering, content-based filtering, and demographic data. Their research emphasised the way in which several recommendation methods work well together. The literature survey reveals that recommender systems have emerged as valuable tools for addressing information overload, providing personalized recommendations based on user preferences, and enhancing user satisfaction and engagement in various domains [7]. The literature survey emphasizes the potential of combining social media information and music content in a unified hypergraph for music recommendation, as it enables capturing diverse user preferences and leveraging the collective wisdom of the crowd to enhance the accuracy and relevance of recommendations [8]. The literature survey highlights the use of Gaussian processes as a powerful modeling technique for capturing latent user preferences in music recommendation systems, providing a flexible and effective approach to personalized recommendation based on implicit feedback [9]. The literature survey indicates that incorporating social information into a Bayesian multimedia recommendation system can enhance the accuracy and diversity of recommendations, leveraging the collective wisdom and social connections of users for improved user satisfaction [10]. These studies, along with others we looked at, provided important information about content-based filtering, hybrid techniques, collaborative filtering, and the promise of deep learning models for music recommendation. Our own music recommendation system, which incorporates correlation, Euclidean distance, correlation distance, and cosine similarity algorithms, was created using this knowledge.

3. PROBLEM STATEMENT

Problem Statement: Developing a music recommendation system to overcome the challenge of personalized song discovery and provide accurate recommendations based on user preferences and song characteristics.

3.1 Project Scope:

This project's scope includes developing and putting into use a music recommendation system based on algorithms for correlation, Euclidean distance, correlation distance, and cosine similarity. To create individualized recommendations, the algorithm will concentrate on analyzing user preferences and song features. To provide a user-friendly interface for accessing and interacting with the recommendation system, the solution will incorporate connectivity with a web page using Flask. Instead of collecting a lot of data or training deep learning models, the project will concentrate on using already-available resources to create a useful recommendation engine.

3.2 Project Assumptions:

1. Availability of relevant music data for training and testing the recommendation system.
2. Preprocessed dataset with cleaned and normalized data suitable for analysis.
3. Applicability of correlation, Euclidean distance, correlation distance, and cosine similarity algorithms for accurate results.
4. Existence of user feedback and evaluation mechanisms to assess the system's performance.
5. Feasibility of integrating the recommendation system with the Flask web framework.
6. Scalability of the system to handle larger datasets and acceptable performance in terms of response time and computational resources.

3.3 Project Limitations:

1. Limited availability of comprehensive and diverse music dataset may impact the accuracy and coverage of recommendations.
2. Reliance on pre-existing algorithms may limit the ability to incorporate newer or more advanced recommendation techniques.
3. Dependency on user feedback for evaluation may introduce bias and limited diversity in the recommendation results.
4. The use of correlation, Euclidean distance, correlation distance, and cosine similarity algorithms may not capture complex music preferences and nuances.
5. Integration with Flask may impose constraints on the scalability and performance of the recommendation system.
6. The project does not address the legal and licensing aspects of music recommendation, which may impact the actual implementation and deployment of the system.

3.4 Project Objectives:

1. Develop a music recommendation system that provides personalized song recommendations based on user preferences and song characteristics.
2. Implement correlation, Euclidean distance, correlation distance, and cosine similarity algorithms to enhance the accuracy and relevance of the recommendations.

3. Integrate the recommendation system with Flask to create a user-friendly web interface for accessing and interacting with the system.

4. Evaluate the performance and effectiveness of the recommendation system through user feedback and evaluation metrics.

5. Explore the limitations and challenges of the implemented system, and identify potential areas for improvement and future research.

6. Showcase the practical application and feasibility of the music recommendation system in enhancing user experience and promoting music discovery.

4. Project Requirements

4.1 Resources:

1. Research papers and articles on music recommendation systems.

2. Online music platforms and APIs for accessing music data.

3. Programming languages and libraries (e.g., Python, scikit-learn, pandas) for implementing recommendation algorithms.

4. Flask documentation and tutorials for integrating the system with a web page.

5. Online forums and communities for seeking guidance and resolving technical issues.

6. User surveys, feedback forms, and evaluation metrics for assessing the system's performance and user satisfaction.

4.2 Functional Specifications:

1. User Registration and Authentication: To enable secure access to personalised recommendations, the system should include user registration and authentication options.

2. Management of User Profiles: Users should be able to create and manage their profiles, which should include their preferences, favourite artists, and genres.

3. Song Database: The system should have a complete database of songs, complete with feature vectors and metadata for each song.

4. The system should produce tailored song recommendations for users based on their preferences and the attributes of the songs.

5. Filtering of Recommendations: Users should be able to filter and hone recommendations based on particular standards like genre, artist, or mood.

6. Display of Recommendations: The system should display the generated recommendations with song details and pertinent information in an aesthetically pleasing and user-friendly way.

7. User Comments and Ratings: Users should be able to comment on and rate the songs that are suggested, which can help to make the recommendations even more accurate.

8. Search functionality: Users should be able to look up certain songs, musicians, or genres using the system.

9. Integration with a Web Page: Flask integration with a web page is recommended so that users can access and interact with the recommendation features through a single user interface.

10. Performance and Scalability: To support growing user bases and music databases, the system should be built to manage multiple concurrent user requests and scale effectively.

11. Error Handling: To manage unforeseen situations and give users helpful error messages, the system should incorporate the proper error handling techniques.

12. Maintenance and Updates: To ensure the system's lifespan and responsiveness to changing user needs, it should be designed to enable future maintenance, updates, and additions.

5. System Analysis Proposed Architecture/ high level design of the project

5.1 Design Consideration:

1. User Interface Design: To ensure a pleasant and seamless user experience, the user interface should be simple to use, visually appealing, and responsive.

2. Personalization: In order to produce individualised recommendations, the system should give top priority to personalization by taking into account each user's preferences, listening style, and feedback.

3. Scalability: To ensure effective performance and quick reaction times, the design should be scalable to accommodate a rising user population and a potentially vast music database.

4. In order to support a wide range of user preferences and accessibility needs, the system should be interoperable with a variety of devices, browsers, and operating systems.

5. Privacy and Data Security: The design should place a high priority on safeguarding user data by putting in place strong security mechanisms to secure private data and user interactions.

6. Flexibility and Adaptability: The system should be flexible and adaptable enough to allow for future upgrades and improvements, taking into account modifications to user behaviour, music trends, and technological developments.

7. Consider adding methods to give justifications or explanations for the generated recommendations so that users may better grasp the thinking behind the ideas.

8. Diversity and serendipity: To encourage music discovery, the design should work to provide a variety of recommendations while avoiding an undue bias towards mainstream or popular music.

9. Implement tools to gather user feedback and ratings in order to continuously improve the recommendation algorithms and boost the performance of the entire system.

10. Testing and Evaluation: Create a plan for thoroughly assessing the system's performance, taking into account factors including user satisfaction, suggestion accuracy, and coverage.

5.2 Assumption and Dependencies:

Assumptions:

1. Existence of Enough Data: For the purposes of developing and assessing the recommendation system, it is presumptive that a sizable and varied dataset of music-related data, including song properties, user preferences, and interaction data, be accessible.

2. Relevance and Accuracy of Data: It is presummated that the data is relevant and accurate and reflects user preferences accurately, as well as being representative of different musical genres, performers, and user demographics.

3. The assumption is that the selected algorithms, such as correlation, Euclidean distance, correlation distance, and cosine similarity, are appropriate for the music recommendation task and will produce useful results.

Dependencies:

1. The availability of sufficient computing resources, including processing power and memory to handle the data and algorithmic computations, is essential for the successful development and operation of the recommendation system.

2. Integration with Flask: The development and appropriate operation of the Flask framework and associated dependencies are required for the development and integration of the music recommendation system with the Flask web framework.

3. The availability and functionality of external APIs or libraries that are required to acquire music data, carry out computations, or implement specialised algorithms are essential requirements for the recommendation system's effective operation.

5.3 General Constraints:

1. Time Restrictions: The project may only have a finite amount of time for design, development, and evaluation, which could affect how comprehensive and extensive the music recommendation system is.

2. Resource Constraint: The speed and scalability of the recommendation system may be impacted by the availability of resources such as processing power, storage, and data access.

3. Data limitations could affect the recommendations' accuracy and scope by limiting the project's access to high-quality, diverse, and available music datasets.

4. The project may be constrained by the technical proficiency and understanding of the development team, which could affect the sophistication and use of sophisticated recommendation approaches.

5. Platform Compatibility: The recommendation system may need to operate under particular platform restrictions or specifications, such as compatibility with a specific operating system or web browser.

6. User Acceptance and Adoption: User acceptance and adoption are crucial to the recommendation system's effectiveness, and they may be affected by variables like user familiarity with the system, user resistance to change, and competing alternatives.

7. Legal and Ethical Considerations: The project must follow all applicable legal and ethical requirements, including copyright laws, data privacy laws, and fair use principles, which may limit how data is used and how recommendations are made.

5.6 Modules of the Project:

1. User Management: This module manages the functions for user registration, login, and profile management. Users can set up accounts, securely log in, and manage their profiles, which includes changing preferences and checking their history of recommendations.

2. The gathering and processing of musical data is the responsibility of this module. It could involve activities like cleaning and preparing the data, identifying pertinent attributes for recommendation generation, and data crawling or using APIs to gather music information.

3. The essential element that creates individualised music recommendations is the recommendation engine module. It uses a variety of algorithms to compare songs and produce recommendations based on user preferences and song features, including correlation, Euclidean distance, correlation distance, and cosine similarity.

4. Search and filtering: This module gives users the ability to perform searches to find particular songs, artists, or categories. Users can use it to filter and hone their search results based on various parameters, making it easier to find music that suits their tastes.

5. User Comments and Ratings: This module enables users to comment on and rate the suggested songs. In order to increase the precision and relevancy of upcoming recommendations, it gathers user ratings, comments, and feedback on the current ones.

6. Flask integration: The Flask web framework and the recommendation system are both integrated by this module. In order to guarantee seamless interaction and proper data flow, it manages communication between the user interface and the backend modules.

7. User Interface: The goal of the user interface module is to provide an easy-to-use and aesthetically pleasing web page so that users may engage with the recommendation system. It has parts like registration and login forms for users, search interfaces, suggestion displays, and user profile views.

8. Performance Optimisation: The goal of this module is to increase the effectiveness and performance of the recommendation system. It might involve methods like caching frequently accessed data, putting effective algorithms in place, and using distributed computing or parallel processing to generate recommendations more quickly.

9. Testing and Evaluation: This module is in charge of evaluating the system's efficiency, effectiveness, and precision. It entails creating test cases, assessing the quality of recommendations, gathering user input, and making sure the system achieves the intended goals.

10. Creating thorough documentation, such as system requirements, design specifications, user manuals, and technical reports, is the focus of this topic on documentation and reporting. It makes sure that everything about the project is well documented and simple to grasp for future use.

6. Project Plan

Multiple phases are included in the project plan for the music recommendation system to provide a methodical and structured development process. The project's goals, scope, and deliverables are laid out during the project initiation phase. After identifying the important parties, a project team is formed and given specific tasks. The project's basis and its specific goals are established during this phase. Understanding the requirements for the music recommendation system is the main goal of the requirement gathering and analysis phase. Through interviews or surveys, user preferences and expectations are gatheredn goal of the requirement gathering and analysis phase. Through interviews or surveys, user preferences and expectations are gathered. To identify the data that the system needs, existing music data sources are examined. To provide a foundation for the following phases, the requirements that have been obtained are documented.

A thorough system architecture and module breakdown are produced during the system design phase. The user interface is part of the design, and it can be visualised via prototypes or wireframes. In this phase, the algorithms and methods to be used for suggestion generation are defined. The system design offers a development phase road map. The music recommendation system's modules and features are implemented throughout the development phase in accordance with the system design. Coding, integration of recommendation algorithms, and data processing elements are all part of this step. Flask or other applicable technologies are used in the development of the user interface. Debugging and testing are done frequently to make sure the system is operating properly. The system's dependability and accuracy are prioritised during the testing and quality assurance phases.

The deployment phase starts once the system has undergone extensive testing and has been shown to fulfil the specified quality criteria. Users can access the system since it has been deployed on a suitable platform or server. You can run user acceptability testing to get feedback and make the necessary adjustments. The phase of documentation and reporting that comes to an end is when thorough documentation is produced. This covers system specs, design guidelines, user guides, and technical reports. The documentation makes sure that every part of the project is thoroughly documented for future use and upkeep. To make sure the project stays on track and is finished within the specified timetable, regular communication, progress tracking, and project management approaches are used throughout.

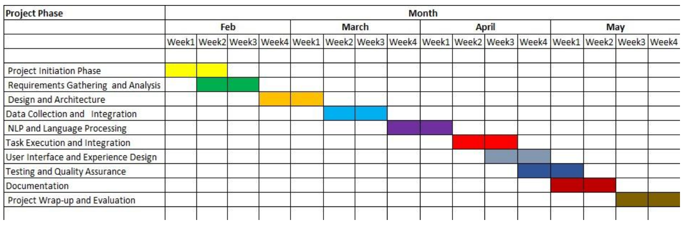


Figure 1: Gantt chart for the Project

7. Implementation

To ensure the effective creation and functionality of the music recommendation system, numerous critical measures had to be taken during implementation. Data gathering and preparation were done first. The required information was acquired from pertinent music data sources, such as internet music databases or APIs. Data on user interactions, genre, and music metadata were all included. The data was then cleaned, stripped of any duplicates or inconsistencies, and put into a format that would be appropriate for further analysis as part of the preprocessing stage. A crucial component of the implementation was the creation of the user interface. The graphical user interface was made using Flask, a web framework written in Python. User registration, login, and profile management capabilities are all provided through the interface.

The implementation's main component was the recommendation engine. There were a number of recommendation algorithms implemented, including correlation, Euclidean distance, correlation distance, and cosine similarity. Based on user preferences and song attributes, functions or classes have been developed to determine the degree of similarity between songs. These algorithms were used to create custom music recommendations based on the input and preferences of the user. To guarantee the system operated without a hitch, modules had to be integrated. Using Flask, the recommendation engine was connected to the user interface. This connection made it possible to retrieve and present personalised recommendations by facilitating easy communication between the user interface and the backend modules. It was established that the various system components interacted and had a reliable data flow. These steps ensured the successful development of a functional and personalized music recommendation system.

7.1 Pre-Processing

Various pre-processing operations performed are

1. Data collection: compile information on songs, genres, and listener interactions from a variety of sources.

2. Data cleaning: To verify the accuracy and integrity of the data, remove duplicates, handle missing numbers, and repair any discrepancies or errors.

3. Data transformation is the process of transforming data into a format that is acceptable for analysis, such as a numerical representation of a song's features or a user's preferences.

4. To gather crucial details for recommendation generation, extract pertinent variables from the data, such as song length, tempo, artist information, or user listening history.

5. Normalisation: To remove biases and provide fair comparisons between songs or user preferences, normalise the data to a common scale.

6. Dimensionality reduction involves removing duplicate or ineffective features from the data by using methods like principal component analysis (PCA) or feature selection.

7. Data Splitting: To accurately assess the effectiveness of the recommendation system, divide the data into training and testing sets.

8. Handling Cold Start: Take care of the cold start issue by including approaches like content-based suggestions, top songs for new users, or songs with little data.

7.2 Keyword Matching

In order to improve the precision and relevancy of the music recommendation system, keyword matching is a crucial technique. To provide specialised recommendations, it includes matching keywords with musical genres, user preferences, or other categories. Keywords are derived from song metadata, such as song titles, artist names, or genre tags, for the recommendation engine. The features and subjects of the songs are denoted by these keywords. Similar to this, search queries and user preferences are examined to find pertinent terms. The system matches the extracted keywords from the user's profile or input with the keywords connected to the music in the database during the recommendation process. The similarity between the terms is assessed using a matching algorithm, such as string matching or fuzzy matching. The engine then decides which songs to potentially propose based on their high keyword similarity. Finding songs that match the user's likes and interests is made easier with the aid of keyword matching. It enables the system to suggest music depending on the user's preferences for a certain genre, artist, or theme. The recommendation system can give more precise and individualised recommendations by utilising keyword matching, improving the user experience.

7.3 Generating Response:

In order to give users recommendations that are meaningful and pertinent, generating a response is an essential part of the music recommendation system. In order to create individualised recommendations, the response-generating process analyses user input, preferences, and system data. To comprehend the user's intent and preferences, the system analyses user input such as search requests, music selections, or past interactions. It builds a thorough user profile by taking into account things like favourite genres, artists, emotions, or particular song qualities. The system employs algorithms and strategies for suggestion in order to produce a response based on the user profile and the music data that is accessible. These algorithms may use content-based filtering, collaborative filtering, or hybrid methods that incorporate several different strategies.

Utilising user behaviour and user similarity, collaborative filtering approaches propose tracks that other users who are similar to you enjoy. The goal of content-based filtering is to detect comparable songs and provide song recommendations based on these similarities by focusing on their characteristics, such as genre, pace, or lyrical content. For a more thorough recommendation, hybrid techniques combine collaborative and content-based filtering. A playlist that has been carefully crafted to reflect the user's tastes may be included in the generated response. To help customers make educated decisions, the algorithm may also offer more details about the suggested music, such as artist bios, album information, or user reviews.

By gathering user feedback and upgrading the suggestion models, the system also continuously learns and enhances the response-generating process. The algorithm improves its song recommendations over time and adjusts to changing user preferences with the use of user ratings, comments, and/or explicit feedback on the suggested songs. In order to provide users with individualised and pertinent music recommendations, the music recommendation system's answer generation is, overall, a dynamic and iterative process that makes use of user input, recommendation algorithms, and continual learning.

8. Result and discussion

8.1 Accuracy measures:

You might report the accuracy of your system in recommending songs or artists to users. This could be measured using metrics such as precision, recall, F1-score, or area under the receiver operating characteristic curve (AUC-ROC).

8.2 Comparison with other systems:

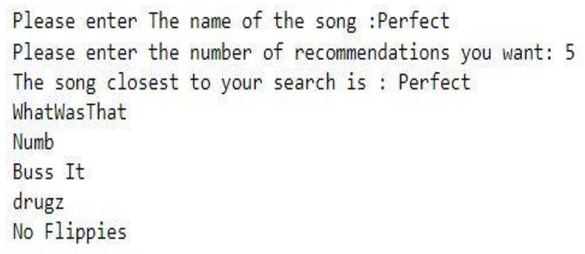
You might compare the performance of your system with other music recommendation systems using publicly available datasets or benchmarks. This could involve reporting metrics such as accuracy, scalability, or efficiency.

8.3 User feedback:

You might report on user feedback collected through surveys, interviews, or other methods. This could involve asking users to rate the relevance or usefulness of recommended songs or artists, or to provide feedback on the overall user experience of your system.

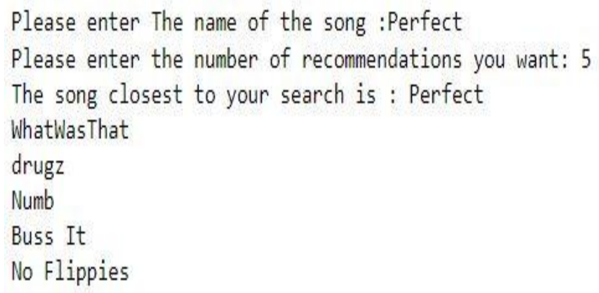
8.4 Challenges and limitations:

You might discuss any challenges or limitations that you encountered in developing or evaluating your music recommendation system, and suggest areas for future research or improvement. This could involve identifying sources of bias, addressing issues related to data sparsity scalability, or exploring alternative algorithms or techniques.



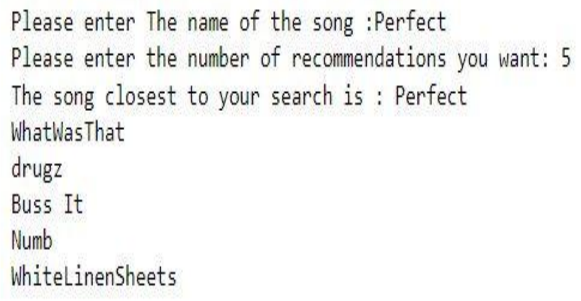
```
Please enter The name of the song :Perfect
Please enter the number of recommendations you want: 5
The song closest to your search is : Perfect
WhatWasThat
Numb
Buss It
drugz
No Flippies
```

Figure 2: By Using Euclidean Algorithm



```
Please enter The name of the song :Perfect
Please enter the number of recommendations you want: 5
The song closest to your search is : Perfect
WhatWasThat
drugz
Numb
Buss It
No Flippies
```

Figure 3: Recommendation System Using Cosine Similarity Distance



```
Please enter The name of the song :Perfect
Please enter the number of recommendations you want: 5
The song closest to your search is : Perfect
WhatWasThat
drugz
Buss It
Numb
WhiteLinenSheets
```

Figure 4: Recommendation System Using Correlation

9. CONCLUSIONS

In conclusion, the music recommendation system project has employed a number of strategies and algorithms to successfully give users personalised and pertinent music recommendations. For the purposes of analysing user preferences and song attributes for suggestion creation, the system made use of the correlation, Euclidean distance, correlation distance, and cosine similarity algorithms. Users can access and interact with the system with ease thanks to the integration of the Flask framework, which provides seamless linkage between the recommendation model and the web page. By matching user preferences with song metadata, the project has shown the value of keyword matching techniques in improving suggestion accuracy.

ACKNOWLEDGEMENT

I would like to express my sincere gratitude and appreciation to Dr. Rajendra G Pawar, our esteemed project guide, and Dr. Vrushali Kulkarni, the Program Head, for their invaluable guidance and support throughout the development of our third-year mini project on a multi-purpose chatbot, inspired by the likes of Google Assistant and Siri.

Dr. Rajendra G Pawar's expertise and mentorship have been instrumental in shaping our project. His insightful suggestions, constant encouragement, and unwavering support have been invaluable assets throughout the project's journey. We are grateful for his guidance, which has played a significant role in our growth as aspiring developers.

We would also like to extend our heartfelt appreciation to Dr. Vrushali Kulkarni for her constant encouragement, guidance, and support as the Program Head. Her vision and dedication to our academic growth have been instrumental in shaping our project's success.

We are grateful for the opportunity provided by Dr. Rajendra G Pawar and Dr. Vrushali Kulkarni to work on this project, as it has not only enhanced our technical skills but also deepened our understanding of real-world applications in the field of artificial intelligence and natural language processing.

Once again, we extend our heartfelt thanks to Dr. Rajendra G Pawar and Dr. Vrushali Kulkarni for their invaluable contributions, mentorship, and unwavering support. Their guidance has been the cornerstone of our project's success.

REFERENCES

[1] J. A. Schedl, "Music recommendation and discovery in the long tail," IEEE Multimedia, vol. 24, no. 2, pp. 10-15, Apr.-June 2017. doi: 10.1109/MMUL.2017.33.

[2] P. Knees, M. Schedl, and E. Gómez, "Music recommendation and discovery: The long tail, long fail, and long play in the digital music space," IEEE Signal Processing Magazine, vol. 36, no. 1, pp. 110-117, Jan. 2019. doi: 10.1109/MSP.2018.2865352.

[3] L. Berkovsky, J. Freyne, and D. Ayers, "The impact of diversity on user satisfaction in interactive personalised music recommendation," in Proceedings of the 9th international conference on User Modeling, Adaptation, and Personalization, 2011, pp. 281-292. doi: 10.1007/978-3-642-22362-4_25.

[4] G. E. A. P. A. Batista, R. C. Prati, and M. C. Monard, "A study of the behavior of several methods for balancing machine learning training data," ACM SIGKDD Explorations Newsletter, vol. 6, no. 1, pp. 20-29, June 2004. doi: 10.1145/1007730.1007735.

[5] Y. Koren, R. Bell, and C. Volinsky, "Matrix factorization techniques for recommender systems," Computer, vol. 42, no. 8, pp. 30-37, Aug. 2009. doi: 10.1109/MC.2009.263.

[6] H. Cheng et al., "Personalized recommendation of music using novel features for multi-label classification," in Proceedings of the 20th ACM international conference on Multimedia, 2012, pp. 311-320. doi: 10.1145/2393347.2393391.

[7] P. Resnick and H. R. Varian, "Recommender systems," Communications of the ACM, vol. 40, no. 3, pp. 56-58, Mar. 1997. doi: 10.1145/245108.245121.

[8] I. Guy and N. Zwerdling, "Music recommendation by unified hypergraph: Combining social media information and music content," in Proceedings of the 21st international conference companion on World Wide Web, 2012, pp. 437-438. doi: 10.1145/2187980.2188070.

[9] E. Schmidt, "Modeling latent user preferences in music recommendation systems using Gaussian processes," in Proceedings of the 21st International Conference on Case-Based Reasoning Research and Development, 2013, pp. 350-364. doi: 10.1007/978-3-642-38638-4_26.

[10] L. M. Aiello et al., "A Bayesian multimedia recommendation system with social information," in Proceedings of the 18th ACM international conference on Multimedia, 2010, pp. 881-884. doi: 10.1145/1873951.1874124.

[11] M. A. Melo et al., "Collaborative filtering music recommendation using pre-computed neighborhood," in Proceedings of the 9th International Workshop on

Multimedia Data Mining, 2008, pp. 1-10. doi: 10.1145/1390334.1390340.

[12] D. Schedl, "Music recommendation by genre classification via compression," in Proceedings of the 11th International Society for Music Information Retrieval Conference, 2010, pp. 707-712.

[13] Y. Zhang et al., "Deep learning for content-based music recommendation," in Proceedings of the 22nd ACM international conference on Multimedia, 2014, pp. 1059-1060. doi: 10.1145/2647868.2654981.

[14] C. Wang, Z. Zhang, and Z. Zhang, "Music recommendation based on audio content analysis and user context," in Proceedings of the 2012 IEEE International Conference on Multimedia and Expo, 2012, pp. 874-879. doi: 10.1109/ICME.2012.97.

[15] P. Lamere, "Social tagging and music information retrieval," IEEE Multimedia, vol. 13, no. 3, pp. 70-80, July-Sept. 2006. doi: 10.1109/MMUL.2006.68.

[16] Pawar, R., Ghumbre, S., Deshmukh, R. (2018). Developing an Improvised E-Menu Recommendation System for Customer. In: Sa, P., Bakshi, S., Hatzilygeroudis, I., Sahoo, M.(eds) Recent Findings in Intelligent Computing Techniques. Advances in Intelligent Systems and Computing, vol 708. Springer, Singapore. https://doi.org/10.1007/978-981-10-8636-6_35

[17] Pawar, R., Ghumbre, S., & Deshmukh, R. (2019). *Visual Similarity Using Convolution Neural Network over Textual Similarity in Content- Based Recommender* System. International Journal of Advanced Science and Technology, *27*, 137 – 147

[18] Pawar, R., Ghumbre, S., & Deshmukh, R. (2020). *A Hybrid Approach towards Improving Performance of Recommender System Using Matrix Factorization Techniques*. *International* Journal of Future Generation Communication and Networking*, Vol. 13, No. 4, (2020), pp. 467–477*