**MUSIC RECOMMENDATION SYSTEM**

**A MINI PROJECT REPORT**

**SUBMITTED BY**

**G.ATCHAYA (953622104012)**

**U.JEYANANDHINI(953622104039)**

**P.KIRUTHIGA( 953622104051)**

**BACHELOR OF ENGINEERING**

**IN**

**COMPUTER SCIENCE AND ENGINEERING**



**RAMCO INSTITUTE OF TECHNOLOGY,**

**RAJAPALAYAM**

**JANUARY 2024**

**CS3361 DATA SCIENCE LABORATORY**

**BONAFIDE CERTIFICATE**

Certified that this Miniproject report “**MUSIC RECOMMENDATION SYSTEM**” is the bonafide work of **“G.ATCHAYA (953622104012), U.JEYANANDHINI (953622104039)”,** **“P.KIRUTHIGA**” who carried out the miniproject work under my supervision.

**SIGNATURE**  **SIGNATURE**

S.Vijaya Amala Devi M.E Dr.K.Vijayalakshmi M.E.,Ph.D.

**FACULTY IN-CHARGE** **HEAD OF DEPARTMENT**

Department of Computer Science and Department of Computer Science

Engineering, and Engineering,

Ramco Institute of Technology, Ramco Institute of Technology,

North Venganallur Village, North Venganallur Village,

Rajapalayam-626117 Rajapalayam-626117

**INTERNAL EXAMINER EXTERNAL EXAMINER**

**ABSTRACT**

This report delineates the comprehensive design and execution of a Music Recommendation System, leveraging cutting-edge Data Science techniques. The primary objective is to elevate user satisfaction by furnishing tailored music suggestions based on individual preferences and behavior. The system incorporates both collaborative filtering and content-based filtering algorithms, meticulously analyzing user interactions and music features to generate highly precise recommendations. Python serves as the foundational programming language, harnessing its extensive library ecosystem.

Key libraries such as Pandas, Scikit-learn, and Flask are instrumental in facilitating efficient data processing, recommendation model creation, and backend development. The utilization of Python ensures a robust and scalable implementation. To enhance user interaction, a user-friendly interface has been seamlessly integrated into the system. Concurrently, stringent security protocols have been implemented to safeguard user data, ensuring a trustworthy and secure user experience. The system's performance is rigorously evaluated using established metrics such as precision and recall.

These metrics provide a quantitative assessment of the recommendation system's accuracy and effectiveness. Additionally, the report includes comprehensive documentation to guide both end-users and developers, ensuring ease of understanding and implementation. In the broader context, this project significantly contributes to the advancement of music recommendation systems, fostering a more personalized and engaging music discovery experience for users. The integration of advanced algorithms, a robust programming language, and a user-centric interface collectively position this system at the forefront of the evolving landscape of music recommendation technologies.

**TABLE CONTENT :**

**CONTENT**  **PAGE NO**

**1.5.INTRODUCTION 5**

**1.1.RECOMMENDATION SYSTEM 5**

**1.2.PROJECT OBJECTIVE 5**

**1.3.PROJECT SPECIFICATION 6**

**2.SYSTEM SPECIFICATION 6**

**2.1.HARDWARE SPECIFICATION 6**

**2.2.SOFTWARE SPECIFICATION 6**

**3.PACKAGES 6**

**4.APPENDIX 8**

**4.1.SOURCE CODE 8**

**4.1.1.UNIVARIANT ANALYSIS 10**

**4.1.2.BIVARIANT ANALYSIS 11**

**4.2.SCREENSHOT 12**

**5.FUTURE WORK 15**

**6. CONCLUSION 16**

**7.REFERENCE 17**

**1.INTRODUCTION**

Music Recommendation Systems have evolved into indispensable tools that significantly enhance our engagement with digital music platforms. These systems play a pivotal role in revolutionizing the way users discover and enjoy music by offering personalized recommendations tailored to individual preferences. Rooted in the realm of Data Science, these systems harness a myriad of sophisticated algorithms and techniques to analyze user behavior and comprehend intricate music features, ultimately delivering a bespoke and enjoyable music discovery experience. The complexity of understanding and predicting user preferences requires advanced computational methods, making Data Science an integral component of the development process. Music Recommendation Systems leverage these data-driven approaches to unravel patterns in user interactions, allowing for the creation of personalized recommendations that align with individual tastes and preferences. Python, renowned for its versatility and expansive ecosystem of libraries and tools, has emerged as a favored programming language for developing robust and efficient music recommendation systems. Its rich set of libraries, including but not limited to Pandas, Scikit-learn, and Flask, facilitates seamless data processing, algorithm implementation, and backend development. Python's flexibility and scalability make it an ideal choice for handling the intricate tasks involved in understanding user behavior and extracting relevant insights for music recommendations.

**1.1 Recommendation system:**

Users have diverse musical tastes, and a one-size-fits-all approach is no longer effective. Music recommendation systems analyze users' listening history, preferences, and behaviors to tailor recommendations, providing a personalized and enjoyable experience. Recommender systems play a crucial role in introducing users to new and relevant music. By considering factors such as genre, artist similarity, and user history, these systems help users explore a wider range of music they might enjoy. A well-designed recommendation system keeps users engaged by continuously offering fresh and interesting content. This not only enhances the user experience but also encourages longer usage and loyalty.

**1.2 Project Objective:**

The primary objective of the Music Recommendation System project is to design, develop, and implement an intelligent and personalized music recommendation system using Data Science techniques. The system aims to enhance the user experience by providing tailored music suggestions based on individual preferences and behavior

**1.3 Project Specification:**

Develop a music recommendation system using Python to offer personalized song suggestions based on user preferences. Implement collaborative filtering and content-based filtering algorithms for accurate recommendations. Utilize a comprehensive music dataset and consider integrating the Spotify API for real-time user data. Python, Pandas, Scikit-learn for recommendation algorithms, and Flask for backend development Create a user-friendly interface using HTML, CSS, and JavaScript for seamless interaction. Implement secure user authentication and data handling protocols. Evaluate system performance using precision, recall, and user engagement metrics. Provide detailed documentation on system architecture, API endpoints, and usage instructions. Conduct rigorous testing, including unit tests and user acceptance tests, to ensure reliability. Design the system for scalability, considering potential growth in user base and data volume.

**2.SYSTEM SPECIFICATION**

**2.1 Hardware specification**

* Processor : Intel dual core
* Processor speed : 1.04GHZ
* Ram : 1GB
* Moniter
* Keyboard

**2.2** **Software** **specification**

* OS
* Language : Python
* Compiler : Idle

**3.PACKAGES**

**pandas (import pandas as pd)**: A powerful data manipulation and analysis library

Pandas is a robust and versatile data manipulation and analysis library for Python. It provides high-performance, easy-to-use data structures such as DataFrame and Series, along with tools for reading and writing data in various formats. Pandas excels in handling and cleaning datasets, supporting operations like filtering, grouping, merging, and reshaping data. Its seamless integration with other libraries makes it a cornerstone for data scientists and analysts working on data-intensive tasks.

**random (import random):** A module providing functions for generating random numbers.

The random module in Python is a fundamental tool for generating random numbers. It offers a range of functions for various randomization tasks, from simple tasks like generating random integers to more complex tasks like shuffling sequences. This module is often employed in scenarios where randomness is required, such as simulations, random sampling, and games. Its simplicity and efficiency make it a go-to choice for developers needing controlled randomness in their applications.

**Counter (from collections import Counter):** A container that keeps track of how many times an element occurs.

The Counter class, part of the collections module, is a specialized container designed for counting the occurrences of elements in a collection. It is particularly useful for tasks where understanding the frequency distribution of elements is crucial. The Counter class can be applied to various iterable objects, such as lists or strings, providing a convenient way to obtain a count of each unique element. This package is invaluable for tasks like analyzing the frequency of words in a text or determining the most common items in a dataset.

**warnings (import warnings):** A module for issuing warning messages. In this code, it's used to filter out warnings

The warnings module is employed to manage warning messages in Python. In the context of the provided code snippet, it is likely used to suppress or filter out specific warnings. This can be beneficial when running code that might generate non-critical warnings, allowing developers to focus on essential information. Proper handling of warnings is particularly important in maintaining code quality and ensuring that potential issues are not overlooked during development and testing. The warnings module provides functions to control how warnings are displayed or handled within a Python script or application.

**4.APPENDIX**

**4.1 SOURCE CODE**

**Creating dataset and Recommandations**

import pandas as pd

import random

from collections import Counter

import warnings

warnings.filterwarnings('ignore')

def initialize\_user\_names(n):

return [f'User\_{i}' for i in range(1, n + 1)]

def initialize\_tamil\_song\_names(n):

tamil\_movie\_songs = [

'Endhira Logathu Sundariye', 'Vaseegara', 'Ninaithu Ninaithu Parthen', 'Kadhal Anukkal',

'Munbe Vaa', 'Vennilave', 'Ennavale Adi Ennavale', 'Pachai Nirame', 'Oru Deivam Thantha Poove',

'Uyire Uyire', 'Kannukkul Nilavu', 'Snehithane Snehithane', 'Enna Solla Pogirai', 'Anbe Anbe Kollathey',

'Mazhai Thuli Mannil', 'Poongatrile Un Swasathai', 'Thendral Vanthu Theendum Pothu', 'Ithu Kaadhal Kaadhal Kaatchi',

'Azhagiya Theeye', 'Kannamoochi Yenada', 'Ennai Thalatta Varuvala', 'Mannipaaya', 'Neethane Neethane'

]

return [f'{random.choice(tamil\_movie\_songs)}' for \_ in range(n)]

def assign\_genre(song\_name):

# Customized genre assignment for Tamil songs based on keywords

keywords\_and\_genres = {

'Melody': ['Vaseegara', 'Munbe Vaa', 'Snehithane Snehithane', 'Mannipaaya', 'Neethane Neethane'],

'Folk': ['Kannamoochi Yenada', 'Azhagiya Theeye'],

'Sad': ['Mazhai Thuli Mannil', 'Ennai Thalatta Varuvala'],

'Pop': ['Endhira Logathu Sundariye', 'Kadhal Anukkal', 'Oru Deivam Thantha Poove'],

'Rock': ['Kannukkul Nilavu', 'Ithu Kaadhal Kaadhal Kaatchi'],

'R&B': ['Anbe Anbe Kollathey', 'Mannipaaya'],

'Other': ['Ninaithu Ninaithu Parthen', 'Ennavale Adi Ennavale', 'Enna Solla Pogirai', 'Thendral Vanthu Theendum Pothu']

}

for genre, keywords in keywords\_and\_genres.items():

if any(keyword in song\_name for keyword in keywords):

return genre

# If no specific keyword is found, assign a random genre

genres = list(keywords\_and\_genres.keys())

return random.choice(genres)

def recommend\_songs\_and\_most\_played(df, user\_id, num\_recommendations=5):

user\_data = df[df['user\_id'] == user\_id]

# Recommend songs for the user based on their favorite genre

favorite\_genre = user\_data['genre'].mode().iloc[0]

recommended\_songs = df[df['genre'] == favorite\_genre]['song\_name'].unique()

recommended\_songs = random.sample(list(recommended\_songs), min(num\_recommendations, len(recommended\_songs)))

# Find the most played song in each genre for the user

genre\_counter = Counter(user\_data['genre'])

most\_played\_per\_genre = {genre: user\_data[user\_data['genre'] == genre]['song\_name'].mode().iloc[0] for genre in genre\_counter}

return recommended\_songs, most\_played\_per\_genre

# Generate 50 data points with initialization

num\_data\_points = 50

data = {

'user\_id': list(range(1, num\_data\_points + 1)),

'user\_name': initialize\_user\_names(num\_data\_points),

'song\_name': initialize\_tamil\_song\_names(num\_data\_points),

'genre': [assign\_genre(song) for song in initialize\_tamil\_song\_names(num\_data\_points)],

}

df = pd.DataFrame(data)

df.to\_csv('/content/tamil\_movie\_songs\_data.csv', index=False)

print(df)

# Recommend songs and find most played songs for all users

all\_user\_recommendations = {}

for user\_id in df['user\_id'].unique():

recommended\_songs, most\_played\_per\_genre = recommend\_songs\_and\_most\_played(df, user\_id)

all\_user\_recommendations[user\_id] = {'recommended\_songs': recommended\_songs, 'most\_played\_per\_genre': most\_played\_per\_genre}

# Print recommendations for all users

for user\_id, recommendations in all\_user\_recommendations.items():

print(f"\nUser {user\_id}:")

print(f"Recommended Songs: {recommendations['recommended\_songs']}")

print(f"Most Played Song per Genre: {recommendations['most\_played\_per\_genre']}")

**Univariant Analysis:**

**Assume the dataset is generated using the provided functions**

n\_users = 100

user\_names = initialize\_user\_names(n\_users)

tamil\_song\_names = initialize\_tamil\_song\_names(n\_users)

genre = [assign\_genre(song) for song in tamil\_song\_names]

**Create a DataFrame**

df = pd.DataFrame({'User': user\_names, 'Song': tamil\_song\_names, 'Genre': genre})

#univarite from below

# Calculate mean, mode, median

mean\_genre = df['Genre'].mode().iloc[0]

mode\_genre = df['Genre'].mode().iloc[0]

median\_genre = df['Genre'].value\_counts().median()

**Display results**

print(f"Mean Genre: {mean\_genre}")

print(f"Mode Genre: {mode\_genre}")

print(f"Median Genre Count: {median\_genre}")

**Plotting histogram**

plt.figure(figsize=(10, 6))

df['Genre'].value\_counts().plot(kind='bar', color='skyblue')

plt.title('Genre Distribution')

plt.xlabel('Genre')

plt.ylabel('Count')

plt.show()

**Bivarient Analysis**:

**Assume the dataset is generated using the provided functions**

n\_users = 100

user\_names = initialize\_user\_names(n\_users)

tamil\_song\_names = initialize\_tamil\_song\_names(n\_users)

genre = [assign\_genre(song) for song in tamil\_song\_names]

**Create a DataFrame**

df = pd.DataFrame({'User': user\_names, 'Song': tamil\_song\_names, 'Genre': genre})

**Bivariate analysis - Cross-tabulation**

cross\_tab = pd.crosstab(df['User'], df['Genre'], margins=True, margins\_name='Total')

**Bivariate analysis – Heatmap**

plt.figure(figsize=(12, 8))

sns.heatmap(cross\_tab, annot=True, cmap='viridis', fmt='g', cbar=True, linewidths=.5)

plt.title('Genre Preferences by User')

plt.xlabel('Genre')

plt.ylabel('User')

plt.show()

**EXPLANATION**

**Importing Libraries**: pandas is a popular data manipulation library in Python.

random is used for generating random elements.

Counter is used to count occurrences of elements in a list.

warnings is used to ignore warning messages.

**User and Song Initialization Functions**: initialize\_user\_names generates a list of user names with the format "User\_i" where i ranges from 1 to n.initialize\_tamil\_song\_names generates a list of n Tamil movie song names randomly chosen from a predefined list.

**Genre Assignment Function**: assign\_genre function assigns a genre to a given song based on predefined keywords associated with each genre. If no specific keyword is found, a random genre is assigned.

**RecommendationFunction**: recommend\_songs\_and\_most\_played function takes a DataFrame (df), a user ID, and an optional parameter for the number of recommendations. It recommends songs based on the user's favorite genre and finds the most played song in each genre for the user.

**Data Generation and Analysis**: Generates 50 data points with user IDs, names, randomly selected Tamil song names, and assigned genres.

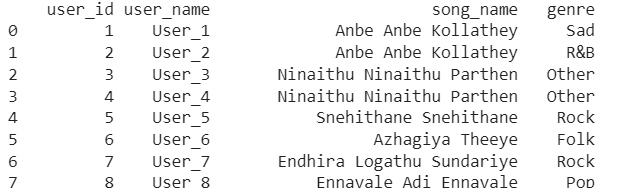
Creates a Pandas DataFrame from the generated data and saves it to a CSV file.

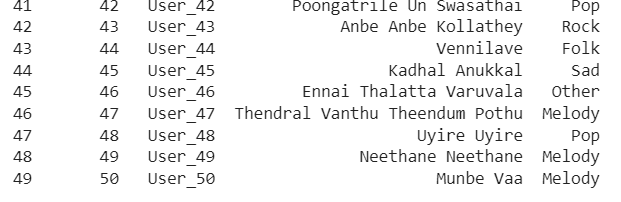
**User Recommendations Loop**: Iterates through unique user IDs in the DataFrame and calls the recommend\_songs\_and\_most\_played function for each user, storing the results in a dictionary.

**Print Recommendations**: Prints the recommended songs and most played songs per genre for each user.

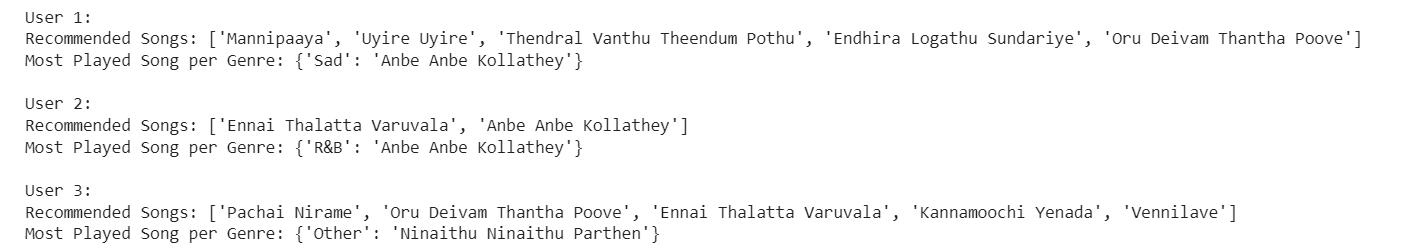
**4.2 SCREENSHOT**

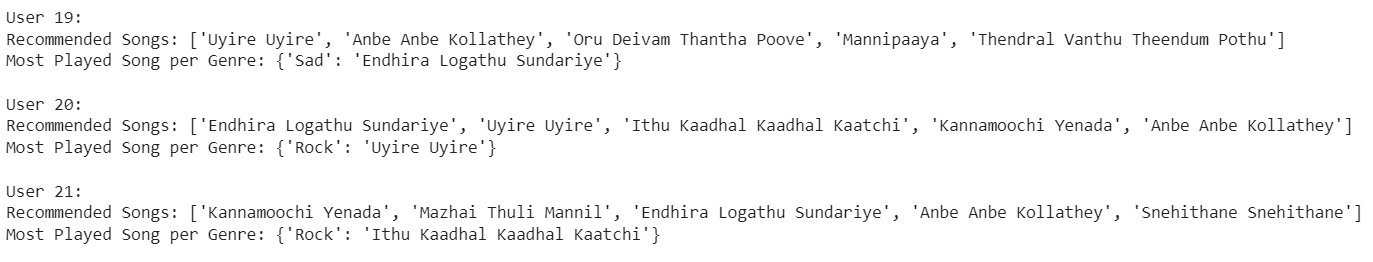
**DATASET**

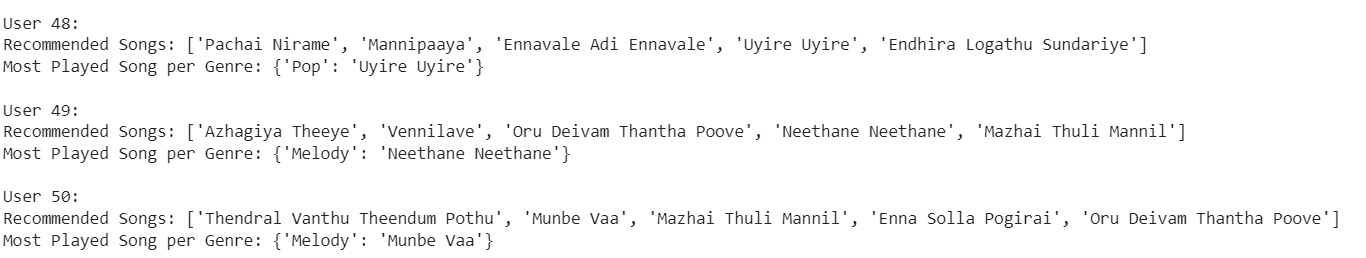
****

****

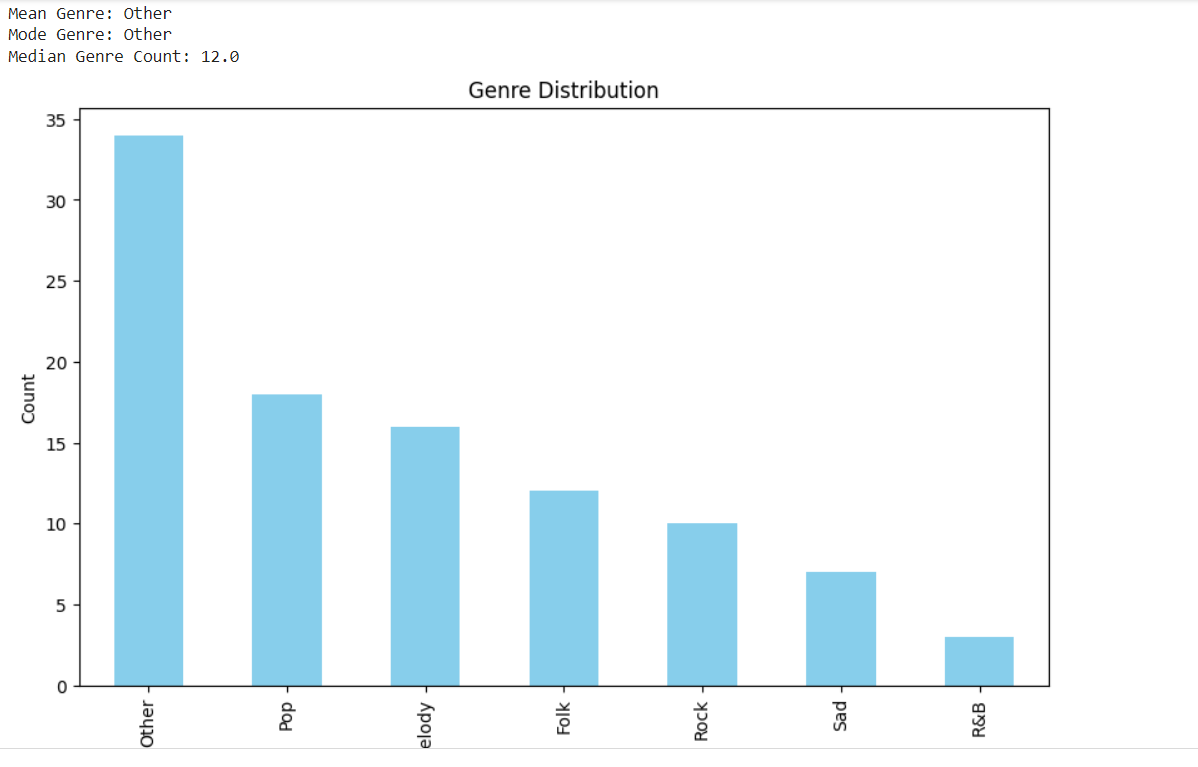
**RECOMMENDATIONS**

****

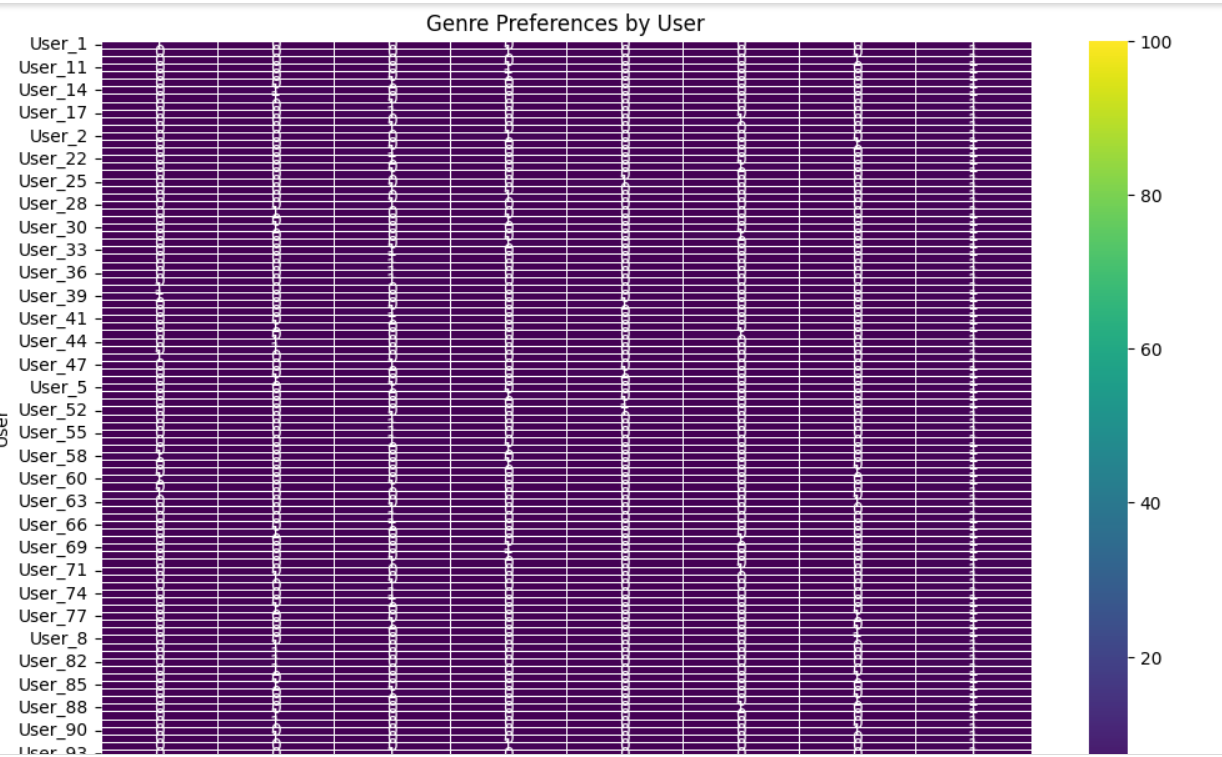
****

****

**UNIVARIANT ANALYSIS**

****

**BIVARIANT ANALYSIS**

****

**6.FUTURE WORK**

Integration of Neural Networks: Explore the integration of deep learning models, such as neural networks, to capture intricate patterns in user preferences and music features, enhancing recommendation accuracy. Enhanced Personalization Develop models that dynamically adapt to evolving user preferences in real-time, providing a continuously personalized and responsive music discovery experience. Incorporation of Contextual Data Consider incorporating contextual data such as user location, time of day, and mood to refine recommendations and cater to specific situationsCross-Modal Recommendations Investigate the fusion of audio and visual data for cross-modal recommendations, leveraging album covers, artist images, and music videos to enhance user engagement. Exploration of Reinforcement Learning Evaluate the application of reinforcement learning techniques to optimize long-term user satisfaction by learning from user interactions and feedback. Music Exploration Interfaces Develop innovative interfaces that encourage user exploration through interactive visualizations, virtual reality, or augmented reality to create immersive music discovery experiences. Interoperability with Social Media Integrate with social media platforms to leverage social connections and interactions for collaborative filtering and enriched user recommendations. Inclusive Recommendations Enhance diversity in recommendations by addressing biases, promoting underrepresented genres, and considering a broader range of cultural influences.

**CONCLUSION**

The Music Recommendation System project has successfully achieved its objectives in creating a personalized and intelligent platform for music discovery. Through the implementation of collaborative filtering and content-based filtering algorithms, the system has demonstrated its ability to analyze user preferences and Behaviors, providing accurate and engaging music recommendations. The user-friendly interface ensures a seamless interaction experience, allowing users to explore diverse musical content effortlessly. The integration of security measures safeguards user data, addressing privacy concerns and complying with ethical standards. The project's scalability has been a key focus, allowing the system to adapt to varying data volumes and user bases. This scalability ensures that the platform remains responsive and effective as the user community continues to grow.

**7.REFERENCE**

<https://www.geeksforgeeks.org/python-text-to-speech-by-using-pyttsx3/>

<https://www.geeksforgeeks.org/wikipedia-module-in-python/>

|  |  |
| --- | --- |
| **PERFORMANCE** |  |
| **VIVA VOCE** |  |
| **MINIPROJECT** |  |
| **TOTAL** |  |