Université Claude Bernard Lyon 1

Machine Learning

Report Recommendation Project

Realized By :
Saad BELAOUAD
Anouar ZAHRAN
Adnan EL MOUTTAKI
Amine IDELHAJ

Under the supervision of :
Dr.Bruno Yun

6 décembre 2023

Table des matières

1 Introduction

Brief Overview

Music streaming platforms are increasingly focusing on personalization to enhance user experience. This project develops a recommendation system that suggests songs based on user preferences and provides explanations for these recommendations. The objective is to create an engaging, transparent system that users trust and enjoy.

Scope of the Report

This report details the journey from conceptualization to the deployment of our music recommendation system. It includes :

— Exploratory Data Analysis : Assessing user preferences and identifying popular trends.

— Building the Recommendation Engine : The methodologies and technologies utilized in crafting our recommendation system.

— Deployment : The process of making our system accessible online.

— Explainability : Integrating a component that elucidates the rationale behind song recommendations.

We aim to provide insights into both the technical processes and the learning outcomes from this project.

2 Exploratory Data Analysis

Dataset Description

The dataset we are working with is a comprehensive collection of music listening data, structured to provide insights into user preferences and listening habits. It includes key information such as the unique ID of each listener ("user"), the ID of each song listened to ("song"), the frequency of listens per song by each user("listen_count"),along with descriptive details like the song's title ("title"), the album it belongs to ("release"), the artist's name ("artist_name"), and the year of the song's release ("year"). This rich dataset serves as the foundation for our analysis and the development of our personalized music recommendation system.

3 Data Processing

In the preprocessing stage, our rigorous examination confirmed the dataset's high quality, with 1,026,270 entries spanning 7 columns, and ready for analysis. Key findings include :

— No missing values across all columns, indicating a dataset well-maintained and primed for modeling without the need for data imputation.

— Descriptive statistics of play_count and year columns showed an average play count of approximately 2.87, but with a significant range peaking at 1890 plays, suggesting the presence of very active listeners or potential outliers.

— The 'year' column displayed entries from 0 to 2010 ; the year 0 entries are evidently erroneous and earmarked for correction.

— The dataset boasts a rich variety, with 44,358 unique songs and 24,042 unique users, reflecting a broad spectrum of user interactions.

4 Analysis

In this analysis, we delved into the music listening patterns captured in the dataset, focusing on three key aspects : Most Listened Songs, Most Popular Artists, and the Distribution of Song Count for Users.

Most listened Songs
We identified and visualized the top 10 most listened songs, revealing the songs that garnered the highest total play counts. The bar plot showcases the significant differences in play counts among these top-ranking songs, providing insights into user preferences.

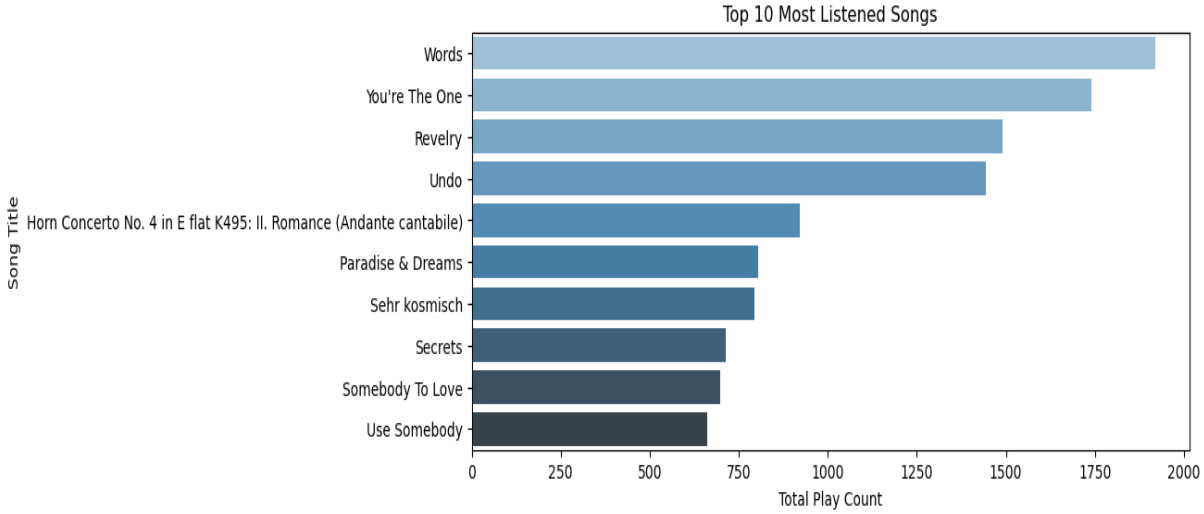

FIGURE 1 – Most listened songs

Most popular Artists
Our exploration extended to the identification of the most popular artists, determined by the count of distinct songs attributed to each artist. The resulting bar plot highlights the artists who consistently captivate listeners, showcasing their prominence in the dataset.

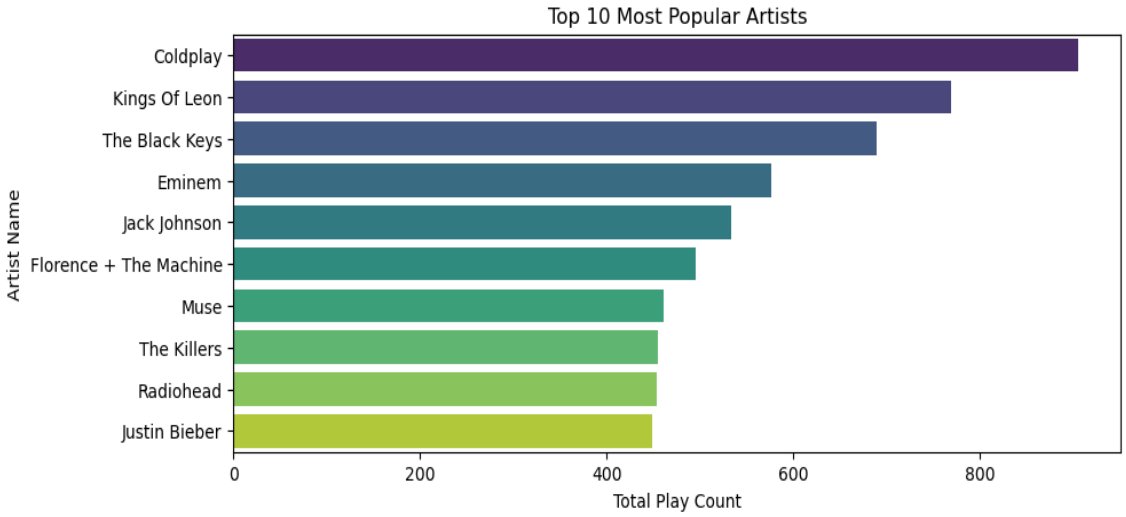

FIGURE 2 – Most popular artists

Distribution of Song Count for Users :
To understand the user engagement, we examined the distribution of song counts per user. The histogram illustrates the frequency of users based on the number of songs they have played, offering a comprehensive overview of user engagement patterns. The majority of users appear to have a moderate song count, with a gradual decline in frequency as the song count per user increases.

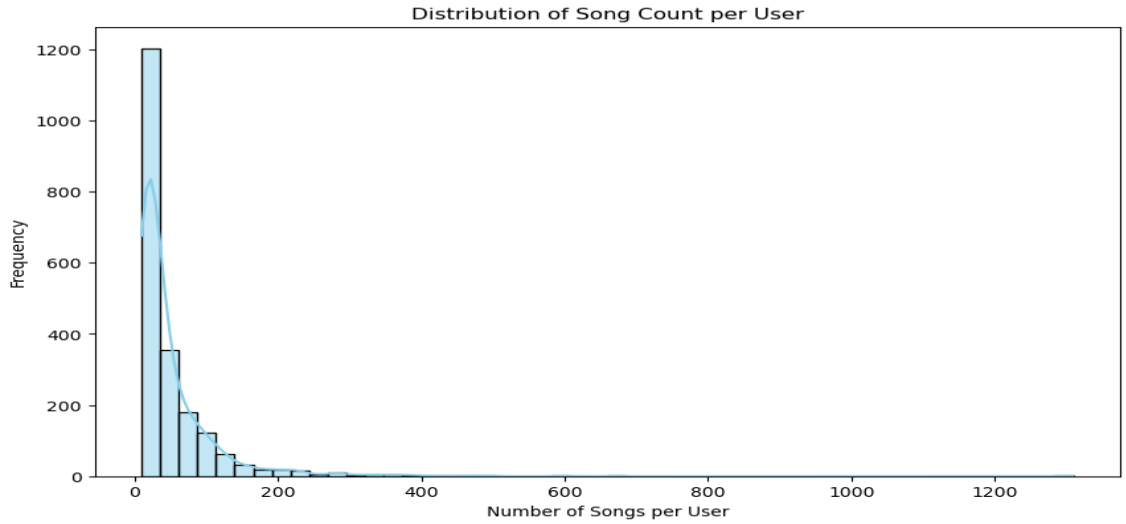

FIGURE 3 – Distribution of Song Count for Users

Findings and Insights
Overall, the analysis uncovered diverse listening habits among users, with certain songs and artists dominating the platform. The distribution of song counts per user provides insights into the variety of user engagement levels. Additionally, it may be beneficial to consider these patterns in the context of user satisfaction and platform dynamics.

5 Fundamentals of Recommendation Systems : Emphasizing Collaborative Filtering

In our research, we encountered a variety of articles that outlined the classification of recommendation techniques, pivotal to the framework of contemporary recommender systems. The methodology is typically categorized into three principal filtering techniques : content-based, collaborative, and hybrid approaches. Content-based filtering recommends items by analyzing the content of the items and the preferences exhibited by the user, focusing on the attributes of the items themselves. Collaborative filtering, which can be either model-based or memory-based, suggests items based on the preferences of similar users. Model-based collaborative filtering employs sophisticated algorithms, including clustering, association techniques, Bayesian networks, and neural networks, to infer user preferences. In contrast, memory-based collaborative filtering operates on the principle of user similarity (user-based) or item similarity (item-based), leveraging existing user-item interactions. Hybrid techniques combine elements from both content-based and collaborative methods to improve recommendation quality and overcome the inherent limitations of each individual approach. Fig. 2 shows the anatomy of different recommendation filtering techniques.

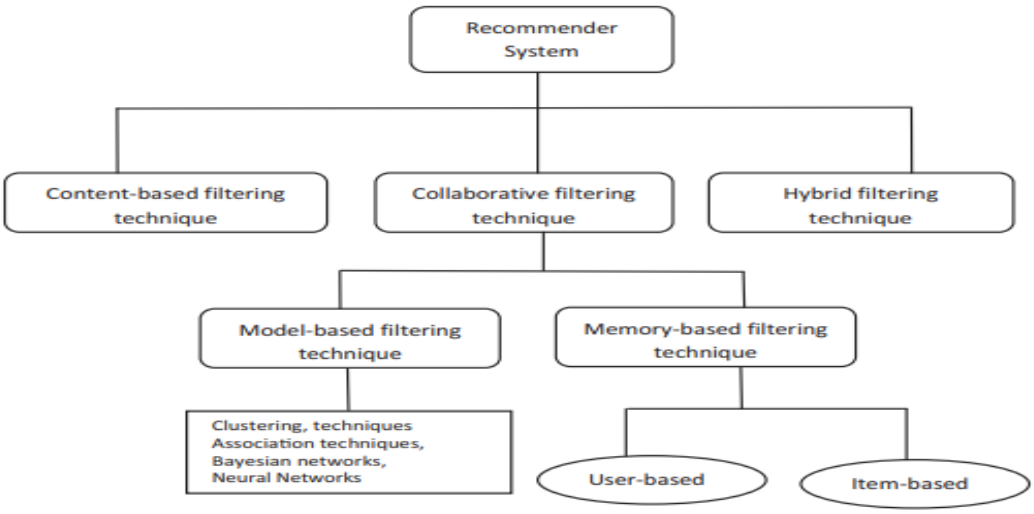

FIGURE 4 – Recommendation System

We will focus only on Collaborative filtering that we are using for our project : Collaborative filtering is a domain-independent prediction technique for content that cannot easily and adequately be described by metadata such as movies and music. Collaborative filtering technique works by building a database (user-item matrix) of preferences for items by users. It then matches users with relevant interest and preferences by calculating similarities between their profiles to make recommendations [2]. Such users build a group called neighborhood. An user gets recommendations to those items that he has not rated before but that were already positively rated by users in his neighborhood. Recommendations that are produced by CF can be of either prediction or recommendation. Prediction is a numerical value, Rij, expressing the predicted score of item j for the user i, while Recommendation is a list of top N items that the user will like the most as shown

in Fig. 4. The technique of collaborative filtering can be divided into two categories : memory-based and model-based [1]

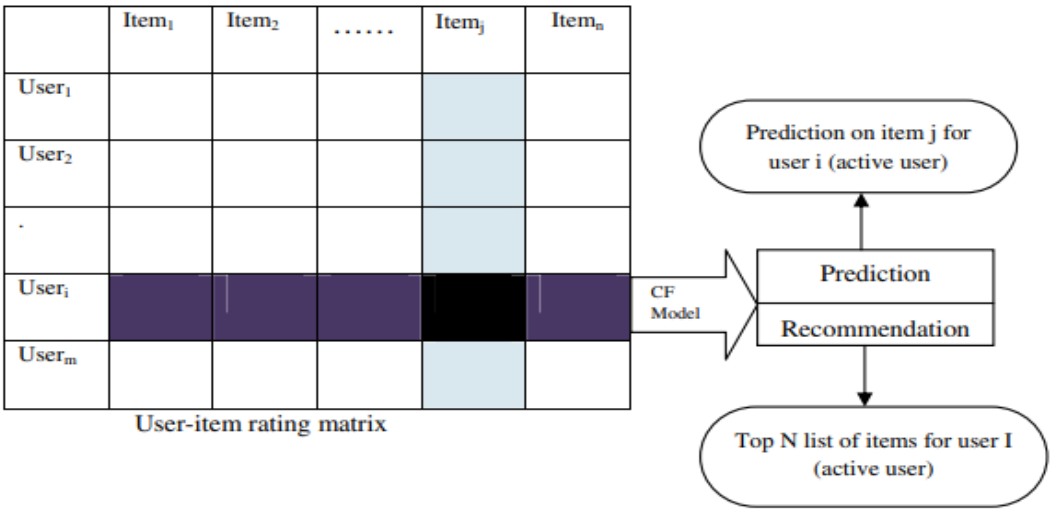

FIGURE 5 – User-Item Rating Matrix

For our project, we have chosen to utilize the collaborative filtering technique for our recommendation system, after careful consideration of our dataset and objectives. This decision is grounded in several key factors : User-Item Interaction Data : Our dataset contains a substantial amount of user-item interaction data, which is well-suited to collaborative filtering. This approach thrives on the richness of user behavior data to infer preferences and make recommendations.

— Preference Learning : Collaborative filtering allows our system to learn from the collective preferences of all users. By identifying patterns and relationships in user behavior, the system can predict what new songs a user might prefer based on what similar users enjoyed.

— No Need for Content Analysis : Unlike content-based methods, collaborative filtering does not require analysis of the content of the items being recommended. This is particularly advantageous when such content data is sparse or when the analysis of content attributes is complex and computationally expensive.

— Scalability and Dynamic Adaptation : Collaborative filtering can scale with our growing dataset and dynamically adapt to changes in user preferences over time. This dynamic nature ensures that our recommendations stay current and relevant.

— Model Versatility : Collaborative filtering models, such as matrix factorization techniques like Singular Value Decomposition (SVD), have proven to be highly effective in many recommendation system applications, providing a strong foundation for our system.

Pros and Cons of collaborative filtering techniques :

Pros and Cons of collaborative filtering techniques Collaborative Filtering has some major advantages over CBF in that it can perform in domains where there is not much content associated with items and where content is difficult for a computer system to analyze (such as opinions and ideal). Also, CF technique has the ability to provide serendipitous recommendations, which means that it can recommend items that are relevant to the

user even without the content being in the user's profile [3]. Despite the success of CF techniques, their widespread use has revealed some potential problems such as follows below :

— The cold-start problem occurs when the system lacks sufficient information about new users or items to make accurate recommendations. Since new profiles have no associated preferences or ratings, the system struggles to determine their tastes.

— The data sparsity problem arises when there are too few ratings given the number of items available. This leads to a sparse user-item matrix, making it difficult to find good neighbor profiles for collaborative filtering, resulting in weaker recommendations and coverage issues, where the system can only generate recommendations for a limited subset of items.

— Scalability refers to the challenge of maintaining recommendation quality as the volume of users and items grows. Many algorithms that perform well with smaller datasets may not provide satisfactory recommendations at a larger scale.

To address these issues, dimensionality reduction techniques like Singular Value Decomposition (SVD) are often employed. SVD can effectively reduce the complexity of the data, allowing for more reliable and efficient recommendations that can scale with the dataset size.

6 Implementation

First Approach : user-item matrix

Upon generating the user-item matrix for our recommendation system, we found that the matrix displayed a high level of disparity. This means that there was a significant variation in the listening habits across different users, with many songs not being listened to by a majority of users. High disparity in the user-item matrix can be challenging for collaborative filtering algorithms, as it may lead to a sparsity problem where there are not enough overlapping listening events between users to accurately predict preferences. This condition necessitates more sophisticated techniques or the incorporation of additional information to enhance the recommendation process.

Here is the results of our user$_matrix$:

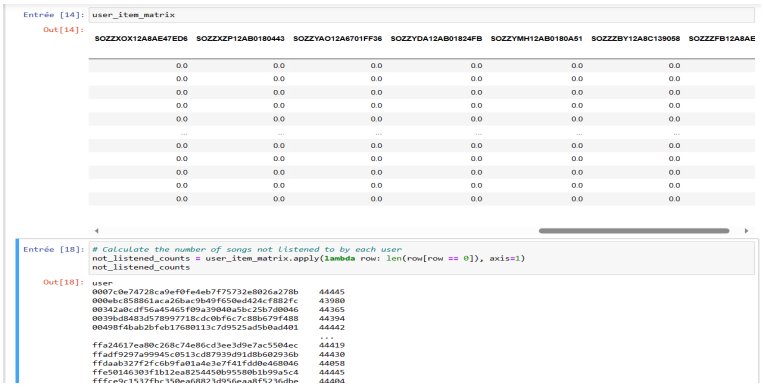

FIGURE 6 – User Item Matrix

Second Approach : Singular Value Decompositio

Singular Value Decomposition (SVD) based on matrix factorization techniques.
We follow these steps to implement a recommendation system using Singular Value Decomposition (SVD) with the Surprise library : Environment Setup :
We import all required libraries and modules, including pandas for data manipulation, numpy for numerical operations, and the Surprise library for recommendation algorithms. The Surprise library's specific classes such as Reader, Dataset, and SVD are crucial for our analysis.

- Data Loading : We load our dataset into a pandas DataFrame to visualize the initial data structure and understand the various data columns, such as user IDs, item IDs, and ratings.
- Data Preprocessing : We preprocess the data by addressing missing values and ensuring data types are consistent. A Reader object is configured to parse the rating scale, and the dataset is then loaded into a Surprise data structure using the Dataset.load_from_df() method.
- Model Configuration : We set up our SVD algorithm parameters, including the number of factors, regularization terms, and iteration counts. We also establish our evaluation criteria, deciding on metrics like RMSE or MAE for model performance assessment. Model Training and Validation : We split our data into training and test sets to validate our model's performance. The SVD model is trained on the training set, and we utilize cross-validation techniques to ensure robustness.
- Hyperparameter Tuning : We conduct hyperparameter tuning using grid search methods provided by GridSearchCV within the Surprise library, optimizing parameters to improve the model's predictive accuracy.
- Model Optimization : We perform cross-validation on the optimized model to ensure its stability and to evaluate its generalization performance.
- Model Evaluation : We exclusively use the Root Mean Square Error (RMSE) to evaluate our model's accuracy. RMSE provides us with a single, comprehensive measure of the model's prediction error magnitude. Prediction and Recommendation Generation : We use the trained model to predict the ratings for user-item pairs. For users, we generate a list of recommended items by predicting ratings for items they have not yet interacted with and selecting the top-rated items.
- Result Interpretation : We interpret the recommendation results, ensuring that they are aligned with user preferences and behavior patterns observed in the dataset. This step also involves a qualitative assessment of the recommendations.
- Model Export : We save our trained model using the dump function from Surprise, allowing us to reuse the model for future predictions without retraining.

Results & Discussion :

In our recommendation system, achieving an RMSE of 2.21 is a promising result, particularly given the complexity and variability inherent in user preferences. Within the context of a 1 to 10 rating scale, this level of RMSE indicates that our model's predictions are reasonably close to the actual ratings, with an average error margin that is within an acceptable range for many practical applications.
This RMSE value reflects the robustness of our Singular Value Decomposition (SVD) approach in capturing the underlying patterns within the user-item interaction data. It

underscores the model's ability to effectively discern and predict user preferences, even in the face of diverse and sparse data.
Furthermore, an RMSE of 2.21 lays a solid foundation for our system, offering a good balance between accuracy and computational efficiency. It suggests that users are likely to receive recommendations that are relevant and, most importantly, reflective of their tastes, which is crucial for a positive user experience.
As we continue to refine our system, we can draw on this initial success, aiming to further enhance the precision of our recommendations.

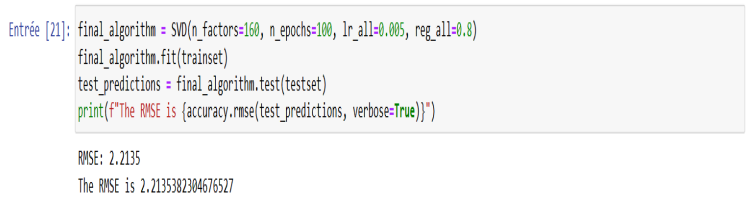

Figure 7 – RMSE

7 Deployment

Tools :
In our project, we have judiciously incorporated a combination of tools and platforms to develop, refine, and deploy our recommendation system. The following are the key components we have used :
Python : We selected Python as our core programming language for its readability, ease of use, and extensive support for data analysis and machine learning. Python's rich ecosystem of libraries has been instrumental in the development and implementation of our project.
Surprise Library : For building the recommendation system, we utilized the Surprise library, a Python toolkit designed specifically for developing and analyzing recommender systems. It offers a range of algorithms, including matrix factorization techniques like SVD, and tools for evaluating their performance. This has been essential in creating an efficient and accurate recommendation engine.
Streamlit : To deploy our recommendation model as an interactive web application, we employed Streamlit. This library enables the rapid conversion of data science scripts into user-friendly web apps. Streamlit's simplicity and interactivity have allowed us to create an accessible and engaging interface for our recommendation system without the need for deep web development expertise.
GitHub for Hosting : To host our project, we chose GitHub. GitHub not only serves as a repository for our codebase, ensuring version control and collaborative development, but also as a hosting platform. This has enabled us to manage our project's code and track changes efficiently while making the application easily accessible to users and collaborators. GitHub's robust platform ensures that our project is securely and reliably hosted, with the added benefit of integrating seamlessly with our development workflow.
Deployment Process
Before we proceeded with the deployment of our music recommendation engine using Streamlit, we worked with a sample of 1000 entries from our dataset. This was a strategic

decision to streamline the development and testing phases of the application, ensuring that all components functioned correctly without the overhead of processing the entire dataset.
To outline the deployment process of our music recommendation engine using Streamlit, the steps we followed are :

- Development : We developed a Streamlit application in Python, integrating functions for loading the dataset, the recommendation model, and generating top song recommendations for users.
- Local Testing : We tested the application locally on our machine using `streamlit run [our_script_name].py` to ensure all components functioned correctly.
- Preparation for Deployment : We prepared the necessary files, including the dataset (`song_dataset.csv`) and the serialized model (`recommandation_model.pickle`), for deployment.
- Hosting Service Selection : We chose an appropriate hosting service like Streamlit Sharing to deploy our app.
- Deployment : For deployment on Streamlit Sharing, we pushed our code to GitHub and connected it to Streamlit Sharing.

Through these steps, we successfully deployed our music recommendation engine online, enabling users to interact with the app by selecting a user from a dropdown menu and viewing personalized song recommendations.

Interface

In our music recommendation system, the interface is designed to be straightforward and user-friendly, primarily utilizing a dropdown menu and a button for user interaction. Here's how it works :

- Dropdown Menu for User Selection : Users can select their profile from a dropdown menu, which is populated with user IDs from our dataset.
- "Get Recommendation" Button : Clicking this button triggers the recommendation model to generate personalized music recommendations.
- Model-Driven Recommendations : The `recommandation_model` processes the selected user ID to output song suggestions tailored to the user's tastes.
- Displaying Recommendations : The recommended songs are displayed in a readable format, allowing users to discover new music aligned with their preferences.

The user interface simplifies interaction with our advanced model, facilitating personalized music discovery with just a few clicks.

Working URL

The URL for your music recommendation system is :
`https://musicrecommandationsystem.streamlit.app/`.
Users can visit this link to interact with the Streamlit-based app and explore personalized music recommendations.

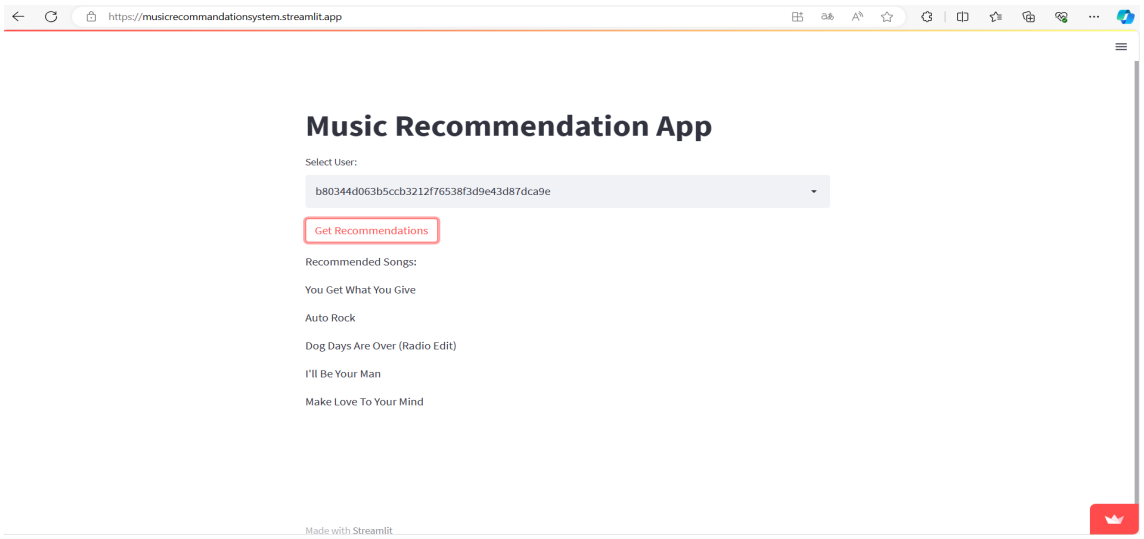

FIGURE 8 – Web Application

The URL for our projects in Github : `https://github.com/amineidel1/Music_Recommandation_System`.

8 Conclusion

In conclusion, our project embarked on the ambitious journey of developing a music recommendation system, aimed at enhancing user experience on music streaming platforms through personalization. The core objective was to create an engaging and transparent system that users could trust and find enjoyable.

Project Summary :

Initial Approach and Challenges : We started by constructing a user-item matrix to analyze user interactions with songs. However, we encountered challenges due to high disparity in the matrix, indicating sparse user-song interactions. This prompted the need for a more advanced approach.

- Adopting SVD and Matrix Factorization : To address the sparsity issue, we implemented Singular Value Decomposition (SVD) based on matrix factorization techniques. This method was chosen for its effectiveness in extracting latent factors from sparse data, enabling us to make more accurate predictions about user preferences.
- Development and Testing : The recommendation system was developed using Python, with a focus on integrating functions for data handling, model integration, and generating top song recommendations. We conducted thorough local testing to ensure the system's functionality and reliability.
- Deployment and Interface Design : For deployment, we utilized Streamlit, an ideal platform for quickly turning data scripts into shareable web apps. The user interface was designed to be straightforward and user-friendly, primarily involving a dropdown menu for user selection and a button to generate recommendations.
- Working with a Data Sample : Initially, we worked with a sample of 1000 entries from our dataset to streamline development and testing. This approach helped in optimizing the performance and testing of the system before full-scale implementation.

— Hosting and Online Interaction : The system was successfully deployed online, allowing users to interact with the application and receive personalized song recommendations. The interface facilitated an easy and engaging interaction for users.
Future Developments :

As we move forward, our team is committed to the continuous development and improvement of our music recommendation system. We plan to :
-Expand the Dataset : Incorporate a larger dataset to enhance the recommendation engine's accuracy and diversify the recommendations.
-Algorithm Optimization : Continuously refine our SVD implementation and explore other machine learning techniques to improve recommendation quality.
-User Feedback Integration : Implement mechanisms to gather and incorporate user feedback, further personalizing the recommendation experience.
-Enhance User Interface : Continuously improve the user interface for better user engagement and experience.

In summary, this project represents a significant step in the realm of personalized music recommendations, and we are enthusiastic about its future development and potential impact on enhancing user experiences in music streaming services.

Références

[1] Jesús Bobadilla, Fernando Ortega, Antonio Hernando, and Abraham Gutiérrez. Recommender systems survey. *Knowledge-based systems*, 46 :109–132, 2013.

[2] Jonathan L Herlocker, Joseph A Konstan, Loren G Terveen, and John T Riedl. Evaluating collaborative filtering recommender systems. *ACM Transactions on Information Systems (TOIS)*, 22(1) :5–53, 2004.

[3] J Ben Schafer, Dan Frankowski, Jon Herlocker, and Shilad Sen. Collaborative filtering recommender systems. In *The adaptive web : methods and strategies of web personalization*, pages 291–324. Springer, 2007.