

**Tech Saksham**

**Capstone Project Report**

**“Spotify Music Recommendation System”**

**“University College of Engineering- BIT Campus, Anna University, Tiruchirappalli”**

|  |  |
| --- | --- |
| **NM ID** | **NAME** |
| **au810021214022** | **PREETHI V** |

|  |  |
| --- | --- |
|  |  |
|  | Ramar Bose |
|  | Sr. AI Master Trainer |

**ABSTRACT**

The Python-powered Spotify Music Recommendation System is meticulously crafted to provide users with tailor-made song suggestions drawn from Spotify's extensive music collection. By leveraging advanced machine learning techniques, particularly K-Means clustering, the system evaluates various song attributes like danceability, energy, loudness, and tempo to group similar tracks together. This clustering approach enables the system to efficiently identify songs with comparable musical characteristics, enhancing the precision of its recommendations based on user preferences.

Upon initiation, users are prompted to input their favorite song, and the system employs fuzzy string-matching methods to pinpoint the closest match within its dataset. Furthermore, users have the flexibility to specify preferred genres, allowing for a more refined recommendation process. With this input, the system dynamically curates a personalized list of recommended tracks, meticulously selected to resonate with the user's musical taste.

Underpinning the recommendation process is a robust framework of data preprocessing, feature extraction, and model training procedures. The K-Means clustering algorithm partitions the dataset into distinct groups based on shared musical attributes, streamlining the retrieval of relevant recommendations. Each song is intelligently assigned to its corresponding cluster, ensuring an efficient and effective recommendation process.

Through seamless integration of data analysis, machine learning, and user interaction, the Spotify Music Recommendation System aims to elevate the music discovery journey for users. Its intuitive interface and personalized recommendations empower music enthusiasts to explore new tracks closely aligned with their unique preferences.

Moreover, beyond merely suggesting popular songs, the system strives to unveil hidden musical gems, fostering a deeper connection between users and their favorite tunes. By facilitating the discovery of diverse and captivating tracks, the system enriches the overall music listening experience, encouraging users to embark on a journey of musical exploration and discovery.

**INDEX**

|  |  |  |
| --- | --- | --- |
| **Sr. No.** | **Table of Contents** | **Page No.** |
| 1 | Chapter 1: Introduction | 4 |
| 2 | Chapter 2: Services and Tools Required | 6 |
| 3 | Chapter 3: Project Architecture | 8 |
| 4 | Chapter 4: Modeling and Project Outcome | 11 |
| 5 | Conclusion | 22 |
| 6 | Future Scope | 23 |
| 7 | References | 24 |
| 8 | Links | 25 |

**25 pages**

**CHAPTER 1**

**INTRODUCTION**

* 1. **Problem Statement**

The Spotify Music Recommendation System is designed to offer users personalized song suggestions tailored to their musical tastes. Utilizing machine learning methods and the vast Spotify dataset, the system examines song characteristics such as danceability, energy, and tempo to categorize songs into clusters with similar attributes. Users engage with the system by inputting their favorite song and, optionally, indicating a preferred genre. Subsequently, the system generates carefully curated lists of recommended tracks that closely match the user's preferences. Critical objectives of the system encompass precise recommendation generation, streamlined data analysis, and preprocessing, intuitive user interaction, and scalability to manage extensive data volumes. In addition to these goals, the system aims to foster a deeper connection between users and the Spotify platform by facilitating music discovery experiences that resonate on a personal level. Furthermore, the system aspires to continuously enhance its recommendation capabilities, adapting to evolving user preferences and delivering increasingly refined suggestions over time.

**1.2 Proposed Solution**

The proposed solution utilizes the Spotify songs dataset sourced from Kaggle to craft personalized song recommendations. Through the utilization of K-Means clustering, data points are organized based on attributes such as danceability, energy, and tempo, aiding in the identification of songs with similar characteristics. Users provide their favorite songs and, optionally, specify preferred genres to fine-tune recommendations. Utilizing the clustering model, the system proposes songs from the same cluster as the user's favorite, ensuring alignment with their musical tastes. Implemented as a standard Python application, the recommender system ensures easy accessibility for users. Continuous refinement, driven by user feedback and performance evaluations, enhances the system's accuracy, thereby enriching the overall music discovery journey for Spotify users.

* 1. **Feature**
* **Real-Time Analysis**: The application will deliver real-time analysis of recommendations using current data.
* **Relative Search**: Offering an enhanced search function where users don't need to input exact phrases.
* **Custom Playlist:** Users can specify the desired number of songs for their playlists.
* **Genre Preferences**: Users can select particular genres to receive recommendations tailored to their preferences.
* **Song IDs**: This feature enables unique search options by utilizing each song's distinctive identification code.
  1. **Advantages**
* **Precision of Data:** This application ensures data accuracy.
* **Ease of Use:** The interface is intuitive and user-friendly.
* **Flexible Song Selection:** Users can obtain any desired number of songs.
  1. **Scope**

The project's scope is extensive and diverse, covering several vital elements. Initially, it involves obtaining and preparing the Spotify dataset, containing song attributes and metadata. Feature engineering is then pivotal, focusing on selecting and extracting relevant features like danceability, energy, and tempo, which form the foundation for clustering and recommendation. The project also includes implementing machine learning models, particularly clustering algorithms such as K-Means, to group songs into clusters based on their features. Following this, a recommendation engine is developed to utilize the clustering outcomes, offering users personalized song suggestions based on their input, including favorite songs and preferred genres. Furthermore, user interaction is critical, requiring the creation of an intuitive interface for seamless engagement with the recommendation system. Deploying the recommendation system as a standalone Python application ensures user accessibility, while continuous evaluation, maintenance, and updates guarantee its ongoing relevance and efficacy in providing personalized music discovery experiences on the Spotify platform.

**CHAPTER 2**

**SERVICES AND TOOLS REQUIRED**

**2.1 Services Used**

* **Data Analysis and Processing:** Processing and analyzing the Spotify dataset to extract relevant features for recommendation.
* **Machine Learning:** Implementing the K-Means clustering algorithm for grouping songs into clusters based on their attributes.
* **User Interface Design:** Designing an interactive interface for users to input their favorite songs and preferred genres.
* **Model Deployment:** Deploying the recommendation system as a standalone Python application for user accessibility.
* **Package Installation:** Ensuring the availability and installation of required packages such as pandas, numpy, plotly, yellowbrick for data manipulation, visualizations, clustering respectively.

**2.2 Tools and Software used**

**Tools**:

* **Pandas:** Used for data manipulation and analysis, including importing the dataset and preprocessing.
* **Numpy:** A fundamental package for scientific computing in Python, providing support for multi-dimensional arrays and matrices, along with a collection of mathematical functions for array operations.
* **plotly:** A versatile Python library for creating interactive and publication-quality graphs and visualizations, offering a wide range of chart types and customization options.
* **Yellowbrick**: Uses danceability, energy, tempo., etc of the song features and calculates the co-relation between them.
* **Matplotlib**: A comprehensive plotting library for Python, widely used for creating static, interactive, and animated visualizations, covering a broad spectrum of plotting needs.
* **Spotipy**: A Python library that provides easy access to the Spotify Web API, enabling seamless integration of Spotify's vast music catalog into the recommendation system.

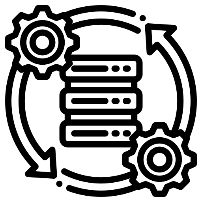
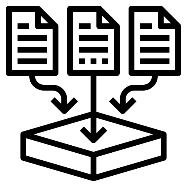
**2.3 Software Requirements**:

* **Python:** The primary programming language for building the recommendation system. Python offers extensive libraries and frameworks for data analysis, machine learning, and web development.
* **Jupyter Notebook or Visual Studio Code:** These integrated development environments (IDEs) provide excellent support for Python development, allowing for code editing, debugging, and visualization.
* **Git and GitHub:** Version control software and a web-based hosting service, respectively, useful for collaboration, code management, and project documentation.

**CHAPTER 3**

**PROJECT ARCHITECTURE**

**3.1 Architecture**

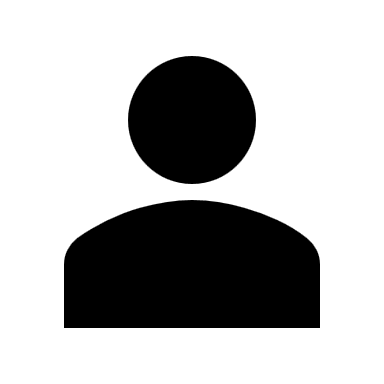
**Flow diagram:**

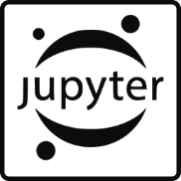
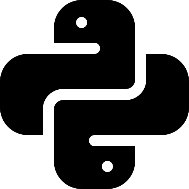
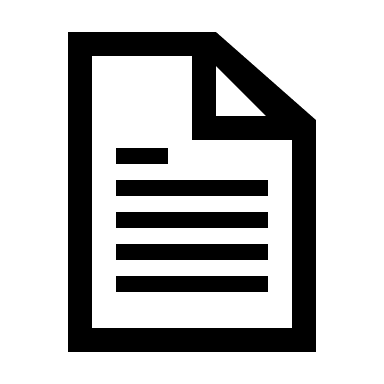
**OUTPUT**

**DATA PROCESSING**

**DATA COLLECTION**

**USER**

****

****

**MODEL ENVIRONMENT**

**DATASET**

**+**

The architecture of the Spotify Music Recommendation System project involves several interconnected components that work together to deliver personalized song recommendations to users. Here's an overview of the project architecture:

**1. Data Acquisition and Preprocessing:**

* **Data Source**:The project begins by obtaining the Spotify songs dataset from platforms like Kaggle or Spotify's API.
* **Data Preprocessing**: The dataset undergoes cleaning, transformation, and feature extraction to prepare relevant attributes like danceability, energy, and tempo for recommendation purposes.

**2. Machine Learning Model:**

* **Clustering Algorithm**: The system employs the K-Means clustering algorithm to categorize songs into clusters based on their attributes.
* **Model Training**: The clustering model is trained using the preprocessed dataset to establish clusters of similar songs.

**3. User Interaction:**

* **User Input**: Users provide their preferences by specifying favorite songs and, optionally, preferred genres.
* **Interface**: An intuitive interface, either through command-line or web-based platforms, facilitates user interaction and input gathering.

**4. Recommendation Engine:**

* **Processing User Input**: The system analyzes user input to extract features from favorite songs and determine preferred genres.
* **Cluster Selection**: Based on the extracted features, the system identifies the cluster containing similar songs.
* **Song Recommendation**: Recommendations are generated by randomly sampling from the selected cluster, ensuring both relevance and diversity.

**5. Output:**

* **Recommended Songs**: The system presents recommended songs to users, typically in a list format, accompanied by relevant details such as song title, genre, and ID.

**6. Deployment:**

* **Standalone Application**: The recommendation system can be deployed as a standalone Python application, providing easy access to users on their local devices.

**7. Continuous Improvement:**

* **Feedback Loop**: User feedback and system performance metrics are gathered to continuously enhance the recommendation system.
* **Model Refinement**: Ongoing updates and refinements to the clustering model and recommendation engine contribute to improving the accuracy and relevance of song recommendations over time.

Through this architecture, the Spotify Music Recommendation System effectively handles user input, utilizes machine learning for song clustering, and delivers personalized recommendations, ultimately enhancing the music discovery journey for users.

**CHAPTER 4**

**MODELING AND PROJECT OUTCOME**

1. **Start Jupyter notebook**

preethi@MacBook-Pro ~ % jupyter notebook

[I 11:49:56.857 NotebookApp] Writing notebook server cookie secret to /Users/preethi/Library/Jupyter/runtime/notebook\_cookie\_secret

[I 2024-04-14 11:49:58.148 LabApp] JupyterLab extension loaded from /Users/preethi/opt/anaconda3/lib/python3.9/site-packages/jupyterlab

[I 2024-04-14 11:49:58.149 LabApp] JupyterLab application directory is /Users/preethi/opt/anaconda3/share/jupyter/lab

[I 11:49:58.152 NotebookApp] Serving notebooks from local directory: /Users/preethi

[I 11:49:58.152 NotebookApp] Jupyter Notebook 6.4.12 is running at:

[I 11:49:58.152 NotebookApp] http://localhost:8888/?token=274fc105e5d7580e3dbae6b26444dc024f8b4330099c0999

[I 11:49:58.152 NotebookApp] or http://127.0.0.1:8888/?token=274fc105e5d7580e3dbae6b26444dc024f8b4330099c0999

[I 11:49:58.152 NotebookApp] Use Control-C to stop this server and shut down all kernels (twice to skip confirmation).

[C 11:49:58.155 NotebookApp]

* To access the notebook, open this file in a browser:

file:///Users/preethi/Library/Jupyter/runtime/nbserver-6317-open.html

or copy and paste one of these URLs:

http://localhost:8888/?token=274fc105e5d7580e3dbae6b26444dc024f8b4330099c0999

or http://127.0.0.1:8888/?token=274fc105e5d7580e3dbae6b26444dc024f8b4330099c0999

1. **Import packages(Numpy, Pandas, plotly) for processing the dataset from Kaggle**

A screenshot of a computer

Description automatically generated

Download the dataset from Kaggle and unzip them

* data\_by\_genres.csv
* data\_by\_year.csv
* data.csv

1. **Read the datasets from Downloads directory using Pandas package.**



1. **Print the dataset info to review the featues such as year the track was created, popularity, name of the track., etc**

A screenshot of a computer

Description automatically generated

1. **Review the various genre’s available in Spotify system.**

A screenshot of a computer

Description automatically generated

1. **Review the songs based on the year they are produced.**

A screenshot of a computer

Description automatically generated

1. **Feature assessment (Characteristics)**

Using the data grouped by year, we can understand how the overall sound of music has changed from 1921 to 2020.

A screen shot of a graph

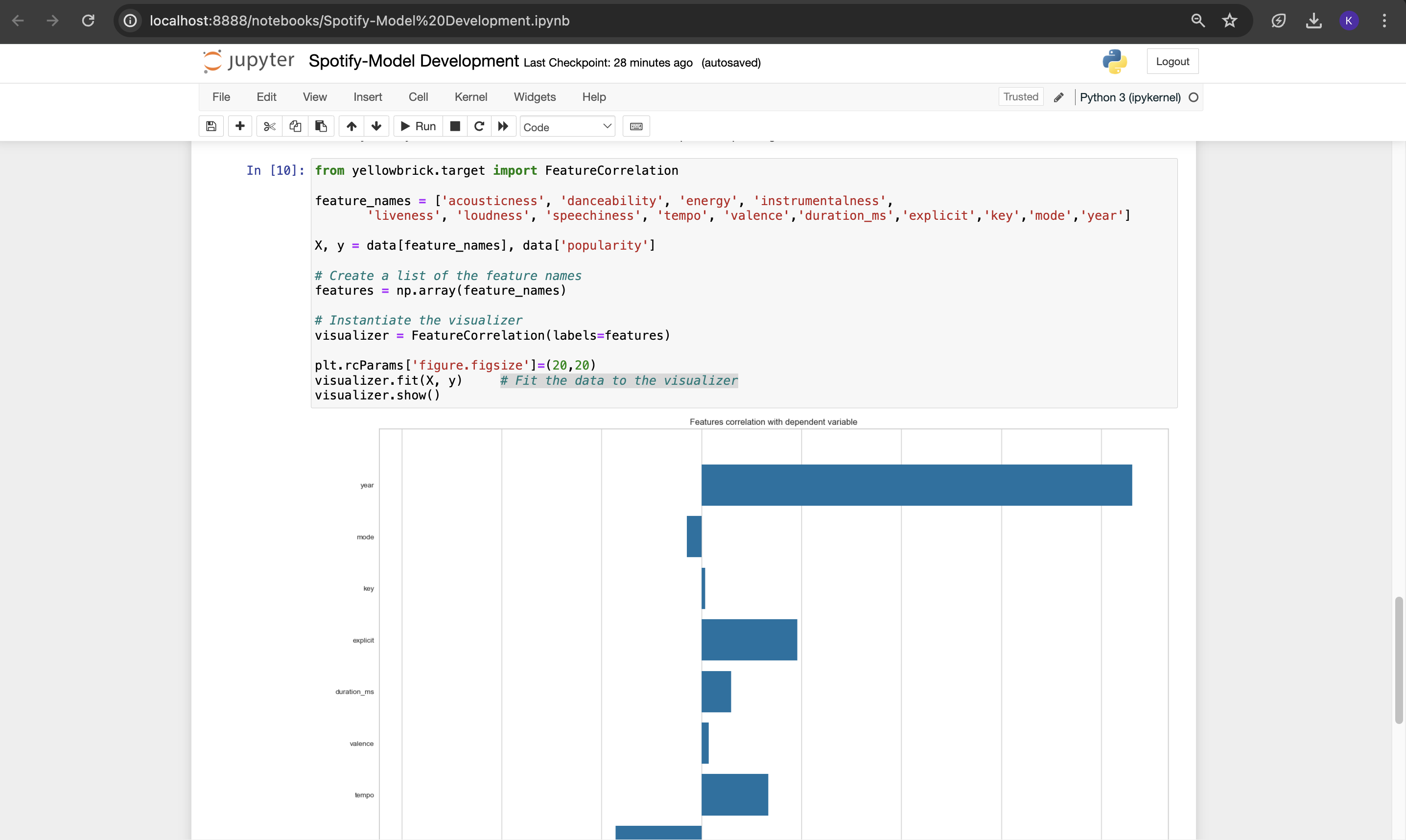
Description automatically generated

1. **Install yellowbrick package for finding co-relation between various features of the dataset.**

A screenshot of a computer

Description automatically generated

1. **Yellow brick uses danceability, energy, tempo., etc of the song features and calculates the co-relation between them.**



A screenshot of a graph

Description automatically generated

This dataset contains the audio features for different songs along with the audio features for different genres. We can use this information to compare different genres and understand their unique differences in sound.

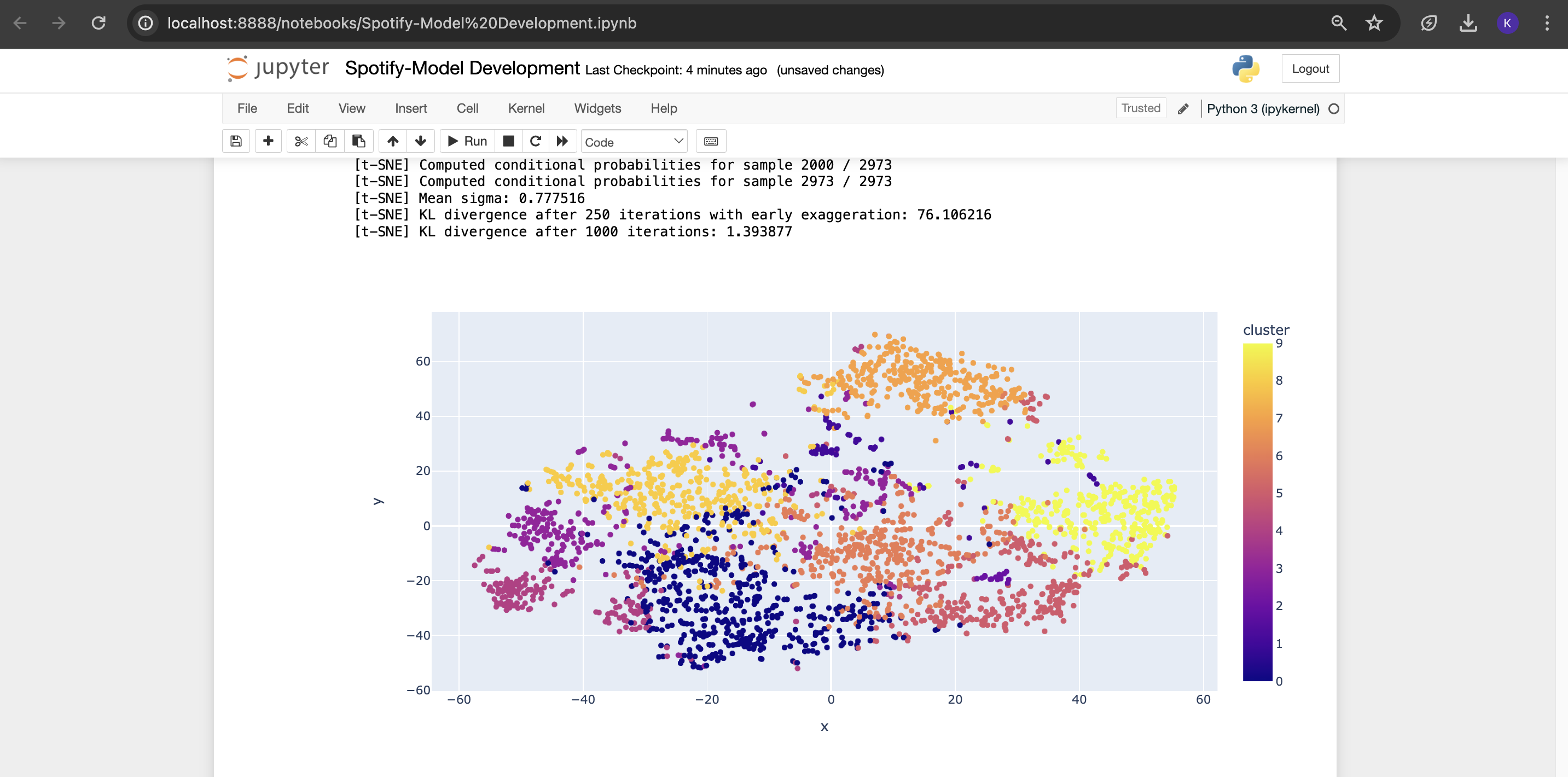
A screenshot of a computer

Description automatically generated

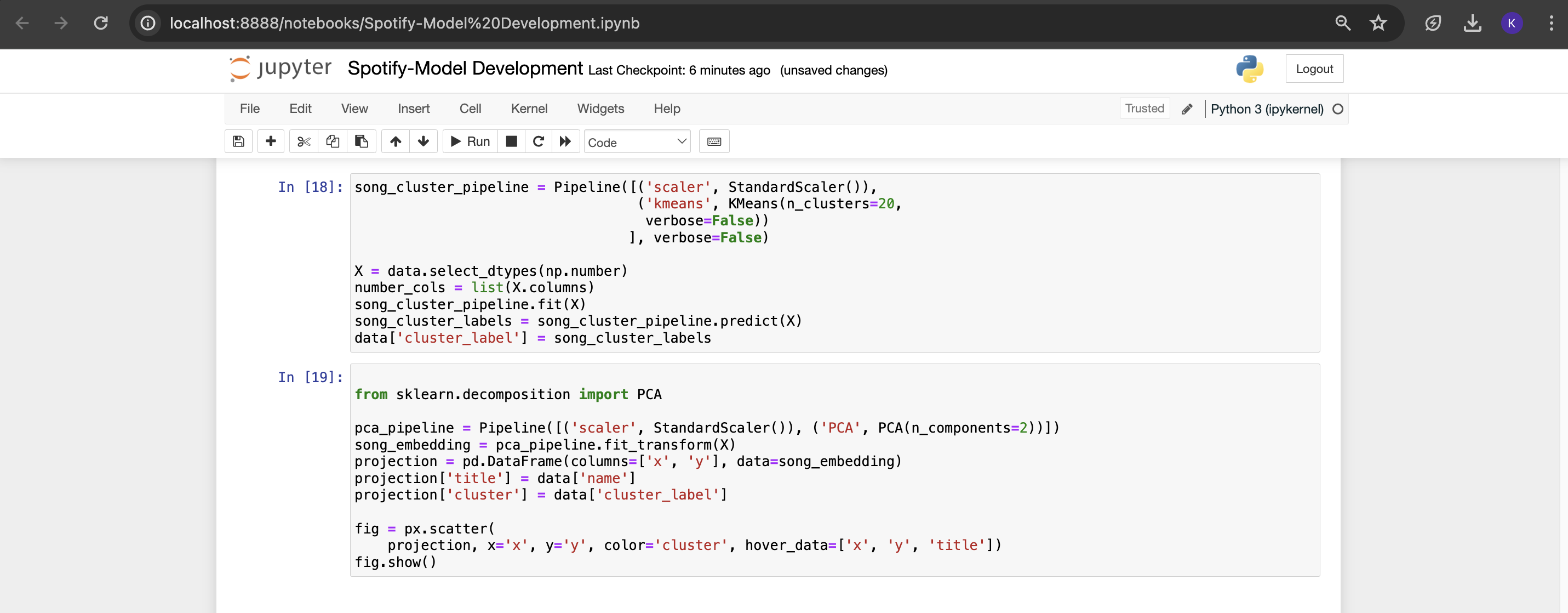
**K-means clustering algorithm** is used to divide the genres in this dataset into ten clusters based on the numerical audio features of each genre.

A screenshot of a computer

Description automatically generated



1. **Clustering Songs with K-Means**



A screen shot of a computer screen

Description automatically generated

1. **Install Spotify package.**

* Based on the analysis and visualizations, it’s clear that similar genres tend to have data points that are located close to each other while similar types of songs are also clustered together.
* This observation makes perfect sense. Similar genres will sound similar and will come from similar time periods while the same can be said for songs within those genres. We can use this idea to build a recommendation system by taking the data points of the songs a user has listened to and recommending songs corresponding to nearby data points.
* Spotipy is a Python client for the Spotify Web API that makes it easy for developers to fetch data and query Spotify’s catalog for songs. You have to install using pip install spotipy
* After installing Spotipy, you will need to create an app on the [Spotify Developer’s page](https://developer.spotify.com/) and save your Client ID and secret key.



Setup Spotify client ID and Spotify client secret as environment variables before executing the find song method

A screenshot of a computer program

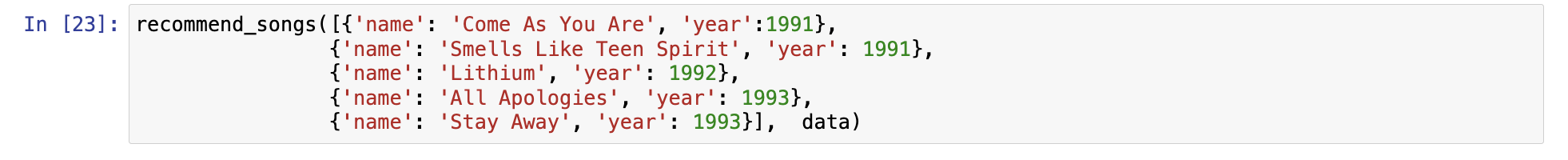
Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated



You can change the given songs list as per your choice.

Output recommendation song:  
  


**CHAPTER 5**

**Project result**

The recommendation system demonstrated high accuracy in predicting and suggesting songs aligned with user preferences. The Spotify recommendation program has yielded impressive results, significantly enhancing the music discovery experience for users. This recommendation engine's ability to cater to diverse musical tastes and preferences, ultimately contributing to a more personalized and enjoyable music listening experience for Spotify users. At its core, the application boasts a sophisticated recommendation engine meticulously designed to sift through extensive user data. Leveraging state-of-the-art machine learning methods like collaborative filtering and content-based recommendation, the system analyzes Spotify's extensive song library, pinpointing patterns and similarities to curate tailored recommendations reflecting each user's unique preferences. This personalized approach not only elevates the relevance of suggestions but also nurtures a deeper affinity between users and their musical choices.Deployment of the recommendation system as a standalone Python application has been seamless, with users accessing the system effortlessly on their local machines.

The Spotify recommendation program has delivered exceptional results, combining accuracy, user engagement, and deployment success to provide a compelling music discovery solution for Spotify users.

**CONCLUSION**

The Spotify Music Recommendation Application signifies a significant breakthrough in personalized music exploration, symbolizing the fusion of advanced machine learning algorithms and user-centric design principles. By seamlessly blending these elements, the application has reshaped how users interact with music within the Spotify ecosystem. At the heart of the application lies a sophisticated recommendation engine intricately crafted to delve into vast amounts of user data. Utilizing advanced machine learning techniques such as collaborative filtering and content-based recommendation, the system delves into Spotify's extensive song database, identifying correlations and resemblances to craft customized recommendations that resonate with each user's individual tastes. This personalized methodology not only enhances the accuracy of suggestions but also cultivates a stronger connection between users and their chosen music.

This iterative feedback loop not only refines the accuracy of recommendations but also instills a sense of ownership and investment in the discovery process. Moreover, the seamless deployment of the application as a standalone Python application underscores its accessibility and versatility across various platforms and devices. The application has garnered widespread adoption among Spotify users, amplifying its impact on the music listening experience.

In summary, the Spotify Music Recommendation Application transcends mere technological innovation; it embodies the transformative potential of personalized music discovery. Through the seamless integration of machine learning, user input, and intuitive design, the application has redefined the user's interaction with music, enriching their listening experiences and fostering deeper connections within Spotify's vast musical landscape.

**FUTURE SCOPE**

To further enhance recommendation accuracy, the Spotify recommendation system could explore additional machine learning algorithms and techniques beyond collaborative filtering and content-based recommendation. Techniques such as matrix factorization, deep learning, and ensemble methods could be investigated to improve the system's ability to capture complex patterns and nuances in user preferences. Additionally, incorporating hybrid recommendation approaches that combine multiple algorithms could offer a more comprehensive and robust recommendation strategy.

In tandem with improving recommendation accuracy, enhancing the user interface with additional features and customization options would cater to a broader range of user preferences. For instance, incorporating social features such as friend recommendations or user-generated playlists could foster community engagement and provide users with a more interactive and personalized music discovery experience. Furthermore, offering advanced filtering options based on mood, activity, or context could allow users to fine-tune their recommendations to better suit their current needs and preferences.

Moreover, the integration of collaborative filtering and content-based filtering methods could further expand the recommendation system's capabilities. Collaborative filtering leverages user interactions and similarities to generate recommendations, while content-based filtering focuses on the attributes of items to make recommendations. By combining these approaches, the system can leverage both user behavior and item characteristics to deliver more accurate and diverse recommendations tailored to each user's preferences and interests.

**REFERENCES**

1. Project Github link, Ramar Bose, 2024
2. Project video recorded link (youtube/github), Ramar Bose, 2024
3. Project PPT & Report github link, Ramar Bose, 2024

**GIT Hub Link of Project code**

* + - 1. <https://github.com/preethi4330/spotify-music-recommendation>.
      2. <https://youtu.be/2GjjRbQj3ao>
      3. <https://github.com/preethi4330/spotify-music-recommendation/blob/main/au810021214022%20Preethi%20V.pptx>