

**Tech Saksham**

**Capstone Project Report**

**“Spotify Music Recommendation System”**

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

|  |  |
| --- | --- |
| **NM ID** | **NAME** |
| **au810021214019** | **LOKESH S** |

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

**ABSTRACT**

The thoughtfully constructed Spotify Music Recommendation System, which is driven by Python, offers customers personalized song recommendations selected from Spotify's vast music library. The system employs sophisticated machine learning methods, including K-Means clustering, to assess and categorize songs based on their energy, danceability, loudness, and pace. By using a clustering approach, the system can more accurately identify songs that share similar musical features, improving the accuracy of its user preference-based suggestions.

Users are asked to select their favourite music at the start of the application, and fuzzy string-matching techniques are used by the system to find the closest match in its database. Additionally, customers are able to indicate which genres they favour, which enables a more sophisticated recommendation system. Using this information, the system creates a dynamic, user-specific list of suggested songs that have been carefully chosen to suit the user's taste in music.

A strong foundation of feature extraction, data preprocessing, and model training techniques forms the basis of the recommendation process. By dividing the dataset into discrete groups according to common musical characteristics, the K-Means clustering technique expedites the retrieval of pertinent recommendations. Every music is intelligently matched with the appropriate cluster, guaranteeing a smooth and successful recommendation process.

The Spotify Music Recommendation System strives to improve users' music discovery experiences by seamlessly integrating machine learning, data analysis, and user involvement. Thanks to its user-friendly design and tailored recommendations, music lovers can discover new songs that closely match their individual tastes.

Furthermore, the technology aims to unearth undiscovered musical treasures in addition to recommending well-known songs, establishing a stronger bond between consumers and their preferred songs. The technology enhances the overall music listening experience by making it easier to find interesting and fascinating tunes, which motivates users to go on a musical research and discovery journey.

**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 purpose of Spotify's music recommendation system is to provide users with customized song recommendations based on their musical preferences. By leveraging machine learning techniques and the extensive Spotify dataset, the system analyses song properties like danceability, energy, and tempo to group songs into clusters based on shared features. By entering their favourite song and, optionally, specifying a preferred genre, users interact with the system. The system then produces lists of suggested songs that are carefully chosen and closely align with the user's tastes. Important goals of the system include accurate suggestion creation, efficient data analysis and preprocessing, user-friendly interface, and scalability to handle large amounts of data. 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 suggested technique creates customized music recommendations by using the Spotify tracks dataset that was obtained from Kaggle. Data points are arranged according to qualities like danceability, energy, and tempo using K-Means clustering, which makes it easier to find music with related features. To improve suggestion accuracy, users can indicate preferred genres and offer their favorite songs. Making use of the clustering approach, the system suggests songs from the user's preferred cluster, guaranteeing that it is in line with their taste in music. The recommender system is implemented as a regular Python application, which guarantees user accessibility. Spotify users' overall music discovery experience is improved by constant system improvement, which is supported by user input and performance evaluations.

**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 broad and varied, encompassing several essential components. First, the Spotify dataset with music properties and information must be obtained and prepared. The next crucial step is feature engineering, which focuses on choosing and extracting pertinent qualities including danceability, energy, and tempo. These features serve as the basis for recommendation and grouping. In order to organize songs into clusters according to their attributes, the project also involves implementing machine learning models, namely clustering methods like K-Means. Next, using the clustering results, a recommendation engine is created to provide users with tailored song recommendations based on their input, including their favorite songs and desired genres. Moreover, user involvement is essential, necessitating the development of a user-friendly interface to enable smooth interaction 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.

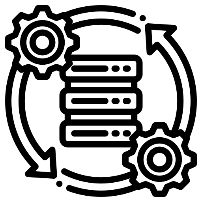
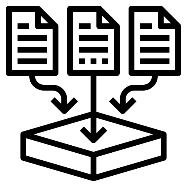
**2.3 Software Requirements**:

* **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**

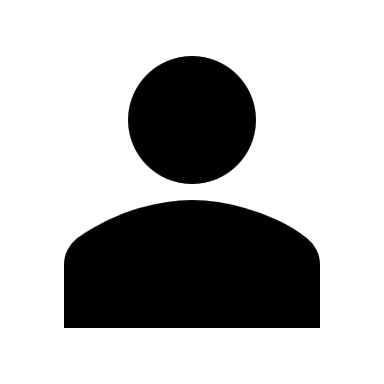
**Flow diagram:**

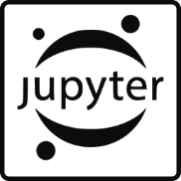
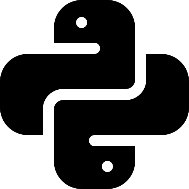
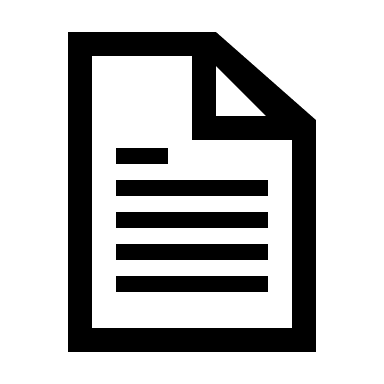
**OUTPUT**

**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**

Jupyter Notebook plays a crucial role in developing Spotify's recommendation system. It facilitates data exploration, model development, and evaluation through interactive coding. With Jupyter, Spotify engineers can experiment with various algorithms, analyze data insights, and refine the recommendation engine to deliver personalized music suggestions to users, enhancing their listening experience.

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

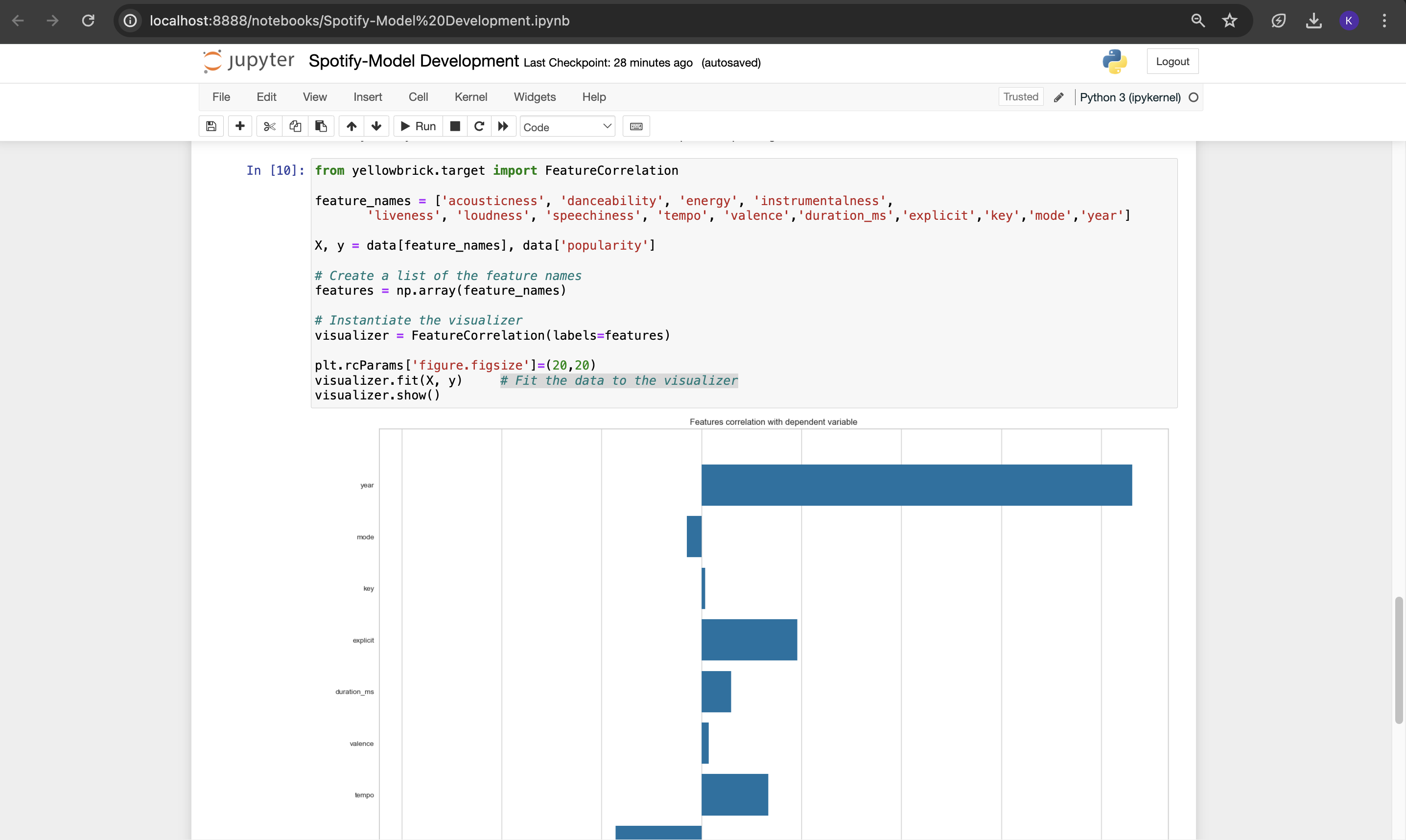
Description automatically generated

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.

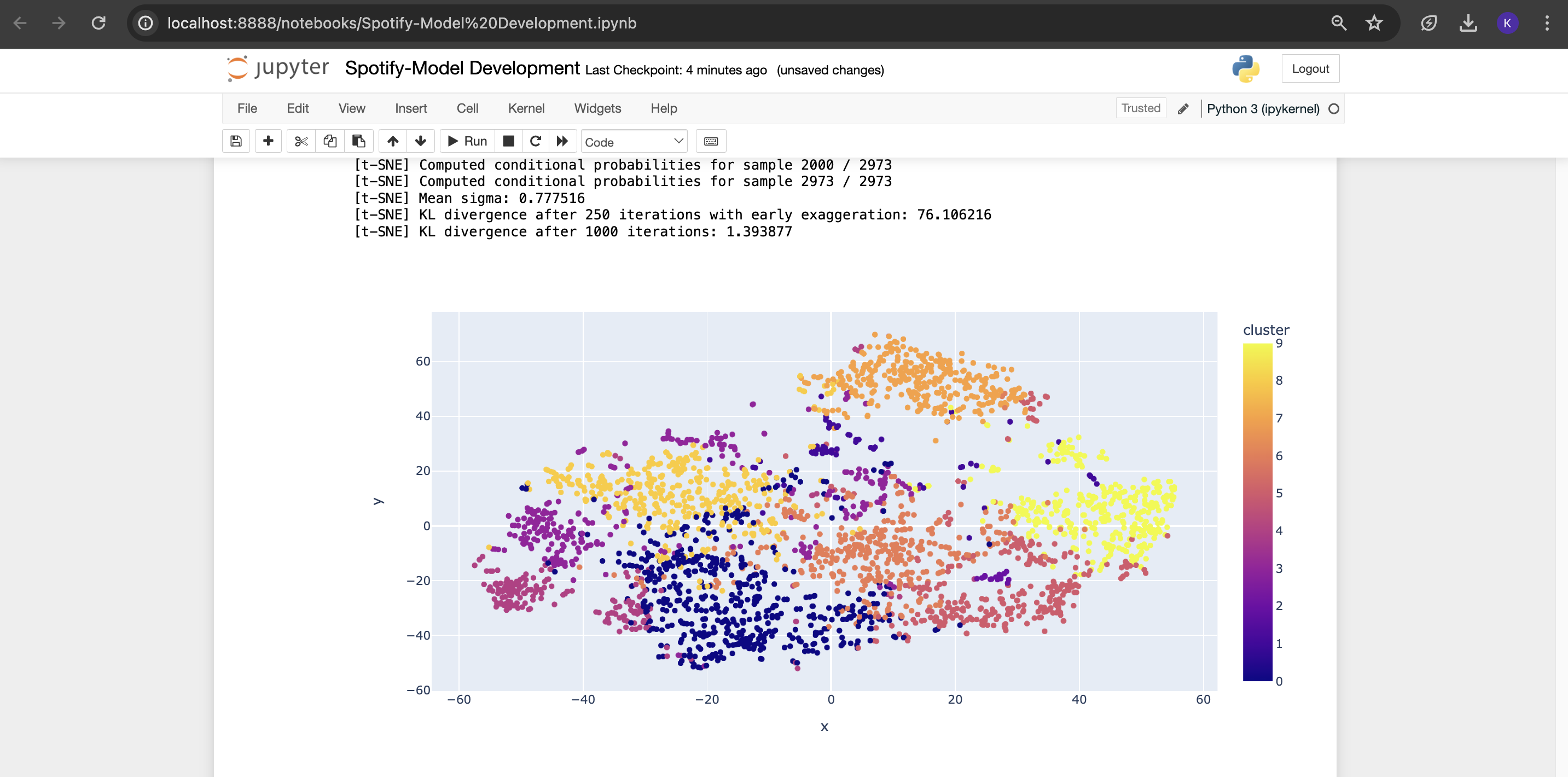
A screenshot of a computer

Description automatically generated

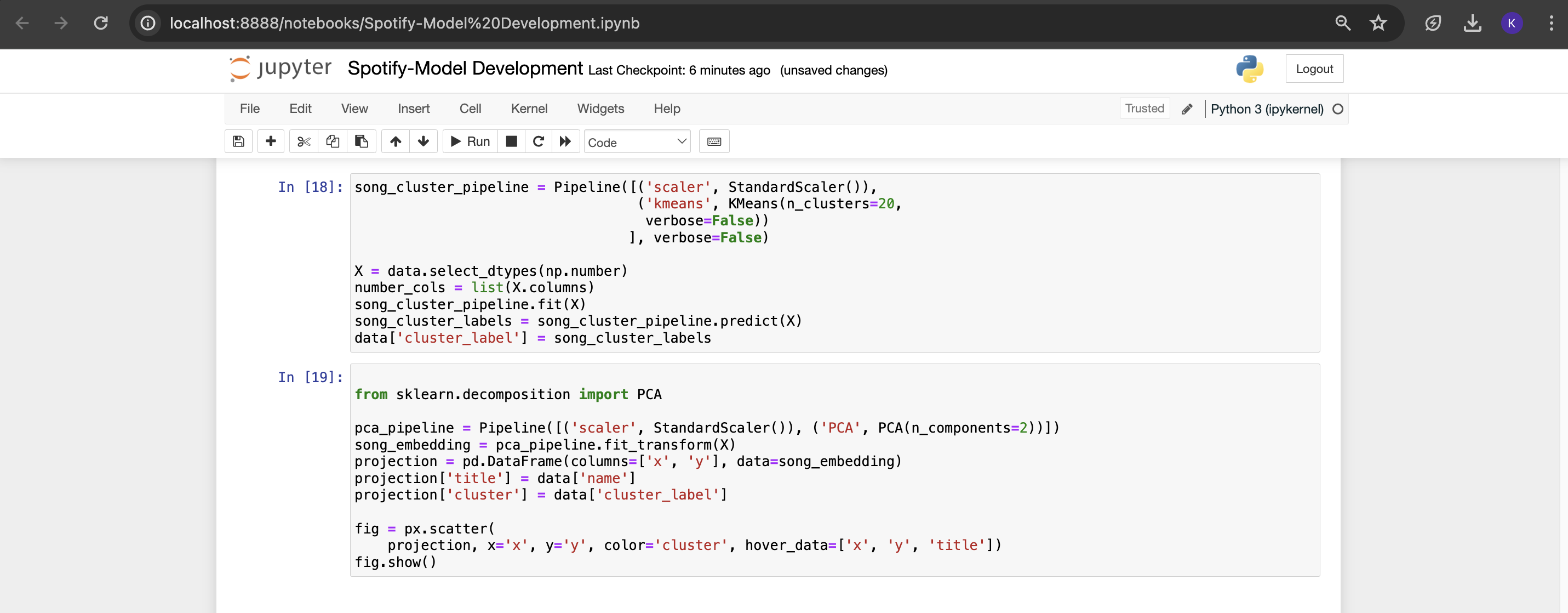
**K-means clustering algorithm** is used to divide the genres in this dataset into ten clusters based on the numerical audio features of each genre.

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.
* Spotify 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 Spotify.
* After installing Spotify, 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

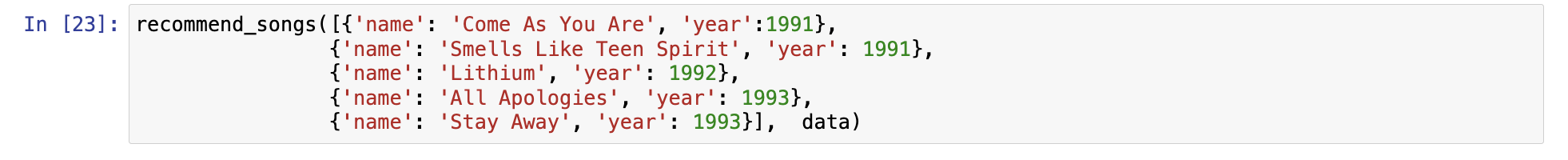
Description automatically generated

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 has excellent accuracy in foretelling and recommending songs that matched user tastes. The Spotify suggestion system has produced remarkable outcomes, greatly improving users' experience finding music. The recommendation engine's versatility allows it to accommodate a wide range of musical interests and preferences, which in turn makes Spotify customers' music-listening experiences more enjoyable and customized. The application's main feature is an advanced recommendation engine that has been painstakingly created to sort through large amounts of user data. The algorithm scans Spotify's vast song library, identifying patterns and similarities to provide personalized recommendations that represent each user's individual interests. It does this by utilizing cutting-edge machine learning techniques including collaborative filtering and content-based recommendation. 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, which combines cutting-edge machine learning algorithms with user-centric design concepts, represents a major advancement in personalized music exploration. The application has changed how consumers interact with music within the Spotify ecosystem by skilfully fusing various components together. A sophisticated recommendation engine that has been painstakingly designed to sift through enormous volumes of user data is at the centre of the program. With the use of sophisticated machine learning methods like content-based recommendation and collaborative filtering, the system explores Spotify's vast song catalog, finding similarities and correlations to create personalized recommendations that speak to the unique preferences of each user. This tailored approach improves recommendation accuracy while strengthening the bond between users and the music they select.

In addition to improving recommendation accuracy, this recurrent feedback loop fosters a sense of ownership and engagement with the discovery process. Furthermore, the program's easy installation as a stand-alone Python application highlights its adaptability and accessibility across a range of platforms and gadgets. Spotify users have embraced the app widely, which has increased its influence on the music-listening experience.

In conclusion, the Spotify Music Recommendation Application represents the revolutionary potential of individualized music discovery, going beyond simple technological innovation. By means of the smooth amalgamation of machine learning, user input, and user-friendly design, the application has revolutionized the way users engage with music. It has enhanced their listening experiences and established more profound connections within Spotify's extensive catalogue of songs.

**FUTURE SCOPE**

Besides collaborative filtering and content-based suggestion, the Spotify recommendation system may investigate other machine learning methods and methodologies to further improve recommendation accuracy. Investigating methods like matrix factorization, deep learning, and ensemble approaches could help the system better grasp intricate patterns and subtleties in human preferences. Furthermore, a more thorough and reliable recommendation system might be provided by combining hybrid recommendation systems, which blend many algorithms.

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.

The recommendation system's capabilities may also be increased by using content-based and collaborative filtering techniques. While content-based filtering makes recommendations based on item attributes, collaborative filtering makes advantage of user interactions and commonalities. Combining these methods allows the system to take use of item attributes as well as user behavior to provide recommendations that are more varied and accurate, based on the interests and preferences of individual users.

**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

**MY REFERENCES**

* + - 1. <https://github.com/lokesh0346/Spotify-recommendation-system>
      2. <https://youtu.be/FS3ExtmpLpY?si=KORQF18VLGd9djWI>
      3. [au810021214019\_Lokesh\_S.pptx](https://1drv.ms/p/s!AjJqLdVBMvdUgQW3YDZIdS_dYpZC?e=IMLRdq)