MUSIC RECOMMENDATION SYSTEM

A MAJOR PROJECT REPORT

Submitted By

Baivab Mukherjee [Roll No. 31001221024]

Under the Supervision of

Mr. Pinaki Mukherjee

(Faculty, Dept. of BCA)

In fulfillment for the award of the degree of

BACHELOR OF COMPUTER APPLICATION

MEGHNAD SAHA INSTITUTE OF TECHNOLOGY

Techno Complex, Madurdaha, Beside NRI Complex, Post-Uchhepota, Kolkata 700150

Affiliated by

MAULANA ABUL KALAM AZAD UNIVERSITY OF TECHNOLOGY

BF 142, BF BLOCK, SALTLAKE CITY, KOLKATA, WEST BENGAL - 700064

Project Certificate

This is to certify that the project entitled “**Music Recommendation System**” has been prepared according to the regulations of the degree of Bachelors in Computer Application(BCA) under the university of “Maulana Abul Kalam Azad University ofTechnology”. The project being submitted by -

____

(STUDENT SIGNATURE)

Student of Bachelor in Computer Application(BCA), 3rd year 6th semester of ***Meghnad Saha Institute of Technology*** (affiliated to Maulana Abul Kalam Azad University of Technology) have fulfilled the requirement for submission of this.

The whole procedure has been carried out under my supervision and guidance.

I have gone through this project and have seen that it is fulfilling the requirements ofMajor Project under MAKAUT, WB

TEAM MEMBERS	ROLL NUMBERS	SIGNATURE
BAIVAB MUKHERJEE	31001221024	
SUBHROJYOTI PYNE	31001220031	
SAYAN GHOSH	31001221041	
SUJOY BERA	31001221034	
TANMOY SIKDAR	31001221030	
SUMAN ROY	31001221018	

Dated: / /2024

(INTERNAL PROJECT GUIDE) **(EXTERNAL PROJECT GUIDE)**

(HEAD OF THE DEPARTMENT) **(EXAMINER)**

Acknowledgement

I would take the opportunity to thank prof. Aparna Datta , Head of Department Computer Application Department Meghnad Saha Institute of Technology for providing me with all the necessary facilities to make my project.

I am also thankful to ***Prof. Pinaki Mukherjee***, my Project Supervisor for constantly supporting and guiding me and also for giving me invaluable insights. Hisguidance and his words of encouragement motivated me to achieve my goal and impetus to excel.

Signature:

Table of Contents

Abstract

In recent years, the availability and popularity of music streaming services have led to an explosion of music consumption. However, with an overwhelming number of options available, users may find it difficult to discover new and relevant music. To address this issue, music recommendation systems have been developed to provide personalized recommendations to users. In this report, we present the design and implementation of a music recommendation system that utilizes collaborative filtering and content-based approaches. We evaluate the system's performance using a dataset of user listening histories and demonstrate the effectiveness of our approach through quantitative and qualitative analysis. Our findings suggest that music recommendation systems can enhance users' listening experiences and provide a more personalized music discovery process.

Overall, music recommendation systems have become an increasingly important tool for music listeners and the music industry as a whole. With the help of AI and machine learning, these systems can provide users with personalized recommendations, enhance their music listening experiences, and help artists reach new audiences.

Introduction

Music recommendation systems are a type of artificial intelligence that aims to provide users with personalized music recommendations based on their listening history, preferences, and behavior. These systems analyze a user's data, such as their past listening history, and then utilize machine learning algorithms to provide personalized recommendations.

There are various approaches to music recommendation systems, including collaborative filtering, content-based filtering, and hybrid systems that combine both methods. Collaborative filtering relies on similarities between users and their listening behaviors, while content-based filtering analyzes the characteristics of the music itself, such as genre, tempo, and mood. Hybrid systems, as the name suggests, combine both methods to provide more accurate and diverse recommendations.

One of the key benefits of music recommendation systems is that they can help users discover new and relevant music that they might not have otherwise found. This can lead to a more enjoyable listening experience and help users find new artists and genres they might not have otherwise explored..

.

Proposed Project Work

Music Recommendation system is designed and built in such a way that it uses machine learning and related tools effectively to recommend top 10 songs bases on user’s input data. The recommendation is based on numerical features of the songs. The system finds the cosine distance to find the songs with highest similarity.

- **Tools used to design the project are explained below**.

Python

Python is a high-level, interpreted, interactive and object-oriented scripting language. Python is designed to be highly readable. It uses English words frequently whereas other languages use punctuation, and it has fewer syntactic constructions than other languages. Python is dynamically typed and garbage-collected. It supports multiple programming paradigms, including structured (particularly, procedural), object-oriented, and functional programming. Python is often described as a "batteries included" language due to its comprehensive standard library.

The Python 2 language was officially discontinued in 2020 (first planned for 2015), and "Python 2.7.18 is the last Python 2.7 release and therefore the last Python 2 release." No more security patches or other improvements will be released for it. With Python 2's end-of-life, only Python 3.5.x and later are supported.

Jupyter Notebook

Jupyter Notebook is an open-source web application that enables users to create and share documents containing live code, equations, visualizations, and narrative text.. This interactive environment allows users to execute code in a step-by-step manner, which is particularly useful for debugging and understanding the program flow. Moreover, Jupyter Notebook integrates rich text support through Markdown and LaTeX, allowing users to include formatted text, equations, and descriptions alongside their code.

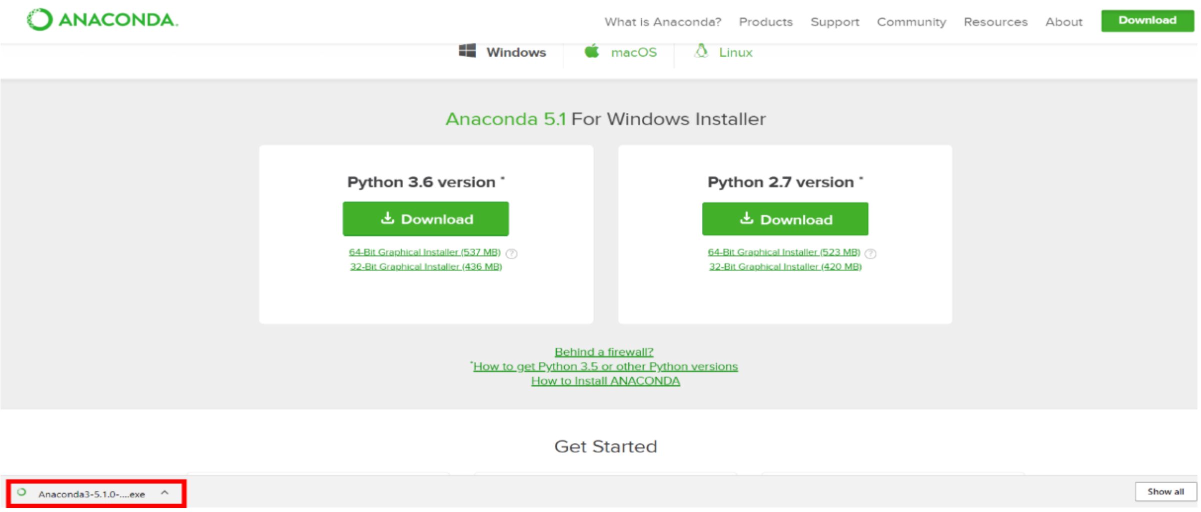
One of the standout features of Jupyter Notebook is its capability for creating and displaying visualizations. It seamlessly integrates with plotting libraries like Matplotlib, Seaborn, and Plotly, enabling users to produce and interact with data visualizations directly within the notebook. Additionally, Jupyter Notebook supports data integration from various sources, including CSV files, databases, and APIs, further enhancing its utility for data-driven projects.

Jupyter Notebook is widely used in data science, academic research, machine learning, and education due to its interactive and user-friendly interface. Data scientists and researchers utilize it for exploring datasets, performing statistical analyses, and documenting their findings in a transparent and reproducible manner. In educational settings, it serves as an effective tool for teaching programming and data analysis concepts. The ability to share notebooks easily via platforms like GitHub and JupyterHub promotes collaboration and knowledge sharing, making Jupyter Notebook awesome.

Anaconda Setup

1. Go to the Anaconda Website and choose a Python 3.x graphical installer (A) or a Python 2.x graphical installer (B).

   If you aren't sure which Python version you want to install, choose Python 3. Do not choose both.



Anaconda 5.1 For Windows Installer

Python 3.6 version *

Download

64-Bit Graphical Installer (537 MB)

32-Bit Graphical Installer (436 MB)

Python 2.7 version *

Download

64-Bit Graphical Installer (523 MB)

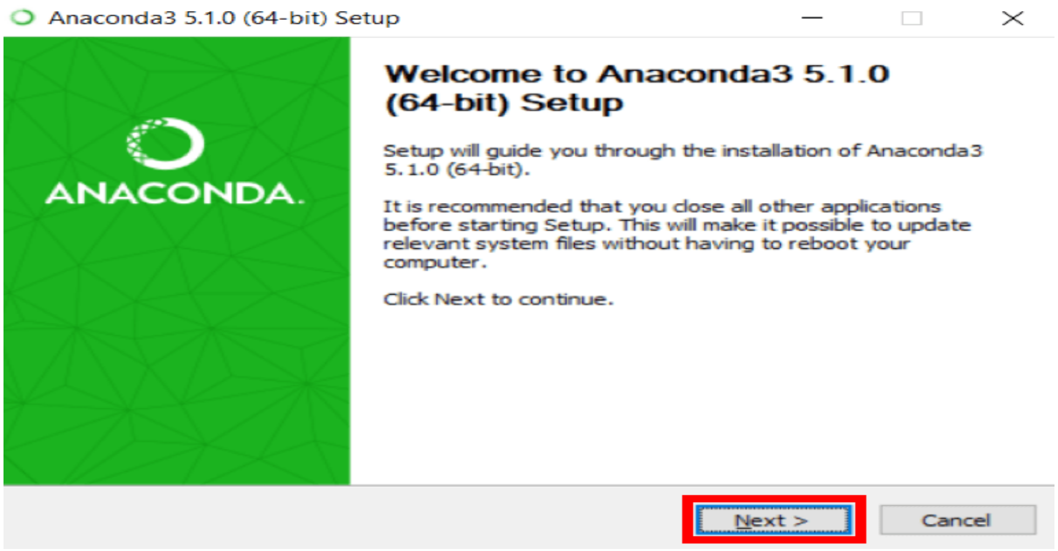
32-Bit Graphical Installer (420 MB)

Behind a firewall?

*How to get Python 3.5 or other Python versions

How to Install ANACONDA

2. Locate your download and double click it. When the screen below appears, click on Next.



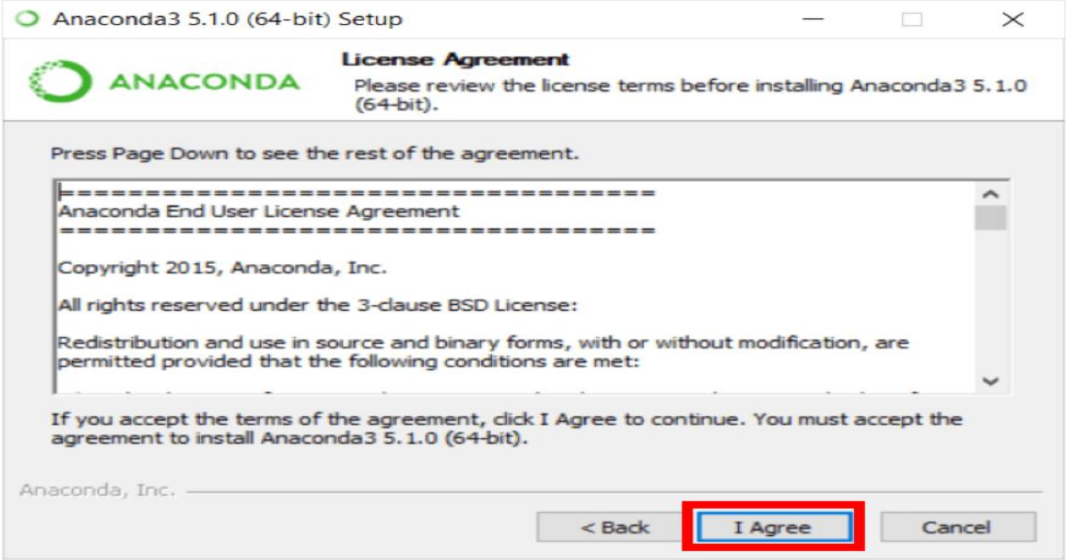
Welcome to Anaconda3 5.1.0 (64-bit) Setup

Setup will guide you through the installation of Anaconda3 5.1.0 (64-bit).

It is recommended that you close all other applications before starting Setup. This will make it possible to update relevant system files without having to reboot your computer.

Click Next to continue.

3. . Read the license agreement and click on I Agree.



Anaconda3 5.1.0 (64-bit) Setup

License Agreement

Please review the license terms before installing Anaconda3 5.1.0 (64-bit).

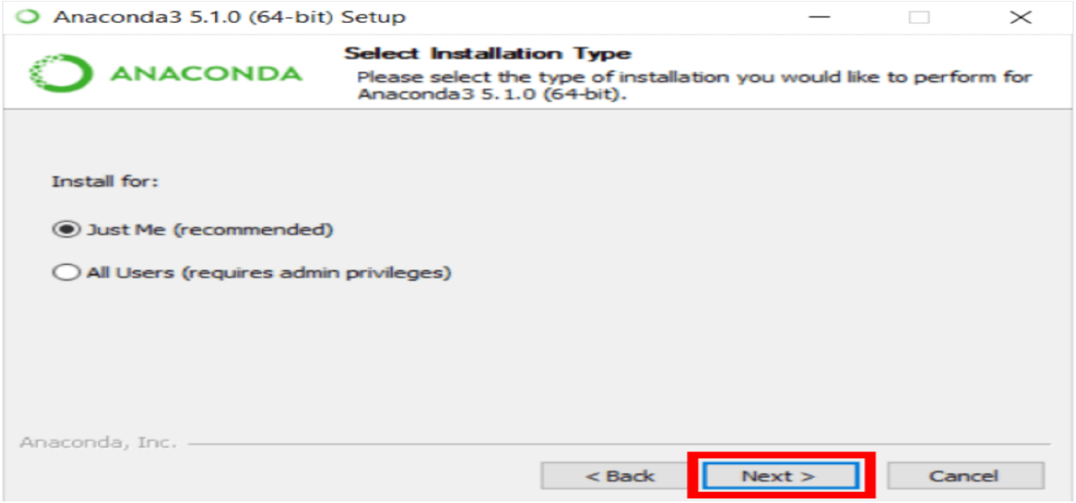
Press Page Down to see the rest of the agreement.

```
===================================
Anaconda End User License Agreement
===================================

Copyright 2015, Anaconda, Inc.

All rights reserved under the 3-clause BSD License:

Redistribution and use in source and binary forms, with or without modification, are
permitted provided that the following conditions are met:
```

If you accept the terms of the agreement, click I Agree to continue. You must accept the agreement to install Anaconda3 5.1.0 (64-bit).

Anaconda, Inc.

4. Click on Next.



Anaconda3 5.1.0 (64-bit) Setup

Select Installation Type

Please select the type of installation you would like to perform for Anaconda3 5.1.0 (64-bit).

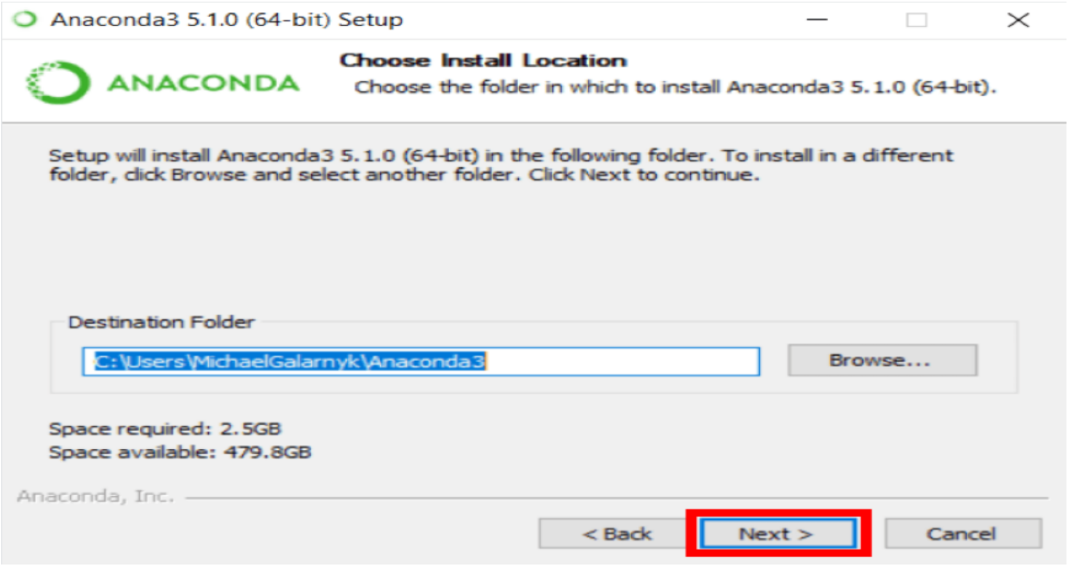
Install for:

● Just Me (recommended)

○ All Users (requires admin privileges)

Anaconda, Inc.

5. . Note your installation location and then click Next



Anaconda3 5.1.0 (64-bit) Setup

Choose Install Location

Choose the folder in which to install Anaconda3 5.1.0 (64-bit).

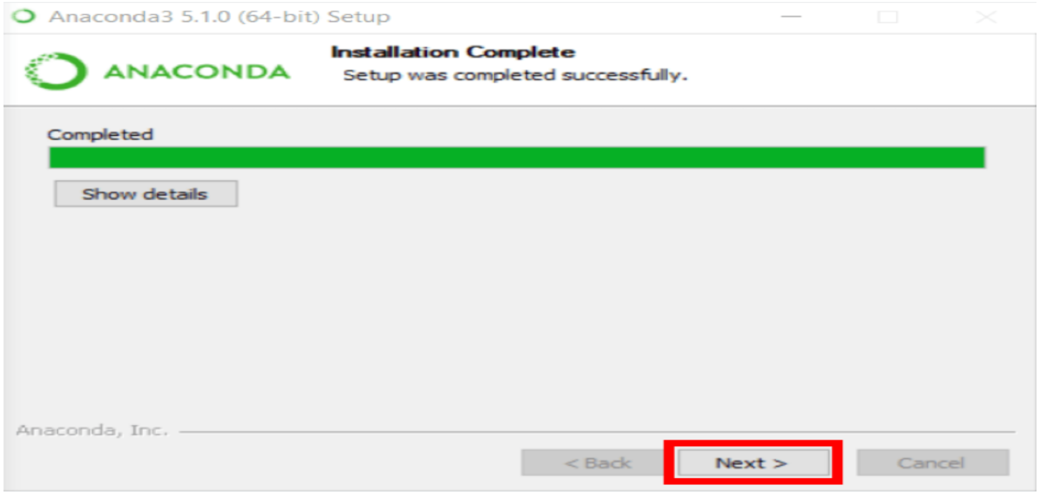
Setup will install Anaconda3 5.1.0 (64-bit) in the following folder. To install in a different folder, click Browse and select another folder. Click Next to continue.

Destination Folder

C:\Users\MichaelGalarnyk\Anaconda3

Space required: 2.5GB
Space available: 479.8GB

6. Click on Next.



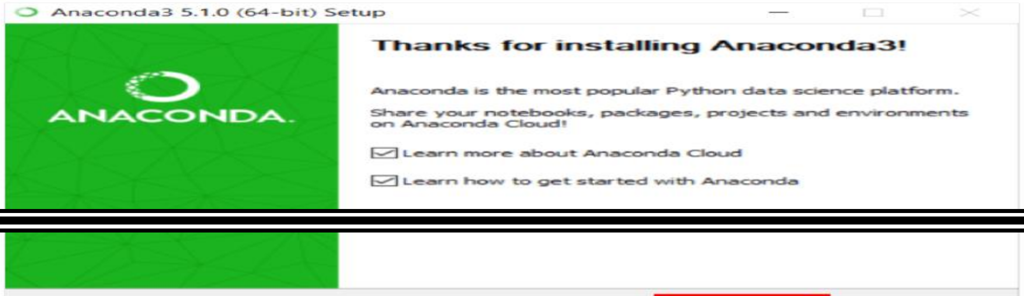
Anaconda3 5.1.0 (64-bit) Setup

Installation Complete

Setup was completed successfully.

Completed

7. Click on Finish.



Anaconda3 5.1.0 (64-bit) Setup

Thanks for installing Anaconda3!

Anaconda is the most popular Python data science platform.

Share your notebooks, packages, projects and environments on Anaconda Cloud!

☑ Learn more about Anaconda Cloud

☑ Learn how to get started with Anaconda

List of Some Essential libraries used while making the Project :

1) Numpy:

NumPy is a powerful Python library for numerical computing and data analysis. It provides a variety of data structures, including arrays and matrices, that enable fast and efficient computation of mathematical operations on large datasets. NumPy's multidimensional array data structure is particularly useful for handling complex data such as images, audio, and video. NumPy is widely used in various domains such as machine learning, scientific computing, and signal processing.

2) Seaborn:

Seaborn is a Python library built on top of Matplotlib that provides a high-level interface for creating informative and visually appealing statistical graphics. It provides several types of charts such as scatterplots, line charts, and heatmaps, that can be customized with different color palettes and themes. Seaborn is especially useful for visualizing complex datasets, and it provides several built-in datasets that can be used for experimentation and learning.

3) Matplotlib:

Matplotlib is a Python library used for creating static, animated, and interactive visualizations in Python. It provides a wide range of plotting functions, including scatterplots, line charts, bar charts, histograms, and more. Matplotlib is highly customizable, and it provides extensive support for controlling various aspects of the plot, such as axes, labels, titles, and legends. It is widely used in scientific computing, data analysis, and machine learning.

4) Streamlit

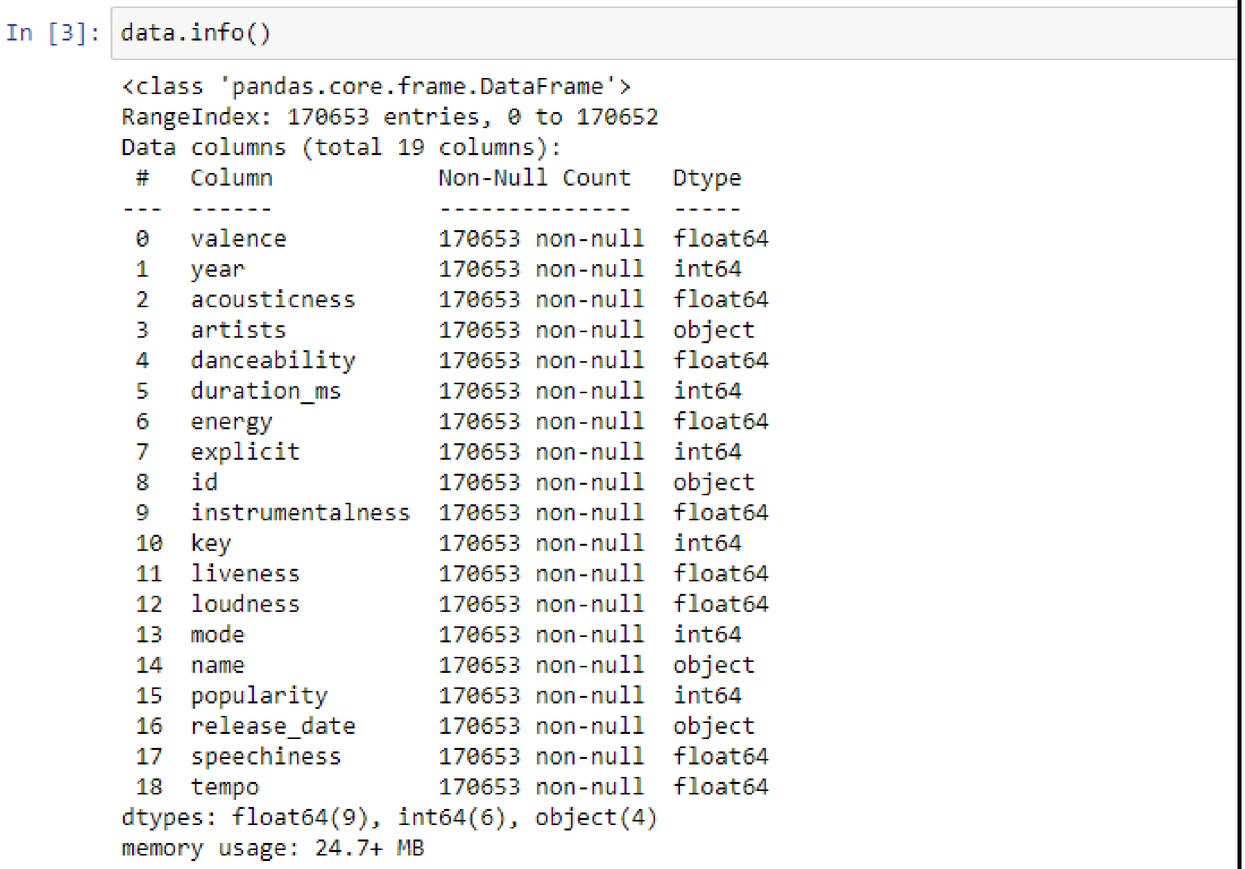
Streamlit is an open-source Python library designed to streamline the creation of interactive, web-based data applications, making it particularly popular among data scientists and machine learning engineers. It allows developers to build sophisticated data apps with just a few lines of Python code, eliminating the need for web development knowledge. With features like interactive widgets, real-time updates, and seamless integration with data science libraries such as Pandas and Plotly, Streamlit enables rapid prototyping and dynamic data visualization.

Project Work

Datasets used

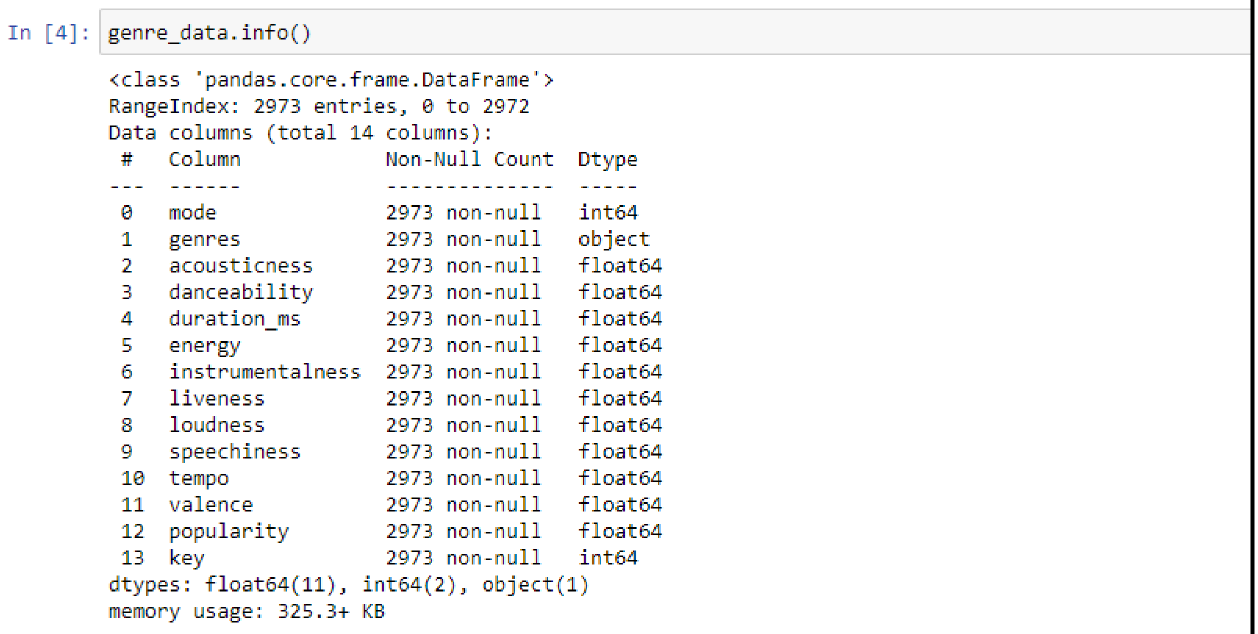
We have used four data sets while building the model ,those are

1) data.csv



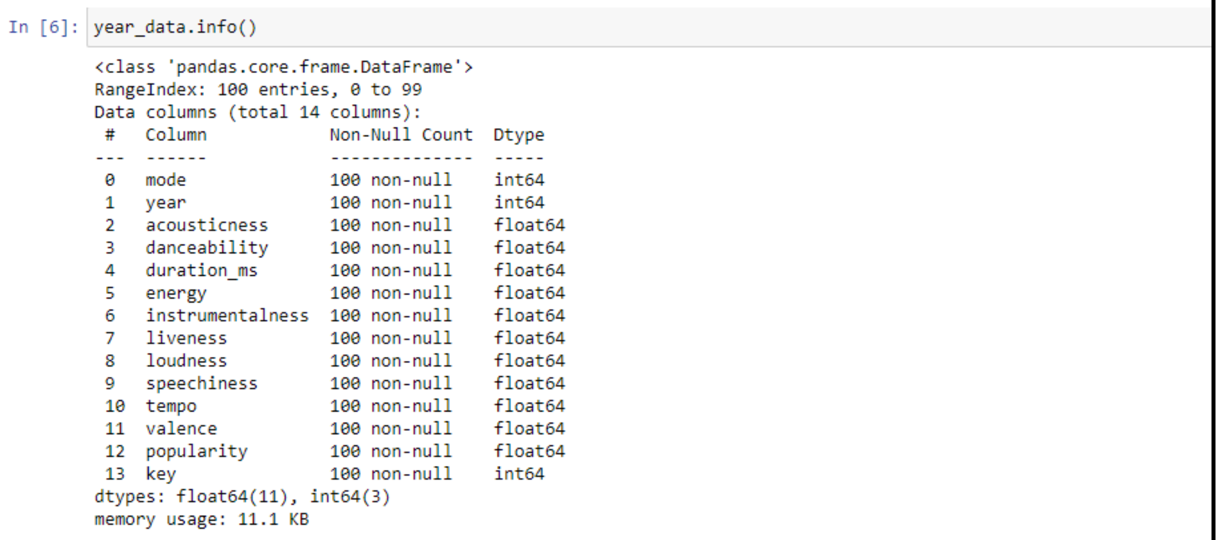
```
In [3]: data.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 170653 entries, 0 to 170652
Data columns (total 19 columns):
 #   Column            Non-Null Count   Dtype
---  ------            --------------   -----
 0   valence           170653 non-null  float64
 1   year              170653 non-null  int64
 2   acousticness      170653 non-null  float64
 3   artists           170653 non-null  object
 4   danceability      170653 non-null  float64
 5   duration_ms       170653 non-null  int64
 6   energy            170653 non-null  float64
 7   explicit          170653 non-null  int64
 8   id                170653 non-null  object
 9   instrumentalness  170653 non-null  float64
 10  key               170653 non-null  int64
 11  liveness          170653 non-null  float64
 12  loudness          170653 non-null  float64
 13  mode              170653 non-null  int64
 14  name              170653 non-null  object
 15  popularity        170653 non-null  int64
 16  release_date      170653 non-null  object
 17  speechiness       170653 non-null  float64
 18  tempo             170653 non-null  float64
dtypes: float64(9), int64(6), object(4)
memory usage: 24.7+ MB
```

2) data_by_genres.csv



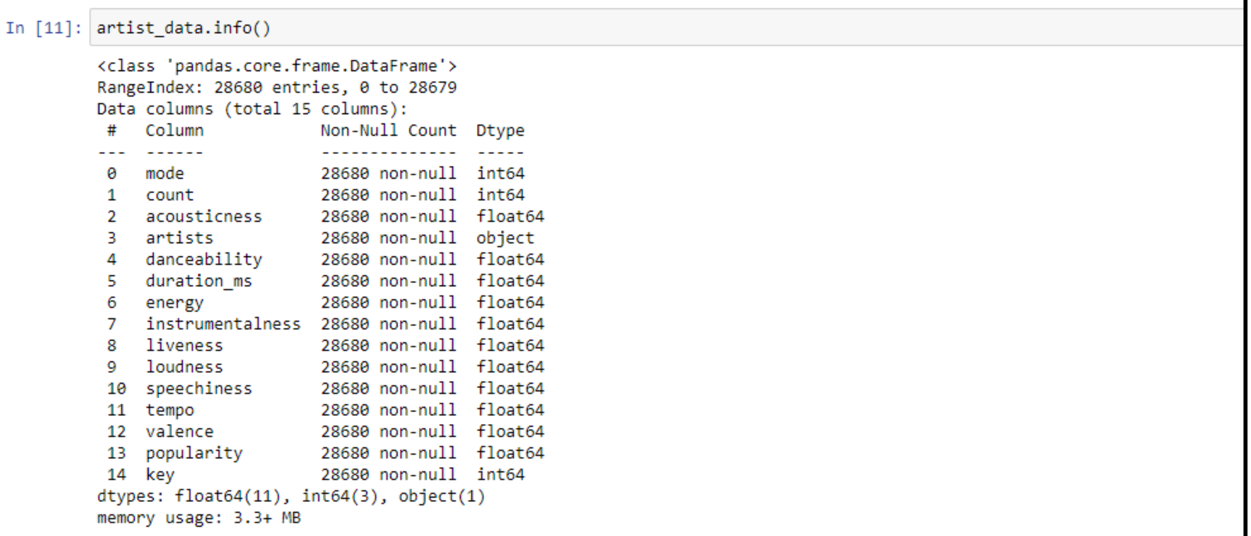
```
In [4]: genre_data.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2973 entries, 0 to 2972
Data columns (total 14 columns):
 #   Column            Non-Null Count  Dtype  
---  ------            --------------  -----  
 0   mode              2973 non-null   int64  
 1   genres            2973 non-null   object 
 2   acousticness      2973 non-null   float64
 3   danceability      2973 non-null   float64
 4   duration_ms       2973 non-null   float64
 5   energy            2973 non-null   float64
 6   instrumentalness  2973 non-null   float64
 7   liveness          2973 non-null   float64
 8   loudness          2973 non-null   float64
 9   speechiness       2973 non-null   float64
 10  tempo             2973 non-null   float64
 11  valence           2973 non-null   float64
 12  popularity        2973 non-null   float64
 13  key               2973 non-null   int64  
dtypes: float64(11), int64(2), object(1)
memory usage: 325.3+ KB
```

3) Data_by_year.csv



```
In [6]: year_data.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 100 entries, 0 to 99
Data columns (total 14 columns):
 #   Column            Non-Null Count  Dtype  
---  ------            --------------  -----  
 0   mode              100 non-null    int64  
 1   year              100 non-null    int64  
 2   acousticness      100 non-null    float64
 3   danceability      100 non-null    float64
 4   duration_ms       100 non-null    float64
 5   energy            100 non-null    float64
 6   instrumentalness  100 non-null    float64
 7   liveness          100 non-null    float64
 8   loudness          100 non-null    float64
 9   speechiness       100 non-null    float64
 10  tempo             100 non-null    float64
 11  valence           100 non-null    float64
 12  popularity        100 non-null    float64
 13  key               100 non-null    int64  
dtypes: float64(11), int64(3)
memory usage: 11.1 KB
```

4) Data_by_artist.csv



```
In [11]: artist_data.info()

         <class 'pandas.core.frame.DataFrame'>
         RangeIndex: 28680 entries, 0 to 28679
         Data columns (total 15 columns):
          #   Column            Non-Null Count  Dtype  
         ---  ------            --------------  -----  
          0   mode              28680 non-null  int64  
          1   count             28680 non-null  int64  
          2   acousticness      28680 non-null  float64
          3   artists           28680 non-null  object 
          4   danceability      28680 non-null  float64
          5   duration_ms       28680 non-null  float64
          6   energy            28680 non-null  float64
          7   instrumentalness  28680 non-null  float64
          8   liveness          28680 non-null  float64
          9   loudness          28680 non-null  float64
          10  speechiness       28680 non-null  float64
          11  tempo             28680 non-null  float64
          12  valence           28680 non-null  float64
          13  popularity        28680 non-null  float64
          14  key               28680 non-null  int64  
         dtypes: float64(11), int64(3), object(1)
         memory usage: 3.3+ MB
```

the datasets are downloaded by a trusted and genuine source which is **Kaggle.com.**

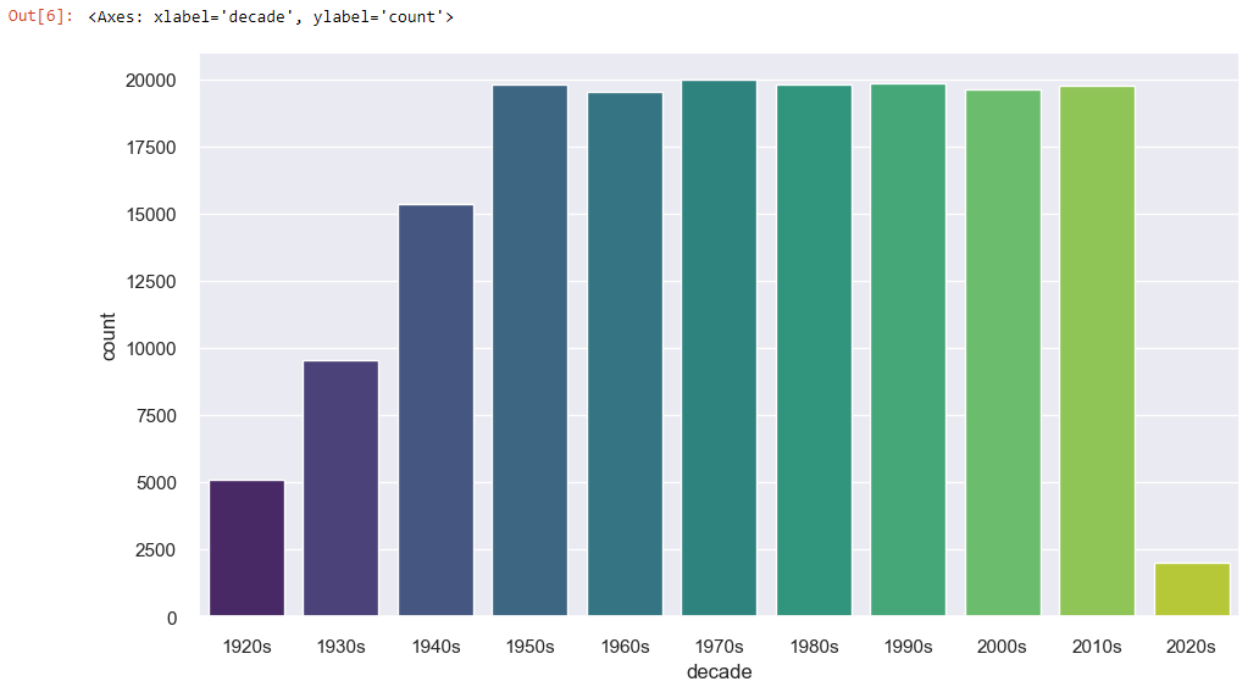
EDA:

EDA stands for Exploratory Data Analysis, and it is a critical step in the machine learning workflow. EDA involves analyzing and understanding the characteristics of the dataset you are working with. The goal of EDA is to gain insights into the data, discover patterns, and identify any potential problems with the data.

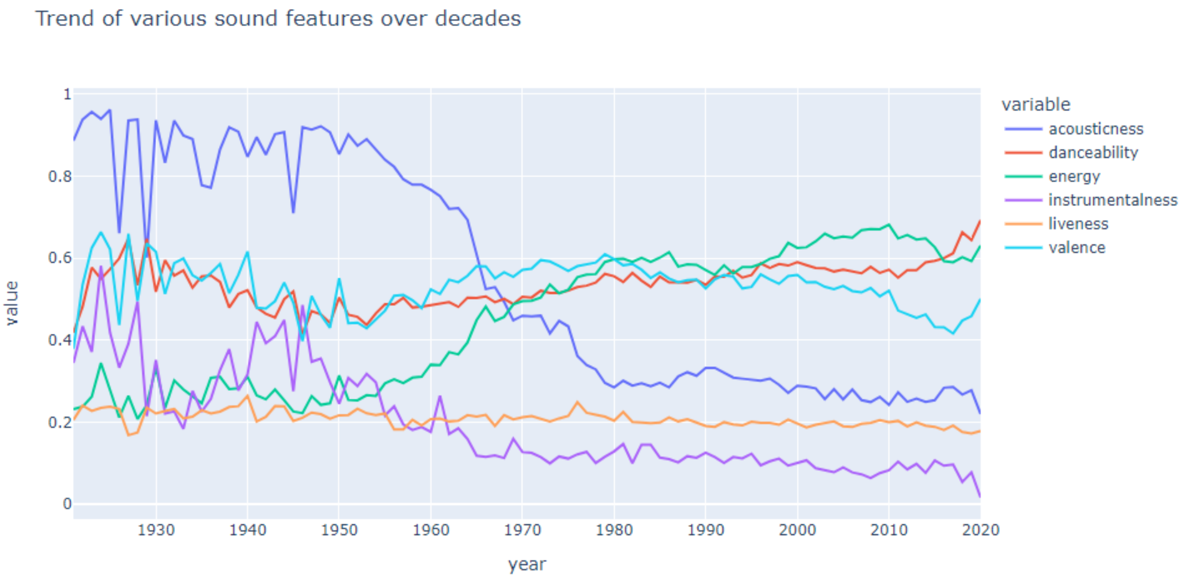
During EDA, you will typically examine the distribution of the data, check for missing values, outliers, and anomalies. You may also visualize the data using plots and charts to understand the relationships between the variables in the dataset. EDA can help you determine which features are important for predicting the target variable and which features may be less important or even irrelevant.

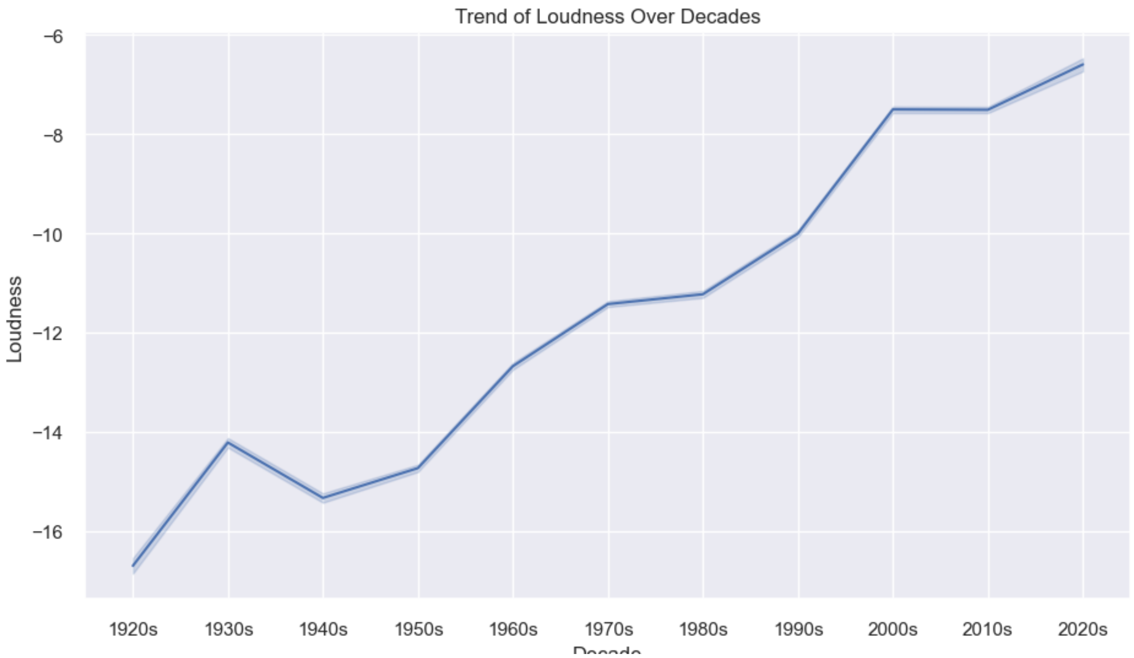
EDA is an iterative process, and it should be performed throughout the entire machine learning workflow. You may discover new insights about the data during the feature engineering or model selection phase, which can inform decisions about which features to include or which models to use.

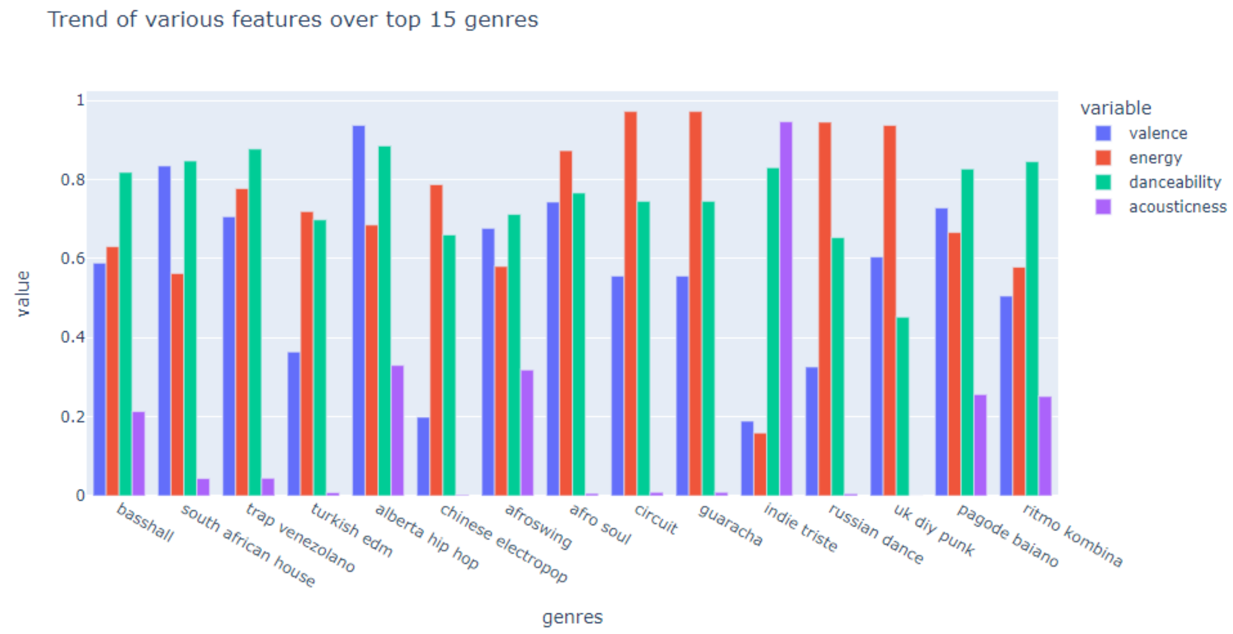
Some of the insights we got while performing the exploratory data analysis is listed below:-



```
Out[6]: <Axes: xlabel='decade', ylabel='count'>
```


Trend of various sound features over decades
value
1
0.8
0.6
0.4
0.2
0
1930
1940
1950
1960
1970
1980
1990
2000
2010
2020
year
variable
acousticness
danceability
energy
instrumentalness
liveness
valence


Trend of Loudness Over Decades
Loudness
−6
−8
−10
−12
−14
−16
1920s
1930s
1940s
1950s
1960s
1970s
1980s
1990s
2000s
2010s
2020s
Decade



Conclusions from EDA

Most of the songs range between 1950s-2010s.
Energy in songs have increased over the time.
Acousticness in songs have reduced greately over the decades.
We can clearly see that loudness has dominantly increased over the decades and is at it's peak in 2020.
In top 10 genres we can see that energy and dancebility are most noticable features.

Clustering:-

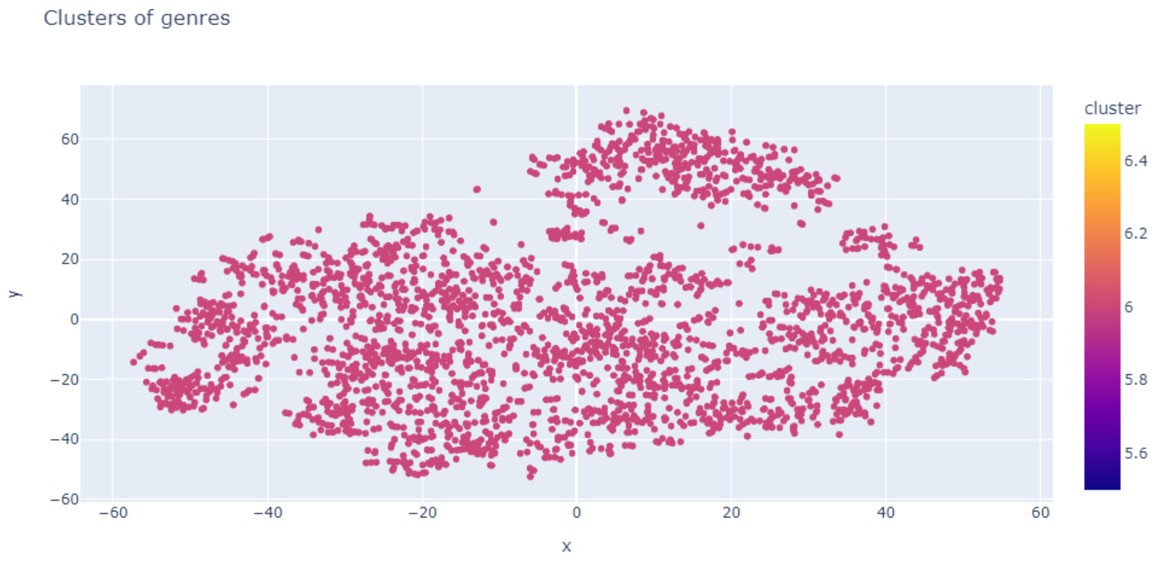
Clustering is a popular unsupervised machine learning technique that involves grouping similar data points together into clusters. The goal of clustering is to identify patterns in the data and group similar data points together based on their features. Clustering algorithms can be used for a wide range of applications, such as customer segmentation, anomaly detection, image analysis, and more.

One of the most commonly used clustering algorithms is k-means. K-means is an iterative algorithm that partitions the data into k clusters, where k is a user-defined parameter. The algorithm starts by randomly selecting k initial centroids, and then it assigns each data point to the nearest centroid. After all the data points have been assigned, the algorithm recalculates the centroid of each cluster and reassigns the data points based on their distances to the new centroids. This process continues until the centroids no longer move or a maximum number of iterations is reached.

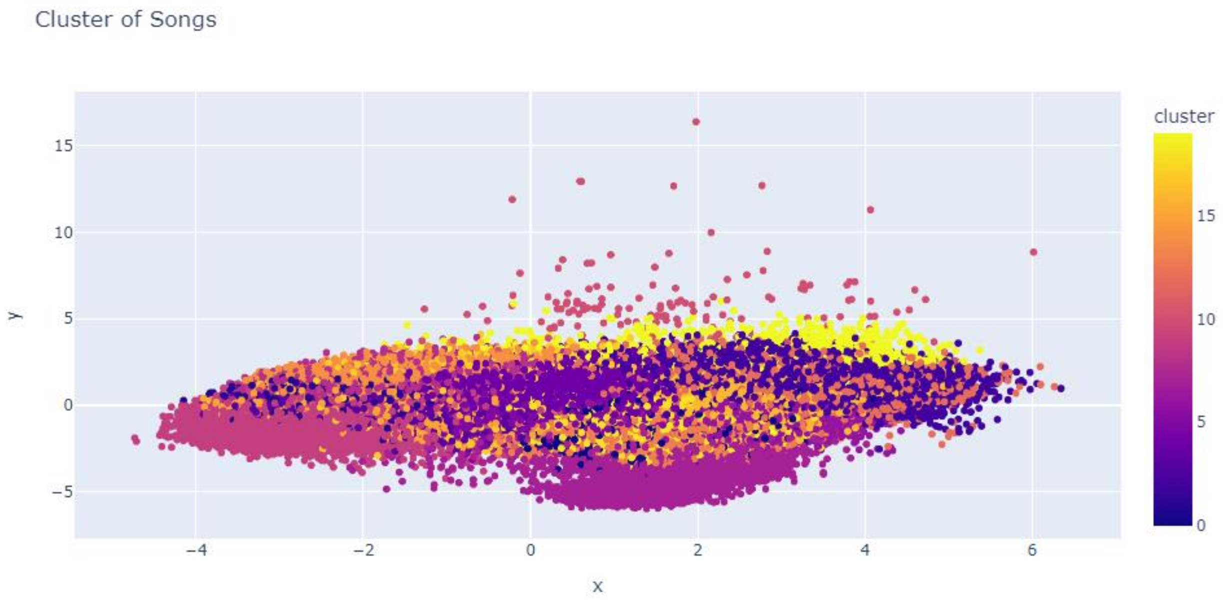
Overall, clustering is a powerful technique for finding patterns in data and grouping similar data points together. It can be used for a wide range of applications and is particularly useful when the data does not have labeled categories. Clustering algorithms like k-means can help identify clusters of data points that share similar characteristics and can provide valuable insights for understanding the underlying structure of the data.

We performed k-means clustering on the datasets to divide the dataset based on genre and songs.

Following are the results we got after performing clustering.

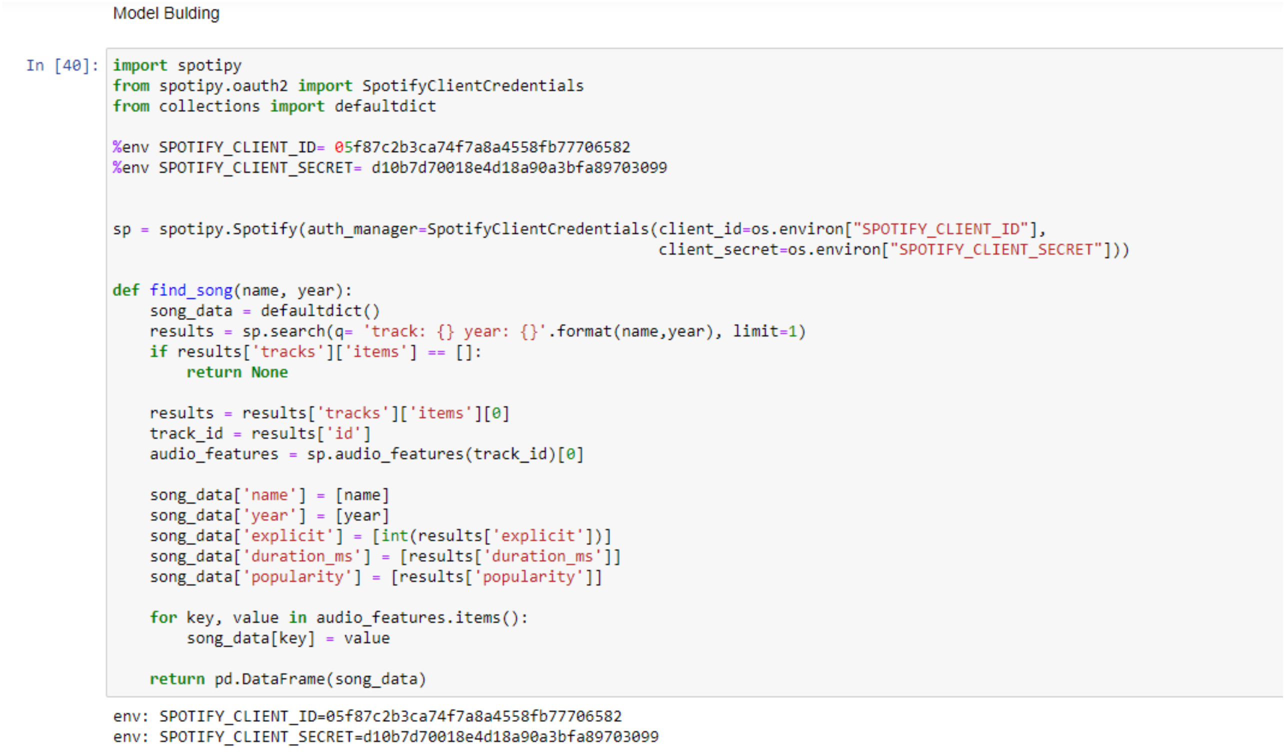


Clusters of genres



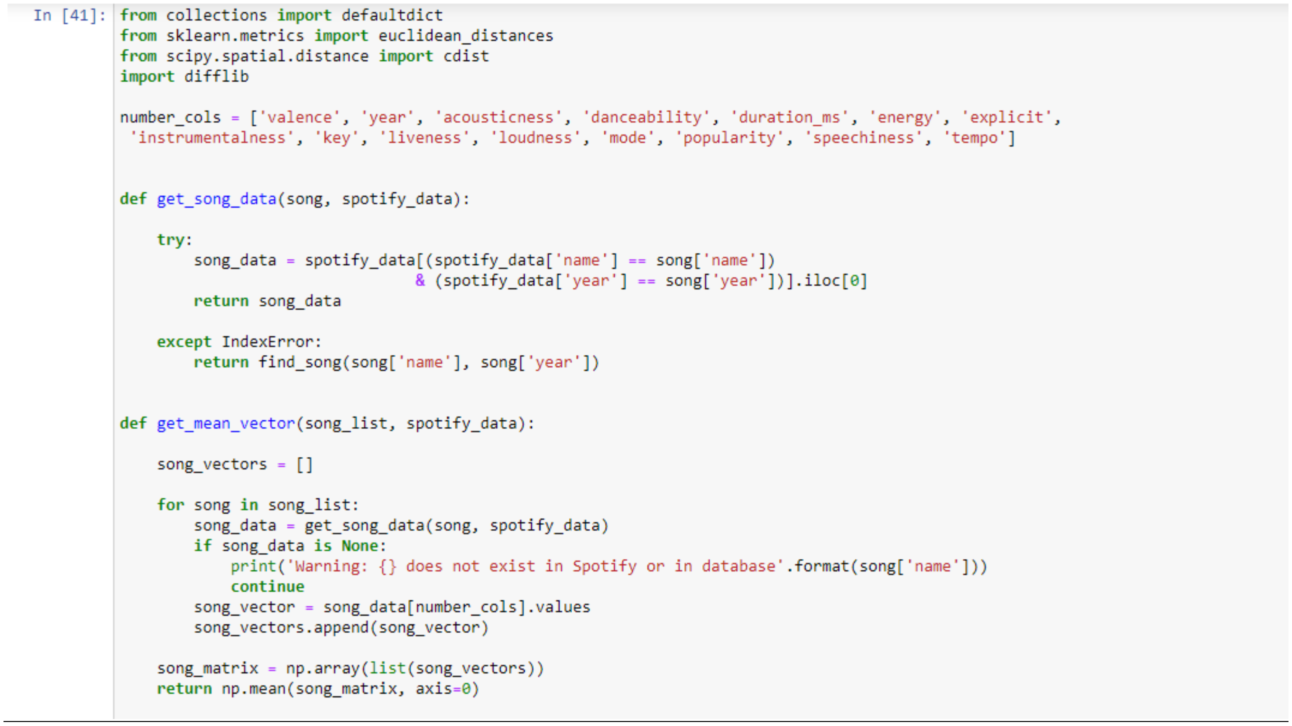
Cluster of Songs

Model Building:

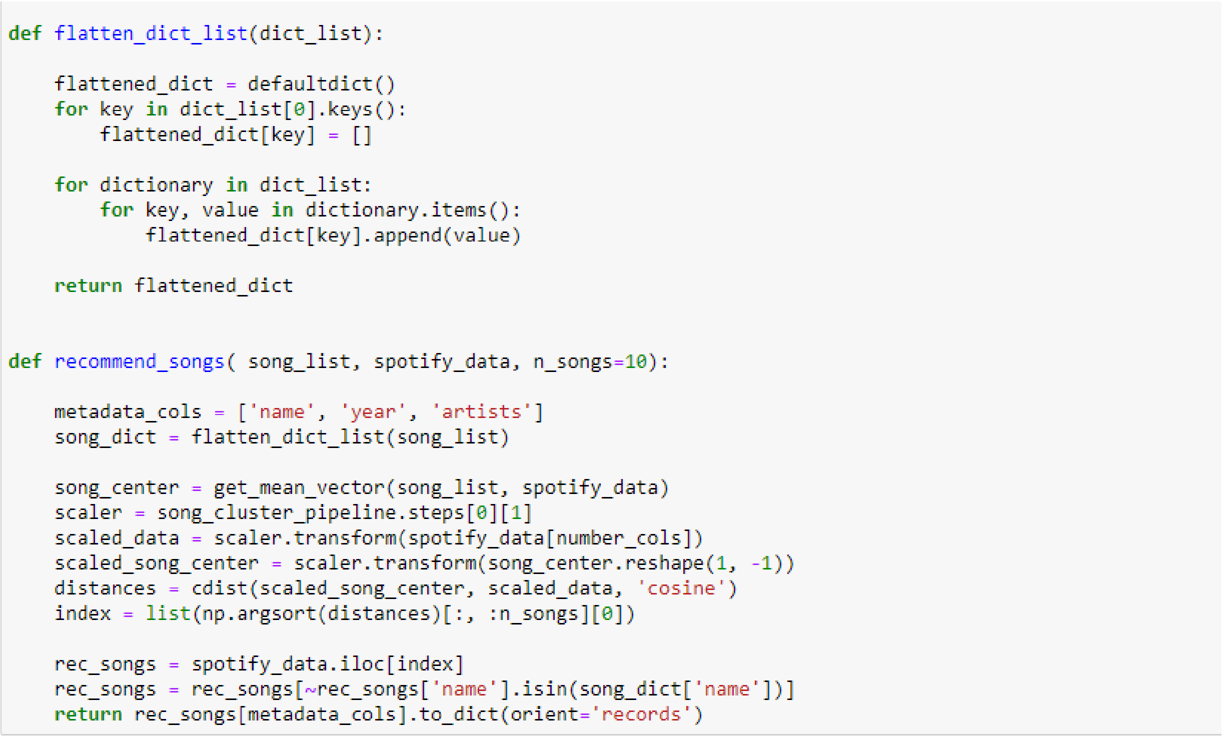


Model Bulding

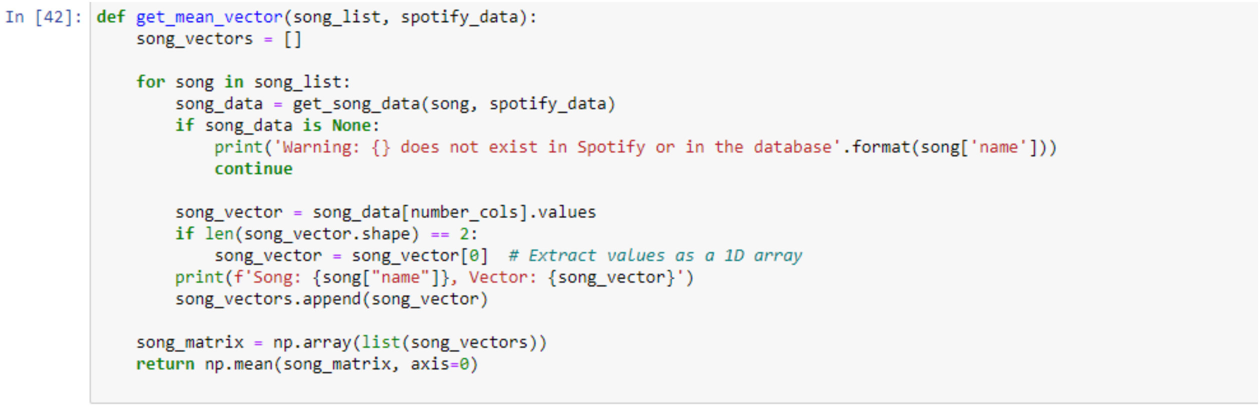
```
In [40]: import spotipy
         from spotipy.oauth2 import SpotifyClientCredentials
         from collections import defaultdict

         %env SPOTIFY_CLIENT_ID= 05f87c2b3ca74f7a8a4558fb77706582
         %env SPOTIFY_CLIENT_SECRET= d10b7d70018e4d18a90a3bfa89703099


         sp = spotipy.Spotify(auth_manager=SpotifyClientCredentials(client_id=os.environ["SPOTIFY_CLIENT_ID"],
                                                                    client_secret=os.environ["SPOTIFY_CLIENT_SECRET"]))

         def find_song(name, year):
             song_data = defaultdict()
             results = sp.search(q= 'track: {} year: {}'.format(name,year), limit=1)
             if results['tracks']['items'] == []:
                 return None

             results = results['tracks']['items'][0]
             track_id = results['id']
             audio_features = sp.audio_features(track_id)[0]

             song_data['name'] = [name]
             song_data['year'] = [year]
             song_data['explicit'] = [int(results['explicit'])]
             song_data['duration_ms'] = [results['duration_ms']]
             song_data['popularity'] = [results['popularity']]

             for key, value in audio_features.items():
                 song_data[key] = value

             return pd.DataFrame(song_data)

         env: SPOTIFY_CLIENT_ID=05f87c2b3ca74f7a8a4558fb77706582
         env: SPOTIFY_CLIENT_SECRET=d10b7d70018e4d18a90a3bfa89703099
```



```
In [41]: from collections import defaultdict
         from sklearn.metrics import euclidean_distances
         from scipy.spatial.distance import cdist
         import difflib

         number_cols = ['valence', 'year', 'acousticness', 'danceability', 'duration_ms', 'energy', 'explicit',
          'instrumentalness', 'key', 'liveness', 'loudness', 'mode', 'popularity', 'speechiness', 'tempo']


         def get_song_data(song, spotify_data):

             try:
                 song_data = spotify_data[(spotify_data['name'] == song['name'])
                                         & (spotify_data['year'] == song['year'])].iloc[0]
                 return song_data

             except IndexError:
                 return find_song(song['name'], song['year'])


         def get_mean_vector(song_list, spotify_data):

             song_vectors = []

             for song in song_list:
                 song_data = get_song_data(song, spotify_data)
                 if song_data is None:
                     print('Warning: {} does not exist in Spotify or in database'.format(song['name']))
                     continue
                 song_vector = song_data[number_cols].values
                 song_vectors.append(song_vector)

             song_matrix = np.array(list(song_vectors))
             return np.mean(song_matrix, axis=0)
```

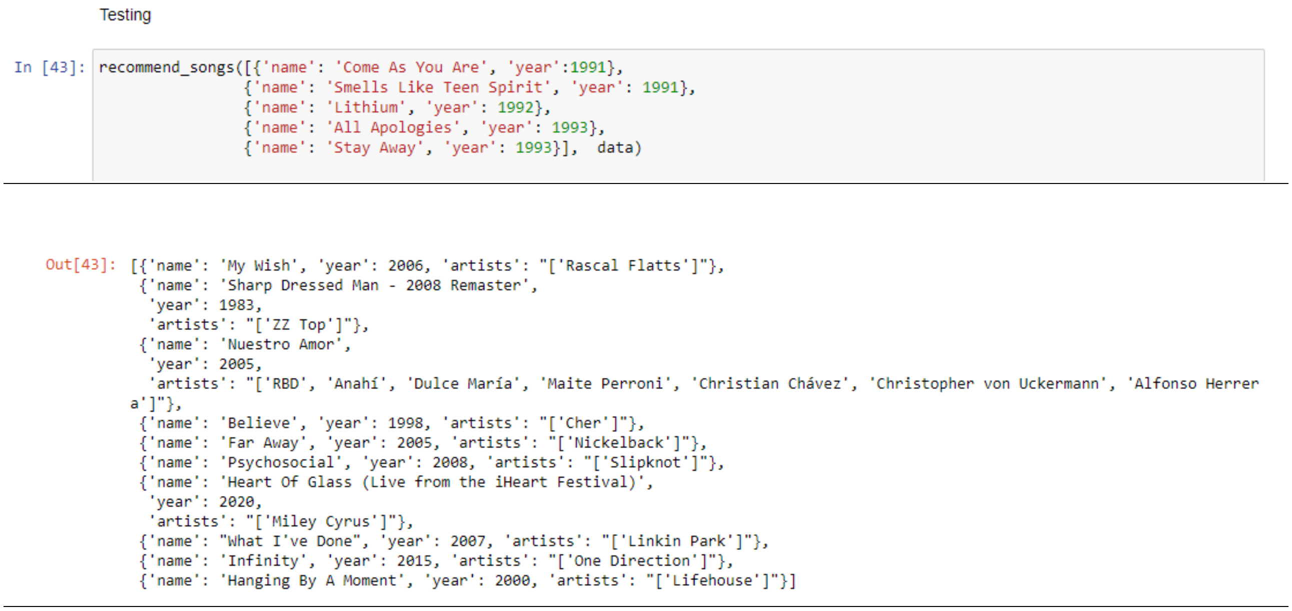


```
def flatten_dict_list(dict_list):

    flattened_dict = defaultdict()
    for key in dict_list[0].keys():
        flattened_dict[key] = []

    for dictionary in dict_list:
        for key, value in dictionary.items():
            flattened_dict[key].append(value)

    return flattened_dict


def recommend_songs( song_list, spotify_data, n_songs=10):

    metadata_cols = ['name', 'year', 'artists']
    song_dict = flatten_dict_list(song_list)

    song_center = get_mean_vector(song_list, spotify_data)
    scaler = song_cluster_pipeline.steps[0][1]
    scaled_data = scaler.transform(spotify_data[number_cols])
    scaled_song_center = scaler.transform(song_center.reshape(1, -1))
    distances = cdist(scaled_song_center, scaled_data, 'cosine')
    index = list(np.argsort(distances)[:, :n_songs][0])

    rec_songs = spotify_data.iloc[index]
    rec_songs = rec_songs[~rec_songs['name'].isin(song_dict['name'])]
    return rec_songs[metadata_cols].to_dict(orient='records')
```



```
In [42]: def get_mean_vector(song_list, spotify_data):
             song_vectors = []

             for song in song_list:
                 song_data = get_song_data(song, spotify_data)
                 if song_data is None:
                     print('Warning: {} does not exist in Spotify or in the database'.format(song['name']))
                     continue

                 song_vector = song_data[number_cols].values
                 if len(song_vector.shape) == 2:
                     song_vector = song_vector[0]  # Extract values as a 1D array
                 print(f'Song: {song["name"]}, Vector: {song_vector}')
                 song_vectors.append(song_vector)

             song_matrix = np.array(list(song_vectors))
             return np.mean(song_matrix, axis=0)
```

Testing :

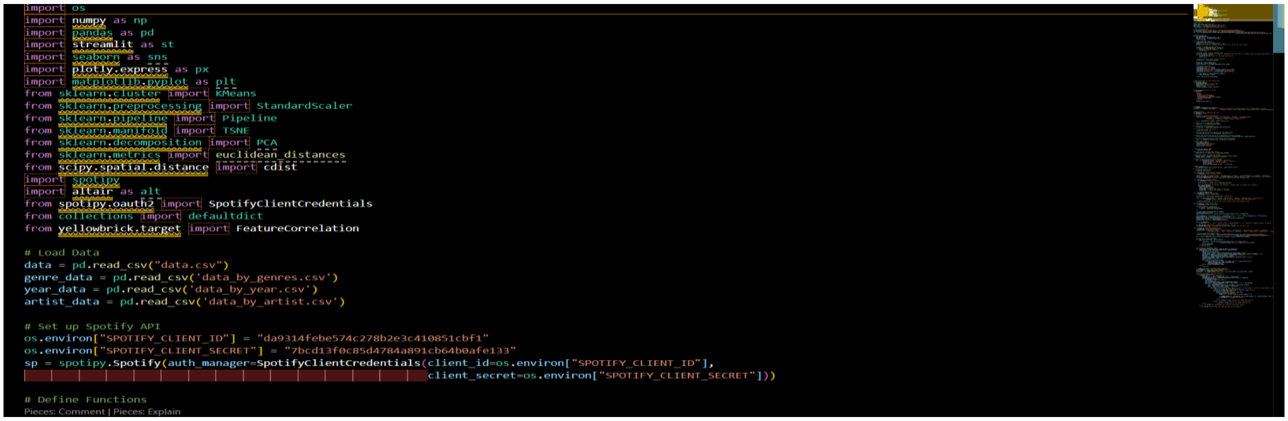


Testing

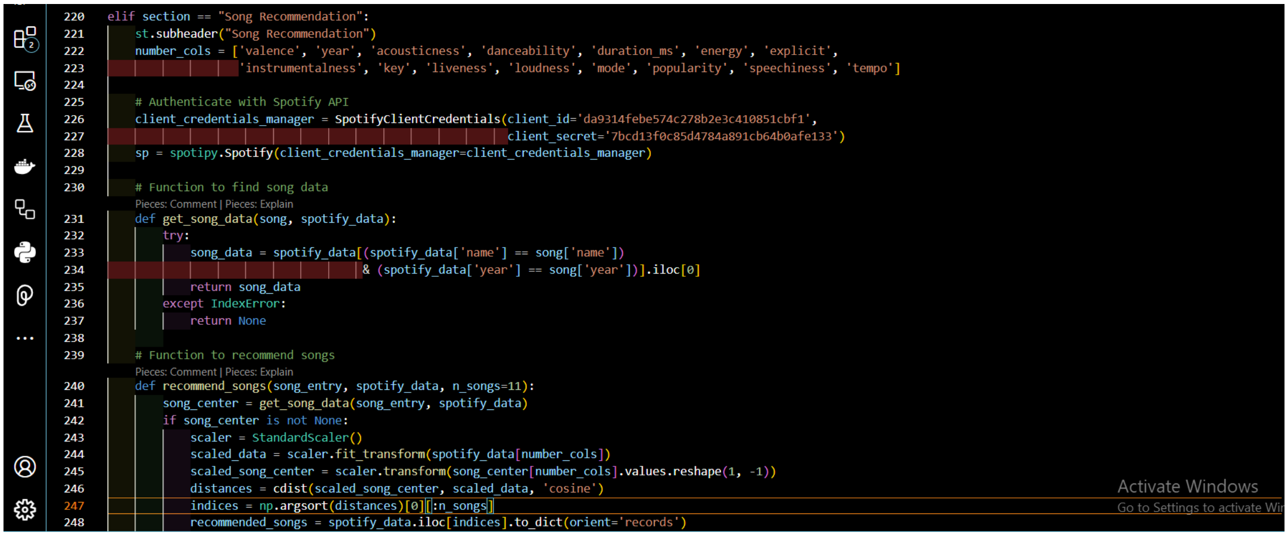
```
In [43]: recommend_songs([{'name': 'Come As You Are', 'year':1991},
                          {'name': 'Smells Like Teen Spirit', 'year': 1991},
                          {'name': 'Lithium', 'year': 1992},
                          {'name': 'All Apologies', 'year': 1993},
                          {'name': 'Stay Away', 'year': 1993}],  data)
```

```
Out[43]: [{'name': 'My Wish', 'year': 2006, 'artists': "['Rascal Flatts']"},
          {'name': 'Sharp Dressed Man - 2008 Remaster',
           'year': 1983,
           'artists': "['ZZ Top']"},
          {'name': 'Nuestro Amor',
           'year': 2005,
           'artists': "['RBD', 'Anahí', 'Dulce María', 'Maite Perroni', 'Christian Chávez', 'Christopher von Uckermann', 'Alfonso Herrer
         a']"},
          {'name': 'Believe', 'year': 1998, 'artists': "['Cher']"},
          {'name': 'Far Away', 'year': 2005, 'artists': "['Nickelback']"},
          {'name': 'Psychosocial', 'year': 2008, 'artists': "['Slipknot']"},
          {'name': 'Heart Of Glass (Live from the iHeart Festival)',
           'year': 2020,
           'artists': "['Miley Cyrus']"},
          {'name': "What I've Done", 'year': 2007, 'artists': "['Linkin Park']"},
          {'name': 'Infinity', 'year': 2015, 'artists': "['One Direction']"},
          {'name': 'Hanging By A Moment', 'year': 2000, 'artists': "['Lifehouse']"}]
```

Python Code for web implementation

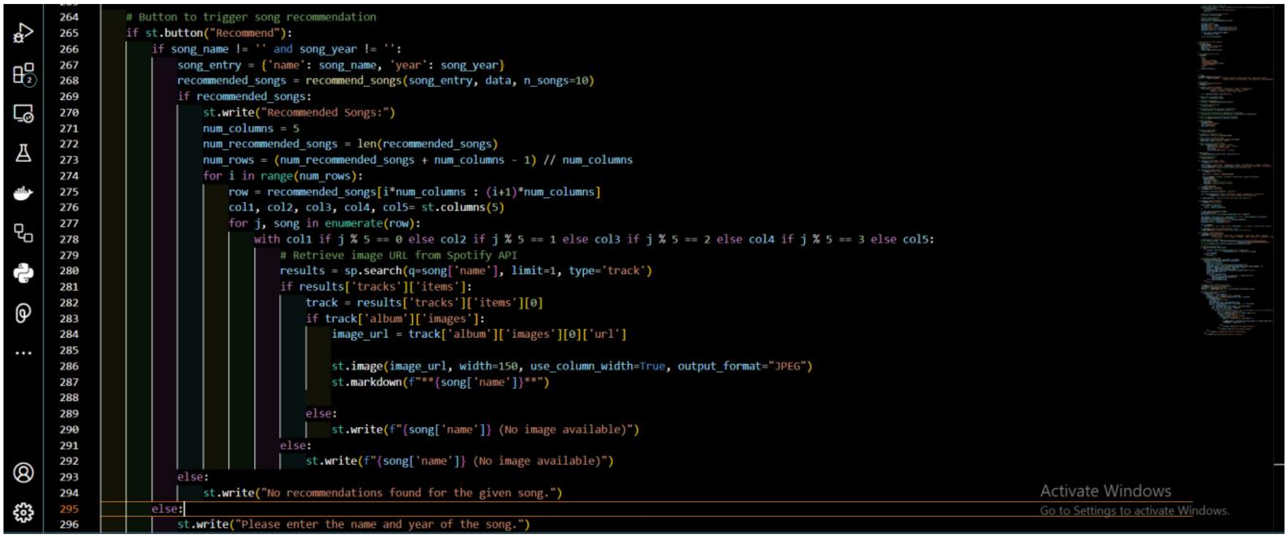


```python
import os
import numpy as np
import pandas as pd
import streamlit as st
import seaborn as sns
import plotly.express as px
import matplotlib.pyplot as plt
from sklearn.cluster import KMeans
from sklearn.preprocessing import StandardScaler
from sklearn.pipeline import Pipeline
from sklearn.manifold import TSNE
from sklearn.decomposition import PCA
from sklearn.metrics import euclidean_distances
from scipy.spatial.distance import cdist
import spotipy
import altair as alt
from spotipy.oauth2 import SpotifyClientCredentials
from collections import defaultdict
from yellowbrick.target import FeatureCorrelation

# Load Data
data = pd.read_csv("data.csv")
genre_data = pd.read_csv('data_by_genres.csv')
year_data = pd.read_csv('data_by_year.csv')
artist_data = pd.read_csv('data_by_artist.csv')

# Set up Spotify API
os.environ["SPOTIFY_CLIENT_ID"] = "da9314febe574c278b2e3c410851cbf1"
os.environ["SPOTIFY_CLIENT_SECRET"] = "7bcd13f0c85d4784a891cb64b0afe133"
sp = spotipy.Spotify(auth_manager=SpotifyClientCredentials(client_id=os.environ["SPOTIFY_CLIENT_ID"],
                                                           client_secret=os.environ["SPOTIFY_CLIENT_SECRET"]))

# Define Functions
```



```python
elif section == "Song Recommendation":
    st.subheader("Song Recommendation")
    number_cols = ['valence', 'year', 'acousticness', 'danceability', 'duration_ms', 'energy', 'explicit',
                   'instrumentalness', 'key', 'liveness', 'loudness', 'mode', 'popularity', 'speechiness', 'tempo']

    # Authenticate with Spotify API
    client_credentials_manager = SpotifyClientCredentials(client_id='da9314febe574c278b2e3c410851cbf1',
                                                          client_secret='7bcd13f0c85d4784a891cb64b0afe133')
    sp = spotipy.Spotify(client_credentials_manager=client_credentials_manager)

    # Function to find song data
    def get_song_data(song, spotify_data):
        try:
            song_data = spotify_data[(spotify_data['name'] == song['name'])
                                     & (spotify_data['year'] == song['year'])].iloc[0]
            return song_data
        except IndexError:
            return None

    # Function to recommend songs
    def recommend_songs(song_entry, spotify_data, n_songs=11):
        song_center = get_song_data(song_entry, spotify_data)
        if song_center is not None:
            scaler = StandardScaler()
            scaled_data = scaler.fit_transform(spotify_data[number_cols])
            scaled_song_center = scaler.transform(song_center[number_cols].values.reshape(1, -1))
            distances = cdist(scaled_song_center, scaled_data, 'cosine')
            indices = np.argsort(distances)[0][:n_songs]
            recommended_songs = spotify_data.iloc[indices].to_dict(orient='records')
```

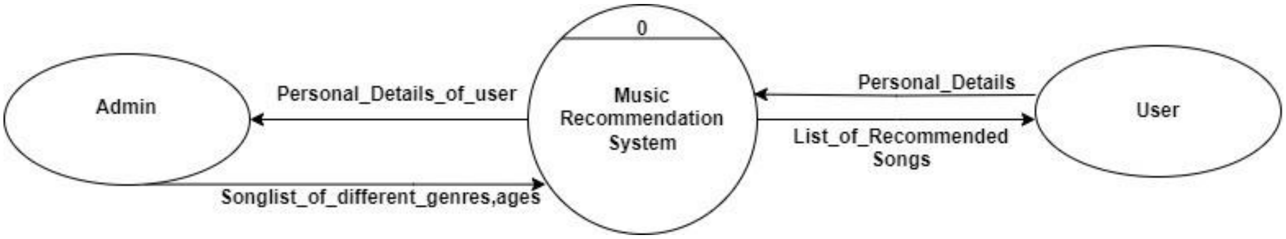
Pieces: Comment | Pieces: Explain
Pieces: Comment | Pieces: Explain
Activate Windows
Go to Settings to activate Wir



```python
    # Button to trigger song recommendation
    if st.button("Recommend"):
        if song_name != '' and song_year != '':
            song_entry = {'name': song_name, 'year': song_year}
            recommended_songs = recommend_songs(song_entry, data, n_songs=10)
            if recommended_songs:
                st.write("Recommended Songs:")
                num_columns = 5
                num_recommended_songs = len(recommended_songs)
                num_rows = (num_recommended_songs + num_columns - 1) // num_columns
                for i in range(num_rows):
                    row = recommended_songs[i*num_columns : (i+1)*num_columns]
                    col1, col2, col3, col4, col5= st.columns(5)
                    for j, song in enumerate(row):
                        with col1 if j % 5 == 0 else col2 if j % 5 == 1 else col3 if j % 5 == 2 else col4 if j % 5 == 3 else col5:
                            # Retrieve image URL from Spotify API
                            results = sp.search(q=song['name'], limit=1, type='track')
                            if results['tracks']['items']:
                                track = results['tracks']['items'][0]
                                if track['album']['images']:
                                    image_url = track['album']['images'][0]['url']

                                    st.image(image_url, width=150, use_column_width=True, output_format="JPEG")
                                    st.markdown(f"**{song['name']}**")

                                else:
                                    st.write(f"{song['name']} (No image available)")
                            else:
                                st.write(f"{song['name']} (No image available)")
            else:
                st.write("No recommendations found for the given song.")
        else:
            st.write("Please enter the name and year of the song.")
```

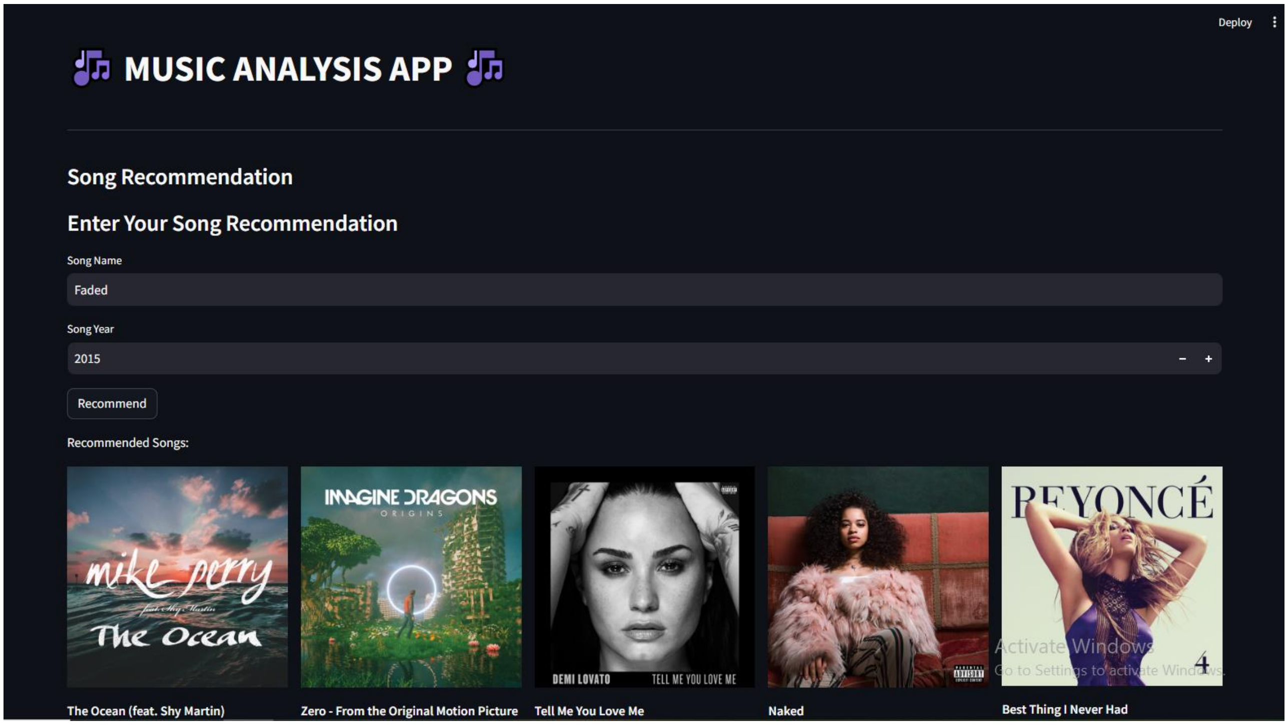
DFD LEVEL 0

Level 0 is the highest-level Data Flow Diagram (DFD), which provides an overview of the entire system. It shows the major processes, data flows, and data stores in the system, without providing any details about the internal workings of these processes. It is also known as a context diagram. It's designed to be an abstraction view, showing the system as a single process with its relationship to external entities. It represents the entire system as a single bubble with input and output data indicated by incoming/outgoing arrows.


0
Admin
Personal_Details_of_user
Songlist_of_different_genres,ages
Music Recommendation System
Personal_Details
List_of_Recommended Songs
User

WEB IMPLEMENTATION

Homepage:



🎶 MUSIC ANALYSIS APP 🎶

Song Recommendation

Enter Your Song Recommendation

Song Name: Faded

Song Year: 2015

Recommended Songs:

The Ocean (feat. Shy Martin)
Zero - From the Original Motion Picture
Tell Me You Love Me
Naked
Best Thing I Never Had

CONCLUSION

In conclusion, the development and implementation of a music recommendation system have proven to be a significant step towards enhancing the music listening experience for users. Through this project we have successfully leveraged the power of machine learning and data analysis techniques to create a personalized and intelligent system that caters to individual preferences and help users discover new music.

By utilizing collaborative filtering algorithm, content-based filtering and hybrid approaches, we have achieved accurate and relevant music recommendations. The system takes into account various factors such as user preferences, listening theory, genre and similar artists, providing a diverse and tailored section of music to users.

The projects has also emphasized the importance of data collection and preprocessing as well as the significance of a well-designed user interface. These aspects have played crucial roles in ensuring a seamless and user-friendly experience allowing users to easily interact with the recommendation system and explore new music effortlessly.

Furthermore, the project has highlighted the potential impact of music recommendation systems in the music industry. By connecting users with lesser known artists and promoting music discovery, these systems can help in fostering a more inclusive and diverse musical landscape . Additionally the system can also benefit music streaming platforms by increasing user engagement, retention and overall satisfaction.

FUTURE PLAN

While our project has achieved considerable success there are still opportunities for future improvements and enhancements. For instance, incorporating more advanced machine learning techniques such as deep learning or reinforcement learning could potentially enhance the accuracy and relevance of recommendations. Additionally integrating real time user feedback and incorporating social aspects such as user reviews and recommendations from friends could further enhance the personalization and discovery capabilities of the system.

Overall , the music recommendation system project has provided valuable insights into the world of recommendation systems, music analysis and user experience designs. It has demonstrated the potential of leveraging data and machine learning to create personalized and engaging experience for music enthusiasts. With further advancements and refinements music recommendation system have the potential to revolutionize the way we discover explore and enjoy music.

BIBIOGRAPHY

- https://www.researchgate.net/publication/353485380_Music_Recommendation_Systems_Models_and_Methods_A_Review
- https://www.datacamp.com/tutorial/installing-anaconda-windows
- https://ieeexplore.ieee.org/document/10275967
- https://towardsdatascience.com/the-state-of-recommender-systems-for-music-in-2020-180b3ddb392f
- https://www.geeksforgeeks.org/levels-in-data-flow-diagrams-dfd/\
- https://www.britannica.com/art/fashion-industry