Spotify Music Recommendation System

Project Overview

The Spotify Music Recommendation System is designed to suggest similar songs to a user's input track by combining musical content analysis and song popularity. By utilizing both content-based filtering and a hybrid approach that incorporates song popularity, the system delivers personalized recommendations. This approach ensures that users receive recommendations based on the characteristics of the music they enjoy, while also factoring in the popularity of tracks, ensuring that the recommendations are relevant and widely appreciated.

Project Workflow

1. Spotify API Authentication

The first step in building the recommendation system is to access Spotify's music data through their API. Spotify requires applications to authenticate using Client ID and Client Secret to make authorized data requests.

Goal: The system needs an access token to fetch music data such as track names, artists, albums, release dates, popularity scores, and musical features from Spotify's extensive database.

Result: After successful authentication, the system can make authorized requests to Spotify's API, enabling the retrieval of detailed song information required for the recommendation engine.

2. Data Retrieval from Spotify

Once the system is authenticated, it retrieves comprehensive data for songs from Spotify.

The relevant attributes for each song include:

Track Name: The name of the song.

Artists: The artists who performed the song.

Album Name: The album in which the song was released.

Release Date: The date when the song or album was published.

Popularity: A numerical score assigned by Spotify, which reflects the song's popularity based on its number of plays and user engagement.

The system also retrieves important musical features that help define the characteristics of each song. These features include:

Danceability: Reflects how suitable the track is for dancing based on tempo and rhythm stability.

Energy: Measures the intensity and activity of a song.

Tempo: The speed of the song.

Valence: A score indicating the musical positivity or happiness of a track.

And more...

These musical features form the foundation of the content-based recommendation process.

3. Content-Based Recommendation Engine

The recommendation engine primarily uses content-based filtering, which suggests songs that are similar to a user's input song based on their musical features. The similarity between

tracks is measured by comparing these features to identify songs that "sound" alike.
How It Works: When a user inputs a song, the system extracts its musical features and compares them with all other songs in the dataset. The songs that share the most similar musical traits are then selected as recommendations.
Result: The top recommendations consist of songs that closely match the input song's musical style, ensuring that users receive suggestions that align with their taste.

4. Hybrid Recommendation System

To further enhance the recommendation quality, the system incorporates a hybrid recommendation model. This hybrid model merges content-based filtering with song popularity, ensuring that the recommended songs are not only musically similar but also have significant popularity.
Popularity Weighting: The hybrid approach adjusts the recommendations by factoring in the popularity of each song. Popularity scores are weighted based on the song's release date, allowing the system to favor newer or older songs depending on user preference or general trends.
Balanced Recommendations: The hybrid system blends content-based similarity with popularity data, making the final recommendations more relevant and well-rounded. This ensures that users are presented with songs that are both similar to their input and widely liked by other listeners.

5. User Input and System Output

The system allows users to input the name of a song, and it returns a list of recommended songs based on the user's musical preferences. Each recommended song is accompanied by key details such as:Track Name, Artists, Album Name, Release Date and Popularity Score.

CODE

To obtain an access token from the Spotify API

```
import requests      #used for making HTTP requests
import base64        #used for encoding and decoding data in Base64 format

# Replace with your own Client ID and Client Secret
client_id = '24f4fc17e73346d695f513ad749da0d8'      #Identifies the application
client_secret = 'bf1c85de7c764795bda229fed19d1ac1' #Used for authentication

# Combining and Base64 encode the client ID and client secret
client_credentials = f"{client_id}:{client_secret}" #Combines
client_credentials_base64 = base64.b64encode(client_credentials.encode()) #Encodes the combined credentials
```

```
# Request the access token
token_url = 'https://accounts.spotify.com/api/token'
headers = {
    'Authorization': f'Basic {client_credentials_base64.decode()}'
#Contains the HTTP headers
}
data = {
    'grant_type': 'client_credentials'        #It is done to request
an access token
}
response = requests.post(token_url, data=data, headers=headers)
#Sends the POST request to the token_url with the specified headers
and data, and stores the response

if response.status_code == 200: #200= HTTP OK
    access_token = response.json()['access_token']    #Extracts the
access token from the JSON response body
    print("Access token obtained successfully.\nNow, we can make
authorized requests to retrieve music data, which is fundamental for
building a music recommendation system.")
else:
    print("Error obtaining access token.")
    exit()
```

```
Access token obtained successfully.
Now, we can make authorized requests to retrieve music data, which
is fundamental for building a music recommendation system.
```

Get music data from Spotify

Now, we will be creating a function **to get and process music data** from any playlist on Spotify. For this task, first we will be installing the **"Spotipy"** library, which is a Python library providing access to Spotify's web API.

To install the python Library use the mentioned command : !pip install spotipy

```
#Importing Python Libraries
import pandas as pd   #Used for for data manipulation and analysis
import spotipy        #provides a Python client for the Spotify Web
API
from spotipy.oauth2 import SpotifyOAuth #Used for handling OAuth
authentication with Spotify
```

```
C:\Users\salon\anaconda3\lib\site-packages\pandas\core\computation\
expressions.py:20: UserWarning: Pandas requires version '2.7.3' or
newer of 'numexpr' (version '2.7.1' currently installed).
  from pandas.core.computation.check import NUMEXPR_INSTALLED
```

```
#Defining Function
def get_playlist_data(playlist_id, client_id, client_secret,
redirect_uri):
```

```
    # Set up Spotipy with SpotifyOAuth
    sp = spotipy.Spotify(auth_manager=SpotifyOAuth(
        client_id=client_id,
        client_secret=client_secret,
        redirect_uri=redirect_uri,
        scope="playlist-read-private"
    ))

    # Get the tracks from the playlist
    tracks = sp.playlist_tracks(playlist_id, fields='items(track(id,
name, artists, album(id, name)))')['items']

    # Extract relevant information and store in a list of
dictionaries
    music_data = []
    for track_info in tracks:
        track = track_info['track']
        artists = ', '.join(artist['name'] for artist in
track['artists']) #Creates a comma-separated string of artist names
for each track

        # Retrieveing audio features, album information and track
information
        audio_features = sp.audio_features(track['id'])[0]
        album_info = sp.album(track['album']['id'])
        track_info = sp.track(track['id'])

        # Add relevant track data in music_data list
        music_data.append({
            'Track Name': track['name'],
            'Artists': artists,
            'Album Name': track['album']['name'],
            'Popularity': track_info['popularity'],
            'Release Date': album_info['release_date'],
            'Duration (ms)': audio_features['duration_ms'],
            'Danceability': audio_features['danceability'],
            'Energy': audio_features['energy'],
            'Tempo': audio_features['tempo'],
            #Can Add more attributes as needed
        })

    # Create a pandas DataFrame from the list of dictionaries
    return pd.DataFrame(music_data)

#Calling 'get_playlist_data' function to collect music data from the
given playlist on Spotify:
playlist_id = '37i9dQZF1DX5KpP2LN299J'
client_id = '24f4fc17e73346d695f513ad749da0d8'
client_secret = 'bf1c85de7c764795bda229fed19d1ac1'
redirect_uri = 'https://statso.io/'

# Call the function to get the music data and store it in a
DataFrame
music_df = get_playlist_data(playlist_id, client_id, client_secret,
```

```
redirect_uri)

# Display the DataFrame
print(music_df)
```

```
                                          Track Name  \
0                                             august
1                      I Can Do It With a Broken Heart
2                       Blank Space (Taylor's Version)
3                        Fortnight (feat. Post Malone)
4                        Love Story (Taylor’s Version)
..                                                 ...
95                        Last Kiss (Taylor's Version)
96  Delicate - Recorded at The Tracking Room Nashv...
97                                              Paris
98  Message In A Bottle (Taylor's Version) (From T...
99                         This Love (Taylor’s Version)

                      Artists                       Album Name
Popularity  \
0                 Taylor Swift                         folklore
87
1                 Taylor Swift  THE TORTURED POETS DEPARTMENT
86
2                 Taylor Swift        1989 (Taylor's Version)
79
3   Taylor Swift, Post Malone THE TORTURED POETS DEPARTMENT
89
4                 Taylor Swift  Love Story (Taylor’s Version)
79
..                         ...                             ...
...
95                Taylor Swift   Speak Now (Taylor's Version)
68
96                Taylor Swift                Spotify Singles
57
97                Taylor Swift        Midnights (3am Edition)
65
98                Taylor Swift         Red (Taylor's Version)
70
99                Taylor Swift   This Love (Taylor’s Version)
66

   Release Date  Duration (ms)  Danceability  Energy    Tempo
0    2020-07-24         261923         0.532   0.623   89.937
1    2024-04-18         218005         0.701   0.751  129.994
2    2023-10-26         231833         0.732   0.719   96.035
3    2024-04-18         228965         0.504   0.386  192.004
4    2021-02-12         235767         0.626   0.790  119.082
..          ...            ...           ...     ...      ...
95   2023-07-07         369120         0.557   0.362   79.573
96   2018-04-13         228461         0.664   0.447   90.976
97   2022-10-22         196259         0.703   0.518  110.940
98   2021-11-12         225960         0.622   0.791  115.915
```

```
99   2022-05-06          250100          0.470   0.494  144.039

[100 rows x 9 columns]
```

```
#To check if the data has some NULL values
print(music_df.isnull().sum())
```

Music Recommendation System

Its a HYBRID RECOMMENDATION SYSTEM. The hybrid approach aims to provide more personalized and relevant recommendations by considering both the content similarity of songs and their weighted popularity, which is basically the combination of two appproches.

Approach 1: In this the approach will be based on recommending music based on **music audio features**.
Approach 2:In this the approach will be based on recommending music based on **weighted popularity**.

Importing Python Libraries

```
import pandas as pd     #Used for data manipulation and analysis
import numpy as np      #Used for for numerical operations
from sklearn.model_selection import train_test_split #Used for
splitting data into training and testing sets
from sklearn.preprocessing import MinMaxScaler         #Used for
scaling features to a given range, usually [0, 1]
from datetime import datetime                    #Used for handling
date and time
from sklearn.metrics.pairwise import cosine_similarity #Used for
calculating the cosine similarity between vectors

data=music_df
```

Approach 1

In this Approach, we will define a function in which we will calculate the similarity scores between the audio features of the input song and all other songs in the dataset"data". It uses **cosine similarity**, a common measure used in **content-based filtering**. The cosine_similarity function from scikit-learn is employed to compute these similarity scores.Here, it will measure how close the input song is to every other song in the dataset, based on features.

```
# Defining Function to get 5 content-based recommendations based on
music features
def content_based_recommendations(input_song_name,
num_recommendations=5):

    if input_song_name not in music_df['Track Name'].values:
        print(f"'{input_song_name}' not found in the dataset. Please
enter a valid song name.")
        return

    # Get the index of the input song in the music DataFrame
    input_song_index = music_df[music_df['Track Name'] ==
input_song_name].index[0]

    # Calculate the similarity scores based on music features
(cosine similarity)
    # The cosine_similarity function takes the scaled features of
the input song and compares it with the scaled features of all songs
    similarity_scores =
cosine_similarity([music_features_scaled[input_song_index]],
music_features_scaled)

    # Get the indices of the most similar songs
    similar_song_indices = similarity_scores.argsort()[0][::-1]
[1:num_recommendations + 1]

    #similarity_scores.argsort() returns the indices that would sort
the similarity scores.
    #[0] selects the first row of similarity scores (since there is
only one input song).
    #[::-1] reverses the list to get songs in descending order of
similarity.(Top Recommendation will be at first)
    #[1:num_recommendations + 1] slices the list to get the top
num_recommendations(5)songs (excluding the input song itself).

    # Get the names of the most similar songs based on content-based
filtering
    content_based_recommendations =
music_df.iloc[similar_song_indices][['Track Name', 'Artists', 'Album
Name', 'Release Date', 'Popularity']]

    return content_based_recommendations
```

Approach 2

While providing music recommendations to users, it is important to recommend the latest releases. For this, we need to give more weight to the latest releases in the recommendations.For which we will use Approach 2. The idea behind this approach, we will define a function where, the **"weight decreases"** when the time span between the release date and today increases, whcih means more recent releases will have a higher weight, while older releases will have a lower weight. As a result, when combining this weighted popularity score with other factors in a recommendation system, **recent tracks will have a more significant impact on the final recommendations**, reflecting user's potential interest in newer music.

```
# Defining Function to calculate weighted popularity scores based on release date
def calculate_weighted_popularity(release_date):
    # Convert the release date to datetime object
    release_date = datetime.strptime(release_date, '%Y-%m-%d') #This will allow to perform the arthimatic operations on dates
    # Calculate the time span between release date and today's date
    time_span = datetime.now() - release_date

    # Calculate the weighted popularity score based on time span (e.g., more recent releases have higher weight)
    weight = 1 / (time_span.days + 1) #Adding 1 to ensure that the weight is never 0
    return weight

# Normalize the music features using Min-Max scaling
scaler = MinMaxScaler()
music_features = music_df[['Danceability', 'Energy', 'Tempo']].values
music_features_scaled = scaler.fit_transform(music_features) #fits the scaler and then applies the transformation to scale
```

Hybrid Approach

In the Hybrid Approach, we will define a function which will retrieve the content based recommendation and the popularity score from the 'music_df' DataFrame.Then, it will **calculate the weighted popularity score** using popularity score and calculate_weighted_popularity function (previously defined) based on the release date of the input song. Also, The **alpha** parameter will control the relative importance of content-based and popularity-based recommendations. Further,This function will **combine** the content-based recommendations with the input song's information (track name, artists, album name, release date, and popularity) and its weighted popularity score.

```
#Defining the function for Hybrid Recommendation
def hybrid_recommendations(input_song_name, num_recommendations=5, alpha=0.5):
    if input_song_name not in music_df['Track Name'].values:
        print(f"'{input_song_name}' not found in the dataset. Please enter a valid song name.")
        return
    #Reteriving the Content Based Recommendations and Popularity score
    content_based_rec = content_based_recommendations(input_song_name, num_recommendations)

    popularity_score = music_df.loc[music_df['Track Name'] == input_song_name, 'Popularity'].values[0]

    #Calulating Weighted Popularity score
    weighted_popularity_score = popularity_score * calculate_weighted_popularity(
```

```
        music_df.loc[music_df['Track Name'] == input_song_name,
'Release Date'].values[0]
    )

    #Creating a dataframe where popularity of the input song is
replaced by weighted popularity score
    new_entry = pd.DataFrame({
        'Track Name': [input_song_name],
        'Artists': [music_df.loc[music_df['Track Name'] ==
input_song_name, 'Artists'].values[0]],
        'Album Name': [music_df.loc[music_df['Track Name'] ==
input_song_name, 'Album Name'].values[0]],
        'Release Date': [music_df.loc[music_df['Track Name'] ==
input_song_name, 'Release Date'].values[0]],
        'Popularity': [weighted_popularity_score]
    })

    #Combines content-based(content_based_rec df) and popularity-
based data(new_entry df)
    hybrid_recommendations = pd.concat([content_based_rec,
new_entry], ignore_index=True)

    #Sorting the Recommendations based on its popularity
    hybrid_recommendations =
hybrid_recommendations.sort_values(by='Popularity', ascending=False)

    #Removing the Input song from the Recommendations
    hybrid_recommendations =
hybrid_recommendations[hybrid_recommendations['Track Name'] !=
input_song_name]

    return hybrid_recommendations

# To list all the tracks in the playlist
print("Name of the Tracks in the Playlist\n")
track_names=data['Track Name'].tolist()
for track in track_names:
     print(track)
```

```
Name of the Tracks in the Playlist

august
I Can Do It With a Broken Heart
Blank Space (Taylor's Version)
Fortnight (feat. Post Malone)
Love Story (Taylor’s Version)
Cruel Summer
I Knew You Were Trouble (Taylor's Version)
Down Bad
You Belong With Me (Taylor’s Version)
Bad Blood (feat. Kendrick Lamar) (Taylor's Version)
Lover
Who’s Afraid of Little Old Me?
Anti-Hero
```

...Ready For It?
Style (Taylor's Version)
But Daddy I Love Him
cardigan
loml
my tears ricochet
Guilty as Sin?
willow
Florida!!! (feat. Florence + The Machine)
The Prophecy
So Long, London
Cassandra
Karma
My Boy Only Breaks His Favorite Toys
Is It Over Now? (Taylor's Version) (From The Vault)
How Did It End?
The Tortured Poets Department
Don’t Blame Me
You’re Losing Me (From The Vault)
us. (feat. Taylor Swift)
The Alchemy
Look What You Made Me Do
Say Don't Go (Taylor's Version) (From The Vault)
Wildest Dreams (Taylor's Version)
Enchanted (Taylor's Version)
You Need To Calm Down
"Slut!" (Taylor's Version) (From The Vault)
New Romantics (Taylor's Version)
Lavender Haze
We Are Never Ever Getting Back Together (Taylor's Version)
Out Of The Woods (Taylor's Version)
Mine (Taylor's Version)
I Don’t Wanna Live Forever (Fifty Shades Darker)
Now That We Don't Talk (Taylor's Version) (From The Vault)
All Of The Girls You Loved Before
Delicate
I Can See You (Taylor’s Version) (From The Vault)
Shake It Off (Taylor's Version)
Midnight Rain
thanK you aIMee
22 (Taylor's Version)
Back To December (Taylor's Version)
All Too Well (10 Minute Version) (Taylor's Version) (From The Vault)
Welcome To New York (Taylor's Version)
Snow On The Beach (feat. More Lana Del Rey)
Mean (Taylor's Version)
Clean (Taylor's Version)
Hits Different
Getaway Car
All You Had To Do Was Stay (Taylor's Version)
Bejeweled
Paper Rings
I Wish You Would (Taylor's Version)

```
Dear John (Taylor's Version)
You're On Your Own, Kid
The Man
Better Than Revenge (Taylor's Version)
If This Was A Movie (Taylor's Version)
Suburban Legends (Taylor's Version) (From The Vault)
Foolish One (Taylor's Version) (From The Vault)
Maroon
When Emma Falls in Love (Taylor's Version) (From The Vault)
Vigilante Shit
Electric Touch (feat. Fall Out Boy) (Taylor's Version) (From The
Vault)
Mastermind
Mr. Perfectly Fine (Taylor's Version) (From The Vault)
Long Live (Taylor's Version)
The Great War
Safe & Sound (feat. Joy Williams and John Paul White) (Taylor's
Version)
Timeless (Taylor's Version) (From The Vault)
The Way I Loved You (Taylor's Version)
Gorgeous
The Story Of Us (Taylor's Version)
Question...?
champagne problems
Castles Crumbling (feat. Hayley Williams) (Taylor's Version) (From
The Vault)
Eyes Open (Taylor's Version)
Ours (Taylor's Version)
Sweet Nothing
Sparks Fly (Taylor's Version)
exile (feat. Bon Iver)
Red
Last Kiss (Taylor's Version)
Delicate - Recorded at The Tracking Room Nashville
Paris
Message In A Bottle (Taylor's Version) (From The Vault)
This Love (Taylor's Version)
```

Spotify Music Recommendation System

Now, Our **Spotify Music Recommendation System** is ready to recommend the Top 5 songs for the given song name from the spotify's playlist based on its music feature and popularity.

```
#Enter the Song Name for which You need the recommendations
input_song_name=input("Enter the Song Name:")

#Calling Hybrid Recommendation Function to Recommend the songs
recommendations = hybrid_recommendations(input_song_name,
num_recommendations=5)
print(f"Hybrid recommended songs for '{input_song_name}':")
```

```
print("\n")
print(recommendations)
```

```
Enter the Song Name:Gorgeous
Hybrid recommended songs for 'Gorgeous':


                                                Track Name
Artists  \
3        "Slut!" (Taylor's Version) (From The Vault)  Taylor Swift

4                                        Lavender Haze  Taylor Swift

2                  All Of The Girls You Loved Before  Taylor Swift

0                       Last Kiss (Taylor's Version)  Taylor Swift

1  Delicate - Recorded at The Tracking Room Nashv...  Taylor Swift


                           Album Name Release Date  Popularity
3            1989 (Taylor's Version)   2023-10-26        76.0
4                           Midnights   2022-10-21        76.0
2  All Of The Girls You Loved Before   2019-08-23        75.0
0       Speak Now (Taylor's Version)   2023-07-07        68.0
1                    Spotify Singles   2018-04-13        57.0
```

Conclusion

The Spotify Music Recommendation System offers an advanced and user-centric approach to music recommendations. By combining content-based filtering (which ensures musical similarity) with a hybrid model that incorporates song popularity, the system delivers high-quality, personalized suggestions. Whether users are seeking lesser-known tracks that share their favorite song's sound or popular hits within the same style, this system provides a well-rounded set of recommendations tailored to each user's preferences.