RECOMMENDATION SYSTEM

Task :

Create a simple recommendation system that suggests items to users based on their preferences. You can use techniques like collaborative filtering or content-based filtering to recommend movies, books, or products to users.

Here "I am going to create a music recommendation system for users using the Spotify dataset."

Importing Libraries

In [1]:
```
import numpy as np
import pandas as pd
from tqdm import tqdm
from sklearn.preprocessing import MinMaxScaler
from sklearn.neighbors import NearestNeighbors
```

Loading The Dataset

In [2]:
```
data = pd.read_csv("data.csv")
```

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

Out[3]:

	id	name	artists	duration_ms	release_date	year	acousticness	danceability	energy	instrumentalness	liveness	loudness	speechiness	tempo	valence	mode	key	popularity	ex
0	6KbQ3uYMLKb5jDxLF7wYDD	Singende Bataillone 1. Teil	['Carl Woitschach']	158648	1928	1928	0.995	0.708	0.1950	0.563	0.1510	-12.428	0.0506	118.469	0.7790	1	10	0	
1	6KuQTIu1KoTTkLXKrwlLPV	Fantasiestücke, Op. 111: Più tosto lento	['Robert Schumann', 'Vladimir Horowitz']	282133	1928	1928	0.994	0.379	0.0135	0.901	0.0763	-28.454	0.0462	83.972	0.0767	1	8	0	
2	6L63VW0PibdM1HDSBoqnoM	Chapter 1.18 - Zamek kaniowski	['Seweryn Goszczyński']	104300	1928	1928	0.604	0.749	0.2200	0.000	0.1190	-19.924	0.9290	107.177	0.8800	0	5	0	
3	6M94FkXd15sOAOQYRnWPN8	Bebamos Juntos - Instrumental (Remasterizado)	['Francisco Canaro']	180760	9/25/28	1928	0.995	0.781	0.1300	0.887	0.1110	-14.734	0.0926	108.003	0.7200	0	1	0	
4	6N6tiFZ9vLTSOIxkj8qKrd	Polonaise-Fantaisie in A-Flat Major, Op. 61	['Frédéric Chopin', 'Vladimir Horowitz']	687733	1928	1928	0.990	0.210	0.2040	0.908	0.0980	-16.829	0.0424	62.149	0.0693	1	11	1	

Drop Unwanted columns

In [4]:
```
features = data.drop(columns=['id', 'name', 'artists', 'release_date', 'year', 'explicit', 'mode', 'key'])
```

Normalize numerical features

In [5]:
```
scaler = MinMaxScaler()
normalized_features = scaler.fit_transform(features)
```

Use Nearest Neighbors to find similar songs

In [6]:
```
nn = NearestNeighbors(n_neighbors=6, algorithm='auto', metric='cosine')
nn.fit(normalized_features)
```

Out[6]:
```
NearestNeighbors(metric='cosine', n_neighbors=6)
```

Recommend songs similar to a given song

In [7]:
```
song_name = "Singende Bataillone 1. Teil"
song_index = data[data['name'].str.lower() == song_name.lower()].index[0]
```

Find similar songs

In [8]:
```
distances, indices = nn.kneighbors([normalized_features[song_index]], n_neighbors=6)
```

Get indices of similar songs excluding the given song

In [9]:
```
similar_songs_indices = indices.flatten()
similar_songs_indices = similar_songs_indices[similar_songs_indices != song_index]
```

Display recommendations

In [11]:
```
top_recommendations = data.iloc[similar_songs_indices]
top_recommendations[['name', 'artists']]
```

Out[11]:

	name	artists
98219	Rostro Palido - Instrumental (Remasterizado)	['Francisco Canaro']
107610	Rostro Palido	['Francisco Canaro']
117589	S.O.S. - Remasterizado	['Francisco Canaro', 'Carlos Galán']
98367	Princesita Sevillana - Remasterizado	['Francisco Canaro', 'Charlo']
126897	Something Came and Got Me in the Spring	['Roy Fox']

Conclusion

In this project, we've implemented a recommendation system for songs using a K-Nearest Neighbors (KNN) approach. By leveraging song features and normalizing them, we've set up a system to recommend songs similar to a given track. Here's a summary of the process:

1.Data Preparation:

We started by loading the dataset and selecting the relevant numerical features for clustering. Non-numeric columns that weren't needed for similarity calculations were dropped.

2.Feature Normalization:

To ensure that all features contribute equally to the distance calculation, we normalized the numerical features using MinMaxScaler.

3.Nearest Neighbors Implementation:

We applied the NearestNeighbors algorithm with a cosine similarity metric to find similar songs. This approach efficiently identifies songs that are most similar to the provided track based on the features.

4.Recommendation:

For a specified song, we used the trained model to find and list the top similar songs, excluding the input song itself to avoid recommending the same song.

The resulting recommendations are based on feature similarity, which can be particularly useful for music discovery.

In [ ]:
```
```