Code Documentation

Full credit to all sources consulted.

1. Install/import the following python packages:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
!pip install spotipy
import spotipy
from spotipy.oauth2 import SpotifyOAuth
from spotipy.oauth2 import SpotifyClientCredentials
import os
%matplotlib inline
import spotipy.util as util
!pip install -U sentence-transformers
from sentence_transformers import SentenceTransformer
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.cluster import KMeans
from sklearn.metrics.pairwise import cosine_similarity
from sklearn.decomposition import PCA
from sklearn.preprocessing import MinMaxScaler
import difflib
```
2. Extract albums from specific artist
    a. Access environment variables, CLIENT_ID and CLIENT_SECRET using Spotify's Web Developer Account
    b. Access artist URI
    c. Loop through artist page and return all albums
3. Extract tracks from individual playlists
    a. Access CLIENT_ID and CLIENT_SECRET to access client credentials
    b. Access playlist URI
    c. Loop through playlist and extract all tracks
        i. Append track title, artist name, song popularity, genre, and album to empty track list
        ii. Extract audio features for each track in list
4. Data Preprocessing
    a. Convert playlist data into dataframe
    b. Append track names and audio features into empty array
    c. Append each entry to new dataframe as a dataframe, final_df
    d. Add URI, track_name, main artist, name, popularity, genre, and album to final_df
5. EDA

   a. Print descriptive stats
   b. Find correlation between audio features and popularity of tracks
   c. Print out other helpful visualizations
   d. Check for missing data (none in this case)
6. Feature Selection
   a. Select the following features to include in model based on feature analysis: danceability, energy, key, loudness, mode, speechiness, acousticness, instrumentalness, liveness, valence, tempo, duration_ms, time_signature, popularity
   b. Scale all feature using MinMaxScaler
7. Constructing Content-Based Recommender
   a. Locate and store indices of all tracks
   b. Normalize final_df and store as df_norm
   c. Calculate cosine similarity score between all track features
   d. Recommender function:
      i. Inputs: track_name, topmost_recs
      ii. Get cosine similarity score between inputted track and all other tracks in dataframe
      iii. Sort scores in descending order
      iv. Get top i recommendations based on topmost_recs input
      v. Return top i recommendations
8. Model Evaluation
   a. Determine model baseline: with recommender vs. without recommender
      i. Baseline: next track
      ii. Recommended results: next track recommended
      iii. Calculate recall
   b. Append top song recommendation to final_df
   c. Compare selected features of next track vs. next recommended track
      i. Store features of next track in array
      ii. Store features of next recommended track in array
      iii. Use sequence matcher to compare similarity between current track and next track vs. current track and next recommended track
      iv. If next recommended track is more similar compared to next track, store as TRUE in rec_better array
      v. Calculate recall
9. Test Model & Predict Tracks for Combined Playlists
   a. Take 3 different genre playlists, combine them, and run recommender on a song to see if it predicts something in the same playlist
      i. Jazz Study playlist: https://open.spotify.com/playlist/7jHEx7oHxN4B7PKCLapcq9?si=bf6e8c0e17404cf4

ii. Mandarin playlist: https://open.spotify.com/playlist/340MRzAs8Ro5dgGtHA8hUZ?si=44ee0e93c0424c86

iii. Meditation playlist: https://open.spotify.com/playlist/3ksy3Zso4vdt4JIzTYvpF9?si=ef7b395459424ec0

b. Join all previously selected features and tracks from all three playlists into one playlist

c. Predict the top 10 songs using one track from each playlist and note whether recommendations are from the same playlist