**IMPLEMENTATION OF MUSIC RECOMMENDATION SYSTEM**

**Name: Kavya Goudar**

**Usn: 01FM19MCA025**

**Project Title: Music Recommendation System.**

**Source code:**

from spotipy.oauth2 import SpotifyClientCredentials

from collections import defaultdict

from spotipy.oauth2 import SpotifyOAuth

sp = spotipy.Spotify(auth\_manager=SpotifyClientCredentials(client\_id="cf0ed68e536848da870c979cdcf39ac7",

                                                        client\_secret="9e244b94ec3e4551a03be2a55278a4f4"))

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)

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')

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}],  spotify\_data)