**Song Recommendation System**

#### ***Motivation***

In this world full of stressful people trying to get some peace and entertainment, I’m trying to help them by recommending some good songs they might like to listen based on their favorite song and artist. I know there are much more serious problems we may look into but I think taking care of an individual mental health is also a very important thing to do. So I selected this model to implement. Music has this magic of letting our spirits shine. Hoping this would help someone to find new music to find peace and clear their heads from the mess they bearing.

#### ***Problem Statement***

For a given song name, artist and genre -a set of similar songs have to be generated.

#### ***Approach***

By doing descriptive analysis on suitable dataset we develop this model. For this problem I selected two datasets called spotify\_dataset.csv and top10.xlsx.

#### ***About datasets***

The spotify\_dataset has columns 'Highest Charting Position', 'Song Name', 'Streams', 'Artist', 'Artist Followers', 'Genre', 'Popularity', 'Danceability', 'Energy', 'Loudness', 'Speechiness', 'Acousticness', 'Liveness', 'Valence'. The top10 dataset has columns 'title', 'artist', 'top genre', 'year', 'dnce', 'nrgy', 'dB', 'spch', 'acous', 'live', 'val'. The values like danceability, energy , loudness, speechiness, acousticeness, liveness, valence describes the song hype and mood. Popularity, streams helps in finding all time favorite songs and songs listened by most people.

#### Dataset source

spotify\_dataset: <https://github.com/chinni1430/Dsp-project/blob/main/spotify_dataset.csv>

top10: https://github.com/chinni1430/Dsp-project/blob/main/project/top10.xlsx

#### Procedure

We take the favorite song, artist and the genre the user listens most as input. And we find similar songs by choosing the popular songs of the given artist, some random songs in the genre given by user and the songs which sounds like the above selected songs. The in-detail procedure will be explained in comments in the code below.

#### Constraints

* Works for all 2010 to 2019 released popular English songs.
* The input should be given properly.

#### Before going further,

* tofloat() method returns “ValueError:could not convert string to float.” Error while trying to format a string data type column to float. So in such case we can use df.apply(df[‘col’],errors = ‘coerce’) .The ‘coerce will convert any value error to NaN.
* Normalization is a scaling technique method in which data points are shifted and rescaled so that they end up in range of 0 and 1.
* The Minkowski distance is a metric intended for real valued vector spaces. In simple words the distance between points in multi dimensions which Is calculated using Minkowski formula is Minkowski distance.

#### Code with explanation

#Importing pandas and numpy to work with datasets and numerical.

import pandas as pd

import numpy as np

#Creating data frames from our datasets

sp = pd.read\_csv(r"C:\Users\Divya DJ\Downloads\spotify\_dataset.csv")

top = pd.read\_excel(r"C:\Users\Divya DJ\Downloads\top10.xlsx")

#Dropping the null values in the dataframmes

top.dropna(inplace = True)

sp.dropna(inplace = True)

#Splitting each data frame into two. One for numerical description of songs and other for the details of the songs.

spc = sp[['Danceability', 'Energy', 'Loudness', 'Speechiness', 'Acousticness','Liveness', 'Valence']]

topc = top[[ 'dnce', 'nrgy', 'dB','spch', 'acous', 'live', 'val']]

sp.drop(['Danceability', 'Energy', 'Loudness', 'Speechiness', 'Acousticness','Liveness', 'Valence'],axis = 1,inplace = True)

top.drop([ 'dnce', 'nrgy', 'dB','spch', 'acous', 'live', 'val'],axis = 1,inplace = True)

#Formatting the numerical descriptive set of spotify\_dataset into float as it is in string. To work with them to find similar songs they should be in numerical form.

spc = spc.apply(pd.to\_numeric,errors = 'coerce')

spc = spc[['Danceability', 'Energy', 'Loudness', 'Speechiness', 'Acousticness','Liveness', 'Valence']].astype(float)

#The numerical descriptive set of spotify had normalized values, so we are normalizing the numerical descriptive set of top10 too.

for i in topc[[ 'dnce', 'nrgy','spch', 'acous', 'live', 'val']]:

topc[i] = topc[i] / topc[i].abs().max()

#Converting all the string type data we are gonna use for comparision into lower case. And taking input from user and converting that input into lower case too for easy comparision.

sp['Song Name'] = sp['Song Name'].str.lower()

top['title'] = top['title'].str.lower()

sp['Artist'] = sp['Artist'].str.lower()

top['artist'] = top['artist'].str.lower()

song = str(input("Enter a song name you listen:"))

artist = str(input("Your favourite artist:"))

genre = str(input("Genre you like:"))

genre = genre.lower()

artist = artist.lower()

song = song.lower()

#Creating a spp i.e, spotify dataset for predicting similar songs dataset. Taking atmost 3 rows with most number of streams from data with the given artist and genre.

s = sp[sp['Artist'].str.contains(artist)].sort\_values(by = ['Streams'],ascending = False)

if(len(s.index) < 3):

spp = s.head(len(s.index))

else:

spp = s.head(3)

s = sp[sp['Genre'].str.contains(genre)].sort\_values(by = ['Streams'],ascending = False)

if(len(s.index) < 3):

spp = spp.append(s.head(len(s.index)))

else:

spp = spp.append(s.head(3))

#Creating a topp i.e, top10 dataset for predicting similar songs dataset. Taking atmost 3 rows of data with the given artist and genre.

s = top[top['artist'].str.contains(artist)]

if(len(s.index) < 3):

topp = s.sample(len(s.index))

else:

topp = s.sample(3)

s = top[top['top genre'].str.contains(genre)]

if(len(s.index) < 3):

topp = topp.append(s.sample(len(s.index)))

else:

topp = topp.append(s.sample(3))

spp = spp.sort\_values(by=['Streams'],ascending = False)

#printing error message if invalid genre and artist is given.

e = 0

if((len(spp.index) == 0) or (len(topp.index) == 0)):

print("You have to provide valid artist and genre")

e = 1

#The above two datasets will provide songs that suits the user taste 90% as they are from same artist and genre. If we didn’t found the song details on our dataset, using the above selected songs numerical description we are gonna find the similar songs they might like to listen. If we the given song details, then we use it’s numerical description to find similar songs. Also prints error message and exits the execution if song not found and error has been occurred before.

a = sp[sp['Song Name'].str.contains(song)].index

b = top[top['title'].str.contains(song)].index

spcp = []

topcp = []

if(len(a)):

spcp = list(spc.loc[a[0]])

elif(e == 0):

spcp = spc.loc[spp.index]

spcp = list(spcp.mean())

if(len(b)):

topcp = list(topc.loc[b[0]])

elif(e == 0):

topcp = topc.loc[topp.index]

topcp = list(topcp.mean())

if(len(spcp) == 0 or len(topcp) == 0):

print("Provide valid song name also!")

exit()

print(spcp,topcp)

#To find similar songs we use Minkowski distance.

def mink\_dis(a,b,p = 1):

dim = len(a)

dis = 0

for d in range(dim):

dis += abs(a[d] - b[d])\*\*p

dis = dis\*\*(1/p)

return dis

dist = []

for i in spc.index:

dist.append(mink\_dis(spcp,spc.iloc[i]))

spc\_dists = pd.DataFrame(data=dist,index=spc.index,columns=['dist'])

spc\_dists = spc\_dists.sort\_values(by=['dist'],axis = 0)[:2]

dist = []

for i in topc.index:

dist.append(mink\_dis(topcp,topc.iloc[i]))

topc\_dists = pd.DataFrame(data=dist,index=topc.index,columns=['dist'])

topc\_dists = topc\_dists.sort\_values(by=['dist'],axis = 0)[:2]

sp\_final = sp.loc[spc\_dists.index]

top\_final = top.loc[topc\_dists.index]

#Now the final sets provide the similar song details. Let’s take all these similar songs and the songs by same artist and genre with most number of streams from spotify and randomly from top10.

spp = spp.sort\_values(by=['Streams'],ascending = False)

import random

l = []

if(len(spp.index) < 3):

l.append(spp['Song Name'].head(len(spp.index)))

else:

l.append(spp['Song Name'].head(3))

if(len(topp.index ) < 3):

l.append(random.sample(sorted(topp['title']),len(topp.index)))

else:

l.append(random.sample(sorted(topp['title']),3))

l.append(list(sp\_final['Song Name'].values))

l.append(list(top\_final['title'].values))

print("songs you may like:")

for i in l:

for j in i:

print(j)

#### Example Input

Enter a song name you listen:gorgeous

Your favourite artist:zayn

Genre you like:hip hop

#### Output for above input

songs you may like:

trampoline

gnat

big body benz

omg (feat. will.i.am)

castle walls (feat. christina aguilera)

let me

trillionaire (feat. youngboy never broke again)

nicht verdient

chained to the rhythm

don't let me down

#### About US

We are G.VANDHANA,G.CHINNI from CSE 2E holding ID number S181018,s181010 studying in RGUKT IIIT SRIKAKULAM. Hoping that this project helps you find some good music.

CONTACT:

[s181018@rguktsklm.ac.in](mailto:s181018@rguktsklm.ac.in), s181010@rguktsklm.ac.in

Download the source code from: <https://github.com/chinni1430/Dsp-project/blob/main/Song%20recommendation%20system.ipynb>