

**Report On Project of Music Recommendation System Using Python**

# INT-423

Submitted By : **G. Gopi Krishna**

Reg No : **12115851**

Section : **KM025**

Submitted To : **Mr. Pranjal** ( Staff Id : **30426** )

# Introduction

A recommendation system plays a major role in providing a good user experience in an application by recommending the most suitable and personalized services for each user.

Recommendation system uses **Collaborative filtering**to recommend songs and podcasts to users. Collaborative filtering recommends products or services by finding similarities between users and the products or services to provide a better user experience.

**Filtering by collaboration:**

By gathering preferences from numerous users (collaborating), Collaborative Filtering (CF) is a recommendation method that automatically predicts (filters) a user's preferences. It is predicated on the notion that customers who have previously agreed will do so again and will continue to favor similar goods.

**Filtering by content:**

Users are given recommendations for products using content-based filtering based on the characteristics or features of the products and a user profile.

**Hybrid Approaches:**

For more thorough and precise recommendations, hybrid approaches integrate elements of both collaborative filtering and content-based filtering. This can be achieved by combining the predictions from content-based and collaborative models independently, or by combining the two strategies into a single model.

# Dataset Used

I will be using a dataset that has been collected from Spotify. The dataset contains over **174,000+** songs with over **19** features grouped by artist, year and genre, etc..

# Libraries Used

* **NumPy: -** NumPy is a library for the Python programming language, adding support for large, multi-dimensional arrays and matrices, along with a large collection of high-level mathematical functions to operate on these arrays.
* **Pandas:** Pandas is a Python package providing fast, flexible, and expressive data structures designed to make working with “relational” or “labelled” data both easy and intuitive. It aims to be the fundamental high-level building block for doing practical, real-world data analysis in Python.
* **Matplotlib:** Itis a comprehensive library for creating static, animated, and interactive visualizations in Python. Matplotlib makes easy things easy and hard things possible.
* **Seaborn:** Seaborn is a Python data visualization library based on . It provides a high-level interface for drawing attractive and informative statistical graphics.
* **TQDM:** tqdm is a library in Python which is **used for creating Progress Meters or Progress Bars**. tqdm got its name from the Arabic name taqaddum which means ‘pro’

**Importing :-**

**#Importing the necessary libraries**

import warnings

import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

import seaborn as sns

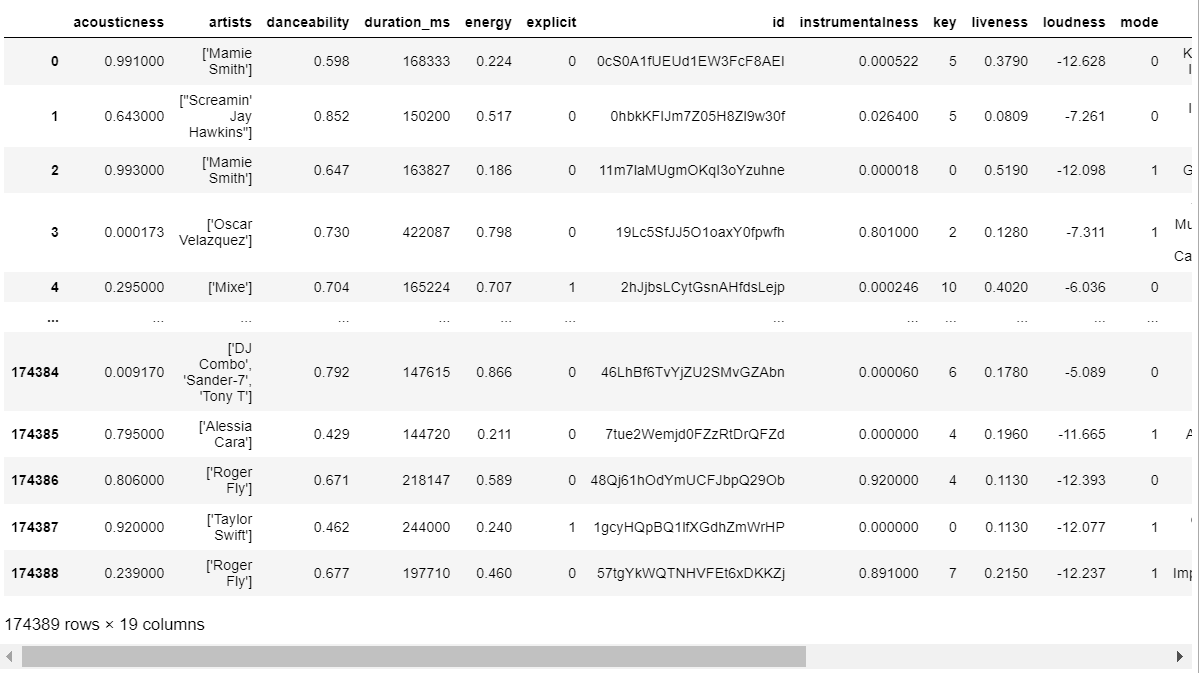
from tqdm import tqdm

sns.set()

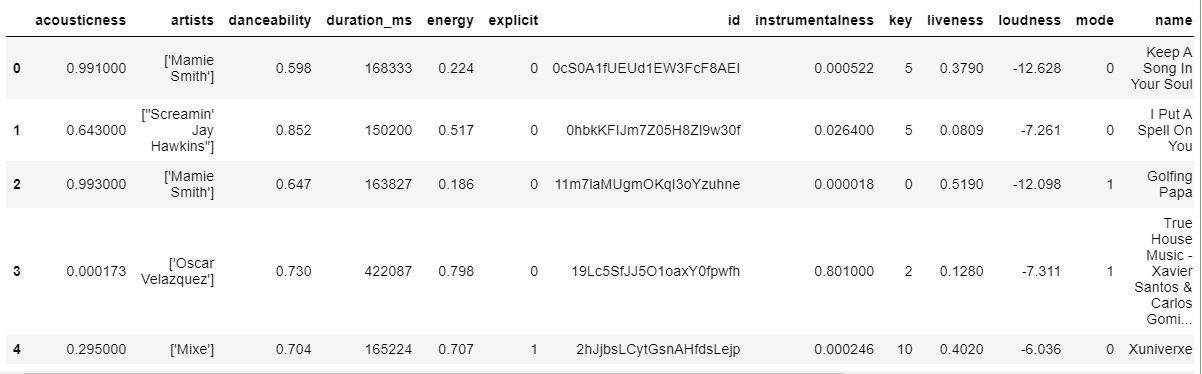
**#Reading an CSV file using pandas**

dataset = pd.read\_csv("dataset.csv")

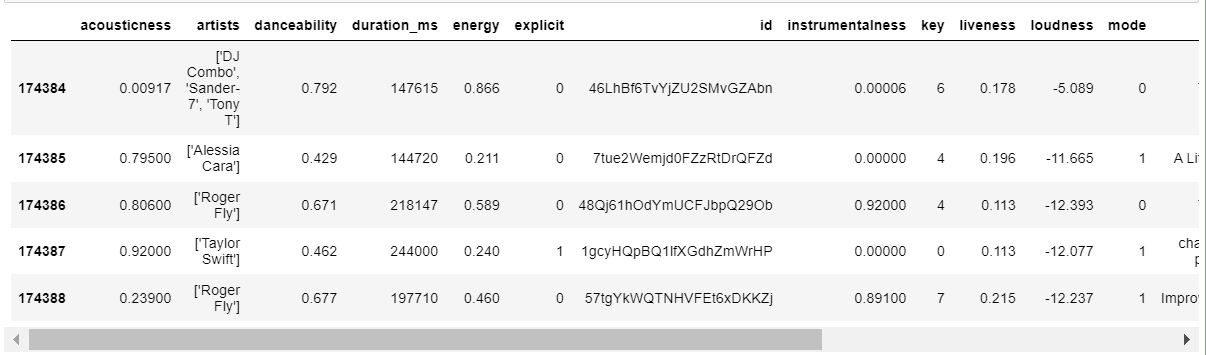
dataset



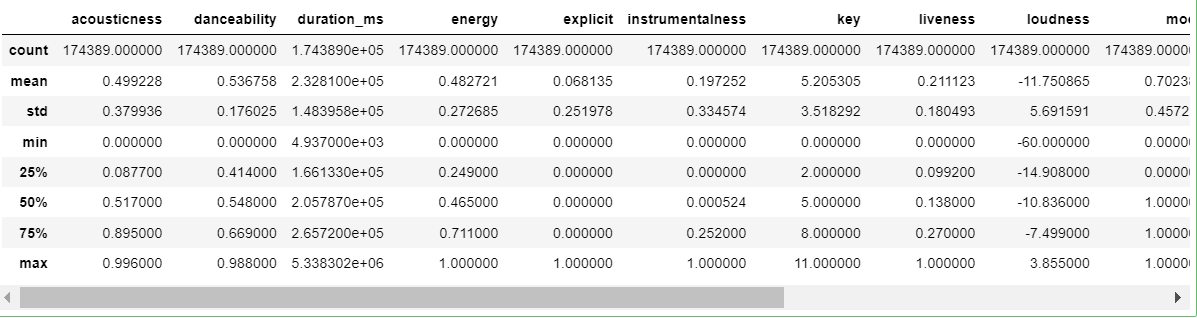
dataset.head() #prints the top 5 rows of the dataset



dataset.tail() #prints the last 5 rows of the dataset

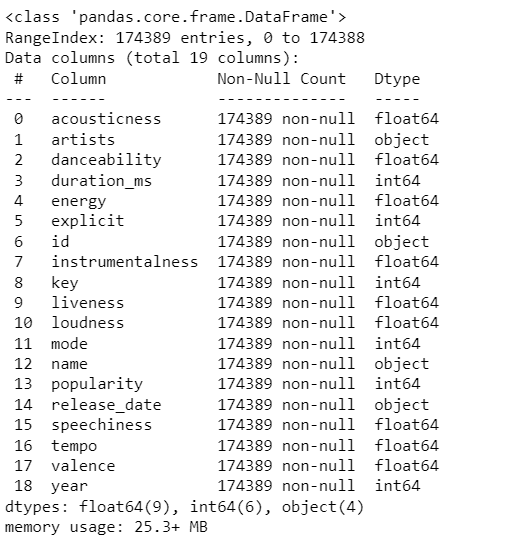


dataset.describe() #prints the statistics summary of the given dataset



dataset.info()

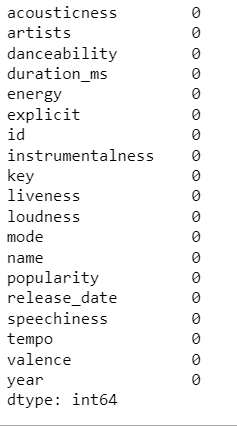
#It will gives the information of the dataset like col names, data types, non null count.



dataset.shape #prints (rows, columns) of the given dataset.

**Output** : (174389, 19)

dataset.isnull().sum() # gives you the count of missing values for each column



dataset.columns #prints the total column names in the dataset.

**Output :**

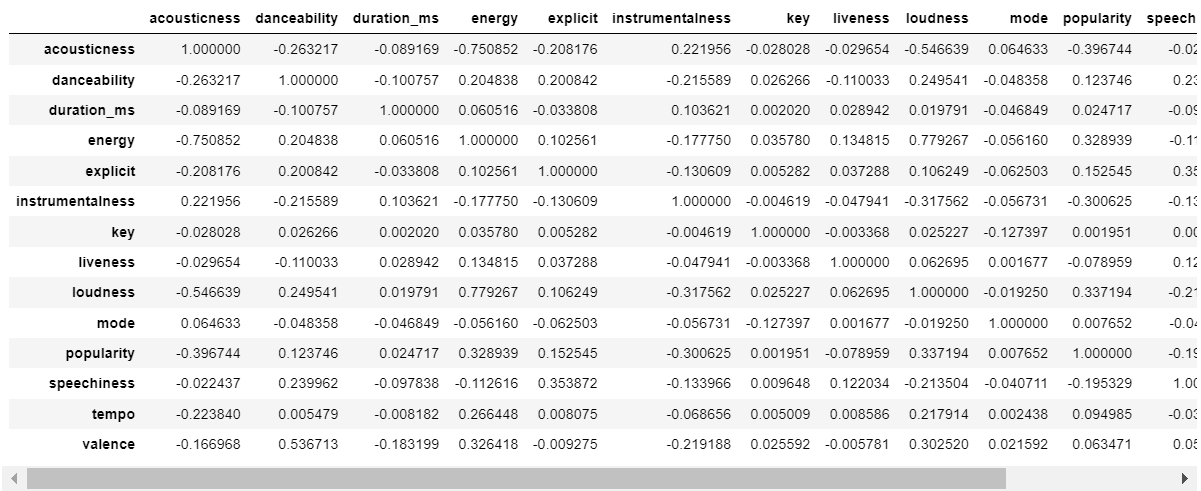
Index(['acousticness', 'artists', 'danceability', 'duration\_ms', 'energy', 'explicit', 'id', 'instrumentalness', 'key', 'liveness', 'loudness', 'mode', 'name', 'popularity', 'release\_date', 'speechiness', 'tempo', 'valence', 'year'],dtype='object')

df = dataset.drop(columns=['id', 'name', 'artists', 'release\_date', 'year'])

# dropping the unwanted columns from the dataset

df.corr()

# To find the pairwise correlation of all the columns in the DataFrame.



**Data Transformation**

from sklearn.preprocessing import MinMaxScaler

datatypes = ['int16', 'int32', 'int64', 'float16', 'float32', 'float64']

normarization = dataset.select\_dtypes(include=datatypes)

for col in normarization.columns:

MinMaxScaler(col)

**# I will now use the Python Scikit-learn library's MinMaxScaler method to normalize the dataset. Here, I'll normalize every numerical column, and to do that, I'll choose every column with the int and float datatypes.**

* Songs from many genres could share traits, which could have an impact on the recommendation engine. As a result, I'm going to introduce a brand-new function that will distinguish songs from various genres.
* I'll apply the **K means clustering** technique to this task:

from sklearn.cluster import Kmeans

#import the library for Kmeans

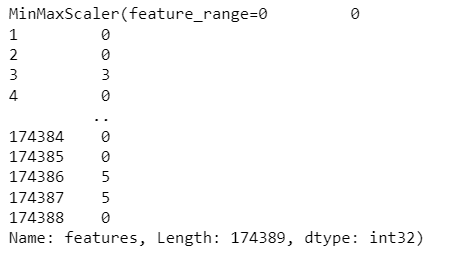
kmeans = KMeans(n\_clusters=10)

features = kmeans.fit\_predict(normarization)

dataset['features'] = features

MinMaxScaler(dataset['features'])

* The **fit\_predict** method of k-means is used to fit the model on the normalized data and obtain the cluster labels
* The KMeans class is initialized with the desired number of clusters (n\_clusters).



**Creating an Class:**

class music\_Recommendation\_System():

def \_\_init\_\_(self, dataset):

self.dataset = dataset

#this is the main function to recommend the songs.

def recommend(self, songs, amount=1):

distance = []

song = self.dataset[(self.dataset.name.str.lower() == songs.lower())].head(1).values[0]

rec = self.dataset[self.dataset.name.str.lower() != songs.lower()]

for songs in tqdm(rec.values):

d = 0

for col in np.arange(len(rec.columns)):

if not col in [1, 6, 12, 14, 18]:

d = d + np.absolute(float(song[col]) - float(songs[col]))

distance.append(d)

rec['distance'] = distance

rec = rec.sort\_values('distance')

columns = ['artists', 'name']

return rec[columns][:amount]

# append : its appends the end of the list/array.

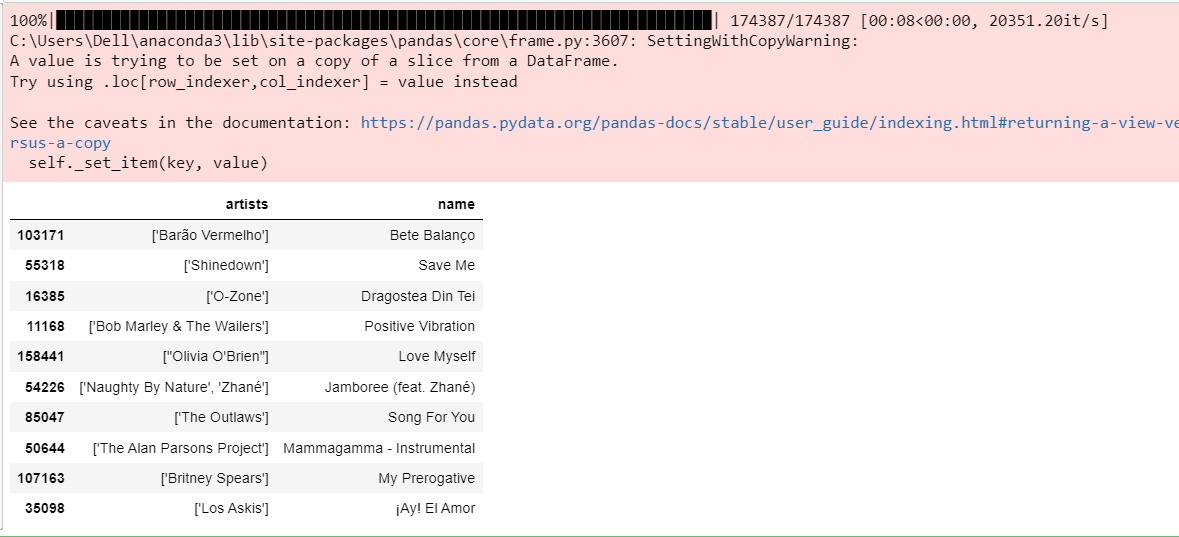
# len() : pints the length of array/list.

**Created an Object for the above Class :**

recommendations = music\_Recommendation\_System(dataset)

recommendations.recommend("Lovers Rock", 10)

# 10 represents no .of songs to recommend from the popular songs.



# Conclusion and Future Scope

The goal of recommendation is to broaden a listener's musical horizons beyond what they already know and enjoy. Once they have used up all of their song/artist search options, it gives listeners more navigational flexibility. Even before the digital revolution, music services relied on multiple points of entry into the music catalogue: filter by genres, decades, hits, new releases, and what's trending; by curators/influencers; playlists by context (moods/activities); and provided means for sharing content and playlists. In contrast to other forms of creative content (movies, novels, etc.), a song is a 3-minute experience, and the decision of what to listen to next keeps coming up. As a result, the album's historical format, which offers a minimum acceptable length, and its artistic goal.

Everyday life presents people with a variety of opportunities to listen to music while doing something else, such as while traveling, eating, exercising, or socializing with others (at a party, for example). Their hearing is available to listen to music in settings where their sight and hands may be occupied with another activity. Additionally useful, music can immediately increase activity. Listeners can experience extensive navigation even in contexts where interaction with screens is constrained, to the point where it encourages designers to build a zero interface with no room for interaction. Digital offers greater granularity (the ability to offer various playlist formats, similar artists, and songs), a higher frequency of selection updates, a much deeper dive into the archive, and personalization (tailored recommendations/playlists/UI to each listener).

## Project Github Link :-

https://github.com/gopi76/Music-Recommendation-System-