LOVELY PROFESSIONAL UNIVERSITY

Report On Project of Music Recommendation System Using Python

INT-254

Submitted By : G. Gopi Krishna

Reg No : 12115851

Section : KM119

Submitted To : Mr. Imran Hussain

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.

Dataset Used

I will be using a dataset that has been collected from Spotify. The dataset contains over 175,000 songs with over 19 features grouped by artist, year and genre.

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:** It is 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'

Import :-

```
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()
```

```
data = pd.read_csv("spotify.csv")
```

```
data
```

4	0.295000	['Mixe']	0.704	165224	0.707	1	2hJjbsLCytGsnAHfdsLejp	0.000246	10	0.4020	-6.036	0
...	...	...	...	...	...	...	...	...	...	...	...	...
174384	0.009170	['DJ Combo', 'Sander-7', 'Tony T']	0.792	147615	0.866	0	46LhBf6TvYjZU2SMvGZAbn	0.000060	6	0.1780	-5.089	0
174385	0.795000	['Alessia Cara']	0.429	144720	0.211	0	7tue2Wemjd0FZzRtDrQFZd	0.000000	4	0.1960	-11.665	1
174386	0.806000	['Roger Fly']	0.671	218147	0.589	0	48Qj61hOdYmUCFJbpQ29Ob	0.920000	4	0.1130	-12.393	0
174387	0.920000	['Taylor Swift']	0.462	244000	0.240	1	1gcyHQpBQ1IfXGdhZmWrHP	0.000000	0	0.1130	-12.077	1
174388	0.239000	['Roger Fly']	0.677	197710	0.460	0	57tgYkWQTNHVFEt6xDKKZj	0.891000	7	0.2150	-12.237	1

174389 rows × 19 columns

```
data.head()
```

	acousticness	artists	danceability	duration_ms	energy	explicit	id	instrumentalness	key	liveness	loudness	mode	name
0	0.991000	['Mamie Smith']	0.598	168333	0.224	0	0cS0A1fUEUd1EW3FcF8AEI	0.000522	5	0.3790	-12.628	0	Keep A Song In Your Soul
1	0.643000	["Screamin' Jay Hawkins"]	0.852	150200	0.517	0	0hbkKFIJm7Z05H8ZI9w30f	0.026400	5	0.0809	-7.261	0	I Put A Spell On You
2	0.993000	['Mamie Smith']	0.647	163827	0.186	0	11m7laMUgmOKqI3oYzuhne	0.000018	0	0.5190	-12.098	1	Golfing Papa
3	0.000173	['Oscar Velazquez']	0.730	422087	0.798	0	19Lc5SfJJ5O1oaxY0fpwfh	0.801000	2	0.1280	-7.311	1	True House Music - Xavier Santos & Carlos Gomi...
4	0.295000	['Mixe']	0.704	165224	0.707	1	2hJjbsLCytGsnAHfdsLejp	0.000246	10	0.4020	-6.036	0	Xuniverxe

```
data.shape
```

Output :- (174389, 19)

data.info()

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 174389 entries, 0 to 174388
Data columns (total 19 columns):
 #   Column            Non-Null Count   Dtype
---  ------            --------------   -----
 0   acousticness      174389 non-null  float64
 1   artists           174389 non-null  object
 2   danceability      174389 non-null  float64
 3   duration_ms       174389 non-null  int64
 4   energy            174389 non-null  float64
 5   explicit          174389 non-null  int64
 6   id                174389 non-null  object
 7   instrumentalness  174389 non-null  float64
 8   key               174389 non-null  int64
 9   liveness          174389 non-null  float64
 10  loudness          174389 non-null  float64
 11  mode              174389 non-null  int64
 12  name              174389 non-null  object
 13  popularity        174389 non-null  int64
 14  release_date      174389 non-null  object
 15  speechiness       174389 non-null  float64
 16  tempo             174389 non-null  float64
 17  valence           174389 non-null  float64
 18  year              174389 non-null  int64
dtypes: float64(9), int64(6), object(4)
memory usage: 25.3+ MB
```

data.isnull().sum()

```
acousticness          0
artists               0
danceability          0
duration_ms           0
energy                0
explicit              0
id                    0
instrumentalness      0
key                   0
liveness              0
loudness              0
mode                  0
name                  0
popularity            0
release_date          0
speechiness           0
tempo                 0
valence               0
year                  0
dtype: int64
```

df = data.drop(columns=['id', 'name', 'artists', 'release_date', 'year'])

df.corr()

corr():-this function to find the correlation among the columns in the Dataframe using the ‘Pearson’ method.

	acousticness	danceability	duration_ms	energy	explicit	instrumentalness	key	liveness	loudness	mode	popularity	speech
acousticness	1.000000	-0.263217	-0.089169	-0.750852	-0.208176	0.221956	-0.028028	-0.029654	-0.546639	0.064633	-0.396744	-0.0
danceability	-0.263217	1.000000	-0.100757	0.204838	0.200842	-0.215589	0.026266	-0.110033	0.249541	-0.048358	0.123746	0.2
duration_ms	-0.089169	-0.100757	1.000000	0.060516	-0.033808	0.103621	0.002020	0.028942	0.019791	-0.046849	0.024717	-0.0
energy	-0.750852	0.204838	0.060516	1.000000	0.102561	-0.177750	0.035780	0.134815	0.779267	-0.056160	0.328939	-0.1
explicit	-0.208176	0.200842	-0.033808	0.102561	1.000000	-0.130609	0.005282	0.037288	0.106249	-0.062503	0.152545	0.3
instrumentalness	0.221956	-0.215589	0.103621	-0.177750	-0.130609	1.000000	-0.004619	-0.047941	-0.317562	-0.056731	-0.300625	-0.1
key	-0.028028	0.026266	0.002020	0.035780	0.005282	-0.004619	1.000000	-0.003368	0.025227	-0.127397	0.001951	0.0
liveness	-0.029654	-0.110033	0.028942	0.134815	0.037288	-0.047941	-0.003368	1.000000	0.062695	0.001677	-0.078959	0.1
loudness	-0.546639	0.249541	0.019791	0.779267	0.106249	-0.317562	0.025227	0.062695	1.000000	-0.019250	0.337194	-0.2
mode	0.064633	-0.048358	-0.046849	-0.056160	-0.062503	-0.056731	-0.127397	0.001677	-0.019250	1.000000	0.007652	-0.0
popularity	-0.396744	0.123746	0.024717	0.328939	0.152545	-0.300625	0.001951	-0.078959	0.337194	0.007652	1.000000	-0.1
speechiness	-0.022437	0.239962	-0.097838	-0.112616	0.353872	-0.133966	0.009648	0.122034	-0.213504	-0.040711	-0.195329	1.0
tempo	-0.223840	0.005479	-0.008182	0.266448	0.008075	-0.068656	0.005009	0.008586	0.217914	0.002438	0.094985	-0.0
valence	-0.166968	0.536713	-0.183199	0.326418	-0.009275	-0.219188	0.025592	-0.005781	0.302520	0.021592	0.063471	0.0

```
from sklearn.preprocessing import MinMaxScaler
datatypes = ['int16', 'int32', 'int64', 'float16', 'float32', 'float64']
normarization = data.select_dtypes(include=datatypes)
for col in normarization.columns:
    MinMaxScaler(col)
```

Created a class:-

```
class Spotify_Recommendation():
  def __init__(self, dataset):
    self.dataset = dataset
  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]
```

lower() method returns a string where all characters are **lower** case.

Created an object for class:-

recommendations = Spotify_Recommendation(data)

recommendations.recommend("Lovers Rock", 10)

```
100%|██████████████████████████████████████████████████████████████| 174387/174387 [00:11<00:00, 14788.83it/s]
C:\Users\Dell\anaconda3\lib\site-packages\pandas\core\frame.py:3607: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  self._set_item(key, value)
```

	artists	name
103171	['Barão Vermelho']	Bete Balanço
55318	['Shinedown']	Save Me
16385	['O-Zone']	Dragostea Din Tei
11168	['Bob Marley & The Wailers']	Positive Vibration
158441	["Olivia O'Brien"]	Love Myself
54226	['Naughty By Nature', 'Zhané']	Jamboree (feat. Zhané)
85047	['The Outlaws']	Song For You
50644	['The Alan Parsons Project']	Mammagamma - Instrumental
107163	['Britney Spears']	My Prerogative
35098	['Los Askis']	¡Ay! El Amor

Conclusion and Future Scope

Recommendation is about extending listeners music universe beyond what they know and like. It empowers listeners once they have exhausted all their songs/artists search capabilities with further navigation celerity. Music services, even before the digital revolution, have been relying on several points of entry in the music catalogue: filter by genres, decades, selections of hits, of new releases/what's trending, by curators/influencers, playlists by context (moods/activities), and provided means for sharing content and playlists.

A song is a 3 minute experience, and the question of what to listen next keeps coming back, contrary to other creative contents (movies, books,...). Hence the historic format of the album, which provides a minimum acceptable duration, along its artistic intention.

People in their day to day life encounter many situations where music can be listened to while doing something else: in transport (cars, traveling,...), while eating, doing sport...or with other people (party,...). In those situations their sight and hands may be busy doing another activity, their hearing is available to listen to music. Music can also be more functional and directly stimulate the activity (dance, yoga, cheering up,...).

The digital revolution provides listeners with devices, apps and algorithms that allow to better capture those listening situation opportunities, and to adapt to each context: rich UI on PC, simplified UI on mobile phones and in cars, voice control in the car and smart speakers.

Even during situations where interaction with screens is limited, listeners can enjoy rich navigation, to the point of inviting designers to create a zero interface where no interaction is possible. Digital brings higher granularity (the possibility to provide multiple types of playlists, similar artists and songs), higher frequency of updates of selections, a much deeper dive in the catalogue, and personalization (personalized recommendations/playlists/UI to each listener).

Project Github Link :-

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