Code Implementation for Muisc Recommendation

Complete Implementation

Part 2

21BCE9336 Santosh Babu Donga

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sb
from sklearn.manifold import TSNE
from sklearn.metrics.pairwise import cosine_similarity

import warnings
warnings.filterwarnings('ignore')
tracks = pd.read_csv('dataset.csv')
tracks.head()
```

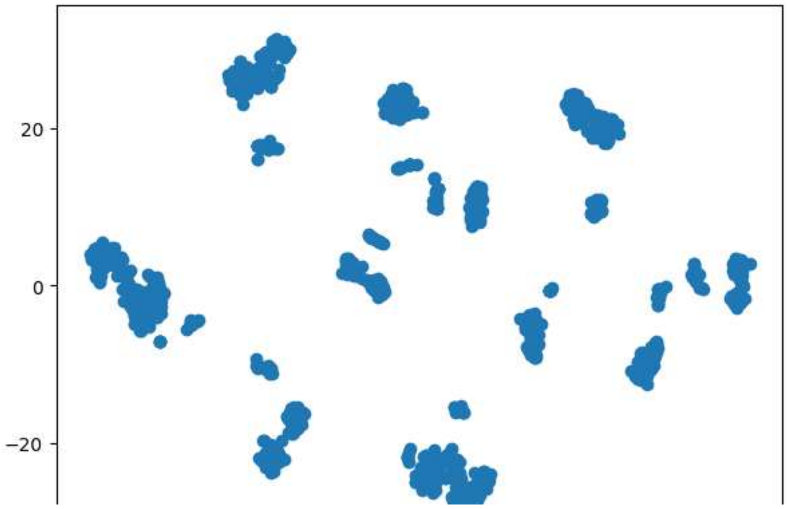
	Unnamed: 0	track_id	artists	album_name	track_name	popul
0	0	5SuOikwiRyPMVoIQDJUgSV	Gen Hoshino	Comedy	Comedy	
1	1	4qPNDBW1i3p13qLCt0Ki3A	Ben Woodward	Ghost (Acoustic)	Ghost - Acoustic	
2	2	1iJBSr7s7jYXzM8EGcbK5b	Ingrid Michaelson;ZAYN	To Begin Again	To Begin Again	
3	3	6Ifxq3CG4xtTiEg7opyCyx	Kina Grannis	Crazy Rich Asians (Original Motion Picture Sou...	Can't Help Falling In Love	
4	4	5vjLSffimiIP26QG5WcN2K	Chord Overstreet	Hold On	Hold On	

5 rows × 21 columns

```
tracks = pd.read_csv('dataset.csv')
tracks.dropna(inplace=True)
tracks=tracks.drop(['track_id','explicit','Unnamed: 0'],axis=1)
```

```
a=tracks.copy()
a['duration_ms']=a['duration_ms']/1000000
a['popularity']=a['popularity']/10000
a['loudness']=a['loudness']*(-1)
a['loudness']=a['loudness']/100
a['tempo']=a['tempo']/1000
a['time_signature']=a['time_signature']/10
a=a.drop(['artists','album_name','track_name','track_genre'],axis=1)
```

```
model = TSNE(n_components = 2, random_state = 0)
tsne_data = model.fit_transform(a.head(1000))
plt.figure(figsize = (7, 7))
plt.scatter(tsne_data[:,0], tsne_data[:,1])
plt.show()
```



```
tracks=tracks.sort_values(by=['popularity'],ascending=False).head(10000)
```

```
tracks.drop_duplicates(subset=['track_name'],keep='first',inplace=True)
```

```
floats = []
for col in a.columns:
  if a[col].dtype == 'float':
    floats.append(col)

len(floats)
```

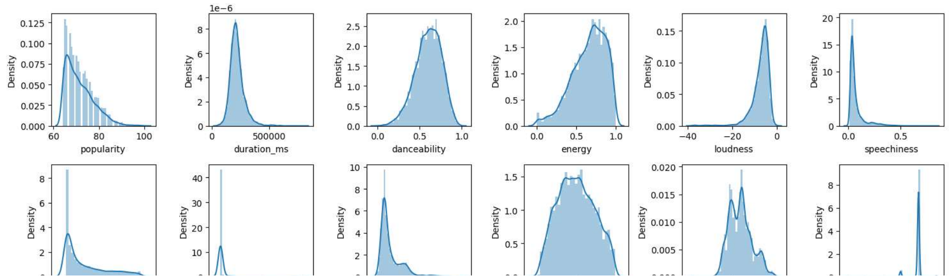
```
12
```

```
a
```

	popularity	duration_ms	danceability	energy	key	loudness	mode	speechiness	acou
0	0.0073	0.230666	0.676	0.4610	1	0.06746	0	0.1430	
1	0.0055	0.149610	0.420	0.1660	1	0.17235	1	0.0763	
2	0.0057	0.210826	0.438	0.3590	0	0.09734	1	0.0557	
3	0.0071	0.201933	0.266	0.0596	0	0.18515	1	0.0363	
4	0.0082	0.198853	0.618	0.4430	2	0.09681	1	0.0526	
...	...	...	...	...	...	...	...	...	
113995	0.0021	0.384999	0.172	0.2350	5	0.16393	1	0.0422	
113996	0.0022	0.385000	0.174	0.1170	0	0.18318	0	0.0401	
113997	0.0022	0.271466	0.629	0.3290	0	0.10895	0	0.0420	
113998	0.0041	0.283893	0.587	0.5060	7	0.10889	1	0.0297	
113999	0.0022	0.241826	0.526	0.4870	1	0.10204	0	0.0725	

113999 rows × 14 columns

```
plt.subplots(2,6,figsize = (15, 5))
for i, col in enumerate(floats):
  plt.subplot(2, 6, i+1)
  sb.distplot(tracks[col])
plt.tight_layout()
plt.show()
```



```
song_vectorizer = CountVectorizer()
song_vectorizer.fit(tracks['track_genre'])
```

```
▾ CountVectorizer
CountVectorizer()
```

```
tracks
```

	artists	album_name	track_name	popularity	duration_ms	danceability	ene
20001	Sam Smith;Kim Petras	Unholy (feat. Kim Petras)	Unholy (feat. Kim Petras)	100	156943	0.714	0.4
81051	Sam Smith;Kim Petras	Unholy (feat. Kim Petras)	Unholy (feat. Kim Petras)	100	156943	0.714	0.4
51664	Bizarrap;Quevedo	Quevedo: Bzrp Music Sessions, Vol. 52	Quevedo: Bzrp Music Sessions, Vol. 52	99	198937	0.621	0.7
89411	Manuel Turizo	La Bachata	La Bachata	98	162637	0.835	0.6
81210	David Guetta;Bebe Rexha	I'm Good (Blue)	I'm Good (Blue)	98	175238	0.561	0.9
...	...	...	...	...	...	...	
101255	Juhani Ahonen	Tauko	Tauko	64	166497	0.226	0.0
3206	Qveen Herby	EP 8	Sugar Daddy	64	203809	0.953	0.4
31678	graves;Tim Gunter;LocateEmilio	Blame (Tim Gunter Remix)	Blame (Tim Gunter Remix)	64	193714	0.566	0.7
32274	Valentino Khan;Dillon Francis	Move It	Move It	64	201259	0.796	0.8
11645	Bombay Bicycle Club	I Had The Blues But I Shook Them Loose	Always Like This	64	245640	0.659	0.5

10000 rows × 18 columns

```
def get_similarities(song_name, data):
    text_array1 = song_vectorizer.transform(data[data['track_name'] == song_name]['track_genre']).toarray()
    num_array1 = data[data['track_name'] == song_name].select_dtypes(include=np.number).to_numpy()

    # We will store similarity for each row of the dataset.
    sim = []
    for idx, row in data.iterrows():
        name = row['track_name']

        # Getting vector for the current song.
        text_array2 = song_vectorizer.transform(data[data['track_name'] == name]['track_genre']).toarray()
        num_array2 = data[data['track_name'] == name].select_dtypes(include=np.number).to_numpy()

        # Calculating similarities for text as well as numeric features
        text_sim = cosine_similarity(text_array1, text_array2)[0][0]
        num_sim = cosine_similarity(num_array1, num_array2)[0][0]
        sim.append(text_sim + num_sim)
    return sim


def recommend_songs(song_name, data=tracks):
    # Base case
    if data[data['track_name'] == song_name].shape[0] == 0:
        print("This song is either not so popular or you have entered an invalid name.\n Some songs you may like:\n")

        for song in data.sample(n=5)['track_name'].values:
            print(song)
        return

    data['similarity_factor'] = get_similarities(song_name, data)

    data.sort_values(by=['similarity_factor', 'popularity'],
                     ascending=[False, False],
                     inplace=True)

    # First song will be the input song itself as the similarity will be highest.


    input_song_row = data[data['track_name'] == song_name][['track_name', 'artists']]
    print(input_song_row)
    display(data[['track_name', 'artists']].iloc[1:6])
```

```
recommend_songs('Solo')
```

	track_name	artists
103256	All I Ask	Adele
11002	All I Ask	Adele
103901	Time Moves Slow	BADBADNOTGOOD;Samuel T. Herring
64650	Time Moves Slow	BADBADNOTGOOD;Samuel T. Herring
103909	Biking	Frank Ocean;JAY-Z;Tyler, The Creator