application

December 11, 2022

[ ]:
```
# In this Python journal, I created the framework to get recommended songs from␣
 ↪the songs csv dataset.
# At the end, a demonstration of this recommendation is done with a song on␣
 ↪Spotify.
```

[1]:
```
import pandas as pd
import authorization
```

[ ]:
```
# The distance function below calculates the distance between two songs'␣
 ↪valence and energy. It ensures that songs
#with a similar valence and energy would be recommended for the song that is␣
 ↪inputted.
```

[2]:
```
def distance(p1, p2):
    distance_x = p2[0]-p1[0]
    distance_y = p2[1]-p1[1]
    distance_vec = [distance_x, distance_y]
    norm = (distance_vec[0]**2 + distance_vec[1]**2)**(1/2)
    return norm
```

[ ]:
```
#In the code below, I read the data from the music csv file and returned how␣
 ↪many songs are in the file.
```

[3]:
```
df = pd.read_csv("music.csv")
df["mood_vec"] = df[["valence", "energy"]].values.tolist()
len(df)
```

[3]: 6299

sp = authorization.authorize()

[ ]:
```
#In the recommend function, a Spotify track_id is inputted, and the function␣
 ↪returns the recommended songs from the
#songs csv file. The songs are recommended using the distance function above in␣
 ↪which it looks at the songs that are
#closest in energy and valence. By doing so, songs would be recommended based␣
 ↪on whether they have a similar mood to
```

```
#the inputted song.
```

[5]:
```
def recommend(track_id, sp, n_recs, genres):

    track_features = sp.track_audio_features(track_id)
    track_moodvec = [track_features.valence, track_features.energy]
    print(track_moodvec)

    ref_df = df.loc[df["genre"].isin(genres)].copy()

    ref_df["distances"] = ref_df["mood_vec"].apply(lambda x:␣
 ↪distance(track_moodvec, x))

    ref_df_sorted = ref_df.sort_values(by = "distances", ascending = True)
    ref_df_sorted = ref_df_sorted[ref_df_sorted["id"] != track_id]

    return ref_df_sorted.iloc[:n_recs].drop(columns=df.columns[0],
        axis=1)
```

[ ]:
```
#In the code below, I did a demo using the Sweet Caroline song on Spotify. The␣
 ↪link for this song is below:
#https://open.spotify.com/track/62AuGbAkt8Ox2IrFFb8GKV#login

#You can see that the track id is right after the "track/" portion of the url.␣
 ↪In the function, alongside this id, I
#inputted the Spotify token, the number of songs that I wanted to be␣
 ↪recommended, and the genres that I wanted to have
#these recommended songs from. In the example below, I asked for 5 Rock songs␣
 ↪to be recommended for Sweet Caroline.
#The results were promising in which I saw that Rock songs of a similar mood␣
 ↪were recommended like Africa and Every
#Breath You Take. This is because all of these songs are classic Rock songs␣
 ↪which are generally happy and laid-back.
#This is why you see a higher valence but lower energy.
```

[6]:
```
recommend("62AuGbAkt8Ox2IrFFb8GKV", sp, 5, ["rock"])
```

```
[0.578, 0.127]
```

[6]:

	id	genre	\
4983	6i0V12jOa3mr6uu4WYhUBr	rock	
4954	2374M0fQpWi3dLnB54qaLX	rock	
4985	6VltRkmJbCTqgKrTHk4Ulw	rock	
4992	5tVA6TkbaAH9QMITTQRrNv	rock	
4961	5C0LFQARavkPpn7JgA4sLk	rock	

	track_name	artist_name	\

```
4983                                          Heathens  Twenty One Pilots
4954                                            Africa              TOTO
4985  My My, Hey Hey (Out of the Blue) - 2016 Remaster        Neil Young
4992                                       Free Fallin'         Tom Petty
4961           Every Breath You Take - Remastered 2003        The Police

      valence  energy        mood_vec  distances
4983    0.548   0.396  [0.548, 0.396]   0.270668
4954    0.732   0.373  [0.732, 0.373]   0.290227
4985    0.435   0.400    [0.435, 0.4]   0.308185
4992    0.572   0.449  [0.572, 0.449]   0.322056
4961    0.740   0.452   [0.74, 0.452]   0.363138
```

[ ]:
```
#This is the results of my project that I showed to my class. It was␣
 ↪recommended that I try to create my own
#recommender system for the songs on my computer rather than just use Spotify's␣
 ↪systems. My attempt at doing that is
#shown in the Songs Python journal.
```