

COMSATS UNIVERSITY



WAH CAMPUS

Submitted By

Name: *Shayan Babar*

Registration No: *Fa19-Bcs-051*

Class/Section: *BSCS/6D*

Submitted To

Teacher Name: *Ammara Zamir*

Date of Submission: *15-06-2022*

Dataset name: Top 50 Spotify Songs – 2019

About Data

- Check the data extracted by year: <https://www.kaggle.com/leonardopena/top-spotify-songs-from-20102019-by-year>

Context

The top 50 most listened songs in the world by spotify. This dataset has several variables about the songs.

Content

50 songs
13 variables

Description

As the name suggests the data set is about the 50 songs on Spotify by year 2019 according to their popularity most listened to.

Algorithm used

Pearson correlation coefficient

Pearson Correlation Coefficient (PCC) is one of the most popular similarity measures for Collaborative filtering recommender system, to evaluate how much two users are correlated. It is a type of correlation coefficient that represents the relationship between two variables that are measured on the same interval or ratio scale. The Pearson coefficient is a measure of the strength of the association between two continuous variables.

let's load our songs dataset into a pandas Data Frame

```
# Import Pandas

import pandas as pd

# Load Movies Metadata

metadata = pd.read_csv('top50.csv', encoding='ISO-8859-1')

# Print the first three rows

print(metadata.head(3))
```

```
"C:\Users\mians\OneDrive\Desktop\sem7\Topics in cs 1\venv\Scripts\python.exe" "C:/Users/mians/OneDrive/Desktop/sem7/Topics in cs 1/ds-assiment4/recommendor.p
Track.Name Artist.Name Genre Beats.Per.Minute Energy Danceability Loudness..dB.. Liveness Valence. Length. Acousti
0 Señorita Shawn Mendes canadian pop 117 55 76 -6 8 75 191
1 China Anuel AA reggaeton flow 105 81 79 -4 8 61 302
2 boyfriend (with Social House) Ariana Grande dance pop 190 80 40 -4 16 70 186 1
```

We only require a small subset of the fields

We can filter the dataframe down to these columns using the code below.

```
df_songs = metadata[['Track.Name', 'Beats.Per.Minute', 'Energy', 'Genre', 'Popularity']]
df_songs.head()
```

	Track.Name	Beats.Per.Minute	Energy	Genre	Popularity	Artist.Name
0	Señorita	117	55	canadian pop	79	Shawn Mendes
1	China	105	81	reggaeton flow	92	Anuel AA
2	boyfriend (with Social House)	190	80	dance pop	85	Ariana Grande
3	Beautiful People (feat. Khalid)	93	65	pop	86	Ed Sheeran
4	Goodbyes (Feat. Young Thug)	150	65	dfw rap	94	Post Malone
5	I Don't Care (with Justin Bieber)	102	68	pop	84	Ed Sheeran
6	Ransom	180	64	trap music	92	Lil Tecca
7	How Do You Sleep?	111	68	pop	90	Sam Smith
8	Old Town Road - Remix	136	62	country rap	87	Lil Nas X
9	bad guy	135	43	electropop	95	Billie Eilish

To get a handle on what are the most popular songs, we can use `groupby()` and `agg()` to calculate some summary statistics for the songs in this dataset.

Create an item matrix

The primary component of our recommendation system is a matrix. This states the quantity of units of each item present in each customer's basket. We can create this matrix easily using the `pivot_table()` function.

```
df_items = df_songs.pivot_table(index="Artist.Name",columns=['Track.Name'],
values='Popularity').fillna(0)
print(df_items.head(10))
```

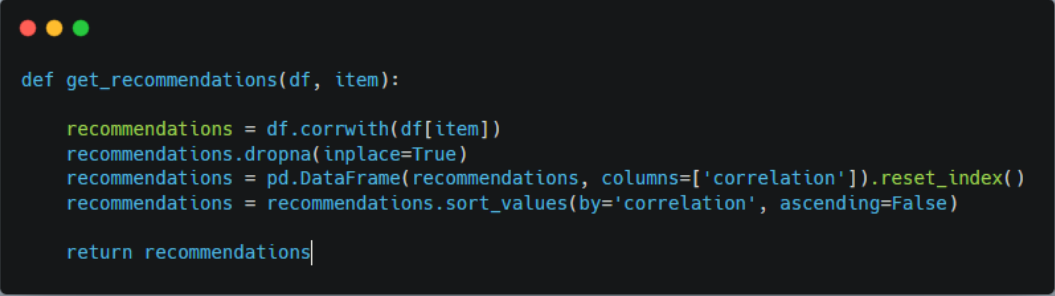
```
Track.Name    0.95833333  7 rings  Antisocial (with Travis Scott)  Beautiful People (feat. Khalid)  Call You Mine  Callaita  China  Con Altura  Con Calma
Artist.Name
Ali Gatie          0.0          0.0              0.0              0.0              0.0          0.0          0.0          0.0
Anuel AA           0.0          0.0              0.0              0.0          0.0          92.0          0.0          0.0
Ariana Grande      0.0          89.0              0.0              0.0          0.0          0.0          0.0          0.0
Bad Bunny          0.0          0.0              0.0              0.0          0.0          93.0          0.0          0.0
Billie Eilish      0.0          0.0              0.0              0.0          0.0          0.0          0.0          0.0

Track.Name    fuck, i'm lonely (with Anne-Marie) - from 13 Reasons Why: Season 3
Artist.Name
Ali Gatie          0.0
Anuel AA           0.0
Ariana Grande      0.0
Bad Bunny          0.0
Billie Eilish      0.0
```

Create Songs Recommendations

we can create a little helper function for our recommendation system to make it quick and easy to identify which songs are associated with others.

First, we use the `corrwith()` function to identify the **Pearson Correlation coefficient** for each product with every other. We then drop the `NaN` values, and place these in a dataframe sorted by descending correlation.



```
def get_recommendations(df, item):  
    recommendations = df.corrwith(df[item])  
    recommendations.dropna(inplace=True)  
    recommendations = pd.DataFrame(recommendations, columns=['correlation']).reset_index()  
    recommendations = recommendations.sort_values(by='correlation', ascending=False)  
  
    return recommendations
```

When we run the `get_recommendations()` function we will pass in our item matrix dataframe containing each product and the number of times it co-occurred in a basket, as well as the column name for our target product

Start recommendations



```
recommendations = get_recommendations(df_items, 'I Don\'t Care (with Justin Bieber)')  
print(recommendations.head())
```

	Track.Name	correlation
15	I Don't Care (with Justin Bieber)	1.000
9	Cross Me (feat. Chance the Rapper & PnB Rock)	1.000
3	Beautiful People (feat. Khalid)	1.000
2	Antisocial (with Travis Scott)	1.000
24	No Me Conoce - Remix	-0.027



```
recommendations = get_recommendations(df_items, 'Sunflower - Spider-Man: Into the Spider-Verse')  
print(recommendations.head())
```

	Track.Name	correlation
39	Sunflower - Spider-Man: Into the Spider-Verse	1.000
11	Goodbyes (Feat. Young Thug)	1.000
6	China	-0.027
21	Money In The Grave (Drake ft. Rick Ross)	-0.027
2	Antisocial (with Travis Scott)	-0.027

Code

```
# Import Pandas
import pandas as pd

pd.set_option("display.max_rows", None)
pd.set_option("display.max_columns", None)
pd.set_option("display.width", 1000)
pd.set_option("display.colheader_justify", "center")
pd.set_option("display.precision", 3)

# Load Movies Metadata
metadata = pd.read_csv("top50.csv", encoding="ISO-8859-1")

# Print the first three rows
print(metadata.head(3))

df_songs = metadata[
    ["Track.Name", "Beats.Per.Minute", "Energy", "Genre", "Popularity", "Artist.Name"]
]
print((df_songs.head(10)))

df_items = df_songs.pivot_table(
    index="Artist.Name", columns=["Track.Name"], values="Popularity"
).fillna(0)
print(df_items.head(5))

def get_recommendations(df, item):

    recommendations = df.corrwith(df[item])
    recommendations.dropna(inplace=True)
    recommendations = pd.DataFrame(
        recommendations, columns=["correlation"]
    ).reset_index()
    recommendations = recommendations.sort_values(by="correlation", ascending=False)

    return recommendations

# Create recommendations like this
recommendations = get_recommendations(df_items, "I Don't Care (with Justin Bieber)")
print(recommendations.head())

recommendations = get_recommendations(
    df_items, "Sunflower - Spider-Man: Into the Spider-Verse"
)
print(recommendations.head())
```

Conclusion

We use the pivot table and correlation coefficient to recommend songs here. If the user likes a particular song, we take that song columns and find the correlation of that column with all the other song columns and get the song that highly correlate with the chosen movie.

This works because, the rows represent users, and a particular user might like similar songs. Hence, we can use correlation coefficient to recommend songs to the users.

One drawback with collaborative filtering is that it suffers from the “cold start problem”.

Recommender systems - whether they're using content based, item based, or user-based filtering methods - all have one requirement in common: their underlying algorithms require a good amount of information for them to generate a relevant product recommendation.