CSE3024- Web Mining

DA-II - Final Report

MUSIC RECOMMENDATION SYSTEM

By

Reg. No: 20BCE1355	Name: VISHAL SANKAR
Reg. No: 20BCE1840	Name: KARTHIC S
Reg. No: 20BCE1855	Name: SANJIL K C

B.Tech CSE

Submitted to

Dr.A.Bhuvaneswari,

Assistant Professor Senior,

SCOPE, VIT, Chennai

School of Computer Science and Engineering

April 2023

VIT®

Vellore Institute of Technology

(Deemed to be University under section 3 of UGC Act, 1956)

BONAFIDE CERTIFICATE

Certified that this project report entitled "Music Recommendation System" is a bonafide work of Vishal Sankar 20BCE1355, Karthic S 20BCE1840, Sanjil K C 20BCE1855, who carried out the J-component under my supervision and guidance. The contents of this Project work, in fullor in parts, have neither been taken from any other source nor have been submitted to any other Institute or University for award of any degree or diploma and the same is certified

Dr.A.Bhuvaneswari,

Assistant Professor Senior,

SCOPE, VIT, Chennai

ABSTRACT

Recommendation systems play a vital role in our day to day lives whether it may provide recommendations on which movie to watch or which item to buy. In this project, we are going to design, implement and analyze a music recommendation system. The dataset that we are using is the Million Song Dataset provided by Kaggle. We are trying to find correlations between users and songs and to learn from the previous listening history of users to provide recommendations for songs which users would prefer to listen most in future. We are planning to implement various algorithms such as popularity based model, memory based collaborative filtering, and content based model using k-NN. In the popularity based model, we find the popularity of each song by looking into the number of users listened to that song. Users are recommended with the top most popular songs and this does not involve personal interests. Collaborative filtering (CF) uses the numerical reviews given by the user andis mainly based upon the historical data of the user available to the system. The historical data available helps to build the user profile and the data available about the item is used to make the item profile. Both the user profile and the item profile are used to make a recommendation system. Collaborative filtering is considered the most basic and the easiest method to find recommendations and make predictions regarding the sales of a product. In the item-based model, it is assumed that songs that are often listened together by some users tend to be similar and are more likely to be listened together in future also by some other user. According to the user-based similarity model, users who have similar listening histories, i.e., have listened to the same songs in the past tend to have similar interests and will probably listen to the same songs in future too. In the KNN method, we create a space of songs according to their features from the data and find out neighborhood of each song. We choose some of the available features (e.g., loudness, genre, mode, etc.) which we find most relevant to distinguish a song from others. After creating the feature space, to recommend songs to the users, we look at each users profile and suggest songs which are neighbors to the songs present in his listening history.

ACKNOWLEDGEMENT

We wish to express our sincere thanks and deep sense of gratitude to our project guide, **Dr. A. Bhuvaneswari Assistant Professor**, School of Computer Science Engineering, for his consistent encouragement and valuable guidance offered to us in a pleasant manner throughout the course of the project work.

We express our thanks to our **HOD Dr. P.Nithyanandam**., for his support throughout the course of this project.

We also take this opportunity to thank all the faculty of the School for their support and their wisdom imparted to us throughout the course.
We thank our parents, family, and friends for bearing with us throughout the course of our project and for the opportunity they provided us in undergoing this course in such a prestigious institution.

Vishal Sankar 20BCE1355

Karthic S 20BCE1840

Sanjil K C 20BCE1855

School of Computing Science and Engineering

VIT Chennai

Vandalur - Kelambakkam Road, Chennai - 600 127

WINTER SEM 22-23

Worklet details

Programme	B.Tech Computer Science and Engineering	
Course Name / Code	Web Mining / CSE3024	
Slot	A2	
Faculty Name	Dr.A.Bhuvaneswari	
Component	DA - II	
J Component Title	MUSIC RECOMMENDATION SYSTEM	
Team Members Name | Reg. No	Vishal Sankar	20BCE1355
	Karthic S	20BCE1840
	Sanjil K C	20BCE1855

Team Members(s) Contributions:

Worklet Tasks	*Contributor's Names*
Preprocessing	Sanjil K C, Karthic S, Vishal Sankar
Model building	Sanjil K C, Karthic S, Vishal Sankar
Visualization	Sanjil K C, Karthic S, Vishal Sankar
Technical Report writing	Sanjil K C, Karthic S, Vishal Sankar
Presentation preparation	Sanjil K C, Karthic S, Vishal Sankar

TABLE OF CONTENTS

1. Introduction

With the rise of digital content distribution, people now have access to music collections on an unprecedented scale. Commercial music libraries easily exceed 15 million songs, which vastly exceeds the listening capability of any single person. With millions of songs to choose from, people sometimes feel overwhelmed. Thus, an efficient music recommender system is necessary in the interest of both music service providers and customers. Users will have no more pain to make decisions on what to listen while music companies can maintain their user group and attract new users by improving users' satisfaction. In the academic field, the domain of user centric music recommendation has always been ignored due to the lack of publicly available, open and transparent data. Million Song Dataset Challenge provides data which is open and large scale which facilitates academic research in user centric music recommender system which hasn't been studied a lot.

We can know the preferences of a user through ratings. There are two ways to collect user ratings.The first one is by using Explicit Rating, this means we explicitly ask the user to give a rating. This represents the most direct feedback from users to show how much they like a song. The second uses Implicit Rating; so for example, we examine whether or not a user listened to a song, for how long or how many times, which may suggest that he/she liked that particular song.

After we collect ratings, we can generate interaction matrices. Interaction matrices are based on the many entries that include a user-song pair as well as a value that represents the user's rating for that song.

2. Literature Survey

Sl no	Title	Author / Journal name / Year	Technique	Result
1	Music Recommendations System using Collaborative Filtering	Euge Inzaugrat Towards Data Science 2020	Collaborative Filtering – User based Approach, Item based approach. KNN algorithm	The KNN algorithm, works with the provided dataset and the suitable recommendations are displayed.

2	Music Recommender System:	Shefali Garg and Fangyan. Indian Institute of Technology, Kanpur. 2014	Popularity Based model, Collaborative Based Model, SVD Model, KNN Model	The best result was got from Collaborative Based Model. SVD based latent factor model gives better results than popularitybased model. K-NN model did not work well and performs worse than even the popularity model.
3	Music recommendations based on User Sentiments, Extracted from Social Networks	Reneta Rosa – Universidada Federal de Lavras Graca Bressan University of Sao Paulo ResearchGate 2015	eSM sentiment Metric	The new sentiment intensity metric, eSM, improved the music recommendation system, showing that the sentiments can change depending on the user's profile
4	Music Recommendation System using Genetic Algorithm	Govind P.Wakure and Vijayalaxmi Kadrolli - Terna Engg.College, researchPortal 2013	Crossover algorithm – BLX-a	The recommender system that was able to accurately recognize the trend of a user’s preference and adaptively provide an appropriate recommendation in a time efficient manner

5	Music Recommendation Using Collaborative Filtering	Vuong Khuat Portfolios.cs 2014	Collaborative Filtering, Clustering	This method saved memory and computation time because it took advantage of the fact that only a subset of the dataset was needed to generate recommendations. While clustering methods could also be efficient, their recommendation quality were relatively low. On the other hand, using a traditional approach may result in good recommendations, but their inefficiency made it difficult to implement in practice.
6	Content Based Music Recommendations System	Euge Inzaugarat Towards Datascience 2020	Keyword Matching,KF-IDF(Term Frequency-Inverse Document Frequency)	The Content based recommendation system works for the given song, and the suitable songs are recommended
7	A music Recommendation System Based on User Behavior	Yajie Hu & Mitsunori Ogihara Science University of Miami 2015	Time series analysis of genre based prediction	The recommendation approach user is effective. This approach is able to fit to a user's taste, and adjust the recommendation strategy quickly whenever user skips a song.

3. Dataset and Tool to be used

Dataset: We would be using the database provided by Kaggle competition:

Million Song Dataset Challenge | Kaggle

It includes metadata (e.g., artist identifiers, tags,etc) , audio content analysis and standardized identifiers.

Tools to be used:

Programming languages: Python, R

4. Proposed Methodology – Framework

The Methodology is to implement three techniques, Popularity based filtering, Content based filtering and Collaborative filtering. For this we use the basic frameworks like Pandas which is a library for data analysis, Numpy which is used to perform mathematical computations on large arrays, Matplotlib and exclusively for recommendation we use the predefined Recommenders and knn_recommenders class which take the k nearest neighbors of the given argument and returns the recommendation.

5. Algorithms / Techniques description

- Popularity based Filtering

  The most trivial recommendation algorithm is to simply present each song in descending order of its popularity skipping those songs already consumed by the user, regardless of the user's taste profile.

- Collaborative Filtering

  It can be either user-based or item-based. In user-based recommendation, users who listen to the same songs in the past tend to have similar interests and will probably listen to the same songs in future. In the item-based recommendation strategy, songs that are often listened by the same user tend to be similar and are more likely to be listened together in future by some other user.

  There are several machine learning algorithms that can be used in the case of collaborative filtering. K - Nearest Neighbors (KNN) is considered the standard method when it comes to both user-based and item-based collaborative filtering approaches. The algorithm is a supervised non-parametric Lazy Learning method used for both classification and regression. It considers a graph based on the rating and plot the rating of the input song in the graph and calculates the distance with all the other songs using cosine similarity and recommends the song based on the distance i.e. least distance is compared. KNN is a machine learning algorithm to find clusters of similar users based on common book ratings, and make predictions using the average rating of top-k nearest neighbors.

- Content based Filtering

Content-based filtering makes recommendations by using keywords and attributes assigned to objects in a database. In this case, we find the songs that has been listened by the user previously and recommend the same.

6. Experimental Results Code:

Importing modules

```
import pandas as pd import numpy
as np
import Recommenders as Recommenders import
matplotlib.pyplot as plt
```

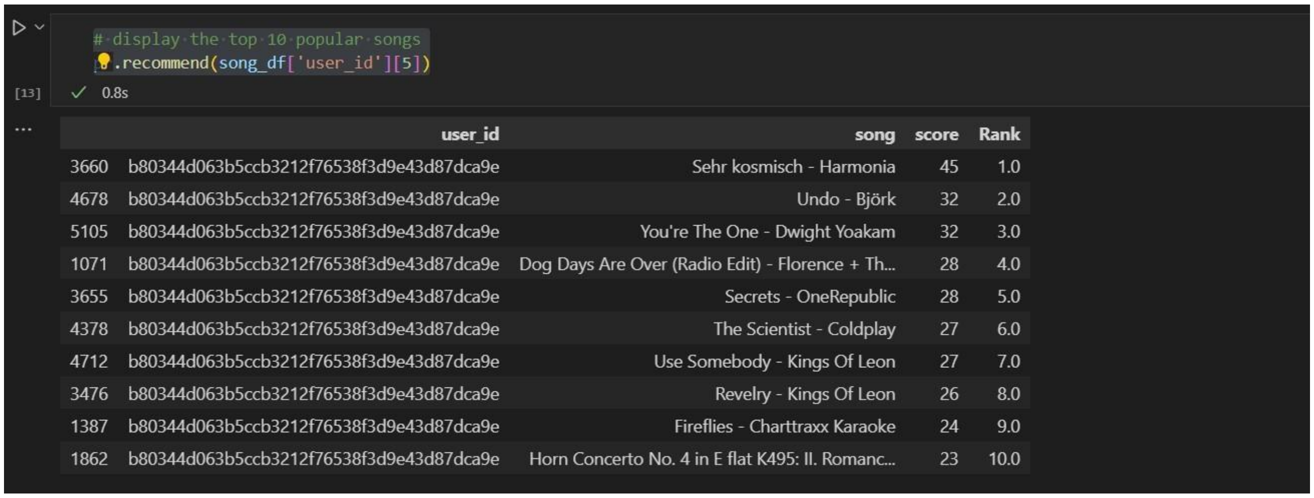
Reading Datasets and Preprocessing

```
song_df_1 = pd.read_csv("triplets_file.csv") song_df_1.head() song_df_2 = pd.read_csv("song_data.csv")
song_df_2.head() # combine both data song_df = pd.merge(song_df_1, song_df_2.drop_duplicates(['song_id']),
on='song_id', how='left') song_df.head() print(len(song_df_1), len(song_df_2))
```

```
len(song_df)
# creating new feature combining title and artist name song_df['song'] = song_df['title']+' -
'+song_df['artist_name'] song_df.head()
# taking top 10k samples for quick results song_df =
song_df.head(10000)
# cummulative sum of listen count of the songs song_grouped =
song_df.groupby(['song']).agg({'listen_count':'count'}).reset_index() song_grouped.head() grouped_sum =
song_grouped['listen_count'].sum() song_grouped['percentage'] = (song_grouped['listen_count'] /
grouped_sum ) * 100 song_grouped.sort_values(['listen_count', 'song'], ascending=[0,1])
```

Popularity Based Recommendation

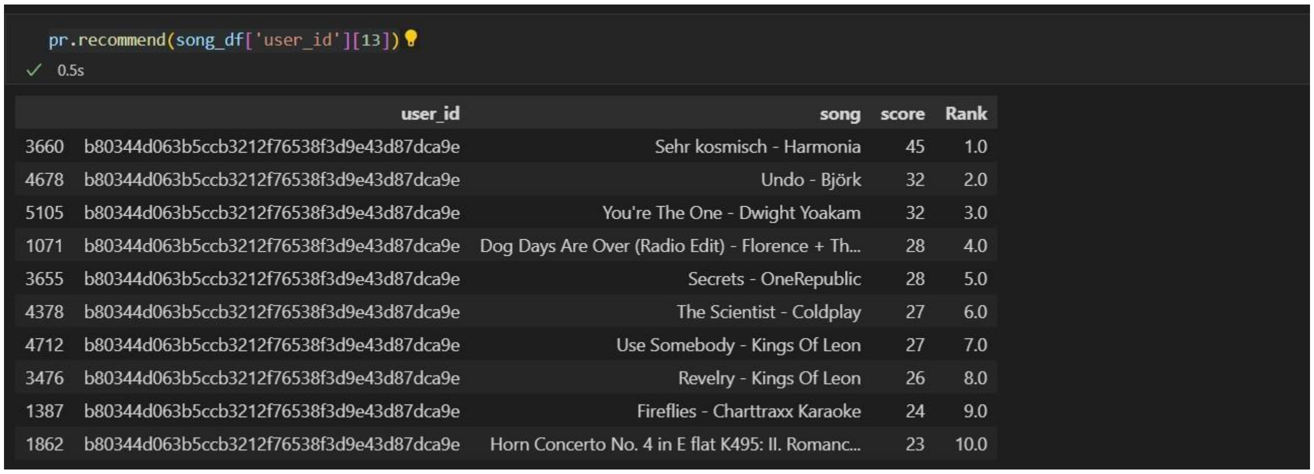
```
pr = Recommenders.popularity_recommender_py() pr.create(song_df, 'user_id', 'song')
# display the top 10 popular songs pr.recommend(song_df['user_id'][5])
pr.recommend(song_df['user_id'][13])
```



```
# display the top 10 popular songs
.recommend(song_df['user_id'][5])
```

[13] 0.8s

	user_id	song	score	Rank
3660	b80344d063b5ccb3212f76538f3d9e43d87dca9e	Sehr kosmisch - Harmonia	45	1.0
4678	b80344d063b5ccb3212f76538f3d9e43d87dca9e	Undo - Björk	32	2.0
5105	b80344d063b5ccb3212f76538f3d9e43d87dca9e	You're The One - Dwight Yoakam	32	3.0
1071	b80344d063b5ccb3212f76538f3d9e43d87dca9e	Dog Days Are Over (Radio Edit) - Florence + Th...	28	4.0
3655	b80344d063b5ccb3212f76538f3d9e43d87dca9e	Secrets - OneRepublic	28	5.0
4378	b80344d063b5ccb3212f76538f3d9e43d87dca9e	The Scientist - Coldplay	27	6.0
4712	b80344d063b5ccb3212f76538f3d9e43d87dca9e	Use Somebody - Kings Of Leon	27	7.0
3476	b80344d063b5ccb3212f76538f3d9e43d87dca9e	Revelry - Kings Of Leon	26	8.0
1387	b80344d063b5ccb3212f76538f3d9e43d87dca9e	Fireflies - Charttraxx Karaoke	24	9.0
1862	b80344d063b5ccb3212f76538f3d9e43d87dca9e	Horn Concerto No. 4 in E flat K495: II. Romanc...	23	10.0



```
pr.recommend(song_df['user_id'][13])
```

0.5s

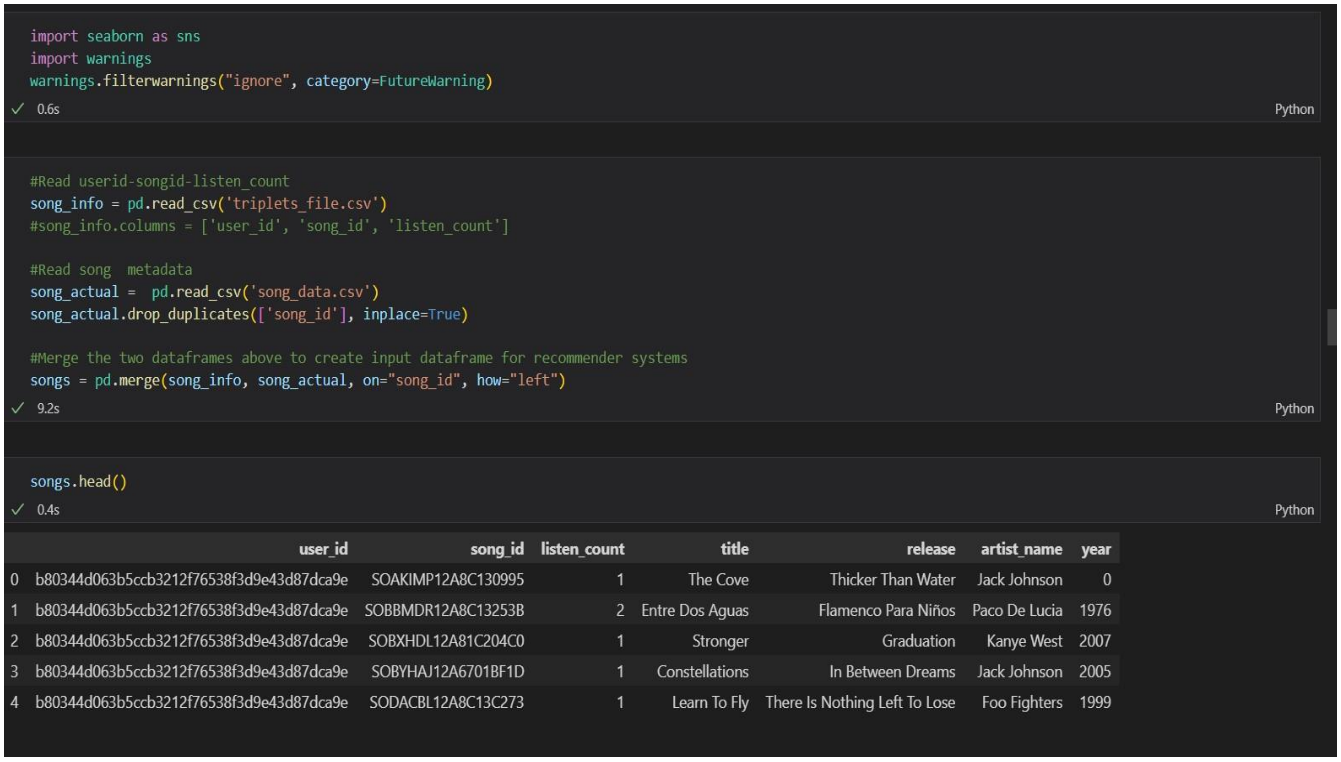
	user_id	song	score	Rank
3660	b80344d063b5ccb3212f76538f3d9e43d87dca9e	Sehr kosmisch - Harmonia	45	1.0
4678	b80344d063b5ccb3212f76538f3d9e43d87dca9e	Undo - Björk	32	2.0
5105	b80344d063b5ccb3212f76538f3d9e43d87dca9e	You're The One - Dwight Yoakam	32	3.0
1071	b80344d063b5ccb3212f76538f3d9e43d87dca9e	Dog Days Are Over (Radio Edit) - Florence + Th...	28	4.0
3655	b80344d063b5ccb3212f76538f3d9e43d87dca9e	Secrets - OneRepublic	28	5.0
4378	b80344d063b5ccb3212f76538f3d9e43d87dca9e	The Scientist - Coldplay	27	6.0
4712	b80344d063b5ccb3212f76538f3d9e43d87dca9e	Use Somebody - Kings Of Leon	27	7.0
3476	b80344d063b5ccb3212f76538f3d9e43d87dca9e	Revelry - Kings Of Leon	26	8.0
1387	b80344d063b5ccb3212f76538f3d9e43d87dca9e	Fireflies - Charttraxx Karaoke	24	9.0
1862	b80344d063b5ccb3212f76538f3d9e43d87dca9e	Horn Concerto No. 4 in E flat K495: II. Romanc...	23	10.0

Content Based Recommendation

```
ir = Recommenders.item_similarity_recommender_py()
ir.create(song_df, 'user_id', 'song') ir =
Recommenders.item_similarity_recommender_py()
ir.create(song_df, 'user_id', 'song')
# display user songs history for user_item in
user_items:
    print(user_item)
# give song recommendation for that user
ir.recommend(song_df['user_id'][5])
ir.recommend(song_df['user_id'][100])
```

Collaborative Filtering:

Preparing the Dataset:



```python
import seaborn as sns
import warnings
warnings.filterwarnings("ignore", category=FutureWarning)
```

✓ 0.6s Python

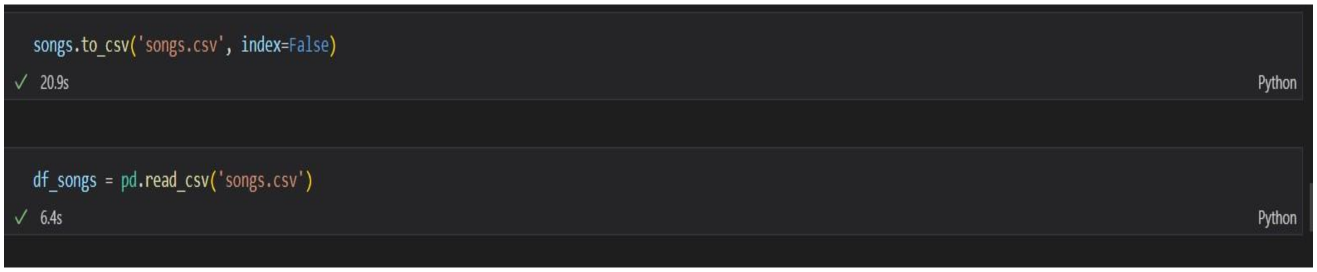
```python
#Read userid-songid-listen_count
song_info = pd.read_csv('triplets_file.csv')
#song_info.columns = ['user_id', 'song_id', 'listen_count']

#Read song  metadata
song_actual =  pd.read_csv('song_data.csv')
song_actual.drop_duplicates(['song_id'], inplace=True)

#Merge the two dataframes above to create input dataframe for recommender systems
songs = pd.merge(song_info, song_actual, on="song_id", how="left")
```

✓ 9.2s Python

```python
songs.head()
```

✓ 0.4s Python

```
                                    user_id             song_id  listen_count           title                       release   artist_name  year
0  b80344d063b5ccb3212f76538f3d9e43d87dca9e  SOAKIMP12A8C130995             1        The Cove            Thicker Than Water  Jack Johnson     0
1  b80344d063b5ccb3212f76538f3d9e43d87dca9e  SOBBMDR12A8C13253B             2  Entre Dos Aguas           Flamenco Para Niños  Paco De Lucia  1976
2  b80344d063b5ccb3212f76538f3d9e43d87dca9e  SOBXHDL12A81C204C0             1        Stronger                    Graduation    Kanye West  2007
3  b80344d063b5ccb3212f76538f3d9e43d87dca9e  SOBYHAJ12A6701BF1D             1  Constellations              In Between Dreams  Jack Johnson  2005
4  b80344d063b5ccb3212f76538f3d9e43d87dca9e  SODACBL12A8C13C273             1     Learn To Fly  There Is Nothing Left To Lose  Foo Fighters  1999
```



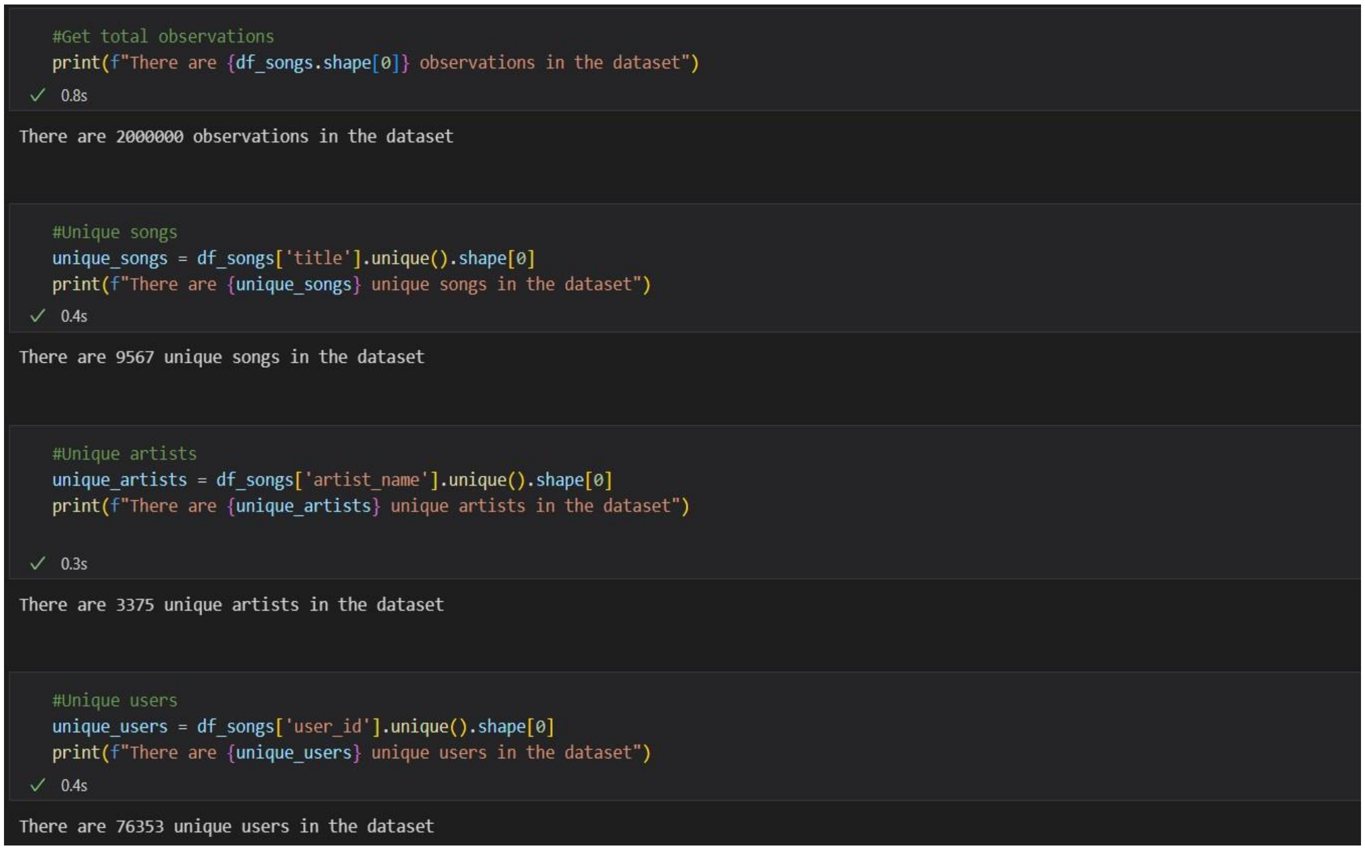
```python
songs.to_csv('songs.csv', index=False)
```

✓ 20.9s Python

```python
df_songs = pd.read_csv('songs.csv')
```

✓ 6.4s Python

Performing Exploratory data analysis to understand the dataset for collaborative filtering



```
#Get total observations
print(f"There are {df_songs.shape[0]} observations in the dataset")
```
✓ 0.8s

```
There are 2000000 observations in the dataset
```

```
#Unique songs
unique_songs = df_songs['title'].unique().shape[0]
print(f"There are {unique_songs} unique songs in the dataset")
```
✓ 0.4s

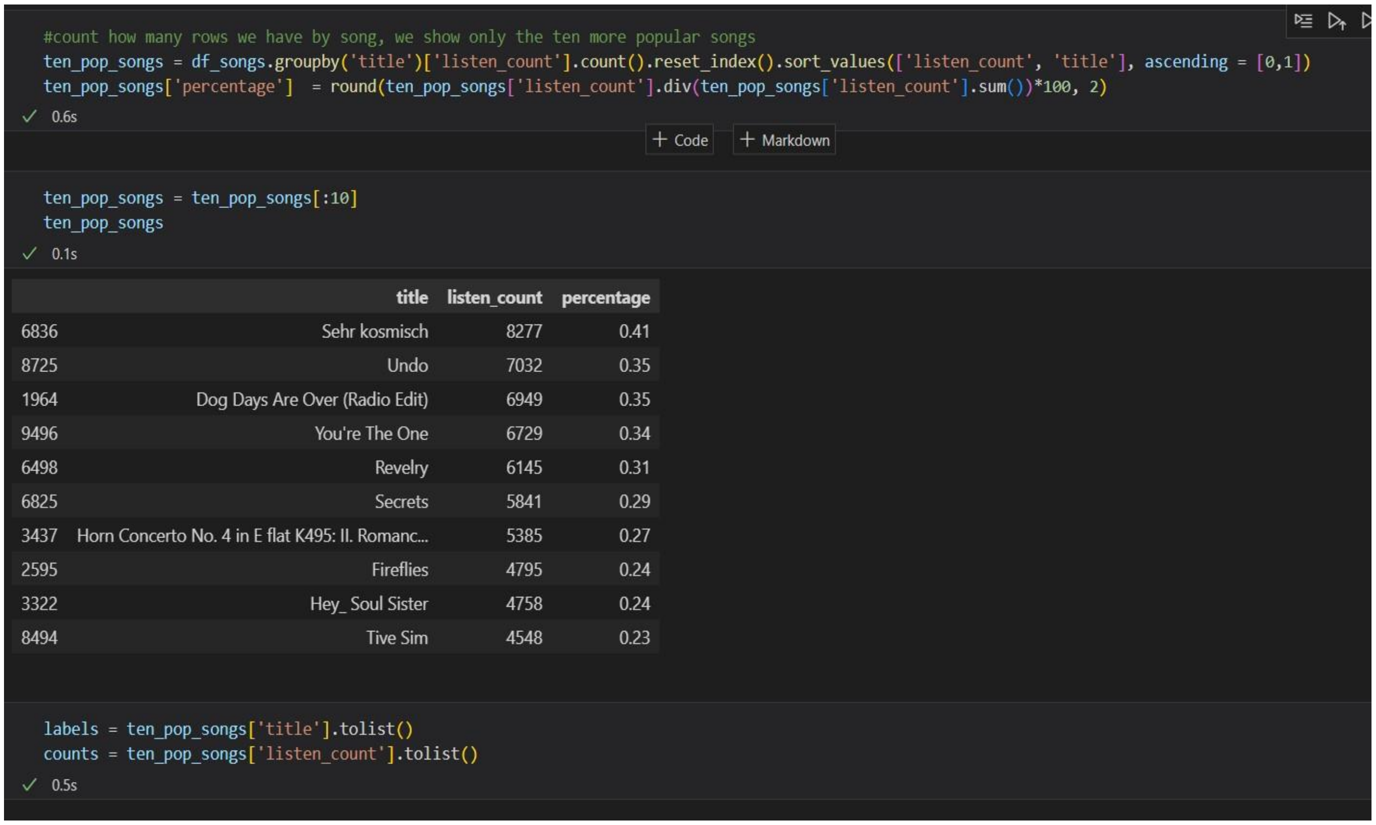
```
There are 9567 unique songs in the dataset
```

```
#Unique artists
unique_artists = df_songs['artist_name'].unique().shape[0]
print(f"There are {unique_artists} unique artists in the dataset")
```
✓ 0.3s

```
There are 3375 unique artists in the dataset
```

```
#Unique users
unique_users = df_songs['user_id'].unique().shape[0]
print(f"There are {unique_users} unique users in the dataset")
```
✓ 0.4s

```
There are 76353 unique users in the dataset
```



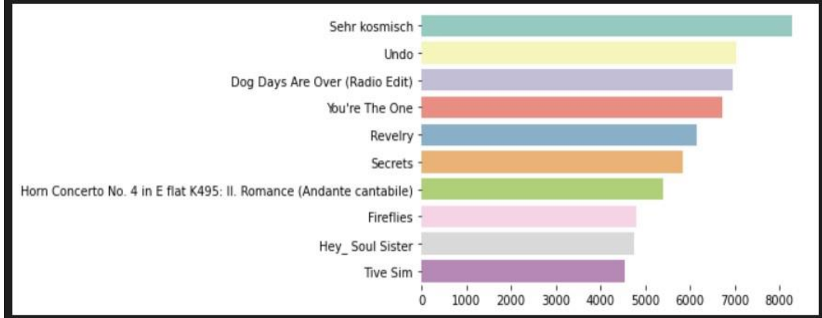
```
#count how many rows we have by song, we show only the ten more popular songs
ten_pop_songs = df_songs.groupby('title')['listen_count'].count().reset_index().sort_values(['listen_count', 'title'], ascending = [0,1])
ten_pop_songs['percentage']  = round(ten_pop_songs['listen_count'].div(ten_pop_songs['listen_count'].sum())*100, 2)
```
✓ 0.6s

```
ten_pop_songs = ten_pop_songs[:10]
ten_pop_songs
```
✓ 0.1s

	title	listen_count	percentage
6836	Sehr kosmisch	8277	0.41
8725	Undo	7032	0.35
1964	Dog Days Are Over (Radio Edit)	6949	0.35
9496	You're The One	6729	0.34
6498	Revelry	6145	0.31
6825	Secrets	5841	0.29
3437	Horn Concerto No. 4 in E flat K495: II. Romanc...	5385	0.27
2595	Fireflies	4795	0.24
3322	Hey_ Soul Sister	4758	0.24
8494	Tive Sim	4548	0.23

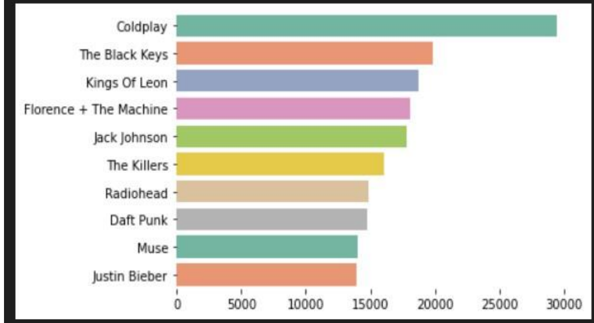
```
labels = ten_pop_songs['title'].tolist()
counts = ten_pop_songs['listen_count'].tolist()
```
✓ 0.5s

```
plt.figure()
sns.barplot(x=counts, y=labels, palette='Set3')
sns.despine(left=True, bottom=True)
```
✓ 0.3s



```
#count how many rows we have by artist name, we show only the ten more popular artist
ten_pop_artists  = df_songs.groupby(['artist_name'])['listen_count'].count().reset_index().sort_values(['listen_count', 'artist_name'],
                                                                                                   ascending = [0,1])
```
✓ 0.3s

```
ten_pop_artists = ten_pop_artists[:10]
ten_pop_artists
plt.figure()
labels = ten_pop_artists['artist_name'].tolist()
counts = ten_pop_artists['listen_count'].tolist()
sns.barplot(x=counts, y=labels, palette='Set2')
sns.despine(left=True, bottom=True)
```
✓ 0.2s



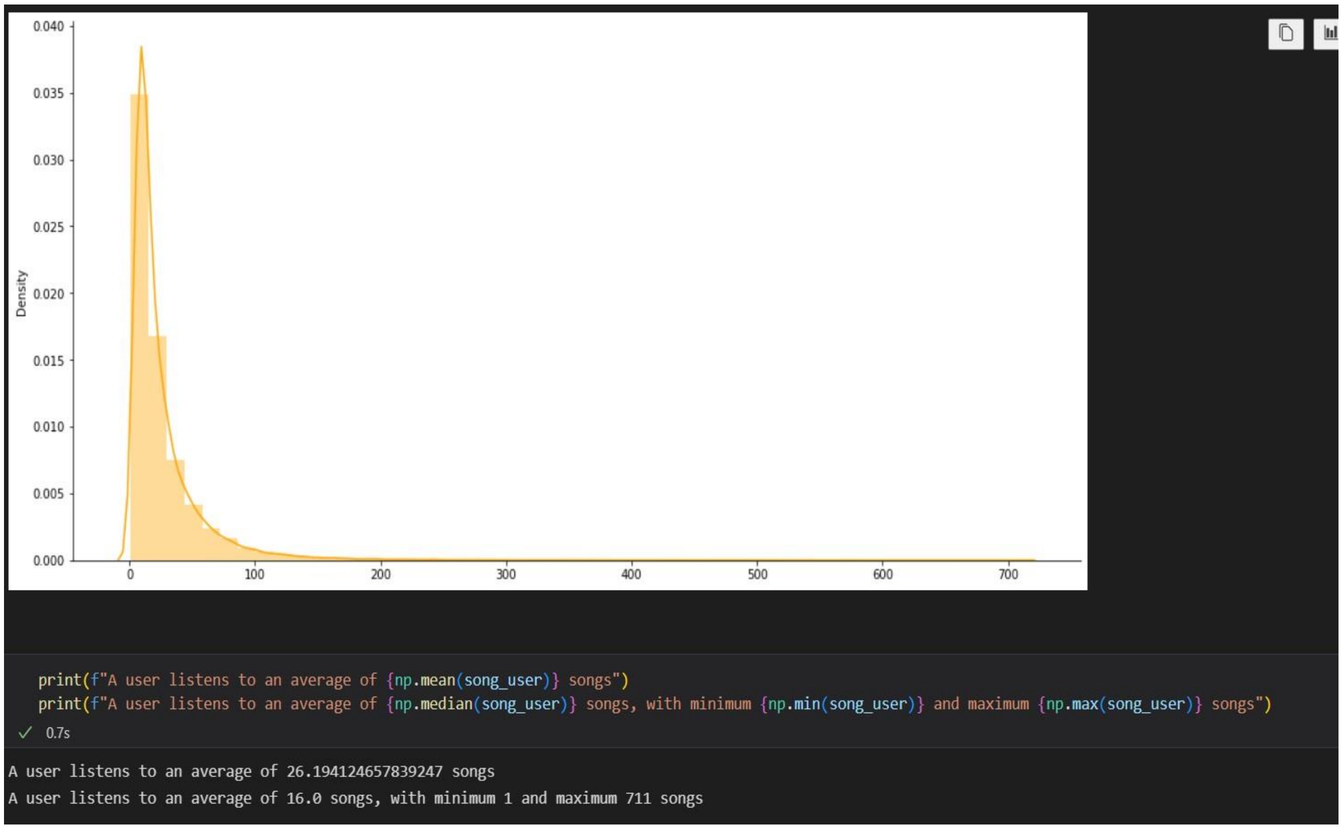
+ Code + Markdown

```
listen_counts = pd.DataFrame(df_songs.groupby('listen_count').size(), columns=['count'])
print(f"The maximum time the same user listened to the same songs was: {listen_counts.reset_index(drop=False)['listen_count'].iloc[-1]}")
print(f"On average, a user listen to the same song {df_songs['listen_count'].mean()} times")
```
✓ 0.1s

```
The maximum time the same user listened to the same songs was: 2213
On average, a user listen to the same song 3.0454845 times
```

```
song_user = df_songs.groupby('user_id')['song_id'].count()
```
✓ 0.7s

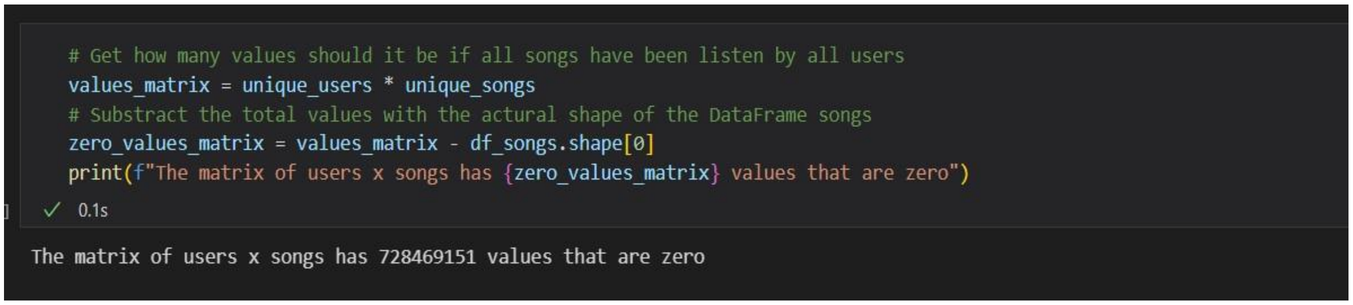
```
plt.figure(figsize=(16, 8))
sns.distplot(song_user.values, color='orange')
plt.gca().spines['top'].set_visible(False)
plt.gca().spines['right'].set_visible(False)
plt.show();
```
✓ 3.3s



```
print(f"A user listens to an average of {np.mean(song_user)} songs")
print(f"A user listens to an average of {np.median(song_user)} songs, with minimum {np.min(song_user)} and maximum {np.max(song_user)} songs")
```

✓ 0.7s

```
A user listens to an average of 26.194124657839247 songs
A user listens to an average of 16.0 songs, with minimum 1 and maximum 711 songs
```

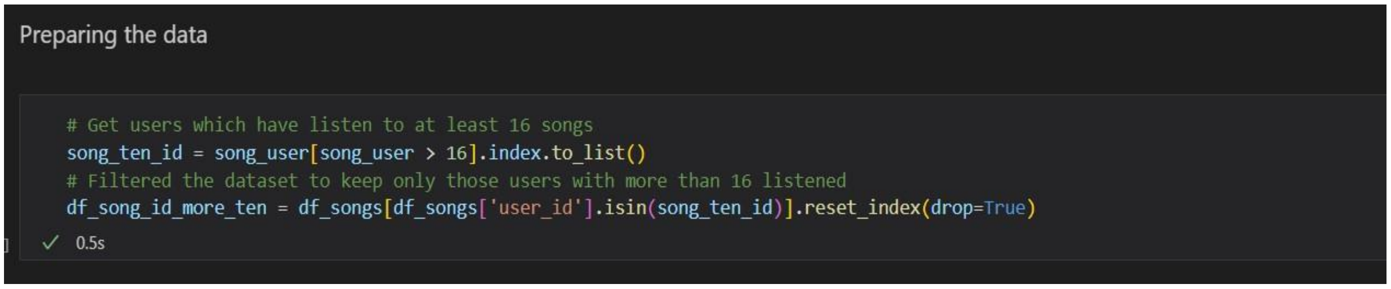


```
# Get how many values should it be if all songs have been listen by all users
values_matrix = unique_users * unique_songs
# Substract the total values with the actual shape of the DataFrame songs
zero_values_matrix = values_matrix - df_songs.shape[0]
print(f"The matrix of users x songs has {zero_values_matrix} values that are zero")
```

✓ 0.1s

```
The matrix of users x songs has 728469151 values that are zero
```

Preparing the data

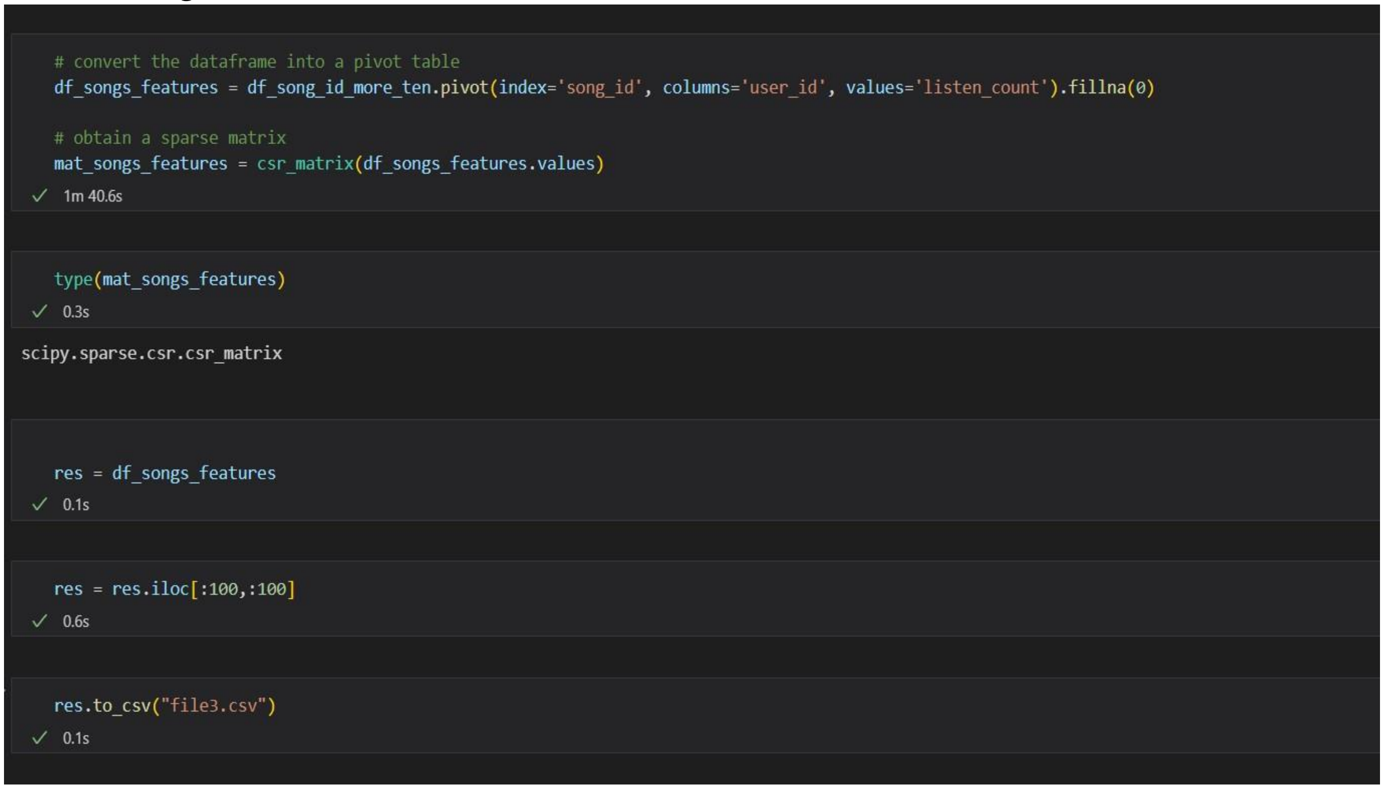


Preparing the data

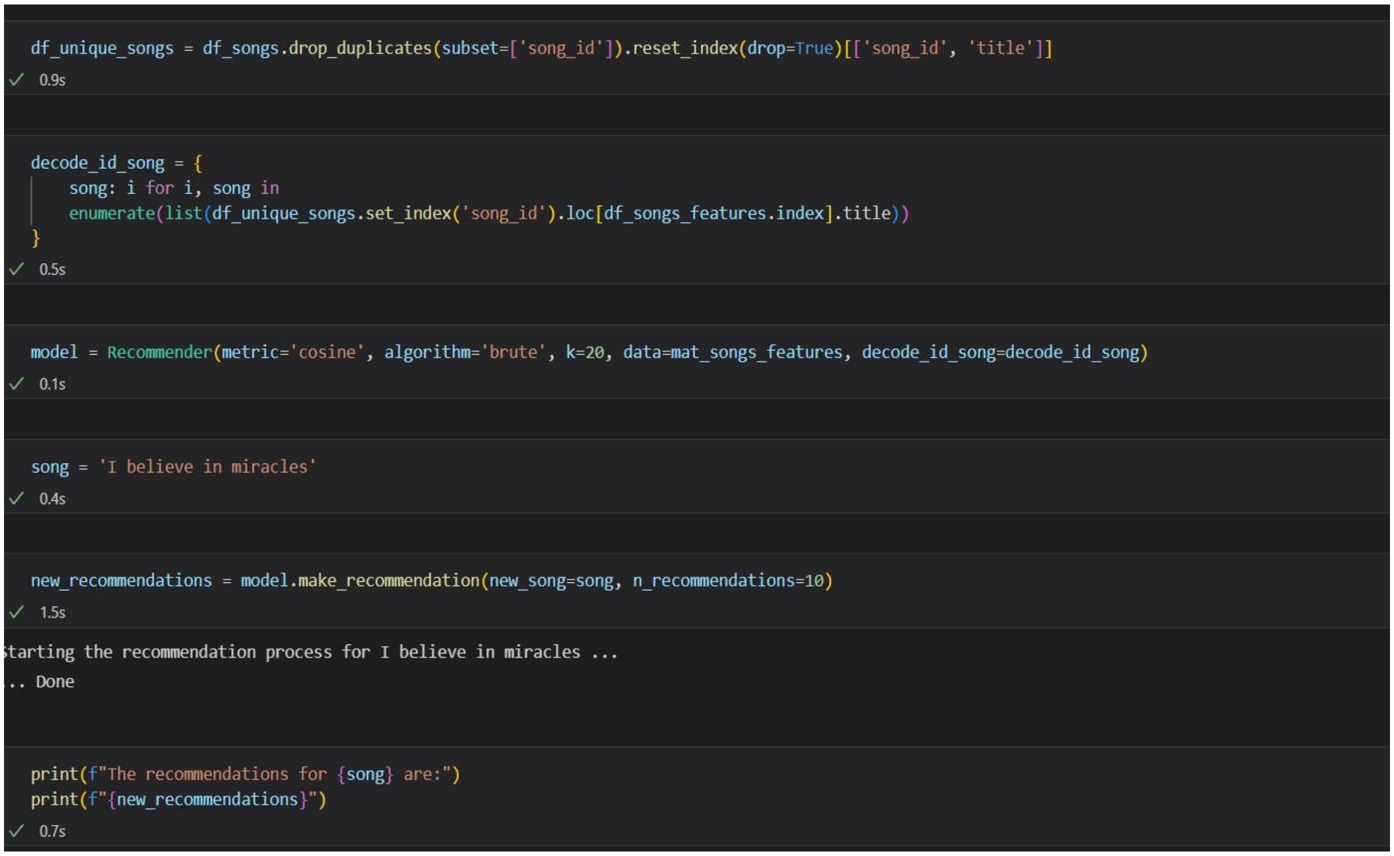
```
# Get users which have listen to at least 16 songs
song_ten_id = song_user[song_user > 16].index.to_list()
# Filtered the dataset to keep only those users with more than 16 listened
df_song_id_more_ten = df_songs[df_songs['user_id'].isin(song_ten_id)].reset_index(drop=True)
```

✓ 0.5s

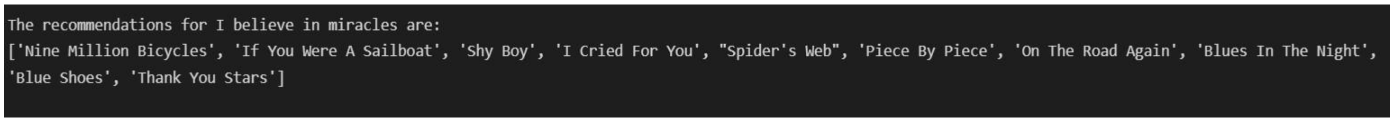
Subsetting the data for visualization:



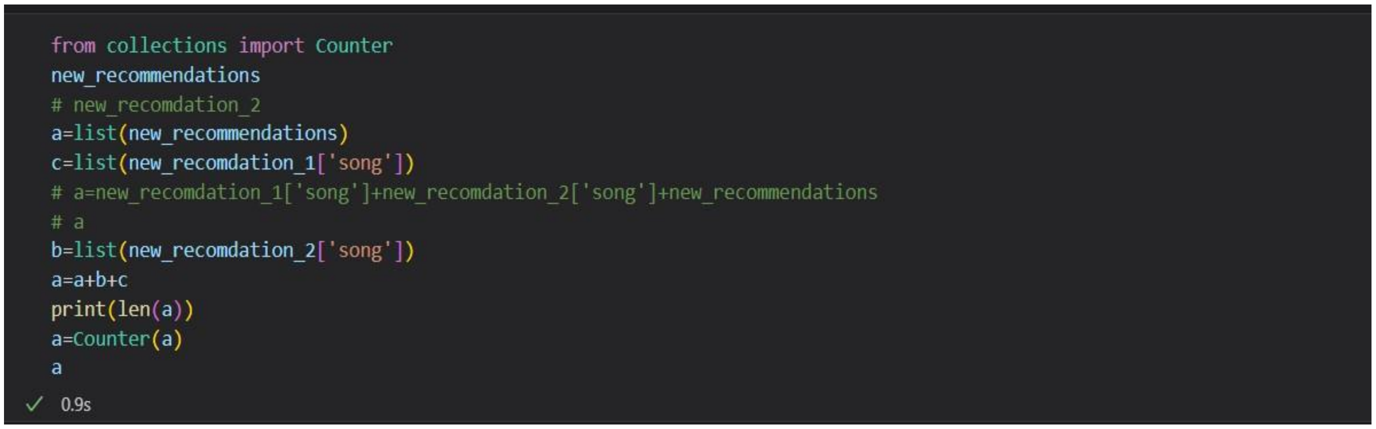
```
# convert the dataframe into a pivot table
df_songs_features = df_song_id_more_ten.pivot(index='song_id', columns='user_id', values='listen_count').fillna(0)

# obtain a sparse matrix
mat_songs_features = csr_matrix(df_songs_features.values)
✓ 1m 40.6s

type(mat_songs_features)
✓ 0.3s

scipy.sparse.csr.csr_matrix

res = df_songs_features
✓ 0.1s

res = res.iloc[:100,:100]
✓ 0.6s

res.to_csv("file3.csv")
✓ 0.1s
```



```
df_unique_songs = df_songs.drop_duplicates(subset=['song_id']).reset_index(drop=True)[['song_id', 'title']]
✓ 0.9s

decode_id_song = {
    song: i for i, song in
    enumerate(list(df_unique_songs.set_index('song_id').loc[df_songs_features.index].title))
}
✓ 0.5s

model = Recommender(metric='cosine', algorithm='brute', k=20, data=mat_songs_features, decode_id_song=decode_id_song)
✓ 0.1s

song = 'I believe in miracles'
✓ 0.4s

new_recommendations = model.make_recommendation(new_song=song, n_recommendations=10)
✓ 1.5s

tarting the recommendation process for I believe in miracles ...
.. Done

print(f"The recommendations for {song} are:")
print(f"{new_recommendations}")
✓ 0.7s
```

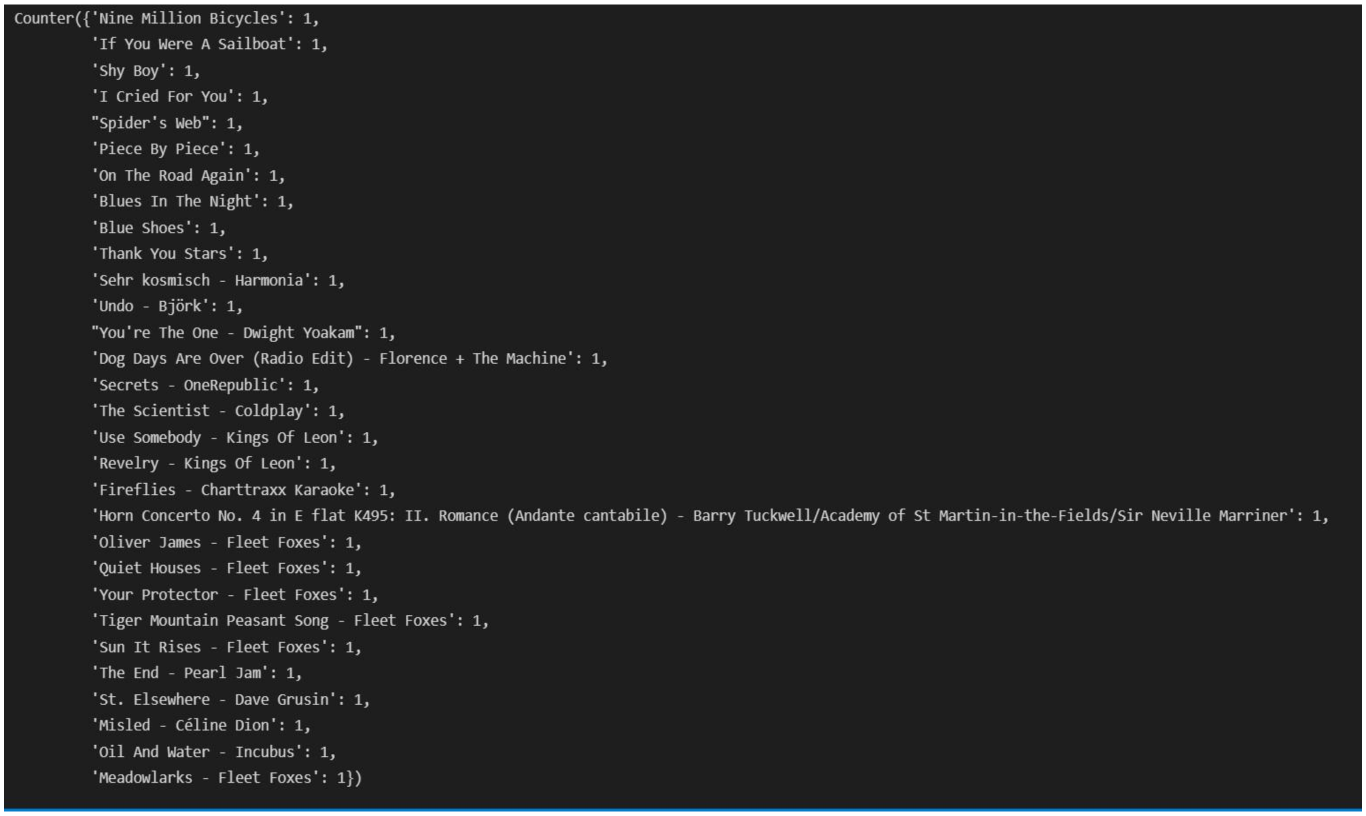


```
The recommendations for I believe in miracles are:
['Nine Million Bicycles', 'If You Were A Sailboat', 'Shy Boy', 'I Cried For You', "Spider's Web", 'Piece By Piece', 'On The Road Again', 'Blues In The Night',
'Blue Shoes', 'Thank You Stars']
```



```python
from collections import Counter
new_recommendations
# new_recomdation_2
a=list(new_recommendations)
c=list(new_recomdation_1['song'])
# a=new_recomdation_1['song']+new_recomdation_2['song']+new_recommendations
# a
b=list(new_recomdation_2['song'])
a=a+b+c
print(len(a))
a=Counter(a)
a
```

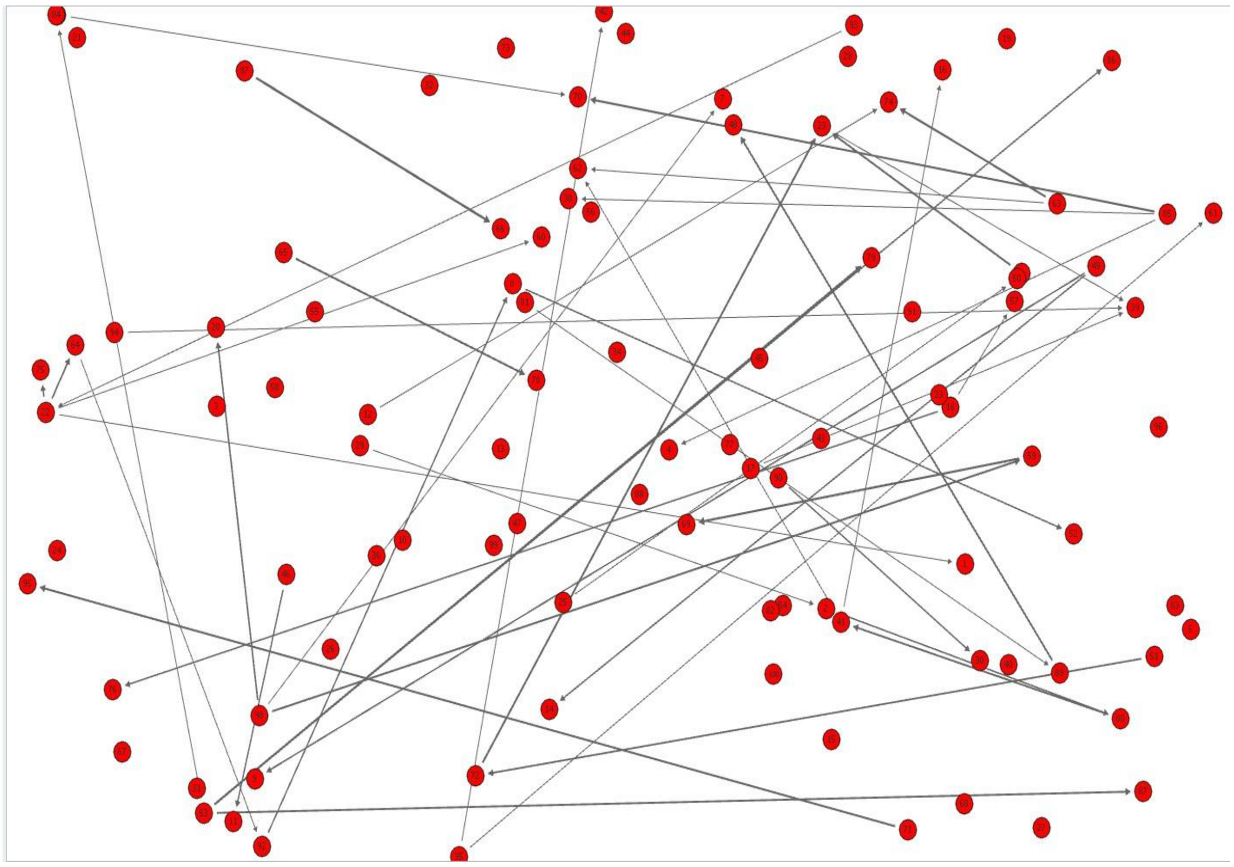
✓ 0.9s



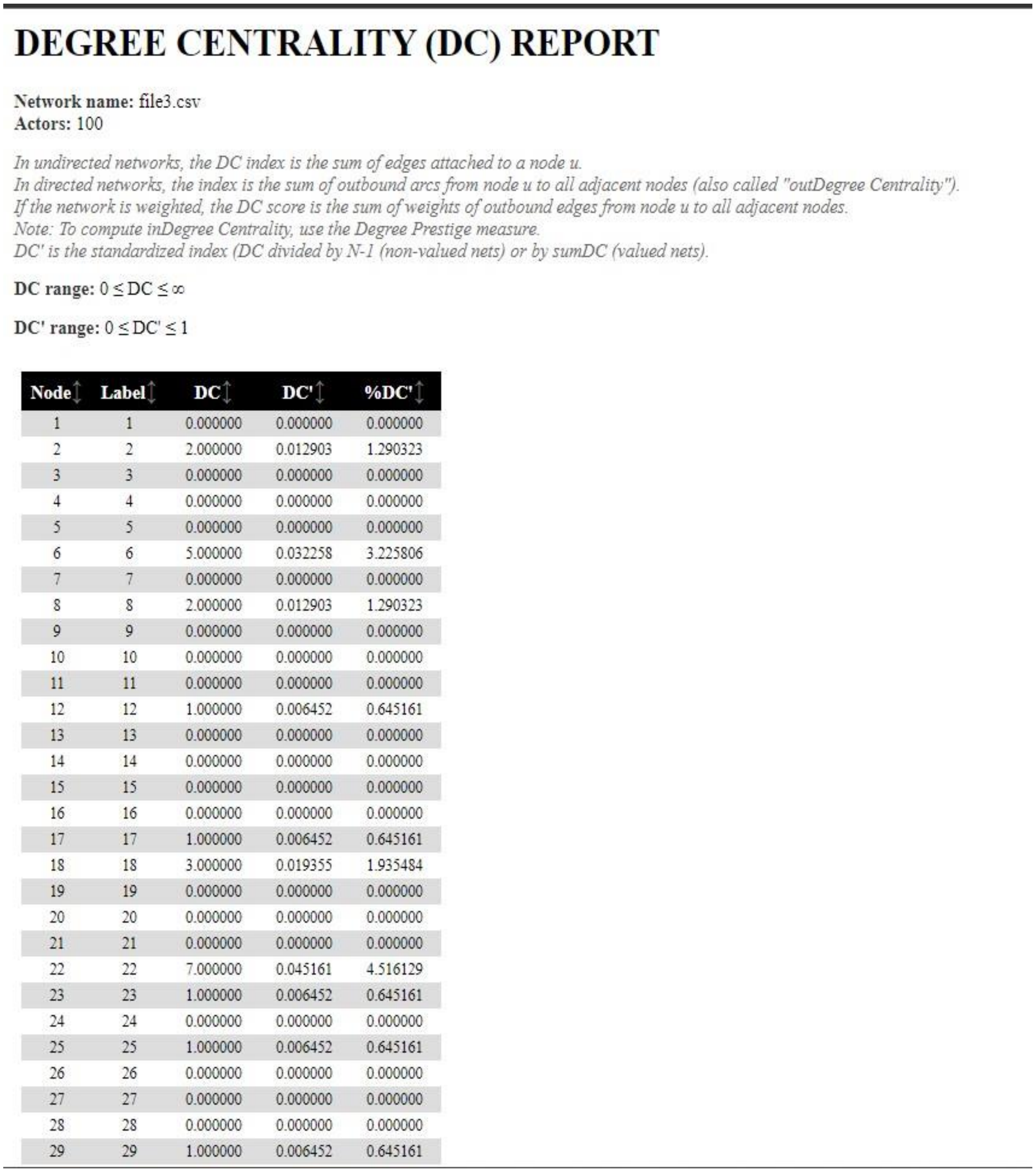
```
Counter({'Nine Million Bicycles': 1,
         'If You Were A Sailboat': 1,
         'Shy Boy': 1,
         'I Cried For You': 1,
         "Spider's Web": 1,
         'Piece By Piece': 1,
         'On The Road Again': 1,
         'Blues In The Night': 1,
         'Blue Shoes': 1,
         'Thank You Stars': 1,
         'Sehr kosmisch - Harmonia': 1,
         'Undo - Björk': 1,
         "You're The One - Dwight Yoakam": 1,
         'Dog Days Are Over (Radio Edit) - Florence + The Machine': 1,
         'Secrets - OneRepublic': 1,
         'The Scientist - Coldplay': 1,
         'Use Somebody - Kings Of Leon': 1,
         'Revelry - Kings Of Leon': 1,
         'Fireflies - Charttraxx Karaoke': 1,
         'Horn Concerto No. 4 in E flat K495: II. Romance (Andante cantabile) - Barry Tuckwell/Academy of St Martin-in-the-Fields/Sir Neville Marriner': 1,
         'Oliver James - Fleet Foxes': 1,
         'Quiet Houses - Fleet Foxes': 1,
         'Your Protector - Fleet Foxes': 1,
         'Tiger Mountain Peasant Song - Fleet Foxes': 1,
         'Sun It Rises - Fleet Foxes': 1,
         'The End - Pearl Jam': 1,
         'St. Elsewhere - Dave Grusin': 1,
         'Misled - Céline Dion': 1,
         'Oil And Water - Incubus': 1,
         'Meadowlarks - Fleet Foxes': 1})
```

7. Model Evaluation

Visualization for the first 100 nodes of the model in SocNetV:



Degree Centrality:



DEGREE CENTRALITY (DC) REPORT

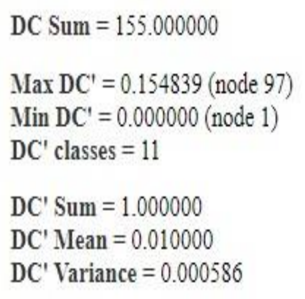
Network name: file3.csv
Actors: 100

In undirected networks, the DC index is the sum of edges attached to a node u.
In directed networks, the index is the sum of outbound arcs from node u to all adjacent nodes (also called "outDegree Centrality").
If the network is weighted, the DC score is the sum of weights of outbound edges from node u to all adjacent nodes.
Note: To compute inDegree Centrality, use the Degree Prestige measure.
DC' is the standardized index (DC divided by N-1 (non-valued nets) or by sumDC (valued nets).

DC range: 0 ≤ DC ≤ ∞

DC' range: 0 ≤ DC' ≤ 1

Node	Label	DC	DC'	%DC'
1	1	0.000000	0.000000	0.000000
2	2	2.000000	0.012903	1.290323
3	3	0.000000	0.000000	0.000000
4	4	0.000000	0.000000	0.000000
5	5	0.000000	0.000000	0.000000
6	6	5.000000	0.032258	3.225806
7	7	0.000000	0.000000	0.000000
8	8	2.000000	0.012903	1.290323
9	9	0.000000	0.000000	0.000000
10	10	0.000000	0.000000	0.000000
11	11	0.000000	0.000000	0.000000
12	12	1.000000	0.006452	0.645161
13	13	0.000000	0.000000	0.000000
14	14	0.000000	0.000000	0.000000
15	15	0.000000	0.000000	0.000000
16	16	0.000000	0.000000	0.000000
17	17	1.000000	0.006452	0.645161
18	18	3.000000	0.019355	1.935484
19	19	0.000000	0.000000	0.000000
20	20	0.000000	0.000000	0.000000
21	21	0.000000	0.000000	0.000000
22	22	7.000000	0.045161	4.516129
23	23	1.000000	0.006452	0.645161
24	24	0.000000	0.000000	0.000000
25	25	1.000000	0.006452	0.645161
26	26	0.000000	0.000000	0.000000
27	27	0.000000	0.000000	0.000000
28	28	0.000000	0.000000	0.000000
29	29	1.000000	0.006452	0.645161



DC Sum = 155.000000

Max DC' = 0.154839 (node 97)
Min DC' = 0.000000 (node 1)
DC' classes = 11

DC' Sum = 1.000000
DC' Mean = 0.010000
DC' Variance = 0.000586

BETWEENNESS CENTRALITY (BC)

Network name: file3.csv
Actors: 100

The BC index of a node u is the sum of $\delta_{(s,t,u)}$ for all s,t ∈ V where $\delta_{(s,t,u)}$ is the ratio of all geodesics between s and t which run through u. Read the Manual for more.
BC' is the standardized index (BC divided by (N-1)(N-2)/2 in symmetric nets or (N-1)(N-2) otherwise.

BC range: 0 ≤ BC ≤ 9702 (Number of pairs of nodes excluding u)

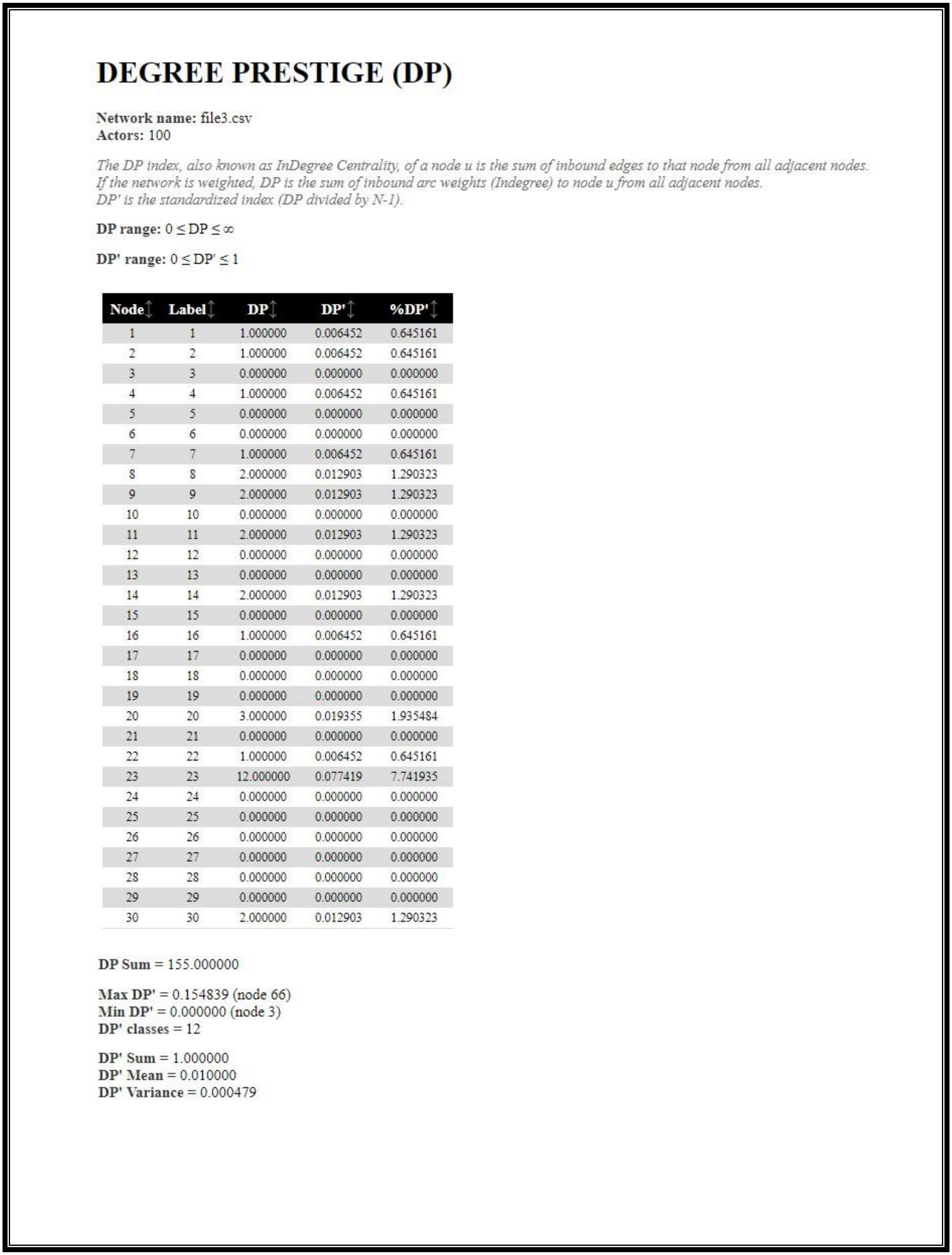
BC' range: 0 ≤ BC' ≤ 1 (BC'=1 when the node falls on all geodesics)

Node	Label	BC	BC'	%BC'
1	1	0.000000	0.000000	0.000000
2	2	4.000000	0.000412	0.041229
3	3	0.000000	0.000000	0.000000
4	4	0.000000	0.000000	0.000000
5	5	0.000000	0.000000	0.000000
6	6	0.000000	0.000000	0.000000
7	7	0.000000	0.000000	0.000000
8	8	4.000000	0.000412	0.041229
9	9	0.000000	0.000000	0.000000
10	10	0.000000	0.000000	0.000000
11	11	0.000000	0.000000	0.000000
12	12	0.000000	0.000000	0.000000
13	13	0.000000	0.000000	0.000000
14	14	0.000000	0.000000	0.000000
15	15	0.000000	0.000000	0.000000
16	16	0.000000	0.000000	0.000000
17	17	0.000000	0.000000	0.000000
18	18	0.000000	0.000000	0.000000
19	19	0.000000	0.000000	0.000000
20	20	0.000000	0.000000	0.000000
21	21	0.000000	0.000000	0.000000
22	22	7.000000	0.000722	0.072150
23	23	3.000000	0.000309	0.030921
24	24	0.000000	0.000000	0.000000
25	25	0.000000	0.000000	0.000000
26	26	0.000000	0.000000	0.000000
27	27	0.000000	0.000000	0.000000
28	28	0.000000	0.000000	0.000000
29	29	0.000000	0.000000	0.000000
30	30	0.000000	0.000000	0.000000

BC Sum = 46.000000

Max BC' = 0.000722 (node 22)
Min BC' = 0.000000 (node 1)
BC' classes = 7

BC' Sum = 0.004741
BC' Mean = 0.000047
BC' Variance = 0.000000



DEGREE PRESTIGE (DP)

Network name: file3.csv
Actors: 100

The DP index, also known as InDegree Centrality, of a node u is the sum of inbound edges to that node from all adjacent nodes.
If the network is weighted, DP is the sum of inbound arc weights (Indegree) to node u from all adjacent nodes.
DP' is the standardized index (DP divided by N-1).

DP range: 0 ≤ DP ≤ ∞

DP' range: 0 ≤ DP' ≤ 1

Node	Label	DP	DP'	%DP'
1	1	1.000000	0.006452	0.645161
2	2	1.000000	0.006452	0.645161
3	3	0.000000	0.000000	0.000000
4	4	1.000000	0.006452	0.645161
5	5	0.000000	0.000000	0.000000
6	6	0.000000	0.000000	0.000000
7	7	1.000000	0.006452	0.645161
8	8	2.000000	0.012903	1.290323
9	9	2.000000	0.012903	1.290323
10	10	0.000000	0.000000	0.000000
11	11	2.000000	0.012903	1.290323
12	12	0.000000	0.000000	0.000000
13	13	0.000000	0.000000	0.000000
14	14	2.000000	0.012903	1.290323
15	15	0.000000	0.000000	0.000000
16	16	1.000000	0.006452	0.645161
17	17	0.000000	0.000000	0.000000
18	18	0.000000	0.000000	0.000000
19	19	0.000000	0.000000	0.000000
20	20	3.000000	0.019355	1.935484
21	21	0.000000	0.000000	0.000000
22	22	1.000000	0.006452	0.645161
23	23	12.000000	0.077419	7.741935
24	24	0.000000	0.000000	0.000000
25	25	0.000000	0.000000	0.000000
26	26	0.000000	0.000000	0.000000
27	27	0.000000	0.000000	0.000000
28	28	0.000000	0.000000	0.000000
29	29	0.000000	0.000000	0.000000
30	30	2.000000	0.012903	1.290323

DP Sum = 155.000000

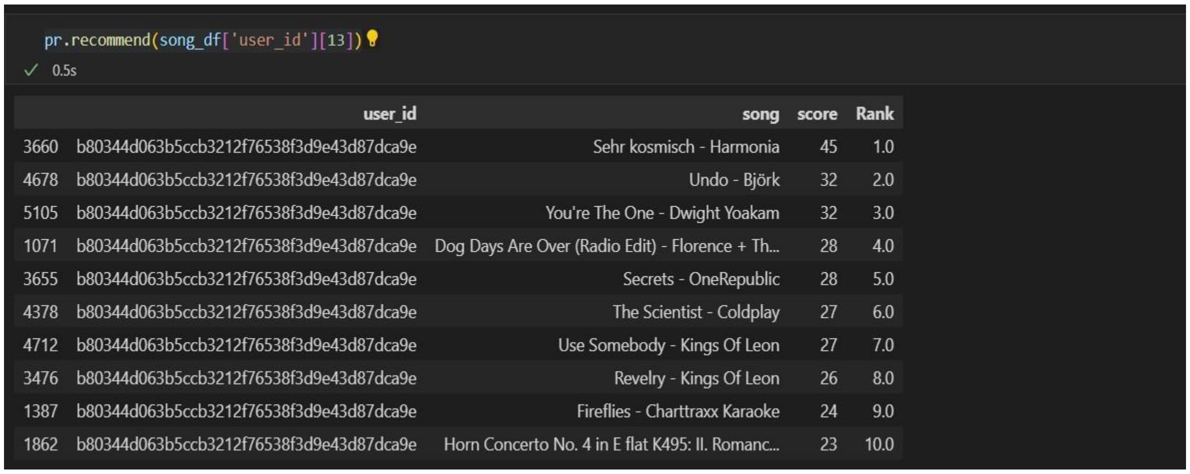
Max DP' = 0.154839 (node 66)
Min DP' = 0.000000 (node 3)
DP' classes = 12

DP' Sum = 1.000000
DP' Mean = 0.010000
DP' Variance = 0.000479

8. Discussion on Results

Results of Popularity Based Filtering:

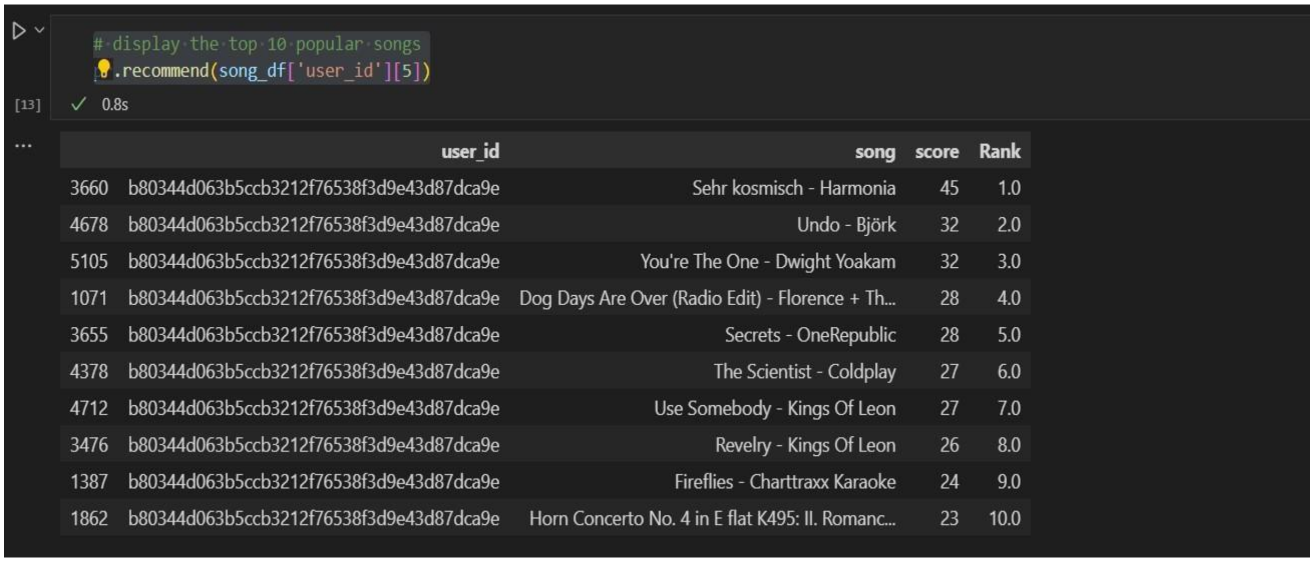
Recommendation for UserId-13:



```
pr.recommend(song_df['user_id'][13])
✓ 0.5s
```

```
                                     user_id                                               song  score  Rank
3660  b80344d063b5ccb3212f76538f3d9e43d87dca9e                           Sehr kosmisch - Harmonia     45   1.0
4678  b80344d063b5ccb3212f76538f3d9e43d87dca9e                                       Undo - Björk     32   2.0
5105  b80344d063b5ccb3212f76538f3d9e43d87dca9e                       You're The One - Dwight Yoakam     32   3.0
1071  b80344d063b5ccb3212f76538f3d9e43d87dca9e  Dog Days Are Over (Radio Edit) - Florence + Th...     28   4.0
3655  b80344d063b5ccb3212f76538f3d9e43d87dca9e                               Secrets - OneRepublic     28   5.0
4378  b80344d063b5ccb3212f76538f3d9e43d87dca9e                             The Scientist - Coldplay     27   6.0
4712  b80344d063b5ccb3212f76538f3d9e43d87dca9e                        Use Somebody - Kings Of Leon     27   7.0
3476  b80344d063b5ccb3212f76538f3d9e43d87dca9e                             Revelry - Kings Of Leon     26   8.0
1387  b80344d063b5ccb3212f76538f3d9e43d87dca9e                      Fireflies - Charttraxx Karaoke     24   9.0
1862  b80344d063b5ccb3212f76538f3d9e43d87dca9e  Horn Concerto No. 4 in E flat K495: II. Romanc...     23  10.0
```

Recommendation for UserId-5



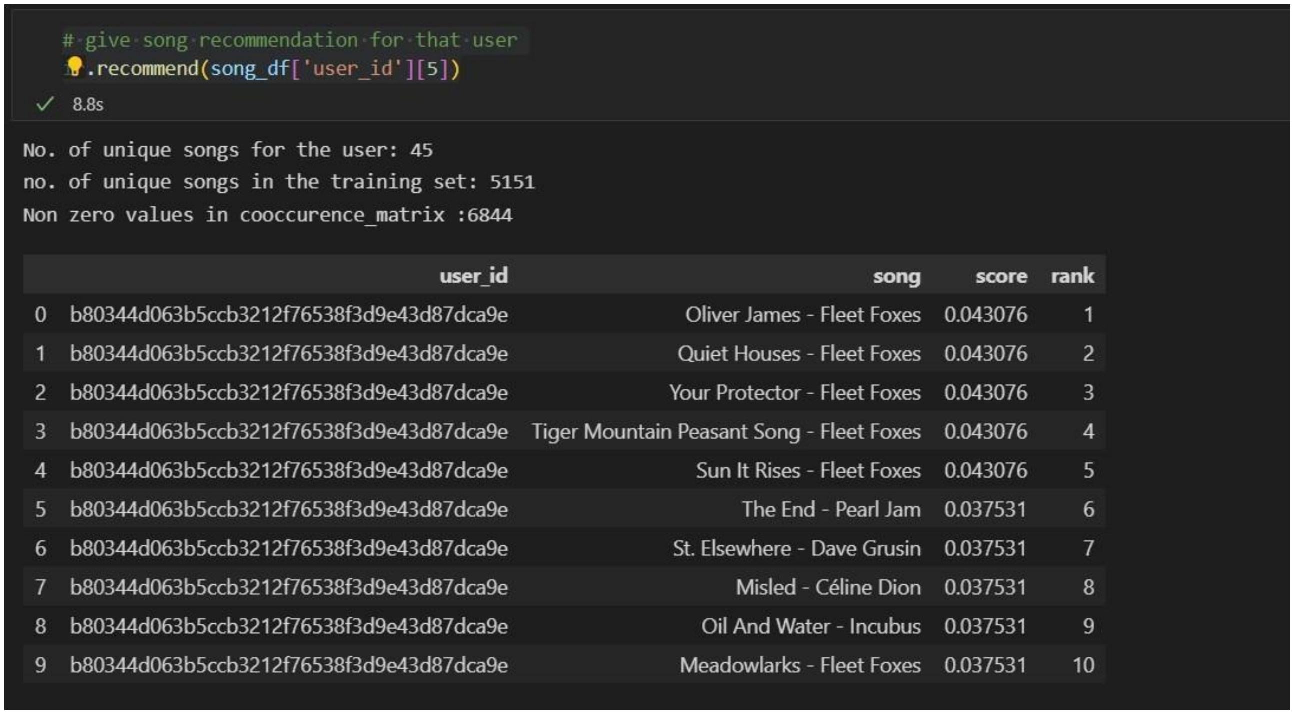
```
# display the top 10 popular songs
.recommend(song_df['user_id'][5])
[13] ✓ 0.8s
```

```
                                     user_id                                               song  score  Rank
3660  b80344d063b5ccb3212f76538f3d9e43d87dca9e                           Sehr kosmisch - Harmonia     45   1.0
4678  b80344d063b5ccb3212f76538f3d9e43d87dca9e                                       Undo - Björk     32   2.0
5105  b80344d063b5ccb3212f76538f3d9e43d87dca9e                       You're The One - Dwight Yoakam     32   3.0
1071  b80344d063b5ccb3212f76538f3d9e43d87dca9e  Dog Days Are Over (Radio Edit) - Florence + Th...     28   4.0
3655  b80344d063b5ccb3212f76538f3d9e43d87dca9e                               Secrets - OneRepublic     28   5.0
4378  b80344d063b5ccb3212f76538f3d9e43d87dca9e                             The Scientist - Coldplay     27   6.0
4712  b80344d063b5ccb3212f76538f3d9e43d87dca9e                        Use Somebody - Kings Of Leon     27   7.0
3476  b80344d063b5ccb3212f76538f3d9e43d87dca9e                             Revelry - Kings Of Leon     26   8.0
1387  b80344d063b5ccb3212f76538f3d9e43d87dca9e                      Fireflies - Charttraxx Karaoke     24   9.0
1862  b80344d063b5ccb3212f76538f3d9e43d87dca9e  Horn Concerto No. 4 in E flat K495: II. Romanc...     23  10.0
```

Since the popularity-based filtering calculates the total score of each song, based on the listen count and returns the songs which is most widely listened, it gives the same recommendation for each and every user. In this example we can see that this technique returns the same set of recommendation for both the given user-ids.

Results of Content Based Filtering:

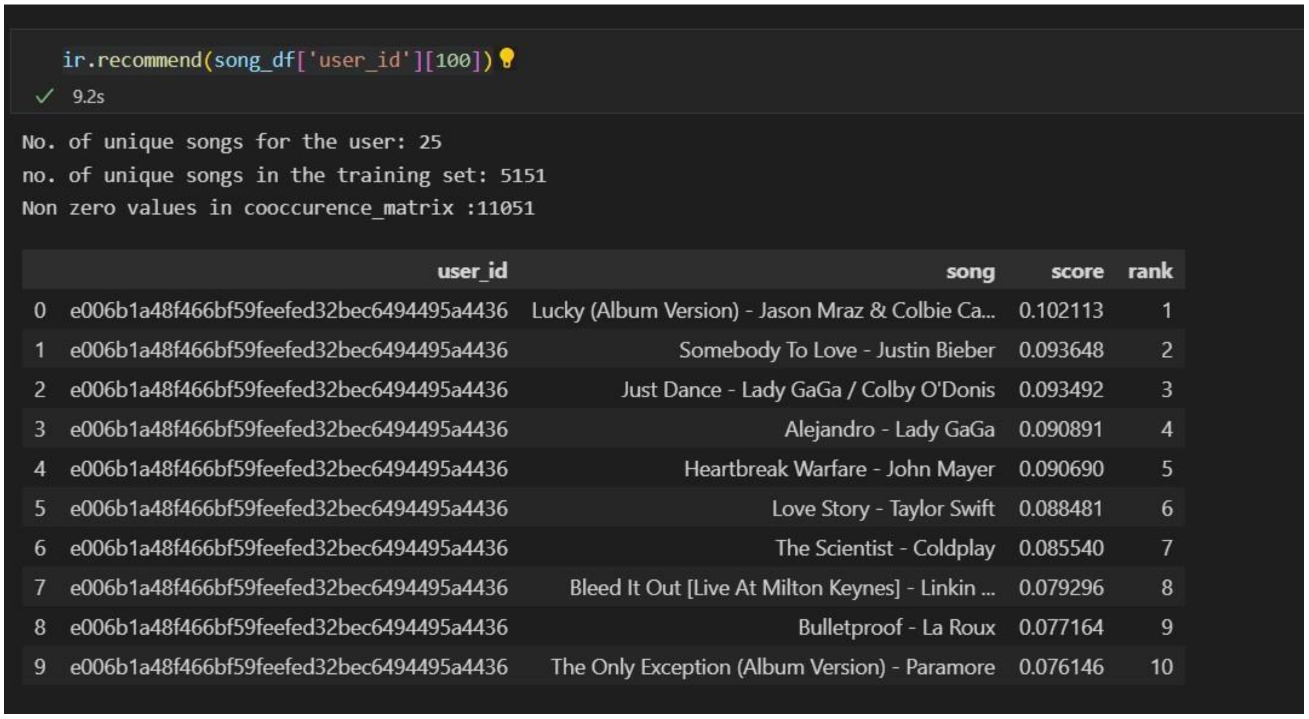
Recommendation for userId-5:



```
# give song recommendation for that user
.recommend(song_df['user_id'][5])
✓ 8.8s

No. of unique songs for the user: 45
no. of unique songs in the training set: 5151
Non zero values in cooccurence_matrix :6844
```

	user_id	song	score	rank
0	b80344d063b5ccb3212f76538f3d9e43d87dca9e	Oliver James - Fleet Foxes	0.043076	1
1	b80344d063b5ccb3212f76538f3d9e43d87dca9e	Quiet Houses - Fleet Foxes	0.043076	2
2	b80344d063b5ccb3212f76538f3d9e43d87dca9e	Your Protector - Fleet Foxes	0.043076	3
3	b80344d063b5ccb3212f76538f3d9e43d87dca9e	Tiger Mountain Peasant Song - Fleet Foxes	0.043076	4
4	b80344d063b5ccb3212f76538f3d9e43d87dca9e	Sun It Rises - Fleet Foxes	0.043076	5
5	b80344d063b5ccb3212f76538f3d9e43d87dca9e	The End - Pearl Jam	0.037531	6
6	b80344d063b5ccb3212f76538f3d9e43d87dca9e	St. Elsewhere - Dave Grusin	0.037531	7
7	b80344d063b5ccb3212f76538f3d9e43d87dca9e	Misled - Céline Dion	0.037531	8
8	b80344d063b5ccb3212f76538f3d9e43d87dca9e	Oil And Water - Incubus	0.037531	9
9	b80344d063b5ccb3212f76538f3d9e43d87dca9e	Meadowlarks - Fleet Foxes	0.037531	10

Recommendation for userId-100:



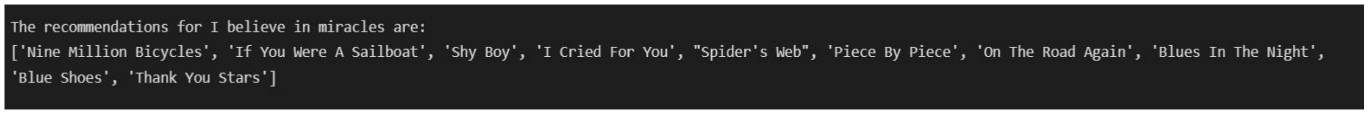
```
ir.recommend(song_df['user_id'][100])
✓ 9.2s

No. of unique songs for the user: 25
no. of unique songs in the training set: 5151
Non zero values in cooccurence_matrix :11051
```

	user_id	song	score	rank
0	e006b1a48f466bf59feefed32bec6494495a4436	Lucky (Album Version) - Jason Mraz & Colbie Ca...	0.102113	1
1	e006b1a48f466bf59feefed32bec6494495a4436	Somebody To Love - Justin Bieber	0.093648	2
2	e006b1a48f466bf59feefed32bec6494495a4436	Just Dance - Lady GaGa / Colby O'Donis	0.093492	3
3	e006b1a48f466bf59feefed32bec6494495a4436	Alejandro - Lady GaGa	0.090891	4
4	e006b1a48f466bf59feefed32bec6494495a4436	Heartbreak Warfare - John Mayer	0.090690	5
5	e006b1a48f466bf59feefed32bec6494495a4436	Love Story - Taylor Swift	0.088481	6
6	e006b1a48f466bf59feefed32bec6494495a4436	The Scientist - Coldplay	0.085540	7
7	e006b1a48f466bf59feefed32bec6494495a4436	Bleed It Out [Live At Milton Keynes] - Linkin ...	0.079296	8
8	e006b1a48f466bf59feefed32bec6494495a4436	Bulletproof - La Roux	0.077164	9
9	e006b1a48f466bf59feefed32bec6494495a4436	The Only Exception (Album Version) - Paramore	0.076146	10

Since, Content based recommendation recommends song based on the user listen history each user gets a unique recommendation. It could be a good recommendation system for the user who wants to listen the same set of songs again and again.

Results of Collaborative filtering:

Results based on the song ‘I believe in miracles’



```
The recommendations for I believe in miracles are:
['Nine Million Bicycles', 'If You Were A Sailboat', 'Shy Boy', 'I Cried For You', "Spider's Web", 'Piece By Piece', 'On The Road Again', 'Blues In The Night',
'Blue Shoes', 'Thank You Stars']
```

Since, Collaborative filtering using knn and fetches results based on the other users who has listened to the same song, this method will provide personalized user experience to the users of the system.

9. Conclusion

In this paper we have built a recommendation system, which users popularity, content and collaborative filtering to recommend personalized songs to users

In the future we would like to try the following things

1.Using audio signal to recommend songs

2.Trying convolutional neutral network

3.Making the recommender system a real-time system

Designing a personalized music recommender is complicated, and its challenging to thoroughly understand the users’ needs and meet their requirements. As discussed above, the future research direction will be mainly focused on user-centric music recommender systems. In the end, we are hoping that through this study , we can build the bridge among isolated research in all other disciplines.

10. Github Repository Link (where your j comp project work can be seen for assessment)

https://github.com/sanjil2/Music-Recommendation-System

REFERENCES

- Euge Inzaugrat,(2020). Music Recommendation System using collaborative Filtering
- Shefali gard and Fangyan(2014).Music Recommender System
- Reneta Rosa, Grace Bressen,(2015) Music Recommendations based on user sentiments
- Govind P.Wakure and Vijayalaxmi Kadrolli,Terna Engineerng College(2013) Music Recommendation System using Genetic Algorithm
- Vuong Khuat,(2014) Music Recommendation Using Collaborative Filtering, Portfolios.cs
- Euge Inzaugarat,(2020), Content Based Music Recommendations System, TowardsDatascience
- Yaue Hu and Mitsunori, Ogihara Science University of Mianu(2015), A music recommendation system based on user behavior