DATA ANALYTICS & VISUALISATION

REVIEW:02

MOVIE RECOMMENDER SYSTEM

SUBMITTED BY:

ABHYUDAY SINGH MANDLOI(18BLC1159)

PRERIT AGGRAWAL(18BLC1156)

1.) Imported Libraries:

a.) **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.

b.) **pandas:** for data manipulation and analysis

c.) **seaborn:** *Seaborn* is a Python data visualization library based on matplotlib. It provides a high-level interface for drawing attractive and informative statistical graphics.

d.) **cosine_similarity:** Cosine similarity, or the cosine kernel, computes similarity as the normalized dot product of X and Y:

```
#  Data Manupulation
import numpy as np
import pandas as pd
# Plotting graphs
import matplotlib.pyplot as plt
import seaborn as sns
from scipy import sparse
from sklearn.metrics.pairwise import cosine_similarity
```

2.) Loading DATA for dataset ratings.csv

In [129]:
```
ratings=pd.read_csv("ratings.csv")
```

In [130]:
```
ratings.head()
```

Out[130]:

	userId	movieId	rating	timestamp
0	1	1	4.0	964982703
1	1	3	4.0	964981247
2	1	6	4.0	964982224
3	1	47	5.0	964983815
4	1	50	5.0	964982931

3.) Merging movie ID with movie name from movies.csv file:

In [131]:
```
ratings=pd.read_csv('ratings.csv')
movies=pd.read_csv('movies.csv')
ratings=pd.merge(movies,ratings)
ratings.head()
```

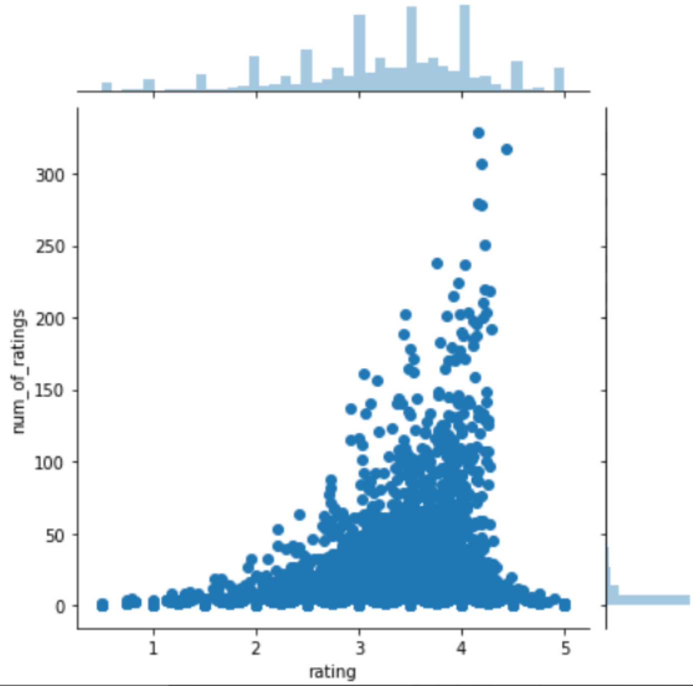
Out[131]:

	movieId	title	genres	userId	rating	timestamp
0	1	Toy Story (1995)	Adventure|Animation|Children|Comedy|Fantasy	1	4.0	964982703
1	1	Toy Story (1995)	Adventure|Animation|Children|Comedy|Fantasy	5	4.0	847434962
2	1	Toy Story (1995)	Adventure|Animation|Children|Comedy|Fantasy	7	4.5	1106635946
3	1	Toy Story (1995)	Adventure|Animation|Children|Comedy|Fantasy	15	2.5	1510577970
4	1	Toy Story (1995)	Adventure|Animation|Children|Comedy|Fantasy	17	4.5	1305696483

4.) Using Seaborn to plot rating vs total num of rating as scatter ploting:

```
In [135]: sns.jointplot(x='rating',y='num_of_ratings',data=rating)
```

```
Out[135]: <seaborn.axisgrid.JointGrid at 0x1f9fe0de748>
```



5.) Pivoting user id as row & movie as column of matrix with rating of movie by particular user:

```
In [136]: movie_matrix=ratings.pivot_table(index='userId',columns='title',values='rating')
movie_matrix.head()
```

Out[136]:

title	'71 (2014)	'Hellboy': The Seeds of Creation (2004)	'Round Midnight (1986)	'Salem's Lot (2004)	'Til There Was You (1997)	'Tis the Season for Love (2015)	'burbs, The (1989)	'night Mother (1986)	(500) Days of Summer (2009)	*batteries not included (1987)	...	Zulu (2013)	[REC] (2007)	[REC]² (2009)	[REC]³ 3 Génesis (2012)	anohana: The Flower We Saw That Day - The Movie (2013)	eXistenZ (1999)
userId																	
1	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN
2	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN
3	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN
4	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN
5	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN

5 rows × 9719 columns

6.) Now converting NILL rating to 0 and droping users with less than 10 reviews:

In [137]:
```
movie_matrix=movie_matrix.dropna(thresh=10,axis=1).fillna(0)
movie_matrix.head()
```

Out[137]:

title	'burbs, The (1989)	(500) Days of Summer (2009)	10 Cloverfield Lane (2016)	10 Things I Hate About You (1999)	10,000 BC (2008)	101 Dalmatians (1996)	101 Dalmatians (One Hundred and One Dalmatians) (1961)	12 Angry Men (1957)	12 Years a Slave (2013)	127 Hours (2010)	...	Zack and Miri Make a Porno (2008)	Zero Dark Thirty (2012)	Zero Effect (1998)	Zodiac (2007)	Zombieland (2009)	Zooland (200
userId																	
1	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0
2	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	3.0	0
3	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0
4	0.0	0.0	0.0	0.0	0.0	0.0	0.0	5.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0
5	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0

5 rows × 2269 columns

7.) Now applying item similarity training method using pearson coefficient , we will apply item to item collaborative filter to obtain similarity values between different movies:

In [138]:
```
item_similarity_df=movie_matrix.corr(method='pearson')
item_similarity_df.head(50)
```

title	'burbs, The (1989)	(500) Days of Summer (2009)	10 Cloverfield Lane (2016)	10 Things I Hate About You (1999)	10,000 BC (2008)	101 Dalmatians (1996)	101 Dalmatians (One Hundred and One Dalmatians) (1961)	12 Angry Men (1957)	12 Years a Slave (2013)	127 Hours (2010)	...	Zack and Miri Make a Porno (2008)	Zero Dark Thirty (2012)	Zero Effect (1998)	Zodiac (2007)	Zo
title																
'burbs, The (1989)	1.000000	0.063117	-0.023768	0.143482	0.011998	0.087931	0.224052	0.034223	0.009277	0.008331	...	0.017477	0.032470	0.134701	0.153158	
(500) Days of Summer (2009)	0.063117	1.000000	0.142471	0.273989	0.193960	0.148903	0.142141	0.159756	0.135486	0.200135	...	0.374515	0.178655	0.068407	0.414585	
10 Cloverfield Lane (2016)	-0.023768	0.142471	1.000000	-0.005799	0.112396	0.006139	-0.016835	0.031704	-0.024275	0.272943	...	0.242663	0.099059	-0.023477	0.272347	
10 Things I																

8.) We defined function "get_recom_movies" which will take movie & its rating by particular user and assign some score according to our given formula:

9.) movie_test is input taking from user as shown below :

10.) running loop for every movie in input data and finding score for each one and then we will merge reccommandation of each movie & sort the final list in descending manner:

11.) Print out the top 60 reccommanded movie to user according to his review on other similar movies:

In [140]:
```python
def get_recom_movies(movie_name,user_rating):
    similar_score=item_similarity_df[movie_name]*(user_rating-2.5)
    similar_score=similar_score.sort_values(ascending=False)

    return similar_score
```

In [146]:
```python
movie_test=[("Jumanji (1995)",3),("Four Rooms (1995)",5),("Othello (1995)",5)]

recom_movies=pd.DataFrame()

for movies,rating in movie_test:
    recom_movies = recom_movies.append(get_recom_movies(movies,rating),ignore_index=True)
print("                    TOP RECOMMANDATION FOR YOU        ")
recom_movies.head()
recom_movies.sum().sort_values(ascending=False).head(60)
```

```
                    TOP RECOMMANDATION FOR YOU
```

Out[146]:
```
Four Rooms (1995)                                         3.079860
Othello (1995)                                            3.075651
Mary Shelley's Frankenstein (Frankenstein) (1994)         1.649036
Frighteners, The (1996)                                   1.603906
Richard III (1995)                                        1.445963
Witches of Eastwick, The (1987)                           1.405030
WarGames (1983)                                           1.395744
Adventures of Baron Munchausen, The (1988)                1.385453
City Hall (1996)                                          1.381131
Muppets, The (2011)                                       1.365413
Master and Commander: The Far Side of the World (2003)    1.347913
Johnny Mnemonic (1995)                                    1.322732
Hot Shots! (1991)                                         1.312350
```

THANK YOU ☺