DCS 630 Predictive Analytics (DSC630-T302 2231-1)

Bellevue University

10.2 Assignment: Recommender System

Author: Jake Meyer

Date: 11/4/2022

Assignment Instructions:

Using the small MovieLens data set, create a recommender system that allows users to input a movie they like (in the data set) and recommends ten other movies for them to watch. In your write-up, clearly explain the recommender system process and all steps performed. If you are using a method found online, be sure to reference the source.
You can use R or Python to complete this assignment. Submit your code and output to the submission link. Make sure to add comments to all of your code and to document your steps, process, and analysis.

Import the Libraries

In [1]:
```
'''
Import the necessary libraries to complete Exercise 10.2.
'''
import numpy as np
import pandas as pd
import seaborn as sns
import scipy.stats
import sklearn
import matplotlib
import matplotlib.pyplot as plt
import matplotlib.dates as mdates

# Ignore warnings throughout the assignment.
import warnings
warnings.filterwarnings('ignore')
```

In [2]:
```
'''
Check the versions of the packages.
'''
print('numpy version:', np.__version__)
print('pandas version:', pd.__version__)
print('seaborn version:', sns.__version__)
```

```
print('matplotlib version:', matplotlib.__version__)
print('sklearn:', sklearn.__version__)
```

```
numpy version: 1.20.3
pandas version: 1.3.4
seaborn version: 0.11.2
matplotlib version: 3.4.3
sklearn: 0.24.2
```

Load the Data

In [3]:
```
'''
Import the dataset.
Note: A copy of the CSV file was placed into the same directory as this notebook.
Utilize pd.read_csv() to read the file as a pandas data frame.
'''
df_movies = pd.read_csv('movies.csv')
df_ratings = pd.read_csv('ratings.csv')
df_tags = pd.read_csv('tags.csv')
df_links = pd.read_csv('links.csv')
```

In [4]:
```
'''
Use head() function to display the first 10 rows of data of df_movies.
'''
df_movies.head()
```

Out[4]:

	movieId	title	genres
0	1	Toy Story (1995)	Adventure|Animation|Children|Comedy|Fantasy
1	2	Jumanji (1995)	Adventure|Children|Fantasy
2	3	Grumpier Old Men (1995)	Comedy|Romance
3	4	Waiting to Exhale (1995)	Comedy|Drama|Romance
4	5	Father of the Bride Part II (1995)	Comedy

In [5]:
```
'''
Use head() function to display the first 10 rows of data of df_ratings.
'''
df_ratings.head()
```

Out[5]:

	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

In [6]:
```
'''
```

```
Use head() function to display the first 10 rows of data of df_tags.
'''
df_tags.head()
```

Out[6]:

	userId	movieId	tag	timestamp
0	2	60756	funny	1445714994
1	2	60756	Highly quotable	1445714996
2	2	60756	will ferrell	1445714992
3	2	89774	Boxing story	1445715207
4	2	89774	MMA	1445715200

In [7]:

```
'''
Use head() function to display the first 10 rows of data of df_links.
'''
df_links.head()
```

Out[7]:

	movieId	imdbId	tmdbId
0	1	114709	862.0
1	2	113497	8844.0
2	3	113228	15602.0
3	4	114885	31357.0
4	5	113041	11862.0

Based on the intitial review of the data files, it appears the two files of interest are movies.csv and ratings.csv. I'll merge the df_movies and df_ratings dataframe by movieId. The other two data files (and dataframes) will not be used moving forward.

In [8]:

```
'''
Merge the df_movies and df_ratings dataframes together.
'''
df = pd.merge(df_movies, df_ratings, on='movieId')
df.head()
```

Out[8]:

	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

In [9]:

```
'''
Remove the unnecessary columns (timestamp) and rearrange the columns in a different ord
```

```
'''
df = df.drop('timestamp', axis=1)
df = df[['userId','movieId','title','rating','genres']]
df.head()
```

Out[9]:

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

In [10]:
```
'''
Show the dataframe after the date has been removed from the title.
'''
df.head()
```

Out[10]:

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

In [11]:
```
'''
Understand the shape of the dataframe.
'''
print('There are {} rows and {} columns in this dataset.'.format(df.shape[0], df.shape[
```

```
There are 100836 rows and 5 columns in this dataset.
```

In [12]:
```
'''
Display the total size of this dataframe.
'''
print('This dataset contains {} records.'.format(df.size))
```

```
This dataset contains 504180 records.
```

In [13]:
```
'''
Understand if there are any missing values in the dataset.
'''
df.isna().sum().sort_values(ascending = False)
```

Out[13]:
```
userId     0
movieId    0
title      0
rating     0
```

```
genres     0
dtype: int64
```

There are no missing values in the dataframe.

In [14]:
```
'''
Summarize the findings above with info().
'''
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 100836 entries, 0 to 100835
Data columns (total 5 columns):
 #   Column   Non-Null Count   Dtype  
---  ------   --------------   -----  
 0   userId   100836 non-null  int64  
 1   movieId  100836 non-null  int64  
 2   title    100836 non-null  object 
 3   rating   100836 non-null  float64
 4   genres   100836 non-null  object 
dtypes: float64(1), int64(2), object(2)
memory usage: 4.6+ MB
```

Data Understanding - Exploratory Data Analysis (EDA)

In [15]:
```
'''
Create a dataframe that includes average rating per movie and number of ratings.
'''
df_rating_num = pd.DataFrame(df.groupby('title')['rating'].mean())
df_rating_num['number_of_ratings'] = pd.DataFrame(df.groupby('title')['rating'].count()
df_rating_num.head()
```

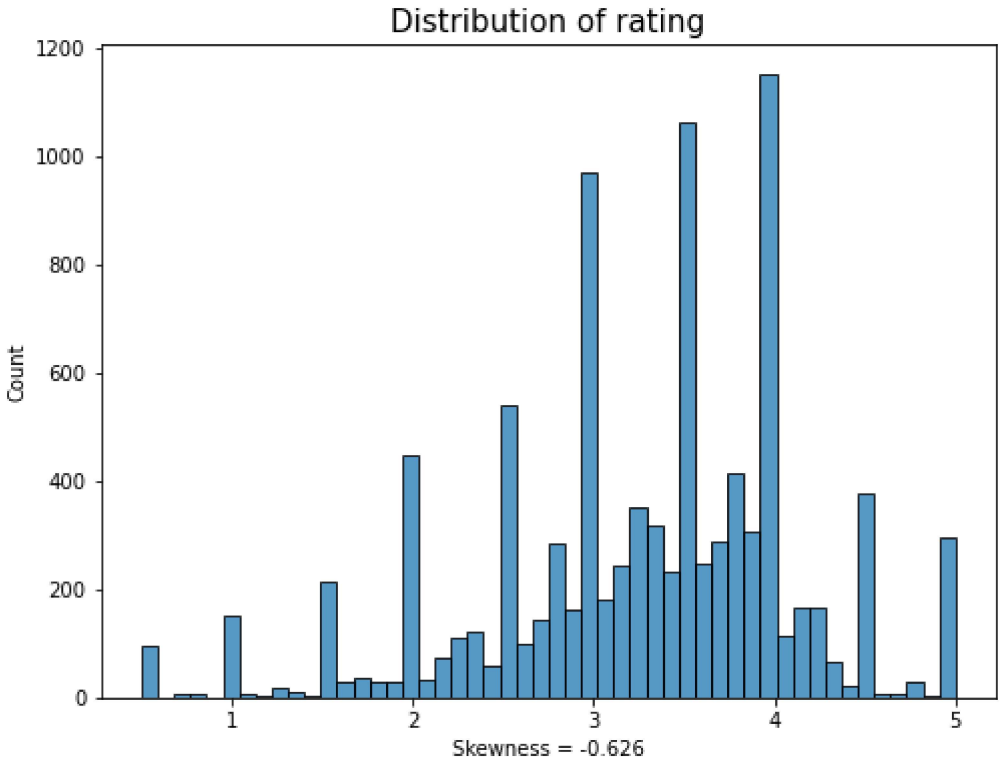
Out[15]:

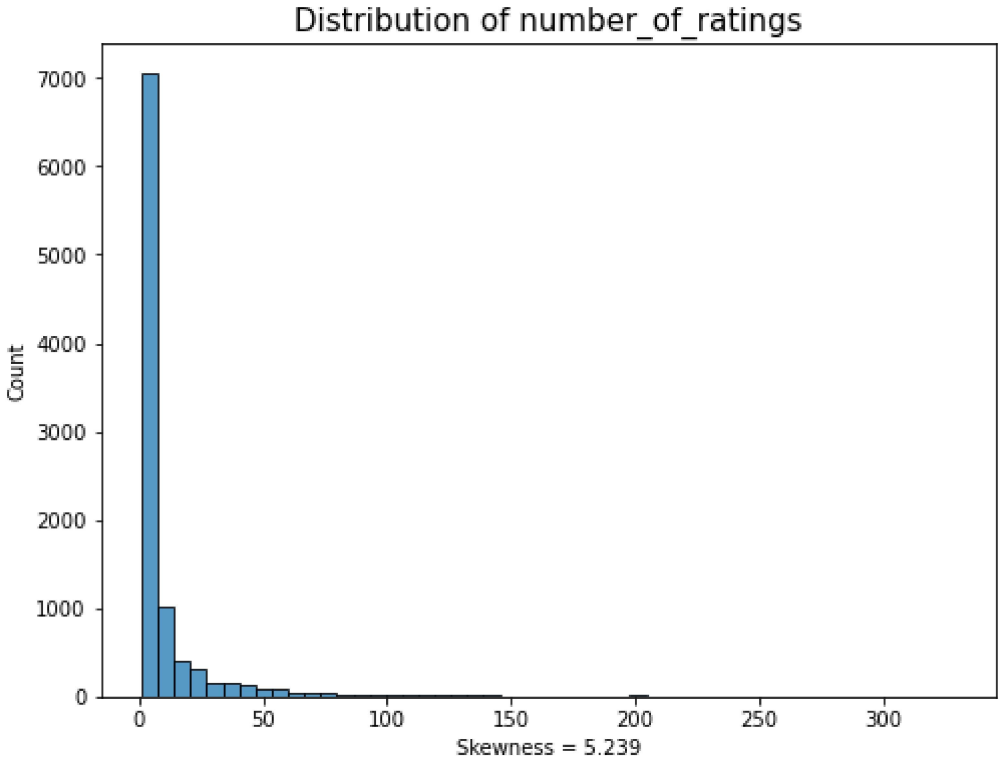
title	rating	number_of_ratings
'71 (2014)	4.0	1
'Hellboy': The Seeds of Creation (2004)	4.0	1
'Round Midnight (1986)	3.5	2
'Salem's Lot (2004)	5.0	1
'Til There Was You (1997)	4.0	2

In [16]:
```
'''
Plot a histogram for both rating and number_of_ratings.
'''
revised_columns = df_rating_num[['rating','number_of_ratings']]

for col in revised_columns:
    plt.figure(figsize=(8,6))
    sns.histplot(x =df_rating_num[col], bins = 50)
    plt.title("Distribution of {}".format(col), fontsize=15)
    plt.xlabel(f"Skewness = {round(df_rating_num[col].skew(),3)}", fontsize=10)
    plt.xticks(fontsize=10)
```

```
    plt.yticks(fontsize=10)
    plt.show()
```

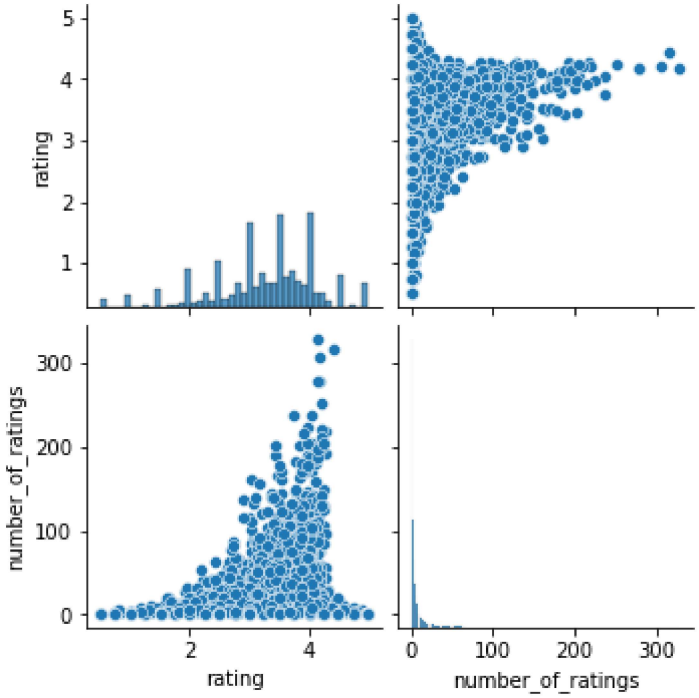




In [17]:
```
'''
Understand how many unique movies are in the dataframe.
'''
df['title'].nunique()
```

Out[17]:
```
9719
```

In [18]:
```
'''
Construct a pairplot() to illustrate bivariate relationships present in the dataset .
'''
sns.pairplot(df_rating_num)
plt.show()
```



Create the Matrix of Movies with Ratings by User ID

In [19]:
```
'''
Create the movie matrix with pivot_table().
'''
movie_matrix = df.pivot_table(index = 'userId', columns = 'title', values = 'rating')
movie_matrix.head()
```

Out[19]:

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)	..
userId											
1	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	..
2	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	..
3	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	..
4	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	..
5	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	..

5 rows × 9719 columns

In [20]:
```
'''
Show the movies that have the most number of ratings.
'''
df_rating_num.sort_values('number_of_ratings', ascending = False).head(20)
```

Out[20]:

title	rating	number_of_ratings
Forrest Gump (1994)	4.164134	329
Shawshank Redemption, The (1994)	4.429022	317
Pulp Fiction (1994)	4.197068	307
Silence of the Lambs, The (1991)	4.161290	279
Matrix, The (1999)	4.192446	278
Star Wars: Episode IV - A New Hope (1977)	4.231076	251
Jurassic Park (1993)	3.750000	238
Braveheart (1995)	4.031646	237
Terminator 2: Judgment Day (1991)	3.970982	224
Schindler's List (1993)	4.225000	220
Fight Club (1999)	4.272936	218
Toy Story (1995)	3.920930	215
Star Wars: Episode V - The Empire Strikes Back (1980)	4.215640	211
Usual Suspects, The (1995)	4.237745	204
American Beauty (1999)	4.056373	204
Seven (a.k.a. Se7en) (1995)	3.975369	203
Independence Day (a.k.a. ID4) (1996)	3.445545	202
Apollo 13 (1995)	3.845771	201
Raiders of the Lost Ark (Indiana Jones and the Raiders of the Lost Ark) (1981)	4.207500	200
Lord of the Rings: The Fellowship of the Ring, The (2001)	4.106061	198

Request a Movie from User

In [37]:
```
'''
Show a list of the 30 most popular movies for the user to review.
'''
df['title'].value_counts()[0:31]
```

```
Forrest Gump (1994)                                                329
```

Out[37]:
```
Shawshank Redemption, The (1994)                                                317
Pulp Fiction (1994)                                                             307
Silence of the Lambs, The (1991)                                                279
Matrix, The (1999)                                                              278
Star Wars: Episode IV - A New Hope (1977)                                       251
Jurassic Park (1993)                                                            238
Braveheart (1995)                                                               237
Terminator 2: Judgment Day (1991)                                               224
Schindler's List (1993)                                                         220
Fight Club (1999)                                                               218
Toy Story (1995)                                                                215
Star Wars: Episode V - The Empire Strikes Back (1980)                           211
American Beauty (1999)                                                          204
Usual Suspects, The (1995)                                                      204
Seven (a.k.a. Se7en) (1995)                                                     203
Independence Day (a.k.a. ID4) (1996)                                            202
Apollo 13 (1995)                                                                201
Raiders of the Lost Ark (Indiana Jones and the Raiders of the Lost Ark) (1981)  200
Lord of the Rings: The Fellowship of the Ring, The (2001)                       198
Star Wars: Episode VI - Return of the Jedi (1983)                               196
Godfather, The (1972)                                                           192
Fugitive, The (1993)                                                            190
Batman (1989)                                                                   189
Lord of the Rings: The Two Towers, The (2002)                                   188
Saving Private Ryan (1998)                                                      188
Lord of the Rings: The Return of the King, The (2003)                           185
Aladdin (1992)                                                                  183
Fargo (1996)                                                                    181
Sixth Sense, The (1999)                                                         179
True Lies (1994)                                                                178
Name: title, dtype: int64
```

In [47]:
```
'''
Use input() to gain movie title from user.
Obtain the ratings for the movie chosen by the user.
Use error handling if the user inputs a movie not in the movie_matrix.
'''
while True:
    try:
        movie = input("Enter Movie Name Followed by Year\n Example: Jurassic Park (1993
        movie_user_ratings = movie_matrix['{}'.format(movie)]
        break
    except KeyError:
        print('Sorry, that movie is not in our records. Please input another movie:')
```

```
Enter Movie Name Followed by Year
 Example: Jurassic Park (1993):
Braveheart (1995)
```

Identify Movies Correlated with Selected Movie

In [48]:
```
'''
Determine which movies are highly correlated with the movie chosen by the user.
'''
similar_movies = movie_matrix.corrwith(movie_user_ratings)
```

Show 10 Additional Recommendations for Movies with Greater

than 50 Reviews

In [49]:
```
'''
Create a dataframe showing the top 10 movies recommended based on the movie chosen by t
Ensure the NaN values are dropped out of the dataframe and also include number_of_ratin
'''

corr_movie = pd.DataFrame(similar_movies, columns = ['correlation'])
corr_movie.dropna(inplace=True)
corr_movie = corr_movie.join(df_rating_num['number_of_ratings'])
corr_movie[corr_movie['number_of_ratings']>50].sort_values('correlation',ascending = Fa
```

Out[49]:

title	correlation	number_of_ratings
Field of Dreams (1989)	0.682274	56
Mystic River (2003)	0.644782	52
Grumpier Old Men (1995)	0.636963	52
Kung Fu Panda (2008)	0.625232	54
Batman Begins (2005)	0.610550	116
Guardians of the Galaxy (2014)	0.604996	59
Top Gun (1986)	0.600929	83
Rocky (1976)	0.599043	64
Dark Knight Rises, The (2012)	0.598218	76
Wolf of Wall Street, The (2013)	0.583161	54

The above process utilized correlation to understand the relationship between movies based on user ratings. Once the user inputs a movie that is in the dataset, then the code will generate a correlation dataframe showing the top 10 movies correlated with the "movie of interest" as long as there are over 50 ratings. Another method that could have been utilized could have been through Cosine similarity, however the correlation method was chosen for this assignment. A recommendation to improve this code in the future is to create functions to streamline the recommendation system for the user. Also, a loop could be created in the event the user would like to continue using the recommendation system.

Reference

The method followed in this code is based from Krish Naik's approach from the You Tube Video "Movie Recommender System using Python". The reference is documented below:

Naik, K. (2019, November). Movie Recommender System Using Python. [Video]. YouTube.https://youtu.be/R64Lh1Qwl_0