9.2 Exercise: Recommender System

In [1]:
```
# loading libraries
import numpy as np
import pandas as pd
```

In [2]:
```
# surpressing warnings
import warnings
warnings.filterwarnings('ignore')
```

In [3]:
```
# loading the ratings data
ratings = pd.read_csv('Datasets/ml-latest-small/ratings.csv')
ratings.head(5)
```

Out[3]:

	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 [4]:
```
# loading the movie data
movie_titles_genre = pd.read_csv("Datasets/ml-latest-small/movies.csv")
movie_titles_genre.head(5)
```

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]:
```
# merging the two data frames on the movieid feature
df = ratings.merge(movie_titles_genre,on='movieId', how='left')
df.head(10)
```

Out[5]:

	userId	movieId	rating	timestamp	title	genres
0	1	1	4.0	964982703	Toy Story (1995)	Adventure|Animation|Children|Comedy|Fantasy
1	1	3	4.0	964981247	Grumpier Old Men (1995)	Comedy|Romance
2	1	6	4.0	964982224	Heat (1995)	Action|Crime|Thriller
3	1	47	5.0	964983815	Seven (a.k.a. Se7en) (1995)	Mystery|Thriller
4	1	50	5.0	964982931	Usual Suspects, The (1995)	Crime|Mystery|Thriller
5	1	70	3.0	964982400	From Dusk Till Dawn (1996)	Action|Comedy|Horror|Thriller
6	1	101	5.0	964980868	Bottle Rocket (1996)	Adventure|Comedy|Crime|Romance
7	1	110	4.0	964982176	Braveheart (1995)	Action|Drama|War
8	1	151	5.0	964984041	Rob Roy (1995)	Action|Drama|Romance|War
9	1	157	5.0	964984100	Canadian Bacon (1995)	Comedy|War

In [6]:

```
# calculating the average rating by movie and creating a new data frame
Average_ratings = pd.DataFrame(df.groupby('title')['rating'].mean())
Average_ratings.head(10)
```

Out[6]:

	rating
title	
'71 (2014)	4.000000
'Hellboy': The Seeds of Creation (2004)	4.000000
'Round Midnight (1986)	3.500000
'Salem's Lot (2004)	5.000000
'Til There Was You (1997)	4.000000
'Tis the Season for Love (2015)	1.500000
'burbs, The (1989)	3.176471
'night Mother (1986)	3.000000
(500) Days of Summer (2009)	3.666667
***batteries not included (1987)**	3.285714

In [7]:

```
# creating a total ratings feature column, which is the count of how many times a movie was ratec
Average_ratings['Total Ratings'] = pd.DataFrame(df.groupby('title')['rating'].count())
Average_ratings.head(10)
```

Out[7]:

title	rating	Total Ratings
'71 (2014)	4.000000	1
'Hellboy': The Seeds of Creation (2004)	4.000000	1
'Round Midnight (1986)	3.500000	2
'Salem's Lot (2004)	5.000000	1
'Til There Was You (1997)	4.000000	2
'Tis the Season for Love (2015)	1.500000	1
'burbs, The (1989)	3.176471	17
'night Mother (1986)	3.000000	1
(500) Days of Summer (2009)	3.666667	42
*batteries not included (1987)	3.285714	7

In [8]:

```
# creates a table of users with each column being how the user rated the movie
movie_user = df.pivot_table(index='userId',columns='title',values='rating')
```

In [9]:

```
# displaying for 10 rows
movie_user.head(10)
```

Out[9]:

title / userId	'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)	...
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	...
6	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...
7	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...
8	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...
9	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...
10	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...

10 rows × 9719 columns

In [10]:
```
# getting correlation values for movies with Toy Story being the movie tested against
correlations = movie_user.corrwith(movie_user['Toy Story (1995)'])
# displaying first 5 rows
correlations.head()
```

Out[10]:
```
title
'71 (2014)                                   NaN
'Hellboy': The Seeds of Creation (2004)      NaN
'Round Midnight (1986)                       NaN
'Salem's Lot (2004)                          NaN
'Til There Was You (1997)                    NaN
dtype: float64
```

In [11]:
```
# creating a correlation column
recommendation = pd.DataFrame(correlations,columns=['Correlation'])
# dropping NaN values
recommendation.dropna(inplace=True)
# joining the correlation with total ratings
recommendation = recommendation.join(Average_ratings['Total Ratings'])
# displaying first 5 rows
recommendation.head()
```

Out[11]:

title	Correlation	Total Ratings
'burbs, The (1989)	0.240563	17
(500) Days of Summer (2009)	0.353833	42
***batteries not included (1987)**	-0.427425	7
10 Cent Pistol (2015)	1.000000	2
10 Cloverfield Lane (2016)	-0.285732	14

In [12]:
```
# get recommendations for
recc = recommendation[recommendation['Total Ratings']>100].sort_values('Correlation',ascending=Fa
# merge the movies dataset for verifying the recommendations
recc = recc.merge(movie_titles_genre,on='title', how='left')
recc.head(10)
```

Out[12]:

	title	Correlation	Total Ratings	movieId	genres
0	Toy Story (1995)	1.000000	215	1	Adventure|Animation|Children|Comedy|Fantasy
1	Incredibles, The (2004)	0.643301	125	8961	Action|Adventure|Animation|Children|Comedy
2	Finding Nemo (2003)	0.618701	141	6377	Adventure|Animation|Children|Comedy
3	Aladdin (1992)	0.611892	183	588	Adventure|Animation|Children|Comedy|Musical
4	Monsters, Inc. (2001)	0.490231	132	4886	Adventure|Animation|Children|Comedy|Fantasy
5	Mrs. Doubtfire (1993)	0.446261	144	500	Comedy|Drama
6	Amelie (Fabuleux destin d'Amélie Poulain, Le) ...	0.438237	120	4973	Comedy|Romance
7	American Pie (1999)	0.420117	103	2706	Comedy|Romance
8	Die Hard: With a Vengeance (1995)	0.410939	144	165	Action|Crime|Thriller
9	E.T. the Extra-Terrestrial (1982)	0.409216	122	1097	Children|Drama|Sci-Fi

In [13]:

```
# I decided to put the code previously walked through into a function so more easily call it in t
def movie_rec(movie):
    if movie in movie_titles_genre.values:
        correlations = movie_user.corrwith(movie_user[movie])
        recommendation = pd.DataFrame(correlations,columns=['Correlation'])
        recommendation.dropna(inplace=True)
        recommendation = recommendation.join(Average_ratings['Total Ratings'])
        recc = recommendation[recommendation['Total Ratings']>100].sort_values('Correlation',asce
        recc = recc.merge(movie_titles_genre,on='title', how='left')
        top10 = recc['title'].iloc[1:11].tolist()
        print(f'Your movie recommendations are: {top10}')
    else:
        print('Not found, be sure to include movie year with parenthesis\nExample: Jumanji (1995
```

In [14]:

```
# Taking user input to look up the movie and return the top 10 recommendations
movie = input('Please enter the title of a Movie: ')
movie_rec(movie)
```

```
Your movie recommendations are: ['Twister (1996)', 'Outbreak (1995)', 'Harry Potter and the Chamb
er of Secrets (2002)', 'Finding Nemo (2003)', "Harry Potter and the Sorcerer's Stone (a.k.a. Harr
y Potter and the Philosopher's Stone) (2001)", 'Jumanji (1995)', 'Home Alone (1990)', 'Spider-Man
(2002)', 'Toy Story (1995)', 'Monsters, Inc. (2001)']
```

For this assignment I used the additional resourced in Blackbaord to help guide me through the process of making a recommender system, specifically I used the article *How To Build Your First Recommender System Using Python & MovieLens Dataset* and it can be found here: ' https://analyticsindiamag.com/ai-mysteries/how-to-build-your-first-recommender-system-using-python-movielens-dataset'.

Following the steps from the guide made it fairly easy to understand the overall goal and process of the recommender system.

We started with importing the movie data and ratings data and then merged them into a single data frame on the 'movieid' feature. We then created an 'averagerating' feature for the data set as well as a total ratings feature since the average rating is proprtionasl to how many times a movei ahs been rated. An example being if a movie is rated only a single time the rating may not be a good reflection of the movie since only a signle person has rated it.

Next, the recommender system was built starting with calcualting the correlation. This was done by creating a table with rows being the users and columns being the movies, the values would be the ratings by user per movie. The corrwith() function was used to compute the pairwise correlation between the rows and columns of the two data frames, then NaN values were theen dropped. The correlation columns were merged into the overall data frame.

The reccomendations were generated by filtering the main data frame by total ratings over 100 ratings and sorting values by correlation. The movie titles were also merged into the dataset to get the titles of the recommended movies.

I ended up encapsulating all these steps into a function for ease of use. I created an input field to get user input that requests the user to enter a movie title to generate some recommendations. I was sure to exclude the movie entered as a recommendation since the correlation would have returned 1.00. /