Movie Recommender System

September 7, 2019

[2]:
```
#Importing Pandas and Numpy
import pandas as pd
import numpy as np
import warnings
warnings.filterwarnings('ignore')
#reading rating dataset
rating = pd.read_csv('/home/sakil/Desktop/DataScience/Project/ml-latest-small/
 ↪ratings.csv')
rating.head()
```

[2]:

	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

[4]:
```
#reading movies data
movie = pd.read_csv('/home/sakil/Desktop/DataScience/Project/ml-latest-small/
 ↪movies.csv')
movie.head()
```

[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

[5]:
```
moviedata=pd.merge(rating,movie,on="movieId")
moviedata.head()
```

[5]:
```
   userId  movieId  rating   timestamp            title  \
0       1        1     4.0   964982703  Toy Story (1995)
1       5        1     4.0   847434962  Toy Story (1995)
2       7        1     4.5  1106635946  Toy Story (1995)
3      15        1     2.5  1510577970  Toy Story (1995)
4      17        1     4.5  1305696483  Toy Story (1995)

                                        genres
0  Adventure|Animation|Children|Comedy|Fantasy
1  Adventure|Animation|Children|Comedy|Fantasy
2  Adventure|Animation|Children|Comedy|Fantasy
3  Adventure|Animation|Children|Comedy|Fantasy
4  Adventure|Animation|Children|Comedy|Fantasy
```

[7]:
```
#removing genres from moviedata
moviedataset_col=["userId","movieId","rating","timestamp","title"]
moviedataset=moviedata[moviedataset_col]
moviedataset.head()
```

[7]:
```
   userId  movieId  rating   timestamp            title
0       1        1     4.0   964982703  Toy Story (1995)
1       5        1     4.0   847434962  Toy Story (1995)
2       7        1     4.5  1106635946  Toy Story (1995)
3      15        1     2.5  1510577970  Toy Story (1995)
4      17        1     4.5  1305696483  Toy Story (1995)
```

[41]:
```
#writing moviedataset to moviedatasetwritetofile,it automatically creates file␣
 ↪moviepreprocesseddata and writes all
#values of moviedataset to moviepreprocesseddata
moviedatasetwritetofile=moviedataset.to_csv('/home/sakil/Desktop/DataScience/
 ↪Project/ moviepreprocesseddata.csv')
moviedatasetwritetofile
```

[8]:
```
#now we have dataset"moviedataset",we have done preprocessing part now we have␣
 ↪to go for other steps using this dataset
moviedataset.shape
#moviedataset has 100836 rows and 5 columns
```

[8]:
```
(100836, 5)
```

[9]:
```
#to know about more dataset
moviedataset.describe()
#The movie dataset has 100836 recors,average rating is 3.50 and max rating is 5
```

[9]:
```
              userId        movieId         rating     timestamp
count  100836.000000  100836.000000  100836.000000  1.008360e+05
mean      326.127564   19435.295718       3.501557  1.205946e+09
std       182.618491   35530.987199       1.042529  2.162610e+08
min         1.000000       1.000000       0.500000  8.281246e+08
25%       177.000000    1199.000000       3.000000  1.019124e+09
```

```
50%       325.000000    2991.000000       3.500000  1.186087e+09
75%       477.000000    8122.000000       4.000000  1.435994e+09
max       610.000000  193609.000000       5.000000  1.537799e+09
```

[10]:
```
#We group the dataset by the title column and compute its mean to obtain the␣
 ↪average rating for each movie.
ratings = pd.DataFrame(moviedataset.groupby('title')['rating'].mean())
ratings.head()
```

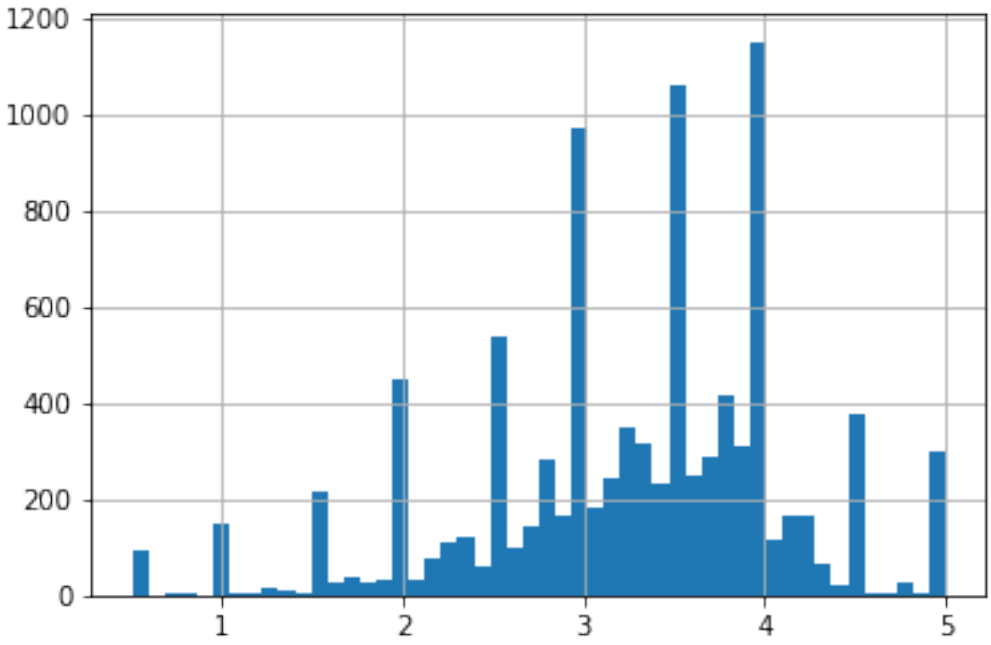
[10]:
```
                                                rating
title
'71 (2014)                                         4.0
'Hellboy': The Seeds of Creation (2004)            4.0
'Round Midnight (1986)                             3.5
'Salem's Lot (2004)                                5.0
'Til There Was You (1997)                          4.0
```

[11]:
```
#Next we would like to see the number of ratings for each movie.
#We do this by creating a number_of_ratings column.
ratings['number_of_ratings'] = moviedataset.groupby('title')['rating'].count()
ratings.head()
```

[11]:
```
                                                rating  number_of_ratings
title
'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
```

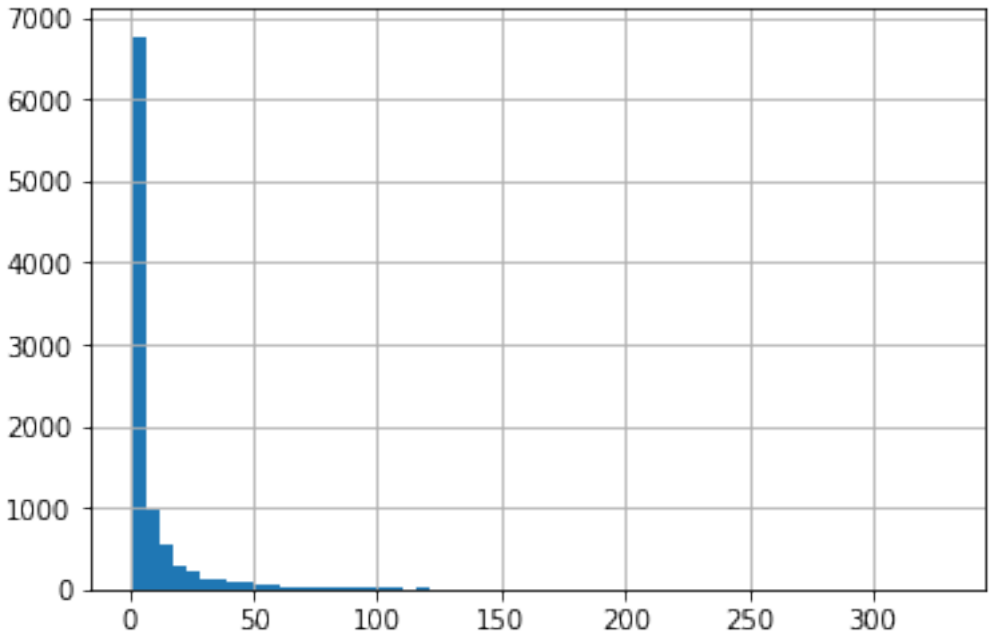
[12]:
```
#now plotting a Histogram using pandas plotting functionality to visualize the␣
 ↪distribution of the ratings
import matplotlib.pyplot as plt
%matplotlib inline
ratings['rating'].hist(bins=50)
```

[12]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7fad88d402b0>
```



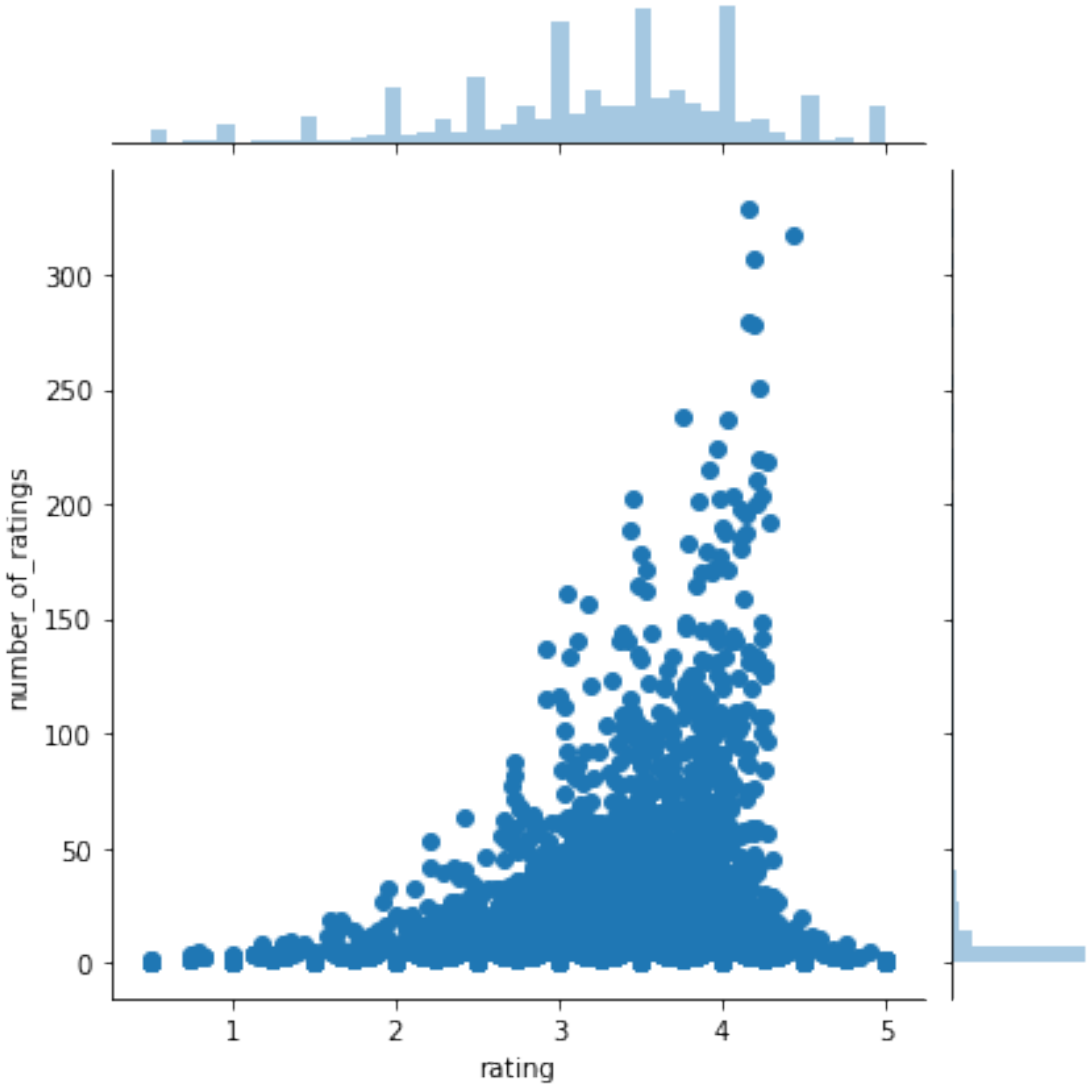
[13]:
```
#visualize the number_of_ratings column in as similar manner.
ratings['number_of_ratings'].hist(bins=60)
```

[13]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7fad898e8278>
```



[14]:
```
#now checking the relationship between the rating of a movie and the number of␣
 ↪ratings
#We do this by plotting a scatter plot using seaborn.
#Seaborn enables us to do this using the jointplot() function.
import seaborn as sns
sns.jointplot(x='rating', y='number_of_ratings', data=ratings)
```

[14]:
```
<seaborn.axisgrid.JointGrid at 0x7fad888a6d30>
```



[18]:
```
#In order to do this we need to convert our dataset into a matrix with the␣
 ↪movie titles as the columns,
#the user_id as the index and the ratings as the values.
#By doing this we shall get a dataframe with the columns as the movie titles␣
 ↪and the rows as the user ids.
```

```
#Each column represents all the ratings of a movie by all users.
# The rating appear as NAN where a user didn't rate a certain movie.
#We shall use this matrix to compute the correlation between the ratings of a␣
 ↪single movie and the rest of the movies in the matrix.
#We use pandas pivot_table utility to create the movie matrix.
moviedataset_matrix = moviedataset.pivot_table(index='userId', columns='title',␣
 ↪values='rating')
moviedataset_matrix.head()
```

[18]:
```
title   '71 (2014)  'Hellboy': The Seeds of Creation (2004)  \
userId
1              NaN                                     NaN
2              NaN                                     NaN
3              NaN                                     NaN
4              NaN                                     NaN
5              NaN                                     NaN

title   'Round Midnight (1986)  'Salem's Lot (2004)  \
userId
1                          NaN                 NaN
2                          NaN                 NaN
3                          NaN                 NaN
4                          NaN                 NaN
5                          NaN                 NaN

title   'Til There Was You (1997)  'Tis the Season for Love (2015)  \
userId
1                             NaN                              NaN
2                             NaN                              NaN
3                             NaN                              NaN
4                             NaN                              NaN
5                             NaN                              NaN

title   'burbs, The (1989)  'night Mother (1986)  (500) Days of Summer (2009)  \
userId
1                      NaN                   NaN                          NaN
2                      NaN                   NaN                          NaN
3                      NaN                   NaN                          NaN
4                      NaN                   NaN                          NaN
5                      NaN                   NaN                          NaN

title   *batteries not included (1987)  ...  Zulu (2013)  [REC] (2007)  \
userId                                  ...
1                                  NaN  ...          NaN           NaN
2                                  NaN  ...          NaN           NaN
3                                  NaN  ...          NaN           NaN
4                                  NaN  ...          NaN           NaN
```

```
5                                   NaN  ...          NaN           NaN

title   [REC]š (2009)  [REC]ş 3 Génesis (2012)  \
userId
1                 NaN                      NaN
2                 NaN                      NaN
3                 NaN                      NaN
4                 NaN                      NaN
5                 NaN                      NaN

title   anohana: The Flower We Saw That Day - The Movie (2013)  \
userId
1                                                     NaN
2                                                     NaN
3                                                     NaN
4                                                     NaN
5                                                     NaN

title   eXistenZ (1999)  xXx (2002)  xXx: State of the Union (2005)  \
userId
1                   NaN         NaN                            NaN
2                   NaN         NaN                            NaN
3                   NaN         NaN                            NaN
4                   NaN         NaN                            NaN
5                   NaN         NaN                            NaN

title   ąThree Amigos! (1986)  À nous la liberté (Freedom for Us) (1931)
userId
1                         4.0                                        NaN
2                         NaN                                        NaN
3                         NaN                                        NaN
4                         NaN                                        NaN
5                         NaN                                        NaN

[5 rows x 9719 columns]
```

[19]:
```
# look at the most rated movies and choose two of them to work with in this␣
 ↪simple recommender system.
#We use pandas sort_values utility and set ascending to false in order to␣
 ↪arrange the movies from the most rated.
#We then use the head() function to view the top 10.
ratings.sort_values('number_of_ratings', ascending=False).head(10)
```

[19]:
```
                                      rating  number_of_ratings
title
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
```

[22]:
```
#I am choosing "Forrest Gump (1994)" and "Shawshank Redemption, The (1994)"␣
 ↪movies.
#We would like like to recommend movies to this user based on this watching␣
 ↪history.
#The goal is to look for movies that are similar to Forrest Gump (1994) and␣
 ↪Shawshank Redemption, The (1994) which we shall recommend
#to this user.
#We can achieve this by computing the correlation between these two movies␣
 ↪ratings
#and the ratings of the rest of the movies in the dataset.
# The first step is to create a dataframe with the ratings of these movies from␣
 ↪our moviedataset_matrix.
#Forrest Gump (1994) user rating
FG_user_rating = moviedataset_matrix['Forrest Gump (1994)']
FG_user_rating.head(10)
```

[22]:
```
userId
1     4.0
2     NaN
3     NaN
4     NaN
5     NaN
6     5.0
7     5.0
8     3.0
9     NaN
10    3.5
Name: Forrest Gump (1994), dtype: float64
```

[23]:
```
#Shawshank Redemption, The (1994) user rating
SR_user_rating = moviedataset_matrix['Shawshank Redemption, The (1994)']
SR_user_rating.head(10)
```

[23]:
```
userId
1     NaN
2     3.0
3     NaN
4     NaN
5     3.0
6     5.0
7     NaN
```

```
8     5.0
9     NaN
10    NaN
Name: Shawshank Redemption, The (1994), dtype: float64
```

[26]:
```
#In order to compute the correlation between two dataframes we use pandas␣
 ↪corwith functionality.
#Corrwith computes the pairwise correlation of rows or columns of two dataframe␣
 ↪objects.
#Let's use this functionality to get the correlation between each movie's␣
 ↪rating
#and the ratings of the Forrest Gump (1994) movie.
similar_to_forest_gump=moviedataset_matrix.corrwith(FG_user_rating)
similar_to_forest_gump.head(10)
#We can see that the correlation between Forrest Gump (1994) movie and␣
 ↪batteries not included (1987) is 0.8927. This indicates a very strong␣
 ↪similarity between these two movies.
```

[26]:
```
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
'Tis the Season for Love (2015)               NaN
'burbs, The (1989)                       0.197712
'night Mother (1986)                          NaN
(500) Days of Summer (2009)              0.234095
*batteries not included (1987)           0.892710
dtype: float64
```

[30]:
```
#Lets move on and compute the correlation between Shawshank Redemption, The␣
 ↪(1994) ratings and the rest of the movies ratings.
#The procedure is the same as the one used above.
similar_to_Shawshank_Redemption  = moviedataset_matrix.corrwith(SR_user_rating)
similar_to_Shawshank_Redemption.head(15)
#We realize from the computation that there is a correlation (of 0.419) between␣
 ↪Shawshank Redemption, The (1994)
#burbs, The (1989)
```

[30]:
```
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
'Tis the Season for Love (2015)               NaN
'burbs, The (1989)                       0.419543
'night Mother (1986)                          NaN
```

```
(500) Days of Summer (2009)                    0.249580
*batteries not included (1987)                 0.404520
...All the Marbles (1981)                           NaN
...And Justice for All (1979)                 -1.000000
00 Schneider - Jagd auf Nihil Baxter (1994)         NaN
1-900 (06) (1994)                                   NaN
10 (1979)                                           NaN
dtype: float64
```

[32]:
```
#As noticed earlier our matrix had very many missing values since not all the␣
 ↪movies were rated by all the users.
#We therefore drop those null values and transform correlation results into␣
 ↪dataframes to make the
#results look more appealing.
corr_FG= pd.DataFrame(similar_to_forest_gump, columns=['Correlation'])
corr_FG.dropna(inplace=True)
corr_FG.head()
```

[32]:
```
                                Correlation
title
'burbs, The (1989)                 0.197712
(500) Days of Summer (2009)        0.234095
*batteries not included (1987)     0.892710
...And Justice for All (1979)      0.928571
10 Cent Pistol (2015)             -1.000000
```

[33]:
```
corr_SR= pd.DataFrame(similar_to_Shawshank_Redemption, columns=['Correlation'])
corr_SR.dropna(inplace=True)
corr_SR.head()
```

[33]:
```
                                Correlation
title
'burbs, The (1989)                 0.419543
(500) Days of Summer (2009)        0.249580
*batteries not included (1987)     0.404520
...And Justice for All (1979)     -1.000000
10 Cloverfield Lane (2016)         0.145671
```

[34]:
```
#These two dataframes above show us the movies that are most similar
#to Forrest Gump (1994) and Shawshank Redemption, The (1994) movies␣
 ↪respectively.
#However we have a challenge in that some of the movies have very few ratings␣
 ↪and may end up being recommended simply because
#one or two people gave them a 5 star rating.
#We can fix this by setting a threshold for the number of ratings.
#From the histogram earlier we saw a sharp decline in number of ratings from␣
 ↪100.
#We shall therefore set this as the threshold, however this is a number you can␣
 ↪play around with until you get a suitable option.
```

```
#In order to do this we need to join the two dataframes with the␣
 ↪number_of_ratings column in the ratings dataframe.

corr_FG = corr_FG.join(ratings['number_of_ratings'])
corr_FG.head()
```

[34]:

title	Correlation	number_of_ratings
'burbs, The (1989)	0.197712	17
(500) Days of Summer (2009)	0.234095	42
*batteries not included (1987)	0.892710	7
...And Justice for All (1979)	0.928571	3
10 Cent Pistol (2015)	-1.000000	2

[35]:
```
corr_SR = corr_SR.join(ratings['number_of_ratings'])
corr_SR.head()
```

[35]:

title	Correlation	number_of_ratings
'burbs, The (1989)	0.419543	17
(500) Days of Summer (2009)	0.249580	42
*batteries not included (1987)	0.404520	7
...And Justice for All (1979)	-1.000000	3
10 Cloverfield Lane (2016)	0.145671	14

[37]:
```
#We shall now obtain the movies that are most similar to Forrest Gump (1994) by␣
 ↪limiting them
#to movies that have at least 100 reviews.
#We then sort them by the correlation column and view the first 10.
corr_FG[corr_FG['number_of_ratings'] > 100].sort_values(by='Correlation',␣
 ↪ascending=False).head(10)
```

[37]:

title	Correlation	number_of_ratings
Forrest Gump (1994)	1.000000	329
Good Will Hunting (1997)	0.484042	141
Aladdin (1992)	0.464268	183
American History X (1998)	0.457287	129
Truman Show, The (1998)	0.432556	125
Braveheart (1995)	0.416976	237
Ferris Bueller's Day Off (1986)	0.405830	109
Mrs. Doubtfire (1993)	0.401408	144
Full Metal Jacket (1987)	0.397241	102
Saving Private Ryan (1998)	0.390074	188

[ ]:
```
#We notice that Forrest Gump (1994)        has a perfect correlation with␣
 ↪itself, which is not surprising.
#The next most similar movie to Air Force One (1997) is Good Will Hunting␣
 ↪(1997) with a correlation of 0.484.
```

```
#Clearly by changing the threshold for the number of reviews we get different␣
 ↪results from the previous way of doing it.
#Limiting the number of rating gives us better results and
#we can confidently recommend the above movies to someone who has watched␣
 ↪Forrest Gump (1994).
```

[38]:
```
#Now lets do the same for Shawshank Redemption, The (1994) movie and see the␣
 ↪movies that are most correlated to it
corr_SR[corr_SR['number_of_ratings'] > 100].sort_values(by='Correlation',␣
 ↪ascending=False).head(10)
```

[38]:

title	Correlation	number_of_ratings
Shawshank Redemption, The (1994)	1.000000	317
Four Weddings and a Funeral (1994)	0.446212	103
Schindler's List (1993)	0.402202	220
Usual Suspects, The (1995)	0.394294	204
Ocean's Eleven (2001)	0.391546	119
Green Mile, The (1999)	0.382818	111
Inception (2010)	0.377839	143
Catch Me If You Can (2002)	0.356612	115
One Flew Over the Cuckoo's Nest (1975)	0.354215	133
Godfather: Part II, The (1974)	0.349872	129

[ ]:
```
#Once again we get different results. The most similar movie to Shawshank␣
 ↪Redemption, The (1994) is
#Four Weddings and a Funeral (1994) with a correlation coefficient of 0.446␣
 ↪with 103 ratings.
#So if somebody liked Shawshank Redemption, The (1994) we can recommend the␣
 ↪above movies to them.
```

So we build a simple movie recommender system.

Dataset used in this project: Please find the following URL for dataset download: dataset Readme file for datasetYou can compare your preprocessed data from my preprocessed data named moviepreprocesseddata Preprocessed dataset