movielens

September 24, 2021

[1]:
```python
# import required libraries
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
```

[2]:
```python
# read the data files and save to variable
data_movie=pd.read_csv('movies.dat', sep='::',␣
↪names=['MovieID','Title','Genre'], engine='python')
```

[3]:
```python
data_movie.head()
```

[3]:
```
   MovieID                               Title                         Genre
0        1                    Toy Story (1995)   Animation|Children's|Comedy
1        2                      Jumanji (1995)  Adventure|Children's|Fantasy
2        3             Grumpier Old Men (1995)                Comedy|Romance
3        4            Waiting to Exhale (1995)                  Comedy|Drama
4        5  Father of the Bride Part II (1995)                        Comedy
```

[4]:
```python
data_movie.shape
```

[4]: (3883, 3)

[5]:
```python
data_ratings=pd.read_csv('ratings.dat', sep='::', names=['UserID', 'MovieID',␣
↪'Rating','Timestamp'], engine='python')
```

[6]:
```python
data_ratings.head()
```

[6]:
```
   UserID  MovieID  Rating  Timestamp
0       1     1193       5  978300760
1       1      661       3  978302109
2       1      914       3  978301968
3       1     3408       4  978300275
4       1     2355       5  978824291
```

[7]:
```python
data_ratings.shape
```

[7]: (1000209, 4)

```
[8]: data_users=pd.read_csv('users.dat', sep='::', names=['UserID', 'Gender', 'Age',␣
      ↪'Occupation', 'Zip-code'], engine='python')
```

```
[9]: data_users.head()
```

```
[9]:    UserID Gender  Age  Occupation Zip-code
     0       1      F    1          10    48067
     1       2      M   56          16    70072
     2       3      M   25          15    55117
     3       4      M   45           7    02460
     4       5      M   25          20    55455
```

```
[10]: data_users.shape
```

```
[10]: (6040, 5)
```

```
[11]: # merge the datasets
      Master_Data_tmp=pd.merge(data_users[['UserID', 'Age', 'Gender',␣
       ↪'Occupation']],data_ratings[['UserID', 'MovieID', 'Rating']], on=['UserID'])
```

```
[12]: Master_Data_tmp.head()
```

```
[12]:    UserID  Age Gender  Occupation  MovieID  Rating
      0       1    1      F          10     1193       5
      1       1    1      F          10      661       3
      2       1    1      F          10      914       3
      3       1    1      F          10     3408       4
      4       1    1      F          10     2355       5
```

```
[13]: Master_Data_tmp.shape
```

```
[13]: (1000209, 6)
```

```
[14]: # the final merged Master Dataset
      Master_Data=pd.merge(Master_Data_tmp,data_movie[['MovieID', 'Title']],␣
       ↪on=['MovieID'])
```

```
[15]: Master_Data.head()
```

```
[15]:    UserID  Age Gender  Occupation  MovieID  Rating  \
      0       1    1      F          10     1193       5
      1       2   56      M          16     1193       5
      2      12   25      M          12     1193       4
      3      15   25      M           7     1193       4
      4      17   50      M           1     1193       5

                                           Title
```

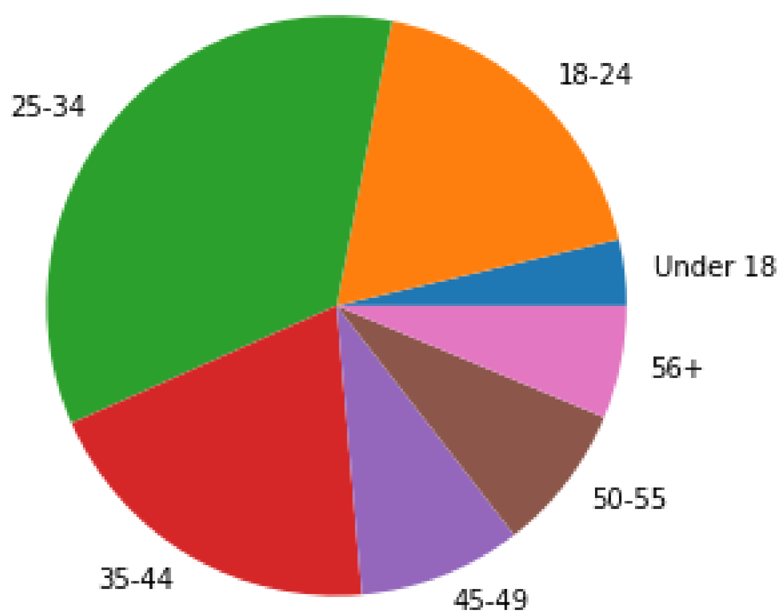
```
0   One Flew Over the Cuckoo's Nest (1975)
1   One Flew Over the Cuckoo's Nest (1975)
2   One Flew Over the Cuckoo's Nest (1975)
3   One Flew Over the Cuckoo's Nest (1975)
4   One Flew Over the Cuckoo's Nest (1975)
```

[16]:
```
Master_Data.shape
```

[16]:
```
(1000209, 7)
```

0.1 Histogram of User Age distribution

[17]:
```
fig = plt.figure(figsize = (10, 5))
counts,_=np.histogram(data_users.Age, bins=[1,18,25,35,45,50,56,100])
plt.pie(counts,labels=["Under␣
↪18","18-24","25-34","35-44","45-49","50-55","56+"])
plt.show()
```



0.2 User rating of the movie "Toy Story"

[18]:
```
ToyStory=Master_Data[Master_Data.Title=='Toy Story (1995)'][['Rating']]
```

[19]:
```
ToyStory
```

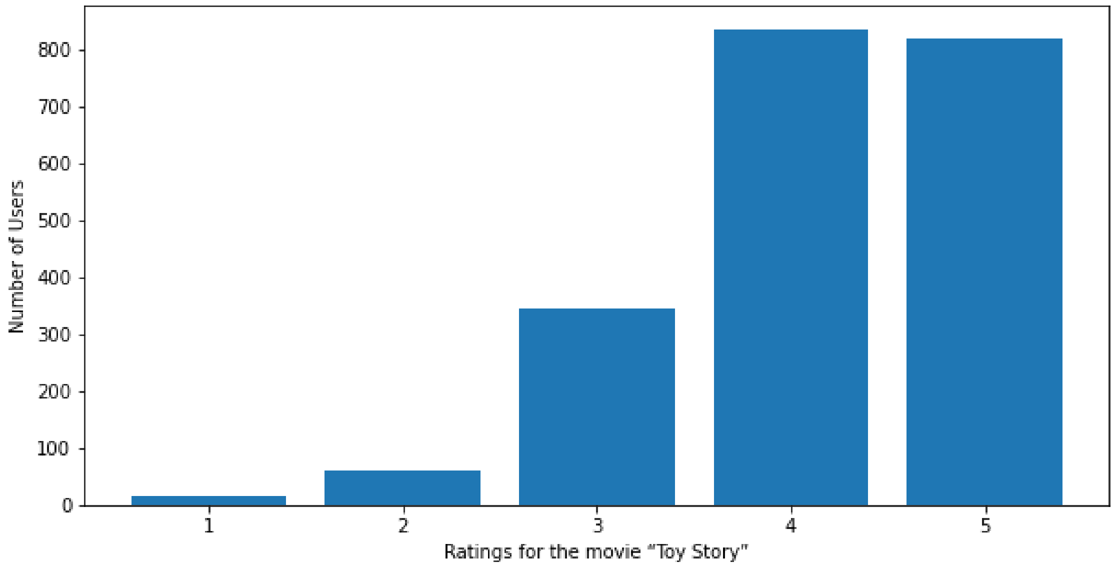
[19]:
```
       Rating
41626       5
41627       4
41628       4
41629       5
41630       5
…         …
43698       5
43699       5
43700       4
43701       4
43702       3

[2077 rows x 1 columns]
```

[20]:
```
rating_count=ToyStory['Rating'].value_counts()
```

[21]:
```
print(rating_count)
```

```
4    835
5    820
3    345
2     61
1     16
Name: Rating, dtype: int64
```

[22]:
```
# Histogram of User rating of the movie "Toy Story"
fig = plt.figure(figsize = (10, 5))
counts1,_=np.histogram(ToyStory, bins=[1,2,3,4,5,6])
plt.bar([1,2,3,4,5],counts1)
plt.xticks([1,2,3,4,5], [1,2,3,4,5])
plt.xlabel('Ratings for the movie "Toy Story"')
plt.ylabel('Number of Users')
plt.show()
```



0.3 Top 25 movies by viewership rating

[23]:
```
gr_count=Master_Data.groupby('Title')['Rating'].count().reset_index(name='No.␣
↪of Ratings')
gr_mean=Master_Data.groupby('Title')['Rating'].mean().reset_index(name='Avg.␣
↪Ratings')
```

[24]:
```
gr_count
```

[24]:
```
                                        Title  No. of Ratings
0                        $1,000,000 Duck (1971)              37
1                          'Night Mother (1986)              70
2                     'Til There Was You (1997)              52
3                            'burbs, The (1989)             303
4                    …And Justice for All (1979)            199
…                                          …             …
3701                      Zed & Two Noughts, A (1985)              29
3702                          Zero Effect (1998)             301
3703  Zero Kelvin (Kj rlighetens kj tere) (1995)               2
3704                       Zeus and Roxanne (1997)              23
3705                               eXistenZ (1999)             410

[3706 rows x 2 columns]
```

[25]:
```
gr_mean
```

[25]:
```
                                      Title  Avg. Ratings
0                     $1,000,000 Duck (1971)      3.027027
1                        'Night Mother (1986)      3.371429
2                   'Til There Was You (1997)      2.692308
3                          'burbs, The (1989)      2.910891
4                    …And Justice for All (1979)      3.713568
…                                       …           …
3701                 Zed & Two Noughts, A (1985)      3.413793
3702                         Zero Effect (1998)      3.750831
3703  Zero Kelvin (Kj rlighetens kj tere) (1995)      3.500000
3704                  Zeus and Roxanne (1997)      2.521739
3705                           eXistenZ (1999)      3.256098

[3706 rows x 2 columns]
```

[26]:
```python
merged_rating=pd.merge(gr_count,gr_mean,on=['Title'])
```

[27]:
```python
merged_rating
```

[27]:
```
                                      Title  No. of Ratings  Avg. Ratings
0                     $1,000,000 Duck (1971)              37      3.027027
1                        'Night Mother (1986)              70      3.371429
2                   'Til There Was You (1997)              52      2.692308
3                          'burbs, The (1989)             303      2.910891
4                    …And Justice for All (1979)             199      3.713568
…                                       …             …           …
3701                 Zed & Two Noughts, A (1985)              29      3.413793
3702                         Zero Effect (1998)             301      3.750831
3703  Zero Kelvin (Kj rlighetens kj tere) (1995)               2      3.500000
3704                  Zeus and Roxanne (1997)              23      2.521739
3705                           eXistenZ (1999)             410      3.256098

[3706 rows x 3 columns]
```

[28]:
```python
# we put a threshold of minimum 1000 ratings for a movie for sorting the␣
↪average ratings so that movies with few but high ratings don't falsely␣
↪influence the result
mr=merged_rating[merged_rating['No. of Ratings'] > 1000]
mr
```

[28]:
```
                                      Title  No. of Ratings  \
16                  2001: A Space Odyssey (1968)            1716
43                            Abyss, The (1989)            1715
68                     African Queen, The (1951)            1057
80                         Air Force One (1997)            1076
84                             Airplane! (1980)            1731
…                                         …               …
```

```
3635  Willy Wonka and the Chocolate Factory (1971)            1313
3655                                Witness (1985)            1046
3656                       Wizard of Oz, The (1939)            1718
3679                                 X-Men (2000)            1511
3693                      Young Frankenstein (1974)            1193

      Avg. Ratings
16        4.068765
43        3.683965
68        4.251656
80        3.588290
84        3.971115
…              …
3635      3.861386
3655      3.996176
3656      4.247963
3679      3.820649
3693      4.250629

[207 rows x 3 columns]
```

[29]:

```python
top_rating = mr.sort_values('Avg. Ratings', ascending = False).head(25)
top_rating
```

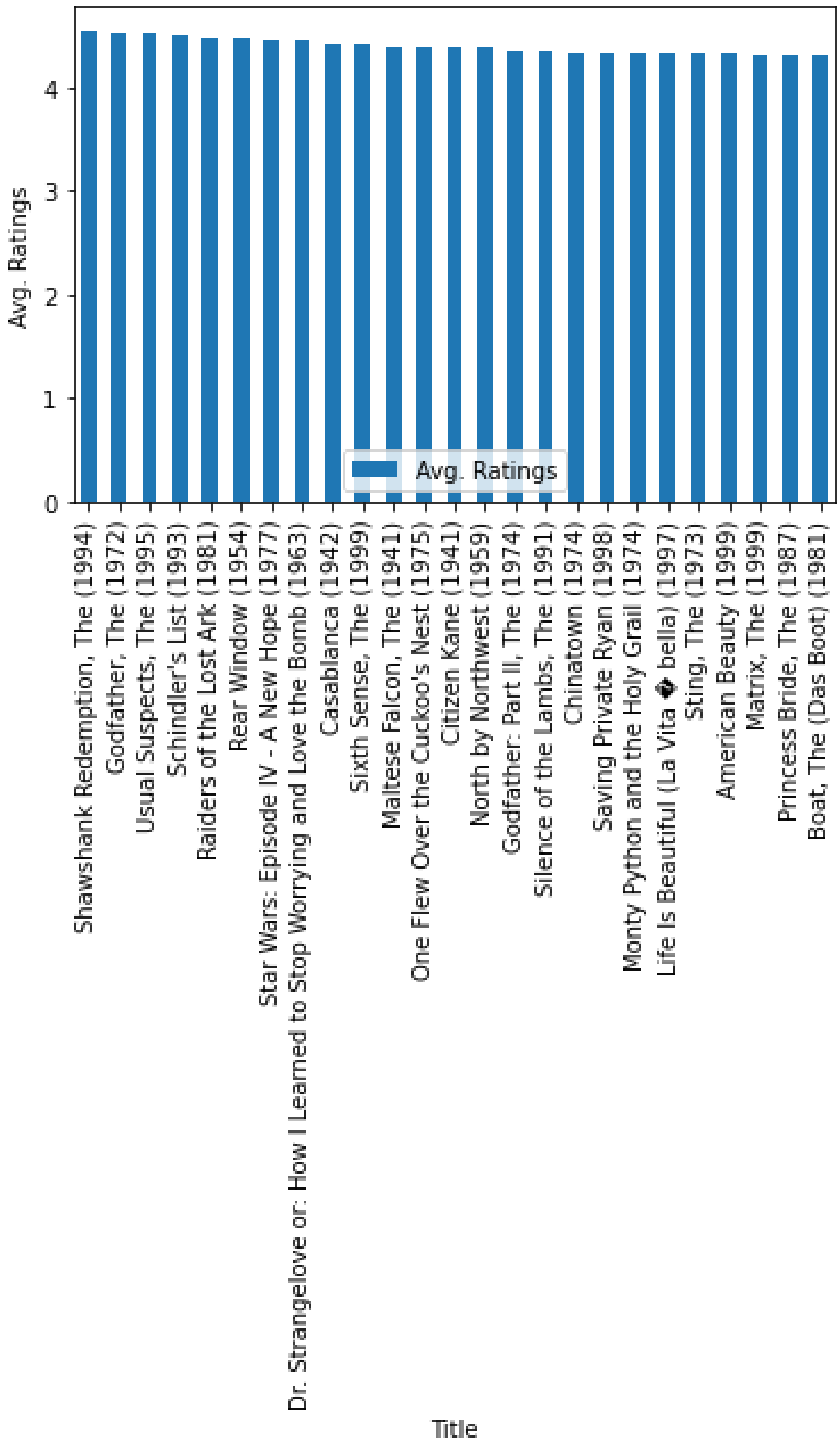
[29]:

```
                                                 Title  No. of Ratings  \
2970                  Shawshank Redemption, The (1994)            2227
1354                             Godfather, The (1972)            2223
3504                         Usual Suspects, The (1995)            1783
2901                            Schindler's List (1993)            2304
2711                    Raiders of the Lost Ark (1981)            2514
2738                                Rear Window (1954)            1050
3153          Star Wars: Episode IV - A New Hope (1977)            2991
975   Dr. Strangelove or: How I Learned to Stop Worr…            1367
609                                  Casablanca (1942)            1669
3015                            Sixth Sense, The (1999)            2459
2055                         Maltese Falcon, The (1941)            1043
2452            One Flew Over the Cuckoo's Nest (1975)            1725
684                                Citizen Kane (1941)            1116
2401                         North by Northwest (1959)            1315
1355                     Godfather: Part II, The (1974)            1692
2990                  Silence of the Lambs, The (1991)            2578
665                                   Chinatown (1974)            1185
2894                         Saving Private Ryan (1998)            2653
2217            Monty Python and the Holy Grail (1974)            1599
1923        Life Is Beautiful (La Vita   bella) (1997)            1152
3178                                  Sting, The (1973)            1049
127                              American Beauty (1999)            3428
```

```
2112                                  Matrix, The (1999)            2590
2654                          Princess Bride, The (1987)            2318
461                          Boat, The (Das Boot) (1981)            1001

      Avg. Ratings
2970      4.554558
1354      4.524966
3504      4.517106
2901      4.510417
2711      4.477725
2738      4.476190
3153      4.453694
975       4.449890
609       4.412822
3015      4.406263
2055      4.395973
2452      4.390725
684       4.388889
2401      4.384030
1355      4.357565
2990      4.351823
665       4.339241
2894      4.337354
2217      4.335210
1923      4.329861
3178      4.320305
127       4.317386
2112      4.315830
2654      4.303710
461       4.302697
```

[30]:

```
ax = top_rating.plot.bar(x='Title', y='Avg. Ratings',rot=90, ylabel='Avg.␣
↪Ratings')
```



Avg. Ratings
0
1
2
3
4
Avg. Ratings
Shawshank Redemption, The (1994)
Godfather, The (1972)
Usual Suspects, The (1995)
Schindler's List (1993)
Raiders of the Lost Ark (1981)
Rear Window (1954)
Star Wars: Episode IV - A New Hope (1977)
Dr. Strangelove or: How I Learned to Stop Worrying and Love the Bomb (1963)
Casablanca (1942)
Sixth Sense, The (1999)
Maltese Falcon, The (1941)
One Flew Over the Cuckoo's Nest (1975)
Citizen Kane (1941)
North by Northwest (1959)
Godfather: Part II, The (1974)
Silence of the Lambs, The (1991)
Chinatown (1974)
Saving Private Ryan (1998)
Monty Python and the Holy Grail (1974)
Life Is Beautiful (La Vita � bella) (1997)
Sting, The (1973)
American Beauty (1999)
Matrix, The (1999)
Princess Bride, The (1987)
Boat, The (Das Boot) (1981)
Title

0.4 The ratings for all the movies reviewed by a particular user of user id = 2696

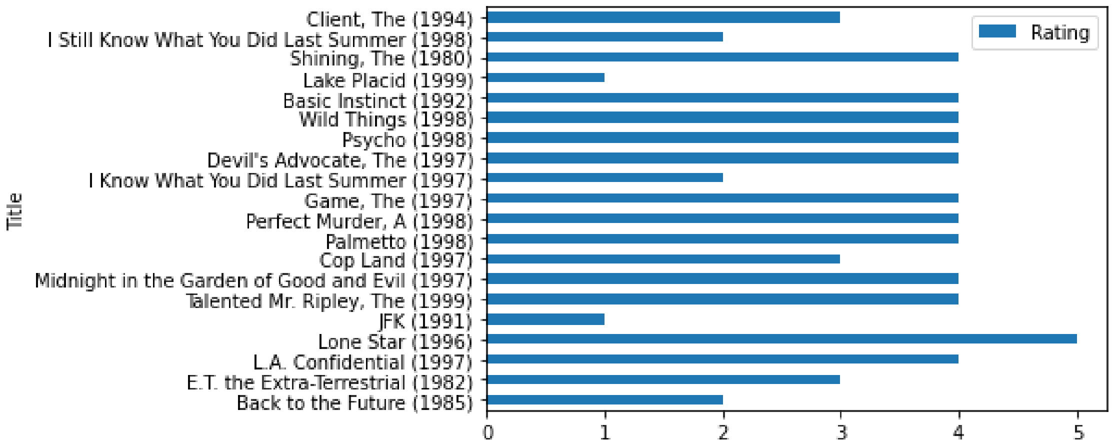
[31]:
```
user2696=Master_Data[Master_Data['UserID']==2696]
user2696=user2696[['Title','Rating']]
```

[32]:
```
user2696
```

[32]:

	Title	Rating
24345	Back to the Future (1985)	2
29848	E.T. the Extra-Terrestrial (1982)	3
244232	L.A. Confidential (1997)	4
250014	Lone Star (1996)	5
273633	JFK (1991)	1
277808	Talented Mr. Ripley, The (1999)	4
371178	Midnight in the Garden of Good and Evil (1997)	4
377250	Cop Land (1997)	3
598042	Palmetto (1998)	4
603189	Perfect Murder, A (1998)	4
609204	Game, The (1997)	4
611956	I Know What You Did Last Summer (1997)	2
612552	Devil's Advocate, The (1997)	4
613486	Psycho (1998)	4
616546	Wild Things (1998)	4
618708	Basic Instinct (1992)	4
621101	Lake Placid (1999)	1
689379	Shining, The (1980)	4
697451	I Still Know What You Did Last Summer (1998)	2
777089	Client, The (1994)	3

[33]:
```
ax1 = user2696.plot.barh(x='Title', y='Rating',rot=0, ylabel='Ratings of User␣
↪2696')
```



0.5 Feature Engineering:

[34]:
```
# splitting the different genres in the column Genre
genre_split=data_movie['Genre'].str.split('|')
genre_split
```

[34]:
```
0         [Animation, Children's, Comedy]
1        [Adventure, Children's, Fantasy]
2                       [Comedy, Romance]
3                         [Comedy, Drama]
4                                [Comedy]
                      …
3878                             [Comedy]
3879                              [Drama]
3880                              [Drama]
3881                              [Drama]
3882                    [Drama, Thriller]
Name: Genre, Length: 3883, dtype: object
```

[35]:
```
#separate column for each genre category with a one-hot encoding ( 1 and 0)␣
↪whether or not the movie belongs to that genre.
genre_sep=data_movie['Genre'].str.get_dummies()
genre_sep
```

[35]:
```
      Action  Adventure  Animation  Children's  Comedy  Crime  Documentary  \
0          0          0          1           1       1      0            0
1          0          1          0           1       0      0            0
2          0          0          0           0       1      0            0
3          0          0          0           0       1      0            0
4          0          0          0           0       1      0            0
```

```
…       …         …         …          …      …    …         …
3878       0          0          0            0       1       0             0
3879       0          0          0            0       0       0             0
3880       0          0          0            0       0       0             0
3881       0          0          0            0       0       0             0
3882       0          0          0            0       0       0             0

      Drama  Fantasy  Film-Noir  Horror  Musical  Mystery  Romance  Sci-Fi  \
0         0        0          0       0        0        0        0       0
1         0        1          0       0        0        0        0       0
2         0        0          0       0        0        0        1       0
3         1        0          0       0        0        0        0       0
4         0        0          0       0        0        0        0       0
…     …      …        …       …      …       …       …       …
3878      0        0          0       0        0        0        0       0
3879      1        0          0       0        0        0        0       0
3880      1        0          0       0        0        0        0       0
3881      1        0          0       0        0        0        0       0
3882      1        0          0       0        0        0        0       0

      Thriller  War  Western
0            0    0        0
1            0    0        0
2            0    0        0
3            0    0        0
4            0    0        0
…          …   …       …
3878         0    0        0
3879         0    0        0
3880         0    0        0
3881         0    0        0
3882         1    0        0

[3883 rows x 18 columns]
```

[36]:
```python
# Finding out all the unique genres
print(genre_sep.columns)
print(len(genre_sep.columns))
```

```
Index(['Action', 'Adventure', 'Animation', 'Children's', 'Comedy', 'Crime',
       'Documentary', 'Drama', 'Fantasy', 'Film-Noir', 'Horror', 'Musical',
       'Mystery', 'Romance', 'Sci-Fi', 'Thriller', 'War', 'Western'],
      dtype='object')
18
```

[37]:
```python
# features affecting the ratings of any particular movie
# gender, age, occupation, genre
```

```python
# we create a dataframe consisting of only those relevant features

Ratings_Data=pd.concat([Master_Data[['Gender', 'Age',␣
 ↪'Occupation','Rating']],genre_sep], axis=1)
# replace Male(M) by 1 and Female(F) by 0 in the Gender column
Ratings_Data['Gender'].replace({'M': 1, 'F': 0 }, inplace=True)
```

[38]:
```python
Ratings_Data.head(10)
```

[38]:
```
   Gender  Age  Occupation  Rating  Action  Adventure  Animation  Children's  \
0       0    1          10       5     0.0        0.0        1.0         1.0
1       1   56          16       5     0.0        1.0        0.0         1.0
2       1   25          12       4     0.0        0.0        0.0         0.0
3       1   25           7       4     0.0        0.0        0.0         0.0
4       1   50           1       5     0.0        0.0        0.0         0.0
5       0   18           3       4     1.0        0.0        0.0         0.0
6       1    1          10       5     0.0        0.0        0.0         0.0
7       0   25           7       5     0.0        1.0        0.0         1.0
8       0   25           1       3     1.0        0.0        0.0         0.0
9       1   45           3       5     1.0        1.0        0.0         0.0

   Comedy  Crime  …  Fantasy  Film-Noir  Horror  Musical  Mystery  Romance  \
0     1.0    0.0  …      0.0        0.0     0.0      0.0      0.0      0.0
1     0.0    0.0  …      1.0        0.0     0.0      0.0      0.0      0.0
2     1.0    0.0  …      0.0        0.0     0.0      0.0      0.0      1.0
3     1.0    0.0  …      0.0        0.0     0.0      0.0      0.0      0.0
4     1.0    0.0  …      0.0        0.0     0.0      0.0      0.0      0.0
5     0.0    1.0  …      0.0        0.0     0.0      0.0      0.0      0.0
6     1.0    0.0  …      0.0        0.0     0.0      0.0      0.0      1.0
7     0.0    0.0  …      0.0        0.0     0.0      0.0      0.0      0.0
8     0.0    0.0  …      0.0        0.0     0.0      0.0      0.0      0.0
9     0.0    0.0  …      0.0        0.0     0.0      0.0      0.0      0.0

   Sci-Fi  Thriller  War  Western
0     0.0       0.0  0.0      0.0
1     0.0       0.0  0.0      0.0
2     0.0       0.0  0.0      0.0
3     0.0       0.0  0.0      0.0
4     0.0       0.0  0.0      0.0
5     0.0       1.0  0.0      0.0
6     0.0       0.0  0.0      0.0
7     0.0       0.0  0.0      0.0
8     0.0       0.0  0.0      0.0
9     0.0       1.0  0.0      0.0

[10 rows x 22 columns]
```

[39]:
```
# Find the correlation of the other features with Rating to see which is␣
↪maximum correlated
Ratings_Data1=Ratings_Data[['Age','Occupation','Rating','Gender']]
Ratings_Data1[Ratings_Data1.columns].corr()['Rating']
```

[39]:
```
Age           0.056869
Occupation    0.006753
Rating        1.000000
Gender       -0.019861
Name: Rating, dtype: float64
```

0.6 An appropriate model to predict the movie ratings: Linear Regression

[40]:
```
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split
from sklearn import metrics
```

[41]:
```
X_feature=Ratings_Data1.drop(['Rating'],axis=1)
Y_target=Ratings_Data1['Rating']
```

[42]:
```
print(X_feature.shape)
X_feature.head
```

```
(1000209, 3)
```

[42]:
```
<bound method NDFrame.head of          Age  Occupation  Gender
0          1          10       0
1         56          16       1
2         25          12       1
3         25           7       1
4         50           1       1
…         …          …        …
1000204   18          17       1
1000205   35          14       1
1000206   18          17       1
1000207   18          20       0
1000208   25           1       1

[1000209 rows x 3 columns]>
```

[43]:
```
print(Y_target.shape)
Y_target.head
```

```
(1000209,)
```

[43]:
```
<bound method NDFrame.head of 0          5
1          5
2          4
3          4
4          5
          ..
1000204    5
1000205    3
1000206    1
1000207    5
1000208    4
Name: Rating, Length: 1000209, dtype: int64>
```

[44]:
```
x_train, x_test, y_train, y_test = train_test_split(X_feature, Y_target,␣
 ↪random_state=1)
```

[45]:
```
print(x_train.shape)
print(x_test.shape)
print(y_train.shape)
print(y_test.shape)
```

```
(750156, 3)
(250053, 3)
(750156,)
(250053,)
```

[46]:
```
lin_reg = LinearRegression()
```

[47]:
```
lin_reg.fit(x_train, y_train)
```

[47]:
```
LinearRegression()
```

[48]:
```
y_pred = lin_reg.predict(x_test)
```

[49]:
```
y_pred
```

[49]:
```
array([3.55002686, 3.65282438, 3.51390249, …, 3.59487431, 3.60326881,
       3.68434351])
```

[50]:
```
# print the values obtained from different classification metrics
print('y-intercept: ', lin_reg.intercept_)
print('Beta coefficients: ',lin_reg.coef_)
print('Mean Sq Error  MSE: ',metrics.mean_squared_error(y_test, y_pred))
print('Root Mean Sq Error RMSE:',np.sqrt(metrics.mean_squared_error(y_test,␣
 ↪y_pred)))
print('r2 value: ',metrics.r2_score(y_test, y_pred))
```

```
y-intercept:  3.4518977639059805
Beta coefficients:  [ 0.00540498  0.00083945 -0.05207392]
Mean Sq Error  MSE:  1.2431690877572024
Root Mean Sq Error RMSE: 1.1149749269634732
r2 value:  0.0034613427979662825
```

[51]:
```python
# print the first 20 actual and predicted responses
print('actual:    ', y_test.values[0:10])
print('predicted: ', y_pred[0:10])
```

```
actual:     [4 4 3 3 4 5 4 3 4 4]
predicted:  [3.55002686 3.65282438 3.51390249 3.70855308 3.58899816 3.50047129
 3.5357878  3.50047129 3.68434351 3.58899816]
```

[ ]:
```python

```