Project - 2

[12]:
```
import numpy as np
import pandas as pd
import re
import matplotlib.pyplot as plt
from matplotlib import style
%matplotlib inline
```

[13]:
```
#Importing all 3 Datasets
users_data = pd.read_csv("users.dat",sep="::", header=None,␣
↪names=['UserID','Gender','Age','Occupation','Zip-code'],
            dtype={'UserID': np.int32, 'Gender': np.str, 'Age': np.int32,␣
↪'Occupation' : np.int32, 'Zip-code' : np.str},
            engine='python')
movie_data = pd.read_csv("movies.dat",
                       sep="::", header=None,␣
↪names=['MovieID','Title','Genres'],
                       dtype={'MovieID': np.int32, 'Title': np.str, 'Genres':␣
↪np.str}, engine='python')
ratings_data = pd.read_csv("ratings.dat",
                       sep="::", header=None,␣
↪names=['UserID','MovieID','Rating','Timestamp'],
                dtype={'UserID': np.int32, 'MovieID': np.int32, 'Rating': np.
↪int32, 'Timestamp' : np.str}, engine='python')
```

[3]:
```
#Analysing the Datasets
#1) Users Data
users_data.head()
```

[3]:

	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

[4]:
```
users_data.isnull().sum()
```

[4]:
```
UserID        0
Gender        0
Age           0
Occupation 0 Zip-
code 0
dtype: int64
```

[5]:
```
users_data.shape
```

[5]: (6040, 5)

[8]:
```
#2) Movie Data
movie_data.head()
```

[8]:

	MovieID	Title	Genres
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

[9]:
```
movie_data.isnull().sum()
```

[9]:
```
MovieID    0
Title      0
Genres     0
dtype: int64
```

[10]:
```
movie_data.shape
```

[10]: (3883, 3)

[11]:
```
#3) Rating data
ratings_data.head()
```

[11]:

	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

[12]:
```
ratings_data.isnull().sum()
```

[12]:
```
UserID 0 MovieID
0
Rating       0
Timestamp    0
dtype: int64
```

[13]:
```
ratings_data.shape
```

[13]: (1000209, 4)

[14]:
```
#Merging the Dataset and creating a Master Dataset
#Merging Users dataset and ratings dataset
Master_Data = pd.merge(users_data,ratings_data,on = 'UserID')
Master_Data.head()
```

[14]: UserID Gender Age Occupation Zip-codeMovieID Rating Timestamp

```
0       1   F    1     10    48067 1193  5     978300760
1       1   F    1     10    48067 661   3     978302109
2       1   F    1     10    48067 914   3     978301968
3       1   F    1     10    48067 3408  4     978300275
4       1   F    1     10    48067 2355  5     978824291
```

[15]:
```
#Merging Master Dataset and movie dataset
Master_Data=pd.merge(Master_Data,movie_data,on = 'MovieID')
Master_Data.head()
```

[15]:
```
  UserID Gender Age Occupation Zip-codeMovieID Rating Timestamp \
0       1   F    1     10    48067 1193  5     978300760
1       2   M    56    16    70072 1193  5     978298413
2       12  M    25    12    32793 1193  4     978220179
3       15  M    25    7     22903 1193  4     978199279
4       17  M    50    1     95350 1193  5     978158471

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

[16]:
```
#Preparing the Master datset as required
Master_Data = Master_Data.drop(['Zip-code'],axis=1)
Master_Data = Master_Data.drop(['Timestamp'],axis=1)
```

[17]:
```
Master_Data =␣
↪Master_Data[['UserID','Gender','Age','Occupation','MovieID','Title','Genres
','Rating']] Master_Data.head()
```

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

                               Title Genres
Rating 0 One Flew Over the Cuckoo's Nest (1975)
Drama 5
1  One Flew Over the Cuckoo's Nest (1975)   Drama 5
2  One Flew Over the Cuckoo's Nest (1975)   Drama 4
```

```
3  One Flew Over the Cuckoo's Nest (1975)   Drama 4
4  One Flew Over the Cuckoo's Nest (1975)   Drama       5
```

[57]:
```
#Data Visualizations
#1) User Age Distribution
```

[18]:
```
Age_count = users_data['Age'].value_counts()
Age_count
```

[18]:
```
25    2096
35    1193
18    1103
45     550
50     496
56     380
1      222
Name: Age, dtype: int64
```

[19]:
```
Age_Category = ('Under 18','18-24','25-34','35-44','45-
49','50-55','56+') x_position = np.arange(len(Age_Category))
x_position
```

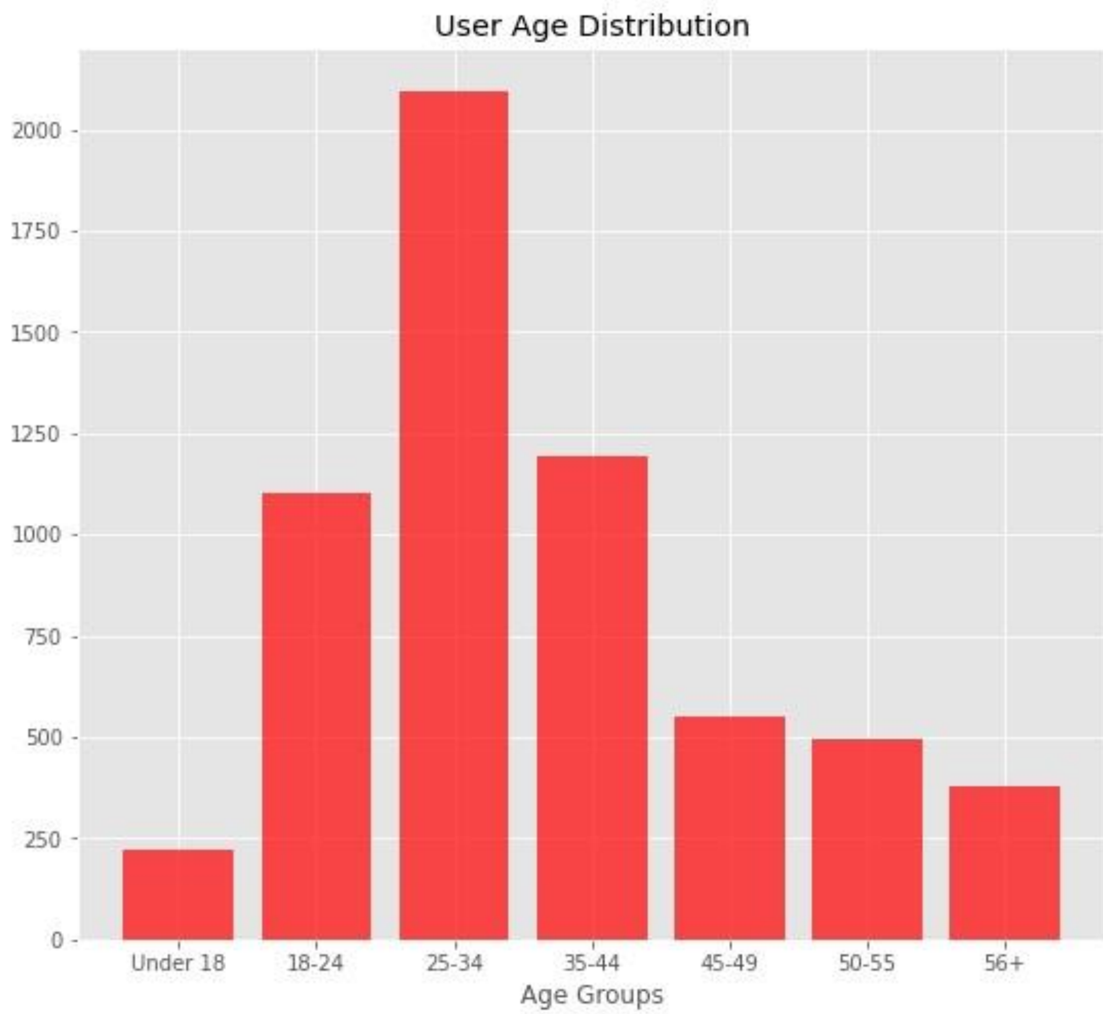
[19]:
```
array([0, 1, 2, 3, 4, 5, 6])
```

[20]:
```
Age_Values = 
 [Age_count[1],Age_count[18],Age_count[25],Age_count[35],Age_count[45],Age_count[50],Age_cou
Age_Values
```

[20]:
```
[222, 1103, 2096, 1193, 550, 496, 380]
```

[10]:
```
#plotting bar chart
style.use('ggplot')
plt.figure(figsize=(9,8))
plt.bar(x_position,Age_Values,align='center',color='r',alpha=0.7)
#set the y axis lable
plt.xlabel('Age Groups')
#set the bar value
plt.xticks(x_position,Age_Category)
```

```
#set the title
plt.title('User Age Distribution')
plt.show()
```




```
[142]: #The above age distribution shows that most of the users are 25-
34 years old
```

```
[1]: #2) User ratings of movie Toy Story
```

```
[21]: #Fetching thr Movie ID of Toy Story
movie_data.MovieID[movie_data.Title=='Toy Story (1995)']
```

```
[21]: 0    1
Name: MovieID, dtype: int32
```

[22]:
```
toystory_data = ratings_data[ratings_data.MovieID==1]
toystory_data.head(10)
```

[22]:

	UserID	MovieID	Rating	Timestamp
40	1	1	5	978824268
469	6	1	4	978237008
581	8	1	4	978233496
711	9	1	5	978225952
837	10	1	5	978226474
1966	18	1	4	978154768
2276	19	1	5	978555994
2530	21	1	3	978139347
2870	23	1	4	978463614
3405	26	1	3	978130703

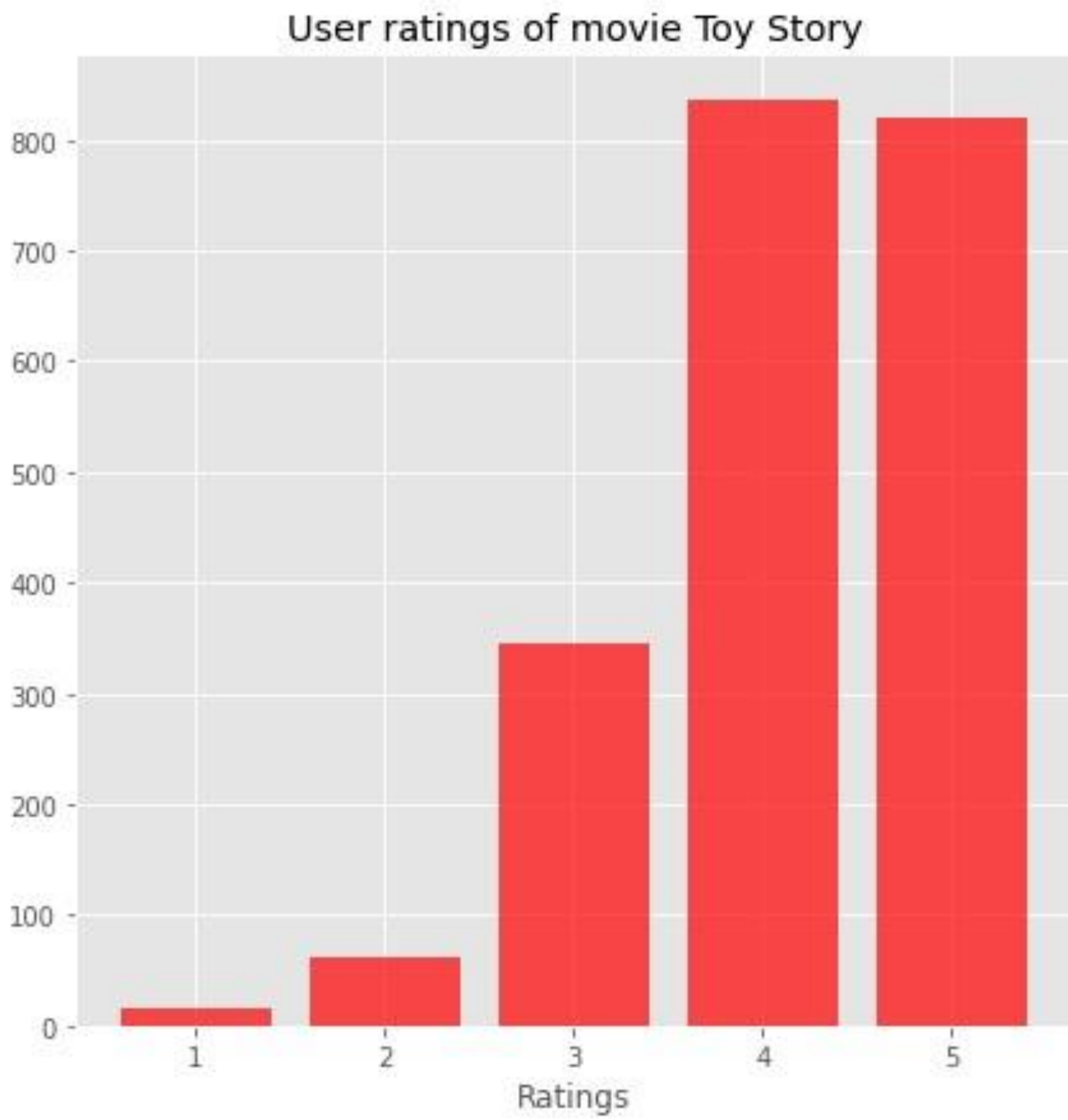
[23]:
```
movie_ratings_toystory = toystory_data.groupby('Rating').size()
movie_ratings_toystory
```

[23]:
```
Rating
1     16
2     61
3     345
4     835
5     820
dtype: int64
```

[24]:
```
ratings_type = ('1','2','3','4','5')
x_pos = np.arange(len(ratings_type))
x_pos
```

[24]:
```
array([0, 1, 2, 3, 4])
```

[15]:
```
#plotting bar chart
style.use('ggplot')
plt.figure(figsize=(7,7))
plt.bar(x_pos,movie_ratings_toystory,align='center',color='r',alpha=0.7)
#set the y axis lable
plt.xlabel('Ratings')
#set the bar value
plt.xticks(x_pos,ratings_type)
#set the title
plt.title('User ratings of movie Toy Story ')
plt.show()
```




```
[53]: #The above plot shows that the movie 'Toystory' has got 4 **
(stars) maximum
```

```
[20]: #3) Top 25 movies by viewership rating
```

```
[25]: #Fetching the Data and ratings of each movie by aggrega
movie_rating = Master_Data.groupby(['Title'],
as_index=False) average_movie_ratings =
movie_rating.agg({'Rating':'mean'})
average_movie_ratings.head(25)
```

[25]:

```
                                   Title   Rating
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
5                        1-900 (1994)2.500000
6      10 Things I Hate About You (1999)3.422857
7               101 Dalmatians (1961)3.596460
8               101 Dalmatians (1996)3.046703
9                 12 Angry Men (1957)4.295455
10          13th Warrior, The (1999)3.158667
11                         187 (1997)2.745455
12        2 Days in the Valley (1996)3.283217
13                    20 Dates (1998)2.856115
14  20,000 Leagues Under the Sea (1954)3.702609
15              200 Cigarettes (1999)2.883978
16        2001: A Space Odyssey (1968)4.068765
17                        2010 (1984)3.417021
18      24 7: Twenty Four Seven (1997)4.000000
19                24-hour Woman (1998)1.777778
20                     28 Days (2000)3.065347
21   3 Ninjas: High Noon On Mega Mountain1.361702
(1998)
22                   3 Strikes (2000)2.750000
23                    301, 302 (1995)2.888889
24                39 Steps, The (1935)4.075099
```

[26]:

```python
top_25_movies = average_movie_ratings.sort_values('Rating', ascending=False).
 ↪head(25)
top_25_movies
```

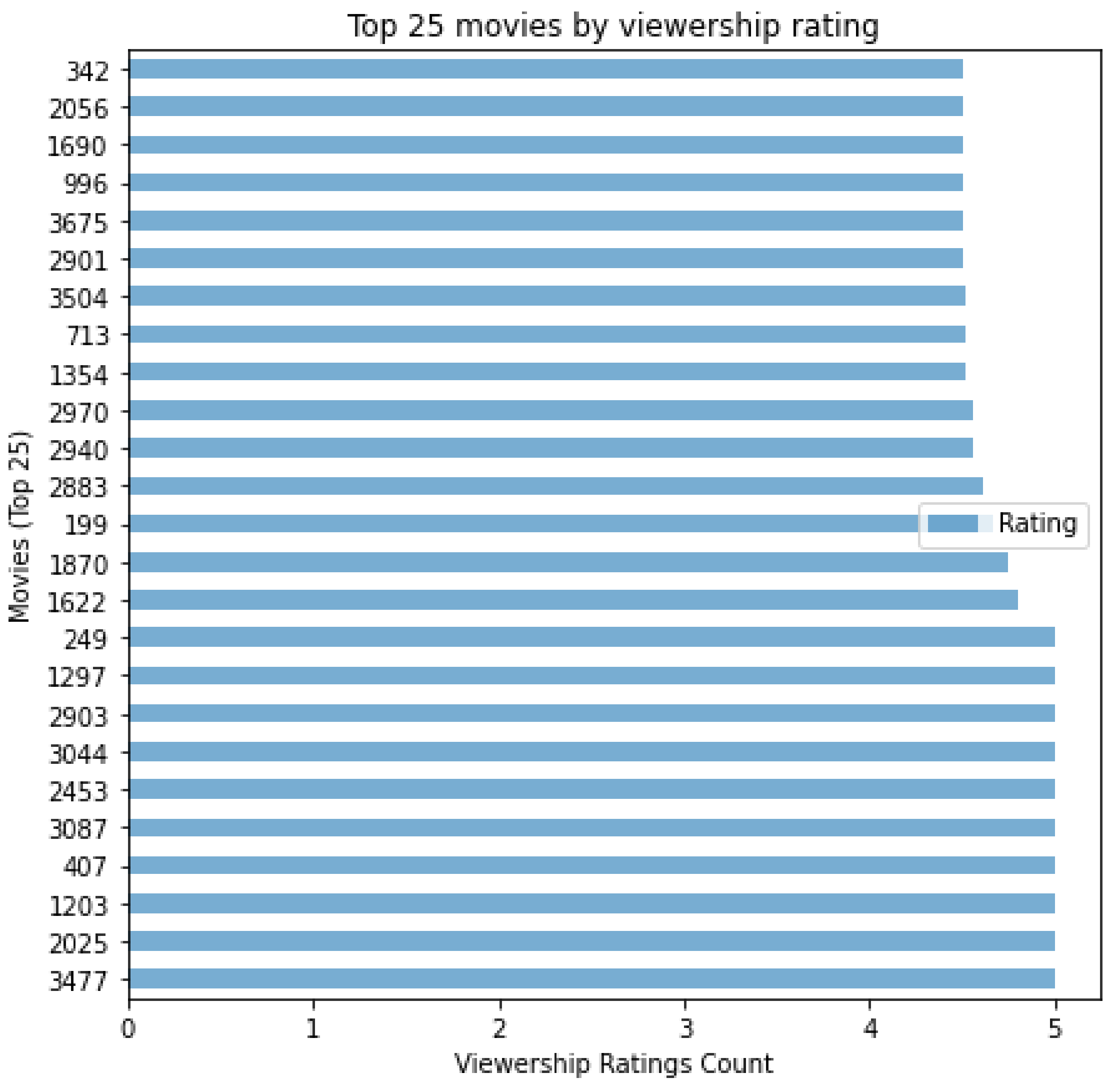
[26]:

```
                                   Title    Rating
3477              Ulysses (Ulisse) (1954)   5.000000
2025                         Lured (1947)   5.000000
1203              Follow the Bitch (1998)   5.000000
407             Bittersweet Motel (2000)   5.000000
3087              Song of Freedom (1936)   5.000000
2453             One Little Indian (1973)   5.000000
3044                Smashing Time (1967)   5.000000
2903      Schlafes Bruder (Brother of Sleep)   5.000000
          (1995)
1297       Gate of Heavenly Peace, The (1995)   5.000000
249                      Baby, The (1973)   5.000000
1622      I Am Cuba (Soy Cuba/Ya Kuba) (1964)   4.800000
1870                      Lamerica (1994)   4.750000
199                Apple, The (Sib) (1998)   4.666667
2883                       Sanjuro (1962)   4.608696
```

```
2940Seven Samurai (The Magnificent Seven) (Shichin…4.560510
2970              Shawshank Redemption, The (1994)4.554558
1354                        Godfather, The (1972)4.524966
713                         Close Shave, A (1995)4.520548
3504 Usual Suspects, The (1995) 4.517106 2901 Schindler's
List (1993) 4.510417
3675                   Wrong Trousers, The (1993)4.507937
996          Dry Cleaning (Nettoyage  sec) (1997)4.500000
1690    Inheritors, The (Die Siebtelbauern) (1998)4.500000
2056                            Mamma Roma (1962)4.500000
```

```
342                                    Bells, The (1926)   4.500000
```

[91]:
```python
top_25_movies.plot(kind='barh',alpha=0.6,figsize=(7,7))
plt.xlabel("Viewership Ratings Count")
plt.ylabel("Movies (Top 25) ")
plt.title("Top 25 movies by viewership rating ")
plt.show()
```



[27]:
```python
#Ratings for all the movies reviewed by for a particular user of user id = 2696 user_rating_data = Master_Data[Master_Data['UserID']==2496] user_rating_data = user_rating_data[['UserID','MovieID','Title','Rating']]
user_rating_data.head(10)
```

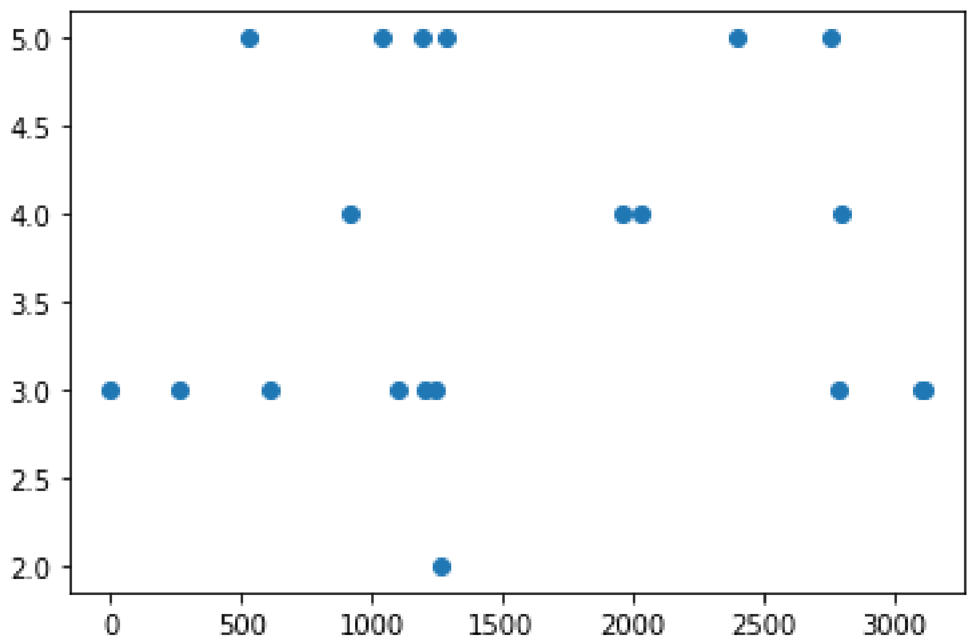
[27]:

	UserID	MovieID	Title	Rating
668	2496	1193	One Flew Over the Cuckoo's Nest (1975)	5
2518	2496	914	My Fair Lady (1964)	4

```
8506     2496    1287                    Ben-Hur (1959)        5
9492     2496    2804          Christmas Story, A (1983)        4
14173    2496    2398           Miracle on 34th Street        5
                                 (1947)
16319    2496    1035         Sound of Music, The (1965)        5
17581    2496    2791                   Airplane! (1980)        3
19798    2496    3105                  Awakenings (1990)        3
24263    2496    1270          Back to the Future (1985)        2
26818    2496     527              Schindler's List (1993)       5
```

[96]:

```python
# plotting the above data
plt.scatter(x=user_rating_data['MovieID'].head(20),␣
↪y=user_rating_data['Rating'].head(20))
plt.show()
```



[19]:

```python
#Feature Engineering
# 1) Find out all the unique genres
```

[28]:

```python
genres = Master_Data['Genres'].str.split("|")
genres
```

[28]:

```
0          [Drama]
1          [Drama]
2          [Drama]
3          [Drama]
```

```
4                          [Drama]
                        …
1000204            [Documentary]
1000205                  [Drama]
1000206                  [Drama]
1000207                  [Comedy, Drama, Western]
1000208                  [Documentary]
Name: Genres, Length: 1000209, dtype: object
```

[29]:
```
unique_genres = set()
for gen in genres:
    unique_genres = unique_genres.union(set(gen))
```

[33]:
```
unique_genres
```

[33]:
```
{'Action',
 'Adventure',
 'Animation',
 "Children's",
 'Comedy',
 'Crime',
 'Documentary',
 'Drama',
 'Fantasy',
 'Film-Noir',
 'Horror',
 'Musical',
 'Mystery',
 'Romance',
 'Sci-Fi',
 'Thriller',
 'War',
 'Western'}
```

[35]:
```
# 2) Create a separate column for each genre category with a one-hot encoding (
↪1 and 0)
```

[30]:
```
oneHotGenre = Master_Data["Genres"].str.get_dummies("|")
oneHotGenre.head()
```

[30]:
```
   Action Adventure Animation Children's Comedy  Crime Documentary  \
0       0    0       0       0       0       0       0
1       0    0       0       0       0       0       0
2       0    0       0       0       0       0       0
3       0    0       0       0       0       0       0
4       0    0       0       0       0       0       0

   Drama Fantasy Film-Noir Horror Musical Mystery Romance Sci-Fi  \
```

```
0      1    0      0      0      0      0      0      0
1      1    0      0      0      0      0      0      0
2      1    0      0      0      0      0      0      0
3      1    0      0      0      0      0      0      0
4      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
```

[31]:
```
oneHotGenre = pd.concat([Master_Data,oneHotGenre],axis=1)
oneHotGenre.head()
```

[31]:
```
  UserID Gender Age Occupation MovieID \
0      1   F     1      10    1193
1      2   M     56     16    1193
2      12  M     25     12    1193 3      15    M     25    7     1193
4     17     M   50          1    1193

                                Title Genres Rating Action Adventure \
0  One Flew Over the Cuckoo's Nest (1975)   Drama 5     0     0
1  One Flew Over the Cuckoo's Nest (1975)   Drama 5     0     0
2  One Flew Over the Cuckoo's Nest (1975)   Drama 4     0     0
3  One Flew Over the Cuckoo's Nest (1975)   Drama 4     0     0
4  One Flew Over the Cuckoo's Nest (1975)   Drama 5     0     0

  … Fantasy Film-Noir Horror Musical Mystery Romance Sci-Fi  \
0 … 0     0      0      0      0      0      0
1 … 0     0      0      0      0      0      0
2 … 0     0      0      0      0      0      0
3 … 0     0      0      0      0      0      0
4 … 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
[5 rows x 26 columns]
```

[49]:
```
oneHotGenre.columns
```

[49]:
```
Index(['UserID', 'Gender', 'Age', 'Occupation', 'MovieID',
'Title', 'Genres', 'Rating', 'UserID', 'Gender', 'Age', 'Occupation',
                                                          'MovieID', 'Title',
        'Genres', 'Rating', 'UserID', 'Gender', 'Age', 'Occupation',
        'MovieID',
        'Title', 'Genres', 'Rating', 'UserID', 'Gender', 'Age',
        'Occupation',
        'MovieID', 'Title', 'Genres', 'Rating', 'UserID', 'Gender',
        'Age',
        'Occupation', 'MovieID', 'Title', 'Genres', 'Rating',
        'Action',
        'Adventure', 'Animation', 'Children's', 'Comedy', 'Crime',
        'Documentary', 'Drama', 'Fantasy', 'Film-Noir', 'Horror',
        'Musical',
        'Mystery', 'Romance', 'Sci-Fi', 'Thriller', 'War', 'Western'],
       dtype='object')
```

[17]:
```
# 3) Determine the features affecting the ratings of any particular movie
```

[32]:
```
Features_Data =Master_Data.copy()
Features_Data
```

[32]:
```
         UserID Gender  Age  Occupation  MovieID  \
0             1   F      1      10     1193
1             2   M     56      16     1193
2            12   M     25      12     1193
3            15   M     25      7      1193
4            17   M     50      1      1193
…             …   …   …          …         …
1000204   5949   M     18      17     2198
1000205   5675   M     35      14     2703
1000206   5780   M     18      17     2845
1000207   5851   F     18      20     3607
1000208   5938   M     25      1      2909

                                          Title              Genres  \
0          One Flew Over the Cuckoo's Nest (1975)     Drama
1          One Flew Over the Cuckoo's Nest (1975)     Drama
2          One Flew Over the Cuckoo's Nest (1975)     Drama
3          One Flew Over the Cuckoo's Nest (1975)     Drama
4          One Flew Over the Cuckoo's Nest (1975)     Drama
…                                               …                 …
1000204  Modulations (1998)  Documentary 1000205  Broken Vessels (1998)
Drama 1000206   White Boys (1999)      Drama
1000207                   One Little Indian (1973)Comedy|Drama|Western
```

```
1000208                    Five Wives, Three Secretaries and Me (1998)
                                Documentary

         Rating
0             5
1             5
2             4
3             4
4             5
…          …
1000204       5
1000205       3
1000206       1
1000207       5
1000208       4

[1000209 rows x 8 columns]
```

[35]:
```
#Fetching the year ehich the movie was released
Features_Data[["Title","Year"]] =
 Features_Data.Title.str.extract("(.)\s\((. →\d+)",expand=True)
Features_Data = Features_Data.drop(['Title'],axis=1)
Features_Data
```

[35]:
```
         UserID Gender Age Occupation MovieID              Genres  \
   0     1     F     1     10    1193  Drama 1     2     M     56
                           16    1193  Drama
2           12   M     25    12    1193  Drama
3           15   M     25    7     1193  Drama
4           17   M     50    1     1193  Drama
…         …    …  …        …       …                  …
  1000204    5949  M     18    17    2198  Documentary 1000205
                   5675  M     35    14    2703  Drama
1000206   5780   M     18    17    2845  Drama
1000207   5851   F     18    20    3607  Comedy|Drama|Western
1000208   5938   M     25    1     2909  Documentary

        Rating  Year
0            5   1975
1            5   1975
2            4   1975
3            4   1975
4            5   1975
…         …   …
1000204      5   1998
```

```
1000205      3   1998
1000206     1  1999
1000207     5  1973
1000208     4  1998
[1000209 rows x 8 columns]
```

[36]:
```python
#Calculating the age of movies
Features_Data['Year'] = Features_Data.Year.astype(int)
Features_Data['Movie_Age'] = 2000 -Features_Data['Year']
Features_Data
```

[36]:
```
        UserID Gender Age Occupation MovieID              Genres  \
  0     1     F     1     10    1193  Drama 1     2     M     56
                               16    1193  Drama
2          12   M     25    12    1193  Drama
3          15   M     25    7     1193  Drama
4          17   M     50    1     1193  Drama
…        …    …  …        …      …                …
  1000204    5949  M     18    17    2198  Documentary 1000205
                  5675  M     35    14    2703  Drama
1000206   5780   M     18    17    2845  Drama
1000207   5851   F     18    20    3607  Comedy|Drama|Western
1000208   5938   M     25    1     2909  Documentary

        Rating Year Movie_Age
0           5   1975  25
1           5   1975  25
2           4   1975  25
3           4   1975  25
4           5   1975  25
…        …   …       …
1000204     5   1998  2
1000205     3   1998  2
1000206     1   1999  1 1000207  5     1973  27
1000208     4  1998         2

[1000209 rows x 9 columns]
```

[37]:
```python
#Creating Gender variable as integer type
Features_Data['Gender'] = Features_Data.Gender.replace('F',1)
Features_Data['Gender'] = Features_Data.Gender.replace('M',0)
Features_Data['Gender'] = Features_Data.Gender.astype(int)
Features_Data.head()
```

[37]:
```
  UserID Gender Age  Occupation MovieID       Rating Year  Movie_Age
                                  Genres
0      1      1    1         10     1193  Drama      5  1975         25
```

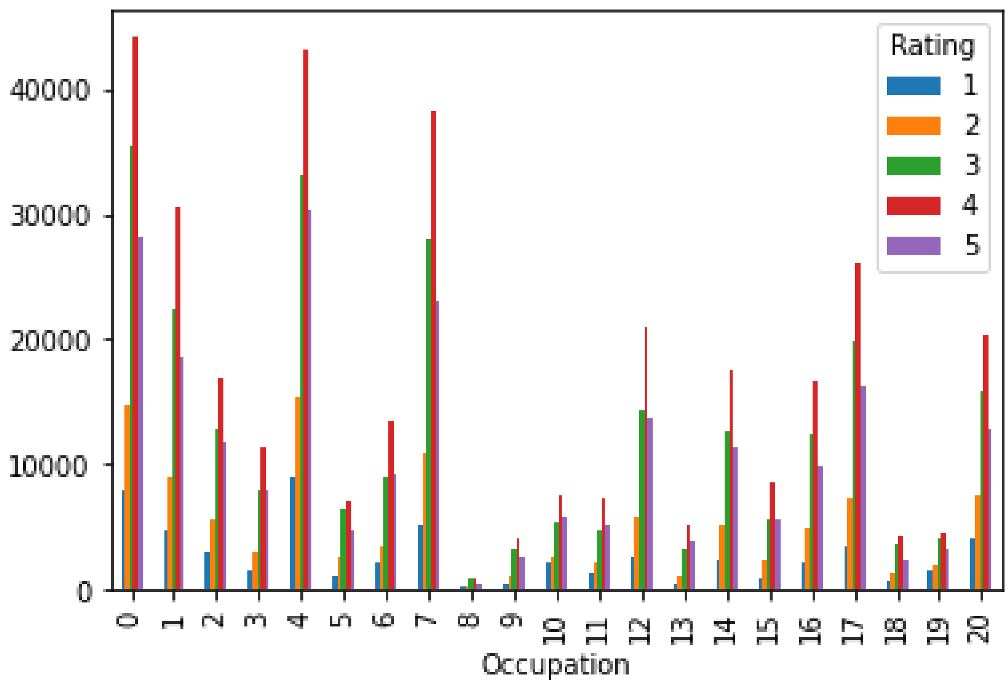
```
1    2   0   56   16   1193  Drama 5   1975  25
2    12  0   25   12   1193  Drama 4   1975  25
3    15  0   25   7    1193  Drama 4   1975  25
4    17  0   50   1    1193  Drama 5   1975  25
```

[38]:
```
#Checking the correlation of features with Rating
Features_Data[['Gender','Occupation', 'Age', 'Movie_Age']].
↪corrwith(Features_Data['Rating'])
```

[38]:
```
Gender       0.019861
Occupation   0.006753
Age          0.056869
Movie_Age    0.156946
dtype: float64
```

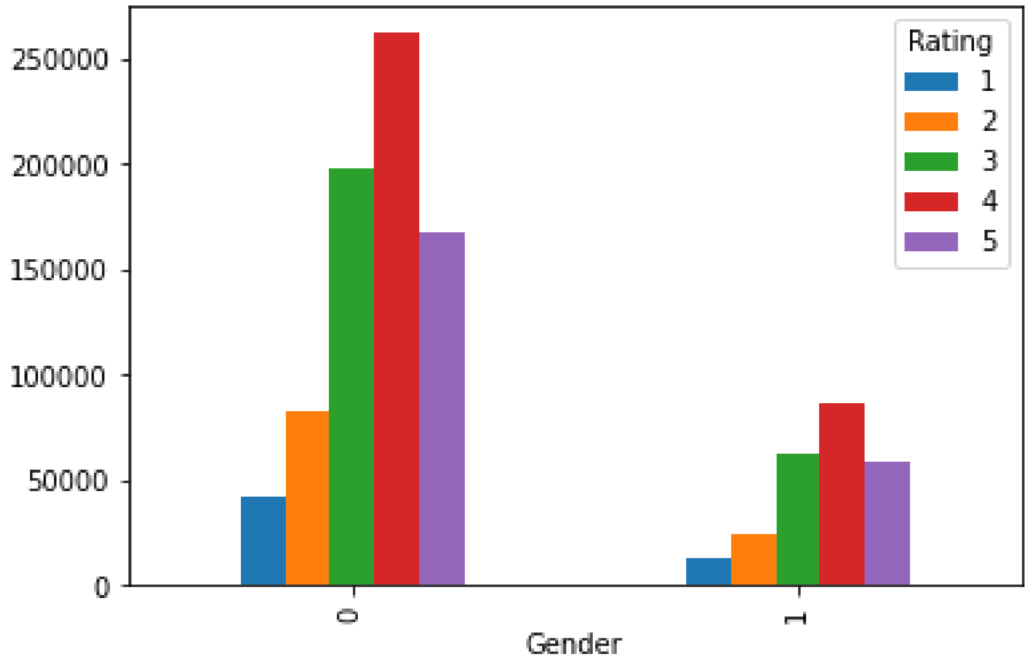
[103]:
```
#Movie_Age has the most positive relationship with Rating
```

[105]:
```
#Occupation relationship with Rating
Features_Data.groupby(["Occupation","Rating"]).size().unstack().
↪plot(kind='bar',stacked=False,legend=True)
plt.show()
```

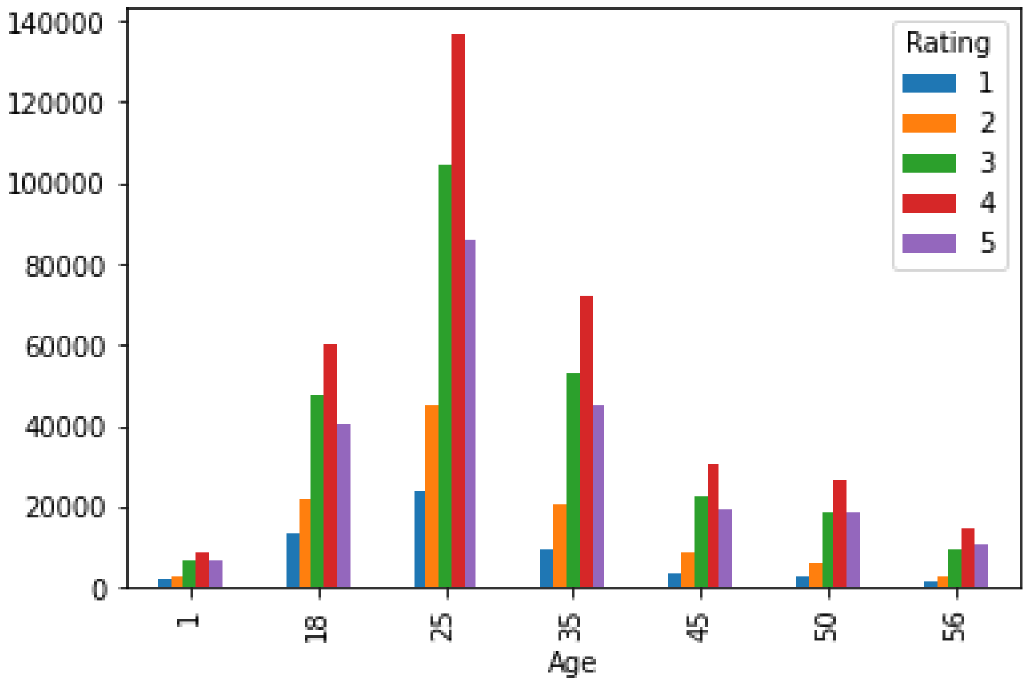


[107]:
```
#Gender relationship with Rating
#1 -> Male, 0 -> Female
```

```
Features_Data.groupby(["Gender","Rating"]).size().unstack().
↪plot(kind='bar',stacked=False,legend=True)
plt.show()
```

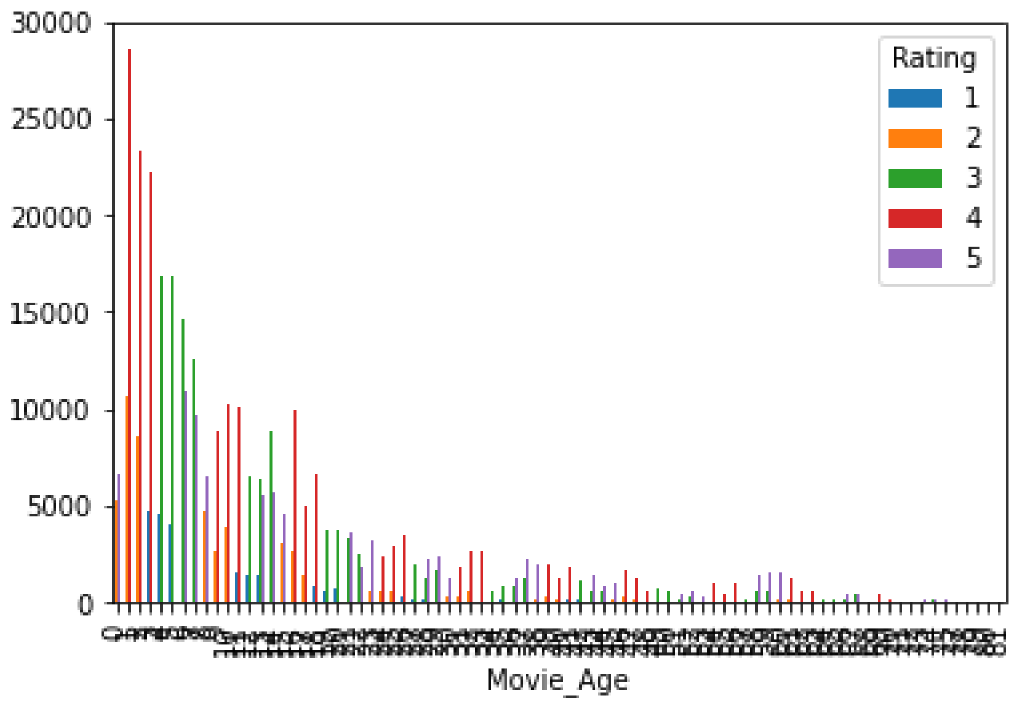



```
[108]: #Age relationship with Rating
Features_Data.groupby(["Age","Rating"]).size().unstack(
                                                       ).
↪plot(kind='bar',stacked=False,legend=True)
plt.show()
```



140000
120000
100000
80000
60000
40000
20000
0
Rating
1
2
3
4
5
1
18
25
35
45
50
56
Age

[109]:
```
#Movie_Age relationship with Rating
Features_Data.groupby(["Movie_Age","Rating"]).size().unstack().
↪plot(kind='bar',stacked=False,legend=True)
plt.show()
```



[39]:
```
#To Predict the values of rating we are using Logistic regression
```

[75]:
```
# Assign independent variables to X dataset
X = Master_Data[['Age','Occupation','MovieID']].head(500)
X
```

[75]:

	Age	Occupation	MovieID
0	1	10	1193
1	56	16	1193
2	25	12	1193
3	25	7	1193
4	50	1	1193
..	…	…	…
495	25	2	1193
496	18	4	1193
497	25	12	1193

```
498   18          4     1193

499   45         14     1193

[500 rows x 3 columns]
```

[76]:

```python
# Assign dependent variables to Y dataset
Y = Master_Data['Rating'].head(500)
Y
```

[76]:

```
0      5
1      5
2      4
3      4
4      5
..
495    4
496    5
497    5
498    5
499    5
Name: Rating, Length: 500, dtype: int32
```

[77]:

```python
# view the shape for both axes
print (X.shape)
print (Y.shape)
```

```
(500, 3)
(500,)
```

[78]:

```python
# Splitting the data into training & testing
datasets(70:30) import sklearn from
sklearn.model_selection import train_test_split
X_train, X_test, Y_train, Y_test = sklearn.model_selection.
↪train_test_split(X,Y,random_state=2,test_size=0.3)
```

[79]:

```python
# use the Logistic regression estimator
from sklearn.linear_model import
LogisticRegression logReg =
LogisticRegression()
```

[81]:

```python
# fit data into the Logistic regression estimator
logReg.fit(X_train,Y_train)
```

```
/usr/local/lib/python3.7/site-
packages/sklearn/linear_model/_logistic.py:940:
ConvergenceWarning: lbfgs failed to converge
(status=1): STOP: TOTAL NO. of ITERATIONS REACHED
LIMIT.

Increase the number of iterations (max_iter) or scale the data as
    shown in: https://scikit-
    learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver
options:
    https://scikit-
learn.org/stable/modules/linear_model.html#logisticregression
  extra_warning_msg=_LOGISTIC_SOLVER_CONVERGENCE_MSG)
```

[81]:
```
LogisticRegression(C=1.0, class_weight=None, dual=False,
                   fit_intercept=True, intercept_scaling=1,
                   l1_ratio=None, max_iter=100, multi_class='auto',
                   n_jobs=None, penalty='l2', random_state=None,
                   solver='lbfgs', tol=0.0001, verbose=0,
                   warm_start=False)
```

[82]:
```
#Model Evaluation
# predict the outcoome using Logistic regression estimator
y_predict=logReg.predict(X_test)
```

[83]:
```
y_predict
```

[83]:
```
array([5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5,
5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5,
                                                    5, 5, 5,
       5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5,
       5,
       5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5,
       5,
       5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5,
       5,
       5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5,
       5,
       5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5],
       dtype=int32)
```

[84]:
```
 # Calculate the accuracy of the
model from sklearn.metrics import
accuracy_score
accuracy_score(y_predict,Y_test)
```

[84]: 0.5866666666666667

[85]:
```
#Check model performance on new dataset
# create Example object with new values for prediction
X_new = [[25,7,1193],[18,17,2198]]
```

[86]:
```
logReg.predict(X_new)
```

[86]: array([5, 5], dtype=int32)

[89]:
```
from sklearn import metrics print
(metrics.confusion_matrix(Y_test, y_predict))
print (metrics.classification_report(Y_test,
y_predict))
```

```
[[ 0  0  0  0  1]
 [ 0  0  0  0  2]
 [ 0  0  0  0  9]
 [ 0  0  0  0 50]
 [ 0  0  0  0 88]]
              precision   recall f1-score  support

           10.00 0.00  0.00  1 2   0.00  0.00
            0.00  2 3   0.00  0.00  0.00  9
           4      0.00  0.00  0.00  50
           5      0.59  1.00  0.74  88

    accuracy      0.59  150 macro avg    0.12  0.20
     0.15  150 weighted avg 0.34  0.59  0.43  150
```

```
                                   /usr/local/lib/python3.7/site-
                packages/sklearn/metrics/_classification.py:1272:
UndefinedMetricWarning: Precision and F-score are ill-defined and
being set to 0.0 in labels with no predicted samples. Use
`zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
```

[ ]: