Import the Library

In [3]:

```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
```

load the dataset

In [4]:

```
df_user=pd.read_csv('users.dat',sep="::",names=['UserID','Gender','Age','Occupation','Zip Code'],engine ='python')
```

In [5]:

```
df_user
```

Out[5]:

	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
...	...	...	...	...	...
6035	6036	F	25	15	32603
6036	6037	F	45	1	76006
6037	6038	F	56	1	14706
6038	6039	F	45	0	01060
6039	6040	M	25	6	11106

6040 rows × 5 columns

In [14]:

```
df_user['Gender'].value_counts().idxmax()
```

Out[14]:

```
'M'
```

In [5]:

```
df_user.shape
```

Out[5]:

```
(6040, 5)
```

In [6]:

```
df_user.isna().sum().any
```

Out[6]:

```
<bound method Series.any of UserID        0
Gender        0
Age           0
Occupation    0
Zip Code      0
dtype: int64>
```

In [7]:

```
df_movies=pd.read_csv('movies.dat',sep="::",names=['MovieID','Title','Genres'],engine ='python')
```

In [8]:

```
df_movies.head()
```

Out[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

In [9]:

```
df_movies.shape
```

Out[9]:

```
(3883, 3)
```

In [10]:

```
df_movies.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 3883 entries, 0 to 3882
Data columns (total 3 columns):
 #   Column   Non-Null Count  Dtype
---  ------   --------------  -----
 0   MovieID  3883 non-null   int64
 1   Title    3883 non-null   object
 2   Genres   3883 non-null   object
dtypes: int64(1), object(2)
memory usage: 91.1+ KB
```

In [11]:

```
df_movies.isna().sum()
```

Out[11]:

```
MovieID    0
Title      0
Genres     0
dtype: int64
```

In [12]:

```
df_ratings=pd.read_csv('ratings.dat',sep="::",names=['UserID','MovieID','Title','Timestamp'],engine='python')
```

In [13]:

```
df_ratings.shape
```

Out[13]:

```
(1000209, 4)
```

In [14]:

```
df_ratings.isna().sum()
```

Out[14]:

```
UserID       0
MovieID      0
Title        0
Timestamp    0
dtype: int64
```

In [15]:

```
df_ratings.head()
```

Out[15]:

	UserID	MovieID	Title	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

Now merge the dataset movies and ratings and then merge the resultant datset with user dataset

In [16]:
```
df_MovieRatings=df_movies.merge(df_ratings,on='MovieID',how='inner')
```

In [17]:
```
df_MovieRatings.head()
```

Out[17]:

	MovieID	Title_x	Genres	UserID	Title_y	Timestamp
0	1	Toy Story (1995)	Animation|Children's|Comedy	1	5	978824268
1	1	Toy Story (1995)	Animation|Children's|Comedy	6	4	978237008
2	1	Toy Story (1995)	Animation|Children's|Comedy	8	4	978233496
3	1	Toy Story (1995)	Animation|Children's|Comedy	9	5	978225952
4	1	Toy Story (1995)	Animation|Children's|Comedy	10	5	978226474

In [18]:
```
df_MovieRatings.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 1000209 entries, 0 to 1000208
Data columns (total 6 columns):
 #   Column     Non-Null Count    Dtype 
---  ------     --------------    ----- 
 0   MovieID    1000209 non-null  int64 
 1   Title_x    1000209 non-null  object
 2   Genres     1000209 non-null  object
 3   UserID     1000209 non-null  int64 
 4   Title_y    1000209 non-null  int64 
 5   Timestamp  1000209 non-null  int64 
dtypes: int64(4), object(2)
memory usage: 53.4+ MB
```

In [19]:
```
df_MovieRatings.shape
```

Out[19]:
```
(1000209, 6)
```

In [20]:
```
df_MovieRatings.isna().sum()
```

Out[20]:
```
MovieID      0
Title_x      0
Genres       0
UserID       0
Title_y      0
Timestamp    0
dtype: int64
```

In [21]:
```
df_master=df_MovieRatings.merge(df_user,on='UserID',how='inner')
```

In [22]:
```
df_master.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 1000209 entries, 0 to 1000208
Data columns (total 10 columns):
 #   Column      Non-Null Count    Dtype 
---  ------      --------------    ----- 
 0   MovieID     1000209 non-null  int64 
 1   Title_x     1000209 non-null  object
 2   Genres      1000209 non-null  object
 3   UserID      1000209 non-null  int64 
 4   Title_y     1000209 non-null  int64 
 5   Timestamp   1000209 non-null  int64 
 6   Gender      1000209 non-null  object
 7   Age         1000209 non-null  int64 
 8   Occupation  1000209 non-null  int64 
 9   Zip Code    1000209 non-null  object
dtypes: int64(6), object(4)
memory usage: 83.9+ MB
```

In [23]:
```
df_master.isna().sum().any()
```

Out[23]:
```
False
```

In [24]:
```
# to csv file
df_master.to_csv('Master Data.csv')
df_master.head()
```

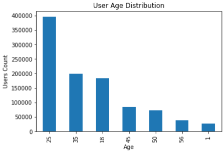
Out[24]:

	MovieID	Title_x	Genres	UserID	Title_y	Timestamp	Gender	Age	Occupation	Zip Code
0	1	Toy Story (1995)	Animation|Children's|Comedy	1	5	978824268	F	1	10	48067
1	48	Pocahontas (1995)	Animation|Children's|Musical|Romance	1	5	978824351	F	1	10	48067
2	150	Apollo 13 (1995)	Drama	1	5	978301777	F	1	10	48067
3	260	Star Wars: Episode IV - A New Hope (1977)	Action|Adventure|Fantasy|Sci-Fi	1	4	978300760	F	1	10	48067
4	527	Schindler's List (1993)	Drama|War	1	5	978824195	F	1	10	48067

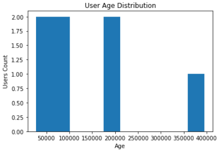
Explore the data using EDA

User Age Distribution

In [25]:
```
df_master['Age'].value_counts().plot(kind='bar')
plt.xlabel('Age')
plt.title('User Age Distribution')
plt.ylabel('Users Count')
plt.show()
```



In [26]:
```
df_master['Age'].value_counts().plot(kind='hist')
plt.xlabel('Age')
plt.title('User Age Distribution')
plt.ylabel('Users Count')
plt.show()
```



as we can see from the bar graph and histogram that users around the age of 25 are the one who are more involved in giving ratings than other users this also means that they are the one who are watching more movies as compared to other users. also we can note that as the age is increasing , less user are involved in ratings

User Ratings of the movie 'Toy Story'

In [27]:

```
#Extract Toy Story
toystory=df_master[df_master['Title_x'].str.contains('Toy Story')==True]
```

In [28]:

```
toystory
```

Out[28]:

	MovieID	Title_x	Genres	UserID	Title_y	Timestamp	Gender	Age	Occupation	Zip Code
0	1	Toy Story (1995)	Animation|Children's|Comedy	1	5	978824268	F	1	10	48067
50	3114	Toy Story 2 (1999)	Animation|Children's|Comedy	1	4	978302174	F	1	10	48067
53	1	Toy Story (1995)	Animation|Children's|Comedy	6	4	978237008	F	50	9	55117
124	1	Toy Story (1995)	Animation|Children's|Comedy	8	4	978233496	M	25	12	11413
263	1	Toy Story (1995)	Animation|Children's|Comedy	9	5	978225952	M	25	17	61614
...	...	...	...	...	...	...	...	...	...	...
998988	3114	Toy Story 2 (1999)	Animation|Children's|Comedy	3023	4	970471948	F	25	7	92108
999027	3114	Toy Story 2 (1999)	Animation|Children's|Comedy	5800	5	958015250	M	35	18	90804
999486	3114	Toy Story 2 (1999)	Animation|Children's|Comedy	2189	4	974607816	M	1	10	60148
999869	3114	Toy Story 2 (1999)	Animation|Children's|Comedy	159	4	989966944	F	45	0	37922
1000192	3114	Toy Story 2 (1999)	Animation|Children's|Comedy	5727	5	958492554	M	25	4	92843

3662 rows × 10 columns

In [29]:

```
toystory.groupby(['Title_x','Title_y']).size()
```

Out[29]:

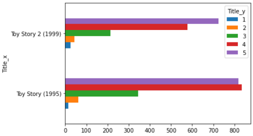
```
Title_x           Title_y
Toy Story (1995)     1           16
                     2           61
                     3          345
                     4          835
                     5          820
Toy Story 2 (1999) 1           25
                     2           44
                     3          214
                     4          578
                     5          724
dtype: int64
```

In [30]:

```
toystory.groupby(['Title_x','Title_y']).size().unstack().plot(kind='barh',legend=True)
```

Out[30]:

```
<AxesSubplot:ylabel='Title_x'>
```

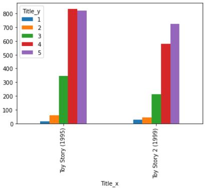


In [31]:

```
toystory.groupby(['Title_x','Title_y']).size().unstack().plot(kind='bar',legend=True)
```

Out[31]:

```
<AxesSubplot:xlabel='Title_x'>
```



The above graph indicates that Toy Story 2 has been rated 5 by the most users where in the case of Toy Story 1 ratings 4 has an little edge over ratings 5. so most of the users have given 4 or 5 ratings which is positive note for the movie.

Top 25 movies by viewership rating

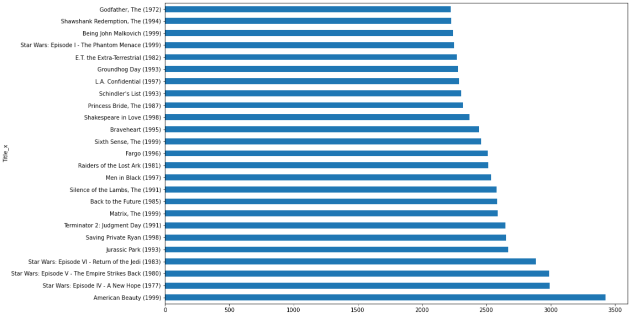
In [32]:

```
dfTop25=df_master.groupby('Title_x').size().sort_values(ascending=False)[:25]
dfTop25
```

Out[32]:
```
Title_x
American Beauty (1999)                                   3428
Star Wars: Episode IV - A New Hope (1977)                2991
Star Wars: Episode V - The Empire Strikes Back (1980)    2990
Star Wars: Episode VI - Return of the Jedi (1983)        2883
Jurassic Park (1993)                                     2672
Saving Private Ryan (1998)                               2653
Terminator 2: Judgment Day (1991)                        2649
Matrix, The (1999)                                       2590
Back to the Future (1985)                                2583
Silence of the Lambs, The (1991)                         2578
Men in Black (1997)                                      2538
Raiders of the Lost Ark (1981)                           2514
Fargo (1996)                                             2513
Sixth Sense, The (1999)                                  2459
Braveheart (1995)                                        2443
Shakespeare in Love (1998)                               2369
Princess Bride, The (1987)                               2318
Schindler's List (1993)                                  2304
L.A. Confidential (1997)                                 2288
Groundhog Day (1993)                                     2278
E.T. the Extra-Terrestrial (1982)                        2269
Star Wars: Episode I - The Phantom Menace (1999)         2250
Being John Malkovich (1999)                              2241
Shawshank Redemption, The (1994)                         2227
Godfather, The (1972)                                    2223
dtype: int64
```

In [33]:
```
plt.figure(figsize=(15,10))
dfTop25.plot(kind='barh')
```

Out[33]:
```
<AxesSubplot:ylabel='Title_x'>
```



here are the top25 movies by viewership ratings .the insights that we can take from the above plot is that American Beauty is the top most movies in the viewership rating

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

In [34]:
```
user_2696 = df_master[df_master.UserID==2696]
user_2696
```

Out[34]:

	MovieID	Title_x	Genres	UserID	Title_y	Timestamp	Gender	Age	Occupation	Zip Code
991035	350	Client, The (1994)	Drama|Mystery|Thriller	2696	3	973308886	M	25	7	24210
991036	800	Lone Star (1996)	Drama|Mystery	2696	5	973308842	M	25	7	24210
991037	1092	Basic Instinct (1992)	Mystery|Thriller	2696	4	973308886	M	25	7	24210
991038	1097	E.T. the Extra-Terrestrial (1982)	Children's|Drama|Fantasy|Sci-Fi	2696	3	973308690	M	25	7	24210
991039	1258	Shining, The (1980)	Horror	2696	4	973308710	M	25	7	24210
991040	1270	Back to the Future (1985)	Comedy|Sci-Fi	2696	2	973308676	M	25	7	24210
991041	1589	Cop Land (1997)	Crime|Drama|Mystery	2696	3	973308865	M	25	7	24210
991042	1617	L.A. Confidential (1997)	Crime|Film-Noir|Mystery|Thriller	2696	4	973308842	M	25	7	24210
991043	1625	Game, The (1997)	Mystery|Thriller	2696	4	973308842	M	25	7	24210
991044	1644	I Know What You Did Last Summer (1997)	Horror|Mystery|Thriller	2696	2	973308920	M	25	7	24210
991045	1645	Devil's Advocate, The (1997)	Crime|Horror|Mystery|Thriller	2696	4	973308904	M	25	7	24210
991046	1711	Midnight in the Garden of Good and Evil (1997)	Comedy|Crime|Drama|Mystery	2696	4	973308904	M	25	7	24210
991047	1783	Palmetto (1998)	Film-Noir|Mystery|Thriller	2696	4	973308865	M	25	7	24210
991048	1805	Wild Things (1998)	Crime|Drama|Mystery|Thriller	2696	4	973308886	M	25	7	24210
991049	1892	Perfect Murder, A (1998)	Mystery|Thriller	2696	4	973308904	M	25	7	24210
991050	2338	I Still Know What You Did Last Summer (1998)	Horror|Mystery|Thriller	2696	2	973308920	M	25	7	24210
991051	2389	Psycho (1998)	Crime|Horror|Thriller	2696	4	973308710	M	25	7	24210
991052	2713	Lake Placid (1999)	Horror|Thriller	2696	1	973308710	M	25	7	24210
991053	3176	Talented Mr. Ripley, The (1999)	Drama|Mystery|Thriller	2696	4	973308865	M	25	7	24210
991054	3386	JFK (1991)	Drama|Mystery	2696	1	973308842	M	25	7	24210

Feature Engineering:

Use column genres: Find out all the unique genres (Hint: split the data in column genre making a list and then process the data to find out only the unique categories of genres)

In [35]:
```
df_master['Genres']
```

Out[35]:
```
0                  Animation|Children's|Comedy
1          Animation|Children's|Musical|Romance
2                                        Drama
3               Action|Adventure|Fantasy|Sci-Fi
4                                    Drama|War
```

```
                              ...
1000204                           Drama|Thriller
1000205                   Comedy|Horror|Thriller
1000206                           Comedy|Romance
1000207                          Action|Thriller
1000208                             Action|Drama
Name: Genres, Length: 1000209, dtype: object
```

In [36]:

```
dfGenre=df_master['Genres'].str.split('|')
```

In [37]:

```
dfGenre
```

Out[37]:

```
0                      [Animation, Children's, Comedy]
1            [Animation, Children's, Musical, Romance]
2                                              [Drama]
3                 [Action, Adventure, Fantasy, Sci-Fi]
4                                         [Drama, War]
                              ...
1000204                              [Drama, Thriller]
1000205                     [Comedy, Horror, Thriller]
1000206                              [Comedy, Romance]
1000207                             [Action, Thriller]
1000208                                [Action, Drama]
Name: Genres, Length: 1000209, dtype: object
```

In [38]:

```
listgenres=set()
for genre in dfGenre:
    listgenres=listgenres.union(set(genre))
```

In [39]:

```
listgenres
```

Out[39]:

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

In [40]:

```
len(listgenres)
```

Out[40]:

```
18
```

Create a separate column for each genre category with a one-hot encoding (1 and 0) whether or not the movie belongs to that genre.

In [41]:

```
df_master['Genres']
```

Out[41]:

```
0                  Animation|Children's|Comedy
1          Animation|Children's|Musical|Romance
2                                        Drama
3               Action|Adventure|Fantasy|Sci-Fi
4                                    Drama|War
                              ...
1000204                           Drama|Thriller
1000205                   Comedy|Horror|Thriller
1000206                           Comedy|Romance
1000207                          Action|Thriller
1000208                             Action|Drama
Name: Genres, Length: 1000209, dtype: object
```

In [42]:

```
GeneresOnehot=df_master['Genres'].str.get_dummies('|')
```

In [43]:

```
GeneresOnehot
```

Out[43]:

	Action	Adventure	Animation	Children's	Comedy	Crime	Documentary	Drama	Fantasy	Film-Noir	Horror	Musical	Mystery	Romance	Sci-Fi	Thriller	War	Western
0	0	0	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	1	1	0	0	0	0	0	0	0	1	0	1	0	0	0	0
2	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0
3	1	1	0	0	0	0	0	0	1	0	0	0	0	0	1	0	0	0
4	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	1	0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
1000204	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	1	0	0
1000205	0	0	0	0	1	0	0	0	0	0	1	0	0	0	0	1	0	0
1000206	0	0	0	0	1	0	0	0	0	0	0	0	0	1	0	0	0	0
1000207	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0
1000208	1	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0

1000209 rows × 18 columns

In [44]:

```
# concatenating the GeneresOnehot with master data
```

In [45]:

```
df_master=pd.concat([df_master,GeneresOnehot],axis =1)
```

In [46]:

```
df_master
```

Out[46]:

	MovieID	Title_x	Genres	UserID	Title_y	Timestamp	Gender	Age	Occupation	Zip Code	...	Fantasy	Film-Noir	Horror	Musical	Mystery	Romance	Sci-Fi	Thriller	War	Western
0	1	Toy Story (1995)	Animation|Children's|Comedy	1	5	978824268	F	1	10	48067	...	0	0	0	0	0	0	0	0	0	0
1	48	Pocahontas (1995)	Animation|Children's|Musical|Romance	1	5	978824351	F	1	10	48067	...	0	0	0	1	0	1	0	0	0	0
2	150	Apollo 13 (1995)	Drama	1	5	978301777	F	1	10	48067	...	0	0	0	0	0	0	0	0	0	0
3	260	Star Wars: Episode IV - A New Hope (1977)	Action|Adventure|Fantasy|Sci-Fi	1	4	978300760	F	1	10	48067	...	1	0	0	0	0	0	1	0	0	0
4	527	Schindler's List (1993)	Drama|War	1	5	978824195	F	1	10	48067	...	0	0	0	0	0	0	0	0	1	0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
1000204	3513	Rules of Engagement (2000)	Drama|Thriller	5727	4	958489970	M	25	4	92843	...	0	0	0	0	0	0	0	1	0	0
1000205	3535	American Psycho (2000)	Comedy|Horror|Thriller	5727	2	958489970	M	25	4	92843	...	0	0	1	0	0	0	0	1	0	0
1000206	3536	Keeping the Faith (2000)	Comedy|Romance	5727	5	958489902	M	25	4	92843	...	0	0	0	0	0	1	0	0	0	0

1000207	3555	U-571 (2000)	Action|Thriller	5727	3	958490699	M	25	4	92843	...	0	0	0	0	0	0	0	1	0	0
1000208	3578	Gladiator (2000)	Action|Drama	5727	5	958490171	M	25	4	92843	...	0	0	0	0	0	0	0	0	0	0

1000209 rows × 28 columns

Determine the features affecting the ratings of any particular movie.

In [47]:
```
# convert the categorical variable Gender into nuumerical Variable
#convert gender M -0 and F-1
df_master['Gender'] = df_master['Gender'].replace("M",0)
```

In [48]:
```
df_master['Gender']
```

Out[48]:
```
0          F
1          F
2          F
3          F
4          F
          ..
1000204    0
1000205    0
1000206    0
1000207    0
1000208    0
Name: Gender, Length: 1000209, dtype: object
```

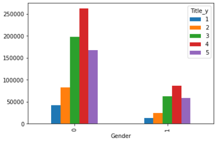
In [49]:
```
df_master['Gender']=df_master['Gender'].replace("F",1)
```

In [50]:
```
df_master['Gender']
```

Out[50]:
```
0          1
1          1
2          1
3          1
4          1
          ..
1000204    0
1000205    0
1000206    0
1000207    0
1000208    0
Name: Gender, Length: 1000209, dtype: int64
```

In [51]:
```
df_master.groupby(['Gender','Title_y']).size().unstack().plot(kind='bar',legend=True)
```

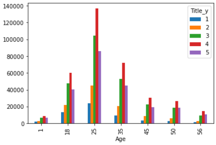
Out[51]:
```
<AxesSubplot:xlabel='Gender'>
```



Male and female both have given 4 as the most ratings. and the number of male users who have given ratings is much higher than the female users

In [52]:
```
#Age vs Ratings (Title_y) plot
df_master.groupby(['Age','Title_y']).size().unstack().plot(kind='bar',legend=True)
```

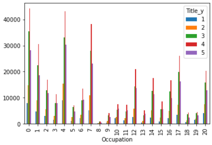
Out[52]:
```
<AxesSubplot:xlabel='Age'>
```



for all the age of users ,ratings 4 is the most opted choice by the users.

In [53]:
```
#Occupation vs Ratings plot
df_master.groupby(['Occupation','Title_y']).size().unstack().plot(kind='bar',legend=True)
```

Out[53]:
```
<AxesSubplot:xlabel='Occupation'>
```

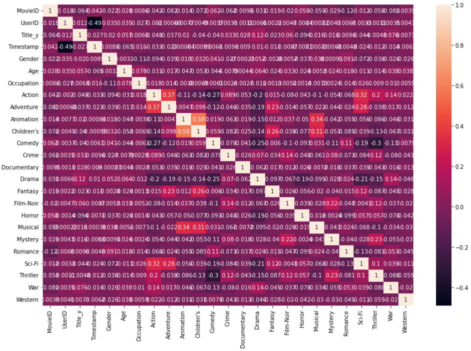


In [54]:
```
df_master.columns
```

Out[54]:
```
Index(['MovieID', 'Title_x', 'Genres', 'UserID', 'Title_y', 'Timestamp',
       'Gender', 'Age', 'Occupation', 'Zip Code', 'Action', 'Adventure',
       'Animation', 'Children's', 'Comedy', 'Crime', 'Documentary', 'Drama',
       'Fantasy', 'Film-Noir', 'Horror', 'Musical', 'Mystery', 'Romance',
       'Sci-Fi', 'Thriller', 'War', 'Western'],
      dtype='object')
```

In [55]:
```
plt.figure(figsize=(15,10))
sns.heatmap(df_master.corr(),annot=True)
```

Out[55]:
```
<AxesSubplot:>
```



```
In [56]: # to build appropriate model to predict movie ratings
         new_data = df_master
```

```
In [57]: new_data.columns

Out[57]: Index(['MovieID', 'Title_x', 'Genres', 'UserID', 'Title_y', 'Timestamp',
                'Gender', 'Age', 'Occupation', 'Zip Code', 'Action', 'Adventure',
                'Animation', 'Children's', 'Comedy', 'Crime', 'Documentary', 'Drama',
                'Fantasy', 'Film-Noir', 'Horror', 'Musical', 'Mystery', 'Romance',
                'Sci-Fi', 'Thriller', 'War', 'Western'],
               dtype='object')
```

```
In [58]: features = new_data[['MovieID','Age','Occupation','Gender']].values
```

```
In [59]: features

Out[59]: array([[   1,    1,   10,    1],
                [  48,    1,   10,    1],
                [ 150,    1,   10,    1],
                ...,
                [3536,   25,    4,    0],
                [3555,   25,    4,    0],
                [3578,   25,    4,    0]], dtype=int64)
```

```
In [60]: label=new_data[['Title_y']].values
```

```
In [61]: label

Out[61]: array([[5],
                [5],
                [5],
                ...,
                [5],
                [3],
                [5]], dtype=int64)
```

```
In [62]: from sklearn.model_selection import train_test_split
         X_train,X_test,Y_train,Y_test=train_test_split(features,label,test_size=0.20,random_state=42)
```

```
In [63]: X_train.shape

Out[63]: (800167, 4)
```

```
In [64]: X_test.shape

Out[64]: (200042, 4)
```

```
In [65]: from sklearn.linear_model import LinearRegression
         lr=LinearRegression()
```

```
In [66]: lr.fit(X_train,Y_train)

Out[66]: LinearRegression()
```

```
In [67]: lr.predict(X_test)

Out[67]: array([[3.59272104],
                [3.59115676],
                [3.62210238],
                ...,
                [3.57009906],
                [3.48812167],
                [3.70360722]])
```

```
In [68]: Y_test

Out[68]: array([[5],
                [4],
                [4],
                ...,
                [2],
                [3],
                [5]], dtype=int64)
```

```
In [69]: y_predict=lr.predict(X_test)
```

```
In [70]: y_predict

Out[70]: array([[3.59272104],
                [3.59115676],
                [3.62210238],
                ...,
                [3.57009906],
                [3.48812167],
                [3.70360722]])
```

```
In [71]: #error in regression
         from sklearn.metrics import mean_squared_error
         print('Mean Squared Error',mean_squared_error(Y_test,y_predict))

         Mean Squared Error 1.2385729136665815
```

```
In [72]: from sklearn.metrics import r2_score
         print('R2 score',r2_score(Y_test,y_predict))

         R2 score 0.008511269991118712
```

```
In [ ]:
```