```
cd C:\Users\dhant\OneDrive\Desktop\simplilearn\DS with python\
project_2\Data science with Python 1

import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
%matplotlib inline
import seaborn as sns
import pylab as p
import missingno as msno
import warnings
import calendar
warnings.filterwarnings('ignore')

Movies_data=pd.read_csv("C:\\Users\\dhant\\OneDrive\\Desktop\\
simplilearn\\DS with python\\project_2\\Data_science_with_Python_1\\
movies.dat",sep="::", header=None, names=['MovieID','Title','Genres'],

                        dtype={'MovieID': np.int32, 'Title': np.str,
'Genres': np.str}, engine='python')
Movies_data.head(3)
```

	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

```
users_data = pd.read_csv("C:\\Users\\dhant\\OneDrive\\Desktop\\
simplilearn\\DS with python\\project_2\\Data_science_with_Python_1\\
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')
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

```
ratings_data = pd.read_csv("C:\\Users\\dhant\\OneDrive\\Desktop\\
simplilearn\\DS with python\\project_2\\Data_science_with_Python_1\\
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')
ratings_data.head(3)
```

	UserID	MovieID	Rating	Timestamp
0	1	1193	5	978300760

```
1       1      661       3  978302109
2       1      914       3  978301968
```

Create a new dataset [Master_Data] with the following columns MovieID Title UserID Age Gender Occupation Rating.

```
ratings_user = pd.merge(ratings_data,users_data, on=['UserID'])
ratings_movie = pd.merge(ratings_data,Movies_data, on=['MovieID'])

master_data = pd.merge(ratings_user,ratings_movie,
                       on=['UserID', 'MovieID', 'Rating'])[['MovieID',
'Title', 'UserID', 'Age', 'Gender', 'Occupation', "Rating"]]
master_data.head()
```

```
   MovieID                                  Title  UserID  Age Gender
\
0     1193  One Flew Over the Cuckoo's Nest (1975)       1    1      F

1      661        James and the Giant Peach (1996)       1    1      F

2      914                   My Fair Lady (1964)       1    1      F

3     3408                 Erin Brockovich (2000)       1    1      F

4     2355                   Bug's Life, A (1998)       1    1      F


   Occupation  Rating
0          10       5
1          10       3
2          10       3
3          10       4
4          10       5
```

```
master_data.shape
```

```
(1000209, 7)
```

```
master_data.size
```

```
7001463
```

```
master_data.columns
```

```
Index(['MovieID', 'Title', 'UserID', 'Age', 'Gender', 'Occupation',
'Rating'], dtype='object')
```

```
master_data.dtypes
```

```
MovieID        int32
Title         object
UserID         int32
Age            int32
Gender        object
Occupation     int32
Rating         int32
dtype: object
```

```
master_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 1000209 entries, 0 to 1000208
Data columns (total 7 columns):
 #   Column      Non-Null Count    Dtype
---  ------      --------------    -----
 0   MovieID     1000209 non-null  int32
 1   Title       1000209 non-null  object
 2   UserID      1000209 non-null  int32
 3   Age         1000209 non-null  int32
 4   Gender      1000209 non-null  object
 5   Occupation  1000209 non-null  int32
 6   Rating      1000209 non-null  int32
dtypes: int32(5), object(2)
memory usage: 42.0+ MB
```

```
master_data.nunique()
```

```
MovieID       3706
Title         3706
UserID        6040
Age              7
Gender           2
Occupation      21
Rating           5
dtype: int64
```

Showing Basics Statistics

```
master_data.describe().style.background_gradient(axis=1,cmap=sns.light
_palette('green', as_cmap=True))
```

```
<pandas.io.formats.style.Styler at 0x187e5abd0d0>
```

```
master_data.describe(include=object)
```

```
                         Title   Gender
count                  1000209  1000209
unique                    3706        2
top     American Beauty (1999)        M
freq                      3428   753769
```

```
print('The dataset has {0} samples.'.format(len(master_data)))
```

```
The dataset has 1000209 samples.
```

```
import pandas_profiling as pp
from pandas_profiling import ProfileReport
pp.ProfileReport(master_data)
```


```
{"version_major":2,"version_minor":0,"model_id":"9edb8f2556f9435ca3764
353f220f32a"}
```

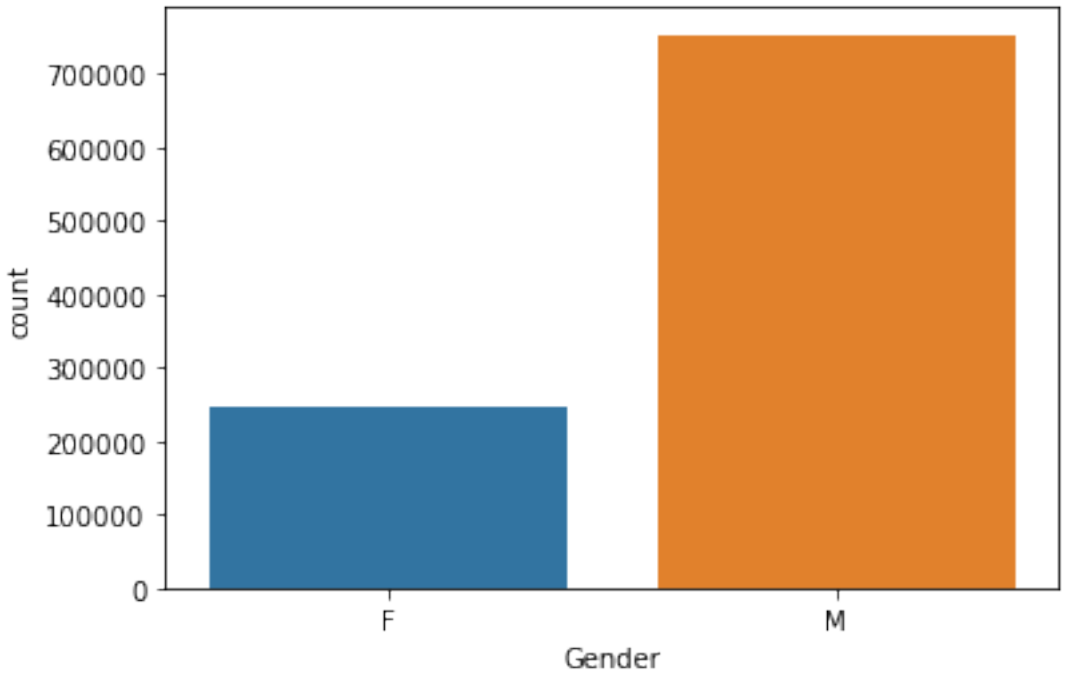
```
{"version_major":2,"version_minor":0,"model_id":"e03d44a44f81442082f51
c5abf9e6212"}
```

```
{"version_major":2,"version_minor":0,"model_id":"be6fa6a7ef9548bc800f8
d2a89dd3627"}
```


```
<IPython.core.display.HTML object>
```

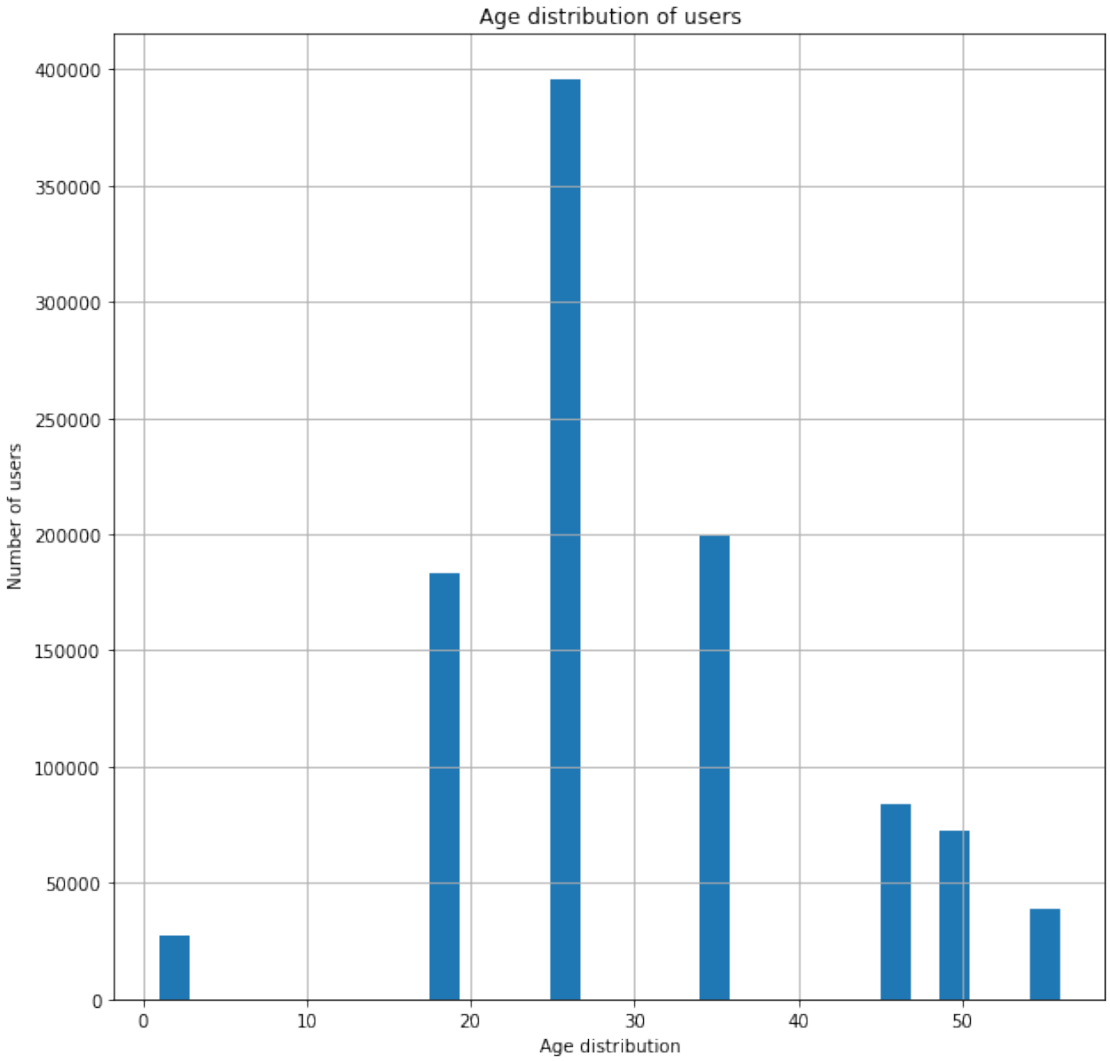
```
master_data['Gender'].value_counts()
sns.countplot('Gender',data=master_data)
```

```
<AxesSubplot:xlabel='Gender', ylabel='count'>
```



User Age Distribution

```
plt.figure(figsize=(10,10))
master_data['Age'].hist(bins=30)
plt.title("Age distribution of users")
plt.xlabel("Age distribution")
plt.ylabel(" Number of users")
plt.show()
```



User rating of the movie “Toy Story”

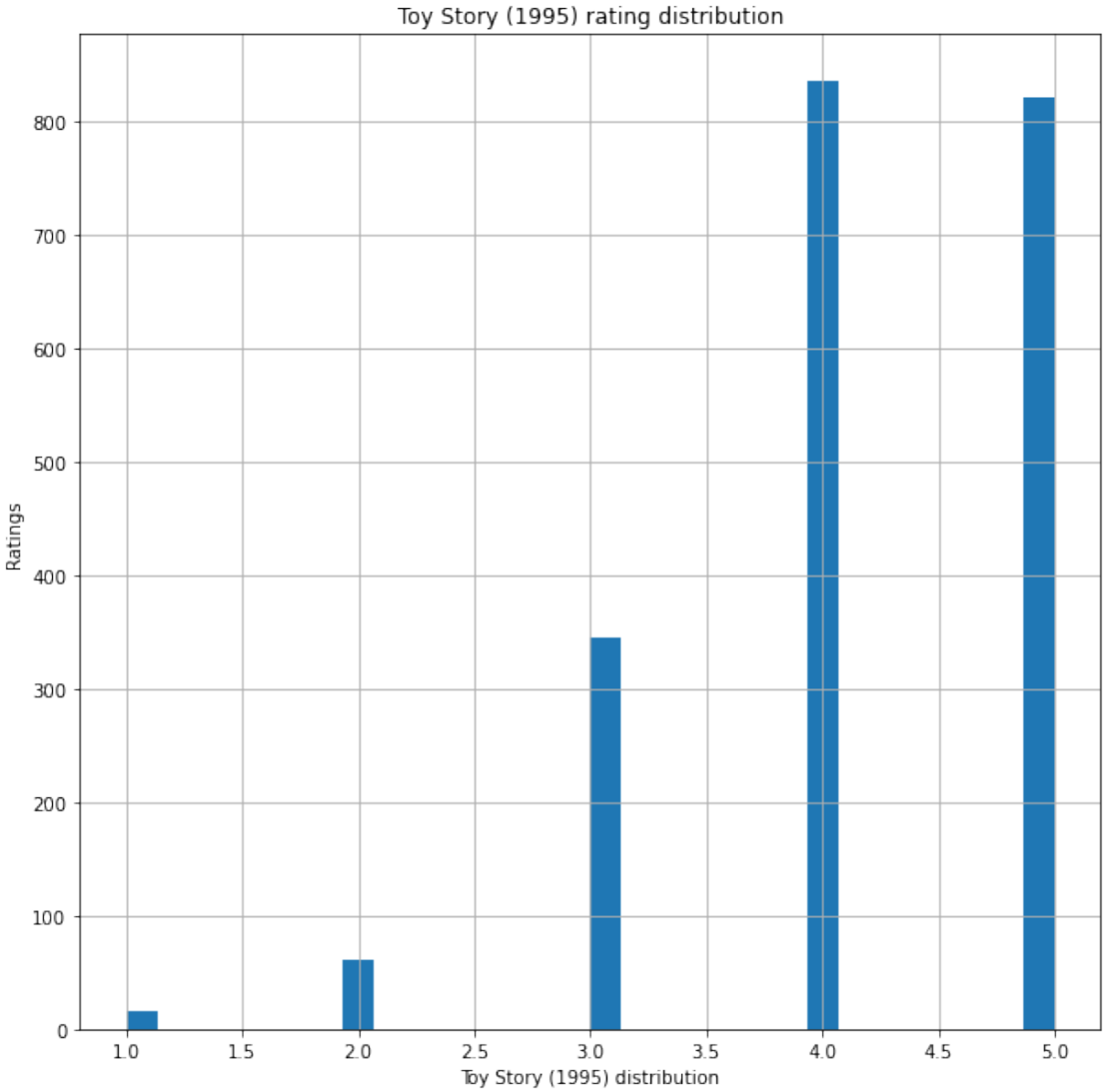
```
Toy_story = master_data[master_data.Title == "Toy Story (1995)"]
Toy_story
```

```
        MovieID             Title  UserID  Age Gender  Occupation
Rating
40            1  Toy Story (1995)       1    1      F          10
5
```

```
469           1  Toy Story (1995)       6   50      F           9
4
581           1  Toy Story (1995)       8   25      M          12
4
711           1  Toy Story (1995)       9   25      M          17
5
837           1  Toy Story (1995)      10   35      F           1
5
...         ...               ...     ...  ...    ...         ...
...
997248        1  Toy Story (1995)    6022   25      M          17
5
997541        1  Toy Story (1995)    6025   25      F           1
5
998170        1  Toy Story (1995)    6032   45      M           7
4
998360        1  Toy Story (1995)    6035   25      F           1
4
999870        1  Toy Story (1995)    6040   25      M           6
3

[2077 rows x 7 columns]
```

```
plt.figure(figsize=(10,10))
Toy_story['Rating'].hist(bins=30)
plt.title("Toy Story (1995) rating distribution")
plt.xlabel("Toy Story (1995) distribution")
plt.ylabel("Ratings")
plt.show()
```

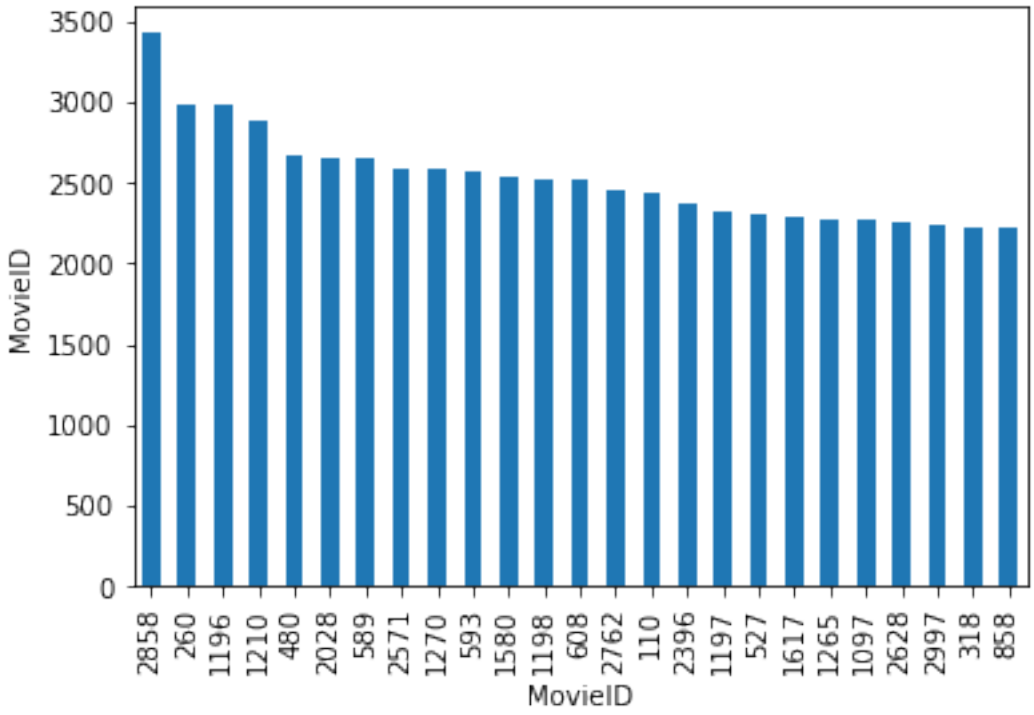


Top 25 movies by viewership rating

```
#3.Top 25 movies by viewership rating
top25_movies =
ratings_data.groupby(['MovieID']).size().sort_values(ascending=False)
[:25]
top25_movies
plt.ylabel("MovieID")

plt.xlabel("Viewership Count")
top25_movies.plot(kind="bar")
```

```
<AxesSubplot:xlabel='MovieID', ylabel='MovieID'>
```




```
movie_rating=ratings_data.groupby(['MovieID'])
avg_movie_rating=movie_rating.agg({'Rating':'mean'})
top25_movies=avg_movie_rating.sort_values('Rating',ascending=False).he
ad(25)

pd.merge(top25_movies, Movies_data, how='left', left_on=['MovieID'],
right_on=['MovieID'])
```

```
    MovieID    Rating
Title  \
0       989  5.000000          Schlafes Bruder (Brother of Sleep)
(1995)
1      3881  5.000000                              Bittersweet Motel
(2000)
2      1830  5.000000                               Follow the Bitch
(1998)
3      3382  5.000000                                Song of Freedom
(1936)
4       787  5.000000                  Gate of Heavenly Peace, The
(1995)
5      3280  5.000000                                      Baby, The
(1973)
6      3607  5.000000                              One Little Indian
(1973)
7      3233  5.000000                                  Smashing Time
(1967)
8      3172  5.000000                               Ulysses (Ulisse)
(1954)
```

```
9      3656  5.000000                                        Lured
(1947)
10     3245  4.800000                I Am Cuba (Soy Cuba/Ya Kuba)
(1964)
11       53  4.750000                                    Lamerica
(1994)
12     2503  4.666667                             Apple, The (Sib)
(1998)
13     2905  4.608696                                     Sanjuro
(1962)
14     2019  4.560510  Seven Samurai (The Magnificent Seven)
(Shichin...
15      318  4.554558                    Shawshank Redemption, The
(1994)
16      858  4.524966                          Godfather, The
(1972)
17      745  4.520548                          Close Shave, A
(1995)
18       50  4.517106                       Usual Suspects, The
(1995)
19      527  4.510417                          Schindler's List
(1993)
20     1148  4.507937                       Wrong Trousers, The
(1993)
21     2309  4.500000         Inheritors, The (Die Siebtelbauern)
(1998)
22     1795  4.500000                Callejón de los milagros, El
(1995)
23     2480  4.500000              Dry Cleaning (Nettoyage à sec)
(1997)
24      439  4.500000                          Dangerous Game
(1993)

                     Genres
0                     Drama
1                Documentary
2                     Comedy
3                     Drama
4                Documentary
5                     Horror
6       Comedy|Drama|Western
7                     Comedy
8                  Adventure
9                      Crime
10                    Drama
11                    Drama
12                    Drama
13          Action|Adventure
14              Action|Drama
15                    Drama
```

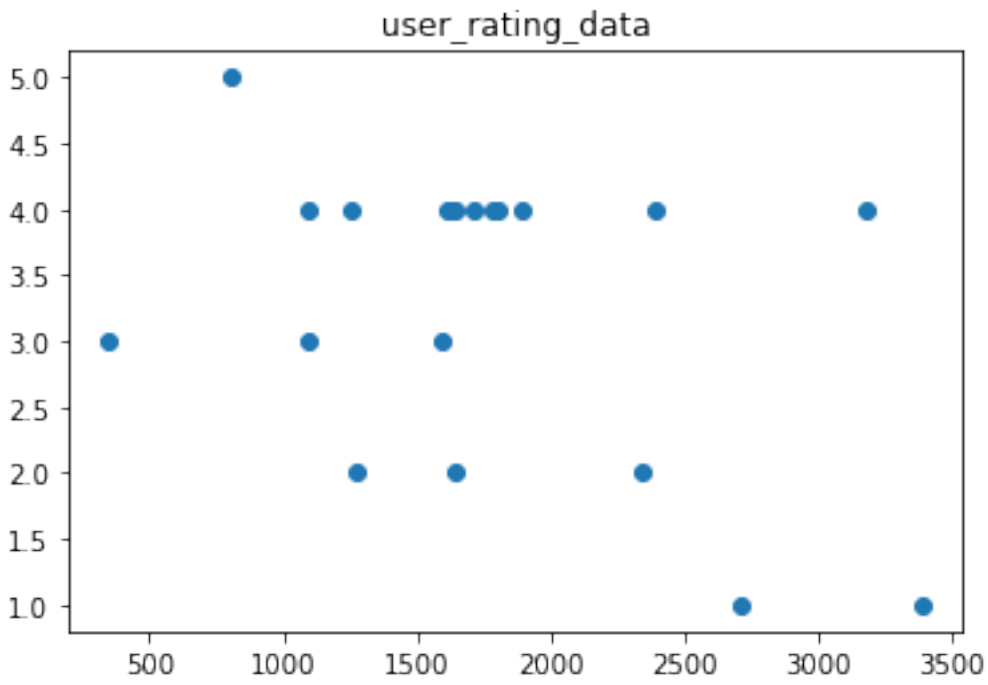
```
16         Action|Crime|Drama
17  Animation|Comedy|Thriller
18             Crime|Thriller
19                  Drama|War
20           Animation|Comedy
21                      Drama
22                      Drama
23                      Drama
24                      Drama
```

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

```
user_rating_data=ratings_data[ratings_data['UserID']==2696]
user_rating_data
```

```
        UserID  MovieID  Rating  Timestamp
440667    2696     1258       4  973308710
440668    2696     1270       2  973308676
440669    2696     1617       4  973308842
440670    2696     1625       4  973308842
440671    2696     1644       2  973308920
440672    2696     1645       4  973308904
440673    2696     1805       4  973308886
440674    2696     1892       4  973308904
440675    2696      800       5  973308842
440676    2696     2338       2  973308920
440677    2696     1711       4  973308904
440678    2696     3176       4  973308865
440679    2696     2389       4  973308710
440680    2696     1589       3  973308865
440681    2696     2713       1  973308710
440682    2696     3386       1  973308842
440683    2696     1783       4  973308865
440684    2696      350       3  973308886
440685    2696     1092       4  973308886
440686    2696     1097       3  973308690
```

```
plt.scatter(x=user_rating_data['MovieID'],y=user_rating_data['Rating']
)
plt.title('user_rating_data')
plt.show()
```



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)

```
x2=Movies_data['Genres'].nunique()
x2
```

```
301
```

```
movies_genres = Movies_data['Genres'].str.split('|')
movies_genres
```

```
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: Genres, Length: 3883, dtype: object
```

```
movies_genres1 = Movies_data['Genres'].str.get_dummies('|')
movies_genres1
```

```
      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]
```

```
res_col = []
for v in movies_genres:
    for i in v:
        if i not in res_col:
            res_col.append(i)

res_col.append("Gender")
res_col.append("Age")
res_col.append("Rating")

df = pd.DataFrame(columns=res_col)
res = master_data.merge(Movies_data, on = ['MovieID'], how="left")
[["Genres","Rating","Gender", "Age"]]

for index, row in res.head(2000).iterrows():
    tmp = row.Genres.split("|")

    for i in tmp:
       # print(i)
        df.loc[index,i] = 1
        df.loc[index,"Gender"] = res.loc[index,"Gender"]
        df.loc[index,"Age"] = res.loc[index,"Age"]
        df.loc[index,"Rating"] = res.loc[index,"Rating"]

    df.loc[index,df.columns[~df.columns.isin(tmp+
["Gender","Rating","Age"])]] = 0
    #df.dropna(inplace=True)

df
```

```
     Animation Children's Comedy Adventure Fantasy Romance Drama
Action Crime  \
0            0          0      0         0       0       0     1
0     0
1            1          1      0         0       0       0     0
```

```
0     0
2            0          0      0         0       0       1     0
0     0
3            0          0      0         0       0       0     1
0     0
4            1          1      1         0       0       0     0
0     0
...        ...        ...    ...       ...     ...     ...   ...    ..
.   ...
1995         0          0      0         0       0       0     1
0     0
1996         1          1      1         0       0       0     0
0     0
1997         1          1      1         0       0       0     0
0     0
1998         0          0      0         1       0       1     1
1     0
1999         0          0      0         1       1       1     0
0     0

     Thriller  ... Sci-Fi Documentary War Musical Mystery Film-Noir
Western  \
0           0  ...      0           0   0       0       0         0
0
1           0  ...      0           0   0       1       0         0
0
2           0  ...      0           0   0       1       0         0
0
3           0  ...      0           0   0       0       0         0
0
4           0  ...      0           0   0       0       0         0
0
...       ...  ...    ...         ...  ..     ...     ...       ...
...
1995        0  ...      0           0   0       0       0         0
0
1996        0  ...      0           0   0       0       0         0
0
1997        0  ...      0           0   0       0       0         0
0
1998        0  ...      0           0   0       0       0         0
0
1999        0  ...      0           0   0       0       0         0
0

     Gender Age Rating
0         F   1      5
1         F   1      3
2         F   1      3
3         F   1      4
```

```
4         F   1      5
...     ...  ..    ...
1995      F  18      5
1996      F  18      5
1997      F  18      3
1998      F  18      1
1999      F  18      5

[2000 rows x 21 columns]
```

```
from sklearn import datasets
from sklearn.metrics import confusion_matrix
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder

X = df[df.columns[~df.columns.isin(["Rating"])]]
y = df.Rating

# dividing X, y into train and test data
X_train, X_test, y_train, y_test = train_test_split(X, y, random_state
= 0)

number = LabelEncoder()
X_train.Gender = number.fit_transform(X_train["Gender"].astype("str"))
X_test.Gender = number.fit_transform(X_test["Gender"].astype("str"))
y_train = number.fit_transform(y_train.astype("int"))
y_test = number.fit_transform(y_test.astype("int"))

#SVM

from sklearn.svm import SVC
svm_model_linear = SVC(kernel = 'linear', C = 1).fit(X_train, y_train)

svm_predictions = svm_model_linear.predict(X_test)


accuracy = svm_model_linear.score(X_test, y_test)

cm = confusion_matrix(y_test, svm_predictions)
accuracy
```

```
0.374
```

```
#KNN

from sklearn.neighbors import KNeighborsClassifier
knn = KNeighborsClassifier(n_neighbors = 8).fit(X_train, y_train)

# accuracy on X_test
accuracy = knn.score(X_test, y_test)
```

```
# creating a confusion matrix
knn_predictions = knn.predict(X_test)
cm = confusion_matrix(y_test, knn_predictions)

accuracy
```

```
0.384
```

```
#Naive Bayes classifier

from sklearn.naive_bayes import GaussianNB
GN = GaussianNB().fit(X_train, y_train)
GN_predictions = GN.predict(X_test)

# accuracy on X_test
accuracy = GN.score(X_test, y_test)

# creating a confusion matrix
cm = confusion_matrix(y_test, GN_predictions)

accuracy
```

```
0.076
```