

Project to Submit – Project 3

Movielens Case Study

Background of Problem Statement :

The GroupLens Research Project is a research group in the Department of Computer Science and Engineering at the University of Minnesota. Members of the GroupLens Research Project are involved in many research projects related to the fields of information filtering, collaborative filtering, and recommender systems. The project is led by professors John Riedl and Joseph Konstan. The project began to explore automated collaborative filtering in 1992 but is most well known for its worldwide trial of an automated collaborative filtering system for Usenet news in 1996. Since then the project has expanded its scope to research overall information by filtering solutions, integrating into content-based methods, as well as, improving current collaborative filtering technology.

Problem Objective :

Here, we ask you to perform the analysis using the Exploratory Data Analysis technique. You need to find features affecting the ratings of any particular movie and build a model to predict the movie ratings.

Domain: Entertainment

Analysis Tasks to be performed:

- Import the three datasets
- Create a new dataset [Master_Data] with the following columns MovieID Title UserID Age Gender Occupation Rating. (Hint: (i) Merge two tables at a time. (ii) Merge the tables using two primary keys MovieID & UserID)
- Explore the datasets using visual representations (graphs or tables), also include your comments on the following:
 1. User Age Distribution
 2. User rating of the movie "Toy Story"
 3. Top 25 movies by viewership rating
 4. Find the ratings for all the movies reviewed by for a particular user of user id = 2696
- Feature Engineering:

Use column genres:

1. 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)
2. 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.
3. Determine the features affecting the ratings of any particular movie.
4. Develop an appropriate model to predict the movie ratings

Dataset Description :

These files contain 1,000,209 anonymous ratings of approximately 3,900 movies made by 6,040 MovieLens users who joined MovieLens in 2000.

Ratings.dat

Format - UserID::MovieID::Rating::Timestamp

Field	Description
UserID	Unique identification for each user
MovieID	Unique identification for each movie
Rating	User rating for each movie

Timestamp	Timestamp generated while adding user review
-----------	--

- UserIDs range between 1 and 6040
- The MovieIDs range between 1 and 3952
- Ratings are made on a 5-star scale (whole-star ratings only)
- A timestamp is represented in seconds since the epoch is returned by time(2)
- Each user has at least 20 ratings

Users.dat

Format - UserID::Gender::Age::Occupation::Zip-code

Field	Description
UserID	Unique identification for each user
Genre	Category of each movie
Age	User's age
Occupation	User's Occupation
Zip-code	Zip Code for the user's location

All demographic information is provided voluntarily by the users and is not checked for accuracy. Only users who have provided demographic information are included in this data set.

- Gender is denoted by an "M" for male and "F" for female
- Age is chosen from the following ranges:

Value	Description
1	"Under 18"
18	"18-24"
25	"25-34"
35	"35-44"

45	"45-49"
50	"50-55"
56	"56+"

- Occupation is chosen from the following choices:

Value	Description
0	"other" or not specified
1	"academic/educator"
2	"artist"
3	"clerical/admin"
4	"college/grad student"
5	"customer service"
6	"doctor/health care"
7	"executive/managerial"
8	"farmer"
9	"homemaker"
10	"K-12 student"
11	"lawyer"

12	"programmer"
13	"retired"
14	"sales/marketing"
15	"scientist"
16	"self-employed"
17	"technician/engineer"
18	"tradesman/craftsman"
19	"unemployed"
20	"writer"

Movies.dat

Format - MovieID::Title::Genres

Field	Description
MovieID	Unique identification for each movie
Title	A title for each movie
Genres	Category of each movie

Titles are identical to titles provided by the IMDB (including year of release)

- Genres are pipe-separated and are selected from the following genres:
 1. Action
 2. Adventure
 3. Animation
 4. Children's
 5. Comedy
 6. Crime
 7. Documentary

- 8. Drama
- 9. Fantasy
- 10. Film-Noir
- 11. Horror
- 12. Musical
- 13. Mystery
- 14. Romance
- 15. Sci-Fi
- 16. Thriller
- 17. War
- 18. Western
- Some MovieIDs do not correspond to a movie due to accidental duplicate entries and/or test entries
- Movies are mostly entered by hand, so errors and inconsistencies may exist

Problem Objective :

Here, we ask you to perform the analysis using the Exploratory Data Analysis technique. You need to find features affecting the ratings of any particular movie and build a model to predict the movie ratings.

Domain: Entertainment

Analysis Tasks to be performed:

- Import the three datasets
- Create a new dataset [Master_Data] with the following columns MovieID Title UserID Age Gender Occupation Rating.
- Explore the datasets using visual representations (graphs or tables), also include your comments on the following:
 1. User Age Distribution
 2. User rating of the movie "Toy Story"
 3. Top 25 movies by viewership rating
 4. Find the ratings for all the movies reviewed by for a particular user of user id = 2696
- Feature Engineering:
Use column genres:
 1. 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)
 2. 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.
 3. Determine the features affecting the ratings of any particular movie.
 4. Develop an appropriate model to predict the movie ratings

Write Up:

Analysis Tasks to be performed:

- Import the three datasets (Done) –
 - Movies.pd, Users.pd, Ratings.pd
- Create a new dataset [Master_Data] with the following columns MovieID Title UserID Age Gender Occupation Rating
 - First merging on **Common field MovieID** on Movies & Ratings database
 - Fields that we do not need are: Genre (column - 3) & Timestamp (column - 6)
 - Now merging User database with Master database
 - Dropping the field – Zip Code
 - Rearrange columns

From	To
'MovieID', 'Title', 'UserID', 'Rating' , 'Gender', 'Age', 'Occupation', 'zip-code'	<ul style="list-style-type: none">◦ MovieID – No Change◦ Title - No Change◦ UserID - No Change◦ Age◦ Gender◦ Occupation -◦ Rating – From No 4 to End

-
- Explore the datasets using visual representations (graphs or tables), also include your comments on the following:
 - These graphs are made on the Master database that is just created
 - 5. User Age Distribution – Barchart
 - 6. User rating of the movie “Toy Story” – Barchart & line graph
 - 7. Top 25 movies by viewership rating – Table Representation
 - 8. Find the ratings for all the movies reviewed by for a particular user of user id = 2696 - Table Representation
- Feature Engineering:
Use column genres:
 - 5. Find out all the unique genres
 - Here – using the original database Movies database
 - #Movie Genre does not house unique in terms of genre. It is showing the combines unique genre, like Drama|Romance, Horror|Suspense|Thriller etc.
 - Joining the dataset rating_df & movie_df – left join
 - 6. 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.
 - Creating dummies for each possible genre, such as Sci-Fi or Drama, and having a single column for each. Creating dummies means creating 0s and 1s
 - 7. Determine the features affecting the ratings of any particular movie.
 - Using Correlation function the factor affecting was calculated
 - 8. Develop an appropriate model to predict the movie ratings
 - Linear model was not found to be good for this dataset.
 - 1. The accuracy was 5%
 - Logistic Regression was tested
 - 1. For Logistics Regression the accuracy is 35%

Code:

Project 2 Simplilearn Moviens - You need to find features affecting the ratings of any particular movie and build a model to predict the movie ratings.

Import Right Libraries

```
import numpy as np
import pandas as pd
from pandas import Series, DataFrame
import matplotlib.pyplot as plt
from matplotlib import style
import seaborn as sns
%matplotlib inline
```

- Import the three datasets

Movie Data base in movies_df

```
movies_df = pd.read_csv('D:\Alok-OneDrive\OneDrive\Knowledge Center\AI - ML\Python\Examples\Projects\\movies.dat', encoding='ISO-8859-1', sep='::',
engine='python',
names=['MovieID','Title','Genres'], header=None)
movies_df.head()
```

In [2]:

Out[3]:

		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

Users Data base in users_df

```
users_df = pd.read_csv('D:\Alok-OneDrive\OneDrive\Knowledge Center\AI - ML\Python\Examples\Projects\\users.dat', encoding='ISO-8859-1',
sep='::', engine='python',
names=['UserID','Gender','Age', 'Occupation', 'zip-code'], header=None)
users_df.head()
```

In [4]:

Out[4]:

	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

Ratings Data base in ratings_df

```
In [5]:
ratings_df = pd.read_csv('D:\Alok-OneDrive\OneDrive\Knowledge Center\AI - ML\Python\Examples\Projects\ratings.dat', encoding='ISO-8859-1', sep='::',
engine='python',
names=['UserID', 'MovieID', 'Rating', 'Timestamp'], header=None)
ratings_df.head()
```

Out[5]:

	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

- Create a new dataset [Master_Data] with the following columns MovieID Title UserID Age Gender Occupation Rating. (Hint: (i) Merge two tables at a time. (ii) Merge the tables using two primary keys MovieID & UserID)

First merging on Common field MovieID on Movies & Ratings database

```
In [6]:
Master_df = pd.merge(movies_df, ratings_df,
on='MovieID')
Master_df.info()
```

```

Master_df.head()
Master_df = Master_df[['MovieID', 'Title', 'UserID', 'Rating']]
Master_df.head()

<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       1000209 non-null  object  
 2   Genres      1000209 non-null  object  
 3   UserID       1000209 non-null  int64  
 4   Rating      1000209 non-null  int64  
 5   Timestamp    1000209 non-null  int64  
dtypes: int64(4), object(2)
memory usage: 53.4+ MB

```

Out[6]:

	MovieID	Title	UserID	Rating
0	1	Toy Story (1995)	1	5
1	1	Toy Story (1995)	6	4
2	1	Toy Story (1995)	8	4
3	1	Toy Story (1995)	9	5
4	1	Toy Story (1995)	10	5

Now merging on Common field MovieID on Master database on UserID

```

Master_df = pd.merge(Master_df, users_df,
                     on='UserID')
Master_df.info()
Master_df.head()

<class 'pandas.core.frame.DataFrame'>
Int64Index: 1000209 entries, 0 to 1000208
Data columns (total 8 columns):
 #   Column      Non-Null Count   Dtype  
---  --  
 0   MovieID     1000209 non-null  int64  
 1   Title       1000209 non-null  object  
 2   UserID       1000209 non-null  int64  
 3   Rating      1000209 non-null  int64  
 4   Gender      1000209 non-null  object  
 5   Age         1000209 non-null  int64  

```

In [7]:

```

6    Occupation  1000209 non-null  int64
7    zip-code     1000209 non-null  object
dtypes: int64(5), object(3)
memory usage: 68.7+ MB

```

Out[7]:

	MovieID		Title	UserID	Rating	Gender	Age	Occupation	zip-code
0	1		Toy Story (1995)	1	5	F	1	10	48067
1	48		Pocahontas (1995)	1	5	F	1	10	48067
2	150		Apollo 13 (1995)	1	5	F	1	10	48067
3	260	Star Wars: Episode IV - A New Hope (1977)		1	4	F	1	10	48067
4	527	Schindler's List (1993)		1	5	F	1	10	48067

Rearranged the Master database as per the requirement

In [8]:

```

# Get the DataFrame column names as a list
clist = list(Master_df.columns)
Master_df.reindex(columns=['MovieID',
                          'Title',
                          'UserID',
                          'Age',
                          'Gender',
                          'Occupation',
                          'Rating'])
Master_df.head()

```

Out[8]:

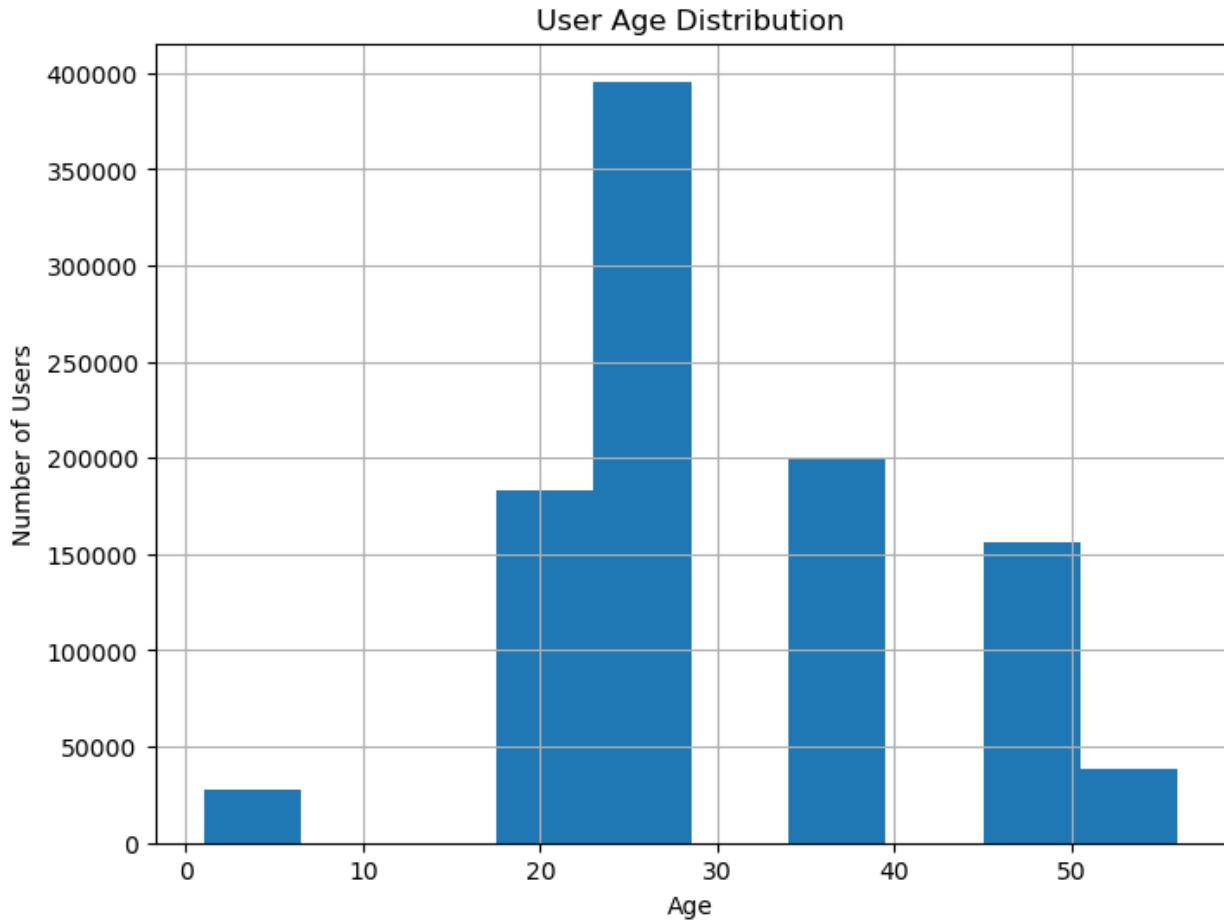
	MovieID		Title	UserID	Rating	Gender	Age	Occupation	zip-code
0	1		Toy Story (1995)	1	5	F	1	10	48067
1	48		Pocahontas (1995)	1	5	F	1	10	48067
2	150		Apollo 13 (1995)	1	5	F	1	10	48067
3	260	Star Wars: Episode IV - A New Hope (1977)		1	4	F	1	10	48067

MovieID		Title	UserID	Rating	Gender	Age	Occupation	zip-code
4	527	Schindler's List (1993)	1	5	F	1	10	48067

- Explore the datasets using visual representations (graphs or tables), also include your comments on the following: User Age Distribution These graphs are made on Master database that is currently created

In [9]:

```
plt.figure(figsize=(8, 6))
Master_df.Age.hist()
plt.title('User Age Distribution')
plt.xlabel('Age')
plt.ylabel('Number of Users')
plt.show()
```



- Explore the datasets using visual representations (graphs or tables), also include your comments on the following:
 - o These graphs are made on the Master database that is just created

1. User rating of the movie "Toy Story"

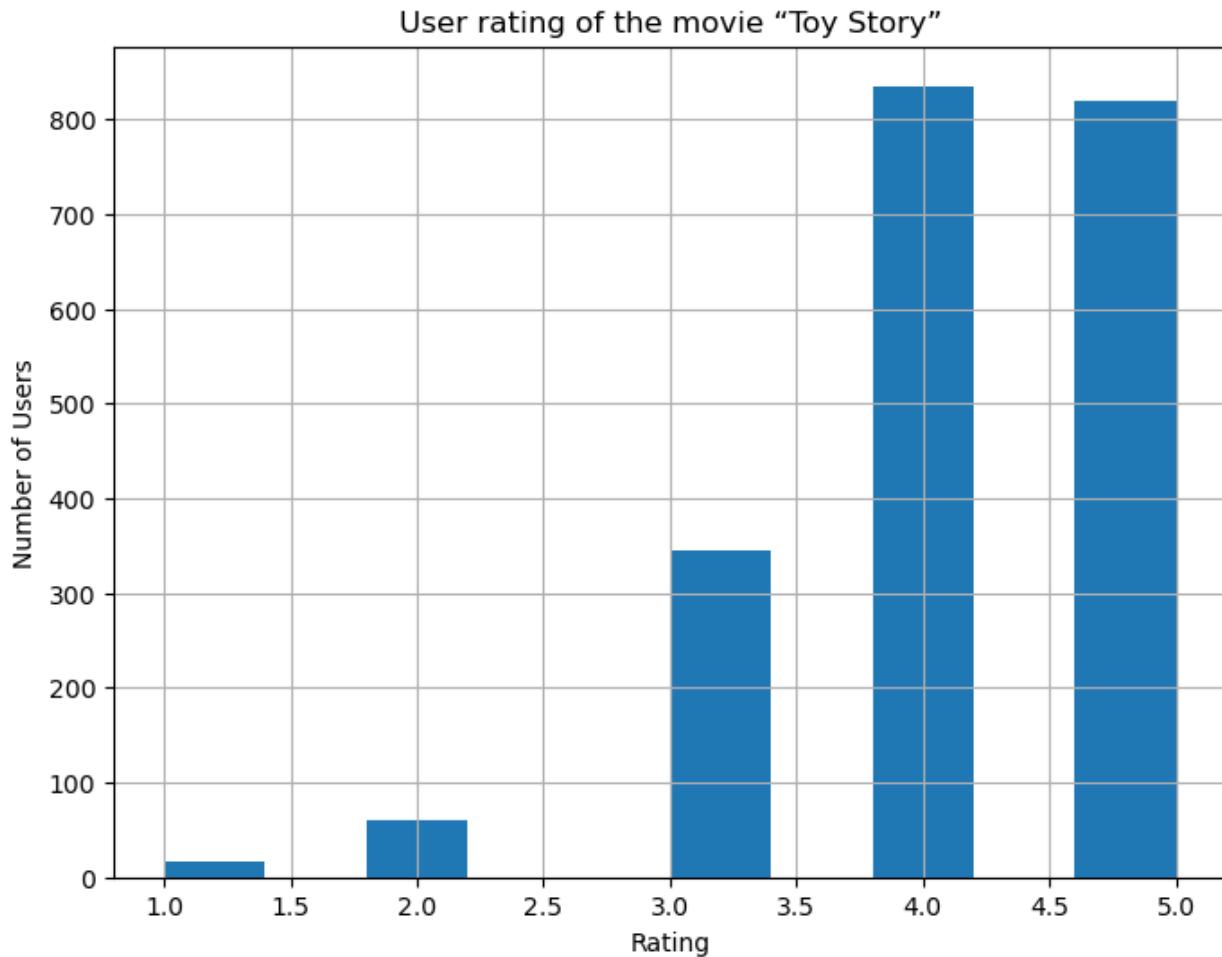
In [10]:

```
plt.figure(figsize=(8, 6))
```

```

movies_grouped = Master_df.groupby('Title')
toy_story = movies_grouped.get_group('Toy Story (1995)')
toy_story['Rating'].hist()
plt.title('User rating of the movie "Toy Story"')
plt.xlabel('Rating')
plt.ylabel('Number of Users')
plt.show()

```



In [11]:

```

#User rating of the movie "Toy Story"
Master_df[Master_df.Title == "Toy Story (1995)"]

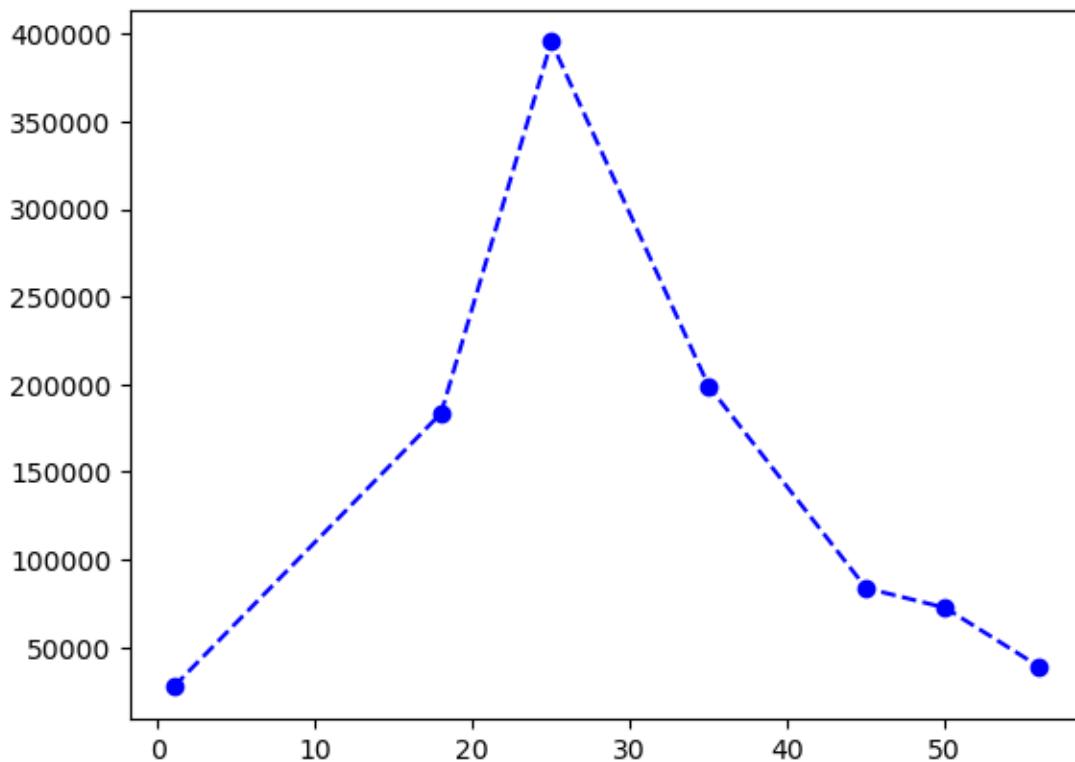
plt.plot(Master_df.groupby("Age") ["MovieID"].count(), '--bo')
Master_df.groupby("Age") ["MovieID"].count()

```

Out[11]:

Age	Count
1	27211
18	183536
25	395556
35	199003
45	83633
50	72490

```
56      38780  
Name: MovieID, dtype: int64
```



```
Master_df.groupby("Age") ["MovieID"].count().describe()
```

In [12]:

```
count      7.000000  
mean     142887.000000  
std      129954.446357  
min      27211.000000  
25%      55635.000000  
50%      83633.000000  
75%      191269.500000  
max      395556.000000  
Name: MovieID, dtype: float64
```

Out[12]:

- Explore the datasets using visual representations (graphs or tables), also include your comments on the following:
 - o These graphs are made on the Master database that is just created

1. Top 25 movies by viewership rating

```
rating_avg = Master_df.groupby('Title')['Rating'].mean()  
rating_avg.head()  
rating_avg = rating_avg.sort_values(ascending=False)  
rating_avg[0:25]
```

In [13]:

```
Title
```

Out[13]:

Ulysses (Ulisse) (1954)	5.0000
00	
Lured (1947)	5.0000
00	
Follow the Bitch (1998)	5.0000
00	
Bittersweet Motel (2000)	5.0000
00	
Song of Freedom (1936)	5.0000
00	
One Little Indian (1973)	5.0000
00	
Smashing Time (1967)	5.0000
00	
Schlafes Bruder (Brother of Sleep) (1995)	5.0000
00	
Gate of Heavenly Peace, The (1995)	5.0000
00	
Baby, The (1973)	5.0000
00	
I Am Cuba (Soy Cuba/Ya Kuba) (1964)	4.8000
00	
Lamerica (1994)	4.7500
00	
Apple, The (Sib) (1998)	4.6666
67	
Sanjuro (1962)	4.6086
96	
Seven Samurai (The Magnificent Seven) (Shichinin no samurai) (1954)	4.5605
10	
Shawshank Redemption, The (1994)	4.5545
58	
Godfather, The (1972)	4.5249
66	
Close Shave, A (1995)	4.5205
48	
Usual Suspects, The (1995)	4.5171
06	
Schindler's List (1993)	4.5104
17	
Wrong Trousers, The (1993)	4.5079
37	
Dry Cleaning (Nettoyage à sec) (1997)	4.5000
00	
Inheritors, The (Die Siebtelbauern) (1998)	4.5000
00	
Mamma Roma (1962)	4.5000
00	
Bells, The (1926)	4.5000
00	

Name: Rating, dtype: float64

In [14]:

```

rating_avg_count = pd.DataFrame(data=rating_avg)
rating_avg_count['number_of_ratings'] = pd.DataFrame(rating_avg_count)
rating_avg_count.head()

```

Out[14]:

	Rating	number_of_ratings
Title		
Ulysses (Ulisse) (1954)	5.0	5.0
Lured (1947)	5.0	5.0
Follow the Bitch (1998)	5.0	5.0
Bittersweet Motel (2000)	5.0	5.0
Song of Freedom (1936)	5.0	5.0

In [15]:

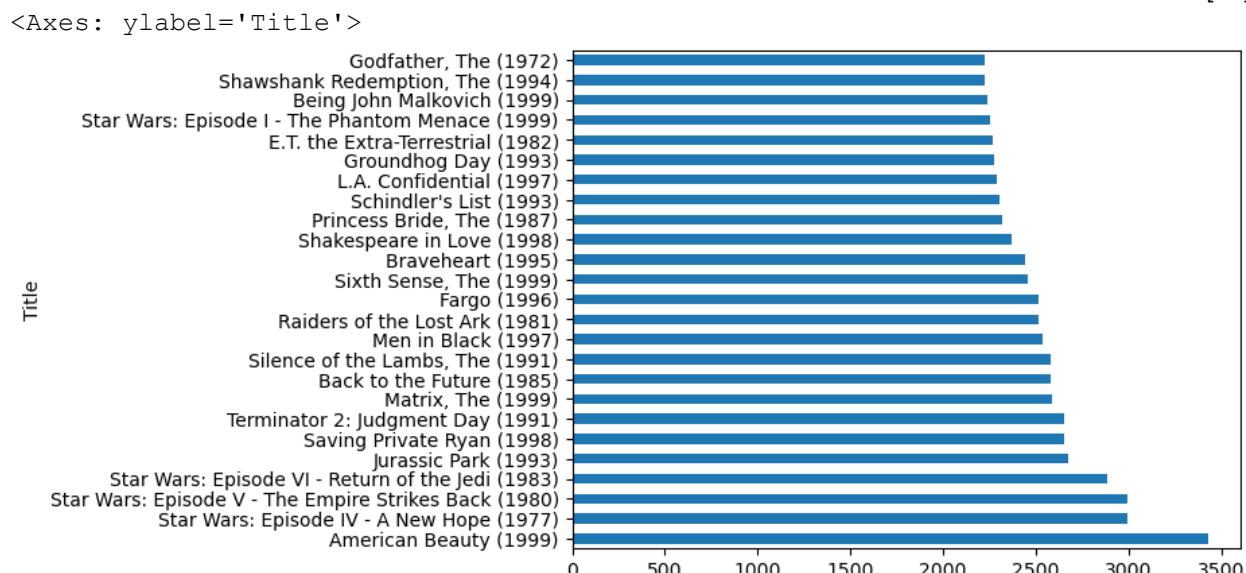
#Top 25 movies by viewership rating

```

res = Master_df.groupby("Title").size().sort_values(ascending=False)[:25]
plt.ylabel("Title")
plt.xlabel("Viewership Count")
res.plot(kind="barh")

```

Out[15]:



- Explore the datasets using visual representations (graphs or tables), also include your comments on the following:
 - o These graphs are made on the Master database that is just created

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

In [16]:

```
user_2696 = Master_df[Master_df['UserID'] == 2696]
user_2696
```

Out[16]:

	MovieID		Title	UserID	Rating	Gender	Age	Occupation	zip-code
991035	350		Client, The (1994)	2696	3	M	25		7 24210
991036	800		Lone Star (1996)	2696	5	M	25		7 24210
991037	1092		Basic Instinct (1992)	2696	4	M	25		7 24210
991038	1097	E.T. the Extra-Terrestrial (1982)		2696	3	M	25		7 24210
991039	1258		Shining, The (1980)	2696	4	M	25		7 24210
991040	1270		Back to the Future (1985)	2696	2	M	25		7 24210
991041	1589		Cop Land (1997)	2696	3	M	25		7 24210
991042	1617		L.A. Confidential (1997)	2696	4	M	25		7 24210
991043	1625		Game, The (1997)	2696	4	M	25		7 24210
991044	1644	I Know What You Did Last Summer (1997)		2696	2	M	25		7 24210
991045	1645		Devil's Advocate, The (1997)	2696	4	M	25		7 24210
991046	1711	Midnight in the Garden of Good and Evil (1997)		2696	4	M	25		7 24210
991047	1783		Palmetto (1998)	2696	4	M	25		7 24210
991048	1805		Wild Things (1998)	2696	4	M	25		7 24210

	MovieID		Title	UserID	Rating	Gender	Age	Occupation	zip-code
991049	1892		Perfect Murder, A (1998)	2696	4	M	25		7 24210
991050	2338	I Still Know What You Did Last Summer (1998)		2696	2	M	25		7 24210
991051	2389		Psycho (1998)	2696	4	M	25		7 24210
991052	2713		Lake Placid (1999)	2696	1	M	25		7 24210
991053	3176	Talented Mr. Ripley, The (1999)		2696	4	M	25		7 24210
991054	3386		JFK (1991)	2696	1	M	25		7 24210

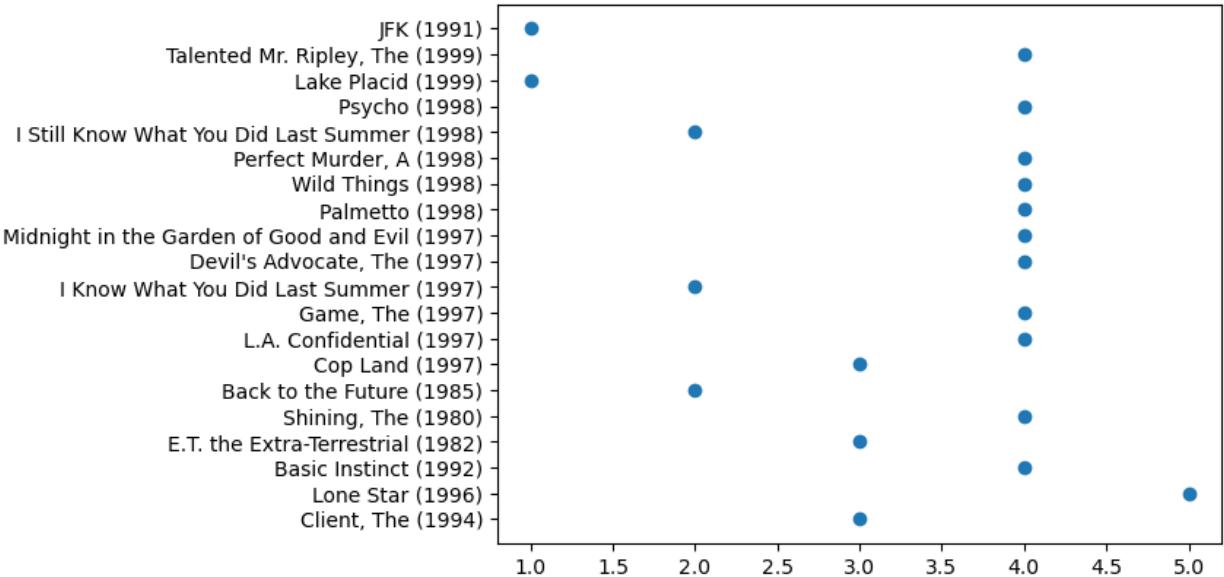
In [17]:

```
res = Master_df[Master_df.UserID == 2696]
plt.scatter(y=res.Title, x=res.Rating)
res
```

Out[17]:

	MovieID		Title	UserID	Rating	Gender	Age	Occupation	zip-code
991035	350		Client, The (1994)	2696	3	M	25		7 24210
991036	800		Lone Star (1996)	2696	5	M	25		7 24210
991037	1092		Basic Instinct (1992)	2696	4	M	25		7 24210
991038	1097	E.T. the Extra-Terrestrial (1982)		2696	3	M	25		7 24210
991039	1258		Shining, The (1980)	2696	4	M	25		7 24210
991040	1270		Back to the Future (1985)	2696	2	M	25		7 24210
991041	1589		Cop Land (1997)	2696	3	M	25		7 24210

	MovieID		Title	UserID	Rating	Gender	Age	Occupation	zip-code
991042	1617	L.A. Confidential (1997)	2696	4	M	25	7	24210	
991043	1625	Game, The (1997)	2696	4	M	25	7	24210	
991044	1644	I Know What You Did Last Summer (1997)	2696	2	M	25	7	24210	
991045	1645	Devil's Advocate, The (1997)	2696	4	M	25	7	24210	
991046	1711	Midnight in the Garden of Good and Evil (1997)	2696	4	M	25	7	24210	
991047	1783	Palmetto (1998)	2696	4	M	25	7	24210	
991048	1805	Wild Things (1998)	2696	4	M	25	7	24210	
991049	1892	Perfect Murder, A (1998)	2696	4	M	25	7	24210	
991050	2338	I Still Know What You Did Last Summer (1998)	2696	2	M	25	7	24210	
991051	2389	Psycho (1998)	2696	4	M	25	7	24210	
991052	2713	Lake Placid (1999)	2696	1	M	25	7	24210	
991053	3176	Talented Mr. Ripley, The (1999)	2696	4	M	25	7	24210	
991054	3386	JFK (1991)	2696	1	M	25	7	24210	



- Feature Engineering: Use column genres: Find out all the unique genres

In [18]:

```
movies_df.head()
```

Out[18]:

	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 [19]:

```
movies_df['Genres'].value_counts().head()
```

Out[19]:

Drama	843
Comedy	521
Horror	178
Comedy Drama	162
Comedy Romance	142
Name: Genres, dtype: int64	

In [20]:

```
movies_df['Genres'].unique()
```

Out[20]:

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

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

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

In [21]:

#However these are not unique in terms of genre. It is showing the combines unique genre

In [22]:

```
Master_df.head()
```

Out[22]:

MovieID		Title	UserID	Rating	Gender	Age	Occupation	zip-code
0	1	Toy Story (1995)	1	5	F	1	10	48067
1	48	Pocahontas (1995)	1	5	F	1	10	48067
2	150	Apollo 13 (1995)	1	5	F	1	10	48067
3	260	Star Wars: Episode IV - A New Hope (1977)	1	4	F	1	10	48067
4	527	Schindler's List (1993)	1	5	F	1	10	48067

In [23]:

```
#Joining the dataset
result = pd.merge(Master_df, movies_df, how="left", on='MovieID')
result.head()
result.columns
result = result[['MovieID', 'Title_x', 'UserID', 'Rating', 'Gender', 'Age',
'Occupation', 'Genres']]
result.head()
result.rename(columns = {'Title_x':'Title'}, inplace = True)
result.head()
```

Out[23]:

MovieID		Title	UserID	Rating	Gender	Age	Occupation	Genres
0	1	Toy Story (1995)	1	5	F	1	10	Animation Children's Comedy
1	48	Pocahontas (1995)	1	5	F	1	10	Animation Children's Musical Romance
2	150	Apollo 13 (1995)	1	5	F	1	10	Drama
3	260	Star Wars: Episode IV - A New Hope (1977)	1	4	F	1	10	Action Adventure Fantasy Sci-Fi

MovielD	Title	UserID	Rating	Gender	Age	Occupation	Genres
4	527 Schindler's List (1993)	1	5	F	1	10	Drama War

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.

Creating dummies for each possible genre, such as Sci-Fi or Drama, and having a single column for each. Creating dummies means creating 0s and 1s

In [24]:

```
dummies = result['Genres'].str.get_dummies()
dummies.head()
```

Out[24]:

	Action	Adventure	Animation	Children's	Comedy	Crim	Documentary	Drama	Fantasy	Film - Noir	Horr	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	0	1	0	0	0	0	0	0	0	1	0

In [25]:

```
tidy_movie_ratings = (pd.concat([result, dummies], axis=1)
                      .drop(['Genres'], axis=1)
                     )
```

tidy_movie_ratings.head()

Out[25]:

5 rows × 25 columns

Now need to add column Year The year is the last 6 from left in Movie Title.

In [26]:

```
tidy_movie_ratings["Year"] = tidy_movie_ratings["Title"].str[-5:-1]
tidy_movie_ratings["movie_title"] = tidy_movie_ratings["Title"].str[:-7]
```

In [27]:

```
tidy_movie_ratings.reset_index(inplace=True)
tidy_movie_ratings[['MovieID', 'Title', 'UserID',
'Rating', 'Gender', 'Age',
'Occupation', 'Action', 'Adventure', 'Animation', "Children's",
'Comedy', 'Crime', 'Documentary', 'Drama', 'Fantasy', 'Film-Noir',
'Horror', 'Musical', 'Mystery', 'Romance', 'Sci-Fi', 'Thriller',
'War',
'Western', 'Year']]
```

In [28]:

```
tidy_movie_ratings.columns
```

Out[28]:

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

In [29]:

```
tidy_movie_ratings.head()
tidy_movie_ratings.shape
```

Out[29]:

```
(1000209, 26)
```

Determine the features affecting the ratings of any particular movie. Removing few parameters like Title, User Id, Movie Id which

In [30]:

```
X_feature = tidy_movie_ratings[['Gender', 'Age', 'Occupation',
                                'Action', 'Adventure', 'Animation', "Children's", 'Comedy', 'Crime',
                                'Documentary', 'Drama', 'Fantasy', 'Film-Noir', 'Horror', 'Musical',
                                'Mystery', 'Romance', 'Sci-Fi', 'Thriller', 'War', 'Western', 'Year']]
X_feature.shape
```

Out[30]:

```
(1000209, 22)
```

In [31]:

```
X_feature.head()
```

Out[31]:

G en de r	A g e	Oc cup atio n	A ct io n	Ad ven tur e	An im ati on	Ch ild ren 's	C o m ed y	C ri m e	Doc ume ntar y	F il m	H or ro r	M us ic al	M ys te ry	R o m an ce	S c i - F i	T hr ill er	W a ill er	W est er n	Y e a r	
									

0 F 1 10 0 0 1 1 1 0 0 . 0 0 0 0 0 0 0 0 0 0 0 1 9 9 9 5

1 F 1 10 0 0 1 1 0 0 0 . 0 0 1 0 1 0 0 0 0 0 0 1 9 9 9 5

2 F 1 10 0 0 0 0 0 0 0 . 0 0 0 0 0 0 0 0 0 0 0 1 9 9 9 5

3 F 1 10 1 1 0 0 0 0 0 . 0 0 0 0 0 0 0 1 0 0 0 1 9 7 7

4 F 1 10 0 0 0 0 0 0 0 . 0 0 0 0 0 0 0 0 1 0 0 1 9 9 3

5 rows × 22 columns

In [32]:

```
Y_target = tidy_movie_ratings[['Rating']]
```

In [33]:

```
#find the correlation using 'corr()' function.  
#it returns a dataframe which contain the correlation between all the numeric  
columns.  
data_corr = tidy_movie_ratings.corr()  
data_corr  
  
C:\Users\aranj\AppData\Local\Temp\ipykernel_17684\3636535412.py:3: FutureWarning:  
The default value of numeric_only in DataFrame.corr is deprecated. In a  
future version, it will default to False. Select only valid columns or specif  
y the value of numeric_only to silence this warning.  
data_corr = tidy_movie_ratings.corr()
```

Out[33]:

M o v ie I D	U se rI D	R at in g	A ge	Oc cu pa tio n	A ct io n	A dv en tu re	A ni m ati on	C hil dr en 's	C o m e d y	F . . as y	F i l m - N oi r	H or r	M us ic al	M ys te ry	R o m an ce	S ci - Fi	T h ri ll er	W ar	W es te r n
-----------------------------	--------------------	--------------------	---------	----------------------------	--------------------	---------------------------	---------------------------	----------------------------	----------------------------	---------------------	---------------------------------------	--------------	---------------------	---------------------	-------------------------	--------------------	--------------------------	---------	-------------------------

Act ion	-	-	-	-	1.	-	-	-	-	0.	-	-	-	-	-	0.	0.	0.	0.	
	-	0.	0.	0.	0.0	0	0.	-	-	0.	0.	0.	0.	0.	-	3	2	1	0.	
	-	0	0	0	0.0	0	37	0.	0.	2	1	0	0	1	0	0.	1	0	3	2
	0.	0	4	3	18	0	49	11	14	6	4	0	4	0	5	06	9	2	5	2
	04	2	7	0	34	0	61	02	13	8	5	2	7	4	0	78	1	7	8	2
	20	0	6	9	7	0	94	14	0	9	1	8	3	3	8	30	1	5	7	4
	46	2	3	7		0				2	8	3	2	4		7	6	2	2	2
	3	3	5																	

Ad ven tur e	-	0.	0.	0.	0.	-	-	-	-	0.	-	-	-	-	-	0.	0.	0.	0.	
	-	0	0	0	0.0	3	1.	0.	0.	1	2	0	0	0	0	-	2	0	1	0
	0.	0	3	1	14	4	00	00	09	2	7	1	5	2	4	02	8	3	6	1
	08	0	6	6	30	9	00	47	82	4	0	4	7	2	3	43	1	8	6	1
	24	6	7	7	9	6	00	32	83	9	4	1	2	3	5	89	9	4	4	9
	13	8	1	3		1				6	6	7	5	2	0	0	0	2	7	4
	3	8	0							0	8	6	7	3						

An im ati on	-	0.	0.	-	0.	-	-	-	-	0.	0.	0.	-	0.	-	0.	0.	0.	0.	
	-	0	1	0	0.0	1	0.	1.	0.	1	1	3	0	3	0	0.	0	0	0	
	0.	0	9	4	03	1	00	00	57	8	2	7	4	5	4	05	5	8	4	3
	01	7	6	7	83	0	47	00	62	5	0	0	0	2	45	5	5	6	0	
	41	6	6	0	83	2	32	00	04	4	2	1	7	3	4	40	5	7	1	9
	77	6	7	0	4	9				4	5	3	0	1	8	6	3	4	8	
	5	0	0			4														

Ch ild ren 's	-	0.	0.	0.	-	0.	-	-	-	0.	0.	0.	-	0.	-	0.	0.	0.	0.	
	-	0	0	0	0.0	1	0.	0.	1.	0	2	0	0	1	0	0.	1	0	0	
	0.	0	3	5	06	4	09	57	00	8	3	7	2	5	08	3	3	6	3	
	07	4	9	2	90	1	82	62	00	7	2	8	7	5	2	45	8	2	6	1
	15	8	8	8	90	3	83	04	00	1	8	0	0	6	7	50	8	6	5	2
	89	6	2	5	6	1				1	0	3	9	7	6	4	4	3	6	
	2	9	8			4										4	2	9	9	

Co me dy	-	-	-	-	-	-	-	-	-	1.	-	-	-	0.	-	0.	0.	0.	0.	
	-	0.	0.	0.	0.	2	0.	0.	0.	0	0.	1	0	0	1	0.	2	1	0.	
	0.	0	3	4	06	6	12	01	05	0	0.	0	9	0	11	8	9	2	7	
	06	3	9	4	14	8	49	85	87	0	6	1	3	5	28	7	9	7	9	
	16	6	6	0	9	9	60	44	11	0	0	4	0	5	3	43	0	5	1	2
	67	5	2	4	9	9				0	1	2	6	6	4	7	0	0	1	7
	1	2	6			2					0	0	5	4	6	6	9	1	1	1

M	U	R	O	A	C	A	C	F	I	H	M	M	R	S	T	W	
o	s	e	c	a	u	d	h	a	m	o	u	y	o	h	h	e	
v	r	rI	cu	ct	en	nv	il	f	l	or	us	ys	om	ci	hi	W	
D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	
Cri me	0.	0.	0.	-	0.	-	-	-	0.	-	0.	-	-	-	0.	0.	-
	-	0	0	0	0.0	0	-	-	0	0	1	0	0	-	0	1	0
	0.	0	3	0	02	8	0.	0.	0	3	3	4	6	8	0	1	4
	06	3	3	7	82	8	04	06	08	6	7	7	1	0	07	3	2
	18	4	4	9	1	5	59	25	19	8	3	2	8	1	33	7	7
	96	6	4	9	1	1	24	20	77	0	7	3	9	9	20	3	1
	9	6	3	1	9	9	0	0	5	4	7	9	9	3	0	5	1
Do cu me nta ry	-	0.	0.	0.	-	0.	-	-	0.	-	0.	-	-	-	0.	0.	-
	0.	0	0	0	-	0	-	-	0	0	0	0	0	-	0	0	0
	0.	0	2	0	0.0	5	0.	0.	0.	4	.	1	1	2	0	03	4
	00	0	8	4	02	03	01	02	0	0	7	2	5	7	8	8	1
	95	1	0	4	68	2	51	89	49	6	3	1	6	1	2	71	6
	44	0	9	0	9	5	09	91	01	9	2	7	7	5	6	5	0
	6	8	7	9	6	09	91	01	9	7	6	5	3	5	5	8	7
Dr am a	0.	0.	0.	-	0.	-	-	-	0.	-	0.	-	-	-	0.	0.	-
	-	0	1	0	-	2	-	-	0	0	0	1	0	0	0.	2	0
	0.	0	2	6	0.0	2	0.	0.	0.	2	.	9	6	8	2	1	3
	03	6	2	3	12	0	19	15	13	4	.	6	7	9	4	7	4
	08	5	5	8	32	2	45	44	57	9	.	6	7	9	7	35	5
	56	7	6	5	6	4	70	79	07	8	.	9	2	5	7	6	5
	2	1	6	6	1	1	70	79	07	4	.	2	9	5	7	8	4
Fa nta sy	0.	-	-	0.	0.	0.	0.	-	0.	1.	-	-	-	-	0.	0.	-
	-	0	0	0	0.0	0	0.	0.	0	0	0	0	0	0	-	1	0
	0.	0	2	2	01	1	22	01	26	0	.	0	2	5	2	2	2
	01	2	3	4	29	4	70	20	32	6	.	0	6	5	0	9	4
	87	2	3	2	9	5	46	25	80	0	.	0	4	8	1	48	8
	92	1	1	2	9	5	46	25	80	1	.	0	6	0	3	7	9
	2	2	2	2	1	1	46	25	80	0	.	0	4	3	4	7	9
Fil m- Noi r	0.	0.	0.	-	0.	0.	-	-	0.	0.	1.	-	-	0.	0.	0.	-
	-	0	0	0	0.0	0	-	0.	0	0	0	0	0	2	-	0	0
	0.	0	6	3	05	8	0.	03	0.	0	.	2	0	3	2	1	3
	01	4	0	3	24	0	01	70	03	1	.	6	0	9	8	5	9
	96	7	2	4	6	2	41	13	80	4	.	4	0	1	3	73	6
	55	0	5	9	6	8	78	33	33	2	.	6	0	5	8	5	3
	1	9	5	5	8	8	0	0	5	4	.	7	4	4	6	4	6

M ov ie I D	U se rI D	R at in g	A ge	Oc cu pa tio n	A ct io n	A dv en tu re	A ni m ati on	C hil dr en 's	C o m e d y	F . . nt as y	Fi l m - N oi r	H or ro r	M us ic al	M ys te ry	R o m an ce	S ci - Fi	T h ri ll er	W ar	W es te r n
-------------------------	--------------------	--------------------	---------	----------------------------	--------------------	---------------------------	---------------------------	----------------------------	----------------------------	---------------------------	-----------------------------------	--------------------	---------------------	---------------------	-------------------------	--------------------	--------------------------	---------	-------------------------

-	-	-	-	0.	2	-	-	-	0.	-	0.	0.	-	0.	-	0.	1.	-	-	-
Th rill er	0. 05 84 18	0. 0 1 1	0. 08 4 8	0.0 0 1 1	2 0 2 5	0. 0. 03 23	0. 0. 08 13	0. 0. 57 42	2 9 9 5	0. 0 7 3	1 5 5 3	0. 1 6 2	0. 0 0 6	2 2 5 8	- 1 08 84	1 0 2 4	0 0 0 6	0 0 0 0	0. 0 8 8	0 0 8 9
					6				1	4					0		6	0	8	7
					7	6	0													

-	0. 0 0 0	0. 7 3 5	0. 10 8 26	0. 0 01 5	1 3 0. 04	0. 0. 06 66	0. 0. 06 61	0. 0. 65 65	0. 1 7 1	- 0. 2 7	- 0. 4 4	- 3 6	- 7 7	- 3 4	- 5 5	0. 0 0 47	0. 0 0 4	0. 0 0 47	0. 0 0 1	0. 0 0 8	0. 0 0 9
Wa r	0. 08 19 51	0. 3 5 0	0. 10 8 4	0. 0 01 4	1 3 5 8	0. 0. 04 47	0. 0. 06 14	0. 0. 65 39	0. 1 7 1	- 0. 2 7	- 0. 4 4	- 3 6	- 7 7	- 3 4	- 5 5	0. 0 0 47	0. 0 0 1	0. 0 0 8	0. 0 0 0	0. 0 0 0	0. 0 0 1
					2				1	8	4	5	9	2	47	1	4	8	0	3	
					2				1	8	4	5	9	2	47	1	4	8	0	3	

0. 0 0 0	0. 0 0 0	0. 3 8 1	0. 05 92 4	0. 2 01 2	0. 0. 03 19	0. 0. 03 09	0. 0. 12 64	0. 0. 9 08	0. 2 7 69	- 0. 2 7	- 0. 4 8	- 3 7	- 7 2	- 2 7	- 0. 4	- 5 50	- 3 9	- 5 7	- 3 3	1. 0 0 0
We ste rn	0. 00 39 40	0. 0 4 1	0. 05 92 4	0. 2 01 2	0. 0. 03 19	0. 0. 03 09	0. 0. 12 64	0. 0. 9 08	0. 2 7 69	- 0. 2 7	- 0. 4 8	- 3 7	- 7 2	- 2 7	- 0. 4	- 5 50	- 3 9	- 5 7	- 3 3	1. 0 0 0
					2				1	8	4	5	9	2	47	1	4	8	0	3
					2				1	8	4	5	9	2	47	1	4	8	0	3

23 rows × 23 columns

In [34]:

```
new_cor = data_corr[0:1]
new_cor
```

Out[34]:

M	U	R	A	Oc	A	A	C	C	.	F	I	Fi	H	M	M	R	Sc	T	W	W	
ov	se	at	ge	cu	ct	dv	ni	hil	o	a	m	or	us	ys	o	i-	hr	ill	ar	es	
ie	ri	in	ge	pa	io	en	m	dr	m	.	nt	ro	ic	te	m	Fi	er	er	er	rn	
I	D	g	n	re	on	tu	ati	en	's	y	as	N	al	ry	an	ce	er	er	er	rn	
D												N	oi	r							
M	-	-	0.	0.0	-	-	-	-	0.	-	-	0.	-	-	-	-	-	-	-	0.	
ov	1.	0.	0.	02	08	0.	0.	0.	0.	.	0.	0.	05	0.	0.	0.	0.	0.	0.	0.	
ie	0	01	06	75	58	04	08	01	07	06	.	01	01	05	05	02	11	01	05	08	0
I	77	40	75	58	20	24	41	15	16	.	87	96	76	93	85	83	17	84	19	3	
D	39	42	75	5	46	13	77	89	67	.	92	55	13	81	61	75	47	18	51	9	
																				4	

1 rows × 23 columns

In [35]:

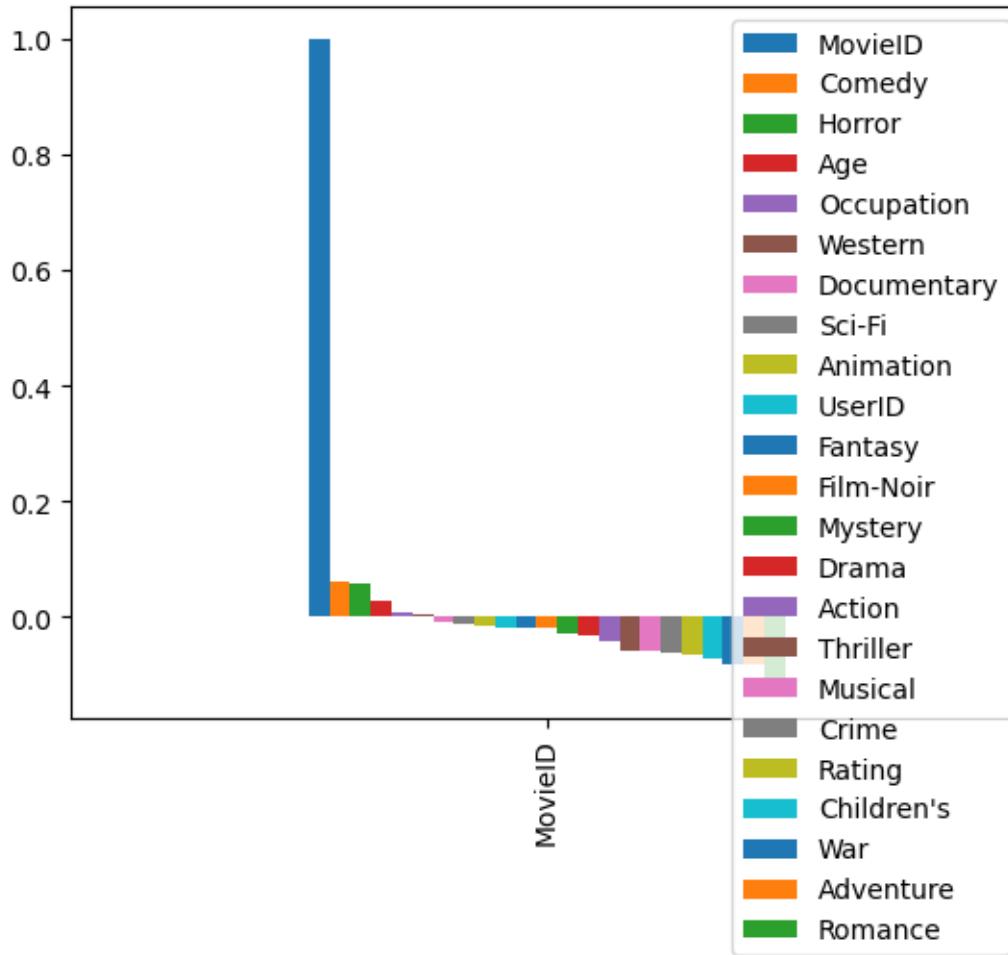
```
rslt_df = data_corr.sort_values(by = 'MovieID', axis = 1, ascending = False)
rslt_df = rslt_df[0:1]
```

In [36]:

```
rslt_df.plot.bar()
```

Out[36]:

<Axes: >



In [37]:

```
#The 3 factors affecting the rating are: Genre, Age & Occupation
#Overall - Rating is affected by 2 Genres - Horror & Comedy
#Genre - (2 of the Sub genre had been highest)
#Followed by Age          Occupation
```

In [38]:

```
from sklearn.metrics import confusion_matrix
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder
#Split the dataset (by default, 75% is the training data and 25% is the testing data)
#by default takin 75%-25% split
#Cross_validation got away from python 2
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(X_feature, Y_target,
random_state=1)
```

In [39]:

```
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"))
C:\ProgramData\anaconda3\Lib\site-packages\sklearn\preprocessing\_label.py:11
6: DataConversionWarning: A column-vector y was passed when a 1d array was ex
pected. Please change the shape of y to (n_samples, ), for example using rave
l().
y = column_or_1d(y, warn=True)
C:\ProgramData\anaconda3\Lib\site-packages\sklearn\preprocessing\_label.py:11
6: DataConversionWarning: A column-vector y was passed when a 1d array was ex
pected. Please change the shape of y to (n_samples, ), for example using rave
l().
y = column_or_1d(y, warn=True)
```

In [40]:

```
#Verify if the training and testing datasets are split correctly
print(x_train.shape)
print(y_train.shape)
print(x_test.shape)
print(y_test.shape)

(750156, 22)
(750156,)
(250053, 22)
(250053,)
```

Logistic regression is best used for predicting categorical data

need to do logistic regression on the training data so we can see how well our test data does the prediction

The dataset kept throwing off a non-convergence error where max iterations had been reached. I used the code below to increase the max iter.

However, for test purpose showing how linear regression model will fail

In [41]:

```
#Create a linear regression model
from sklearn.linear_model import LinearRegression
linreg = LinearRegression()
linreg.fit(x_train, y_train)
LinearRegression(copy_X=True, fit_intercept=True, n_jobs=None,
                 normalize=False)
y_pred = linreg.predict(x_test)

-----
TypeError                                         Traceback (most recent call last)
Cell In[41], line 5
      3 linreg = LinearRegression()
      4 linreg.fit(x_train, y_train)
----> 5 LinearRegression(copy_X=True, fit_intercept=True, n_jobs=None,
      6                 normalize=False)
      7 y_pred = linreg.predict(x_test)

TypeError: LinearRegression.__init__() got an unexpected keyword argument 'no
rmalize'
```

In []:

```
print()
```

```

        'y-intercept: ',
        linreg.intercept_
    )
print(
    'Beta coefficients: ',
    linreg.coef_
)

```

In []:

```

from sklearn import metrics

print(
    'Mean Abs Error MAE: ',
    metrics.mean_absolute_error(y_test, y_pred)
)
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)
)

```

As we see the linear model is not the right model for this dataset Develop an appropriate model to predict the movie ratings

In [42]:

```

# Create a logistic regression model using the training set
from sklearn.linear_model import LogisticRegression
logreg = LogisticRegression()
logreg.fit(x_train, y_train)

C:\ProgramData\anaconda3\Lib\site-packages\sklearn\linear_model\_logistic.py:
458: 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#logistic-regression
    n_iter_i = _check_optimize_result(

```

Out[42]:

LogisticRegression
LogisticRegression()

In [43]:

```
#Make predictions using the testing set
y_pred = logreg.predict(x_test)
```

```

y_pred                                         Out[43]:
array([3, 3, 3, ..., 3, 3, 3], dtype=int64)      In [44]:
#Evaluate the accuracy of your model
print(logreg.intercept_)
print(logreg.coef_)
from sklearn import metrics
import numpy as np
print(metrics.mean_squared_error(y_test, y_pred))
from sklearn import metrics
metrics.accuracy_score(y_test,y_pred)

[-3.72911239e-05 -5.65842971e-05 -5.45592491e-05  2.62081779e-05
 1.22226492e-04]
[[ 1.23526965e-04 -1.68736754e-02 -2.44701381e-03  6.74231056e-04
  2.97300227e-04 -1.42370546e-04  5.51223708e-04  4.50621745e-04
 -4.11938206e-04 -5.46368622e-05 -2.51119580e-03  9.37765430e-05
 -2.42825115e-04  1.49548953e-03 -9.67548776e-05 -1.70485099e-04
 -6.36031008e-04  7.20702191e-04 -1.87337315e-04 -4.60275660e-04
 -2.07144237e-05 -2.93375828e-04]
[ 6.36329084e-04 -2.91680233e-03  1.42350211e-04  1.16542314e-03
 6.95777556e-04 -3.95473120e-04  2.22067412e-04  7.89246682e-04
 -3.71832462e-04 -1.23840425e-04 -2.82996117e-03  2.39911608e-04
 -3.59845550e-04  1.17688745e-03 -2.30021381e-04 -8.79737592e-05
 -1.73466791e-04  1.08272938e-03  2.45195957e-04 -8.14791688e-04
 -8.20160591e-05 -1.72433042e-04]
[ 4.59463515e-04  4.91744717e-03  3.74223100e-04  1.06257850e-03
 9.49359364e-04 -8.53344156e-05  5.05300083e-04  1.23232252e-03
 -4.27855569e-04 -2.10884523e-04 -2.59616289e-03  3.17553939e-04
 -5.19742401e-04  4.86357437e-04 -8.97403255e-05 -9.48512978e-05
 6.29113799e-04  5.61114125e-04  2.17630577e-04 -1.42452022e-03
 -1.27479455e-05  1.58110168e-04]
[-3.95812618e-04  7.10438907e-03  1.79523201e-03 -1.01389841e-03
 -7.39245077e-04  2.67954810e-04 -3.99237138e-04 -1.80630693e-04
 2.38490624e-04  9.16185442e-05  2.48542160e-03 -2.64590354e-04
 1.65551570e-04 -1.48980050e-03  4.45756894e-05  5.87413237e-05
 5.91545527e-04 -1.36478149e-03  1.78227034e-04  6.95661035e-05
 3.41536972e-05  2.63647095e-04]
[-8.23506947e-04  7.76864152e-03  1.35208493e-04 -1.88833429e-03
 -1.20319207e-03  3.55223271e-04 -8.79354065e-04 -2.29156025e-03
 9.73135614e-04  2.97743266e-04  5.45189826e-03 -3.86651736e-04
 9.56861496e-04 -1.66893391e-03  3.71940895e-04  2.94568832e-04
 -4.11161527e-04 -9.99764209e-04 -4.53716252e-04  2.63002146e-03
 8.13247311e-05  4.40516073e-05]]
1.425681755467841                                         Out[44]:
0.3506976520977553                                         In [45]:
#Print the first 30 actual and predicted responses
print('Actual : ', y_test[0:30])
print('Predicted : ', y_pred[0:30])

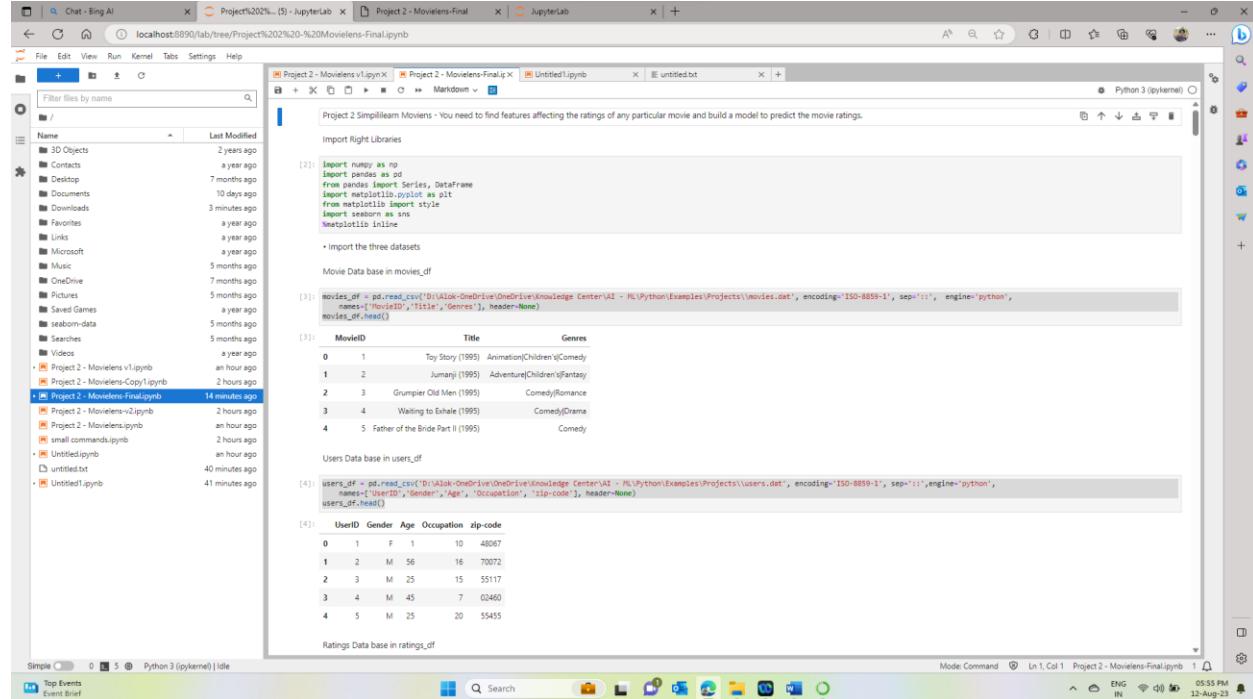
```

```
Actual : [3 4 2 4 3 2 2 2 3 1 4 0 3 4 2 3 3 3 2 4 4 0 4 2 3 4 3 0 3 4 2]
Predicted : [3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3]
```

In []:

ScreenShots:

Import 3 datasets



The screenshot shows a Jupyter Notebook interface with three tabs: Chat - Bing AI, Project%20%20... (5) - JupyterLab, and Project 2 - MovieLens-Final.ipynb. The code cell contains imports for numpy, pandas, and matplotlib, followed by reading the 'movies.dat' file into a DataFrame named movies_df.

```
[2]: import numpy as np
import pandas as pd
from pandas import Series, DataFrame
import matplotlib.pyplot as plt
%matplotlib inline
%pylab inline
import seaborn as sns
sns.set(style="white")
sns.set(style="whitegrid")

# Import the three datasets

Movie Data base in movies_df

[3]: movies_df = pd.read_csv('D:\Alok\OneDrive\OneDrive\Knowledge Center\AI - ML\Python\Examples\Projects\movies.dat', encoding='ISO-8859-1', sep='::', engine='python',
                           names=['MovieID','Title','Genres'], header=None)
movies_df.head()
```

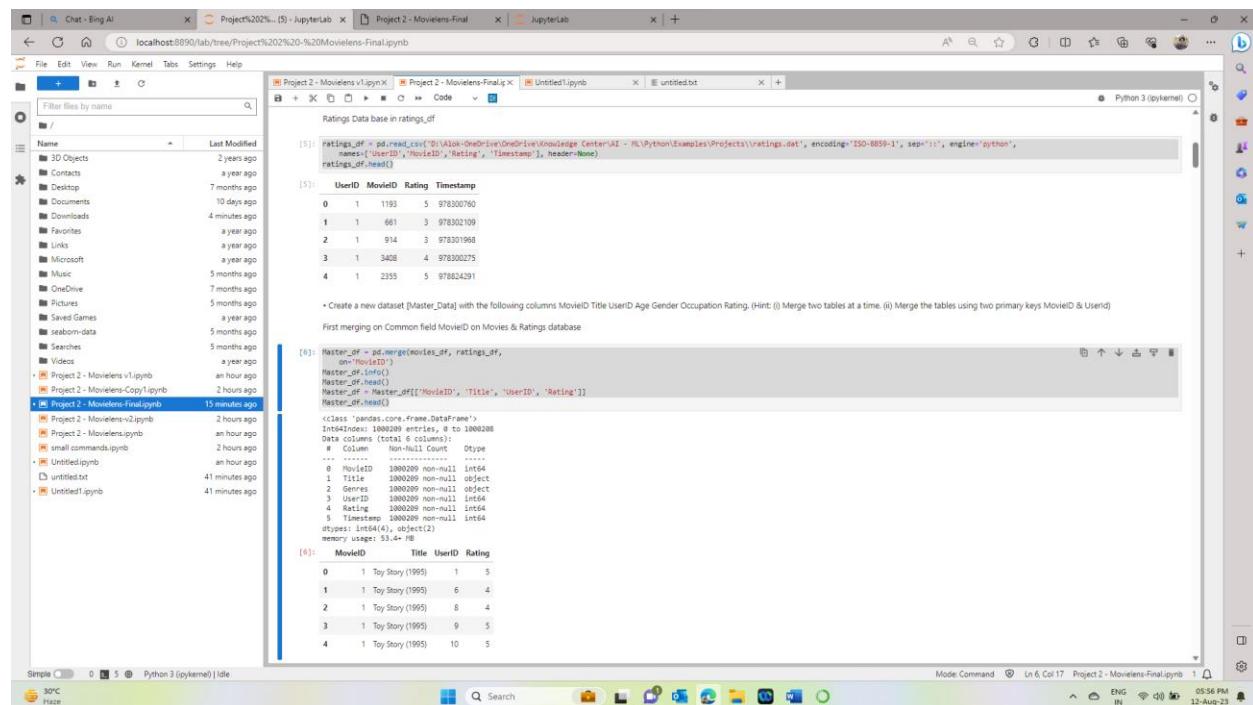
MovieID	Title	Genres
0	1	Toy Story (1995)
1	2	Jumanji (1995)
2	3	Grimspoor Old Men (1995)
3	4	Waiting to Exhale (1995)
4	5	Father of the Bride Part II (1995)

Users Data base in users_df

```
[4]: users_df = pd.read_csv('D:\Alok\OneDrive\OneDrive\Knowledge Center\AI - ML\Python\Examples\Projects\users.dat', encoding='ISO-8859-1', sep='::', engine='python',
                           names=['UserID','Gender','Age','Occupation','zip-code'], header=None)
users_df.head()
```

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

Ratings Data base in ratings_df



The screenshot continues the Jupyter Notebook session. A new ratings_df DataFrame is created from 'ratings.dat'. The code then attempts to merge the movies_df and users_df DataFrames onto the ratings_df based on their primary keys (MovieID and UserID).

```
[5]: ratings_df = pd.read_csv('D:\Alok\OneDrive\OneDrive\Knowledge Center\AI - ML\Python\Examples\Projects\ratings.dat', encoding='ISO-8859-1', sep='::', engine='python',
                           names=['UserID','MovieID','Rating','Timestamp'], header=None)
ratings_df.head()
```

Create a new dataset [Master_Data] with the following columns MovieID Title UserID Age Gender Occupation Rating. (Hint: (i) Merge two tables at a time. (ii) Merge the tables using two primary keys MovieID & UserID)

```
[6]: Master_df = pd.merge(movies_df, ratings_df, on='MovieID')
Master_df.head()
Master_df.info()
Master_df = pd.merge(Master_df, users_df, on='UserID')
Master_df.head()
```

```
[7]: <class 'pandas.core.frame.DataFrame'>
Int64Index: 1000299 entries, 0 to 1000298
Data columns (total 6 columns):
 #   Column          Dtype  
 --- 
 0   MovieID        int64  
 1   Title          non-null object
 2   Genres         non-null object
 3   UserID          int64  
 4   Rating         int64  
 5   Timestamp      int64  
dtypes: int64(4), object(2)
memory usage: 53.4+ KB
```

MovieID	Title	UserID	Rating
0	1 Toy Story (1995)	1	5
1	1 Toy Story (1995)	6	4
2	1 Toy Story (1995)	8	4
3	1 Toy Story (1995)	9	5
4	1 Toy Story (1995)	10	5

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

Screenshot of JupyterLab showing the merging of datasets:

```
Project 2 - MovieLens v1.ipynb X Project 2 - MovieLens-Final.ipynb X Untitled1.ipynb X | + Python 3 (ipykernel) ○
```

Now merging on Common field MovieID on Master database on UserID

```
[7]: Master_df = pd.merge(Master_df, users_df, on='UserID')
Master_df.info()
Master_df.head(10)
```

<class 'pandas.core.frame.DataFrame'>
Int64Index: 1000209 entries: 0 to 1000208
Data columns (total 8 columns):
 # Column Non-Null Count Dtype

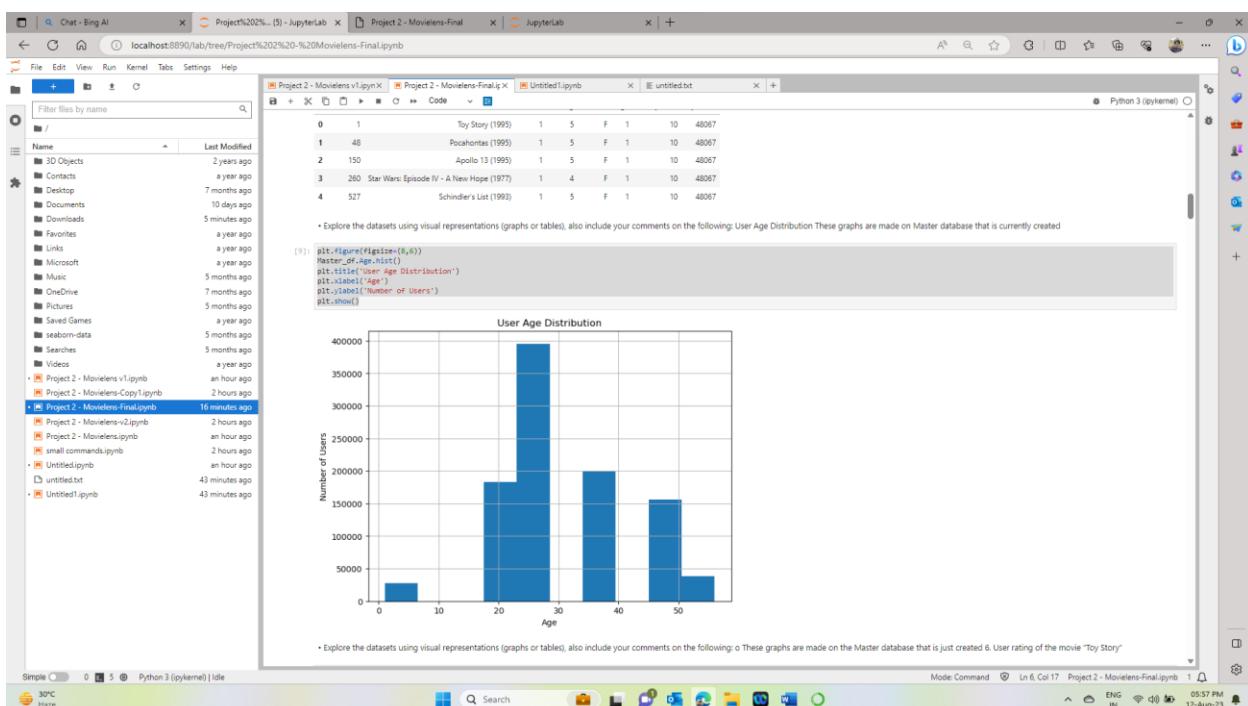
 0 MovieID 1000209 non-null int64
 1 Title 1000209 non-null object
 2 UserID 1000209 non-null int64
 3 Rating 1000209 non-null int64
 4 Gender 1000209 non-null object
 5 Age 1000209 non-null int64
 6 Occupation 1000209 non-null int64
 7 zip-code 1000209 non-null object
 dtypes: int64(5), object(3)
 memory usage: 88.7+ MB

```
[7]: MovieID      Title UserID Rating Gender Age Occupation zip-code
0       1   Toy Story (1995)    1     5      F   1     10    48067
1      48  Pocahontas (1995)    1     5      F   1     10    48067
2     150    Apollo 13 (1995)    1     5      F   1     10    48067
3    260  Star Wars: Episode IV - A New Hope (1977)    1     4      F   1     10    48067
4     527  Schindler's List (1993)    1     5      F   1     10    48067
```

Rearranged the Master database as per the requirement

```
[8]: # Get the DataFrame column names as a list
list(Master_df.columns)
Master_df.reindex(columns=['MovieID',
                          'Title',
                          'UserID',
                          'Rating',
                          'Gender',
                          'Age',
                          'Occupation',
                          'zip-code'])
Master_df.head(5)
```

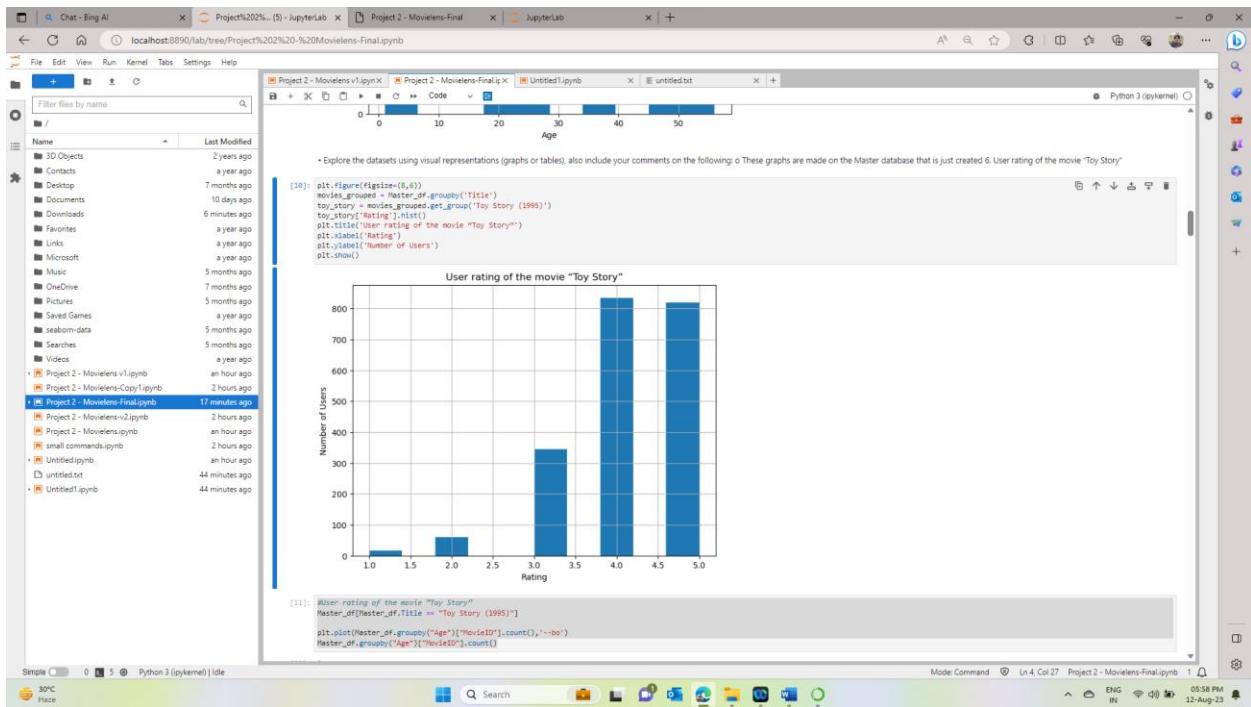
```
[8]: MovieID      Title UserID Rating Gender Age Occupation zip-code
0       1   Toy Story (1995)    1     5      F   1     10    48067
1      48  Pocahontas (1995)    1     5      F   1     10    48067
2     150    Apollo 13 (1995)    1     5      F   1     10    48067
3    260  Star Wars: Episode IV - A New Hope (1977)    1     4      F   1     10    48067
4     527  Schindler's List (1993)    1     5      F   1     10    48067
```



- Explore the datasets using visual representations (graphs or tables), also include your comments on the following:

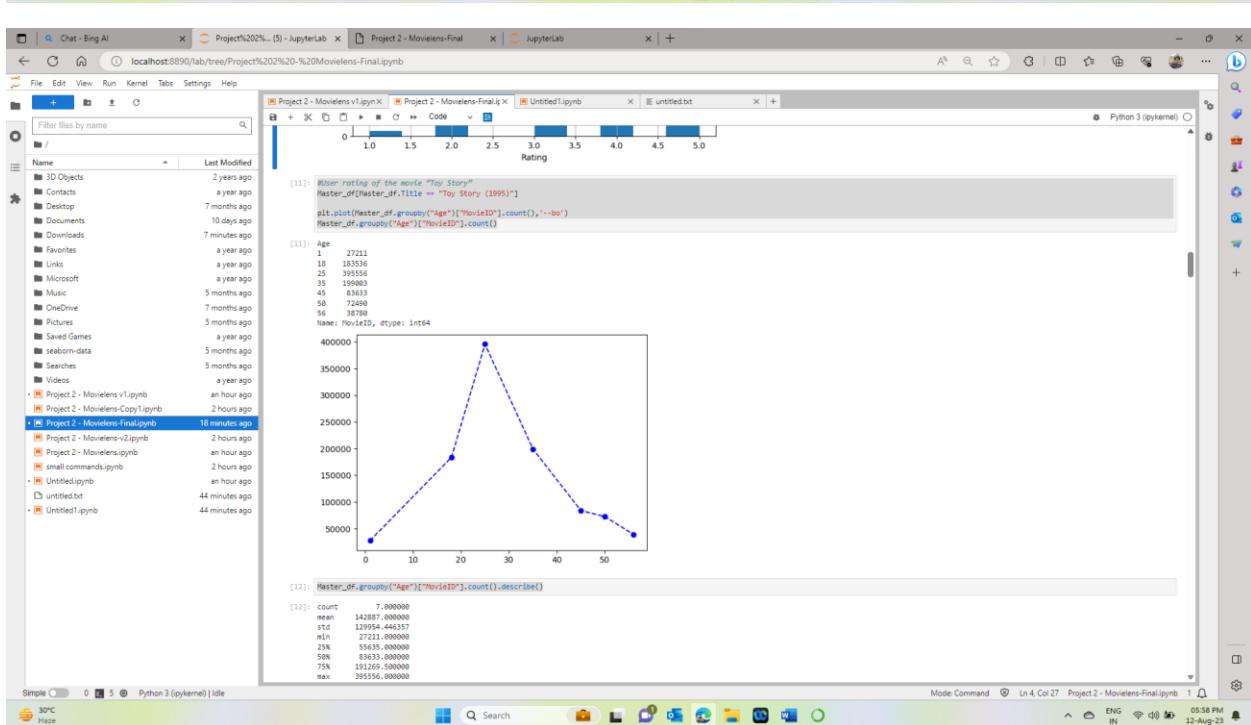
9. User Age Distribution
10. User rating of the movie "Toy Story"

11. Top 25 movies by viewership rating
 12. Find the ratings for all the movies reviewed by for a particular user of user id = 2696



```
[10]: plt.figure(figsize=(8,4))
movies_grouped = Master_df.groupby('Title')
toy_story = movies_grouped.get_group('Toy Story (1995)')
toy_story['User Rating'].hist()
plt.title("User rating of the movie \"Toy Story\"")
plt.xlabel("Rating")
plt.ylabel("Number of Users")
plt.show()
```

User rating of the movie "Toy Story"



```
[11]: #User rating of the movie "Toy Story"
Master_df[Master_df.Title == "Toy Story (1995)"]
plt.plot(Master_df.groupby('Age')[['MovieID']].count(),'bo')
Master_df.groupby('Age')[['MovieID']].count()
```

User rating of the movie "Toy Story"

Age	Count
1	27211
18	183536
25	395556
35	186000
45	93633
50	72489
54	47876

```
[12]: Master_df.groupby("Age")["MovieID"].count().describe()
```

count	mean	std	min	25%	50%	75%	max
7,000000	142887.000000	129594.446357	2711	55631.000000	83633.000000	191269.500000	395556.000000

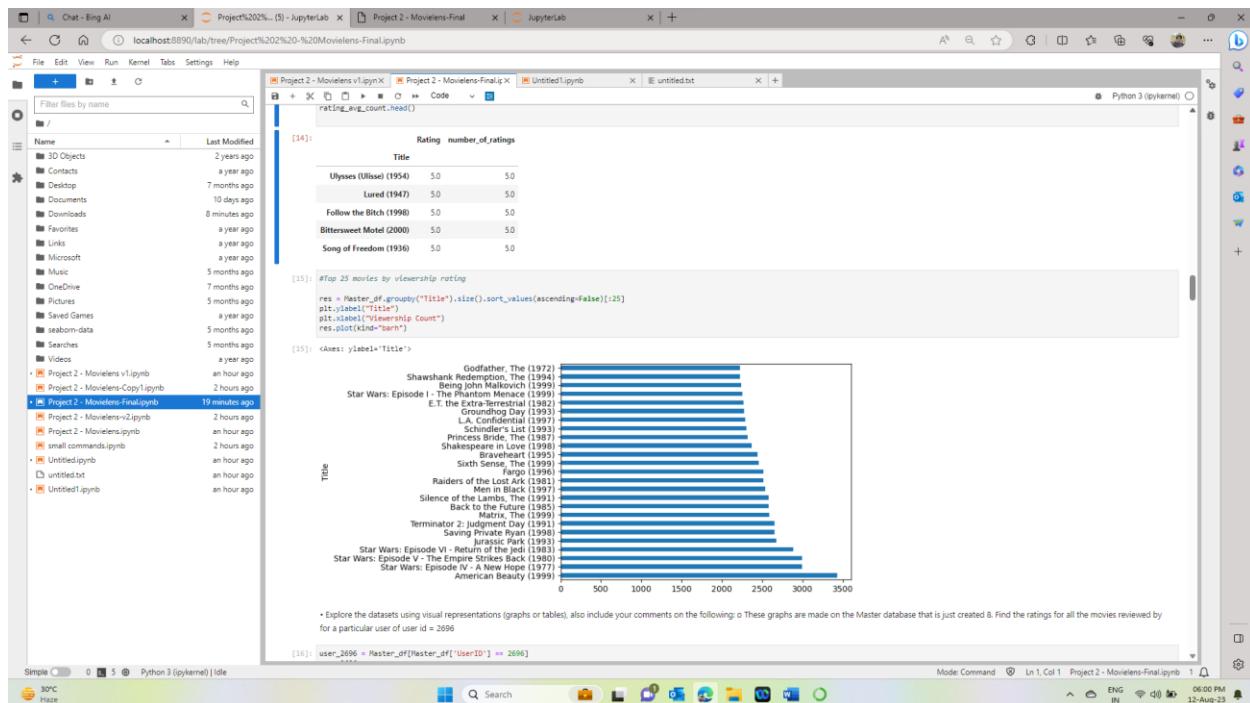
Chat - Bing AI Project%202... (5) - JupyterLab Project 2 - MovieLens-Final.ipynb JupyterLab

```
[12]: Master_df.groupby("Age")["MovieID"].count().describe()
[12]: count    73800000
[12]: mean    142887.000000
[12]: std     129954.446357
[12]: min      2711.000000
[12]: 25%    5730.000000
[12]: 50%    8363.000000
[12]: 75%   151269.500000
[12]: max   395586.000000
[12]: Name: MovieID, dtype: float64
• Explore the datasets using visual representations (graphs or tables), also include your comments on the following: o These graphs are made on the Master database that is just created 7. Top 25 movies by viewership rating
```

```
[13]: rating_avg = Master_df.groupby("Title")["Rating"].mean()
rating_avg = rating_avg.sort_values(ascending=False)
rating_avg.head()
```

Title	Rating
Ulysses (Ulisse) (1954)	5.000000
Lured (1947)	5.000000
Follow the Bitch (1998)	5.000000
Bittersweet Motel (2000)	5.000000
Song of Freedom (1936)	5.000000
One Little Indian (1973)	5.000000
Seaborn-Daten (1998)	5.000000
Schlafer Bruder (Brother of Sleep) (1995)	5.000000
Gate of Heavenly Peace, The (1995)	5.000000
Baby, The (1977)	5.000000
1 Love (1998)	5.000000
Lamerica (1994)	4.750000
Apple, The (SIS) (1998)	4.666667
Sandok (1998)	4.650000
Seven Samurais (The Magnificent Seven) (Shichinin no samurai) (1954)	4.560510
Shawshank Redemption, The (1994)	4.554558
Courtship, The (1998)	4.539548
Close Shave, The (1995)	4.517106
Usual Suspects, The (1995)	4.518417
Schindler's List (1993)	4.518417
House of Cards (1993)	4.507371
Dry Cleaning (Nettoyage à sec) (1997)	4.500000
Inheritors, The (Die Siedebauern) (1998)	4.500000
Name: Rating, dtype: float64	4.500000
Bells, The (1998)	4.500000

```
[14]: rating_avg_count = pd.DataFrame(data=rating_avg)
rating_avg_count['number_of_ratings'] = pd.DataFrame(rating_avg_count)
rating_avg_count.head()
```



Chat - Bing AI Project%20%... (5) - JupyterLab Project 2 - MovieLens-Final.ipynb JupyterLab

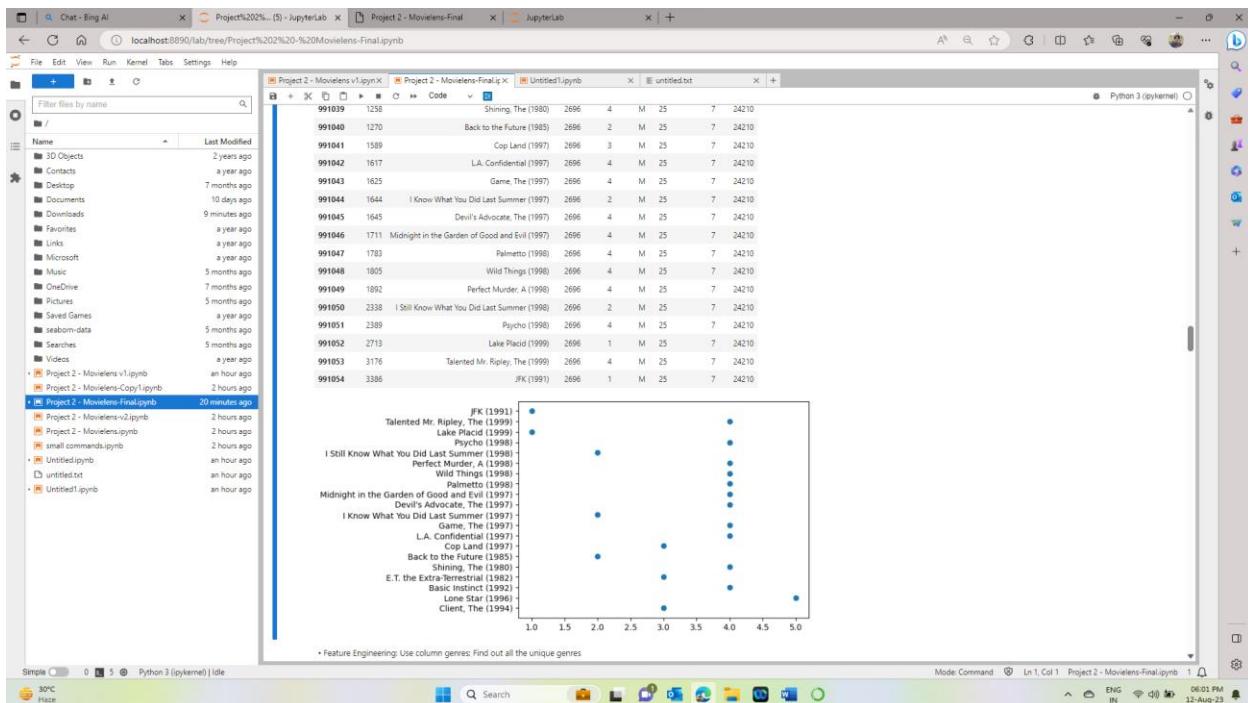
localhost:8890/lab/tree/Project%20%20MovieLens-Final.ipynb

```
[16]: user_2696 = Master_df[Master_df['UserID'] == 2696]
user_2696
```

MovieID	Title	UserID	Rating	Gender	Age	Occupation	zip-code	
991035	350	Client, The (1994)	2696	3	M	25	7	24210
991036	800	Lone Star (1996)	2696	5	M	25	7	24210
991037	1092	Basic Instinct (1992)	2696	4	M	25	7	24210
991038	1097	E.T. the Extra-Terrestrial (1982)	2696	3	M	25	7	24210
991039	1258	Shining, The (1980)	2696	4	M	25	7	24210
991040	1270	Back to the Future (1985)	2696	2	M	25	7	24210
991041	1569	Cop Land (1997)	2696	3	M	25	7	24210
991042	1617	L.A. Confidential (1997)	2696	4	M	25	7	24210
991043	1625	Game, The (1997)	2696	4	M	25	7	24210
991044	1644	I Know What You Did Last Summer (1997)	2696	2	M	25	7	24210
991045	1645	Devil's Advocate, The (1997)	2696	4	M	25	7	24210
991046	1711	Midnight in the Garden of Good and Evil (1997)	2696	4	M	25	7	24210
991047	1783	Pelmetto (1998)	2696	4	M	25	7	24210
991048	1805	Wild Things (1998)	2696	4	M	25	7	24210
991049	1892	Perfect Murder, A (1998)	2696	4	M	25	7	24210
991050	2338	I Still Know What You Did Last Summer (1998)	2696	2	M	25	7	24210
991051	2349	Psycho (1960)	2696	4	M	25	7	24210
991052	2713	Lake Placid (1999)	2696	1	M	25	7	24210
991053	3176	Talented Mr. Ripley, The (1999)	2696	4	M	25	7	24210
991054	3386	JFK (1991)	2696	1	M	25	7	24210

```
[17]: res = Master_df[Master_df.UserID == 2696]
plt.scatter(y=res.Title, x=res.Rating)
res
```

MovieID	Title	UserID	Rating	Gender	Age	Occupation	zip-code	
991035	350	Client, The (1994)	2696	3	M	25	7	24210
991036	800	Lone Star (1996)	2696	5	M	25	7	24210
991037	1092	Basic Instinct (1992)	2696	4	M	25	7	24210
991038	1097	E.T. the Extra-Terrestrial (1982)	2696	3	M	25	7	24210



- Feature Engineering:
Use column genres:

The screenshot shows a Jupyter Notebook interface with multiple tabs open. The active tab is 'Project 2 - MovieLens-Final.ipynb'. The code cell contains several lines of Python code related to movie data analysis, including reading CSV files, creating dataframes, and performing genre counts.

```
[18]: movies_df.head()
[19]: movies_df['Genres'].value_counts().head()
[20]: movies_df['Genres'].unique()
[21]: array(['Animation|Children\'s|Comedy', 'Adventure|Children\'s|Fantasy',
   'Comedy|Romance', 'Comedy|Drama', 'Comedy',
   'Action|Crime|Thriller', 'Adventure|Children\'s', 'Action',
   'Action|Thriller', 'Action|Romance', 'Action|Science Fiction',
   'Comedy|Horror', 'Animation|Children\'s', 'Drama',
   'Action|Adventure|Romance', 'Drama|Thriller', 'Drama|Romance',
   'Thriller', 'Action|Comedy|Drama', 'Drama|Romantic Thriller',
   'Drama|Sci-Fi', 'Romantic|Drama', 'Drama|Romantic Thriller',
   'Children\'s|Comedy', 'Documentary', 'Drama|War',
   'Action|Crime|Drama', 'Action|Adventure', 'Crime|Thriller',
   'Mystery|Drama', 'Action|Romantic Thriller', 'Drama|Thriller',
   'Children\'s|Comedy', 'Drama|Mystery', 'Sci-Fi|Thriller',
   'Action|Comedy|Horror|Thriller', 'Drama|Musical',
   'Crime|Drama', 'Action|Romantic|Drama', 'Action|Thriller',
   'Action|Children\'s|Comedy|Romantic', 'Action|Thriller',
   'Action|Adventure|Crime', 'Crime', 'Drama|Mystery|Romance',
   'Action|Drama', 'Drama|Romantic|Horror',
   'Action|Romantic|Drama', 'Drama|Thriller',
   'Action|Adventure|Mystery|Sci-Fi', 'Drama|Thriller|War',
   'Action|Romantic|Thriller', 'Crime|Film-Noir|Mystery|Thriller',
   'Action|Romantic|Drama', 'Drama|Romantic|Thriller',
   'Action|Sci-Fi|Thriller', 'Action|Adventure|Sci-Fi',
   'Action|Children\'s', 'Horror|Sci-Fi', 'Action|Crime|Sci-Fi',
   'Western', 'Action|Romantic|Children\'s|Comedy|Romance',
   'CSci-Fi', 'Action|Romantic|Drama',
   'Drama|Thriller', 'Drama|Romance|Thriller', 'Drama|Horror', 'Comedy|Sci-Fi',
   'Mystery|Thriller', 'Adventure|Children\'s|Comedy|Fantasy|Romance',
   'Action|Adventure|Sci-Fi', 'Drama|Romantic|Horror|Western',
   'Action|Crime', 'Crime|Drama|Romance|Thriller'])
```

The screenshot shows a browser window with several tabs open, all related to a Jupyter Notebook environment. The tabs include 'Project%202020... (5) - JupyterLab', 'Project 2 - MovieLens-Final.ipynb', and 'Untitled1.ipynb'. The main content area displays Python code and its execution results.

Code Execution Results:

```
[21]: #However these are not unique in terms of genre. It is showing the combines unique genre
[22]: Master.head()
[23]: MovieID      Title UserID Rating Gender Age Occupation zip-code
   0       1 Toy Story (1995)    1     5   F   1    10
   1     48 Pocahontas (1995)    1     5   F   1    10  48067
   2    150 Apollo 13 (1995)    1     5   F   1    10  48067
   3   260 Star Wars Episode IV - A New Hope (1977)    1     4   F   1    10  48067
   4   527 Schindler's List (1993)    1     5   F   1    10  48067
```

Code Block 23:

```
[23]: #Joining the dataset
result = pd.merge(Master_df, movies_df, how='left', on='MovieID')
result.head()
result.columns
result = result[['MovieID', 'Title_x', 'UserID', 'Rating', 'Gender', 'Age', 'Occupation', 'Genres']]
result.head()
result.rename(columns = {'Title_x':'Title'}, inplace = True)
result.head()
```

Code Block 23 Results:

```
MovieID      Title UserID Rating Gender Age Occupation Genres
   0       1 Toy Story (1995)    1     5   F   1    10  Animation|Children's|Comedy
   1     48 Pocahontas (1995)    1     5   F   1    10  Animation|Children's|Romance
   2    150 Apollo 13 (1995)    1     5   F   1    10        Drama
   3   260 Star Wars Episode IV - A New Hope (1977)    1     4   F   1    10  Action|Adventure|Fantasy|Sci-Fi
   4   527 Schindler's List (1993)    1     5   F   1    10        Drama|War
```

1. 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.

localhost:8890/lab/tree/Project%20%20-%20MovieLens-Final.ipynb

```
[38]: tidy_movie_ratings.columns
[39]: Index(['MovieID', 'Title', 'UserID', 'Rating', 'Gender', 'Age', 'Occupation',
   'Action', 'Adventure', 'Animation', 'Childrens', 'Comedy', 'Crime',
   'Documentary', 'Drama', 'Fantasy', 'Film-Noir', 'Horror', 'Musical',
   'Mystery', 'Romance', 'Sci-Fi', 'Thriller', 'War', 'Western', 'Year'])
[39]: tidy_movie_ratings.shape
[39]: (1000209, 26)

Determine the features affecting the ratings of any particular movie. Removing few parameters like Title, User Id, Movie id which

[39]: X_feature = tidy_movie_ratings[['Gender', 'Age', 'Occupation',
   'Action', 'Adventure', 'Animation', 'Childrens', 'Comedy', 'Crime',
   'Documentary', 'Drama', 'Fantasy', 'Film-Noir', 'Horror', 'Musical',
   'Mystery', 'Romance', 'Sci-Fi', 'Thriller', 'War', 'Western']]
X_feature.shape
[39]: (1000209, 22)

[39]: X_feature.head()
[39]:
```

	Gender	Age	Occupation	Action	Adventure	Animation	Childrens	Comedy	Crime	Documentary	Drama	Fantasy	Film-Noir	Horror	Musical	Mystery	Romance	Sci-Fi	Thriller	War	Western	Year
0	F	1	10	0	0	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	1995
1	F	1	10	0	0	0	1	1	0	0	0	0	0	0	0	1	0	0	0	0	0	1995
2	F	1	10	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1995
3	F	1	10	1	1	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	1977
4	F	1	10	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1993

5 rows × 22 columns

```
[39]: y_target = tidy_movie_ratings[['Rating']]
[39]:
```

Warning: The default value of numeric_only in DataFrame.corr is deprecated. In a future version, it will default to False. Select only valid columns or specify the value of numeric_only to silence this warning.

```
[39]: data_corr = tidy_movie_ratings.corr()
```

Mode Command Line 1, Col 1 Project 2 - MovieLens-Final.ipynb 1 ENG IN 06:03 PM 12-Aug-23

1. Determine the features affecting the ratings of any particular movie.

localhost:8890/lab/tree/Project%20%20-%20MovieLens-Final.ipynb

```
[39]: #The 3 factors affecting the rating are: Genre, Age & Occupation
#Overall - Rating is affected by 2 Genres - Horror & Comedy
#Genre - (2 of the Sub genre had been highest)
#Followed by Age - Occupation

[39]: from sklearn.metrics import confusion_matrix
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder
#split the data into training data (75% is the training data and 25% is the testing data)
#by default tokin 75%-25% split
#Cross_valindate got away from python 2
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X_feature, Y_target, random_state=1)

[39]: numencoder = LabelEncoder()
X_train.Gender = numencoder.fit_transform(X_train['Gender'].astype('str'))
X_test.Gender = numencoder.fit_transform(X_test['Gender'].astype('str'))
y_train = numencoder.fit_transform(y_train.astype('int'))
y_test = numencoder.fit_transform(y_test.astype('int'))

C:\ProgramData\Anaconda3\lib\site-packages\sklearn\preprocessing\_label.py:116: DataConversionWarning: A column-vector y was passed when a 1d array was expected. Please change the shape of y to (n_samples, 1), for example by using ravel().
  warnings.warn("A column-vector y was passed when a 1d array was expected. Please change the shape of y to (n_samples, 1), for example by using ravel().")
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\preprocessing\_label.py:116: DataConversionWarning: A column-vector y was passed when a 1d array was expected. Please change the shape of y to (n_samples, 1), for example by using ravel().
  warnings.warn("A column-vector y was passed when a 1d array was expected. Please change the shape of y to (n_samples, 1), for example by using ravel()."
  (250556, 22)
(250557, 22)
(250558, 22)
(250559, 22)

[39]: #verify if the training and testing datasets are split correctly
print(y_train.shape)
print(X_train.shape)
print(y_test.shape)
print(X_test.shape)

(750556, 22)
(750557, 22)
(750558, 22)
(750559, 22)

Logistic regression is best used for predicting categorical data
need to do logistic regression on the training data so we can see how well our test data does the prediction
The dataset kept throwing off a non-convergence error where max iterations had been reached. I used the code below to increase the max iter.
However, for test purpose showing how linear regression model will fit

[39]: #Create a linear regression model
from sklearn.linear_model import LinearRegression
linreg = LinearRegression()
linreg.fit(X_train, y_train)
LinearRegression(cop=0, fit_intercept=True, n_jobs=None)
y_pred = linreg.predict(X_test)
```

Mode Command Line 1, Col 1 Project 2 - MovieLens-Final.ipynb 1 ENG IN 06:10 PM 12-Aug-23

Chat - Bing AI Project%202%-(5)-JupyterLab Project 2 - MovieLens-Final JupyterLab

localhost:8890/lab/tree/Project%202%20MovieLens-Final.ipynb

```
[33]: MovieID UserID Rating Age Occupation Action Adventure Animation Children's Comedy ... Fantasy Film-Noir Horror Musical Mystery Romance Sci-Fi Thriller War
MovieID 1.000000 -0.017739 -0.064042 0.027575 0.008585 -0.042046 -0.082413 -0.014177 -0.071589 0.061667 ... -0.018792 -0.019655 0.057613 -0.059381 -0.028561 -0.118375 -0.011747 -0.058418 -0.081951
UserID -0.017739 0.000000 0.012393 0.034688 -0.026698 -0.002023 -0.00683 -0.07665 -0.049681 -0.03951 ... 0.002312 0.004701 0.011982 -0.009223 0.004334 0.006834 -0.00283 -0.001107 0.003938
Rating -0.064042 0.012393 1.000000 0.056689 0.007653 -0.047633 -0.036718 0.019670 -0.039629 -0.039632 ... -0.023312 0.006259 0.009453 0.015643 0.015848 0.009564 -0.044887 -0.004898 0.075688
Age 0.027575 0.034688 0.056689 1.000000 0.078371 -0.030975 -0.016730 -0.047020 -0.025858 -0.044046 ... -0.024222 0.033495 -0.023891 0.005158 0.024608 0.017503 -0.010879 -0.014100 0.038446
Occupation -0.005885 -0.026698 0.006753 0.078371 1.000000 0.018347 0.000000 0.014039 -0.005349 -0.007731 ... 0.000421 -0.014018 0.026250 0.008981 0.010394
Action -0.042046 -0.002023 -0.047633 -0.030975 0.018347 1.000000 0.0374961 -0.110280 -0.141314 -0.026809 ... -0.014551 -0.008028 0.042733 -0.104032 -0.054084 -0.067830 0.319117 0.020726 0.138672
Adventure -0.082413 -0.00683 -0.036718 -0.016730 0.014309 0.0374961 1.000000 0.009384 -0.006096 -0.001049 ... -0.01299 0.005341 0.004149 -0.007731 0.000421 -0.014018 0.026250 0.008981 0.010394
Animation -0.014177 -0.007665 0.019670 0.047020 0.003884 -0.110280 0.004732 1.000000 0.076324 0.018544 ... -0.013025 0.007013 0.049730 0.035931 -0.043488 -0.055526 -0.085713 -0.046174
Children's -0.071589 -0.004862 -0.036929 -0.052058 -0.009908 -0.141314 0.008283 0.076204 1.000000 0.058711 ... -0.263280 -0.038033 -0.077099 0.312197 -0.052786 -0.084544 -0.038444 -0.132842 -0.065339
Comedy 0.061667 -0.003851 -0.039622 -0.040446 -0.005148 -0.268902 -0.14860 0.018544 0.098711 1.000000 ... -0.006110 -0.104245 -0.003064 0.005568 -0.105346 0.112843 -0.187079 -0.399501 -0.127101
Crime -0.061896 0.003469 0.033446 -0.007931 0.002821 0.008519 -0.045524 -0.062520 -0.081977 -0.078030 ... -0.023745 0.136237 0.047899 -0.061179 0.006093 -0.075320 -0.083730 0.115095 -0.079715
Documentary -0.005954 -0.001084 0.026998 0.004407 -0.005269 -0.052965 -0.053109 -0.081891 -0.024901 -0.040697 ... -0.017334 -0.021275 -0.025963 -0.007155 -0.018265 -0.037137 -0.039886 -0.04191 -0.016662
Drama -0.030856 0.008572 0.123561 0.063856 -0.032326 -0.020415 -0.14570 -0.154479 -0.135707 -0.248940 ... -0.009929 -0.087297 -0.189551 -0.047478 -0.037689 0.023552 -0.212747 -0.137117 0.185882
Fantasy -0.017972 0.002312 -0.023312 -0.001299 0.004551 0.027704 0.007205 0.023255 0.028239 -0.069610 ... 0.000000 0.026464 -0.055803 -0.020134 -0.039700 -0.014882 0.121943 -0.087374 -0.044038
Film-Noir -0.019655 0.004701 0.060259 0.033495 0.005246 -0.080288 -0.014178 0.037013 -0.030833 -0.101425 ... -0.026464 0.000000 -0.029157 -0.028384 0.215354 -0.047351 -0.040456 0.115231 -0.036984
Horror -0.057613 -0.001392 -0.094333 -0.023901 0.001439 -0.047733 -0.057250 -0.049790 -0.077099 -0.093040 ... -0.025803 -0.009157 -0.001892 0.003423 -0.009434 0.005695 -0.056629 -0.077985
Musical -0.059181 -0.000222 0.015643 0.005158 -0.005158 -0.007312 -0.104032 -0.022337 0.035231 0.012567 0.005054 ... -0.020134 0.003884 0.018924 1.000000 0.042581 0.023506 -0.060012 -0.100890 -0.034429
Mystery -0.023561 0.004334 0.015948 0.024098 0.002421 -0.054094 -0.043503 -0.042468 -0.057298 -0.105346 ... -0.003970 0.0215354 0.002423 -0.042581 1.000000 0.040162 -0.028273 0.225281 -0.055482
Romance -0.118375 0.006834 0.009644 0.017503 -0.014018 -0.067680 0.024389 -0.054540 -0.084950 0.112843 ... -0.014822 -0.047351 -0.094943 0.023506 -0.04162 1.000000 0.137752 -0.081384 0.053347
Sci-Fi -0.011747 -0.003283 -0.044867 -0.010189 0.026256 0.031917 0.028419 -0.055554 -0.068844 -0.180779 ... -0.121843 0.004056 0.056505 -0.068012 0.028273 -0.133752 1.000000 0.102546 0.339314
Thriller -0.054818 -0.001107 -0.004806 -0.014100 0.009881 0.020756 -0.038423 -0.085713 -0.132642 -0.299501 ... -0.087374 0.115231 0.056629 -0.100690 0.225281 -0.081384 0.102546 1.000000 -0.080818
War -0.081951 0.003502 0.075688 0.038446 0.010264 0.135872 0.016647 -0.046114 -0.066539 -0.127101 ... -0.044928 -0.036984 -0.077985 -0.024429 -0.055482 0.053347 -0.039914 -0.080818 1.000000
Western 0.035940 0.004114 0.007311 0.038177 0.005934 0.022242 -0.011964 -0.050908 -0.031269 0.007927 ... -0.028199 -0.079816 -0.041784 -0.030245 -0.029727 -0.044650 -0.010935 -0.058897 -0.019803
```

2 rows × 23 columns

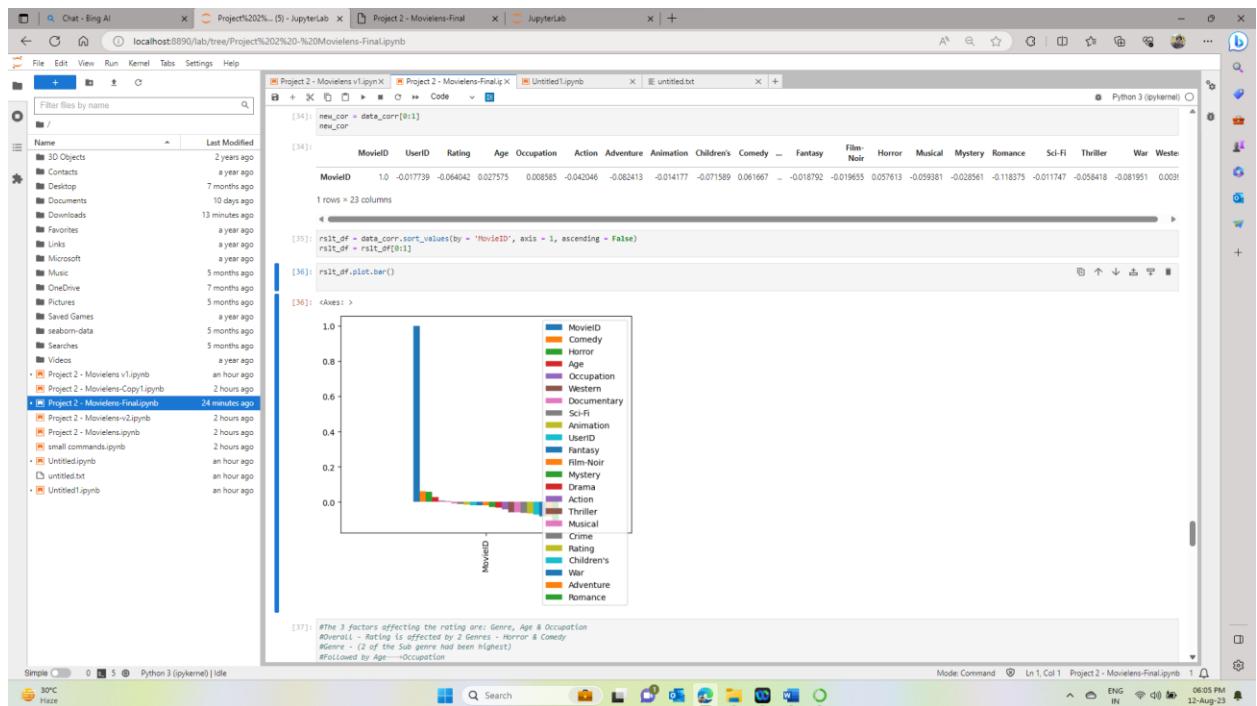
[34]: new_corr = data_corr[0:1]

[34]: new_corr

[34]: MovieID UserID Rating Age Occupation Action Adventure Animation Children's Comedy ... Fantasy Film-Noir Horror Musical Mystery Romance Sci-Fi Thriller War Weste

MovieID 1.0 -0.017739 -0.064042 0.027575 0.008585 -0.042046 -0.082413 -0.014177 -0.071589 0.061667 ... -0.018792 -0.019655 0.057613 -0.059381 -0.028561 -0.118375 -0.011747 -0.058418 -0.081951 0.039

Mode Command Ln 1, Col 1 Project 2 - MovieLens-Final.ipynb 1 ENG IN 06:04 PM 12-Aug-23



2. Develop an appropriate model to predict the movie ratings

Chat - Bing AI Project%20%20... (5) - JupyterLab Project 2 - MovieLens-Final.ipynb JupyterLab

localhost:8890/lab/tree/Project%20%20.../Project%20%20MovieLens-Final.ipynb

File Edit View Run Kernel Tabs Settings Help

Project files by name

```

[39] #Create a linear regression model
from sklearn.linear_model import LinearRegression
linreg = LinearRegression()
linreg.fit(x_train, y_train)
LinearRegression(copy_X=True, fit_intercept=True, n_jobs=None)
y_pred = linreg.predict(x_test)

[40] print(
    f"\ny-intercept: {linreg.intercept_}
    Beta coefficients: {linreg.coef_}
    ")

y-intercept: 17.24068511e+002
Beta coefficients: [-0.07578815  0.00171463  0.00113505 -0.0042597 -0.00328118  0.43476483
-0.34521155  0.02147105  0.11664644  0.85193036  0.16946938  0.87071357
0.365522 -0.35344409 -0.02792581  0.00224498 -0.02054086
0.12063908  0.21325392  0.02808645 -0.01696708 -0.02568372]

[41] from sklearn import metrics

print(
    f"\nMean Abs Error : {metrics.mean_absolute_error(y_test, y_pred)}
    Mean Sq Error : {metrics.mean_squared_error(y_test, y_pred)}
    Root Mean Sq Error : {np.sqrt(metrics.mean_squared_error(y_test, y_pred))}

Mean Abs Error : 0.887080242578026
Mean Sq Error : 1.173074930231644
Root Mean Sq Error : 1.083068554166893
r2 value: 0.899521255993674
As we see the linear model is not the right model for this dataset Develop an appropriate model to predict the movie ratings

```

[42] # Create a logistic regression model using the training set
from sklearn.linear_model import LogisticRegression
logreg = LogisticRegression()

Mode Command Ln 1, Col 1 Project 2 - MovieLens-Final.ipynb

Python 3 (ipykernel) | idle

30°C Haze

Chat - Bing AI Project%20%20... (5) - JupyterLab Project 2 - MovieLens-Final.ipynb JupyterLab

localhost:8890/lab/tree/Project%20%20.../Project%20%20MovieLens-Final.ipynb

File Edit View Run Kernel Tabs Settings Help

Project files by name

```

[42] # Create a logistic regression model using the training set
from sklearn.linear_model import LogisticRegression
logreg = LogisticRegression()
logreg.fit(x_train, y_train)

C:\ProgramData\anaconda3\lib\site-packages\sklearn\linear_model\_logistic.py:458: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. OF ITERATIONS REACHED !!

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#logistic-regression
n_iter_1 = _check_optimize_result(
    'LogisticRegression',
    LogisticRegression())

[43] # Make predictions using the testing set
y_pred = logreg.predict(x_test)
y_pred

[44] array([1, 0, 1, ..., 0, 1, 0], dtype=int64)

[45] #Evaluate the accuracy of your model
print(logreg.intercept_)
print(logreg.coef_)
from sklearn import metrics
import numpy as np
y_pred = logreg._count_probas_(x_test, y_pred)
from sklearn import metrics
metrics.accuracy_score(y_test, y_pred)

[-3.7291239e-05 -5.5854297e-05 -5.45902491e-05 2.62081779e-05
1.2223642e-04]
[1.3952111e-04 -1.6873674e-04 2.4479181e-03 6.74231059e-03
2.9730227e-04 -1.42379546e-04 5.51223705e-04 4.59621455e-04
-4.11938206e-04 -5.46386622e-05 -2.51139508e-03 9.3765430e-05
-2.42825115e-04 1.40546139e-04 -9.7594776e-05 -1.76485999e-04
-2.30822123e-04 -1.27432122e-04 1.87377332e-04 -4.60276606e-04
-2.07144237e-05 -2.9375823e-04]
[-3.7291239e-05 -5.5854297e-05 -5.45902491e-05 2.62081779e-05
6.36329604e-04 -2.9168923e-03 1.42359221e-04 1.16542314e-03
-1.1952111e-04 -1.6873674e-04 2.4479181e-03 6.74231059e-03
-3.71832462e-04 -1.23840425e-04 -2.4299117e-03 2.3991160e-04
-3.59845505e-04 1.7688745e-03 -2.30021381e-04 -4.79737592e-05
-1.7592111e-04 -1.27432122e-04 2.039957e-04 -8.14791688e-04
-2.08168951e-04 -1.72433842e-04]
[ 4.9463515e-04 4.91744717e-03 3.74221108e-04 1.06237850e-03
9.49359364e-04 -8.53344156e-04 5.65308083e-04 1.2323252e-03
-4.27038221e-04 -1.27432122e-04 1.87377332e-04 -4.60276606e-04
-5.19742406e-04 4.96374374e-03 -8.747042355e-05 -9.45512977e-05
6.29113779e-04 5.61114125e-04 2.1763957e-04 -1.42452022e-03

```

Mode Edit Ln 1, Col 37 Project 2 - MovieLens-Final.ipynb

Python 3 (ipykernel) | idle

30°C Haze

Chat - Bing AI Project%20%20-(5)-JupyterLab Project 2 - MovieLens-Final.ipynb JupyterLab

localhost:8890/lab/tree/Project%20%20-%20MovieLens-Final.ipynb

File Edit View Run Kernel Tabs Settings Help

Project 2 - MovieLens v1.ipynb Project 2 - MovieLens-Final.ipynb Untitled1.ipynb Python 3 (ipykernel)

Filter files by name

Name Last Modified

- 3D Objects 2 years ago
- Contacts a year ago
- Desktop 7 months ago
- Documents 10 days ago
- Downloads 21 minutes ago
- Favorites a year ago
- Links a year ago
- Microsoft a year ago
- Music 5 months ago
- OneDrive 7 months ago
- Pictures 5 months ago
- Saved Games a year ago
- seaborn-data 5 months ago
- Searches 5 months ago
- Videos a year ago
- Project 2 - MovieLens v1.ipynb an hour ago
- Project 2 - MovieLens-Copy1.ipynb seconds ago
- Project 2 - MovieLens-Final.ipynb 2 hours ago
- Project 2 - MovieLens-v2.ipynb 2 hours ago
- Project 2 - MovieLens.ipynb 2 hours ago
- small commands.ipynb 2 hours ago
- Untitled.ipynb an hour ago
- Untitled1.ipynb an hour ago
- Untitled2.ipynb an hour ago

```
[44]: metrics.mae(y_true, y_pred)

[45]: #Print the first 30 actual and predicted responses
print('Actual : ', y_test[:30])
print('Predicted : ', y_pred[:30])

Actual : [3 4 2 4 3 2 2 3 1 4 0 3 4 2 3 3 2 4 4 0 4 2 3 4 9 8 3 4 2]
Predicted : [3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3]
```

Chat - Bing AI Project%20%20-(5)-JupyterLab Project 2 - MovieLens-Final.ipynb JupyterLab

localhost:8890/lab/tree/Project%20%20-%20MovieLens-Final.ipynb

File Edit View Run Kernel Tabs Settings Help

Project 2 - MovieLens v1.ipynb Project 2 - MovieLens-Final.ipynb Untitled1.ipynb Python 3 (ipykernel)

Filter files by name

Name Last Modified

- 3D Objects 2 years ago
- Contacts a year ago
- Desktop 7 months ago
- Documents 10 days ago
- Downloads 14 minutes ago
- Favorites a year ago
- Links a year ago
- Microsoft a year ago
- Music 5 months ago
- OneDrive 7 months ago
- Pictures 5 months ago
- Saved Games a year ago
- seaborn-data 5 months ago
- Searches 5 months ago
- Videos a year ago
- Project 2 - MovieLens v1.ipynb an hour ago
- Project 2 - MovieLens-Copy1.ipynb 2 hours ago
- Project 2 - MovieLens-Final.ipynb 25 minutes ago
- Project 2 - MovieLens-v2.ipynb 2 hours ago
- Project 2 - MovieLens.ipynb 2 hours ago
- small commands.ipynb 2 hours ago
- Untitled.ipynb an hour ago
- Untitled1.ipynb an hour ago
- Untitled2.ipynb an hour ago

```
[41]: #Create a linear regression model
from sklearn.linear_model import LinearRegression
linreg = LinearRegression()
linreg.fit(x_train, y_train)
LinearRegression(copy=True, fit_intercept=True, n_jobs=None,
normalize=False)
y_pred = linreg.predict(x_test)

[42]: ---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[41], line 5
      1 linreg = LinearRegression()
      2 4 linreg.fit(x_train, y_train)
----> 5 LinearRegression(copy=True, fit_intercept=True, n_jobs=None,
      6 normalize=False)
      7 y_pred = linreg.predict(x_test)

TypeError: LinearRegression.__init__() got an unexpected keyword argument 'normalize'

[43]: print(
    '-y-intercept: ',
    linreg.intercept_
)
print(
    '-Beta coefficients: ',
    linreg.coef_
)

[44]: from sklearn import metrics
print(
    '-Mean Abs Error MAE: ',
    metrics.mean_absolute_error(y_test, y_pred)
)
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)
)
```

As we see the linear model is not the right model for this dataset Develop an appropriate model to predict the movie ratings

```
[45]: # Create a logistic regression model using the training set
from sklearn.linear_model import LogisticRegression
logreg = LogisticRegression()
```

Embedded File:

Phython File:



Project 2 - Movielens.ipynb

Code hosted on GitHub :

[ks-alokranjan/Simplilearn-Project \(github.com\)](https://github.com/ks-alokranjan/Simplilearn-Project)