Project to Submit – Project 2

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.

**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 |
| Genere | 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' | * + - 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

1. User Age Distribution – Barchart
2. User rating of the movie “Toy Story” – Barchart & line graph
3. Top 25 movies by viewership rating – Table Representation
4. Find the ratings for all the movies reviewed by for a particular user of user id = 2696 - Table Representation

* Feature Engineering:

            Use column genres:

1. 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
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.
   * 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
3. Determine the features affecting the ratings of any particular movie.
   * Using Correlation function the factor affecting was calculated
4. 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 Simpililearn Movielens - You need to find features affecting the ratings of any particular movie and build a model to predict the movie ratings.

Import Right Libraries

In [1]:

**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

In [2]:

movies\_df **=** pd**.**read\_csv('C:\\Users\\naseh\\OneDrive\\Studies\\AI - ML\\Python\\Examples\\Project to Submit\\movies.dat', sep**=**'::', engine**=**'python',

names**=**['MovieID','Title','Genres'], header**=None**)

movies\_df**.**head()

Out[2]:

|  | **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 |

Users Data base in users\_df

In [3]:

users\_df **=** pd**.**read\_csv('C:\\Users\\naseh\\OneDrive\\Studies\\AI - ML\\Python\\Examples\\Project to Submit\\users.dat', sep**=**'::',engine**=**'python',

names**=**['UserID','Gender','Age', 'Occupation', 'zip-code'], header**=None**)

users\_df**.**head()

Out[3]:

|  | **UserID** | **Gender** | **Age** | **Occupation** | **zip-code** |
| --- | --- | --- | --- | --- | --- |
| **0** | 1 | F | 1 | 10 | 48067 |
| **1** | 2 | M | 56 | 16 | 70072 |
| **2** | 3 | M | 25 | 15 | 55117 |
| **3** | 4 | M | 45 | 7 | 02460 |
| **4** | 5 | M | 25 | 20 | 55455 |

Ratings Data base in ratings\_df

In [4]:

ratings\_df **=** pd**.**read\_csv('C:\\Users\\naseh\\OneDrive\\Studies\\AI - ML\\Python\\Examples\\Project to Submit\\ratings.dat', sep**=**'::', engine**=**'python',

names**=**['UserID','MovieID','Rating', 'Timestamp'], header**=None**)

ratings\_df**.**head()

Out[4]:

|  | **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 [5]:

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[5]:

|  | **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

In [6]:

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

6 Occupation 1000209 non-null int64

7 zip-code 1000209 non-null object

dtypes: int64(5), object(3)

memory usage: 68.7+ MB

Out[6]:

|  | **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 [7]:

*# 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[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 |

• 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 [8]:

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

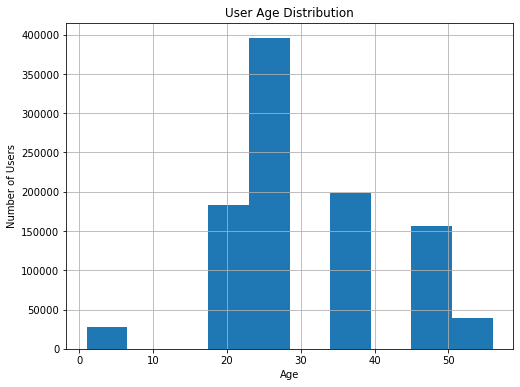
Master\_df**.**Age**.**hist()

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 [9]:

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

movies\_grouped **=** Master\_df**.**groupby('Title')

toy\_story **=** movies\_grouped**.**get\_group('Toy Story (1995)')

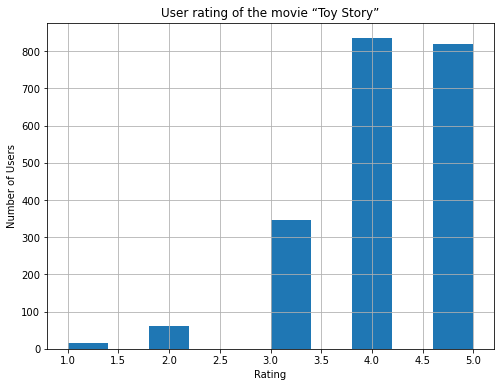
toy\_story['Rating']**.**hist()

plt**.**title('User rating of the movie “Toy Story”')

plt**.**xlabel('Rating')

plt**.**ylabel('Number of Users')

plt**.**show()



In [10]:

*#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[10]:

Age

1 27211

18 183536

25 395556

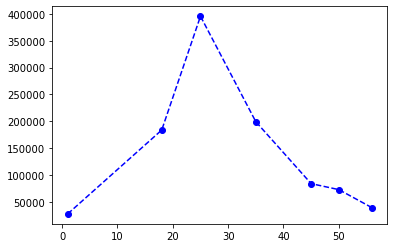
35 199003

45 83633

50 72490

56 38780

Name: MovieID, dtype: int64



In [11]:

Master\_df**.**groupby("Age")["MovieID"]**.**count()**.**describe()

Out[11]:

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

• 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

In [12]:

rating\_avg **=** Master\_df**.**groupby('Title')['Rating']**.**mean()

rating\_avg**.**head()

rating\_avg **=** rating\_avg**.**sort\_values(ascending**=False**)

rating\_avg[0:25]

Out[12]:

Title

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

Smashing Time (1967) 5.000000

Schlafes Bruder (Brother of Sleep) (1995) 5.000000

Gate of Heavenly Peace, The (1995) 5.000000

Baby, The (1973) 5.000000

I Am Cuba (Soy Cuba/Ya Kuba) (1964) 4.800000

Lamerica (1994) 4.750000

Apple, The (Sib) (1998) 4.666667

Sanjuro (1962) 4.608696

Seven Samurai (The Magnificent Seven) (Shichinin no samurai) (1954) 4.560510

Shawshank Redemption, The (1994) 4.554558

Godfather, The (1972) 4.524966

Close Shave, A (1995) 4.520548

Usual Suspects, The (1995) 4.517106

Schindler's List (1993) 4.510417

Wrong Trousers, The (1993) 4.507937

Dry Cleaning (Nettoyage � sec) (1997) 4.500000

Inheritors, The (Die Siebtelbauern) (1998) 4.500000

Mamma Roma (1962) 4.500000

Bells, The (1926) 4.500000

Name: Rating, dtype: float64

In [13]:

rating\_avg\_count **=** pd**.**DataFrame(data**=**rating\_avg)

rating\_avg\_count['number\_of\_ratings'] **=** pd**.**DataFrame(rating\_avg\_count)

rating\_avg\_count**.**head()

Out[13]:

|  | **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 [14]:

*#Top 25 movies by viewership rating*

res **=** Master\_df**.**groupby("Title")**.**size()**.**sort\_values(ascending**=False**)[:25]

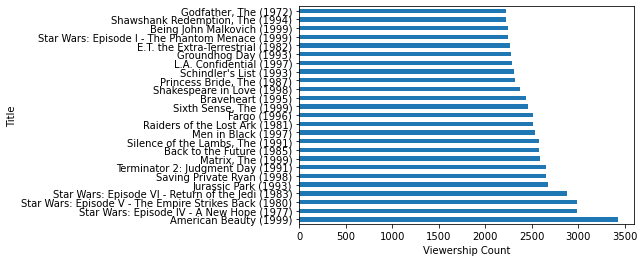
plt**.**ylabel("Title")

plt**.**xlabel("Viewership Count")

res**.**plot(kind**=**"barh")

Out[14]:

<AxesSubplot:xlabel='Viewership Count', ylabel='Title'>



• 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 [15]:

user\_2696 **=** Master\_df[Master\_df['UserID'] **==** 2696]

user\_2696

Out[15]:

|  | **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 |
| **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 [16]:

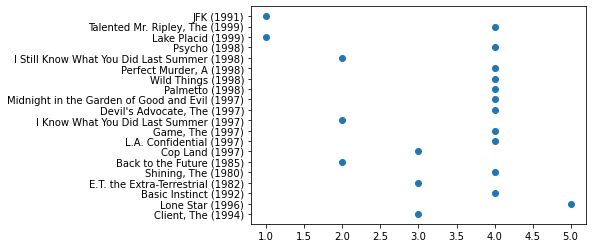
res **=** Master\_df[Master\_df**.**UserID **==** 2696]

plt**.**scatter(y**=**res**.**Title, x**=**res**.**Rating)

res

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 |
| **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 [17]:

movies\_df**.**head()

Out[17]:

|  | **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 [18]:

movies\_df['Genres']**.**value\_counts()**.**head()

Out[18]:

Drama 843

Comedy 521

Horror 178

Comedy|Drama 162

Comedy|Romance 142

Name: Genres, dtype: int64

In [19]:

movies\_df['Genres']**.**unique()

Out[19]:

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 [20]:

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

In [21]:

Master\_df**.**head()

Out[21]:

|  | **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 [22]:

*#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[22]:

|  | **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 |
| **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 [23]:

dummies **=** result['Genres']**.**str**.**get\_dummies()

dummies**.**head()

Out[23]:

|  | **Action** | **Adventure** | **Animation** | **Children's** | **Comedy** | **Crime** | **Documentary** | **Drama** | **Fantasy** | **Film-Noir** | **Horror** | **Musical** | **Mystery** | **Romance** | **Sci-Fi** | **Thriller** | **War** | **Western** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **0** | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| **1** | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 0 |
| **2** | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| **3** | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 |
| **4** | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 |

In [24]:

tidy\_movie\_ratings **=** (pd**.**concat([result, dummies], axis**=**1)

**.**drop([ "Genres"], axis**=**1)

)

tidy\_movie\_ratings**.**head()

Out[24]:

|  | **MovieID** | **Title** | **UserID** | **Rating** | **Gender** | **Age** | **Occupation** | **Action** | **Adventure** | **Animation** | **...** | **Fantasy** | **Film-Noir** | **Horror** | **Musical** | **Mystery** | **Romance** | **Sci-Fi** | **Thriller** | **War** | **Western** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **0** | 1 | Toy Story (1995) | 1 | 5 | F | 1 | 10 | 0 | 0 | 1 | ... | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| **1** | 48 | Pocahontas (1995) | 1 | 5 | F | 1 | 10 | 0 | 0 | 1 | ... | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 0 |
| **2** | 150 | Apollo 13 (1995) | 1 | 5 | F | 1 | 10 | 0 | 0 | 0 | ... | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| **3** | 260 | Star Wars: Episode IV - A New Hope (1977) | 1 | 4 | F | 1 | 10 | 1 | 1 | 0 | ... | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 |
| **4** | 527 | Schindler's List (1993) | 1 | 5 | F | 1 | 10 | 0 | 0 | 0 | ... | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 |

5 rows × 25 columns

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

In [25]:

tidy\_movie\_ratings["Year"] **=** tidy\_movie\_ratings["Title"]**.**str[**-**5:**-**1]

tidy\_movie\_ratings["movie\_title"] **=** tidy\_movie\_ratings["Title"]**.**str[:**-**7]

In [26]:

tidy\_movie\_ratings**.**reset\_index(inplace**=True**)

tidy\_movie\_ratings**=**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 [27]:

tidy\_movie\_ratings**.**columns

Out[27]:

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 [28]:

tidy\_movie\_ratings**.**head()

tidy\_movie\_ratings**.**shape

Out[28]:

(1000209, 26)

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

In [29]:

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[29]:

(1000209, 22)

In [30]:

X\_feature**.**head()

Out[30]:

|  | **Gender** | **Age** | **Occupation** | **Action** | **Adventure** | **Animation** | **Children's** | **Comedy** | **Crime** | **Documentary** | **...** | **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 | 1995 |
| **1** | F | 1 | 10 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | ... | 0 | 0 | 1 | 0 | 1 | 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 | 1995 |
| **3** | F | 1 | 10 | 1 | 1 | 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 | 1 | 0 | 1993 |

5 rows × 22 columns

In [31]:

Y\_target **=** tidy\_movie\_ratings[['Rating']]

In [32]:

*#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

Out[32]:

|  | **MovieID** | **UserID** | **Rating** | **Age** | **Occupation** | **Action** | **Adventure** | **Animation** | **Children's** | **Comedy** | **...** | **Fantasy** | **Film-Noir** | **Horror** | **Musical** | **Mystery** | **Romance** | **Sci-Fi** | **Thriller** | **War** | **Western** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **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 | 0.003940 |
| **UserID** | -0.017739 | 1.000000 | 0.012303 | 0.034688 | -0.026698 | -0.002023 | -0.000683 | -0.007665 | -0.004862 | -0.003651 | ... | 0.002212 | 0.004701 | -0.001392 | -0.000222 | 0.004334 | 0.006834 | -0.003283 | -0.001107 | 0.003502 | 0.004114 |
| **Rating** | -0.064042 | 0.012303 | 1.000000 | 0.056869 | 0.006753 | -0.047633 | -0.036718 | 0.019670 | -0.039829 | -0.039622 | ... | -0.023312 | 0.060259 | -0.094353 | 0.015643 | 0.015848 | 0.009644 | -0.044487 | -0.004806 | 0.075688 | 0.007311 |
| **Age** | 0.027575 | 0.034688 | 0.056869 | 1.000000 | 0.078371 | -0.030975 | -0.016730 | -0.047020 | -0.052858 | -0.044046 | ... | -0.024222 | 0.033495 | -0.023901 | 0.005158 | 0.024308 | 0.017503 | -0.010879 | -0.014100 | 0.038446 | 0.038177 |
| **Occupation** | 0.008585 | -0.026698 | 0.006753 | 0.078371 | 1.000000 | 0.018347 | 0.014309 | -0.003834 | -0.006906 | -0.006149 | ... | 0.001299 | 0.005246 | 0.001439 | -0.007312 | 0.002421 | -0.014018 | 0.026250 | 0.008981 | 0.010264 | 0.005924 |
| **Action** | -0.042046 | -0.002023 | -0.047633 | -0.030975 | 0.018347 | 1.000000 | 0.374961 | -0.110294 | -0.141314 | -0.268092 | ... | 0.014551 | -0.080288 | -0.042733 | -0.100432 | -0.054084 | -0.067830 | 0.319117 | 0.202756 | 0.135872 | 0.022242 |
| **Adventure** | -0.082413 | -0.000683 | -0.036718 | -0.016730 | 0.014309 | 0.374961 | 1.000000 | 0.004732 | 0.098283 | -0.124960 | ... | 0.227046 | -0.014178 | -0.057256 | -0.022327 | -0.043503 | -0.024389 | 0.284190 | -0.038423 | 0.016647 | -0.011964 |
| **Animation** | -0.014177 | -0.007665 | 0.019670 | -0.047020 | -0.003834 | -0.110294 | 0.004732 | 1.000000 | 0.576204 | 0.018544 | ... | 0.012025 | 0.037013 | -0.049730 | 0.335231 | -0.042488 | -0.054540 | -0.055526 | -0.085713 | -0.046114 | -0.030908 |
| **Children's** | -0.071589 | -0.004862 | -0.039829 | -0.052858 | -0.006906 | -0.141314 | 0.098283 | 0.576204 | 1.000000 | 0.058711 | ... | 0.263280 | -0.038033 | -0.077099 | 0.312567 | -0.052786 | -0.084550 | -0.038844 | -0.132642 | -0.066539 | -0.031269 |
| **Comedy** | 0.061667 | -0.003651 | -0.039622 | -0.044046 | -0.006149 | -0.268092 | -0.124960 | 0.018544 | 0.058711 | 1.000000 | ... | -0.006010 | -0.101425 | -0.093064 | 0.030566 | -0.105346 | 0.112843 | -0.187079 | -0.299501 | -0.127101 | 0.007927 |
| **Crime** | -0.061896 | 0.003469 | 0.033446 | -0.007931 | 0.002821 | 0.088519 | -0.045924 | -0.062520 | -0.081977 | -0.078030 | ... | -0.033745 | 0.136237 | -0.047899 | -0.061179 | 0.080093 | -0.073320 | -0.083730 | 0.115095 | -0.079715 | -0.042711 |
| **Documentary** | -0.009544 | -0.001064 | 0.028098 | 0.004407 | -0.002689 | -0.052565 | -0.035109 | -0.018991 | -0.024901 | -0.040697 | ... | -0.017326 | -0.012175 | -0.025673 | -0.007155 | -0.018265 | -0.037137 | -0.038568 | -0.043191 | -0.016082 | -0.012974 |
| **Drama** | -0.030856 | 0.006572 | 0.122561 | 0.063856 | -0.012326 | -0.202415 | -0.194570 | -0.154479 | -0.135707 | -0.249840 | ... | -0.096929 | -0.067297 | -0.189551 | -0.094778 | -0.027689 | 0.023552 | -0.212747 | -0.153717 | 0.136582 | -0.045945 |
| **Fantasy** | -0.018792 | 0.002212 | -0.023312 | -0.024222 | 0.001299 | 0.014551 | 0.227046 | 0.012025 | 0.263280 | -0.006010 | ... | 1.000000 | -0.026464 | -0.055803 | -0.020134 | -0.039700 | -0.014822 | 0.121843 | -0.087374 | -0.044928 | -0.028199 |
| **Film-Noir** | -0.019655 | 0.004701 | 0.060259 | 0.033495 | 0.005246 | -0.080288 | -0.014178 | 0.037013 | -0.038033 | -0.101425 | ... | -0.026464 | 1.000000 | -0.039157 | -0.028384 | 0.215354 | -0.047351 | -0.004056 | 0.115231 | -0.036984 | -0.019816 |
| **Horror** | 0.057613 | -0.001392 | -0.094353 | -0.023901 | 0.001439 | -0.042733 | -0.057256 | -0.049730 | -0.077099 | -0.093064 | ... | -0.055803 | -0.039157 | 1.000000 | -0.018924 | -0.002423 | -0.099434 | 0.056505 | 0.056629 | -0.077985 | -0.041784 |
| **Musical** | -0.059381 | -0.000222 | 0.015643 | 0.005158 | -0.007312 | -0.100432 | -0.022327 | 0.335231 | 0.312567 | 0.030566 | ... | -0.020134 | -0.028384 | -0.018924 | 1.000000 | -0.042581 | 0.023506 | -0.068012 | -0.100690 | -0.034429 | -0.030245 |
| **Mystery** | -0.028561 | 0.004334 | 0.015848 | 0.024308 | 0.002421 | -0.054084 | -0.043503 | -0.042488 | -0.052786 | -0.105346 | ... | -0.039700 | 0.215354 | -0.002423 | -0.042581 | 1.000000 | -0.040162 | -0.028273 | 0.225281 | -0.055482 | -0.029727 |
| **Romance** | -0.118375 | 0.006834 | 0.009644 | 0.017503 | -0.014018 | -0.067830 | -0.024389 | -0.054540 | -0.084550 | 0.112843 | ... | -0.014822 | -0.047351 | -0.099434 | 0.023506 | -0.040162 | 1.000000 | -0.133752 | -0.081384 | 0.053347 | -0.044650 |
| **Sci-Fi** | -0.011747 | -0.003283 | -0.044487 | -0.010879 | 0.026250 | 0.319117 | 0.284190 | -0.055526 | -0.038844 | -0.187079 | ... | 0.121843 | -0.004056 | 0.056505 | -0.068012 | -0.028273 | -0.133752 | 1.000000 | 0.102546 | 0.039314 | -0.010935 |
| **Thriller** | -0.058418 | -0.001107 | -0.004806 | -0.014100 | 0.008981 | 0.202756 | -0.038423 | -0.085713 | -0.132642 | -0.299501 | ... | -0.087374 | 0.115231 | 0.056629 | -0.100690 | 0.225281 | -0.081384 | 0.102546 | 1.000000 | -0.088018 | -0.058897 |
| **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 | -0.034429 | -0.055482 | 0.053347 | 0.039314 | -0.088018 | 1.000000 | -0.019803 |
| **Western** | 0.003940 | 0.004114 | 0.007311 | 0.038177 | 0.005924 | 0.022242 | -0.011964 | -0.030908 | -0.031269 | 0.007927 | ... | -0.028199 | -0.019816 | -0.041784 | -0.030245 | -0.029727 | -0.044650 | -0.010935 | -0.058897 | -0.019803 | 1.000000 |

23 rows × 23 columns

In [33]:

new\_cor **=** data\_corr[0:1]

new\_cor

Out[33]:

|  | **MovieID** | **UserID** | **Rating** | **Age** | **Occupation** | **Action** | **Adventure** | **Animation** | **Children's** | **Comedy** | **...** | **Fantasy** | **Film-Noir** | **Horror** | **Musical** | **Mystery** | **Romance** | **Sci-Fi** | **Thriller** | **War** | **Western** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **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 | 0.00394 |

1 rows × 23 columns

In [34]:

rslt\_df **=** data\_corr**.**sort\_values(by **=** 'MovieID', axis **=** 1, ascending **=** **False**)

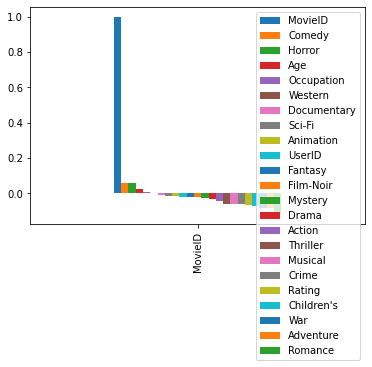
rslt\_df **=** rslt\_df[0:1]

In [35]:

rslt\_df**.**plot**.**bar()

Out[35]:

<AxesSubplot:>



In [36]:

*#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 [37]:

**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 [38]:

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\pandas\core\generic.py:5494: SettingWithCopyWarning:

A value is trying to be set on a copy of a slice from a DataFrame.

Try using .loc[row\_indexer,col\_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user\_guide/indexing.html#returning-a-view-versus-a-copy

self[name] = value

C:\ProgramData\Anaconda3\lib\site-packages\sklearn\utils\validation.py:63: DataConversionWarning: A column-vector y was passed when a 1d array was expected. Please change the shape of y to (n\_samples, ), for example using ravel().

return f(\*args, \*\*kwargs)

In [39]:

*#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 witll fail

In [40]:

*#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)

In [41]:

print(

'y-intercept: ',

linreg**.**intercept\_

)

print(

'Beta coefficients: ',

linreg**.**coef\_

)

y-intercept: 27.624005321159167

Beta coefficients: [-0.03796836 0.00171463 0.00113855 -0.04842597 -0.00326118 0.43476483

-0.34522135 0.05247497 0.11664944 0.54390266 0.26949838 0.07071527

0.266522 -0.31644409 -0.02792981 -0.01696708 0.00224498 -0.02054036

0.12063909 0.21825392 0.02808645 -0.01268372]

In [42]:

**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)

)

Mean Abs Error MAE: 0.8876802429578026

Mean Sq Error MSE: 1.1730374930251644

Root Mean Sq Error RMSE: 1.0830685541668932

r2 value: 0.05995212559930674

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

In [ ]:

*# Create a logistic regression model using the training set*

**from** sklearn.linear\_model **import** LogisticRegression

logreg **=** LogisticRegression()

logreg**.**fit(x\_train, y\_train)

In [ ]:

*#Make predictions using the testing set*

y\_pred **=** logreg**.**predict(x\_test)

y\_pred

In [ ]:

*#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)

In [ ]:

*#Print the first 30 actual and predicted responses*

print('Actual : ', y\_test[0:30])

print('Predicted :', y\_pred[0:30])

In [ ]:

**from** sklearn **import** svm

In [ ]:

*#Create a svm Classifier*

clf **=** svm**.**SVC(kernel**=**'poly', degree **=** 5)

In [ ]:

*#Train the model using the training sets*

clf**.**fit(x\_train, y\_train)

In [ ]:

svc **=** SVC()

svc**.**fit(x\_train, y\_train)

y\_pred **=** svc**.**predict(y\_test)

acc\_svc **=** round(svc**.**score(y\_test, y\_pred) **\*** 100, 2)

acc\_svc

In [ ]:

*#Looking at KNN*

*#import required packages*

**from** sklearn **import** neighbors

**from** sklearn.neighbors **import** KNeighborsRegressor

**from** sklearn.metrics **import** mean\_squared\_error

**from** math **import** sqrt

**import** matplotlib.pyplot **as** plt

**%matplotlib** inline

In [ ]:

knn **=** KNeighborsClassifier(n\_neighbors **=** 3)

knn**.**fit(x\_train, y\_train)

y\_pred **=** knn**.**predict(y\_test)

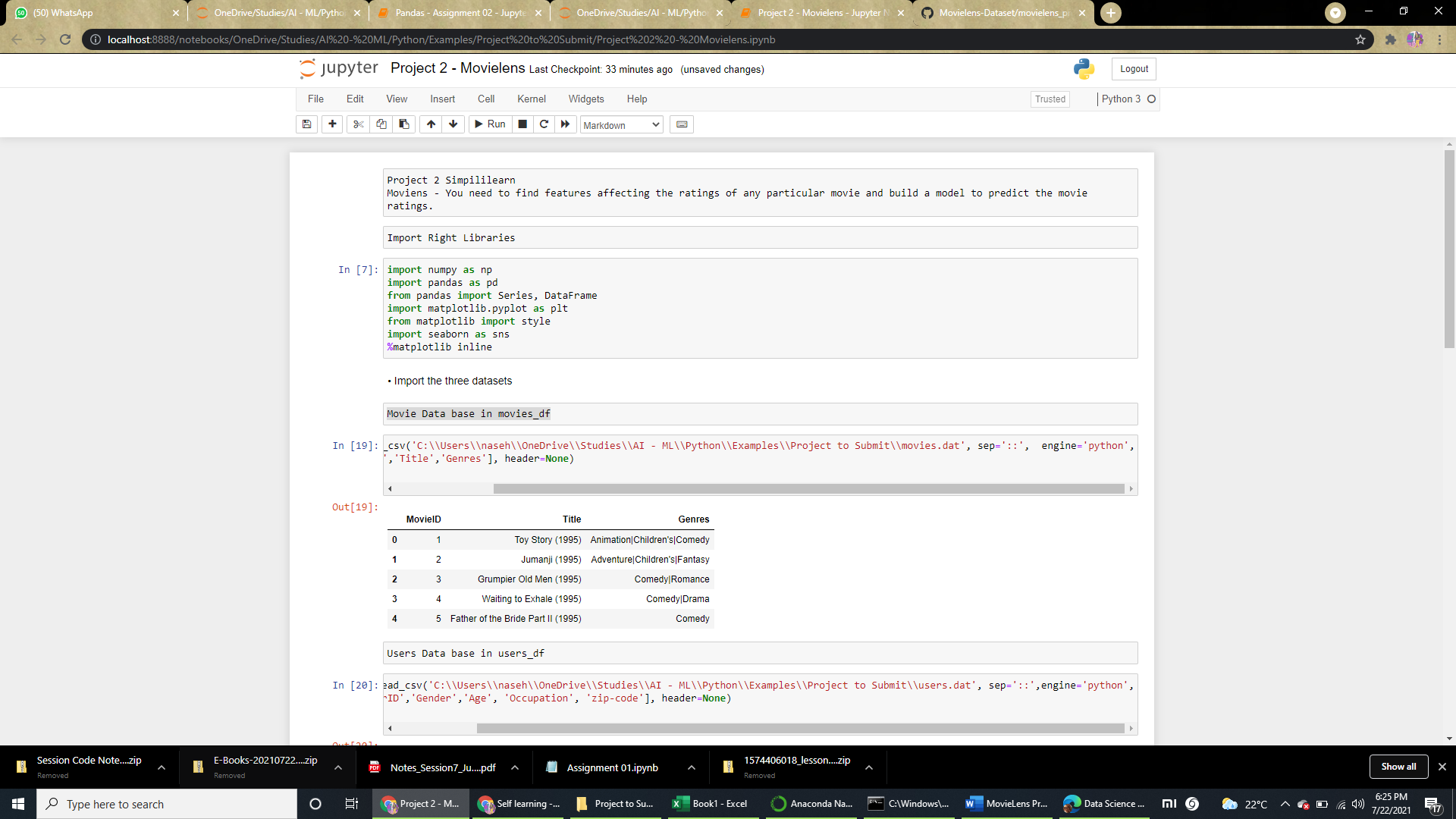
acc\_knn **=** round(knn**.**score(train, train\_labels) **\*** 100, 2)

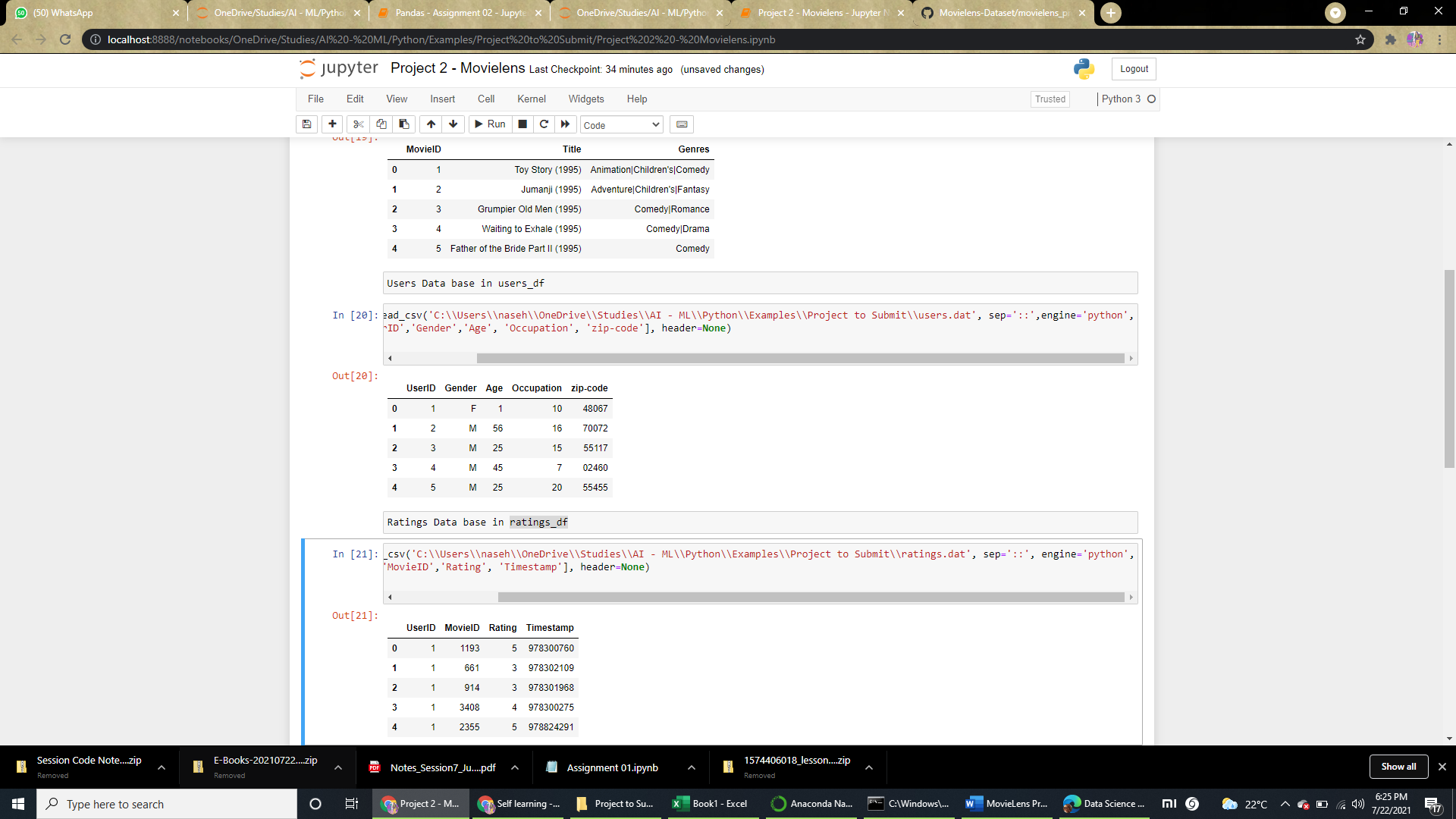
acc\_knn

In [ ]:

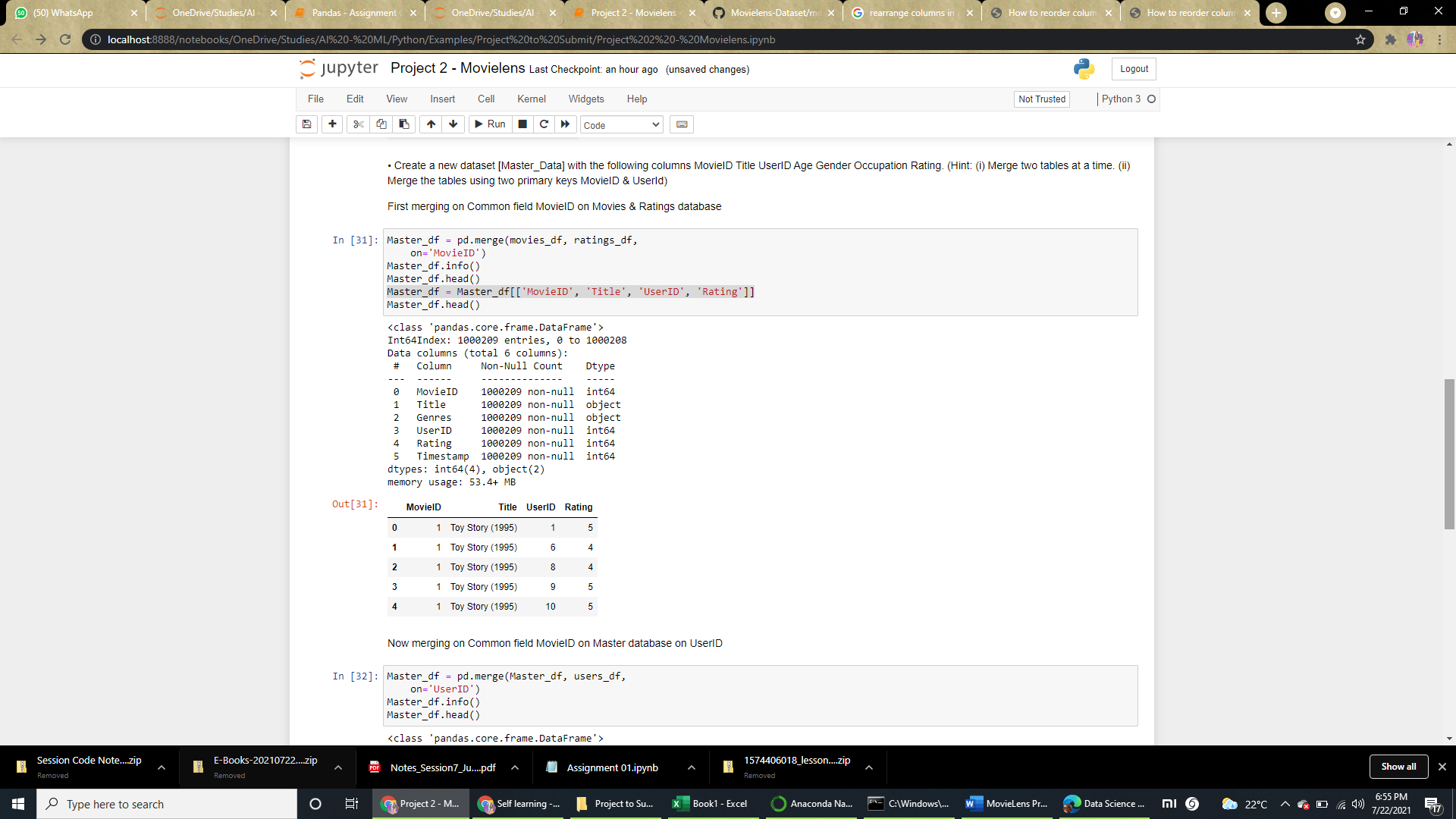
# ScreenShots:

Import 3 datasets

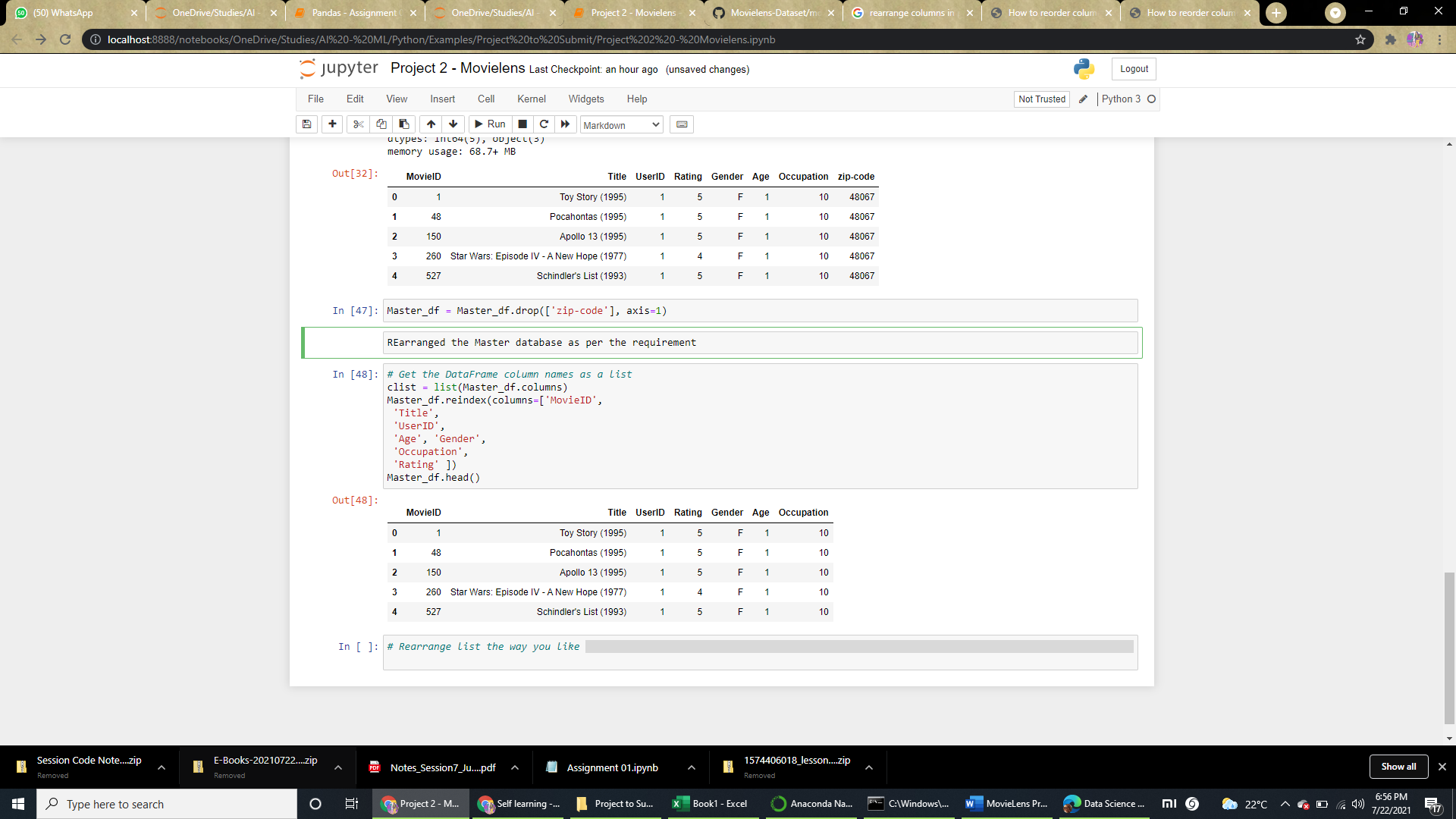




* Create a new dataset [Master\_Data] with the following columns MovieID Title UserID Age Gender Occupation Rating.

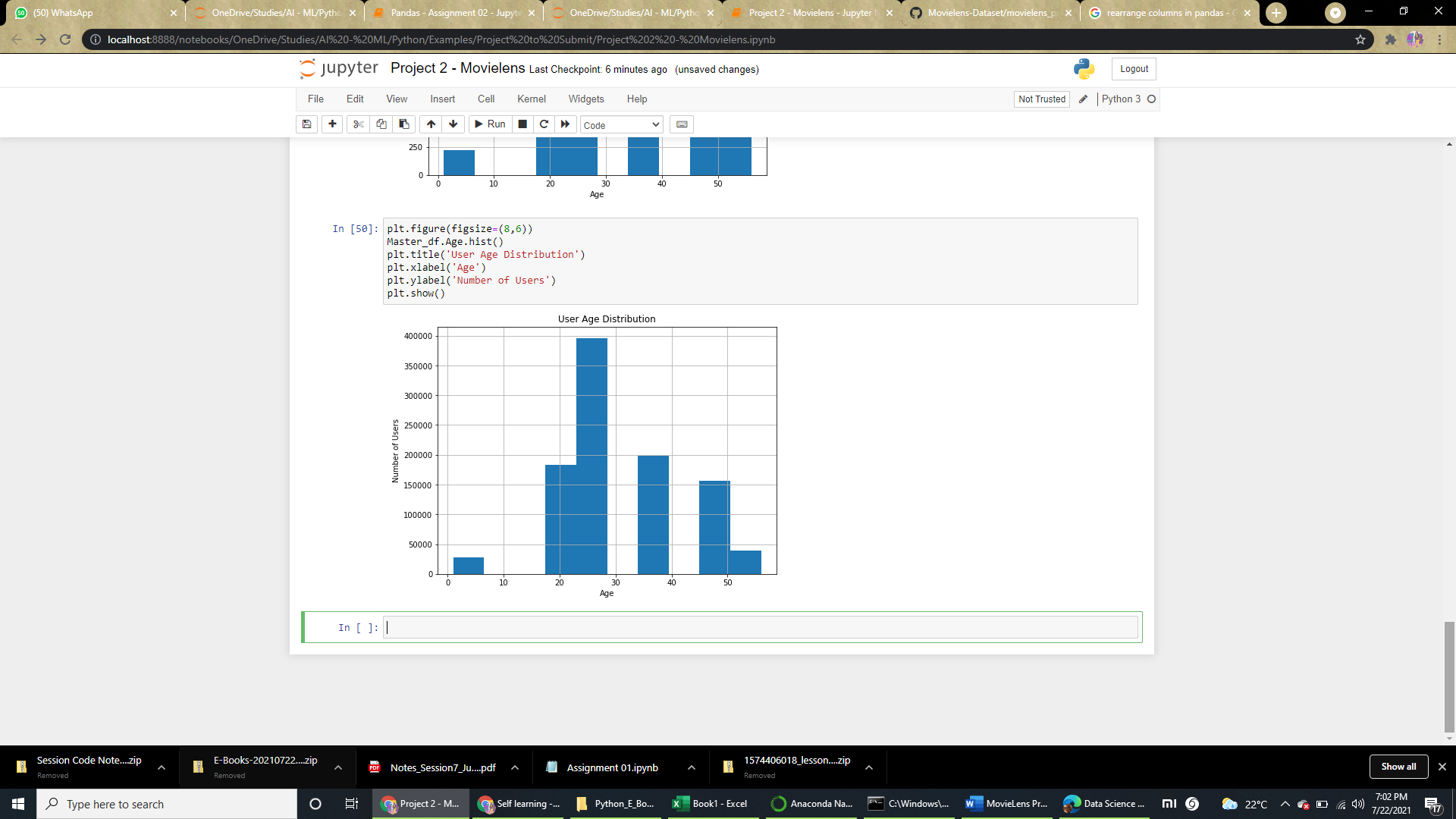


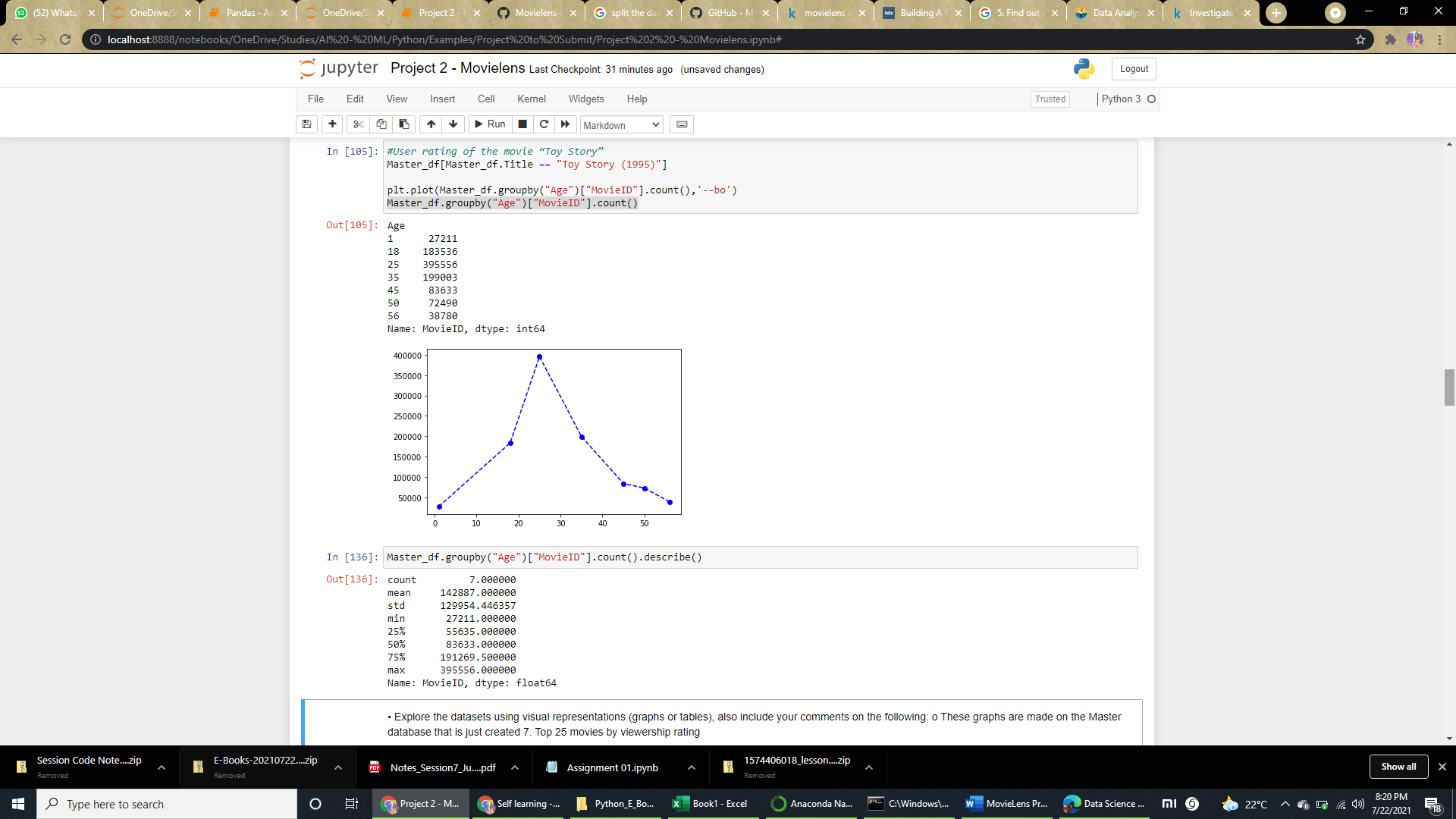


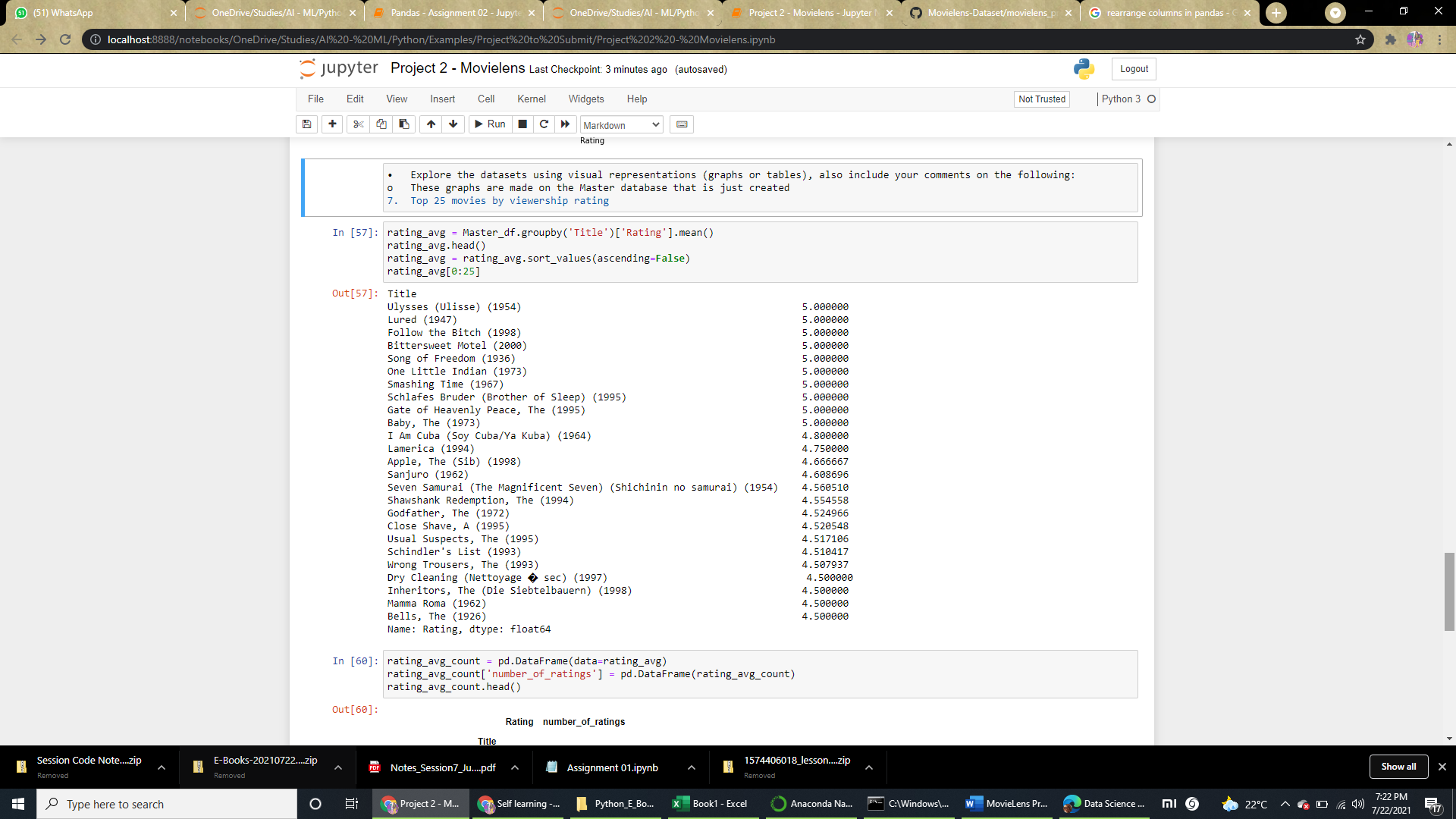


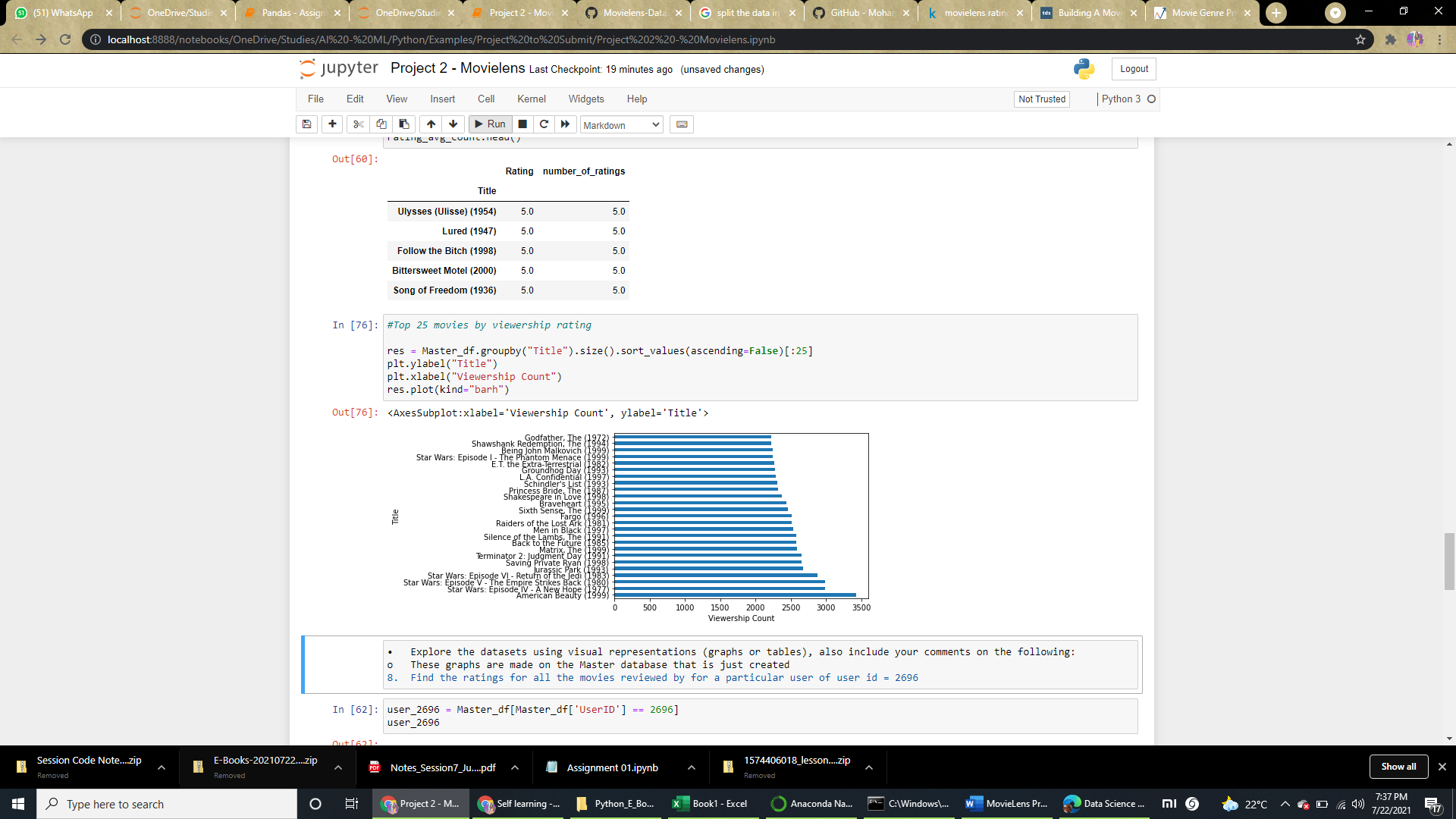
* Explore the datasets using visual representations (graphs or tables), also include your comments on the following:

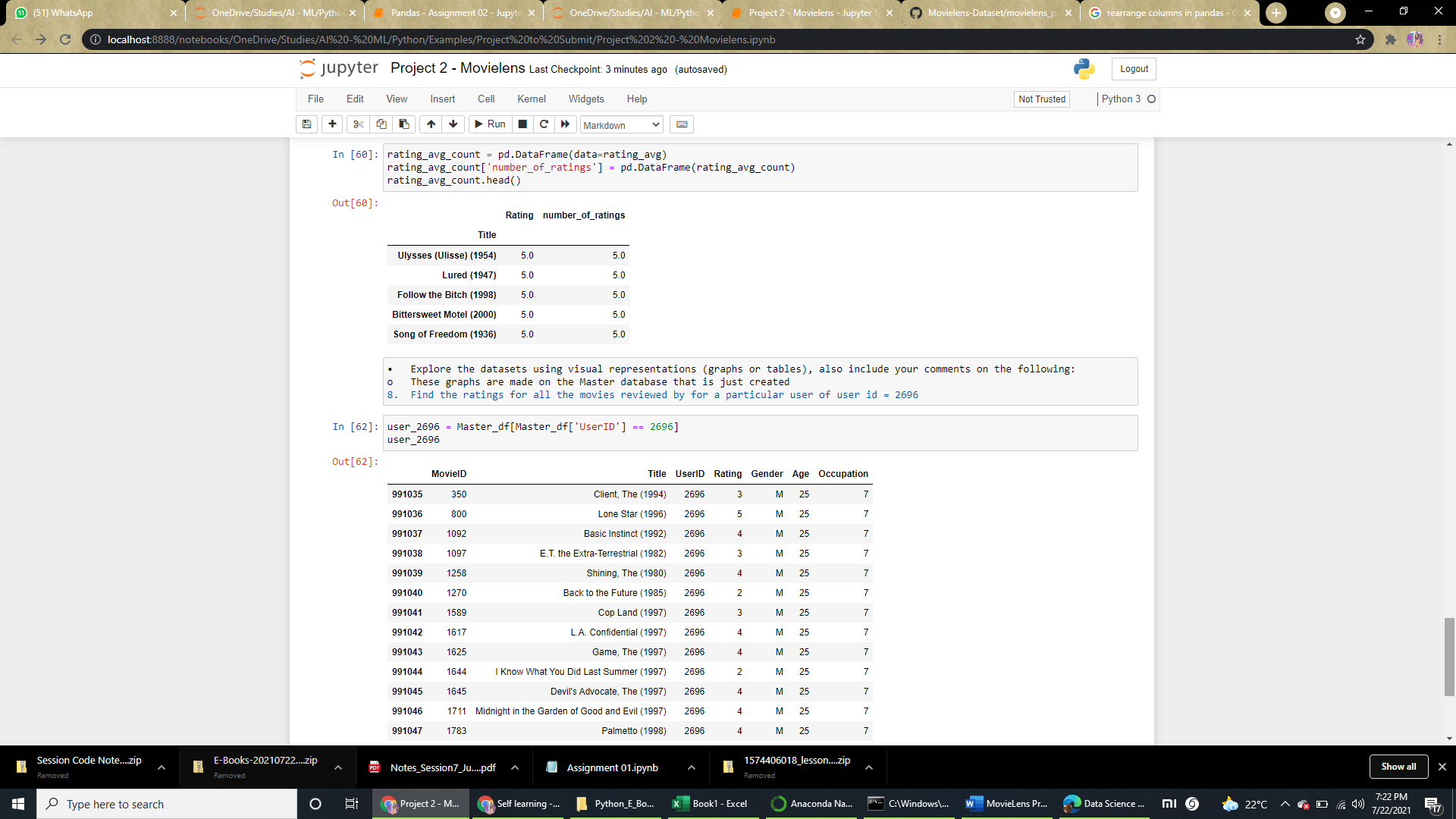
1. User Age Distribution
2. User rating of the movie “Toy Story”
3. Top 25 movies by viewership rating
4. Find the ratings for all the movies reviewed by for a particular user of user id = 2696





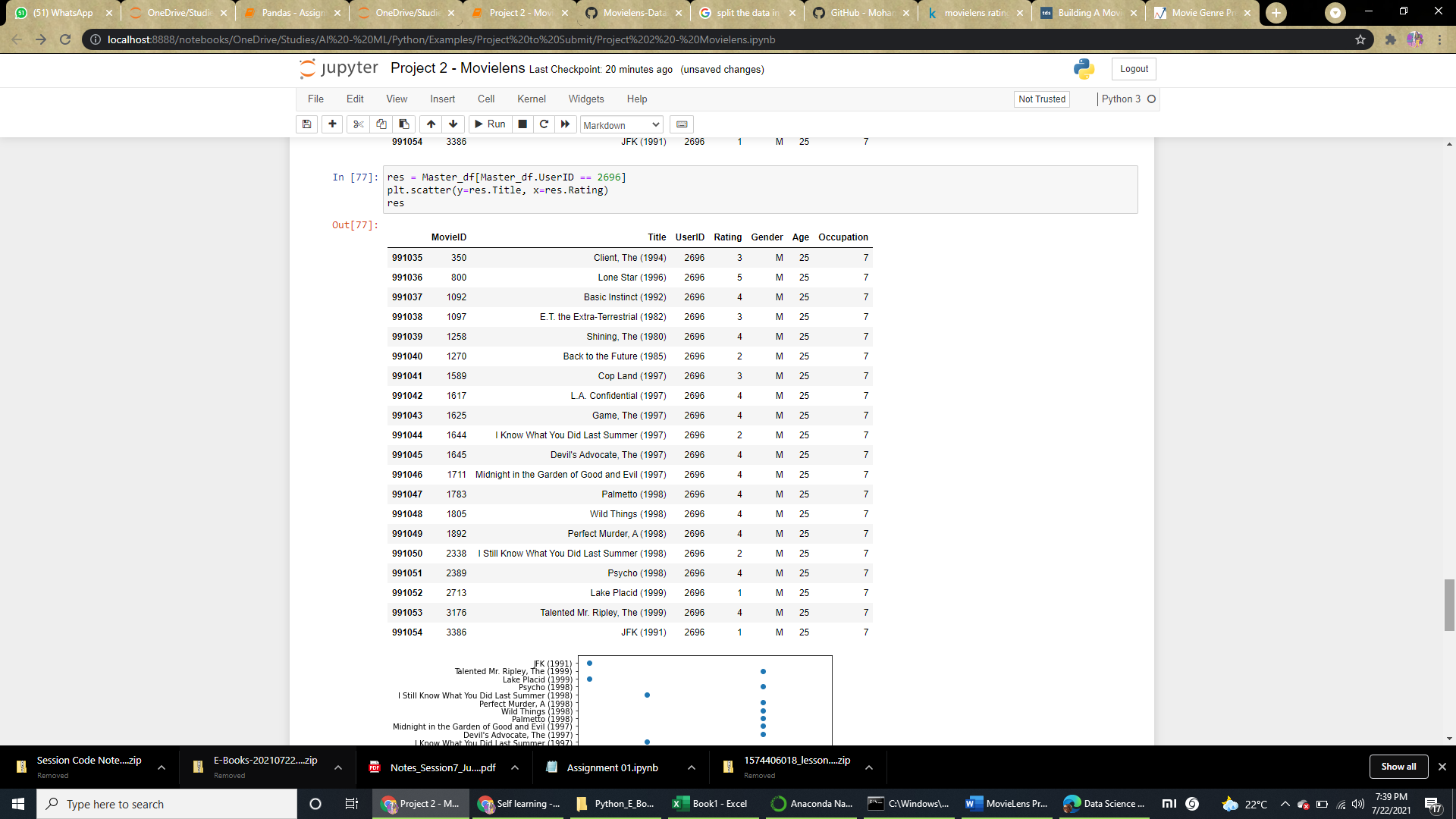


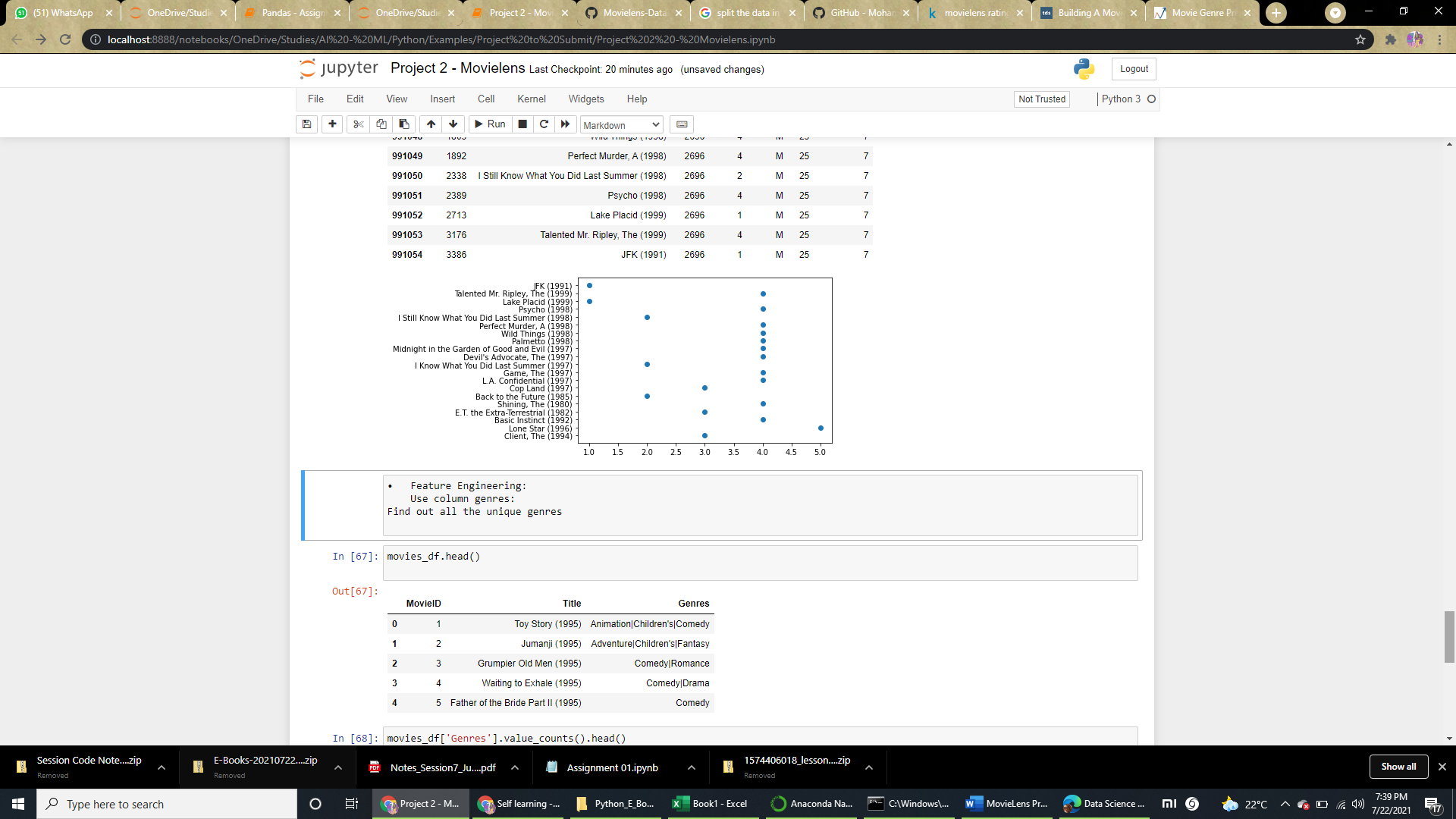


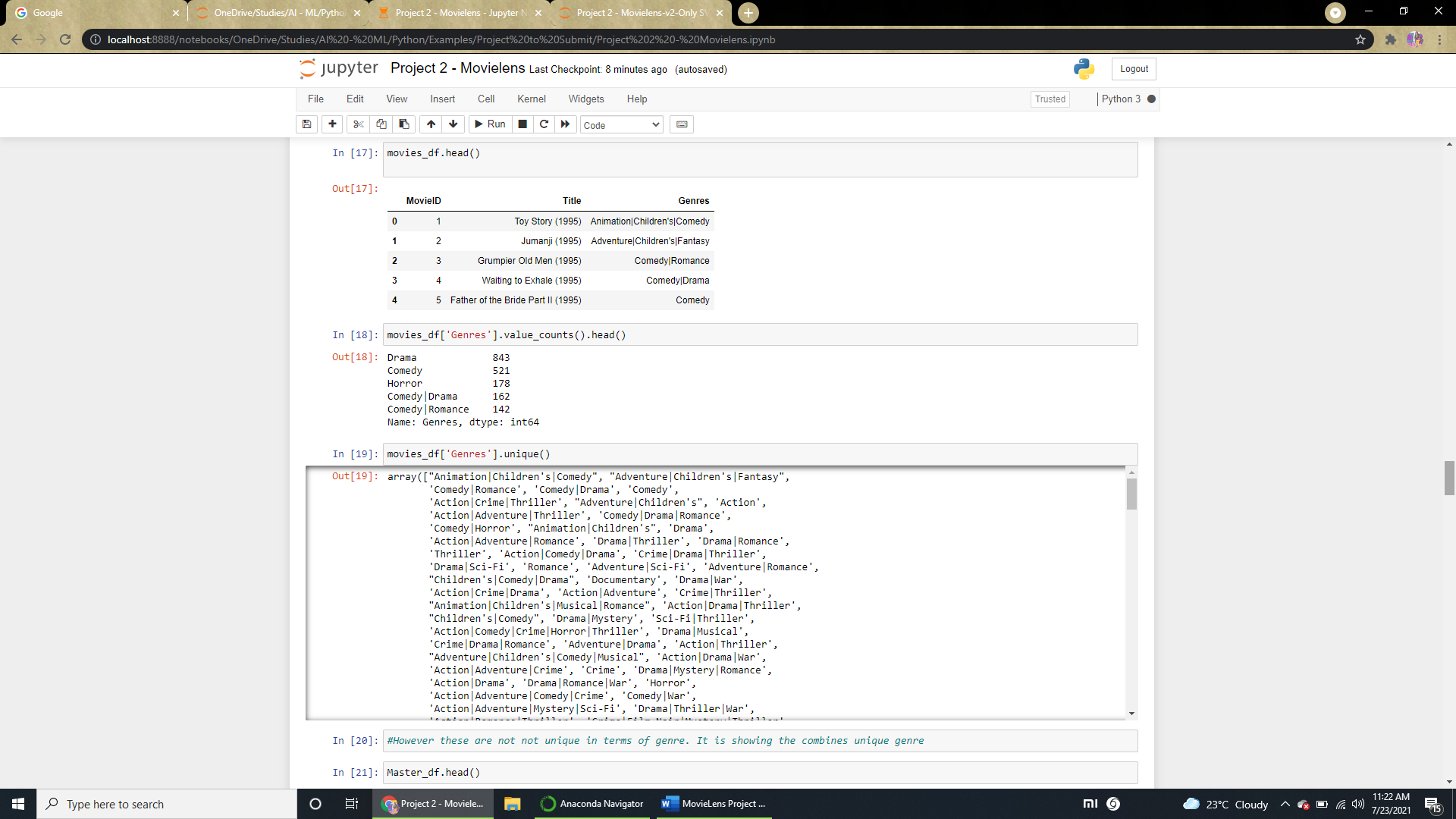


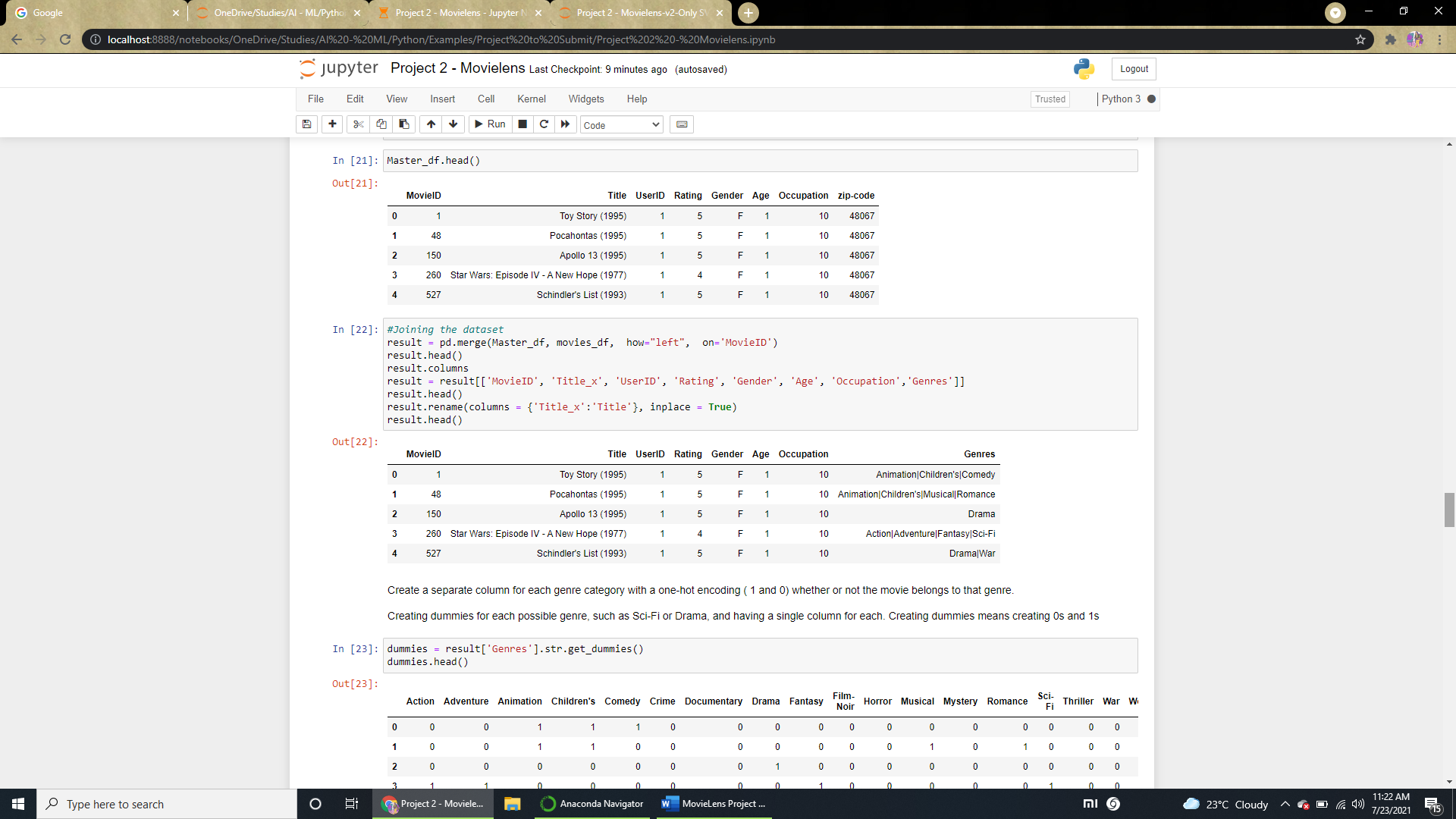
* Feature Engineering:

            Use column genres:

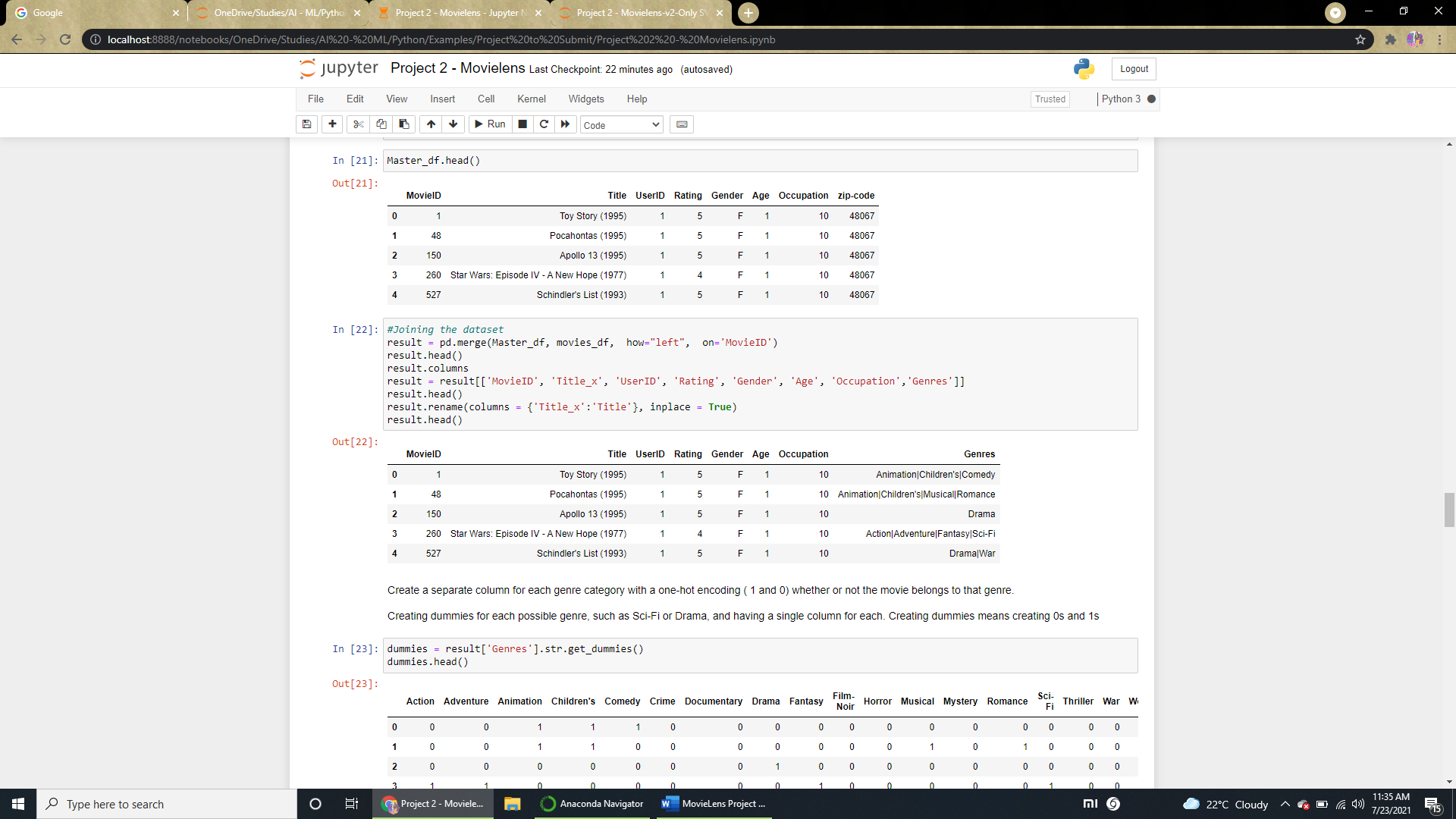


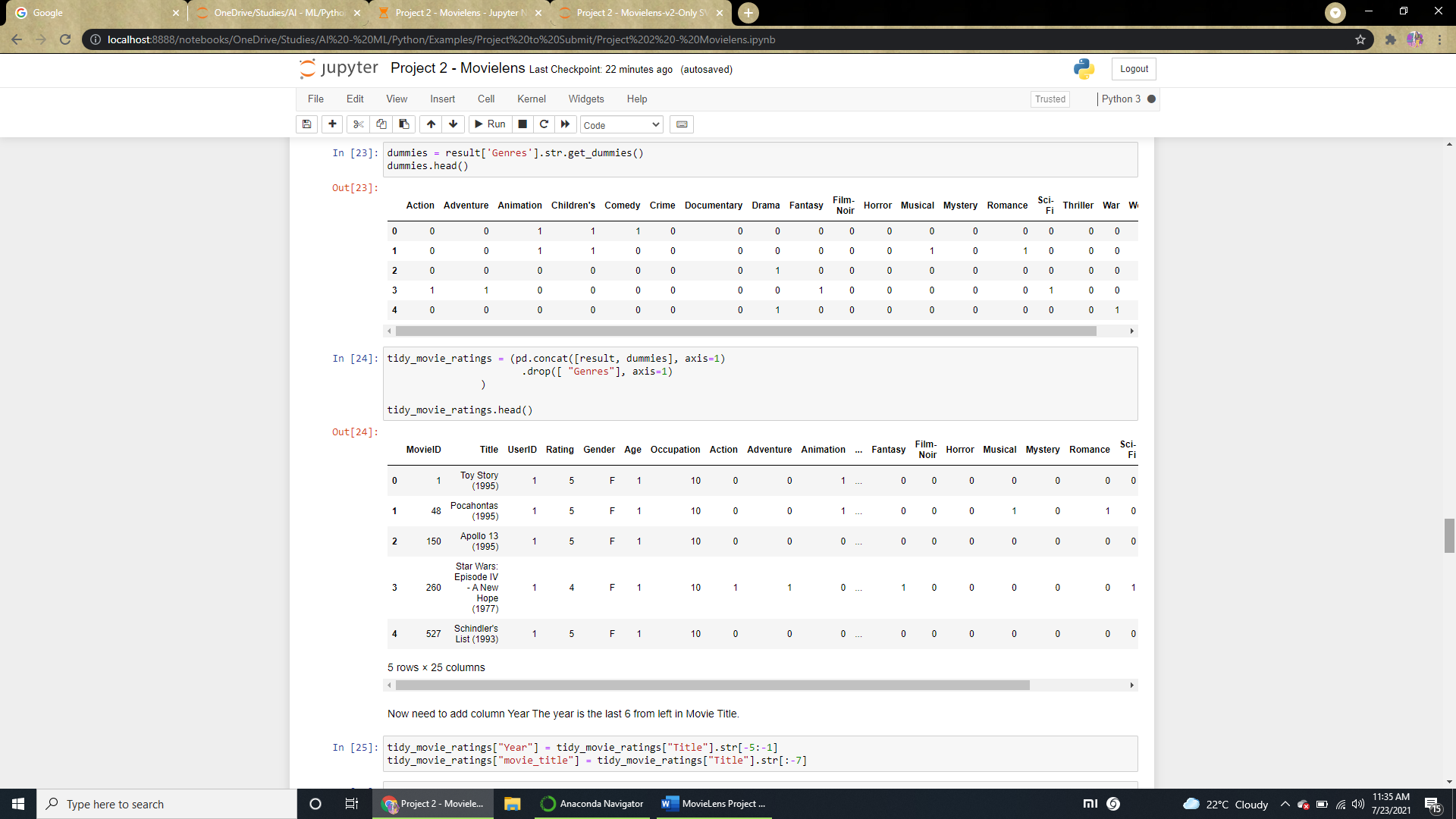


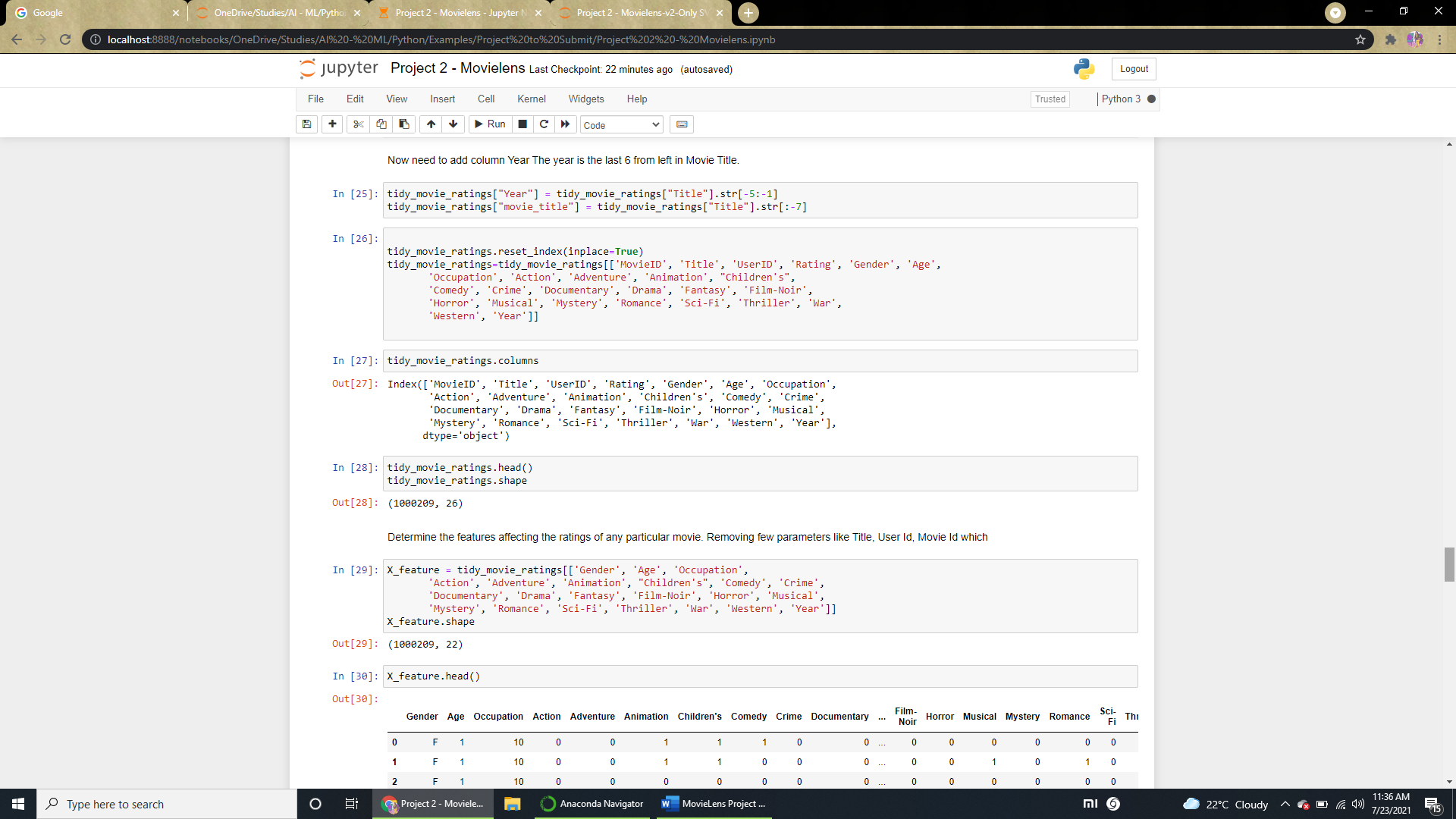




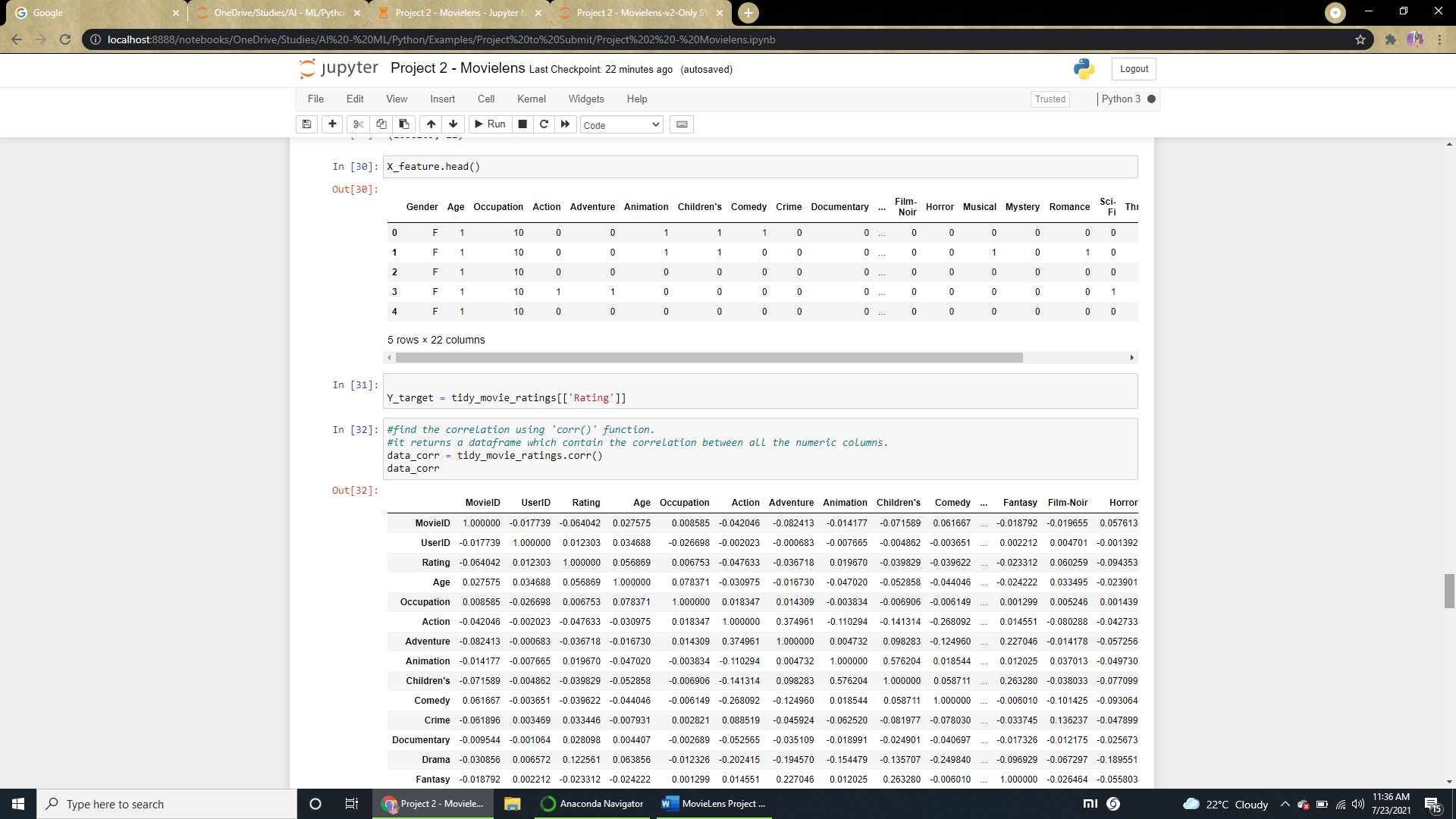
1. Create a separate column for each genre category with a one-hot encoding ( 1 and 0) whether or not the movie belongs to that genre.

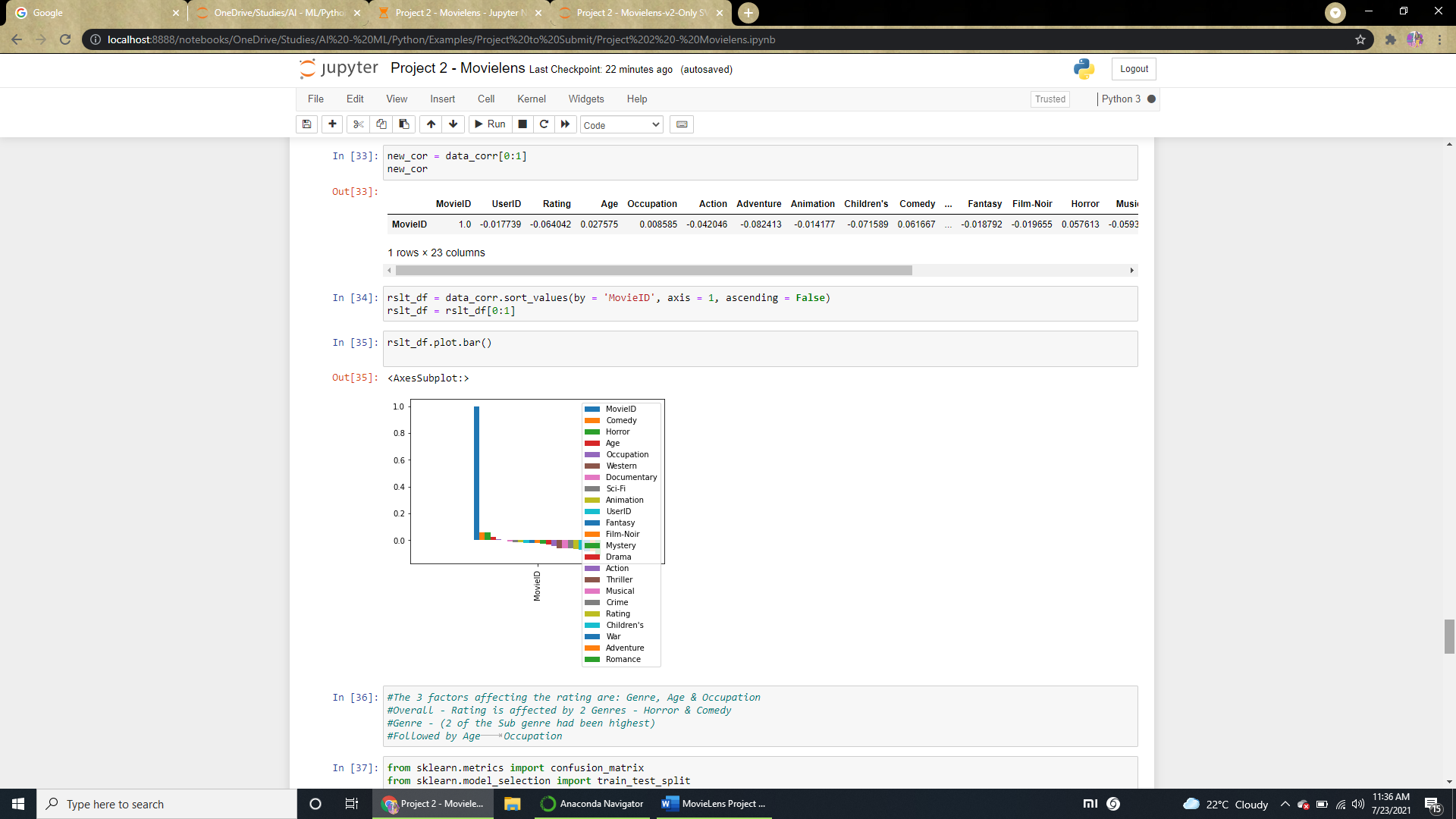




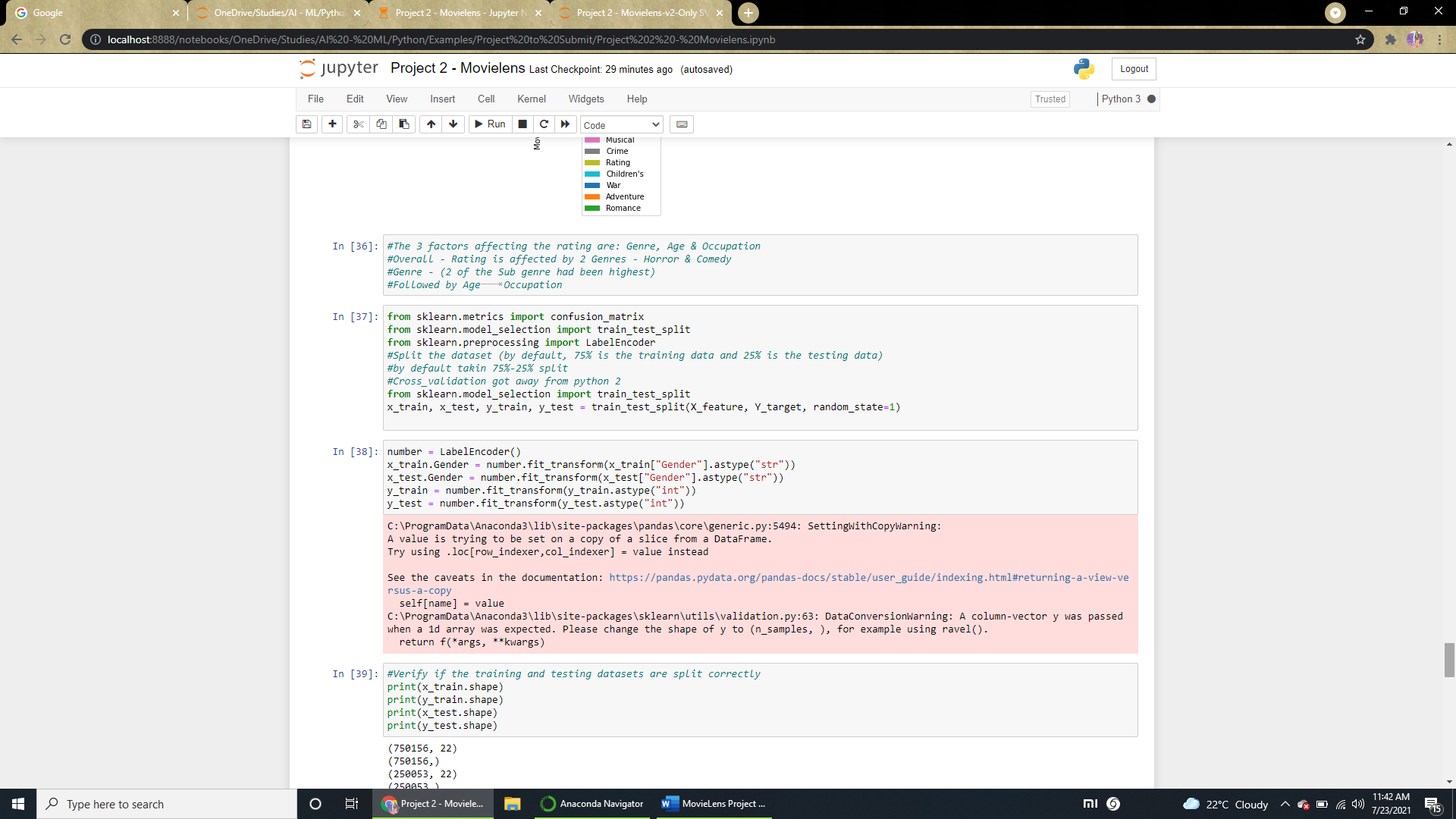


1. Determine the features affecting the ratings of any particular movie.

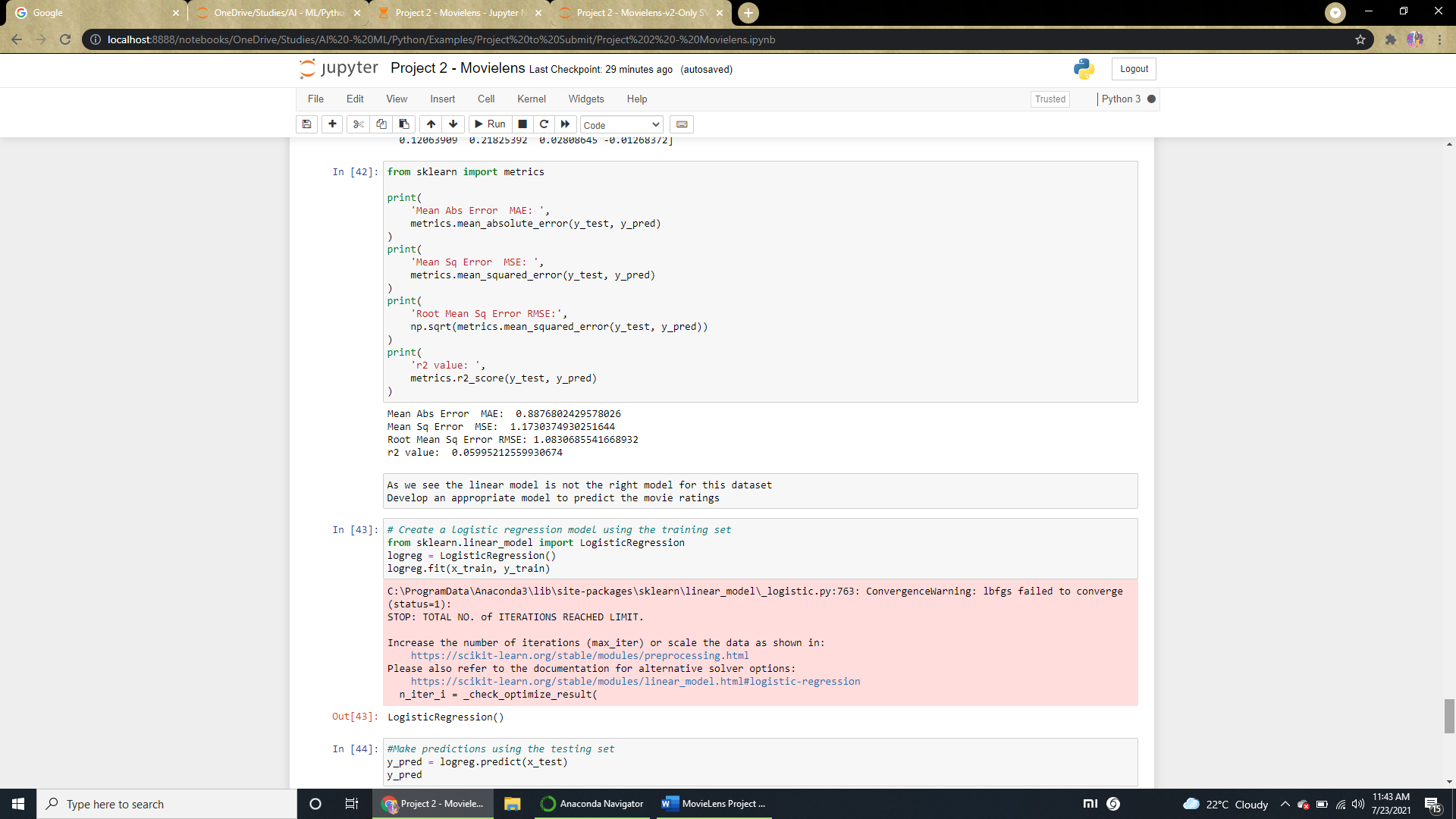


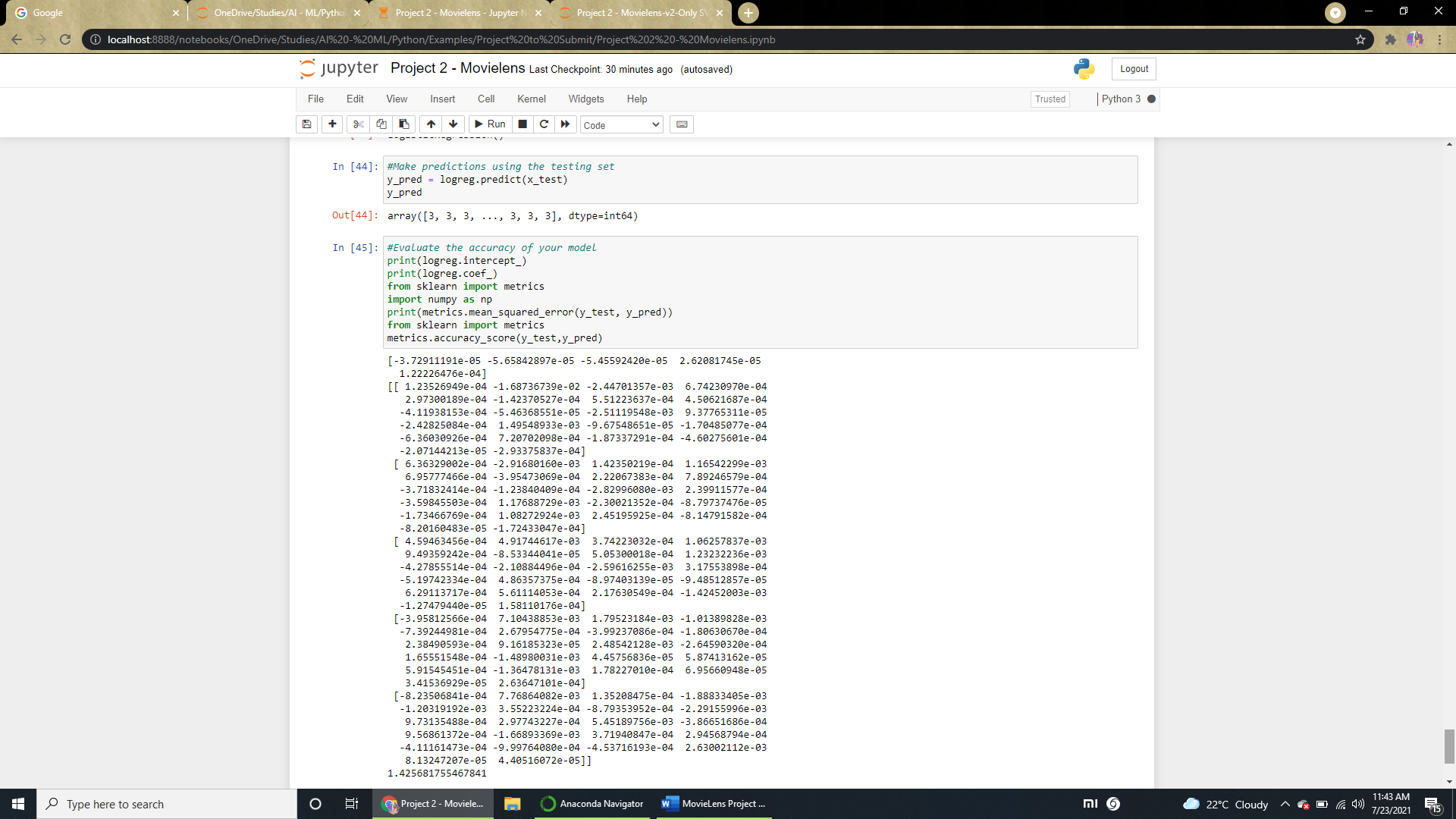


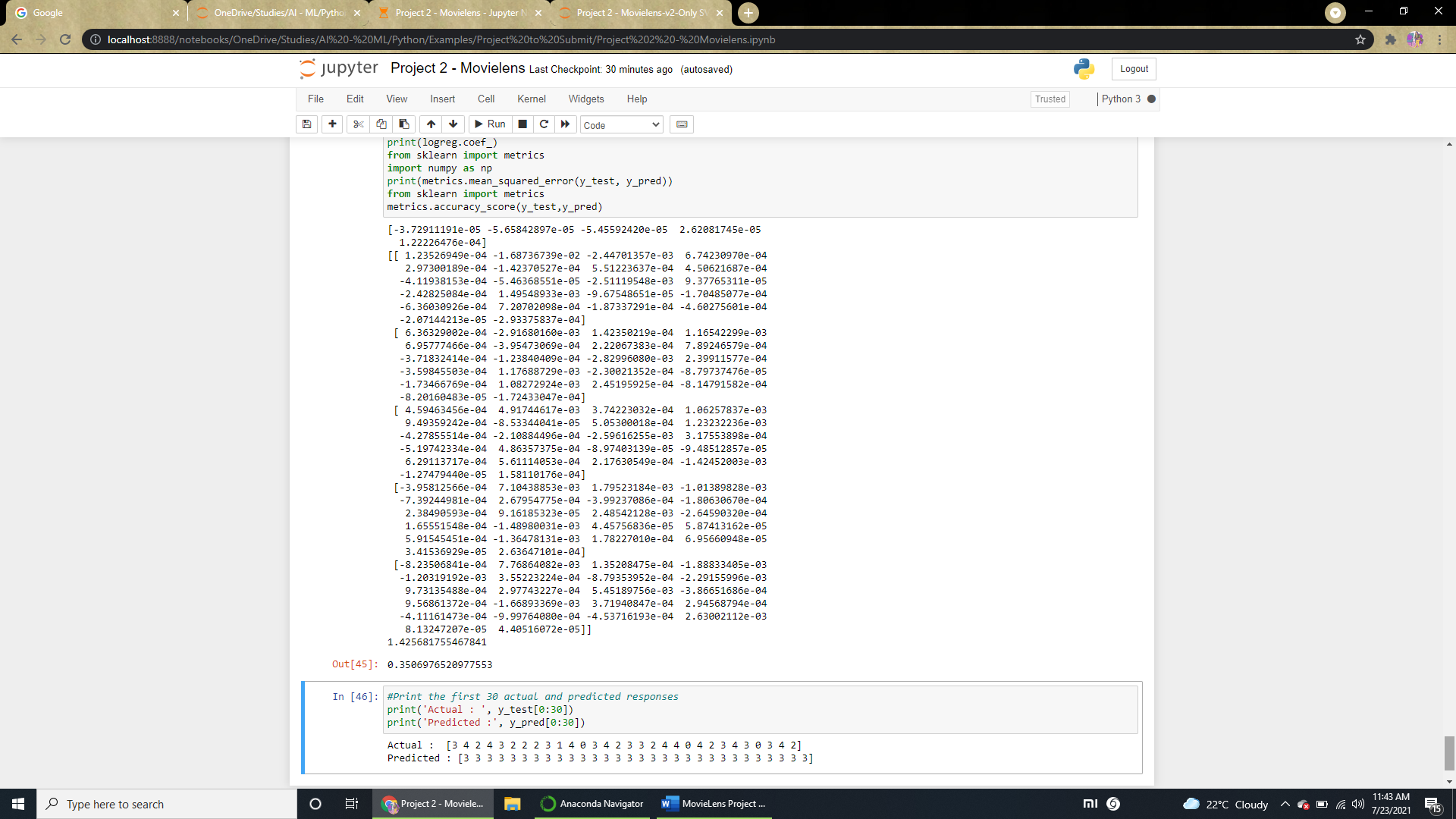
1. Develop an appropriate model to predict the movie ratings











# ScreenShots:

Phython File:

