Data Science with Python Project

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.

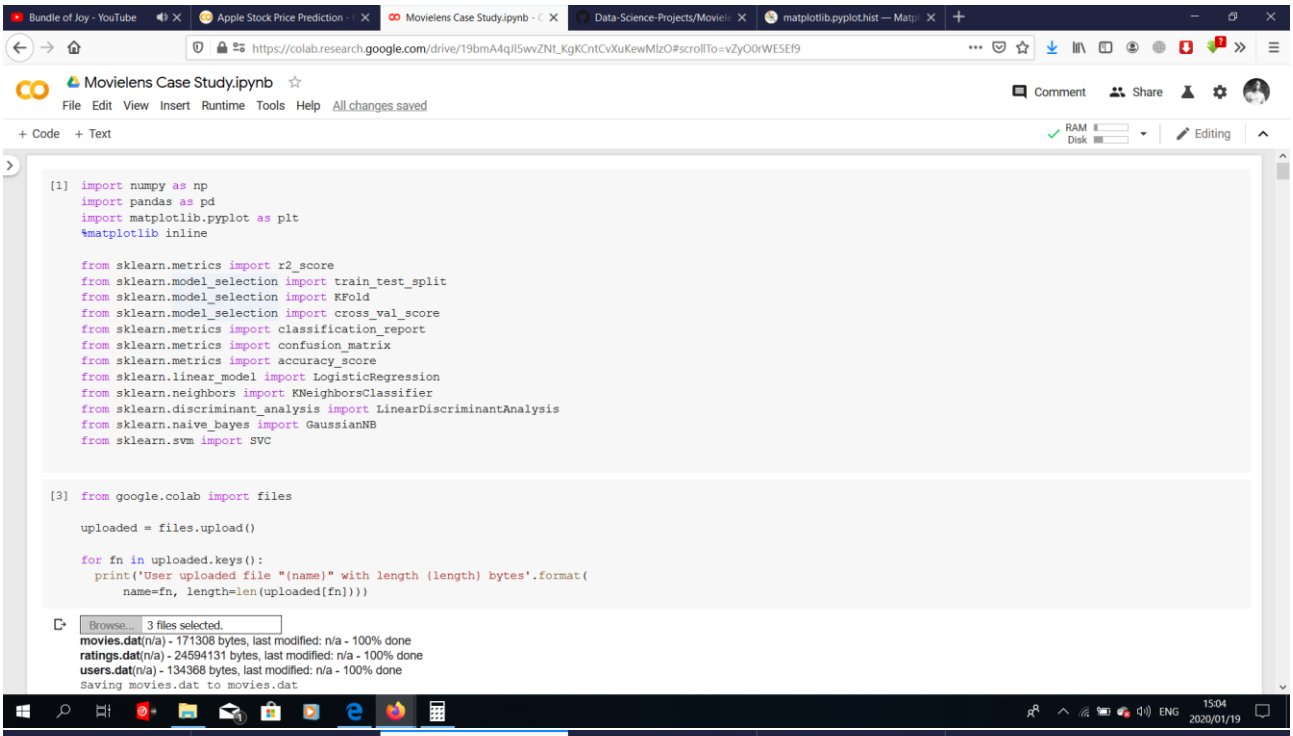
Analysis:

- 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

Screenshots



Movielens Case Study.ipynb

File Edit View Insert Runtime Tools Help All changes saved

Comment Share Editing

+ Code + Text

```
[1] import numpy as np
    import pandas as pd
    import matplotlib.pyplot as plt
    %matplotlib inline

    from sklearn.metrics import r2_score
    from sklearn.model_selection import train_test_split
    from sklearn.model_selection import KFold
    from sklearn.model_selection import cross_val_score
    from sklearn.metrics import classification_report
    from sklearn.metrics import confusion_matrix
    from sklearn.metrics import accuracy_score
    from sklearn.linear_model import LogisticRegression
    from sklearn.neighbors import KNeighborsClassifier
    from sklearn.discriminant_analysis import LinearDiscriminantAnalysis
    from sklearn.naive_bayes import GaussianNB
    from sklearn.svm import SVC
```

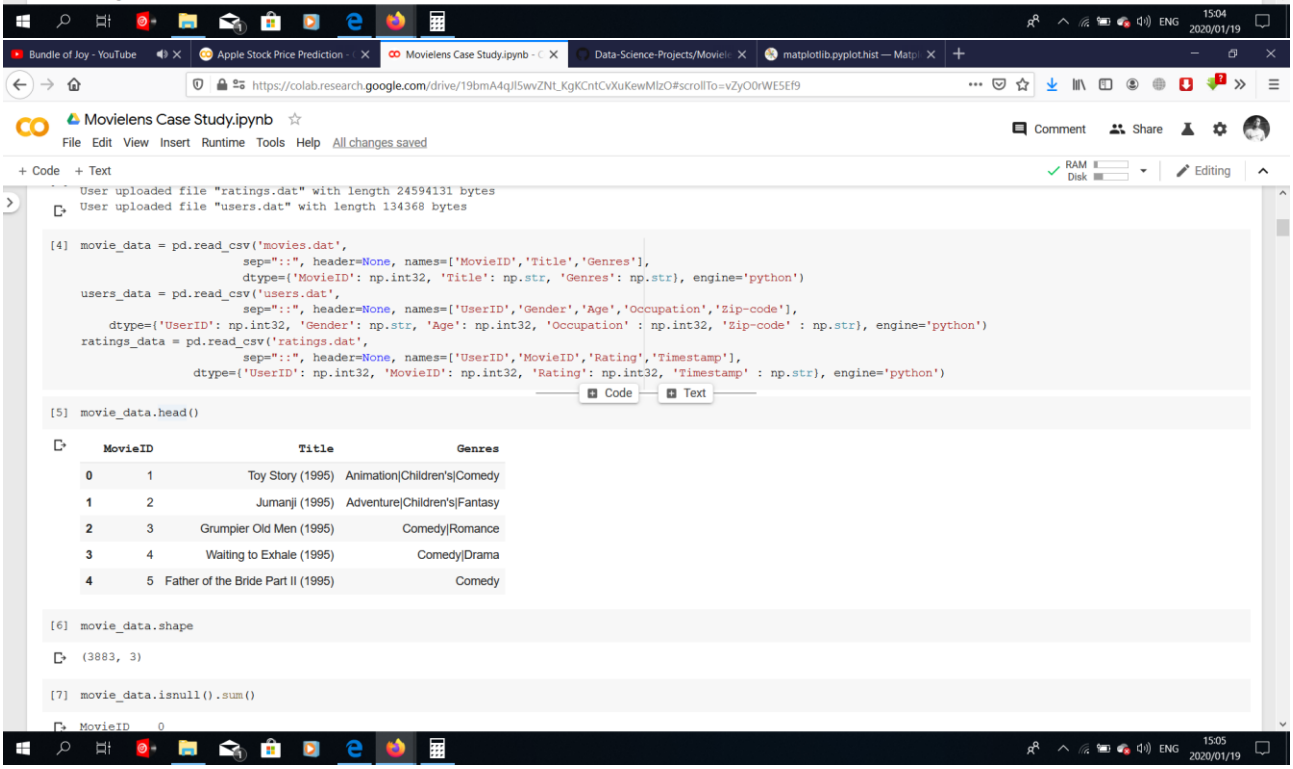
```
[3] from google.colab import files

    uploaded = files.upload()

    for fn in uploaded.keys():
      print('User uploaded file "{name}" with length {length} bytes'.format(
          name=fn, length=len(uploaded[fn])))
```

Browse... 3 files selected.

movies.dat(n/a) - 171308 bytes, last modified: n/a - 100% done
ratings.dat(n/a) - 24594131 bytes, last modified: n/a - 100% done
users.dat(n/a) - 134368 bytes, last modified: n/a - 100% done
Saving movies.dat to movies.dat



Movielens Case Study.ipynb

File Edit View Insert Runtime Tools Help All changes saved

Comment Share Editing

+ Code + Text

```
User uploaded file "ratings.dat" with length 24594131 bytes
User uploaded file "users.dat" with length 134368 bytes
```

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

+ Code + Text

```
[5] movie_data.head()
```

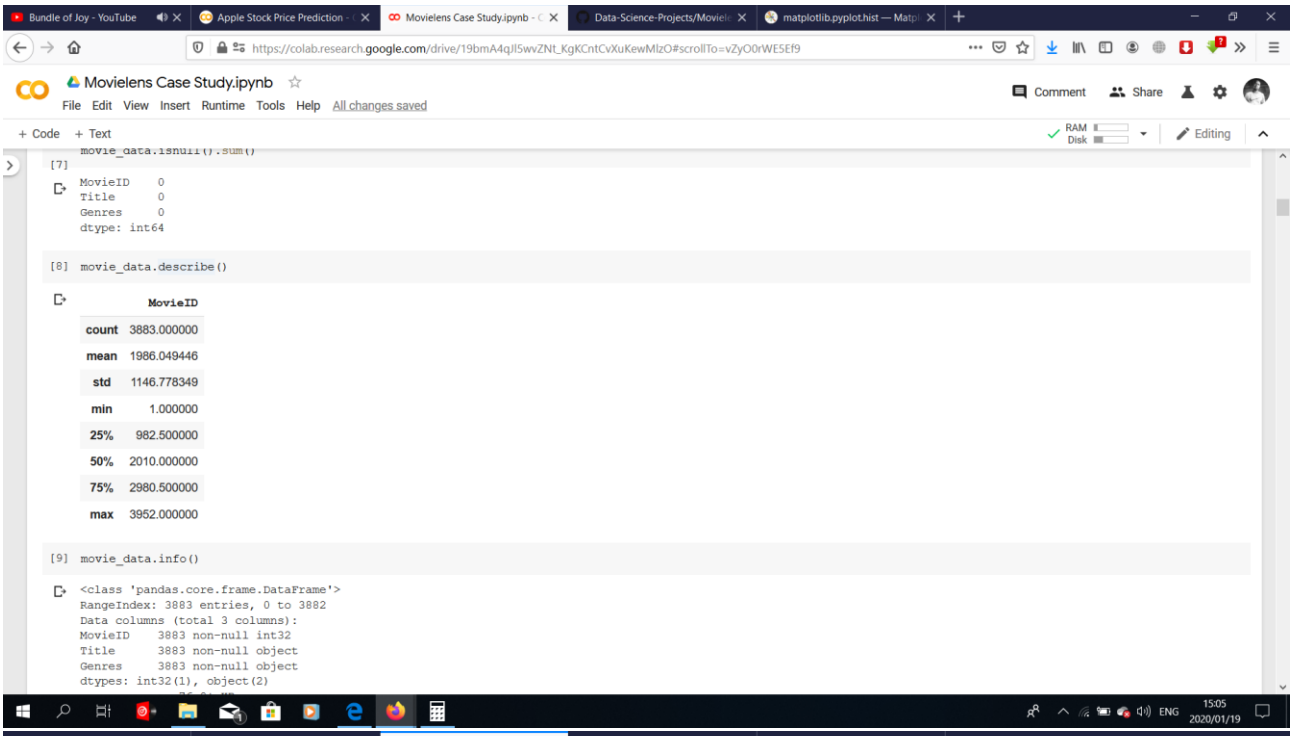
	MovieID	Title	Genres
0	1	Toy Story (1995)	Animation|Children's|Comedy
1	2	Jumanji (1995)	Adventure|Children's|Fantasy
2	3	Grumpier Old Men (1995)	Comedy|Romance
3	4	Waiting to Exhale (1995)	Comedy|Drama
4	5	Father of the Bride Part II (1995)	Comedy

```
[6] movie_data.shape
```

```
(3883, 3)
```

```
[7] movie_data.isnull().sum()
```

```
MovieID    0
```



Movielens Case Study.ipynb

File Edit View Insert Runtime Tools Help All changes saved

Comment Share Editing

+ Code + Text

```
movie_data.isnull().sum()
```

[7]

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

[8]

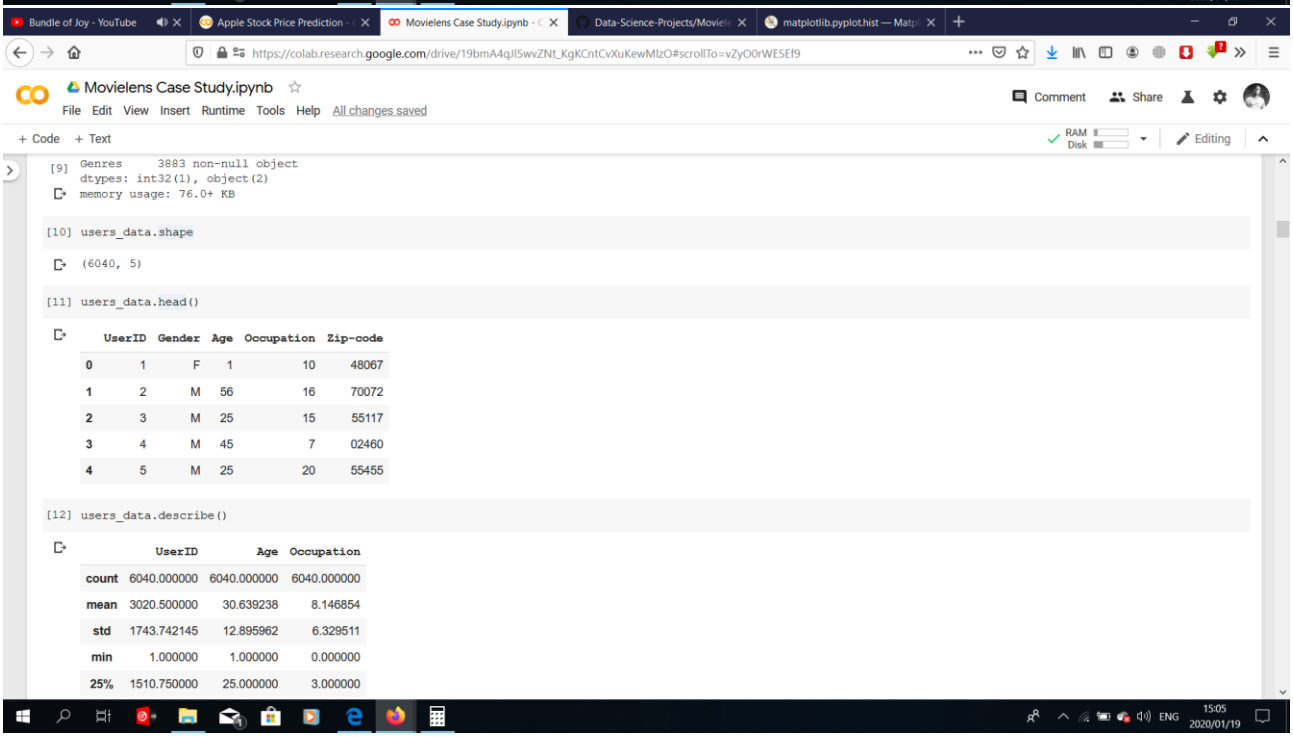
```
movie_data.describe()
```

	MovieID
count	3883.000000
mean	1986.049446
std	1146.778349
min	1.000000
25%	982.500000
50%	2010.000000
75%	2980.500000
max	3952.000000

[9]

```
movie_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 3883 entries, 0 to 3882
Data columns (total 3 columns):
MovieID    3883 non-null int32
Title      3883 non-null object
Genres     3883 non-null object
dtypes: int32(1), object(2)
```



Movielens Case Study.ipynb

File Edit View Insert Runtime Tools Help All changes saved

Comment Share Editing

+ Code + Text

[9]

```
Genres     3883 non-null object
dtypes: int32(1), object(2)
memory usage: 76.0+ KB
```

[10]

```
users_data.shape
```

```
(6040, 5)
```

[11]

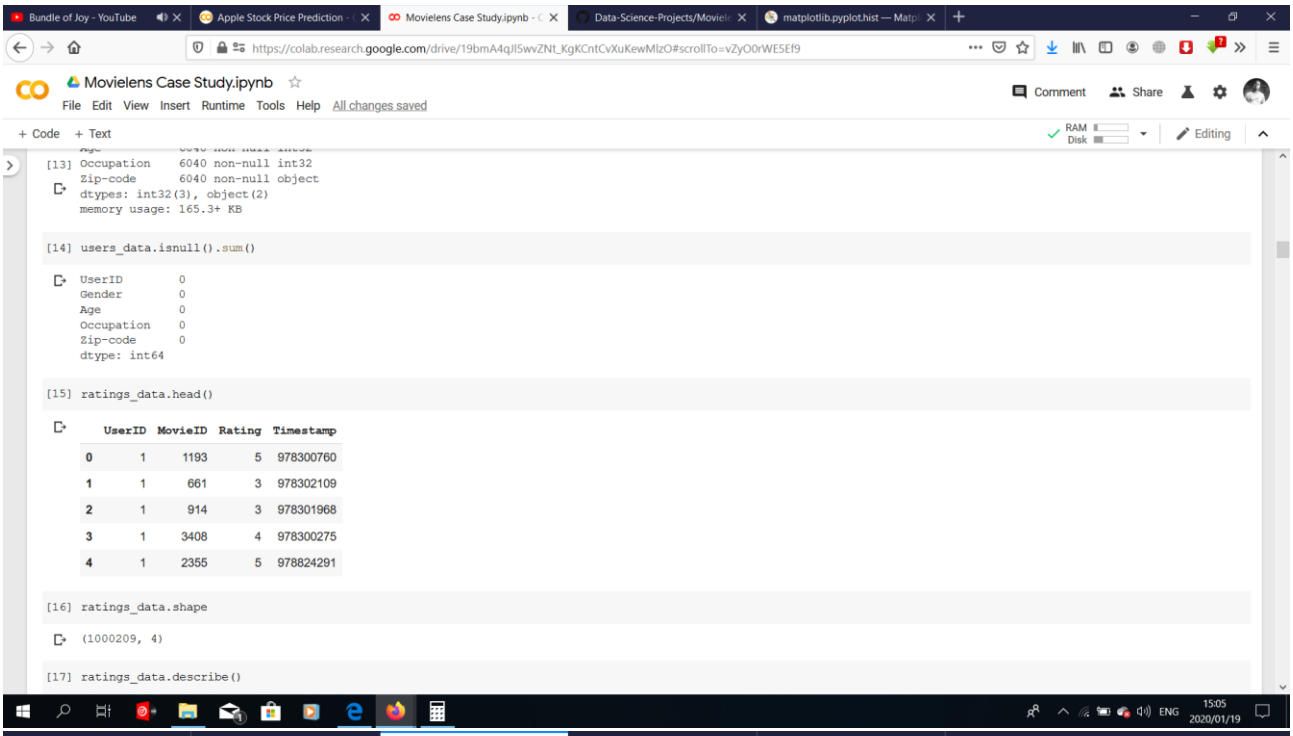
```
users_data.head()
```

	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

[12]

```
users_data.describe()
```

	UserID	Age	Occupation
count	6040.000000	6040.000000	6040.000000
mean	3020.500000	30.639238	8.146854
std	1743.742145	12.895962	6.329511
min	1.000000	1.000000	0.000000
25%	1510.750000	25.000000	3.000000



Movielens Case Study.ipynb

File Edit View Insert Runtime Tools Help All changes saved

Comment Share Editing

+ Code + Text

```
[13] Occupation    6040 non-null int32
     Zip-code      6040 non-null object
     dtypes: int32(3), object(2)
     memory usage: 165.3+ KB
```

```
[14] users_data.isnull().sum()
```

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

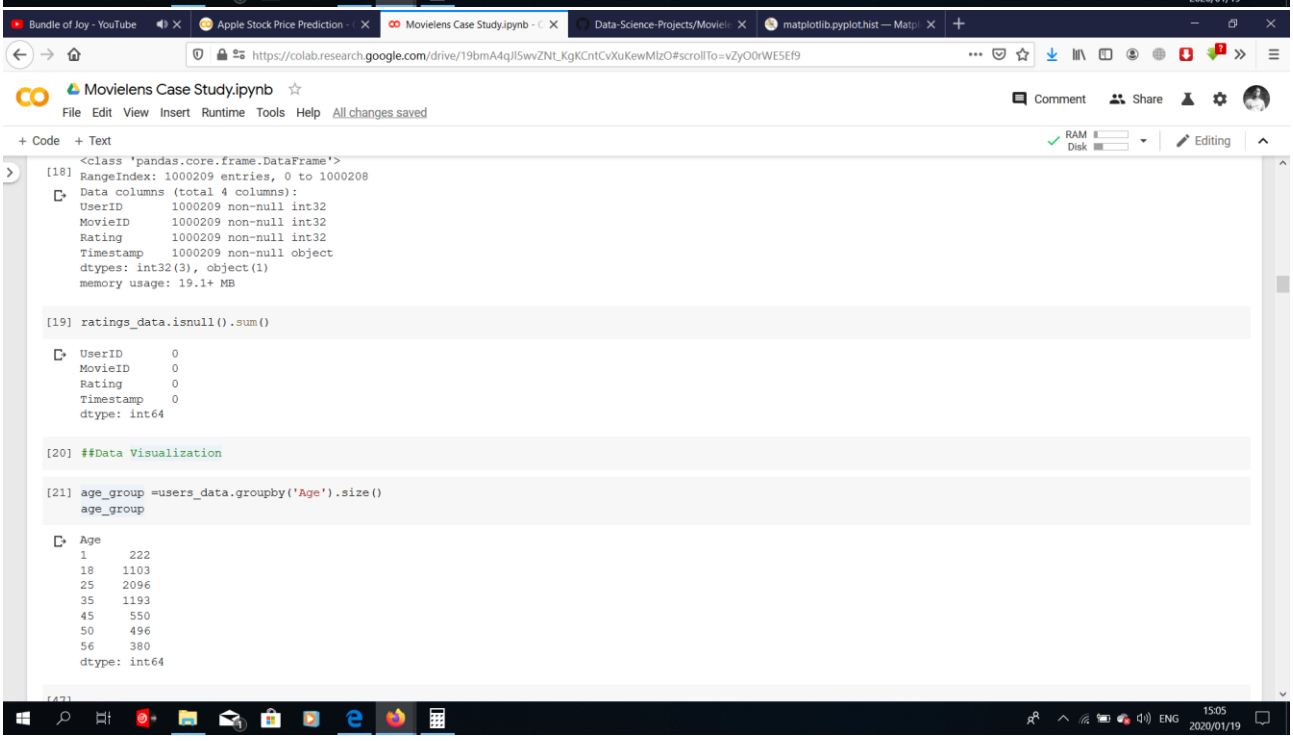
```
[15] ratings_data.head()
```

	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

```
[16] ratings_data.shape
```

```
(1000209, 4)
```

```
[17] ratings_data.describe()
```



Movielens Case Study.ipynb

File Edit View Insert Runtime Tools Help All changes saved

Comment Share Editing

+ Code + Text

```
[18] <class 'pandas.core.frame.DataFrame'>
     RangeIndex: 1000209 entries, 0 to 1000208
     Data columns (total 4 columns):
     UserID       1000209 non-null int32
     MovieID      1000209 non-null int32
     Rating       1000209 non-null int32
     Timestamp    1000209 non-null object
     dtypes: int32(3), object(1)
     memory usage: 19.1+ MB
```

```
[19] ratings_data.isnull().sum()
```

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

```
[20] ##Data Visualization
```

```
[21] age_group =users_data.groupby('Age').size()
     age_group
```

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

Bundle of Joy - YouTube | Apple Stock Price Prediction - C | Movielens Case Study.ipynb - C | Data-Science-Projects/Movielens | matplotlib.pyplot.hist — Matplotlib

https://colab.research.google.com/drive/19bmA4qJl5wvZNt_KgKCntCvXuKewMlzO#scrollTo=vZyO0rWE5Ef9

Movielens Case Study.ipynb

File Edit View Insert Runtime Tools Help All changes saved

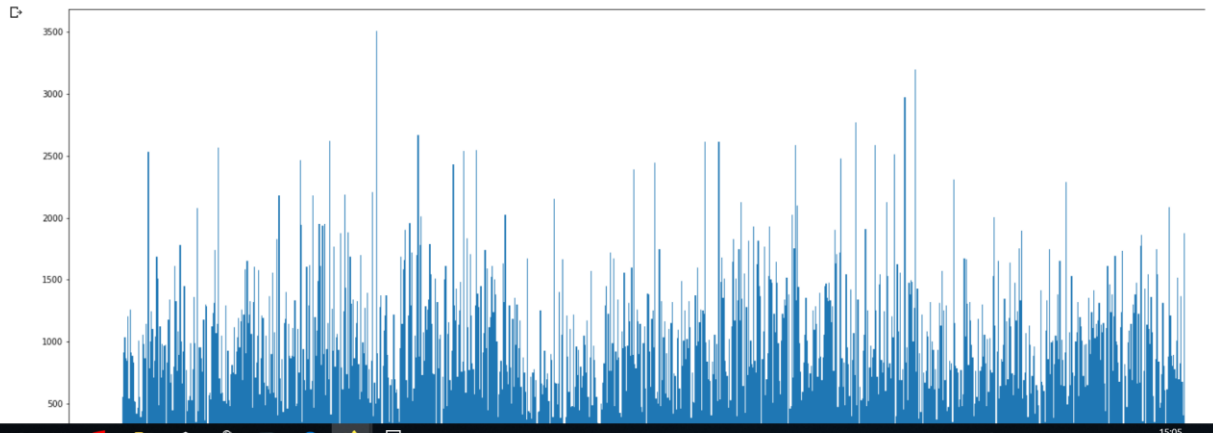
Comment Share

+ Code + Text

RAM Disk Editing

```
[25] 9      106
     10     401
     dtype: int64
```

```
[26] plt.figure(figsize=(25,10))
     plt.hist(x=[ratings_data.UserID], bins=1000)
     plt.show()
```



ENG 15:05 2020/01/19

Bundle of Joy - YouTube | Apple Stock Price Prediction - C | Movielens Case Study.ipynb - C | Data-Science-Projects/Movielens | matplotlib.pyplot.hist — Matplotlib

https://colab.research.google.com/drive/19bmA4qJl5wvZNt_KgKCntCvXuKewMlzO#scrollTo=vZyO0rWE5Ef9

Movielens Case Study.ipynb

File Edit View Insert Runtime Tools Help All changes saved

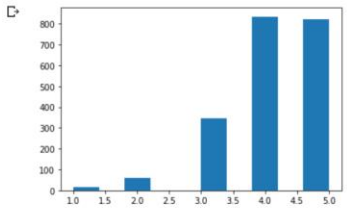
Comment Share

+ Code + Text

RAM Disk Editing

```
[29]  4      4
      5      5
```

```
[30] plt.hist(x=toystory_data['Rating'])
     plt.show()
```




```
[31] viewership =pd.merge(ratings_data ,users_data ,how='left' ,left_on=['UserID'] ,right_on=['UserID'])
```

```
[32] viewership.shape

     (1000209, 8)
```

```
[33] ratings_data.shape

     (1000209, 4)
```

```
[34] viewership.head()
```

ENG 15:05 2020/01/19

Movielens Case Study.ipynb

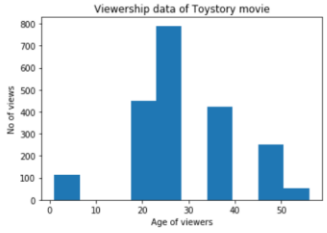
File Edit View Insert Runtime Tools Help All changes saved

Comment Share Editing

+ Code + Text

```
dtype: int64
```

```
[44] plt.hist(x=viewership_of_toystory['Age'] )
     plt.xlabel("Age of viewers")
     plt.ylabel("No of views")
     plt.title("Viewership data of Toystory movie")
     plt.show()
```




```
[48] movie_rating =ratings_data.groupby(['MovieID'] ,as_index =False)
     average_movie_ratings = movie_rating.agg({'Rating':'mean'})
     top_25_movies = average_movie_ratings.sort_values('Rating', ascending=False).head(25)
     top_25_movies
```

	MovieID	Rating
926	989	5.000000

Movielens Case Study.ipynb

File Edit View Insert Runtime Tools Help All changes saved

Comment Share Editing

+ Code + Text

[48]

2287	2480	4.500000
425	439	4.500000

```
[49] #The below list shows top 25 movies by viewership data
     pd.merge(top_25_movies, movie_data, how='left', left_on=['MovieID'], right_on=['MovieID'])
```

	MovieID	Rating	Title	Genres
0	989	5.000000	Schlafes Bruder (Brother of Sleep) (1995)	Drama
1	3881	5.000000	Bittersweet Motel (2000)	Documentary
2	1830	5.000000	Follow the Bitch (1998)	Comedy
3	3382	5.000000	Song of Freedom (1936)	Drama
4	787	5.000000	Gate of Heavenly Peace, The (1995)	Documentary
5	3280	5.000000	Baby, The (1973)	Horror
6	3607	5.000000	One Little Indian (1973)	Comedy|Drama|Western
7	3233	5.000000	Smashing Time (1967)	Comedy
8	3172	5.000000	Ulysses (Ulisse) (1954)	Adventure
9	3656	5.000000	Lured (1947)	Crime
10	3245	4.800000	I Am Cuba (Soy Cuba/Ya Kuba) (1964)	Drama
11	53	4.750000	Lamerica (1994)	Drama
12	2503	4.666667	Apple, The (Sib) (1998)	Drama
13	2905	4.608696	Sanjuro (1962)	Action|Adventure
14	2019	4.560510	Seven Samurai (The Magnificent Seven) (Shichin	Action|Drama

Bundle of Joy - YouTube | Apple Stock Price Prediction - C | Movielens Case Study.ipynb - C | Data-Science-Projects/Movielens | matplotlib.pyplot.hist — Matplotlib

https://colab.research.google.com/drive/19bmA4qIl5wvZNt_KgKCntCvXuKewMlzO#scrollTo=vZyO0rWE5Ef9

Movielens Case Study.ipynb

File Edit View Insert Runtime Tools Help All changes saved

Comment Share

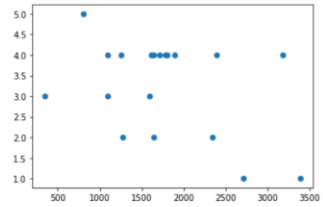
+ Code + Text

RAM Disk Editing

```
[50] user_rating_data =ratings_data[ratings_data['UserID']==2696]
     user_rating_data.head()
```

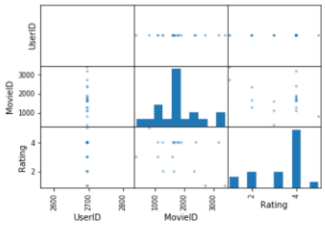
	UserID	MovieID	Rating	Timestamp
440667	2696	1258	4	973308710
440668	2696	1270	2	973308676
440669	2696	1617	4	973308842
440670	2696	1625	4	973308842
440671	2696	1644	2	973308920

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




```
[53] from pandas.plotting import scatter_matrix
     scatter_matrix(user_rating_data)
     plt.show()
```

```
/usr/local/lib/python3.6/dist-packages/pandas/plotting/_matplotlib/misc.py:71: UserWarning: Attempting to set identical left == right == 2696.0 results in singular tr
  ax.set_xlim(boundaries_list[i])
/usr/local/lib/python3.6/dist-packages/pandas/plotting/_matplotlib/misc.py:81: UserWarning: Attempting to set identical bottom == top == 2696.0 results in singular tr
  ax.set_ylim(boundaries_list[i])
/usr/local/lib/python3.6/dist-packages/pandas/plotting/_matplotlib/misc.py:80: UserWarning: Attempting to set identical left == right == 2696.0 results in singular tr
  ax.set_xlim(boundaries_list[j])
```




```
[54] few_viewership =viewership.head(500)
     few_viewership.shape
```

```
(500, 8)
```

```
[58] #Preprocess data
     from sklearn.preprocessing import LabelEncoder
```

ENG 15:06 2020/01/19

Movielens Case Study.ipynb

File Edit View Insert Runtime Tools Help All changes saved

```
[58]
#Preprocess data
from sklearn.preprocessing import LabelEncoder
le=LabelEncoder()

le.fit(few_viewership['Age'])
x_age =le.transform(few_viewership['Age'])
x_age
```

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

```
[59]
le.fit(few_viewership['Occupation'])
x_occ =le.transform(few_viewership['Occupation'])
x_occ
```

Movielens Case Study.ipynb

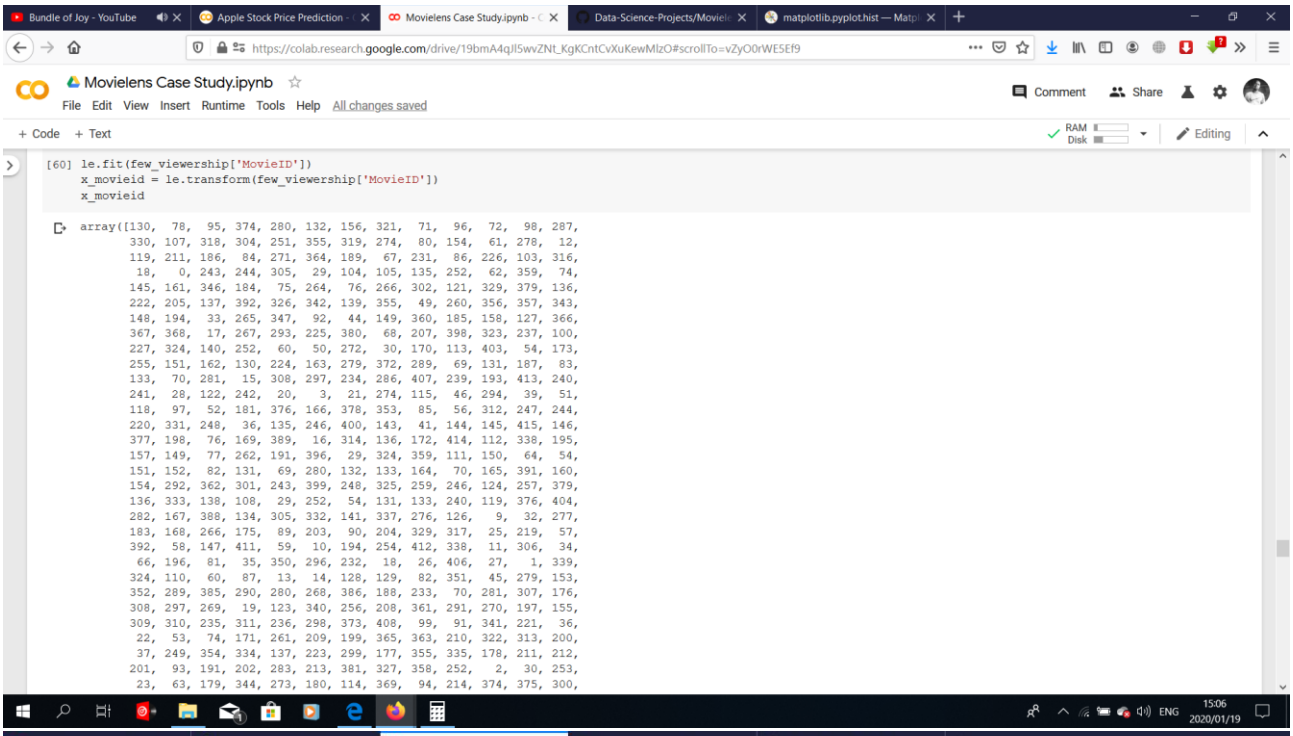
File Edit View Insert Runtime Tools Help All changes saved

```
1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 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, 3, 3, 3,
3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3])
```

```
[59]
le.fit(few_viewership['Occupation'])
x_occ =le.transform(few_viewership['Occupation'])
x_occ
```


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

```
[60]
le.fit(few_viewership['MovieID'])
x_movieid = le.transform(few_viewership['MovieID'])
```



Movielens Case Study.ipynb

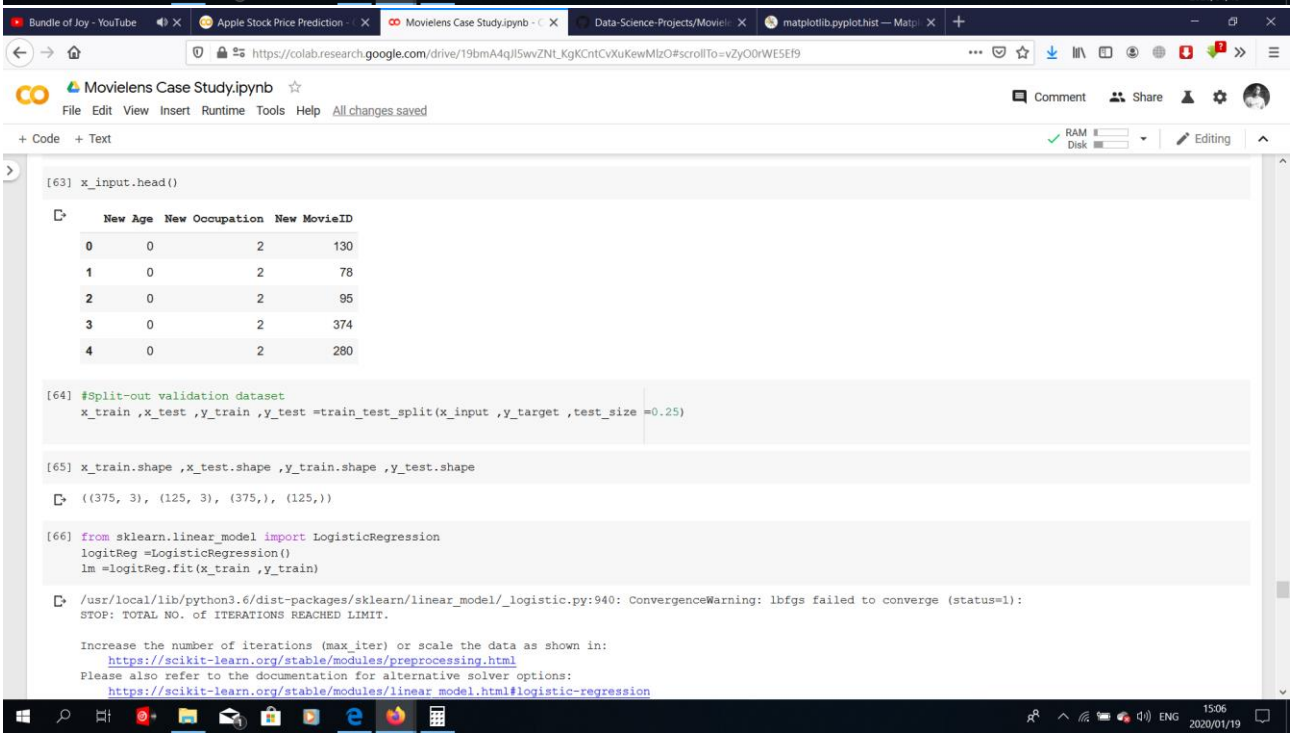
File Edit View Insert Runtime Tools Help All changes saved

Comment Share Editing

+ Code + Text

```
[60] le.fit(few_viewership['MovieID'])
     x_movieid = le.transform(few_viewership['MovieID'])
     x_movieid
```

```
array([130,  78,  95, 374, 280, 132, 156, 321,  71,  96,  72,  98, 287,
       330, 107, 318, 304, 251, 355, 319, 274,  80, 154,  61, 278,  12,
       119, 211, 186,  84, 271, 364, 189,  67, 231,  86, 226, 103, 316,
        18,   0, 243, 244, 305,  29, 104, 105, 135, 252,  62, 359,  74,
       145, 161, 346, 184,  75, 264,  76, 266, 302, 121, 329, 379, 136,
       222, 205, 137, 392, 326, 342, 139, 355,  49, 260, 356, 357, 343,
       148, 194,  33, 265, 347,  92,  44, 149, 360, 185, 158, 127, 366,
       367, 368,  17, 267, 293, 225, 380,  68, 207, 398, 323, 237, 100,
       227, 324, 140, 252,  60,  50, 272,  30, 170, 113, 403,  54, 173,
       255, 151, 162, 130, 224, 163, 279, 372, 289,  69, 131, 187,  83,
       133,  70, 281,  15, 308, 297, 234, 286, 407, 239, 193, 413, 240,
       241,  28, 122, 242,  20,   3,  21, 274, 115,  46, 294,  39,  51,
       118,  97,  52, 181, 376, 166, 378, 353,  85,  56, 312, 247, 244,
       220, 331, 248,  36, 135, 246, 400, 143,  41, 144, 145, 415, 146,
       377, 198,  76, 169, 389,  16, 314, 136, 172, 414, 112, 338, 195,
       157, 149,  77, 262, 191, 396,  29, 324, 359, 111, 150,  64,  54,
       151, 152,  82, 131,  69, 280, 132, 133, 164,  70, 165, 391, 160,
       154, 292, 362, 301, 243, 399, 248, 325, 259, 246, 124, 257, 379,
       136, 333, 138, 108,  29, 252,  54, 131, 133, 240, 119, 376, 404,
       282, 167, 388, 134, 305, 332, 141, 337, 276, 126,   9,  32, 277,
       183, 168, 266, 175,  89, 203,  90, 204, 329, 317,  25, 219,  57,
       392,  58, 147, 411,  59,  10, 194, 254, 412, 338,  11, 306,  34,
        66, 196,  81,  35, 350, 296, 232,  18,  26, 406,  27,   1, 339,
       324, 110,  60,  87,  13,  14, 128, 129,  82, 351,  45, 279, 153,
       352, 289, 385, 290, 280, 268, 386, 188, 233,  70, 281, 307, 176,
       308, 297, 269,  19, 123, 340, 256, 208, 361, 291, 270, 197, 155,
       309, 310, 235, 311, 236, 298, 373, 408,  99,  91, 341, 221,  36,
        22,  53,  74, 171, 261, 209, 199, 365, 363, 210, 322, 313, 200,
        37, 249, 354, 334, 137, 223, 299, 177, 355, 335, 178, 211, 212,
       201,  93, 191, 202, 283, 213, 381, 327, 358, 252,   2,  30, 253,
        23,  63, 179, 344, 273, 180, 114, 369,  94, 214, 374, 375, 300,
```



Movielens Case Study.ipynb

File Edit View Insert Runtime Tools Help All changes saved

Comment Share Editing

+ Code + Text

```
[63] x_input.head()
```

	New Age	New Occupation	New MovieID
0	0	2	130
1	0	2	78
2	0	2	95
3	0	2	374
4	0	2	280

```
[64] #Split-out validation dataset
     x_train ,x_test ,y_train ,y_test =train_test_split(x_input ,y_target ,test_size =0.25)
```

```
[65] x_train.shape ,x_test.shape ,y_train.shape ,y_test.shape
```

```
((375, 3), (125, 3), (375,), (125,))
```

```
[66] from sklearn.linear_model import LogisticRegression
     logitReg =LogisticRegression()
     lm =logitReg.fit(x_train ,y_train)
```

```
/usr/local/lib/python3.6/dist-packages/sklearn/linear_model/_logistic.py:940: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

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

Bundle of Joy - YouTube | Apple Stock Price Prediction - C | Movielens Case Study.ipynb - C | Data-Science-Projects/Movielens | matplotlib.pyplot.hist — Matplotlib

https://colab.research.google.com/drive/19bmA4qJl5wvZNt_KgKCntCvXuKewMlzO#scrollTo=vZyO0rWE5Ef9

Movielens Case Study.ipynb

File Edit View Insert Runtime Tools Help All changes saved

Comment Share

+ Code + Text

RAM Disk Editing

```
[70] estimated =pd.Series(result ,name ='Estimated Values')

[71] final_result =pd.concat([y_test ,estimated] ,axis=1)

[72] #Test the options amd evaluation metrix
     print(accuracy_score(y_test ,result))
     print(confusion_matrix(y_test ,result))
     print(classification_report(y_test ,result))
```

```
0.28
[[ 0  0  1  6  0]
 [ 0  0  6  9  0]
 [ 0  0  6 21  2]
 [ 0  0 17 26  6]
 [ 0  0  8 14  3]]
              precision    recall  f1-score   support

           1       0.00      0.00      0.00         7
           2       0.00      0.00      0.00        15
           3       0.16      0.21      0.18        29
           4       0.34      0.53      0.42        49
           5       0.27      0.12      0.17        25

    accuracy                           0.28       125
   macro avg       0.15      0.17      0.15       125
weighted avg       0.23      0.28      0.24       125

/usr/local/lib/python3.6/dist-packages/sklearn/metrics/_classification.py:1272: UndefinedMetricWarning: Precision and F-score are ill-defined and being set to 0.0 in
  _warn_prf(average, modifier, msg_start, len(result))
```

```
[74] #Plot the histogram
```

ENG 15:06 2020/01/19

Bundle of Joy - YouTube | Apple Stock Price Prediction - C | Movielens Case Study.ipynb - C | Data-Science-Projects/Movielens | matplotlib.pyplot.hist — Matplotlib

https://colab.research.google.com/drive/19bmA4qJl5wvZNt_KgKCntCvXuKewMlzO#scrollTo=vZyO0rWE5Ef9

Movielens Case Study.ipynb

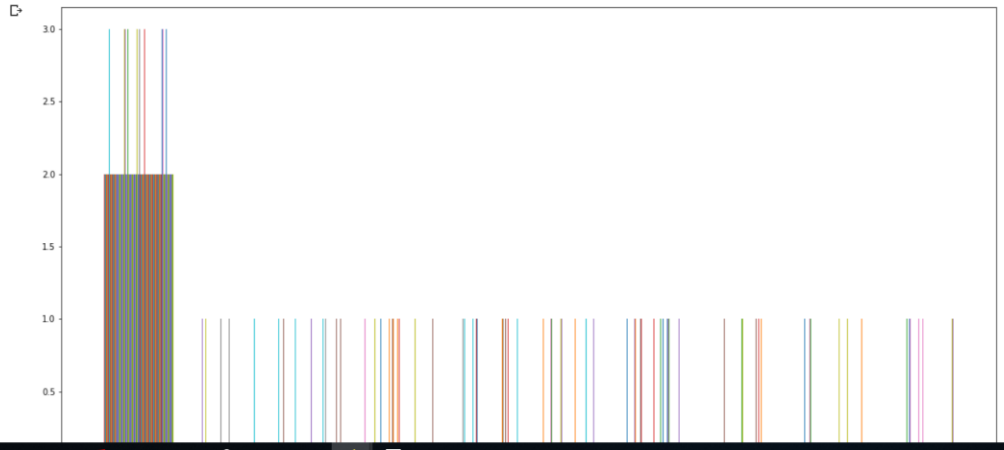
File Edit View Insert Runtime Tools Help All changes saved

Comment Share

+ Code + Text

RAM Disk Editing

```
[74] #Plot the histogram
     plt.figure(figsize=(20,10))
     plt.hist(x=x_input)
     plt.show()
```



ENG 15:06 2020/01/19