Industrial Project Report

Submitted in partial fulfillment of the degree

B-tech in Computer Science Engineering

PROJECT ON :- MOVIE RECOMMENDATION SYSTEM USING TENSOR FLOW

Second-year student of

SILIGURI INSTITUTE OF TECHNOLOGY

THIS IS SUMBITTED IN FULFILLMENT OF THE REQUIREMENTS FO THE DEGREE OF

AFFLIATED BY

MAULANA ABDUL KALAM AZAD UNIVERSITY OF TECHNOLOGY

Sikharthy Infotech
Start with a Dream, Finish with a success

UNDER THE SUPERVISION OF :- MR. RIPAM KUNDU

SIKHARTHY INFOTECH Pvt. Ltd.

Siliguri Institute Of Technology Hill Cart Road, Salbari, Sukna, West Bengal 734009

SUBMITTED BY-

TEAM LEADER:

Kamalika Saha~11900121057

TEAM MEMBERS:

Sweety Nag~11900121055

Monami Ghosh~11900121045

Annesha Basu~11900121004

Greeny Kundu~11900121024

Priyadarshini Sen~11900121049

DEPARTMENT OF COMPUTER SCIENCE ENGINEERING

I hereby forward the documentation prepared under my supervision by Mr. **Ripam Kundu Sir** entitled Siliguri Institute Of Technology to be accepted as fulfillment of the requirement for the Degree of Bachelor of Technology in Electrical Engineering, **Siliguri Institute Of Technology** affiliated to **Maulana Abul Kalam Azad University of Technology** (**MAKAUT**).

Mr.Ripam Kundu **(Software Developer)** **Project Guide** **Sikharthy Infotech Pvt. Ltd.**	**HOD** **Department Of Electrical Engineering, SIT**
Shilpi Ghosal **(Director)** **Sikharthy Infotech Pvt. Ltd.**	**TPO** **Siliguri Institute of Technology**

CERTIFICATE OF APPROVAL

The foregoing project is hereby approved as a creditable study for the B.Tech in Computer Science Engineering presented in a manner of satisfactory to warrant its acceptance as a prerequisite to the degree for which it has been submitted. It is understood that by this approval the undersigned do not necessarily endorsed or approved any statement made, opinion expressed or conclusion therein but approve this project only for the purpose for which it is submitted.

Final Examination for
Evaluation of the Project ---------------------------------------

----

Signatures of Examiners

ABSTRACT

A movie recommendation is important in our social life due to its strength in providing enhanced entertainment. Such a system can suggest a set of movies to users based on their interest, or the popularities of the movies. Although, a set of movie recommendation systems have been proposed, most of these either cannot recommend a movie to the existing users efficiently or to a new user by any means. In this paper we propose a movie recommendation system that has the ability to recommend movies to a new user as well as the others. It mines movie databases to collect all the important information, such as, popularity and attractiveness, required for recommendation. It generates movie swarms not only convenient for movie producer to plan a new movie but also useful for movie recommendation. Experimental studies on the real data reveal the efficiency and effectiveness of the proposed system.

ACKNOWLEDGMENT

It is a great pleasure for me to acknowledge the assistance and participation of a large number of individuals in this attempt. Our project report has been structured under the valued suggestion, support, and guidance of **Mr. Ripam Kundu**. Under his guidance, we have accomplished the challenging task in a very short time. Finally, we express our sincere thankfulness to our family members for inspiring me all throughout and always encouraging us.

Group Members' Signature

____

____

____

____

____

____

TABLE OF CONTENTS

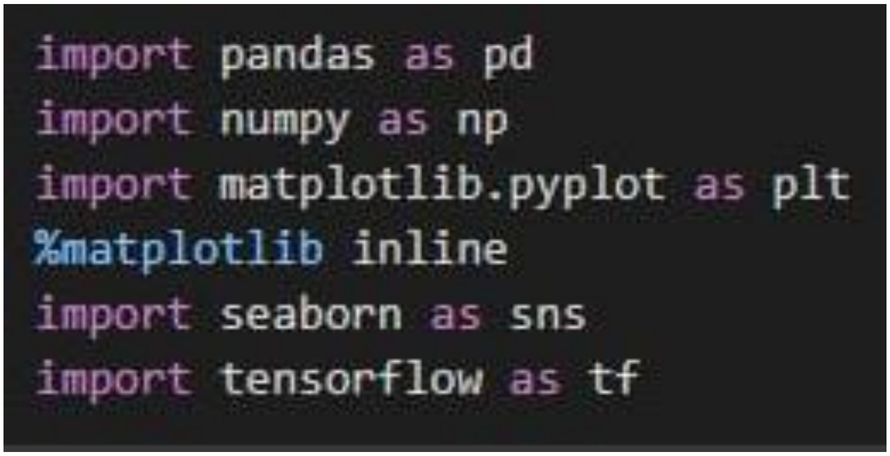
LIBRARIES USED

Importing Libraries

The analysis will be done using the following libraries :

- Pandas : This library helps to load the data frame in a 2D array format and has multiple functions to perform analysis tasks in one go.
- NumPy: NumPy arrays are very fast and can perform large computations in a very short time.
- Matplotlib/Seaborn : This library is used to draw visualizations.
- Tensorflow: This library is a free and open-source software library for machine learning and artificial intelligence. It can be used across a range of tasks but has a particular focus on training and inference of deep neural networks.

To import all these libraries, we can use the code below:

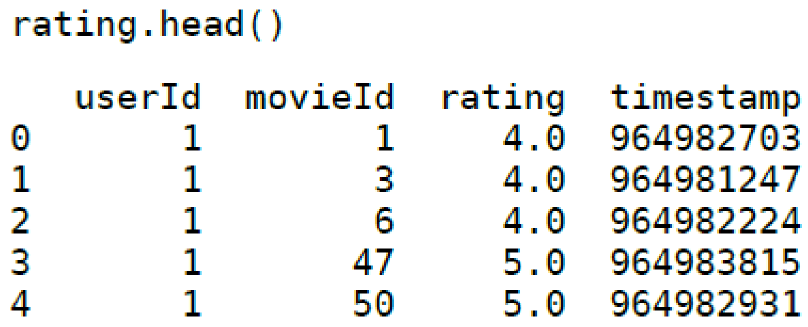


```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
import seaborn as sns
import tensorflow as tf
```

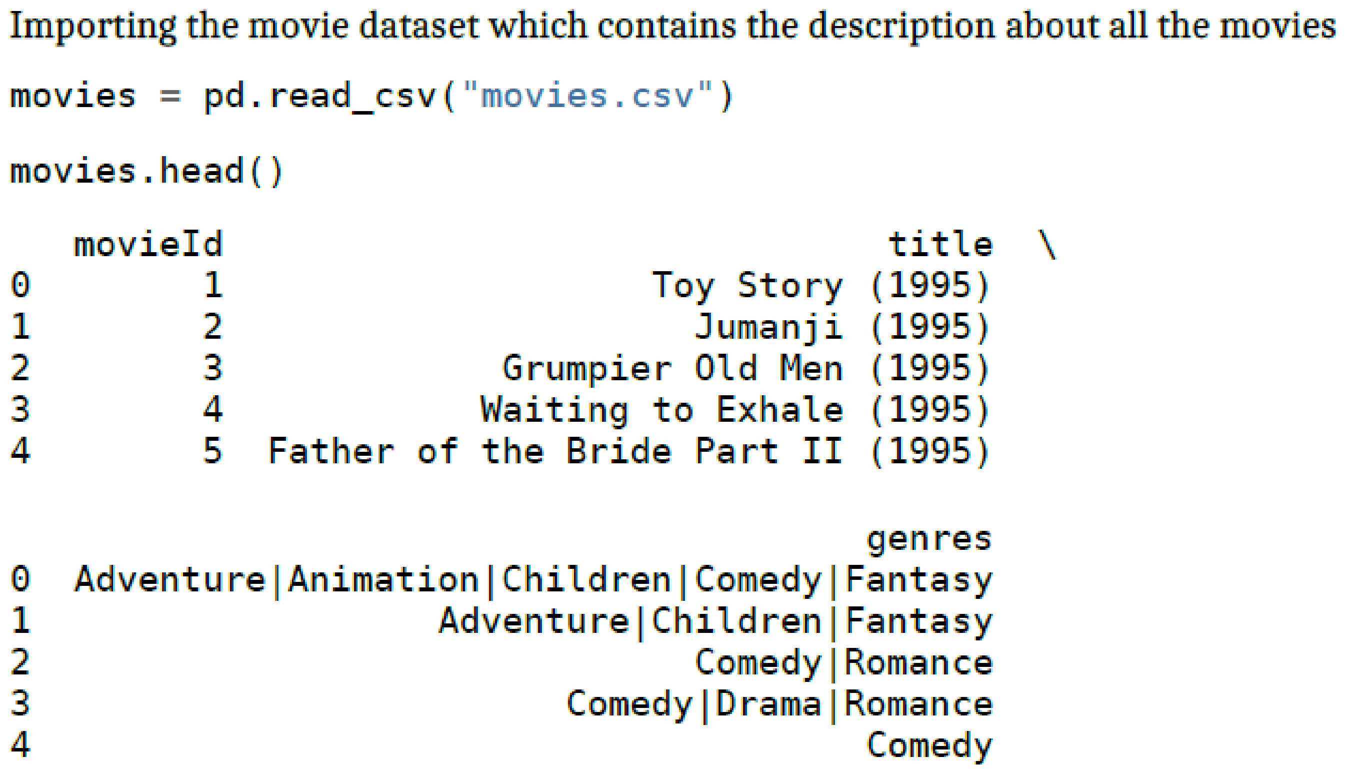
IMPORTING DATASET

Importing the rating data set which contains ratings given by the users to the movies they
watched.

rating = pd.read_csv (' ratings.csv ')

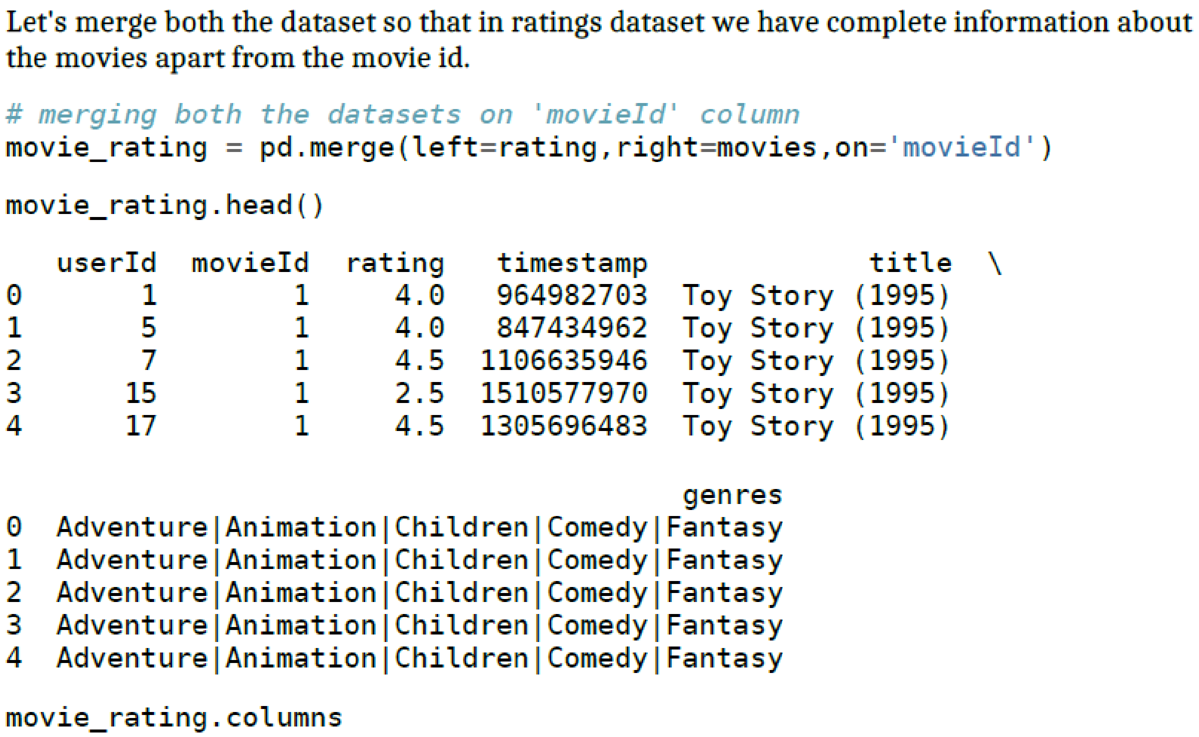


```
rating.head()

   userId  movieId  rating  timestamp
0       1        1     4.0  964982703
1       1        3     4.0  964981247
2       1        6     4.0  964982224
3       1       47     5.0  964983815
4       1       50     5.0  964982931
```



Importing the movie dataset which contains the description about all the movies

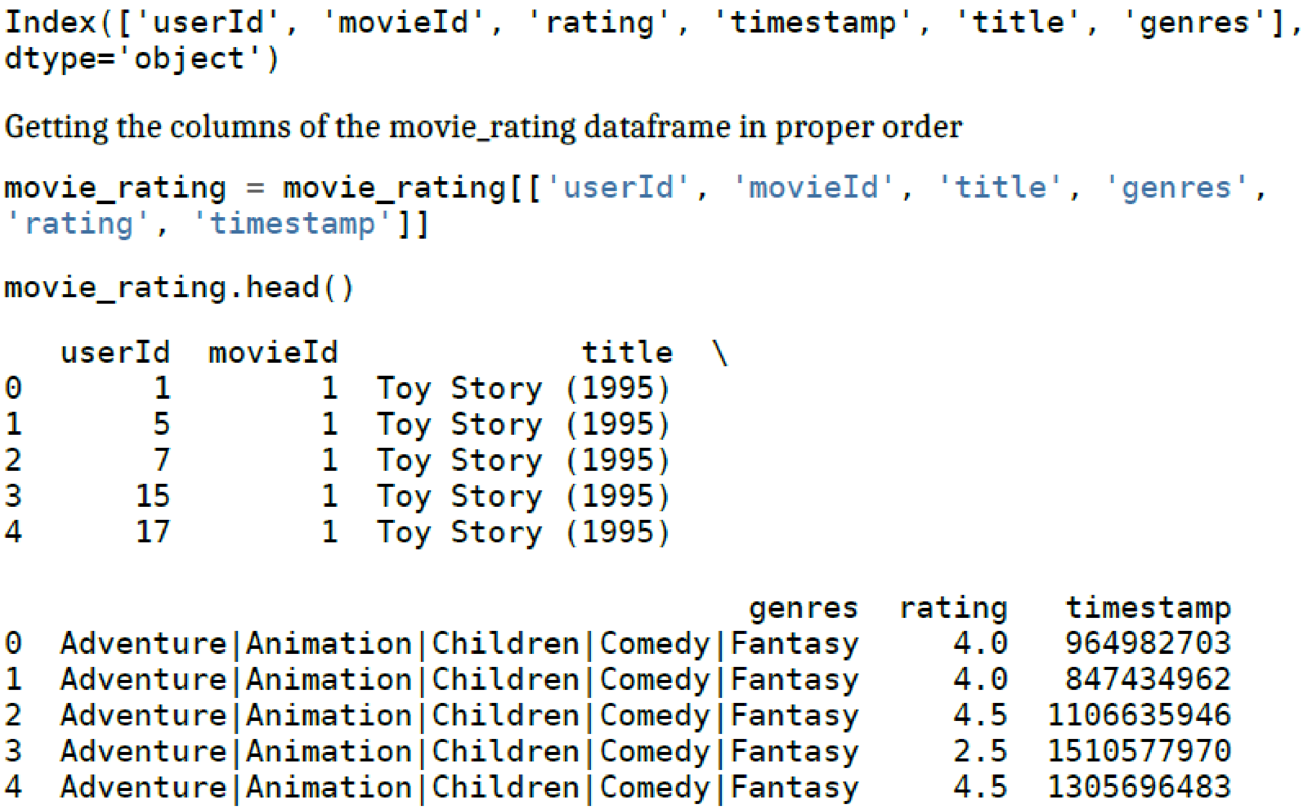
```
movies = pd.read_csv("movies.csv")

movies.head()

   movieId                               title  \
0        1                    Toy Story (1995)
1        2                      Jumanji (1995)
2        3             Grumpier Old Men (1995)
3        4            Waiting to Exhale (1995)
4        5  Father of the Bride Part II (1995)

                                        genres
0  Adventure|Animation|Children|Comedy|Fantasy
1                   Adventure|Children|Fantasy
2                               Comedy|Romance
3                         Comedy|Drama|Romance
4                                       Comedy
```



Let's merge both the dataset so that in ratings dataset we have complete information about the movies apart from the movie id.

```
# merging both the datasets on 'movieId' column
movie_rating = pd.merge(left=rating,right=movies,on='movieId')

movie_rating.head()

   userId  movieId  rating   timestamp             title  \
0       1        1     4.0   964982703  Toy Story (1995)
1       5        1     4.0   847434962  Toy Story (1995)
2       7        1     4.5  1106635946  Toy Story (1995)
3      15        1     2.5  1510577970  Toy Story (1995)
4      17        1     4.5  1305696483  Toy Story (1995)

                                        genres
0  Adventure|Animation|Children|Comedy|Fantasy
1  Adventure|Animation|Children|Comedy|Fantasy
2  Adventure|Animation|Children|Comedy|Fantasy
3  Adventure|Animation|Children|Comedy|Fantasy
4  Adventure|Animation|Children|Comedy|Fantasy

movie_rating.columns
```

movie_rating.columns



```
Index(['userId', 'movieId', 'rating', 'timestamp', 'title', 'genres'],
dtype='object')
```

Getting the columns of the movie_rating dataframe in proper order

```
movie_rating = movie_rating[['userId', 'movieId', 'title', 'genres',
'rating', 'timestamp']]

movie_rating.head()

   userId  movieId             title  \
0       1        1  Toy Story (1995)
1       5        1  Toy Story (1995)
2       7        1  Toy Story (1995)
3      15        1  Toy Story (1995)
4      17        1  Toy Story (1995)

                                        genres  rating   timestamp
0  Adventure|Animation|Children|Comedy|Fantasy     4.0   964982703
1  Adventure|Animation|Children|Comedy|Fantasy     4.0   847434962
2  Adventure|Animation|Children|Comedy|Fantasy     4.5  1106635946
3  Adventure|Animation|Children|Comedy|Fantasy     2.5  1510577970
4  Adventure|Animation|Children|Comedy|Fantasy     4.5  1305696483
```

EXPLORATORY DATA ANALYSIS

```
movie_rating.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 100836 entries, 0 to 100835
Data columns (total 6 columns):
userId       100836 non-null int64
movieId      100836 non-null int64
title        100836 non-null object
genres       100836 non-null object
rating       100836 non-null float64
timestamp    100836 non-null int64
dtypes: float64(1), int64(3), object(2)
memory usage: 5.4+ MB
```

```
movie_rating.isnull().sum()
```

```
userId       0
movieId      0
title        0
genres       0
rating       0
timestamp    0
dtype: int64
```

Let's create a dataframe with number of ratings and average rating for each movie

```
movie_rating.head(2)
```

```
   userId  movieId             title  \
0       1        1  Toy Story (1995)
1       5        1  Toy Story (1995)

                                        genres  rating  timestamp
0  Adventure|Animation|Children|Comedy|Fantasy     4.0  964982703
1  Adventure|Animation|Children|Comedy|Fantasy     4.0  847434962
```

```
# grouping the movies based on average rating
average_rating_movies = movie_rating.groupby('title')
['rating'].mean().sort_values(ascending=False)
```

```
average_rating_movies.head(10)
```

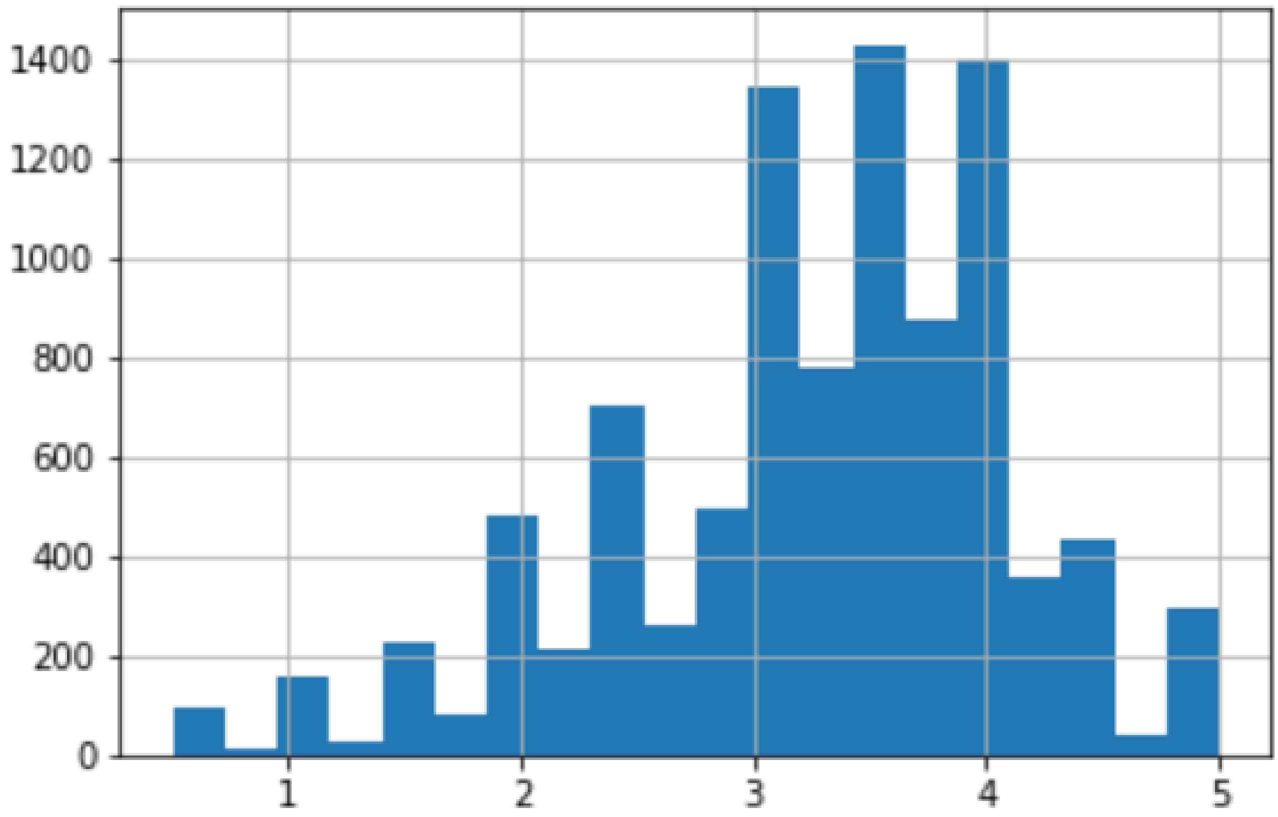
```
title
Karlson Returns (1970)                           5.0
Winter in Prostokvashino (1984)                  5.0
My Love (2006)                                   5.0
Sorority House Massacre II (1990)                5.0
Winnie the Pooh and the Day of Concern (1972)    5.0
Sorority House Massacre (1986)                   5.0
Bill Hicks: Revelations (1993)                   5.0
My Man Godfrey (1957)                            5.0
Hellbenders (2012)                               5.0
In the blue sea, in the white foam. (1984)       5.0
Name: rating, dtype: float64
```

```
average_rating_movies.hist(bins=20)
plt.show()
```

DATA VIZUALISATION :-

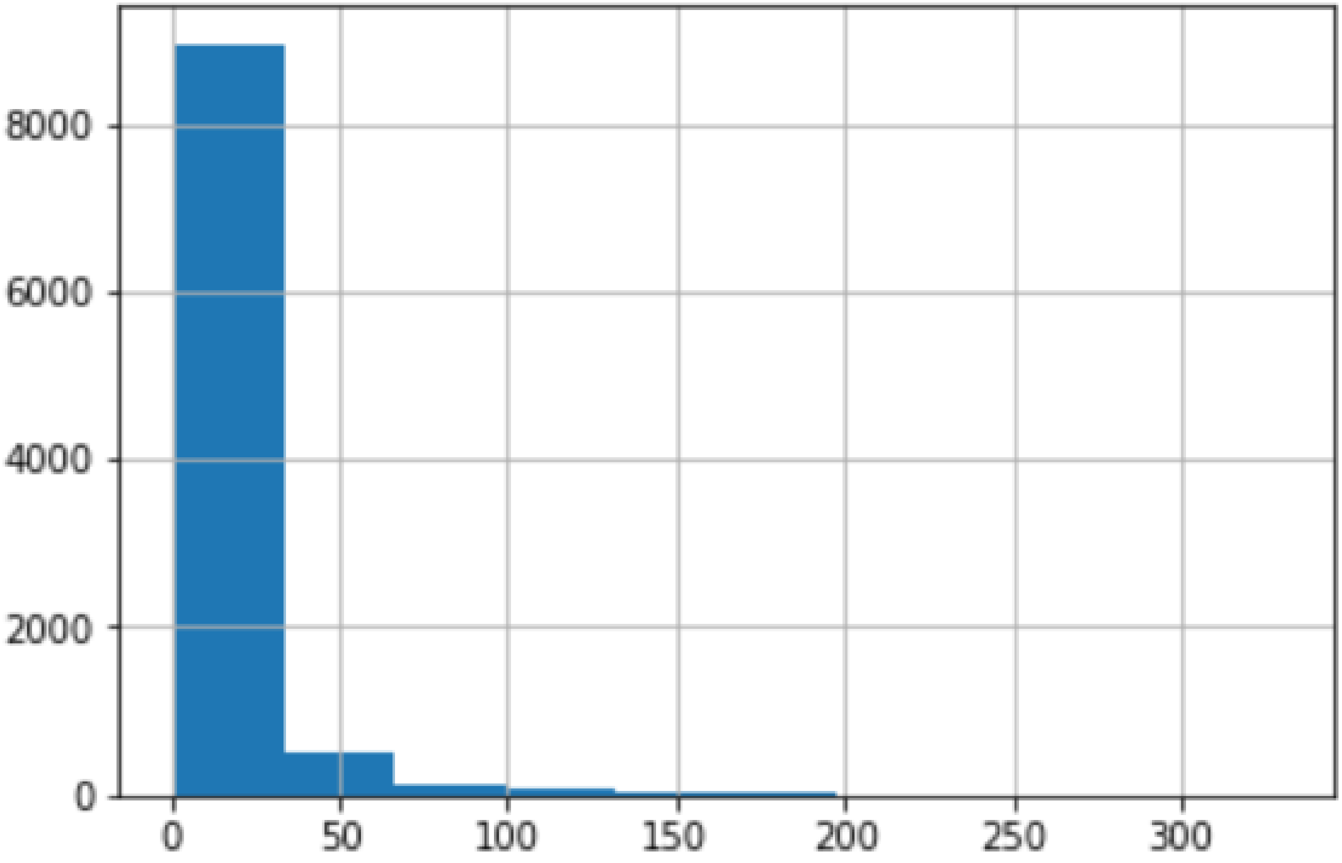
Data visualization is the graphical representation of information and data. By using **visual elements like charts, graphs and maps**, data visualization tools provide an accessible way to see and understand trends, outliers, and patterns in data. Additionally, it provides an excellent way for employees or business owners to present data to non-technical audiences without confusion.

```
average_rating_movies.hist(bins=20)
plt.show()
```



Maximum movies have average rating in the range 3 to 4. The movies which have average =
5.0 may be the ones which may have been watched once or twice.

```
count_userid.hist()
plt.show()
```



Maximum movies have been viewed in the range 0 - 40 views
The movies which have average = 5.0 may be the ones which may have been watched once
or twice. Let's see number of ratings given to movies which have average rating = 5.0

BUILDING RECOMMENDATION SYSTEM

Collaborative Filtering is the most common technique used when it comes to building intelligent recommender systems that can learn to give better recommendations as more information about users is collected.

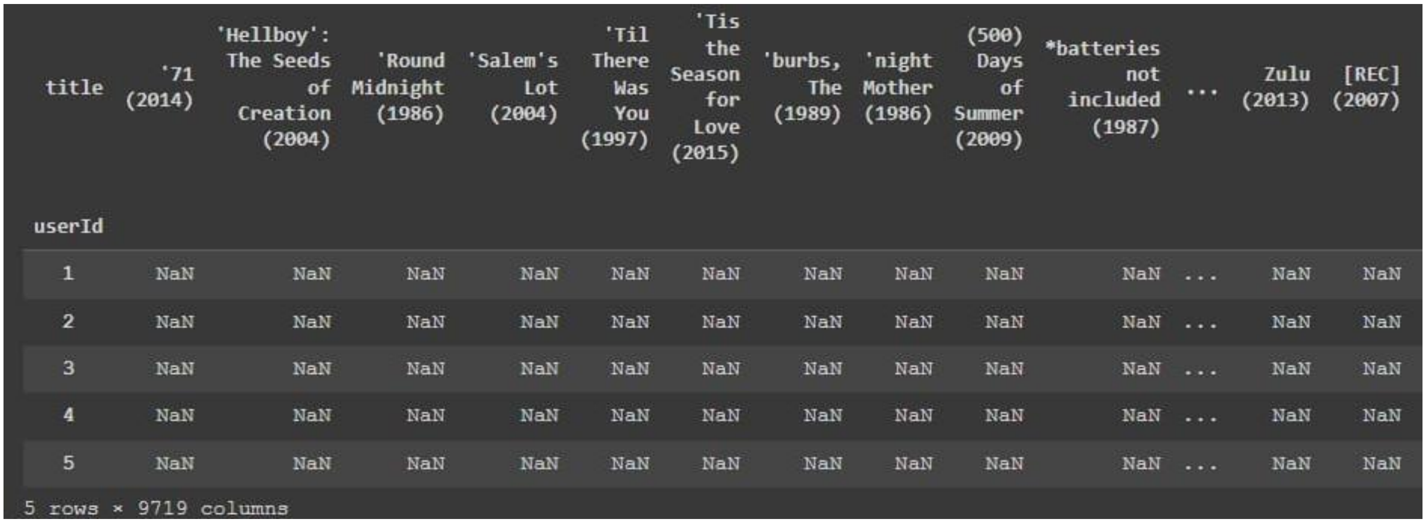
Most websites like Amazon, YouTube, and Netflix use collaborative filtering as a part of their sophisticated recommendation systems. You can use this technique to build recommenders that give suggestions to a user on the basis of the likes and dislikes of similar users.

Creating pivot table to create item by item collaborative filtering

```
movie_rating_pivot = pd.pivot_table(index='userId',columns='title',values='rating',data=movie_rating)
```

There will be many Nan values because users have watched only few of the movies and given ratings only to those movies.

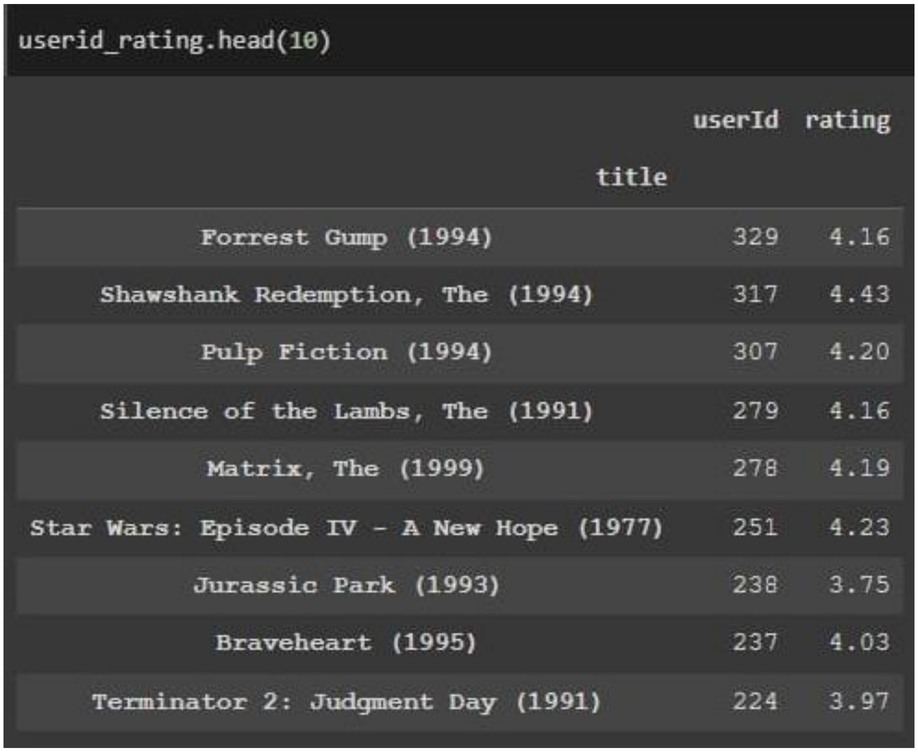
```
movie_rating_pivot.head()
```



title	'71 (2014)	'Hellboy': The Seeds of Creation (2004)	'Round Midnight (1986)	'Salem's Lot (2004)	'Til There Was You (1997)	'Tis the Season for Love (2015)	'burbs, The (1989)	'night Mother (1986)	(500) Days of Summer (2009)	*batteries not included (1987)	...	Zulu (2013)	[REC] (2007)
userId													
1	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN
2	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN
3	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN
4	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN
5	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN

5 rows × 9719 columns

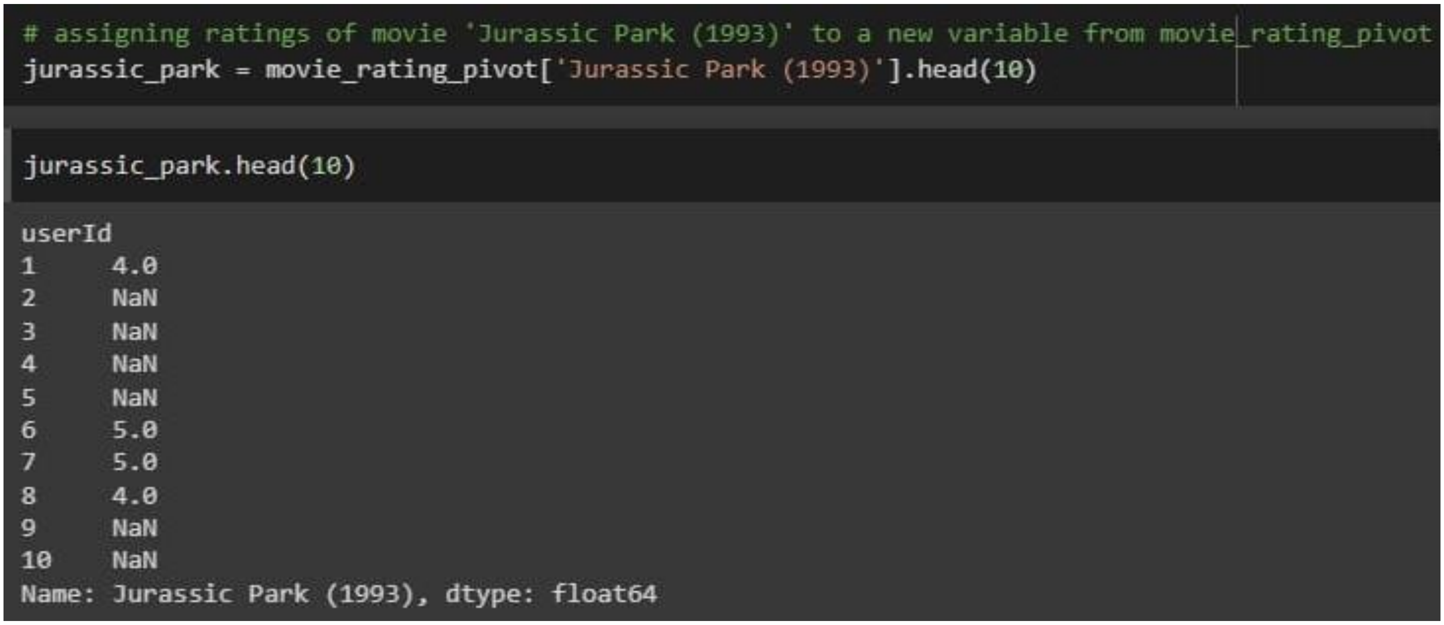
MOST RATED MOVIES :



```
userid_rating.head(10)
```

	userId	rating
title		
Forrest Gump (1994)	329	4.16
Shawshank Redemption, The (1994)	317	4.43
Pulp Fiction (1994)	307	4.20
Silence of the Lambs, The (1991)	279	4.16
Matrix, The (1999)	278	4.19
Star Wars: Episode IV - A New Hope (1977)	251	4.23
Jurassic Park (1993)	238	3.75
Braveheart (1995)	237	4.03
Terminator 2: Judgment Day (1991)	224	3.97

Let's find which movies to recommend to the users who have watched 'Jurassic Park (1993)'.
To do this we have to find correlation of 'Jurassic Park (1993)' with other movies which have been rated in a similar way by the users .

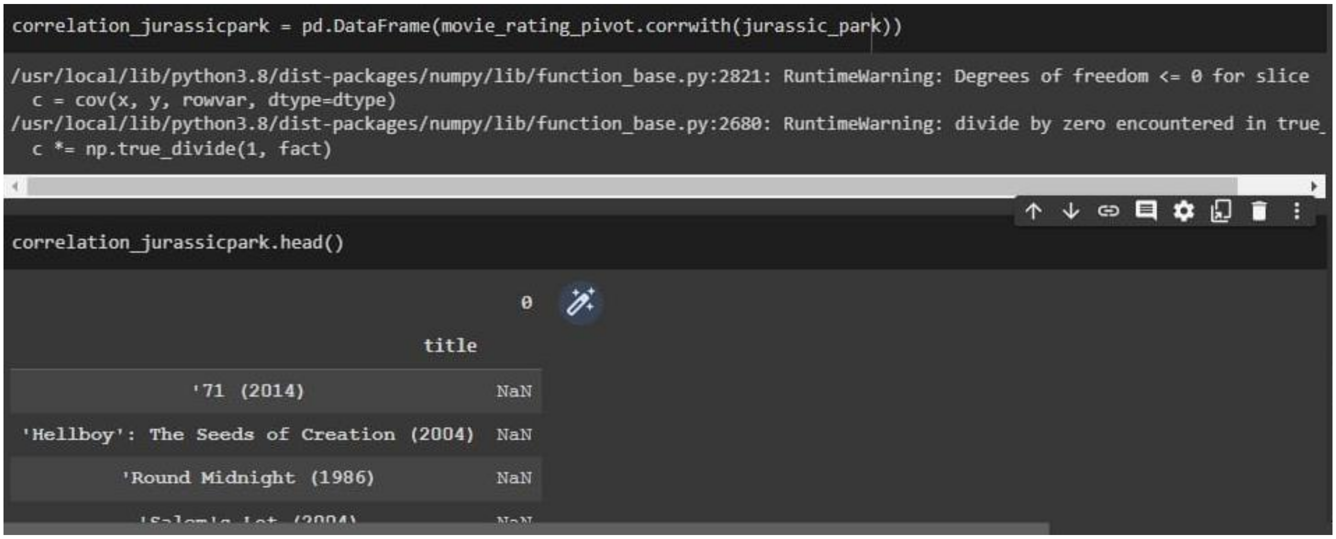


```
# assigning ratings of movie 'Jurassic Park (1993)' to a new variable from movie_rating_pivot
jurassic_park = movie_rating_pivot['Jurassic Park (1993)'].head(10)
```

```
jurassic_park.head(10)
```

```
userId
1     4.0
2     NaN
3     NaN
4     NaN
5     NaN
6     5.0
7     5.0
8     4.0
9     NaN
10    NaN
Name: Jurassic Park (1993), dtype: float64
```

Find the correlation with other movies from movie_rating_pivot table



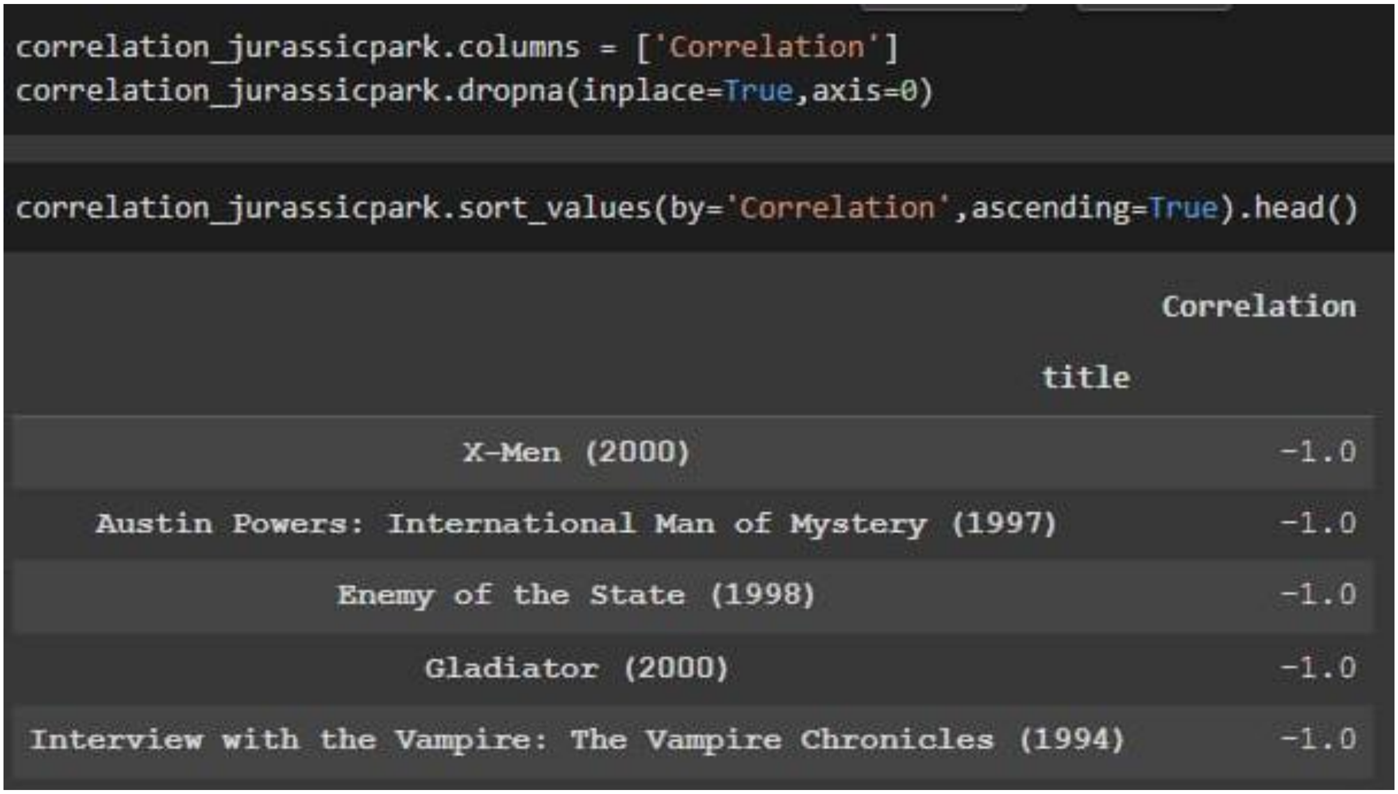
```
correlation_jurassicpark = pd.DataFrame(movie_rating_pivot.corrwith(jurassic_park))
```

```
/usr/local/lib/python3.8/dist-packages/numpy/lib/function_base.py:2821: RuntimeWarning: Degrees of freedom <= 0 for slice
  c = cov(x, y, rowvar, dtype=dtype)
/usr/local/lib/python3.8/dist-packages/numpy/lib/function_base.py:2680: RuntimeWarning: divide by zero encountered in true_
  c *= np.true_divide(1, fact)
```

```
correlation_jurassicpark.head()
```

```
                                        0
                            title
                      '71 (2014)      NaN
'Hellboy': The Seeds of Creation (2004)  NaN
           'Round Midnight (1986)      NaN
              'Salem's Lot (2004)      NaN
```

Removing Nan values and naming the column as 'Correlation'



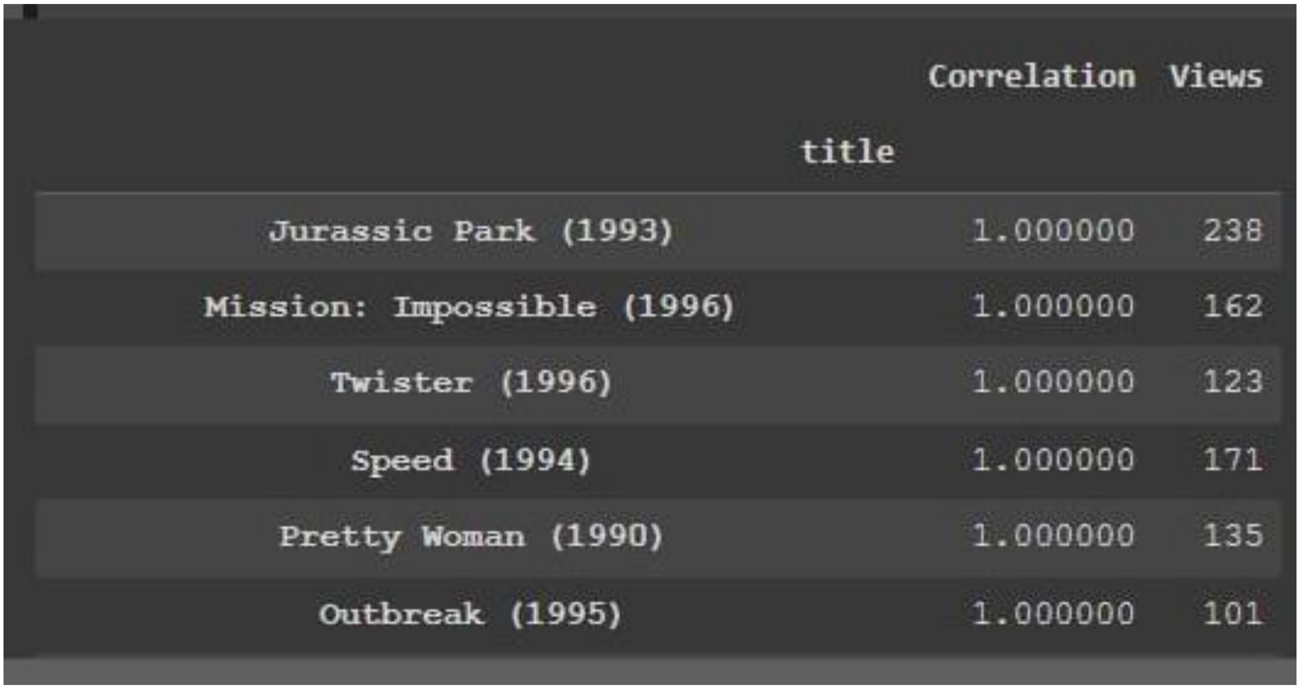
```
correlation_jurassicpark.columns = ['Correlation']
correlation_jurassicpark.dropna(inplace=True,axis=0)
```

```
correlation_jurassicpark.sort_values(by='Correlation',ascending=True).head()
```

```
                                                            Correlation
                                                    title
                                      X-Men (2000)                 -1.0
      Austin Powers: International Man of Mystery (1997)          -1.0
                            Enemy of the State (1998)             -1.0
                                   Gladiator (2000)               -1.0
Interview with the Vampire: The Vampire Chronicles (1994)        -1.0
```

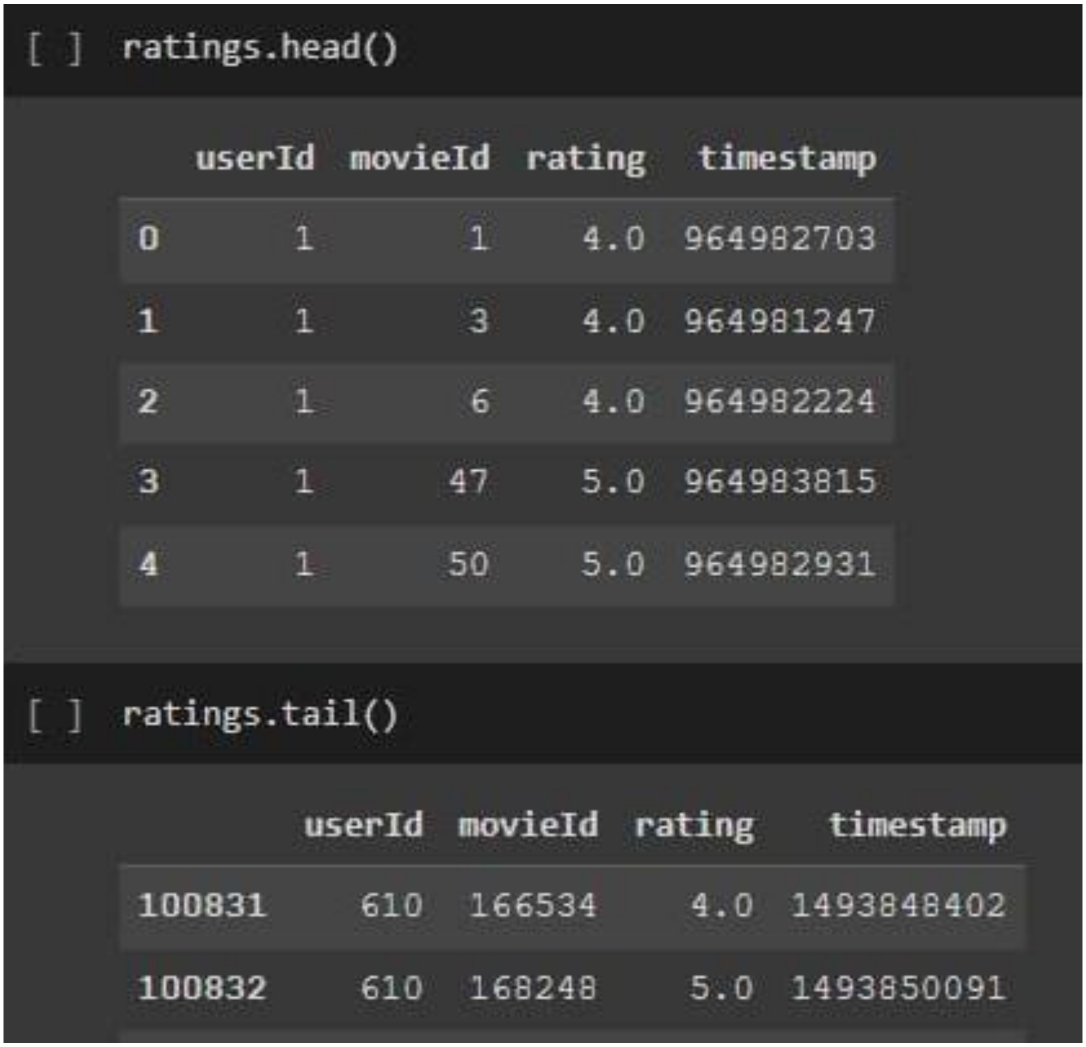
Now filtering out top 20 movies which have views greater than 100

```
correlation_jurassicpark[correlation_jurassicpark['Views'] > 
100].sort_values(by='Correlation',ascending=False).head(20)
```



	Correlation	Views
title		
Jurassic Park (1993)	1.000000	238
Mission: Impossible (1996)	1.000000	162
Twister (1996)	1.000000	123
Speed (1994)	1.000000	171
Pretty Woman (1990)	1.000000	135
Outbreak (1995)	1.000000	101

USER-BASED COLLABORATIVE FILTERING



```
[ ]  ratings.head()
```

```
   userId  movieId  rating  timestamp
0       1        1     4.0  964982703
1       1        3     4.0  964981247
2       1        6     4.0  964982224
3       1       47     5.0  964983815
4       1       50     5.0  964982931
```

```
[ ]  ratings.tail()
```

```
        userId  movieId  rating   timestamp
100831     610   166534     4.0  1493848402
100832     610   168248     5.0  1493850091
```

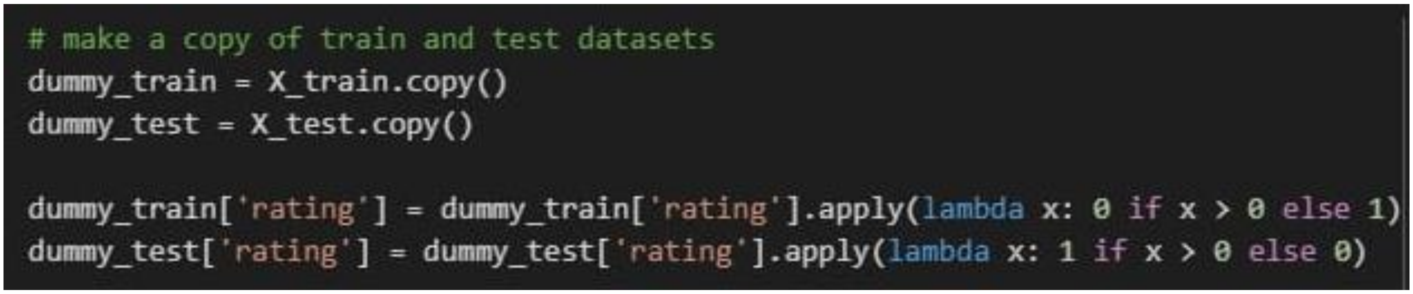
CREATE A COPY OF TRAIN AND TEST DATASET

These datasets will be used for prediction and evaluation.

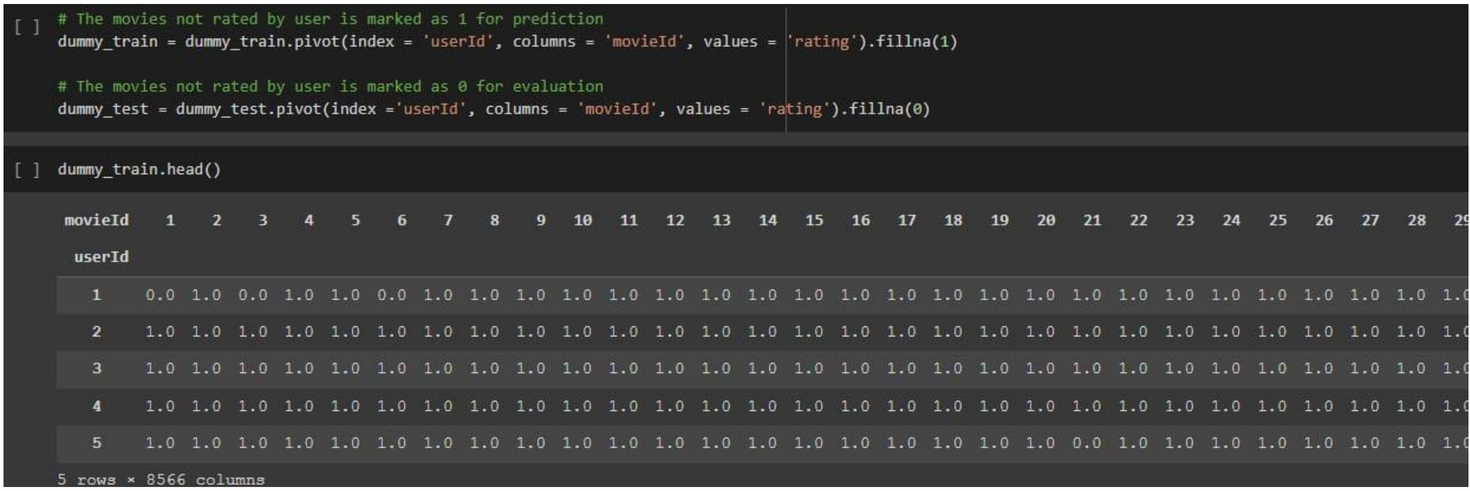
Dummy train will be used later for prediction of the movies which has not been rated by the user. To ignore the movies rated by the user, we will mark it as 0 during prediction. The movies not rated by user is marked as 1 for prediction.

Dummy test will be used for evaluation. To evaluate, we will only make prediction on the movies rated by the user. So, this is marked as 1.

This is just opposite of dummy_train.



```
# make a copy of train and test datasets
dummy_train = X_train.copy()
dummy_test = X_test.copy()

dummy_train['rating'] = dummy_train['rating'].apply(lambda x: 0 if x > 0 else 1)
dummy_test['rating'] = dummy_test['rating'].apply(lambda x: 1 if x > 0 else 0)
```

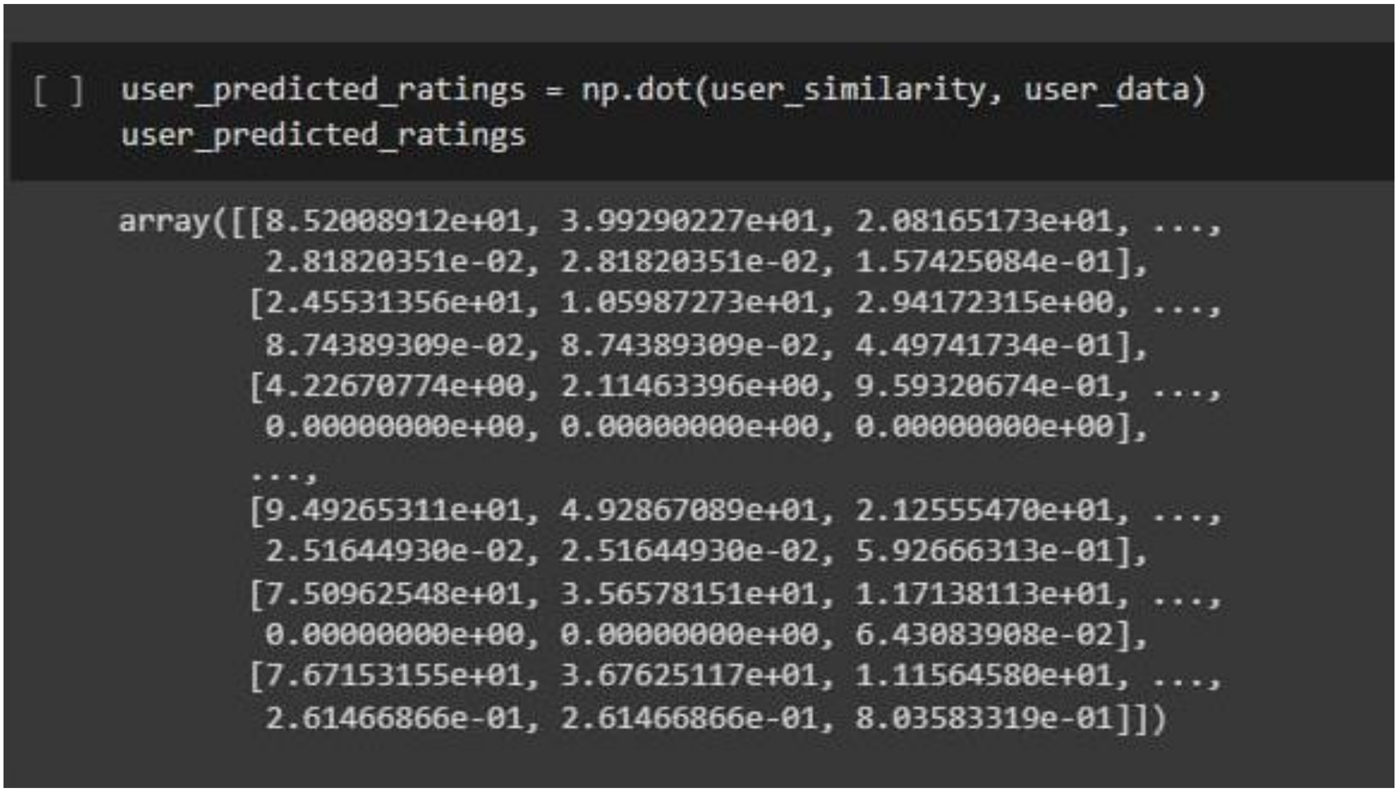


```
[ ]  # The movies not rated by user is marked as 1 for prediction
     dummy_train = dummy_train.pivot(index = 'userId', columns = 'movieId', values = 'rating').fillna(1)

     # The movies not rated by user is marked as 0 for evaluation
     dummy_test = dummy_test.pivot(index ='userId', columns = 'movieId', values = 'rating').fillna(0)
```

```
[ ]  dummy_train.head()
```

```
movieId   1   2   3   4   5   6   7   8   9   10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29
 userId
   1     0.0 1.0 0.0 1.0 1.0 0.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.
   2     1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.
   3     1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.
   4     1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.
   5     1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 0.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.
5 rows × 8566 columns
```

PREDICTING THE USER RATINGS ON THE MOVIES



```
[ ]  user_predicted_ratings = np.dot(user_similarity, user_data)
     user_predicted_ratings
```

```
array([[8.52008912e+01, 3.99290227e+01, 2.08165173e+01, ...,
        2.81820351e-02, 2.81820351e-02, 1.57425084e-01],
       [2.45531356e+01, 1.05987273e+01, 2.94172315e+00, ...,
        8.74389309e-02, 8.74389309e-02, 4.49741734e-01],
       [4.22670774e+00, 2.11463396e+00, 9.59320674e-01, ...,
        0.00000000e+00, 0.00000000e+00, 0.00000000e+00],
       ...,
       [9.49265311e+01, 4.92867089e+01, 2.12555470e+01, ...,
        2.51644930e-02, 2.51644930e-02, 5.92666313e-01],
       [7.50962548e+01, 3.56578151e+01, 1.17138113e+01, ...,
        0.00000000e+00, 0.00000000e+00, 6.43083908e-02],
       [7.67153155e+01, 3.67625117e+01, 1.11564580e+01, ...,
        2.61466866e-01, 2.61466866e-01, 8.03583319e-01]])
```

FUNCTIONAL REQUIREMENTS OF THE SYSTEM

SOFTWARE:

- *Operating System*
- Windows OS 11
- TensorFlow

WEB BROWSER:

- Internet Explorer 7
- Google Chrome

CODING LANGUAGE :

- Python

REFERENCE

www.tensorflow.org

www.geeksforgeeks.org

www.slideshare.net

CONCLUSION

Building a movie recommendation system using TensorFlow can be a great way to improve the user experience on a movie streaming platform. By utilizing machine learning algorithms, the system can analyze user data and provide personalized movie recommendations, increasing user engagement and satisfaction.

CONTRIBUTION

1. Kamalika Saha - building recommendation system, user based and item based collaborative filtering

2. Sweety Nag - building recommendation system,data analysis

3. Monami Ghosh - ppt making

4. Priyadarshini Sen - ppt making

5. Annesha Basu - documentation

6. Greeny Kundu - documentation

Thank you