

MID TERM REPORT ON

Movie Recommendation System

# Submitted by: Submitted To:

### **Prince Arjariya(171500238) Mr. Amir Khan**

**Course: B. tech Technical Trainer**

**Branch:CS**

**Section : B**

**INDEX**

ABSTRACT 3

INTRODUCTION 4

TYPES OF RECOMMENDER SYSTEM 5-6

EXISTING SYSTEM 7

PROPOSED SYSTEM 8

HARDWARE DESCRIPTION 9

* 1. PERSONAL COMPUTER(PC) 9

SOFTWARE DESCRIPTION 10

1.ANACONDA 10

**2.SPYDER 10**

SOFTWARE DEVELOPEMENT 11

CODE 12-13

CODE WITH OUTPUT 14-27

CONCLUSION 28

FUTUER SCOPE 29

* + WEBSITES 29

## 

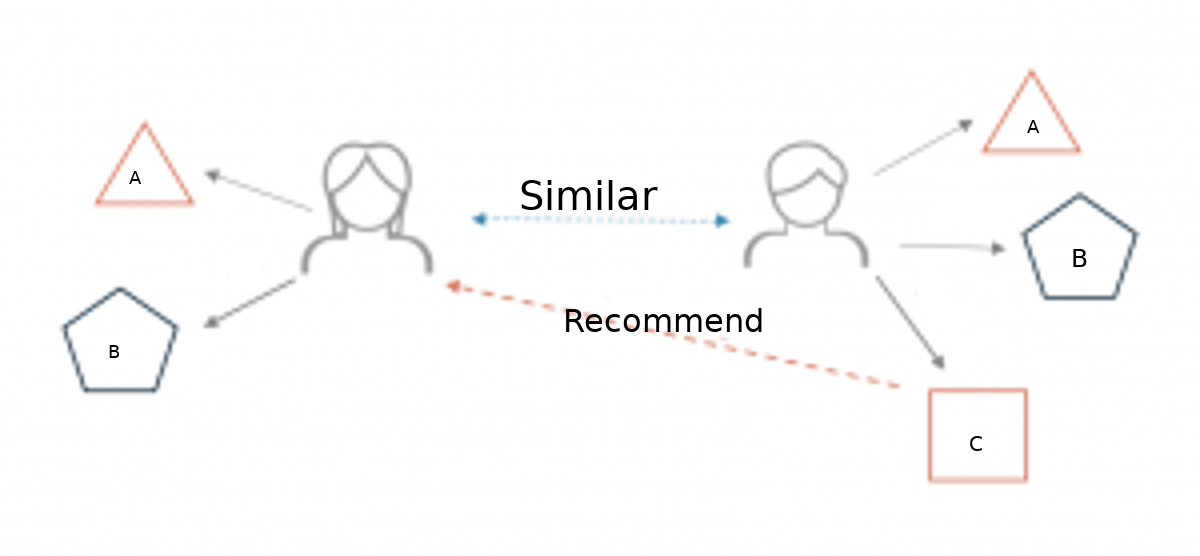
## ABSTRACT:

This report describes the Recommendation systems using python. Recommendation system have become ubiquitous in our lives. In today’s digital world where there is an endless variety of content to be consumed like books, videos, articles, movies, etc., finding the content of one’s liking has become an irksome task. On the other hand, digital content providers want to engage as many users on their service as possible for the maximum time. This is where recommender system comes into picture where the content providers recommend users the content according to the users’ liking in this paper, we have proposed a movie recommender system Movie Mender. The objective of Movie Mender is to provide accurate movie recommendations to users Usually the basic recommender systems consider one of the following factors for generating recommendations, the preference of user is that content based filtering or the preference of similar users is that collaborative filtering. To build a stable and accurate recommender system a hybrid of content-based filtering as well as collaborative filtering will be used.

**INTRODUCTION:**

A recommender system is a simple algorithm whose aim is to provide the most relevant information to a user by discovering patterns in a dataset. The algorithm rates the items and shows the user the items that they would rate highly. An example of recommendation in action is when you visit Amazon and you notice that some items are being recommended to you or when Netflix recommends certain movies to you. They are also used by Music streaming applications such as Spotify and Deezer to recommended movie that you like.

It is a very simple example of how recommender systems work in the context of an e-commerce site.



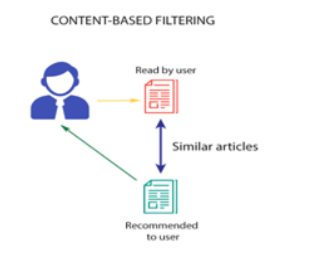
Two users buy the same items A and B from an e-commerce store. When this happens the similarity index of these two users is computed. Depending on the score the system can recommend item C to the other user because it detects that those two users are similar in terms of the item they purchase.

**TYPES OF RECOMMENDATION SYSTEM:**

The most common types of recommendation systems are content**-**based and collaborativefiltering recommender systems. In collaborative filtering, the behavior of a group of users is used to make recommendations to other users. The recommendation is based on the preference of other users. A simple example would be recommending a movie to a user based on the fact that their friend liked the movie. There are two types of collaborative models Memory**-**basedmethods and Model**-**based methods. The advantage of memory-based techniques is that they are simple to implement and the resulting recommendations are often easy to explain. They are divided into two:

**1-Content-based Filtering Systems (CBF based systems):**

In content-based filtering, items are recommended based on comparisons between item profile and user profile. A user profile is content that is found to be relevant to the user in form of keywords or features. A user profile might be seen as a set of assigned keywords terms and features collected by algorithm from items found relevant or interesting by the user. A set of keywords or features of an item is the Item profile. For example: consider a scenario in which a person goes to buy his favorite cake ‘X’ to a pastry. Unfortunately, cake ‘X’ has been sold out and as a result of this the shopkeeper recommends the person to buy cake ‘Y’ which is made up of ingredients similar to cake ‘X’. This is an instance of content-based filtering.



**2-Collaborative filtering-based systems (CF based systems):**

Collaborative filtering system recommends items based on similarity measures between users or items. The system recommends items preferred by similar users. This is based on the scenario where a person asks his friends, who have similar tastes, to recommend him some movies.

****

**EXISTING SYSTEM:**

**DISADVANTAGES:**

**Disadvantages of content-based filtering are:**

1. It does not work for a new user who has not rated any item yet as enough ratings are required content-based recommender evaluates the user preferences and provides accurate recommendations.
2. No recommendation of serendipitous items.
3. Limited Content Analysis- The recommender does not work if the system fails to distinguish the items that a user likes from the items that he does not like.

**Disadvantages of collaborative filtering are:**

1. Early rater problem: Collaborative filtering systems cannot provide recommendations for new items since there are no user ratings on which to base a prediction.
2. Gray sheep: In order for CF based system to work, group with similar characteristics are needed. Even if such groups exist, it will be very difficult to recommend users who do not consistently agree or disagree to these groups.
3. Sparsity problem: In most cases, the number of items exceed the number of users by a great margin which makes it difficult to find items that are rated by enough people.

## PROPOSED SYSTEM:

**ADVANTAGES:**

**Advantages of content-based filtering are:**

1. They capable of recommending unrated items.
2. We can easily explain the working of recommender system by listing the Content features of an item.
3. Content-based recommender systems use need only the rating of the concerned user, and not any other user of the system.

**Advantages of collaborative filtering based systems:**

1. It is dependent on the relation between users which implies that it is content-independent.
2. CF recommender systems can suggest serendipitous items by observing similar-minded people’s behavior.
3. They can make real quality assessment of items by considering other people’s experience.

**HARDWARE DESCRIPTION:**

**PERSONAL COMPUTER(PC):**

A personal computer (pc) is a multi-purpose computer whose size, capabilities, and price make it feasible for individual use. Since the early 1990s, Microsoft operating systems and Intel hardware dominated much of the personal computer market, first with MS-DOS and then with Microsoft Windows.





**SOFTWARE DESCRIPTION:**

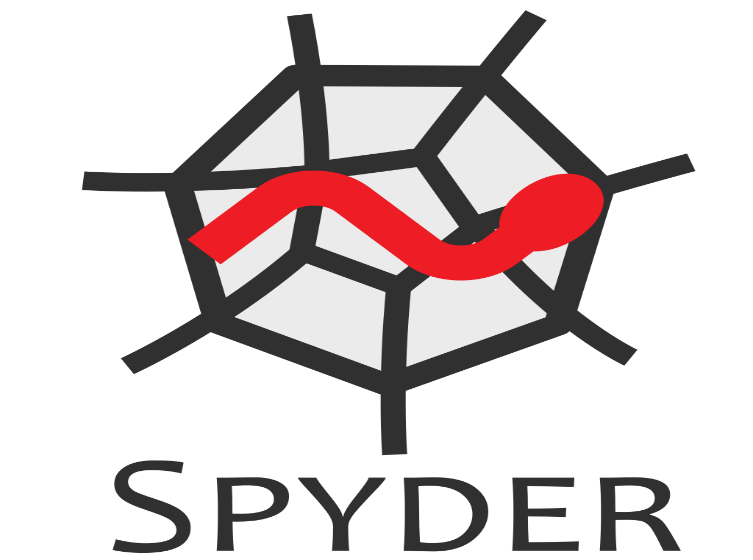
**ANACONDA:**

Anaconda is a free and open-source distribution of the Python and R programming languages for scientific computing, that aims to simplify package management and deployment. The distribution includes data-science packages suitable for Windows, Linux, and macOS.



**SPYDER:**

Spyder is an open source cross-platform integrated development environment for scientific programming in the Python language.

****

**SOFTWARE DEVELOPMENT:**

First of all, anaconda should be installed in your pc. And then using of Spyder I run the above code which is describes in code with outputs. Here I make a figure which is describes the how the system works exactly and how recommender system is recommend movie for user.

Algo1.png

**Figure:** System overview

**CODE:**

import numpy as np

import pandas as pd

column\_names = ['user\_id', 'item\_id', 'rating', 'timestamp']

df = pd.read\_csv('u.data', sep='\t', names=column\_names)

df.head()

movie\_titles = pd.read\_csv("Movie\_Id\_Titles")

movie\_titles.head()

df = pd.merge(df,movie\_titles,on='item\_id')

df.head()

import matplotlib.pyplot as plt

import seaborn as sns

sns.set\_style('white')

df.groupby('title')['rating'].mean().sort\_values(ascending=False).head()

df.groupby('title')['rating'].count().sort\_values(ascending=False).head()

ratings = pd.DataFrame(df.groupby('title')['rating'].mean())

ratings.head()

ratings['num of ratings'] = pd.DataFrame(df.groupby('title')['rating'].count())

ratings.head()

plt.figure(figsize=(10,4))

ratings['num of ratings'].hist(bins=70)

plt.figure(figsize=(10,4))

ratings['rating'].hist(bins=70)

moviemat = df.pivot\_table(index='user\_id',columns='title',values='rating')

moviemat.head()

ratings.sort\_values('num of ratings',ascending=False).head(10)

ratings.head()

starwars\_user\_ratings = moviemat['Star Wars (1977)']

liarliar\_user\_ratings = moviemat['Liar Liar (1997)']

starwars\_user\_ratings.head()

similar\_to\_starwars = moviemat.corrwith(starwars\_user\_ratings)

similar\_to\_liarliar = moviemat.corrwith(liarliar\_user\_ratings)

corr\_starwars = pd.DataFrame(similar\_to\_starwars,columns=['Correlation'])

corr\_starwars.dropna(inplace=True)

corr\_starwars.head()

corr\_starwars.sort\_values('Correlation',ascending=False).head(10)

corr\_starwars = corr\_starwars.join(ratings['num of ratings'])

corr\_starwars.head()

corr\_starwars[corr\_starwars['num of ratings']>100].sort\_values('Correlation',ascending=False).head()

corr\_liarliar = pd.DataFrame(similar\_to\_liarliar,columns=['Correlation'])

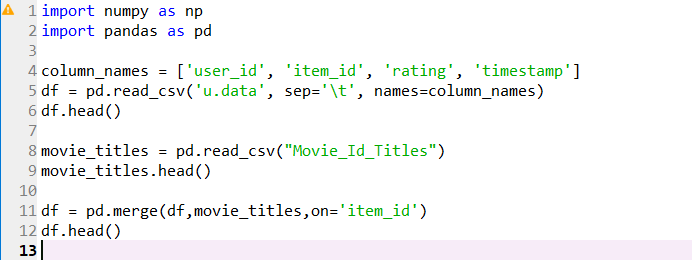
corr\_liarliar.dropna(inplace=True)

corr\_liarliar = corr\_liarliar.join(ratings['num of ratings'])

corr\_liarliar[corr\_liarliar['num of ratings']>100].sort\_values('Correlation',ascending=False).head()

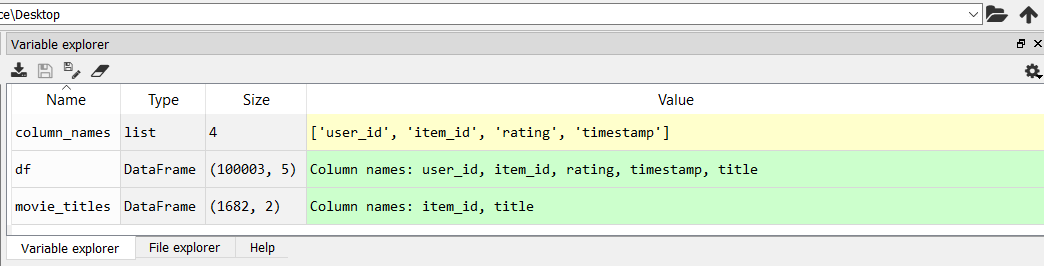
**CODE WITH OUTPUT:**

**Code (1-13):**

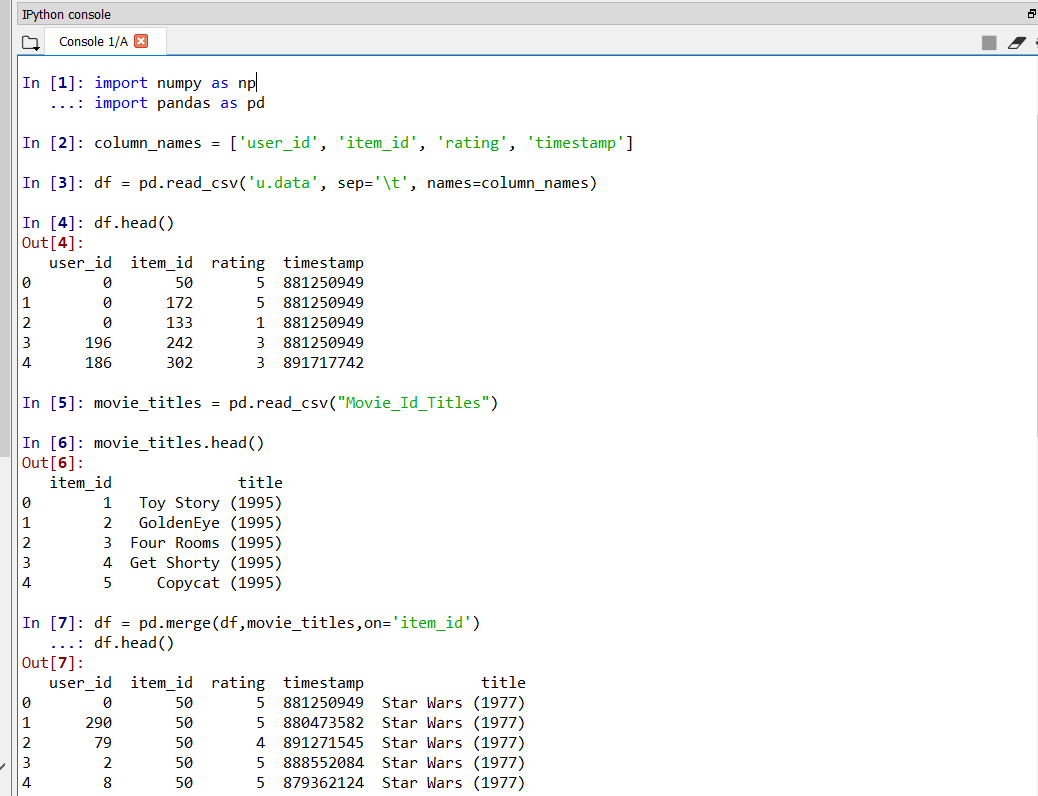
****

**Output (1-13):**

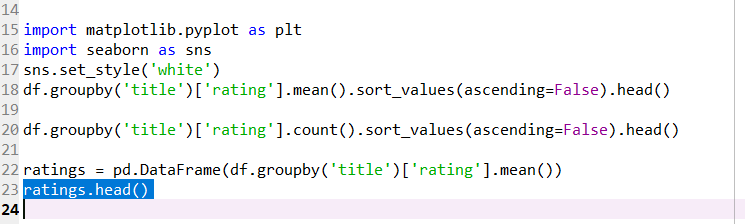
**Variable explorer:**

****

**IPython console:**

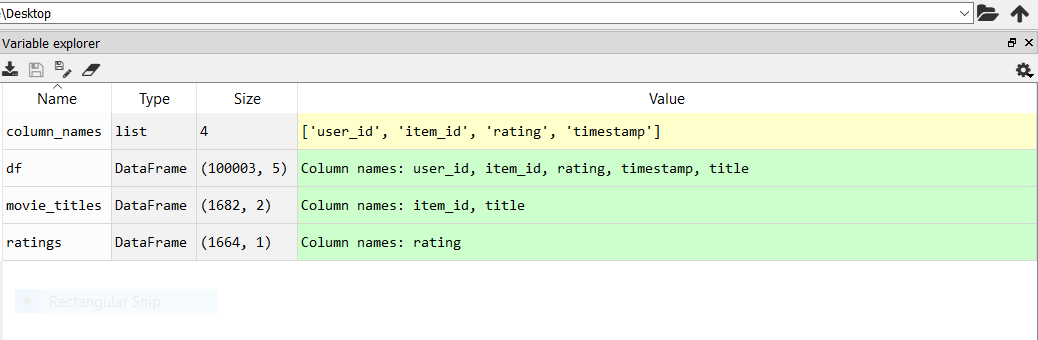
****

**Code (13-24):**

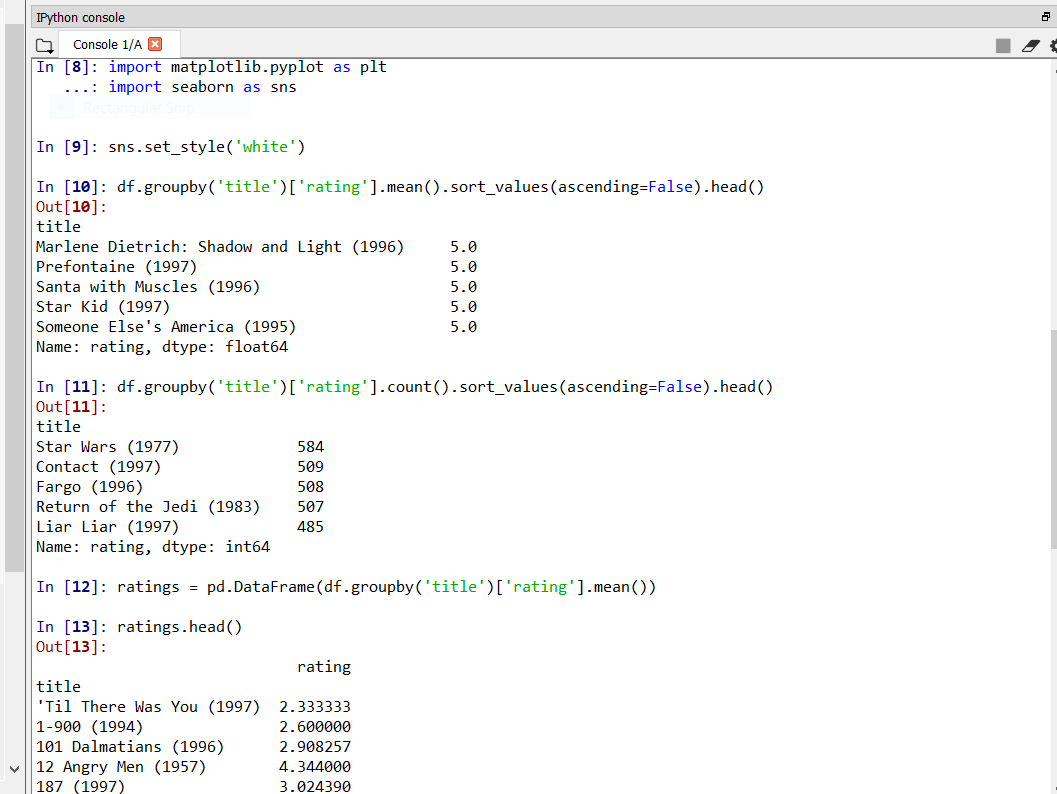
****

**Output (13-24):**

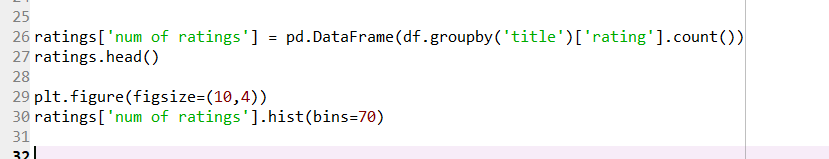
**Variable explorer:**

****

**IPython console:**

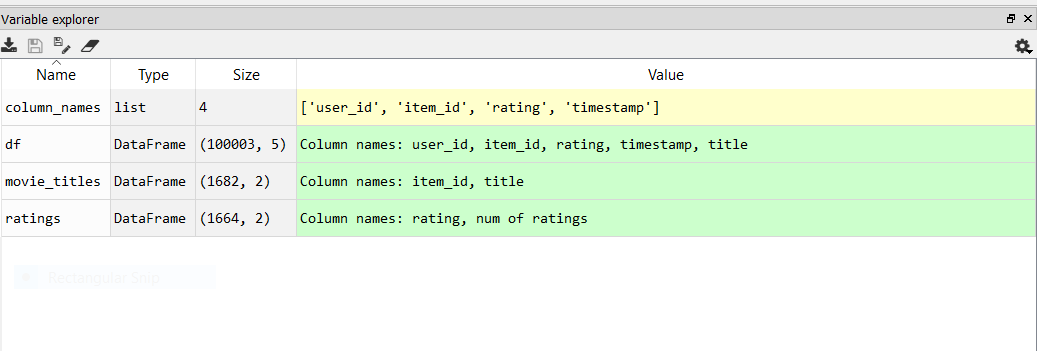
****

**Code (25-31):**

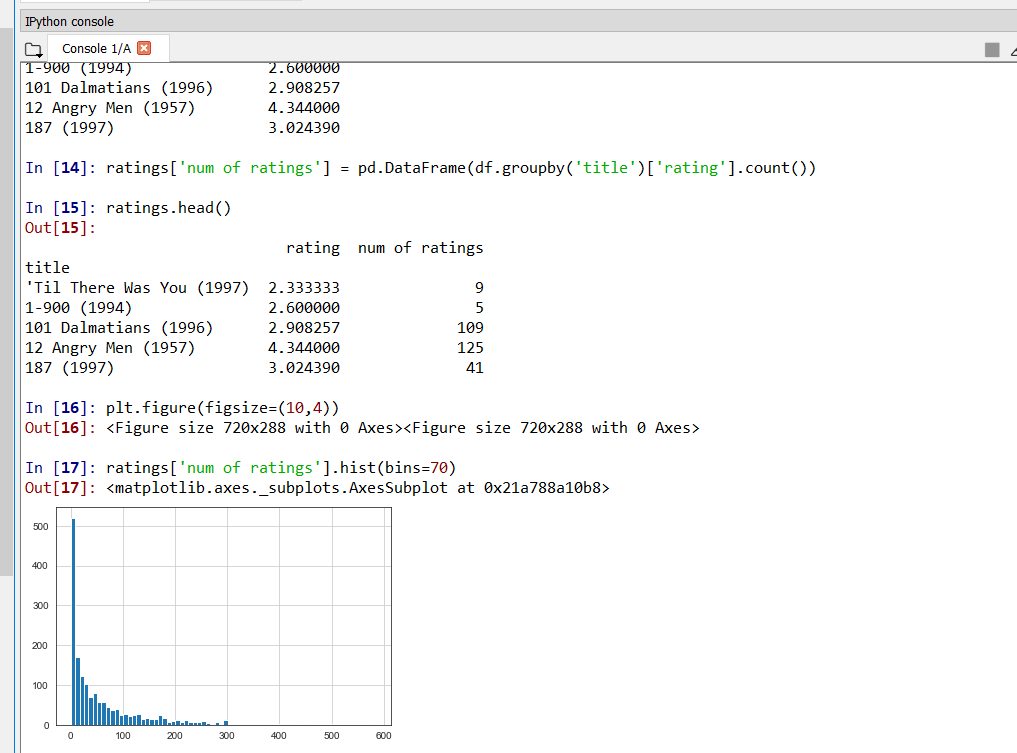
****

**Output (25-31):**

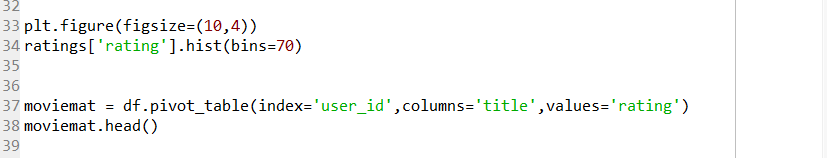
**Variable explorer:**

****

**IPython console:**

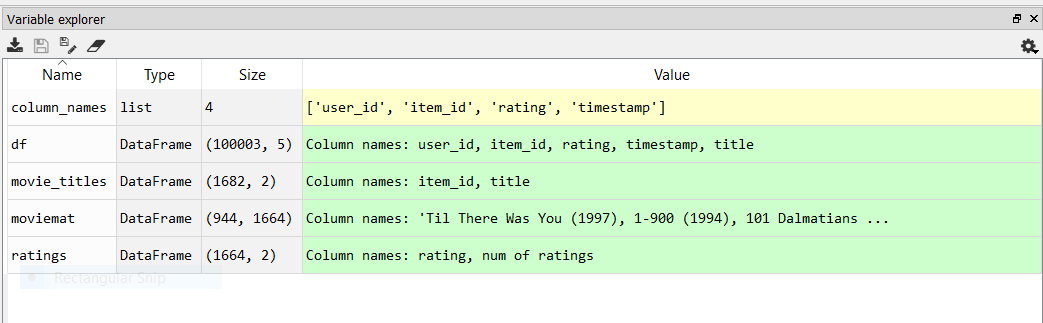
****

**Code (32-39):**

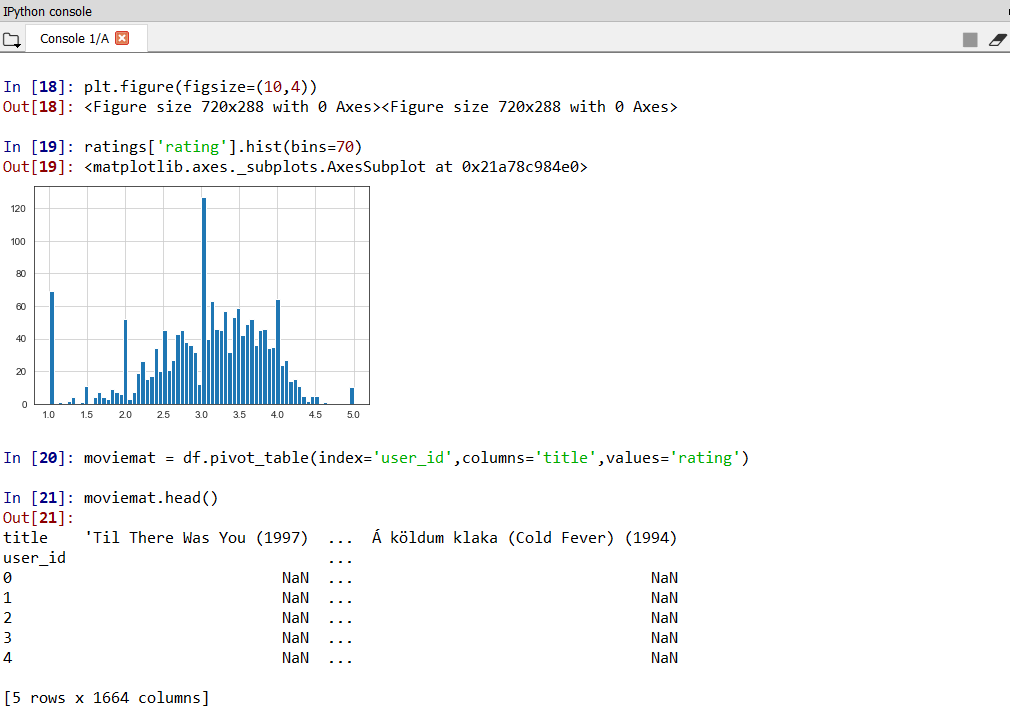
****

**Output (32-39):**

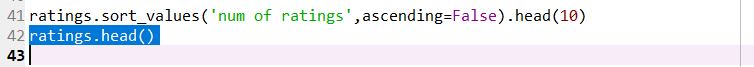
**Variable explorer:**

****

**IPython console:**

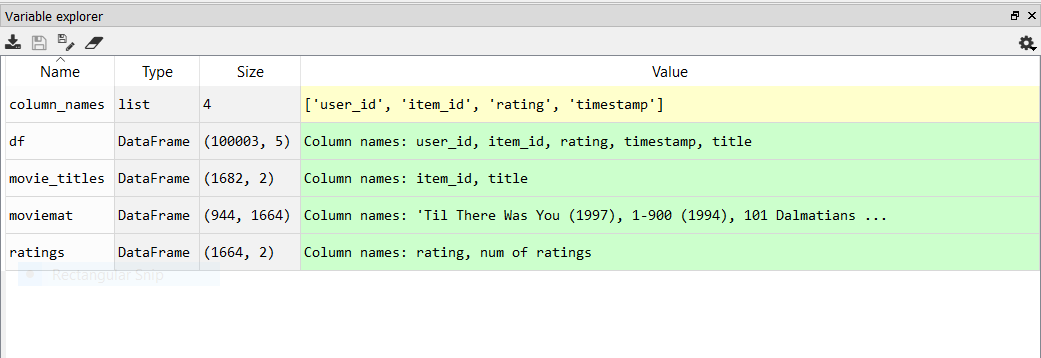
****

**Code (41-43):**

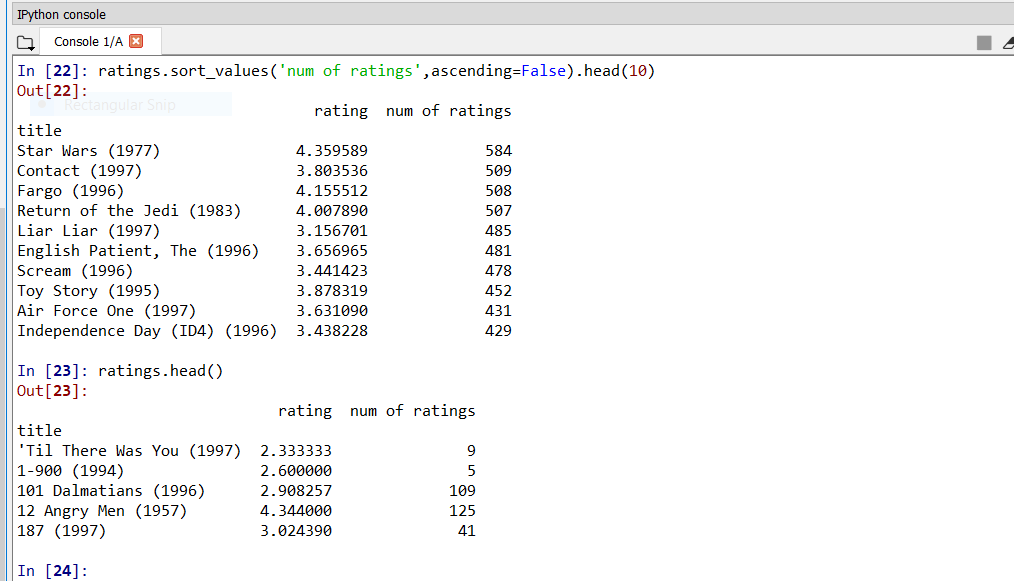
****

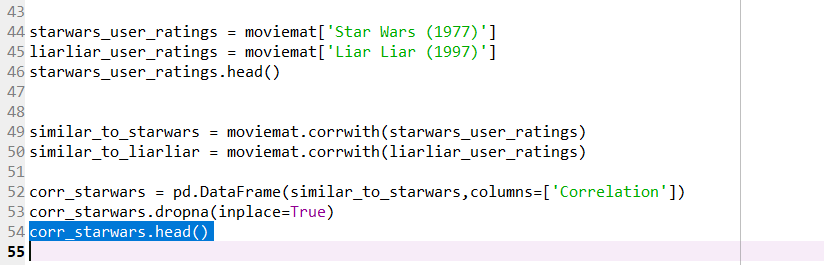
**Output (41-43):**

**Variable explorer:**

****

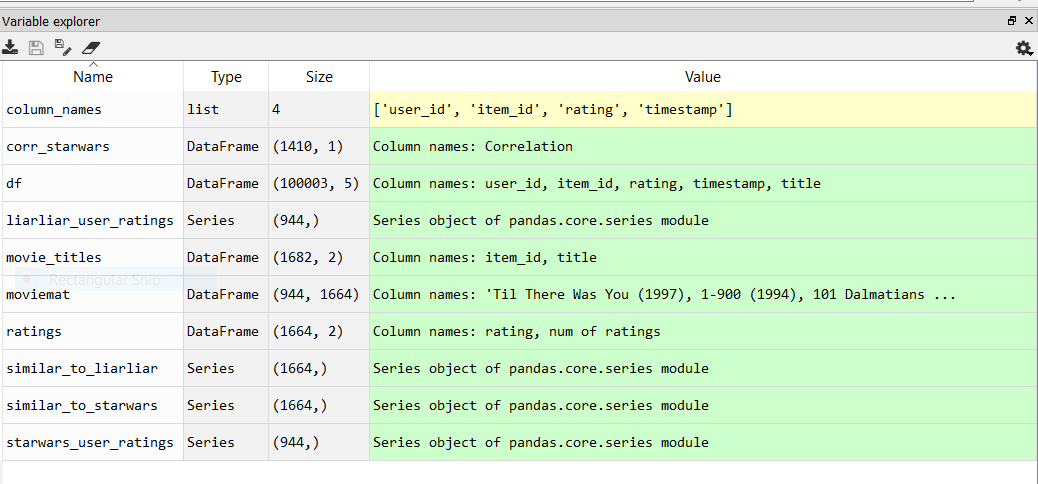
**IPython console:**

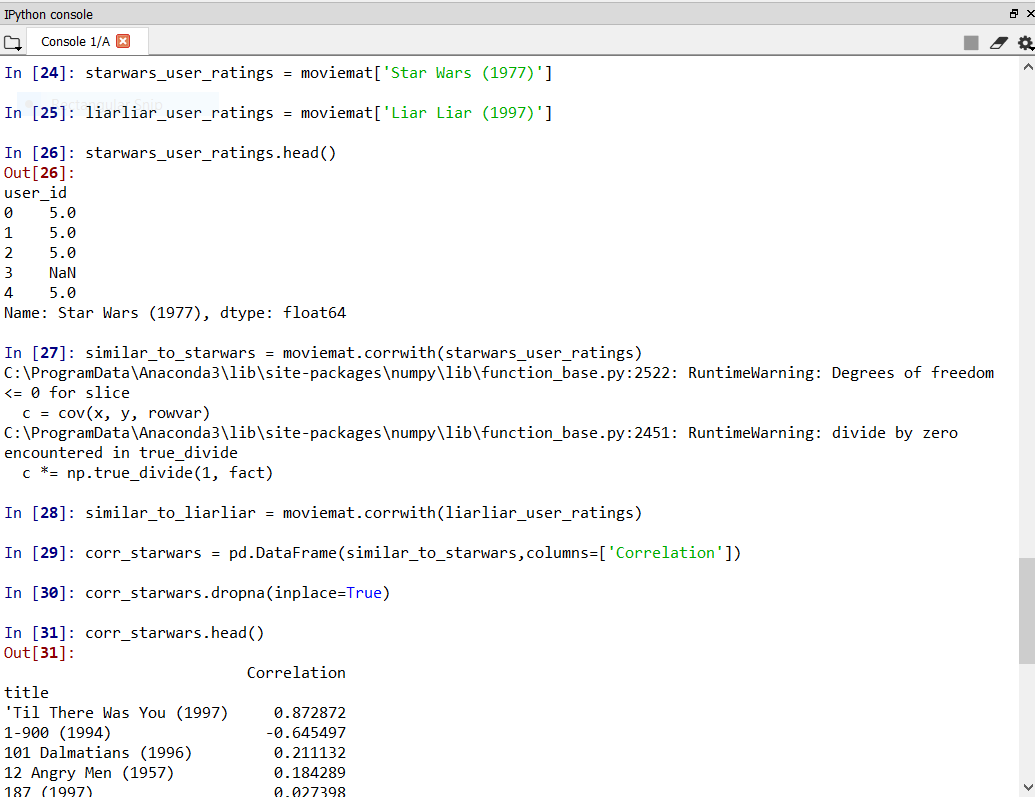
**Code (44-55):**

****

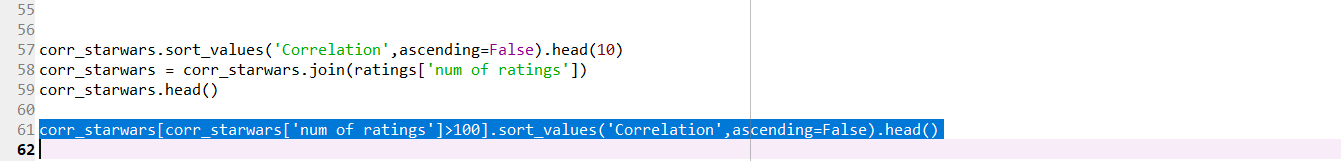
**Output (44-55):**

**Variable explorer:**

**IPython console:**

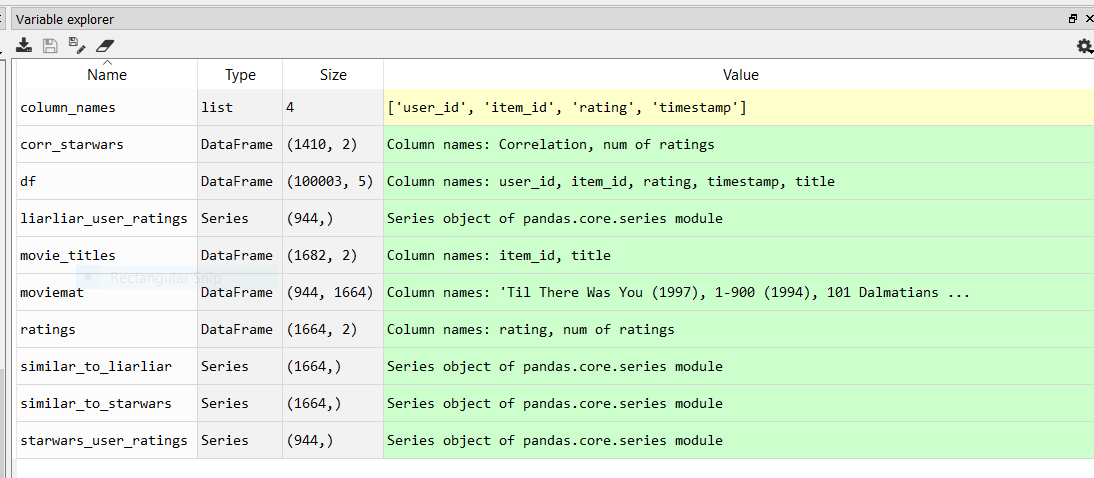
****

**Code (56-62):**

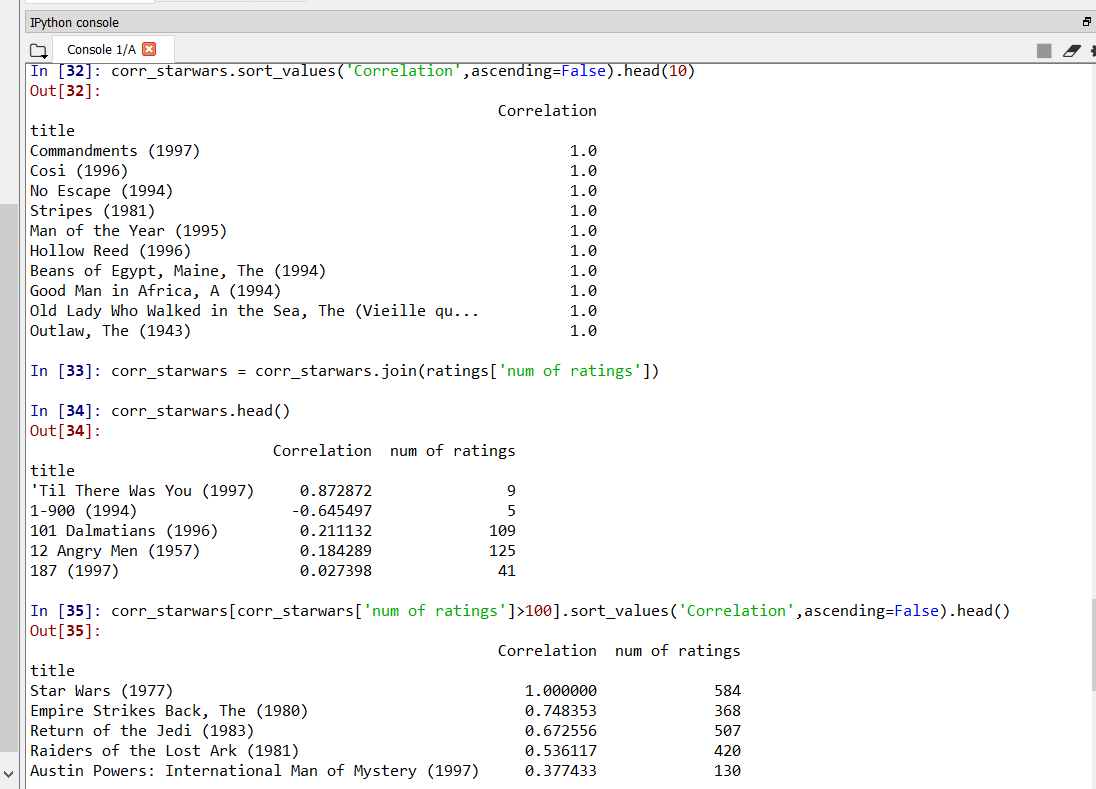
****

**Output (56-62):**

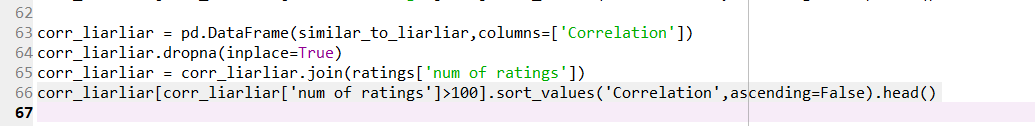
**Variable explorer:**

****

**IPython console:**

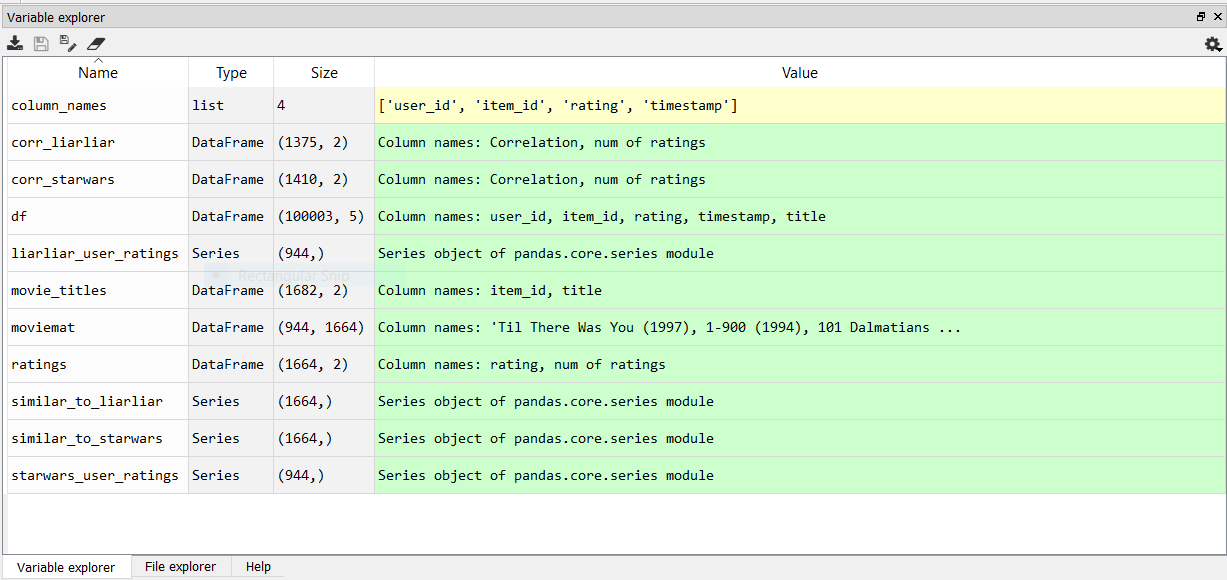
****

**Code (63-67):**

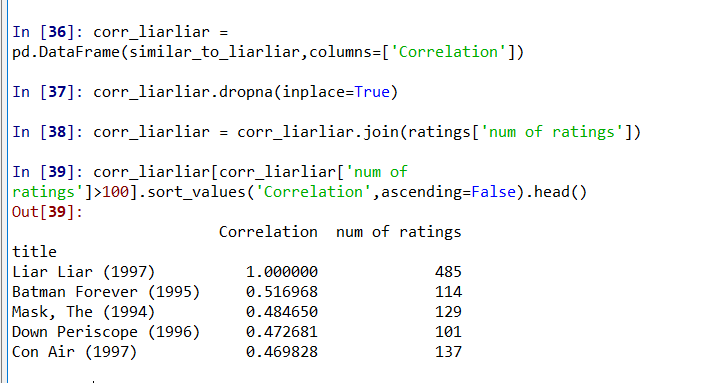
****

**Output (63-67):**

**Variable explorer:**

****

**IPython console:**

****

**CONCLUSION:**

Recommender systems open new opportunities of retrieving personalized information on the Internet. It also helps to alleviate the problem of information overload which is a very common phenomenon with information retrieval systems and enables users to have access to products and services which are not readily available to users on the system. We come up with a strategy that focuses on dealing with user’s personal interests and based on his previous reviews, movies are recommended to users. This strategy helps in improving accuracy of the recommendations. A personal profile is created for each user, where each user has access to his own history, his likes, ratings, comments, password modification processes. It also helps in collecting authentic data with improved accuracy and makes the system more responsive. Recommender systems are a powerful new technology for extracting additional value for a business from its user databases. These systems help users find items they want to buy from a business. Recommender systems benefit users by enabling them to find items they like.

## FUTURE SCOPE:

There are plenty of way to expand on the work done in this project. Firstly, the content-based method can be expanded to include more criteria to help categorize the movies the most obvious ideas is to add features to suggest movies with common actors, directors or writers in addition, movies released within the same time period could also receive a boost in likelihood for recommendation. Similarly, the movies total gross could be used to identify a user’s taste in terms of whether he/she prefers large release blockbusters, or smaller indie films. However, the above ideas may lead to overfitting, given that a user’s taste can be highly varied, and we only have a guarantee that 20 movies (less than 0.2%) have been reviewed by the user. In addition, we could try to develop hybrid methods that try to combine the advantages of both content-based methods and collaborative filtering into one recommendation system.

**WEBSITES:**

1. <https://www.anaconda.com/>
2. <https://en.wikipedia.org/wiki/Personal_computer>
3. <https://www.spyder-ide.org/>
4. <https://developers.google.com/machine-learning/recommendation/content-based/basics>
5. <https://realpython.com/build-recommendation-engine-collaborative-filtering/>