MOVIE RECOMMENDER SYSTEM

A SUMMER INTERN REPORT

Submitted by

SUMIT

Enrollment no : 20396403117

in partial fulfilment of Summer Internship for the award of the degree

of

BACHELOR OF TECHNOLOGY

IN

INFORMATION TECHNOLOGY

Under the supervision of

S. Rai

(Instructor) Udemy

MAHARAJA AGRASEN INSTITUTE OF TECHNOLOGY

ROHINI, NEW DELHI

CONTENTS

CERTIFICATE

Maharaja Agrasen Institute of Technology

To Whom It May Concern

I, **SUMIT**, Enrollment No. **20396403117** , a student of **Bachelors of Technology (IT), a class of 2017-21, Maharaja Agrasen Institute of Technology, Delhi** hereby declare that the Summer Training project report entitled **"Movie Recommender System"** is an original work and the same has not been submitted to any other Institute for the award of any other degree.

Date: 14th September 2019

Place: New Delhi

SUMIT

Enrollment No: 20396403117

Information Technology

5I789

DECLARATION

I hereby declare that the work, which is being presented in the training report, entitled "Movie Recommender System" in partial fulfillment for the award of Degree of "Bachelor of Technology" in Department of Information and Technology and submitted to the Department of Information and Technology, Maharaja Agrasen Institute of Technology, Rohini , New Delhi is a record of my own investigations carried under the Guidance of S. Rai.

I have not submitted the matter presented in this report anywhere for the award of any other Degree.

SUMIT

Information Technology

Enrolment no: 20396403117

ACKNOWLEDGEMENT

First and foremost, I wish to express my profound gratitude to S.Rai (Instructor) Udemy Online courses for giving me the opportunity to carry out my project . I find great pleasure to express my unfeigned thanks to the instructor for his invaluable guidance, support and useful suggestions at every stage of this project work.

No words can express my deep sense of gratitude to S. Rai without whom this project would not have turned up this way. My heartfelt thanks to him for his immense help and support, useful discussions and valuable recommendations throughout the course of my project work.

I wish to thank my respected faculty and my lab mates for their support.

Last but not the least I thank the almighty for enlightening me with his blessings.

SUMIT

Enrollment Number:20396403117

ABSTRACT

Recommendation System belongs to the class of Information Retrieval , Data Mining and Machine Learning . Recommender systems play a major role in today's ecommerce industry. Recommender systems recommend items to users such as books, movies, videos, electronic products and many other products in general. Recommender systems help the users to get personalized recommendations, helps users to take correct decisions in their online transactions, increase sales and redefine the users web browsing experience, retain the customers, enhance their shopping experience. Information overload problem is solved by search engines, but they do not provide personalization of data. Recommendation engines provide personalization. There are different type of recommender systems such as content-based, collaborative filtering, hybrid recommender system and keyword based recommender system. Variety of algorithms are used by various researchers in each type of recommendation system. Lot of work has been done on this topic, still it is a very favourite topic among data scientists. It also comes under the domain of data Science(Machine learning).

INTRODUCTION

A recommendation system is a type of information filtering system which attempts to predict the preferences of a user, and make suggests based on these preferences. There are a wide variety of applications for recommendation systems. These have become increasingly popular over the last few years and are now utilized in most online platforms that we use. The content of such platforms varies from movies, music, books and videos, to friends and stories on social media platforms, to products on e-commerce websites, to people on professional and dating websites, to search results returned on Google. Often, these systems are able to collect information about a users choices, and can use this information to improve their suggestions in the future. For example, Facebook can monitor your interaction with various stories on your feed in order to learn what types of stories appeal to you . Sometimes, recommender systems can make improvements based on the activities of a large number of people. For example, if Amazon observes that a large number of customers who buy the latest Apple Macbook also buy a USB-C-toUSB Adapter, they can recommend the Adapter to a new user who has just added a Macbook to his cart. Due to the advances in recommender systems, users constantly expect good recommendations . They have a low threshold for services that are not able to make appropriate suggestions. If a music streaming app is not able to predict and play music that the user likes, then the user will simply stop using it. This has led to a high emphasis by tech companies on improving their recommendation systems. However, the problem is more complex than it seems. Every user has different preferences and likes. In addition, even the taste of a single user can vary depending on a large number of factors, such as mood, season, or type of activity the user is doing. For example, the type of music one would like to hear while exercising differs greatly from the type of music he'd listen to when cooking dinner. Another issue that recommendation systems have to solve is the exploration versus exploitation problem . They must explore new domains to discover more about the user, while still making the most of what is already known about the user. Two main approaches are widely used for recommender systems. One is content-based filtering, where we try to profile the users interests using information collected, and recommend items based on that profile. The other is collaborative filtering, where we try to group similar users together and use information about the group to make recommendations to the user.

TECHNOLOGIES USED

Language Used : PYTHON

version:3.7.4

Python is an interpreted, object-oriented, high-level programming language with dynamic semantics. Its high-level built in data structures, combined with dynamic typing and dynamic binding, make it very attractive for Rapid Application Development, as well as for use as a scripting or glue language to connect existing components together. Python's simple, easy to learn syntax emphasizes readability and therefore reduces the cost of program maintenance. Python supports modules and packages, which encourages program modularity and code reuse. The Python interpreter and the extensive standard library are available in source or binary form without charge for all major platforms, and can be freely distributed.

- Pandas : pandas is a software library written for the Python programming language for data manipulation and analysis. In particular, it offers data structures and operations for manipulating numerical tables and time series.

- NumPy : NumPy is a library for the Python programming language, adding support for large, multi-dimensional arrays and matrices, along with a large collection of high-level mathematical functions to operate on these arrays.

MACHINE LEARNING

implemented on jupyter notebook (anaconda navigator)

Machine learning is an application of artificial intelligence (AI) that provides systems the ability to automatically learn and improve from experience without being explicitly programmed. Machine learning focuses on the development of computer programs that can access data and use it to learn for themselves.

The process of learning begins with observations or data, such as examples, direct experience, or instruction, in order to look for patterns in data and make better decisions in the future based on the examples that we provide. **T**he primary aim is to allow the computers learn automatically without human intervention or assistance and adjust actions accordingly Classification of Machine Learning Machine learning implementations are classified into three major categories, depending on the nature of the learning “signal” or “response” available to a learning system which are as follows:-

1. Supervised learning: When an algorithm learns from example data and associated target responses that can consist of numeric values or string labels, such as classes or tags, in order to later predict the correct response when posed with new examples comes under the category

of Supervised learning. This approach is indeed similar to human learning under the supervision of a teacher. The teacher provides good examples for the student to memorize, and the student then derives general rules from these specific examples.

2. Unsupervised learning: When an algorithm learns from plain examples without any associated response, leaving to the algorithm to determine the data patterns on its own. This type of algorithm tends to restructure the data into something else, such as new features that may represent a class or a new series of un-correlated values. They are quite useful in providing humans with insights into the meaning of data and new useful inputs to supervised machine learning algorithms. As a kind of learning, it resembles the methods humans use to figure out that certain objects or events are from the same class, such as by observing the degree of similarity between objects. Some recommendation systems that you find on the web in the form of marketing automation are based on this type of learning.

3. Reinforcement learning : It is about taking suitable action to maximize reward in a particular situation. It is employed by various software and machines to find the best possible behavior or path it should take in a specific situation. Reinforcement learning differs from the supervised learning in a way that in supervised learning the training data has the answer key with it so the model is trained with the correct answer itself whereas in reinforcement learning, there is no answer but the reinforcement agent decides what to do to perform the given task. In the absence of training dataset, it is bound to learn from its experience. In the human world, it is just like learning by trial and error. Errors helps to learn because they have a penalty added (cost, loss of time, regret, pain, and so on).

Categorizing on the basis of required Output Another categorization of machine learning tasks arises when one considers the desired output of a machine-learned system:

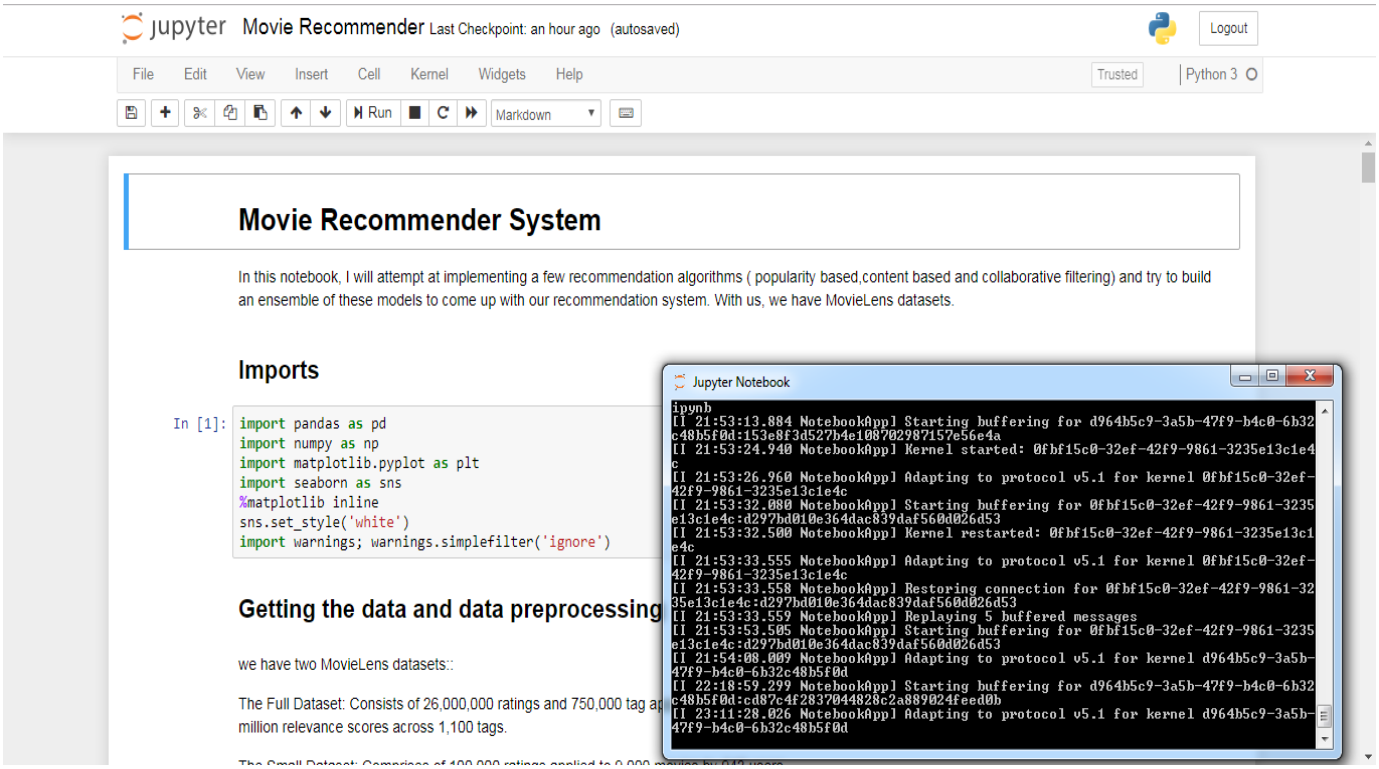
1. Classification : When inputs are divided into two or more classes, and the learner must produce a model that assigns unseen inputs to one or more (multi-label classification) of these classes. This is typically tackled in a supervised way. Spam filtering is an example of classification, where the inputs are email (or other) messages and the classes are “spam” and “not spam”. Other examples include recognizing the breed of a dog with its image.

2. Regression : Which is also a supervised problem, A case when the outputs are continuous rather than discrete. Example include prediction the price of a house, estimating the sales in a particular time period,etc.

3. Clustering : When a set of inputs is to be divided into groups. Unlike in classification, the groups are not known beforehand, making this typically an unsupervised task. Example includes dividing the people in groups on the basis of their preferences.

PLATFORMS USED

Jupyter Notebook (Anaconda Navigator)



PROJECT DESCRIPTION

Recommender System is a system that seeks to predict or filter preferences according to the user's choices. Recommender systems are utilized in a variety of areas including movies, music, news, books, research articles, search queries, social tags, and products in general.

Recommender systems produce a list of recommendations in any of the two ways –

Simple filtering: The Simple Recommender offers generalized recommendations to every user based on movie popularity and (sometimes) genre. The basic idea behind this recommender is that movies that are more popular and more critically acclaimed will have a higher probability of being liked by the average audience. This model does not give personalized recommendations based on the user.

The implementation of this model is extremely trivial. All i did was sort our movies based on ratings and popularity and display the top movies of our list .

Content-based filtering: The recommender we built in the previous section suffers some severe limitations. For one, it gives the same recommendation to everyone, regardless of the user's personal taste. For instance, consider a person who loves Dilwale Dulhania Le Jayenge, My Name is Khan and Kabhi Khushi Kabhi Gham. One inference we can obtain is that the person loves the actor Shahrukh Khan and the director Karan Johar. Even if s/he were to access the romance chart, s/he wouldn't find these as the top recommendations. To personalise our recommendations more, I am going to build an engine that computes similarity between movies based on certain metrics and suggests movies that are most similar to a particular movie that a user liked. Since we will be using movie metadata (or content) to build this engine, this also known as Content Based Filtering. Content-based filtering approaches uses a series of discrete characteristics of an item in order to recommend additional items with similar properties. Content-based filtering methods are totally based on

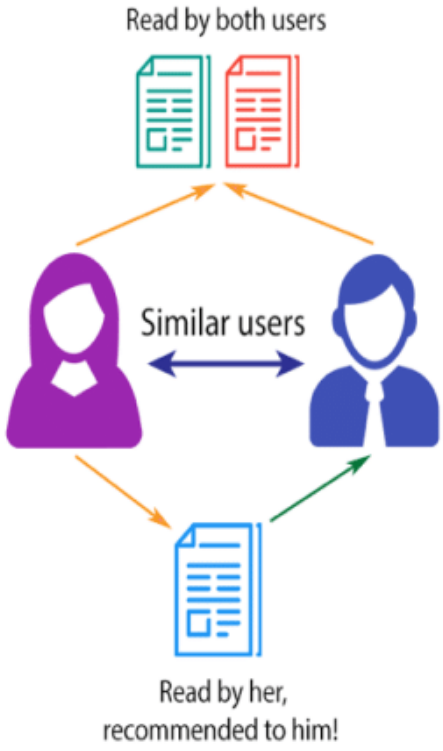
a description of the item and a profile of the user's preferences. It recommends items based on user's past preferences.

Collaborative filtering: Collaborative filtering produces recommendations based on the knowledge of users' attitude to items, that is it uses the "wisdom of the crowd" to recommend items. Content-based recommender systems focus on the attributes of the items and give you recommendations based on the similarity between them. Our content based engine suffers from some severe limitations. It is only capable of suggesting movies which are close to a certain movie. That is, it is not capable of capturing tastes and providing recommendations across genres.Also, the engine that we built is not really personal in that it doesn't capture the personal tastes and biases of a user. Anyone querying our engine for recommendations based on a movie will receive the same recommendations for that movie, regardless of who s/he is.

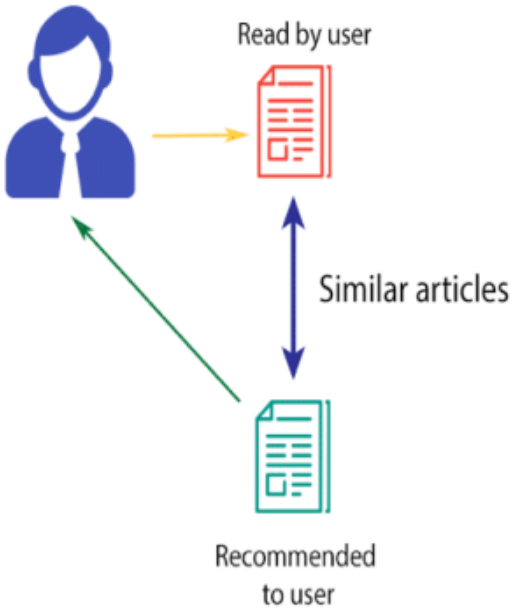
Therefore, in this section, we will use a technique called Collaborative Filtering to make recommendations to Movie Watchers.Collaborative Filtering is based on the idea that users similar to a me can be used to predict how much I will like a particular product or service those users have used/experienced but I have not.actual mathematics behind recommender systems is pretty heavy in Linear Algebra.Collaborative Filtering

In general, Collaborative filtering (CF) is more commonly used than content-based systems because it usually gives better results and is relatively easy to understand (from an overall implementation perspective).CF can be divided into Memory-Based Collaborative Filtering and Model-Based Collaborative filtering

COLLABORATIVE FILTERING



CONTENT-BASED FILTERING



MOVIE LENS DATASET

https://grouplens.org/datasets/movielens/

we have two MovieLens datasets:

1)The Full Dataset: Consists of 26,000,000 ratings and 750,000 tag applications applied to 45,000 movies by 270,000 users. Includes tag genome data with 12 million relevance scores across 1,100 tags.

2)The Small Dataset: Comprises of 100,000 ratings applied to 9,000 movies by 943 users.

Recommender systems will make use of the small dataset (due to the computing power I possess being very limited)

After some pre-processing our data set looks like:

```
In [8]: df=pd.merge(df,movie_titles.drop(columns=['release date' , 'video release date', 'IMDb URL ', 'unknown','Action' ,'Adventure',
```

```
In [47]: df.head(5)
```

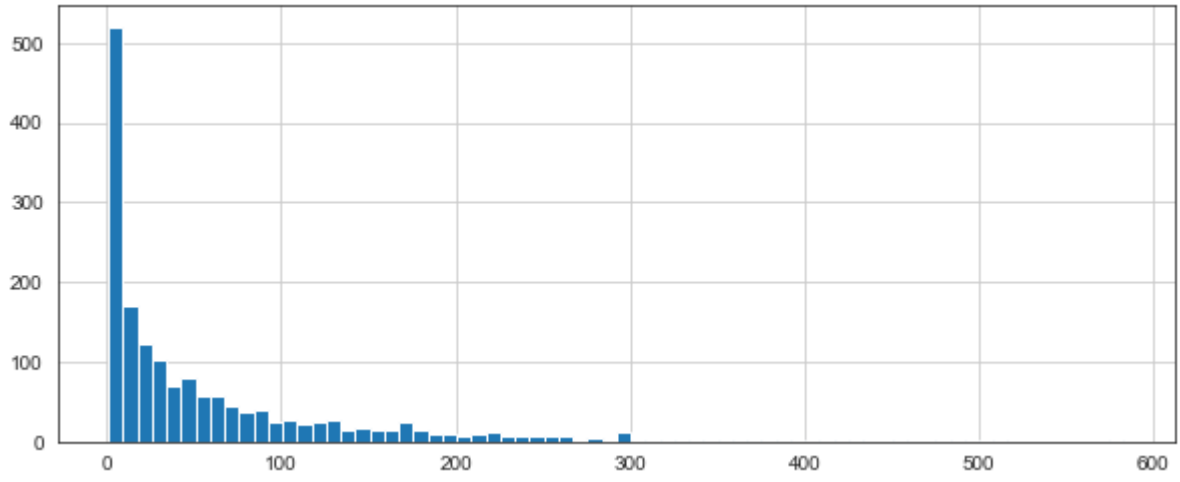
Out[47]:

	user_id	item_id	rating	timestamp	movie title
0	196	242	3	881250949	Kolya (1996)
1	63	242	3	875747190	Kolya (1996)
2	226	242	5	883888671	Kolya (1996)
3	154	242	3	879138235	Kolya (1996)
4	306	242	5	876503793	Kolya (1996)

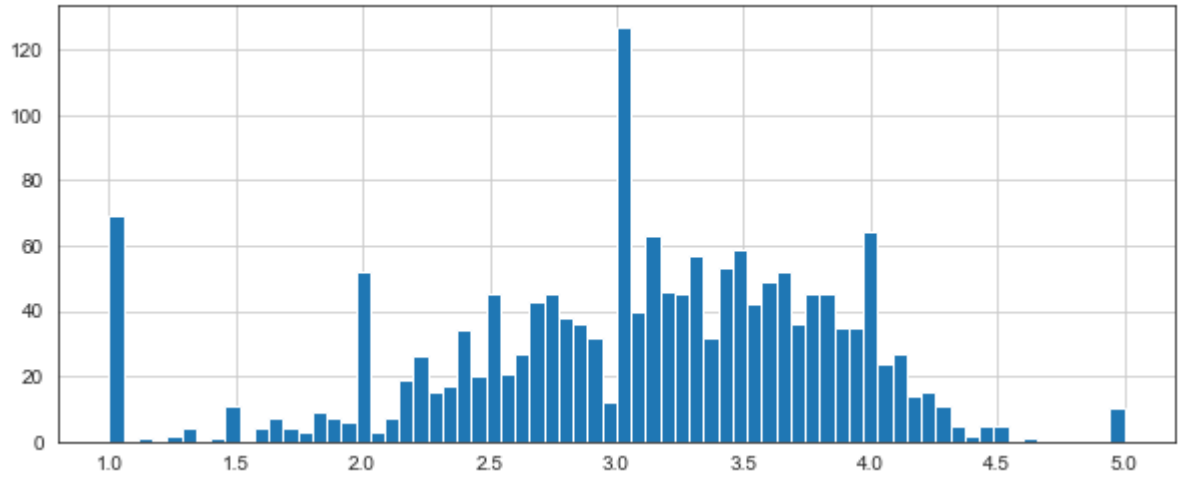
EDA ON DATASET

Histogram of number of ratings to a movie

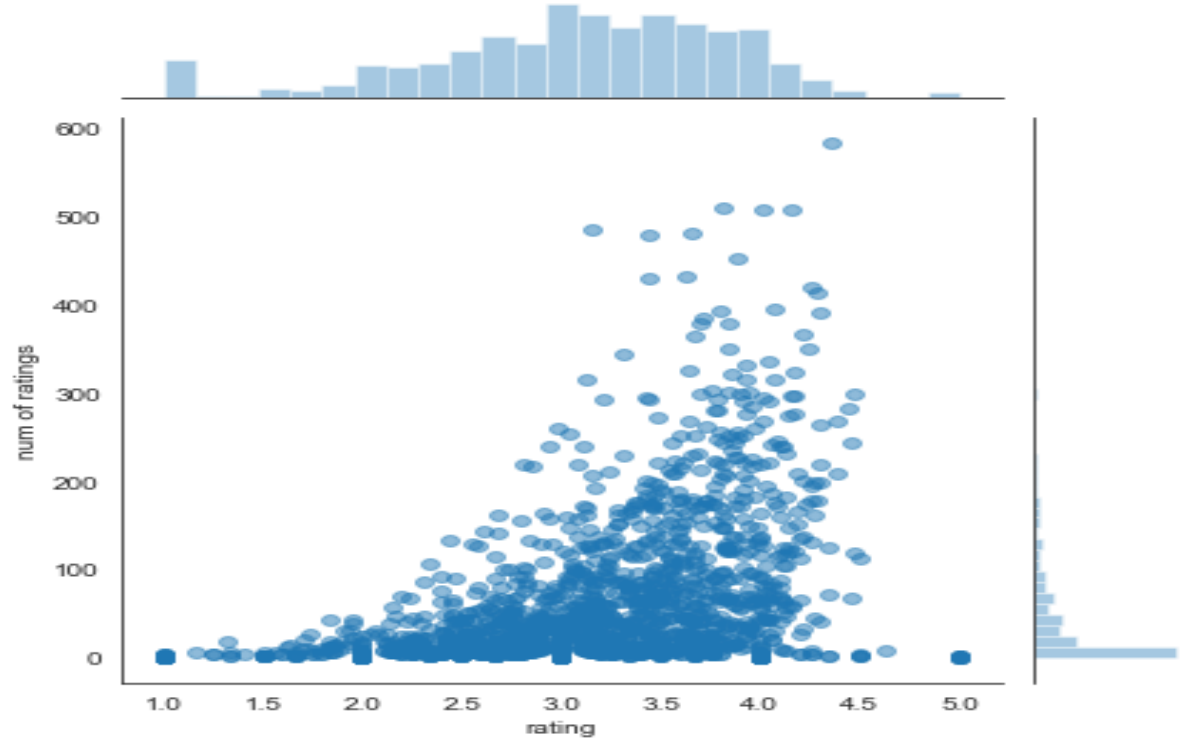
(x-axis = num of rating given to a movie ,y-axis=count)



Histogram of rating given by user :(x-axis = rating (range0-5),y-axis=count)



plot shows both the above histogram joined together by sns.jointplot a function of seaborn library



SNAP SHOTS OF THE PROJECT

- **Simple Recommender :**

Suggestions are:

```
In [21]: temp['rating'].sort_values(ascending=False).head(10)
```

```
Out[21]: movie title
Close Shave, A (1995)                4.491071
Schindler's List (1993)              4.466443
Wrong Trousers, The (1993)           4.466102
Casablanca (1942)                    4.456790
Shawshank Redemption, The (1994)     4.445230
Rear Window (1954)                   4.387560
Usual Suspects, The (1995)           4.385768
Star Wars (1977)                     4.358491
12 Angry Men (1957)                  4.344000
Citizen Kane (1941)                  4.292929
Name: rating, dtype: float64
```

Thank you

Collaborative Filtering Recommendations are :

```
In [46]: get_high_recommended_movies(217)
```

```
Out[46]: {174, 385, 405}
```

- **Content Based recommender**

therefore similar movies are::

```
In [30]: corr_[corr_['num of ratings']>100].sort_values('Correlation',ascen
```

Out[30]:

movie title	Correlation	num of ratings
Young Guns (1988)	1.0	101
Devil's Advocate, The (1997)	1.0	188
Die Hard 2 (1990)	1.0	166
Die Hard: With a Vengeance (1995)	1.0	151
Seven (Se7en) (1995)	1.0	236
Scream (1996)	1.0	478
Rumble in the Bronx (1995)	1.0	174
Rock, The (1996)	1.0	378
River Wild, The (1994)	1.0	146
Reservoir Dogs (1992)	1.0	148

Thank you

- **Collaborative Filtering**

```
In [32]: column_name=['userId','movieId','rating','timestamp']
         ratings=pd.read_csv('u.data',sep='\t',names=column_name)

In [33]: ratings.head()
```

Out[33]:

	userId	movieId	rating	timestamp
0	196	242	3	881250949
1	186	302	3	891717742
2	22	377	1	878887116
3	244	51	2	880606923
4	166	346	1	886397596

```
In [34]: user_item = ratings.groupby(['userId', 'movieId'])['rating'].first().unstack(fill_value=0.0)

In [35]: user_item.head()
```

Out[35]:

movieId	1	2	3	4	5	6	7	8	9	10	...	1673	1674	1675	1676	1677	1678	1679	1680	1681	1682
userId																					
1	5.0	3.0	4.0	3.0	3.0	5.0	4.0	1.0	5.0	3.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
2	4.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	2.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
3	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	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
5	4.0	2.0	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

RESULTS

All the models run successfully and efficiently with almost no error are capable of prediction movies according to user demand and taste.

In this project, I have built 3 different recommendation engines based on different ideas and algorithms. They are as follows:

Simple Recommender:

This system used overall Vote Count and Vote Averages to build Top Movies Charts, in general and for a specific genre.

Content Based Recommender:

We built content based engines that took movie overview and taglines as input to come up with predictions. We also devised a simple filter to give greater preference to movies with more votes and higher ratings.

Collaborative Filtering:

We used the powerful Surprise Library to build a collaborative filter based on single value decomposition(SVD). The RMSE obtained was less than 1 and the engine gave estimated ratings for a given user and movie.

REFERENCES

[1] Kaggle

https://www.kaggle.com/

[2] Google News Personalization: Scalable Online Collaborative Filtering; Das et al; https://www2007.org/papers/paper570.pdf

[3] geeksforgeeks

https://www.geeksforgeeks.org/python-implementation-of-movie-recommender-system/

[4] github and medium

https://alyssaq.github.io/2015/20150426-simple-movie-recommender-using-svd