

# **MOVIE RECOMMENDATION SYSTEM**

## **A Project Report**

*Submitted in partial fulfillment of the requirements for the award of the  
degree of*

### **BACHELOR OF TECHNOLOGY IN COMPUTER SCIENCE ENGINEERING**

*Under the supervision  
of*

**Dr. Amit Kumar**

**Assistant Professor (Senior Grade)**

*By*

**AYUSH AGRAWAL (161279)**



**JAYPEE UNIVERSITY OF INFORMATION TECHNOLOGY**

**WAKNAGHAT, SOLAN – 173234**

**HIMACHAL PRADESH, INDIA**

**DEC–2019**

## CERTIFICATE

I hereby declare that the work presented in this report entitled “Movie Recommendation System” in partial fulfillment of the requirements for the award of the degree of Bachelor of Technology in Computer Science and Engineering submitted in the department of Computer Science & Engineering and Information Technology, Jaypee University of Information Technology Waknaghat is an authentic record of my own work carried out over a period from August 2019 to Dec 2019 under the supervision Dr. Amit Kumar , Associate Professor

(Senior Grade).

The matter embodied in the report has not been submitted for the award of any other degree or diploma.

*Ayush Agrawal*

(Student Signature)

Student Name, Rollno.

This is to certify that the above made statement made by the candidate is that to the best of my knowledge.

*Amit Kumar*

(Supervisor Signature)

Supervisor's Name: Dr. Amit Kumar

Designation: Assistant Professor (Senior Grade)

Department Name: Computer Science and Engineering

Dated:

## **ACKNOWLEDGEMENT**

I would like to express my greatest gratitude to the people who have helped & supported us throughout my project. I am grateful to my mentor **Dr. Amit Kumar** for his continuous support for the project, from initial advice & contacts in the early stages of conceptual inception & through ongoing advice & encouragement to this day.

A special thank of us to our group members who helped each other in completing the project & exchanged their interesting ideas, thoughts & made this project easy and accurate.

Date:

**AYUSH AGRAWAL (161279)**

## TABLE OF CONTENTS

Certificate.....		
Acknowledgement.....		
Chapter-1	INTRODUCTION	5-10
	<ul style="list-style-type: none"><li>• Problem Statement</li><li>• Objectives</li><li>• Methodology</li></ul>	
Chapter-2	LITERATURE SURVEY	11-25
	<ul style="list-style-type: none"><li>• Research Paper 1</li><li>• Research Paper 2</li><li>• Research Paper 3</li></ul>	
Chapter-3	SYSTEM DEVELOPMENT	26-31
	<ul style="list-style-type: none"><li>• System Requirements</li><li>• Technical Stack</li><li>• Flow Chart</li><li>• Utility Matrix</li><li>• Predictions</li></ul>	
Chapter-4	PERFORMANCE ANALYSIS	32-50
Chapter-5	CONCLUSIONS	51
References .....		52

# CHAPTER I

## INTRODUCTION

- **Movie Recommendation system:**

In modern world or the coming generation the web has completely transformed into a noteworthy bit of human life, individuals are seen normally confronting the issue of settling on a choice. Straight forwardly from scanning for an inn to looking for insightful hypothesis options, there is a vast measure of information available. To empower the customers to adjust to this information impact, associations have alluded to proposal structures to tackle with their customers. The test in the district of proposal structures has been proceeding with delayed now, yet the top notch despite everything remainders high due to the affluence of helpful and robust applications and the issues. Various types of e-proposition approaches made and put to utilize gives a proposition structure of distributions at Flipkart, of shows at Amazon Prime, etc. These gave contribution to riches for a portion identified with electronic business locales, (for example, Flipkart.com) and do motion pictures.

Recommender Systems make recommendations and the system will recognize the shows and movies according to their choice and may in like method give results instantly/or on extra levels, a comprehended/unquestionable info. These undertakings identified with customers and customers' reactions can put a route for this proposal set and will be used if there should arise an occurrence of creating new recommendations for following customer structure associations. The budgetary capacity of hypotheses recommender structures has driven most likely the best web business destinations (for example : flipkart.com ,amazon.in) and e-commerce recruited association. Amazon

Prime brands those systems very exceptional section for these information and systems. Huge bore altered endorsements add-on other estimation for customer occurrences. Work redid recommender softwares are starting late associated with it give different kinds of adjusted info to their registered customers. These systems can be associated in different sorts of usages as per the item applied upon and are extraordinarily typical these days. We can segregate the recommender structures in two general orders:

Given below are the two types of Movie Recommendation Systems

- I. Content-based sifting method
- II. Collaborative sifting method

### **a) Collaborative filtering method:**

Collaborative filtering method proposes info are based over the similarity of ratings of a particular movie, among different registered users. Collaborative filtering, Community oriented sifting framework proposes things are reliant over similarity strategies among customers possibly things. Cooperative separating, otherwise called social sifting, channels data utilizing others' suggestions. Customers who have recently concurred on the assessment of specific components are bound to acknowledge it again later on. For instance, an individual who needs to see a film can request suggestions from companions. The proposals of certain companions with comparable interests are more solid than others. This data is utilized on what film to watch. Communitarian Filtering, doesn't need whatever else with the exception of clients' past inclination on a lot of things. As it depends on authentic information, the center theory here is that the clients who have concurred earlier will be in general additionally concur later on. This system proposes such stuffs which were particular from comparable kind of customers. Community sifting has a few good conditions

1. In Content Based Filtering customer does assessments subsequently for authentic viewpoint and estimation and prediction of item is done.
2. It predicts precise results and proposals since proposals depend on client's comparability as opposite to data proportionality.

## **b) Content-based filtering:**

Content-based filtering is created based on keeping in mind the profile of the client's affinity and the initial database data. In this, to precisely predict the things we have castoff the ratings recorded by the clients to the movies or TV Series and users favored likes and dislikes to the shows. And by the end of the day at the background of the software using Collaborative Filtering method and estimations endorse those things or like those things that were favored before previously. It calculates and predicts the new shows and or earlier based predicted things and proposes best movies or shows based on his likes and dislikes items. It uses different strategies and projection methods on different areas of use. This method is mostly used in Hybrid Recommender Systems. An older calculations or the predictions of motion pictures or movies through MOVIEGEN datasets have different implementations, for ex, this demonstrates the movement of users request, with what had been searched in the past is also saved in the history or in the database. On learning these mistakes, we have developed Movie Recommendation System using Pearson Correlation Method, an advance method based structure that predicts and outputs movies to customers reliant for a data given by the customers in the past and the present examination, a customer is given the decision to pick his choices from a great deal of qualities based on No. Of Ratings and Rating to each movie, etc. We update the users choices in the database and computes a new set of results from the new data provided and based on the choices of the past visited history of customers. The software is developed using Python and a simple user interface.

### **• Problem Statement**

- This framework will prescribe movies and shows to clients.  
This strategy will give increasingly exact outcomes contrasted



with existing frameworks. The present framework takes a shot at the evaluations of individual clients. It might be futile for clients who have various tastes. The suggestions appeared by the framework might be distinctive for each client. This framework processes similitude between various clients and afterward suggests the film in like manner appraisals given by various clients of comparative tastes. This will give an exact suggestion to the client. It depends on a web just as Android framework where a Movie web administration that offers types of assistance to the client to rate movies, see proposals compose remarks and peruse comparative movies.

We have used the Movie Lens dataset, which has a gigantic no of evaluations gave by clients to films.

- **Objectives**

The main goal of or project is:

- To develop a robust set of configuration and a good model that predicts and gives best results with min error and maximum accuracy to the user.
- Training the developed model. Training is done to generalise the prediction of future requests.
- Testing the model. Testing of the model is done to reduce to detect any error and is done many times to increase efficiency of the developed model

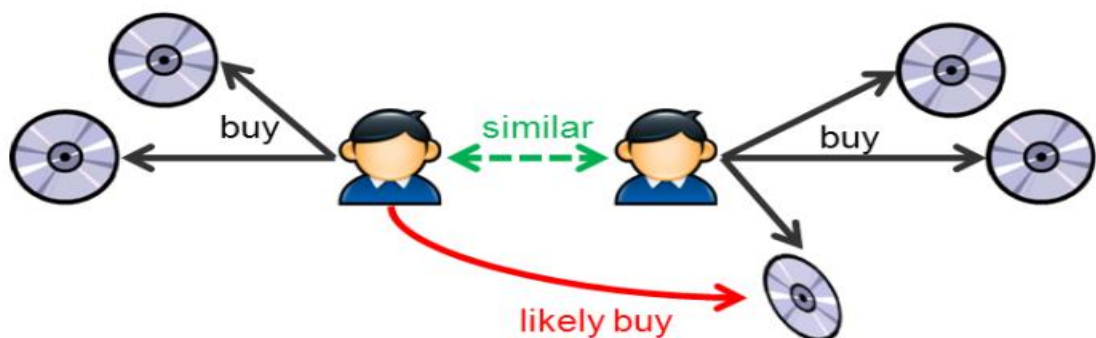
- To predict movies to the customers views by the past watched or searched items on the portal and using his/her history.

- **Methodology**

Nearest Neighbourhood algorithm approach used to build recommendation systems is explained below :

- **Nearest Neighborhood:**

The standard strategy for Collaborative Filtering for the movies is given by the Nearest Neighbourhood calculation. There are client grounded CBF and items grounded CBF. How about we view Customer Based Filtering. We have a  $x \times y$  lattice of client's appraisals, with customer  $q_i$ ,  $T = 1 \dots x$  and thing  $p=r_j$ ,  $U=1, \dots y$ . Presently we need to anticipate the rating of the movie from the movie matrix using  $z_{ij}$  if current client  $T$  didn't rate an item  $U$ . The procedure is to find the similarities between current client  $T$  and all the other customers, Please find the top  $M$  items from the results of the Pearson Correlation Output, and find the weighted normal of appraisals given by the clients from these  $X$  clients with likenesses on the appraisals given by them.



While various individuals may have distinctive reasoning and mind-set when offering evaluations to the movies, a few people grade to give high scores by

and large to the shows, some are progressively exacting despite the fact that they are for the most part get happy with shows and movies. To evade this biasness, we can expel every client's normal rating of the considerable number of movies and shows when processing normal, and include it back for the present client shifting, appeared as underneath.

Two different ways to get similitudes are Pearson Correlation

$$r_{ij} = \frac{\sum_k \text{Similarities}(u_i, u_k) r_{kj}}{\text{number of ratings}}$$

As many people have different thinking and mood or likes and dislikes when they give ratings to the movies or shows, some users generally think of giving high ratings generally to the shows or movies, some show higher strictness even though they get gratified with movies while watching. To remove this biasness, we removed each customer's individual ratings of all the movies and shows when finding the total average, and sum it to the current customer rating to do correct calculation and prediction, given below is the formula of finding the mean of ratings

$$r_{ij} = \bar{r}_i + \frac{\sum_k \text{Similarities}(u_i, u_k)(r_{kj} - \bar{r}_k)}{\text{number of ratings}}$$

$$\text{Cosine Similarity : } \text{Sim}(u_i, u_k) = \frac{r_i \cdot r_k}{|r_i||r_k|} = \frac{\sum_{j=1}^m r_{ij}r_{kj}}{\sqrt{\sum_{j=1}^m r_{ij}^2 \sum_{j=1}^m r_{kj}^2}}$$

$$\text{Pearson Correlation : } \text{Sim}(u_i, u_k) = \frac{\sum_j (r_{ij} - r_i)(r_{kj} - r_k)}{\sqrt{\sum_j (r_{ij} - r_i)^2 \sum_j (r_{kj} - r_k)^2}}$$

Finally, the trick is to find the extreme similar movies and shows to the current customer and calculate the individual rating of the particular item and then predicting the new movies to the current customer.

Prior, knowing anything about the database movies or items and the customers themselves, we supposed two customers are near similar when they give the same item with similar ratings.



**Content Based Filtering** , In this method , the two movies are similar when they get similar ratings from the two different customers. Finally, we tried to do the prediction for new customer on a new movie or show by calculating weighted average of ratings on most liked or disliked items from the current particular customer.

One major benefit of Item-based Content Filtering is the consistency with which the ratings of a given movie or show will not change significantly overtime, unlike human beings.

## CHAPTER 2

### LITERATURE SURVEY

**PAPER 1 : A Movie Recommendation System: MOVREC** by Manoj Kumar (Assistant Professor, Deptt. of IT BBDNITM Lucknow), D. K. Yadav (Assistant Professor, Deptt. Of CS, MNNIT Allahabad, Allahabad), Ankur Singh(M.Tech(P), Software Engineering, BBDU Lucknow, India), Vijay Kr. Gupta(Assistant Professor, Deptt. Of CS, BBdnitm Lucknow).

#### **ABSTRACT**

Presently a-days proposal framework has now transformed our way of discovering things in intrigue. This is generally utilized data sifting approach that is utilized to gauge the need of that specific client. The most well known regions where proposal frameworks are appropriate are Netflix, books, news, articles, music, recordings, films, and so forth. In this published paper we have planned a film prediction and calculation framework called MOVIE RECOMMENDATION SYSTEM. It depends on the community oriented sifting approach that utilizes the data gave by clients, investigations them and afterward suggests the movies that are most appropriate to the client at that point. The prescribed movies list is placed by the evaluations given to all the motion pictures or movies in the database by renouncing clients and usefulness the K-neighbourhood calculation for this reason. Film RECOMMENDATION SYSTEM, additionally assists clients with discovering motion pictures of their decision dependent on the film understanding of different clients in a proficient and powerful way without burning through much time in pointless perusing. This framework is created in Node.js and Apache Server 2.0. The introduced recommender framework creates suggestions utilizing various kinds of information and information about clients, accessible items, and past exchanges put away in tweaked databases. At that point clients can without much of a stretch peruse the proposals and discover their preferred film and TV Shows.

## **RELATED WORK**

A few suggestion strategies have been created in the course of the most recent a very long while. These frameworks utilize different strategies, for example, communitarian approach, content based methodology, an utility base methodology, cross breed approach and so forth and taking a gander at the purchasing conduct and past of the businesspeople Lawrence et al. 2001 presented a proposal framework that set forward another item in the market. Further substance based separating approaches were applied to refine proposal coordinated effort. Most suggestion frameworks today use evaluations given by doing without clients to discover potential clients. These evaluations are useful to anticipate and allude a thing of the customer decision. In 2007, Weng, Lin, and Chen completed an evaluation study that coordinated that utilizing multidimensional examination and extra customer profiles expands proposal quality.

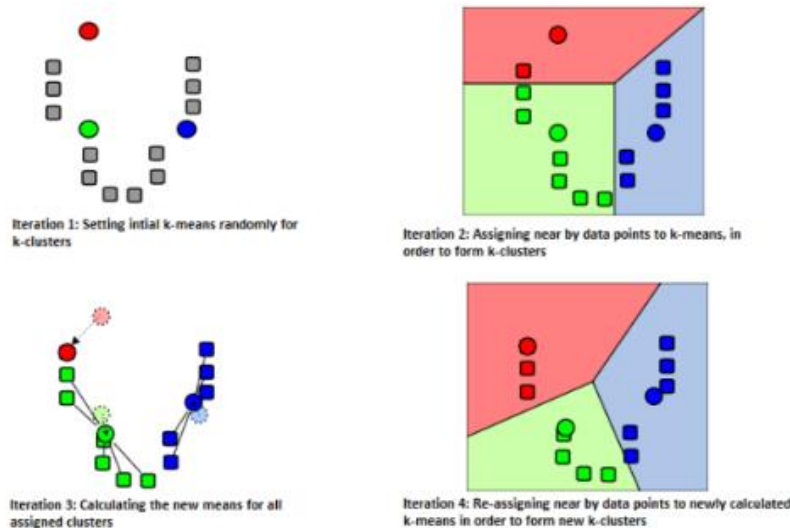
### **3. Research Method**

#### **3.1 Original M-Instrument Algorithm**

Starting M-instrument calculation was estimated by MacQueen and the ISODATA calculation by crude however refined variant of the m-implies. Bunching parts objects into important gatherings. Grouping is a doubtful learning. Archive bunching is programmed report association. In the M-implies grouping strategy we select M introductory centroids, here M is the quantity of wanted bunches. All fact is then allotted to the group with the very nearest mean for example the centroid of the group. At that point we update the inside purpose of each group that is allocated to the bunch. We rehash this procedure till there is an adjustment in the group place (core).

The complete software is based on the calculation of the information metadata. The calculation is comprised of the following procedures:

1. Select M focuses as beginning stages
2. Repeat
3. k by appointing each point from the gatherings to its closest centroid.
4. Re-process the centroid of each group.
5. Up until the centroids change.



The 5 steps of the K-Means algorithm diagram.

### 3.2 Data Description

The projected model we have used a filter before using K-means algorithm, Algorithm means. Terminologies that is used to calculate distance of every point from centroid int formula are given below:

1. Customer
2. No. Of Ratings to each movie
3. Rating
4. Year
5. Actor / User

To process the learning information, the K-implies calculation in information mining begins with a first gathering of arbitrarily chosen centroids, which



are utilized as the starting focuses for each bunch, and afterward performs iterative (monotonous) estimations to upgrade the places of the centroids

It stops making and enhancing bunches when either:

The centroids have settled — there is no adjustment in their qualities on the grounds that the bunching has been effective.

The characterized number of emphasises has been accomplished.

The research which was conducted by them, it showed that as the number of rating increases, the mean weight of the rating of that particular movie item should also increase progressively. Therefore, we have classified the ratios in 1: 1, 1: 2, and 1: 3 based on the total number of ratings received for the movie. This also found that movies which have a mean rating of less than 5 are very less appropriate for approval, and are the less required by customers. Customers generally want to watch a nice movie and a high rating of that movie which confirms that the system projected movie set or show is among the best rated movies that a large number of customers liked. The mean weights given to other ratings or number of ratings are usually based on the average rating of the total number of films in the data set.

### **3.3 Working of Movie Recommendation System**

When a customer enters on our Movie Recommendation Platform they will have some choices on the system. He can search for a particular film or see a list of upcoming films or visit the rating page. On the ratings page customer can give a rating according to his/her choice to the movies having various attributes. Based on these values, the system perform the search at the backend from our database and then using the Parson Correlation and then produce the output of suitable movies. Movies remembered for the exhibit are those whose one quality worth matches with the information estimation of the client. At that point we included the quantity of motion pictures in our group with the support of a counter. At the happening and the counter worth is not exactly or equivalent to twenty, we show a film list arranged by the evaluations related with the movies. On the quantity of motion pictures is more than twenty, at

that point we apply a pre-channel and pick the best twenty motion pictures by rating. If two films have the same rating, then a film with a large number of ratings is preferred. After filtering the list of movies, we match the property values of their respective weights and calculate the total weight of each movie.

### **Weightage and matching of attributes**

#### **1. ACTOR ( $W_a$ )**

$W_a = \frac{\text{No. of movies of Actor(a) in data set}}{\text{Total no. of movies in dataset}}$

$W_d = \frac{\text{No. of movies of Director(d) in data set}}{\text{Total no. of movies in data set}}$

Total no. of movies in data set

#### **2. GENRE ( $W_g$ )**

$W_g = \frac{\text{No. of movies of Genre(g) in data set}}{\text{Total no. of movies in data set}}$

Total no. of movies in data set

#### **3. Year ( $W_y$ )**

$W_y = \frac{\text{No. of movies in Year (Y) in data set}}{\text{Total no. of movies in data set}}$

Total no. of movies in data set

Total weight of a particular movie m is given below:

$$W_m = W_r + W_a + W_d + W_g + W_y$$

### 3.4 Proposed Algorithm

**Input:** a number of movies:  $m$

**Output:** a number of clusters:  $K$

**Step 1** Choose  $n$  movies from  $m$  movies  $n < m$

**Step 2** If  $n > 20$  then choose top 20 movies from  $n$  movies based on ratings.

Else display the output movies sorted on the basis of rating.

**Step 3** If rating of movies  $x, y$  are same i.e. If

$$R_x = R_y$$

Then choose those movies which have greater number of users' votes.

**Step 4** let  $K=4$ .

**Step 5** REPEAT (6, 7)

**Step 6** Chose primary centroid  $C_1, C_2, C_3, C_4$ .

**Step 7** Compute Euclidean distance of all data points w.r.t.  $C_1, C_2, C_3, C_4$  and re-calculate the centroid of each cluster.

**Step 8** Up Until centroid change.

## Weightage and matching of attributes

### 1. ACTOR ( $W_a$ )

$W_a = \frac{\text{No. of movies of Actor(a) in data set}}{\text{Total no. of movies in dataset.}}$

$W_d = \frac{\text{No. of movies of Director(d) in data set}}{\text{Total no. of movies in data set}}$

Total no. of movies in data set

### 2. GENRE ( $W_g$ )

$W_g = \frac{\text{No. of movies of Genre(g) in data set}}{\text{Total no. of movies in data set}}$

Total no. of movies in data set

### 3. Year ( $W_y$ )

$W_y = \frac{\text{No. of movies in Year (Y) in data set}}{\text{Total no. of movies in data set}}$

Total no. of movies in data set

Total weight of a particular movie m is given below:

$$W_m = W_r + W_a + W_d + W_g + W_y$$

Where,

**m:** Total count of the movies in dataset taken

**n:** Total count of movies after the user's request

**x, y:** Two random movies selected

**R<sub>x</sub>, R<sub>y</sub>:** Representing Ratings of movies x, y respectively

**K:** Number of groups

**C<sub>1</sub>, C<sub>2</sub>, C<sub>3</sub>, C<sub>4</sub>:** Initial Centroid.

## 3.5 Problems faced

The biggest encounter while developing any software is to meet the requirements of the end customers and working on the interaction of that user with the Interface, for whom the system is being developed. We faced few issues and challenges while developing the architecture and dataflow of the software . Some of them are:

- Developing a software that has user friendly Interface and very easy to understand and use of the new customers.
- Challenge to find a data set that contains all the movies lists, their ratings and information about a particular film.
- Biggest challenge was the recommended list of the most appropriate films.
- To expand our software so that it can gratify the customers of different geographical locations.

### **3.6 Overcome Problems**

The developed system after completion was tested on a small group of people, and we have noted some good results or output or feedback from them, and we have received positive feedback those people. We have tried to keep our system simple and customer interactive and to implement this we have applied Node.js a simple Javascript Runtime Server.

To collect data information, we searched for data asset available online or MOVIEGEN movies data and extracted all data that was suitable for our planned system.

We have implemented a K-means clustering algorithm with a prior filter to recommend the film correctly to the user.

We added movies information from the csv files to the database regardless of their language or location so that customers from all over the world can have access to our software.

To assign weight to attributes and to prioritize them we have completed a survey on some people and based on the results found we have prioritized our characteristics.

## **4. Results**

In this paper we offered a new calculation utilizing verifiable appraisals alongside meta information filtering in proposal frameworks to prescribe new motion pictures and shows to the clients which are existing in the product. The proposed structure was tried on two arrangements of film focal point dataset by enorning two diverse meta information like occupation and sex. Noteworthy distinction in MAE shows that metadata which reasonably covers the client database ought to be picked. MAE shows that the proposed system

performs superior to gauge. This shows utilizing metadata alongside recently put away information gives us a superior proposal framework.

For future work we might want to test our structure on datasets having progressively complex metadata. Connection between various highlights of a metadata can be utilized to accomplishment to additionally build the correctness of the suggestion framework. This system can likewise expanded and can be applied and tried on various fields for durability.

**PAPER 2: Movie Recommendations Using the Deep Learning Approach by Jeffery Lund and Yiu-Kai Ng, Computer Science Department, Brigham Young University, Provo, Utah 84602, USA.**

Content Lettering Content-based suggested frameworks that work with the assistance of clients' regularly called clients' supporting frameworks were at first made. A prof. Le contains realities about a customer and its preference. Taste is developed on how the customer has evaluated the items or movies. For the most part, while making a le, the recommender framework leads an overview to get starting data about the client, to evade another client issue. In the suggestion procedure, the motor compares things that were at that point appraised decidedly from the customer with the items that it did not rated and discovers resemblances. Items that are emphatically like those evaluated will be prescribed to the client. Figure 1 shows a case of a client supporter appears with motion pictures that he has seen and a client made rating. Figure 2 displays the rundown of movies and shows then their separate trademark esteems. A substitute built suggestion framework will dispense with from the rundown of films (Figure 2) that the client has just watched and evaluated emphatically. At that point, it will contrast and the remainder of the movies from those movies list (Figure 2) and discover likenesses. Comparative movies will be prescribed to the client. In the inquisitive model we can see that a film like "American Pie" is an "unnerving film" that the client has appraised decidedly. The client didn't rate the "Blood and gore flick", so it will be prescribed to him.

## **B. Affiliated Filtering**

This methodology was referenced and portrayed by Paul Resnick and Hal Varian in 1997, so community lettering developed into one of the most looked into procedures. The sign of cooperative separating is in a network ding driving clients who share appreciation. In the event that two clients have the same or about the equivalent evaluated things, they will have a similar taste. Such clients structure a gathering or a purported neighborhood. A client gets suggestions for things that he/she had not before evaluated, however was at that point esteemed emphatically by clients in the neighborhood. Figure 4 displays that every one of the three clients assess motion pictures with positive and equivalent scores. This alludes they have comparative tastes and structure

an area. Client A has not appraised the film "TRON: Legacy", which implies he has not seen it till now. Since the film was evaluated decidedly by different clients, he/she would be suggested this thing/film. As differentiating to basic suggesting frameworks, where proposals are the establishment for the most exceptionally evaluated thing and most mainstream thing strategies, partner proposal frameworks care about the flavor of the client before suggesting. Taste is viewed as steady or if nothing else gradually changing with time.

Sifting is broadly utilized in internet business. Clients can rate books, tunes, films and afterward can get proposals about those issues/things in the up and coming future. Additionally acquainted lettering is utilized in the surfing of specific records, (for example, reports between logical works, articles, and diaries).

Delving into the subtleties of synergistic in sifting techniques permits us to separate the most well known methodologies: client based, thing grounded and model-grounded methodologies.

### **I) User-based methodology:**

The methodology was founded and forwarded by Professor Jonathan L. coming from the University of Minnesota in around 1990s. Was proposed by Herocker. In [the] client based methodology, clients assume a crucial job. In the event that a few clients have comparable tastes, at that point they join a set/gathering.

Suggestions are given to the client dependent on the assessment of items by different clients, shaping a similar set with which he shares common inclinations. On the off chance that the thing was appraised decidedly by the network, it would be recommended to the client inside the gathering. Accordingly in a client based methodology those things that were at that point evaluated by the client assume a significant job in scanning for a gathering that offers acclaim with it.



## **ii) Item-based methodology:**

The procedure was presented by scientists at the University of Minnesota in 2001 . Alluding to the point that clients 'tastes stay consistent or fundamentally the same as things change, building neighborhoods dependent on clients' appreciation. Later the framework makes proposals with things in the local that a client would support.

## **C. Cross breed Recommendation Approach**

Some recommender plans join various strategies of communitarian approaches and substance based methodologies for better suggestion results. Utilizing the mixture approach we can avoid a portion of the proposals like cold-start issue and issues with unadulterated suggesting framework. The blend of approaches can proceed in various manners : 1) executing the usage of the calculation and association the outcomes. 2) Use a few guidelines of material-situated in separating in cooperative strategies. 3) Use a few rules of communitarian in sifting in the material methodology. 4) Create a UI Ad Advisor framework, which unites the two strategies. Robin Burke requested cross breed recommender frameworks by characterizing them. Film Screen is a case of a proposal operator that gives its clients suggestions dependent on half and half movies lettering about motion pictures appeared in theaters. A client on the site can produce a record and assess all the films found in films. Utilizations sifting. This substance dependent on the consequence of communitarian sifting. Executes lettering.

## **D. The cross-space based methodology**

It is a significant piece of the way toward suggesting finding like clients and building and exact neighborhood synergistic. The closeness of the two clients is uncovered dependent on their commitment of the things. Yet, comparable thankfulness in one area positively doesn't imply that valuations are similar in another space. Clients who share tendencies in satire are not through a similar sort of mental trip.

A case of various assessment of movies in various spaces, ordinary recommender frameworks dependent on synergistic sifting look at clients

without separating things into different areas. Similarities of clients in cross-space frameworks are registered area subordinate. A motor produces a nearby neighborhood for every client as indicated by the space. At that point, the balance equality esteems and compact sets of closest neighbors are sent for the general closeness computation. The suggested framework chooses by and large closeness, assembles generally speaking neighborhoods, and makes expectations and proposals for motion pictures.

D. Shared strategies are decentralized frameworks suggested with P2P approach. Each friend can have a place with a gathering of different companions with the same interests and get proposals from clients in that gathering. Proposals may likewise be given dependent on an associate's history. Decentralization of proposal frameworks can take care of the issue of adaptability.

## **E. Cross-etymological methodology**

The recommender framework, based on a cross-etymological methodology, permits clients to get suggestions for the items having depictions in tongues they can't talk and comprehend. Yang, Chen, and Wu made a way to deal with the suggestions of a passing newsgroup. The fundamental sign is to delineate content and catchphrases in a similar component space in divergent dialects, or, in other words likelihood dispersion on idle subjects. Depicting objects parses the watchwords than utilizing framework vocabularies to make an interpretation of them into a de etymological language. From that point onward, the colleague or other, utilizing lettering, offers proposals to the framework clients. It is conceivable to make a language-autonomous portrayal of a book with the assistance of semantic investigation. Pascal Lopes utilized a strategy with a proposal association called MARS. In view of the word detecting inconsistency calculation that utilizes an online multilingual lexical database since the motor as a semantic storehouse gives right importance to words that keep away from equivalent issues. A similar way of thinking was looked for after by Murad Magaebel and Antonio Kau. They use online WordNet lexical oncology with the Euro WordNet multilingual lexical advantage for escape from the issue of identical words and to make logically portrayed relationship between names in wordonomy. At whatever point a client adds the tag to the thing equivalent words of the tag-word, it is uncovered by the collective labeling association and added to the label

utilizing WordNet ontology. EuroWordNet creates associations between labels in disparate European tongues, which suggest contrasting and arranging things and labels in various tongues. Cross-etymological recipient living beings break the language obstruction and give opportunities to see things, data, papers, or books in different tongues.

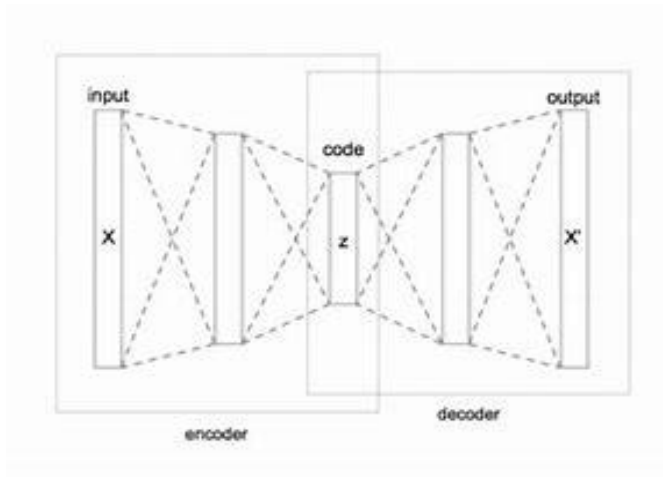
#### **IV. Difficulties and Issues**

- **Cold-Start:** Cold-Start is the strange to isolate its suggestions to new customers as its supporter is almost canceled and has not yet evaluated anything, so its rating is new to the framework. That was alluded to cold beginning matter. In some recommender frameworks this issue is settled by estimating while at the same time making framework. Things can likewise have a cool beginning when they are new to the structure and the principal thing hasn't been evaluated. Both these troubles can likewise be split with the half breed approach.

- **Convictions:** The advice of individual customers with little history may not be as significant as those whose advices are wealthy in their rich history. The issue of confidence ascends toward the appraisal of a particular customer. The issue can be split by the dissipating of likings to clients.

- **Adaptability:** With the development in the figure of customers and articles, the software desires more qualities for preparing metadata and building suggestions. The greater part of the qualities are utilized to characterize the things with the coordinating taste and related subtleties as the clients. This topic is furthermore split by a merger of various styles of sifting and physical enhancements of the plans. Portions of numerous calculations can be yielded out to accelerate the conveying on the web suggestions.

In Sparsity online shops that have a cumbersome measure of purchasers and things, there are pretty much consistently clients who have positioned

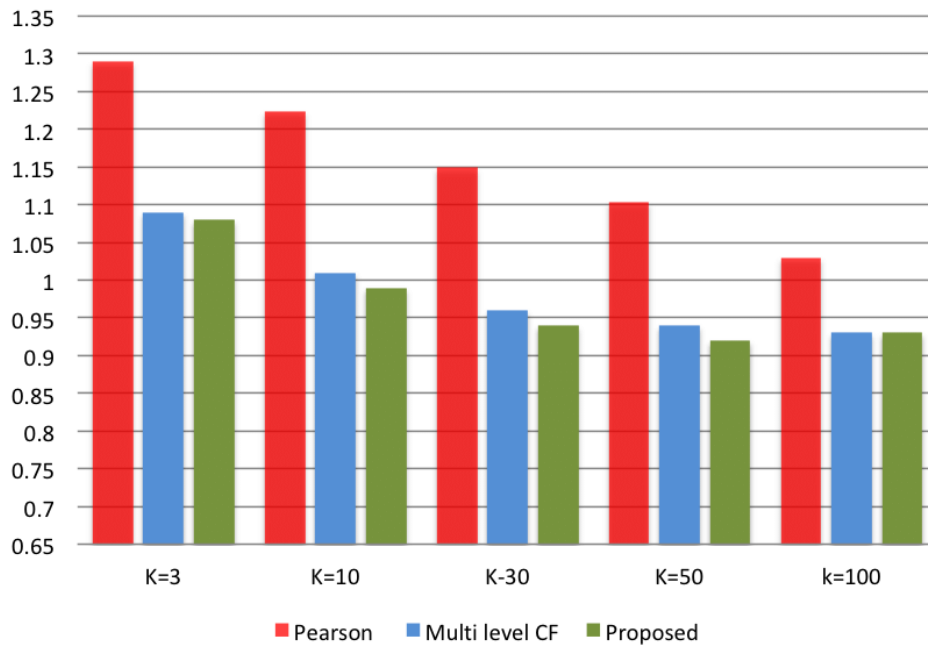


An autoencoder with three fully-connected hidden layers.

## V. Results

We have anticipated a straightforward neural system model that performs well for affiliated sifting of motion pictures in films suggestion framework. This will add to the current framework recommending that escalated learning can be a persuasive apparatus for an assortment of issues in data bringing . At last, the work improves as far as suggesting films and shows based on appraisals to the present clients. Our proposal framework utilizes regularization to additionally diminish any undesirable expectation blunders. Furthermore, our framework had the option to make sense of neighborhood-based zeroes better, and give better movies and shows suggestions.

### RMSE results for MovieLens 100.000

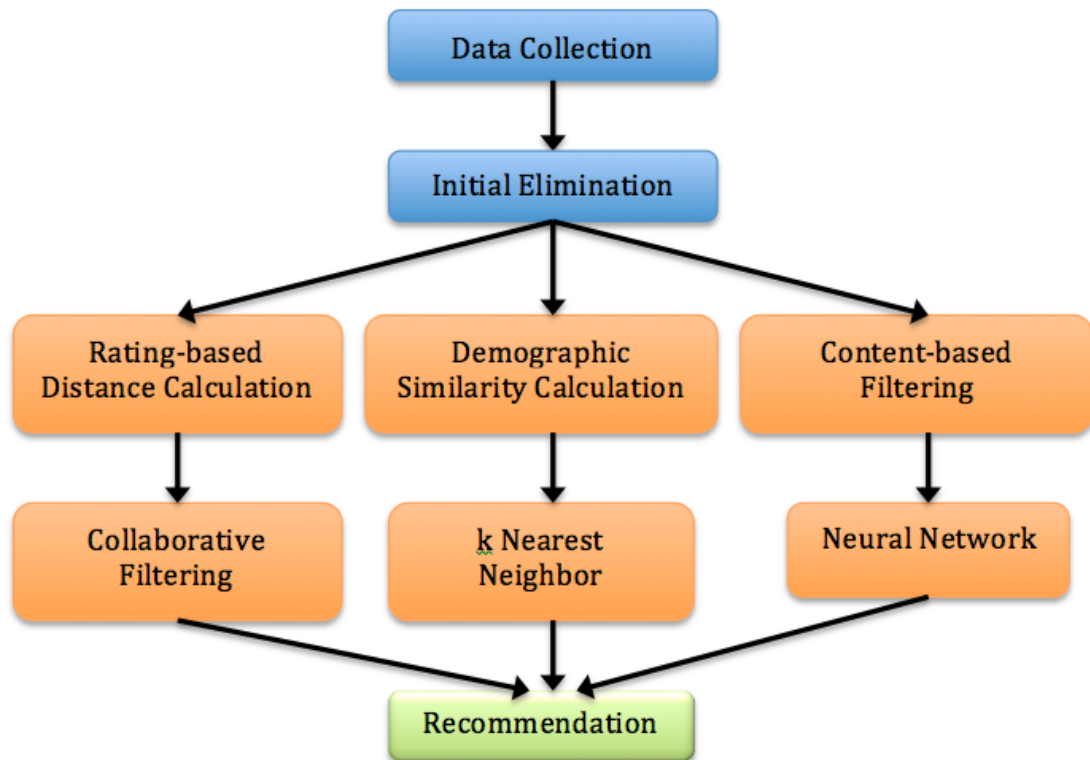


### **PAPER 3: Cold Start, Warm Start and Everything in Between: Autoencoder Based Approach for Recommendation by Anant Jain and Angshul Majumdar, IIT Delhi, New Delhi, India**

This enhances the work having issues of keeping it cooling & heating, and predicted structures activate to increase. By the large initial factor models we subject to the matrix formation which are used to separate Communitarian . Similarly at the late start, a set of papers have been disengaged that utilization autoencoders for corresponding exertion; Grid factorization has produced more favourable results in these tests. It is the main work that is based on a inclusive auto encoder that orders to talk both cold and very hot start topics. It is based on open ratings of customers on things like registered customer having different metadata. The developed system is distinguished and arrivals to position work processes to shift them.

This business subject community-oriented relocation. We examine both subjects - a warm start when charges from customers open on things like movies and TV shows. And cold starts where the customer or item is newly published. Another new autoencoder based system is developed. At the starting of the open assessments, we used the users previous metadata. It prints the output of predicted related movies to the earlier viewed and the system always starts with the initial structure using a particular initial data in the csv files.

The projection approach is characterized as isolated and best in class frameworks in isolated structures. For both simple and mostly quiet start issues, our system produces the best outputs. Now, for the hot start issue. On this project we all have focused on overall prediction of movies and TV Shows recommendation. Customer metadata used are age, occupation, ratings. Metadata is just a kind of users characteristics data. We observed that integrating customer information into the project settings gives us the improvements in the prediction result regarding customer information. Perhaps consolidating other data, for example, on-screen character, on-screen characters, cinematographer, and so forth into metadata will additionally advance in great outcomes. We have proposed another structure in this paper. We have demonstrated the pre-owned motion pictures dataset. In any case, the proposed system is average and can be utilized for different fields, for instance, news, accounts, music, books, and so on.



## VI. Results

In this paper we offered a new calculation utilizing verifiable appraisals alongside meta information filtering in proposal frameworks to prescribe new motion pictures and shows to the clients which are existing in the product. The proposed structure was tried on two arrangements of film focal point dataset by enorming two diverse meta information like occupation and sex. Noteworthy distinction in MAE shows that metadata which reasonably covers the client database ought to be picked. MAE shows that the proposed system performs superior to gauge. This shows utilizing metadata alongside recently put away information gives us a superior proposal framework.

For future work we might want to test our structure on datasets having progressively complex metadata. Connection between various highlights of a metadata can be utilized to accomplishment to additionally build the correctness of the suggestion framework. This system can likewise expanded and can be applied and tried on various fields for durability.

## **CHAPTER 3**

### **SYSTEM DEVELOPMENT**

#### **3.1 System Requirements**

The system requirements for the algorithms to process and platform to run the software fluently are given below:

- I. Windows 10 (64-bit )
- II. ANACONDA Software.
- III. Python 3+
- IV. React JS (FRONTEND)
- V. Node JS (BACKEND)
- VI. MYSQL

#### **Python Libraries**

Python is a language generally written in direct English language which is clear and adjust regardless, for an individual new to nature. One of the crucial reasons is a colossal choice of libraries is Python is the most standard programming language used for AI. Python libraries give base level things, so designs don't have to code them as a matter of course.

ML requires endless data dealing with, and Python's libraries empower you to get the chance to, direct, and control information. These are likely the most wide libraries you can use for ML and AI:

##### **3.2.1 NumPy**

NumPy is very popular python library having gigantic multi-dimensional show and structure taking care of, with the help of a colossal collection of raised level numerical limits. It is incredibly significant for head legitimate figures in Machine Learning. It is particularly useful for direct factor based math, Fourier change, and discretionary number limits. Excellent quality libraries like TensorFlow uses NumPy inside for control of Tensors.



### **3.2.3 SciPy**

SciPy is a no matter how you look at it library among AI lovers as it fuses separate modules for development, straight factor based math, blend and bits of knowledge. Difference between SciPy library and SciPy stack. SciPy is one of the primary wraps that make up the SciPy stack. SciPy is in like manner incredibly accommodating for picture control.

### **3.2.4 Scikit Learn**

Scikit-Learn is one of the exceptional common ML libraries for old style ML figures. It is based more than two extraordinary Python libraries i.e., NumPy and SciPy. Scikit-Learn supports commonly coordinated and unusable learning figures. Scikit-Learn can in like manner be used for data mining and data assessment, making it an immense instrument that ML is starting with.

### **3.2.5 Tensor Flow**

TensorFlow is a very notable open-source library for tip top numerical figuring made by the Google Brain bunch at Google. As the name advocates, TensorFlow is a structure that consolidates describing and running counts including tensors. It can get ready and procedure significant neural frameworks that can be used to make various AI applications. TensorFlow is extensively used in the field of concentrated learning investigation and application.

### **3.2.6 Keras**

Keras is an incredibly well known Machine Learning library for Python. It is a critical level neural frameworks API which can run over TensorFlow, CNTK, or Theano. It can run reliably on both CPU and GPU. Keras makes it really for ML beginners to collect and structure a Neural Network. A

champion among other thing about Keras is that it awards for basic and fast prototyping.

### **3.2.7 Pandas**

Pandas is an unmistakable Python library for data study. It isn't clearly related to AI. As we most likely am mindful the dataset should be set up before setting up the model. For this circumstance, pandas end up being helpful in light of the fact that it was developed unequivocally for data extraction and pre-getting ready of data. It gives raised level data structure and wide grouping of gadgets for data examination. It gives a couple of inbuilt procedures to getting, gathering and isolating dataset.

### **3.2.8 Matplotlib**

Matplotlib is a notable Python library for data portrayal. Like pandas, it isn't honestly related to AI. This is especially important when a product engineer needs to picture plans in data and necessities pictorial depiction of data. It is a 2D plotting library used to make 2D outlines and plots. A module called Pilepot makes it accommodating for programming engineers for plotting as it offers features to control line styles, printed style properties, planning tomahawks, etc. It gives various types of graphs and plots for data recognition, that is, histograms, botch diagrams, bar talks , e.t.c.

### **3.2.9 NLTK**

This set-up of libraries is named for the Natural Language Toolkit and, as the name itself suggests, it is used for essential endeavours of agent and truthful trademark language getting ready.

The convenience of NLTK permits a huge amount of exercises for content naming, request and tokenizing, recognizing verification of name components, creation of corpus trees that reveal cover and between sentence

conditions, stems, semantic justification. All structure squares grant the creation of complex research portrayals for various assignments, for example, presumption assessment, robotized overviews.

### 3.3 Anaconda

Boa constrictor dissemination has more than 1,500 packages too conda group and besides has a virtual space boss. It moreover consolidates a Graphic User Interface, Anaconda Navigator, as a graphical alternative as opposed to the course line interface gave.

### 3.4 Node.js

Node.js, a very popular runtime javascript platform that depends on Chrome's V8 Engine. It is a JavaScript runtime for building capable and adaptable framework or highly scalable applications. The above framework practices an event driven method, non-blocking I/O model and therefore it is very lightweight and very successful backend framework, It is perfect for building realtime applications that run in appropriated contraptions.

#### 3.4.1 Main Properties of Node.js

Coming up next are the key properties that choose Node.js as the essential choice of programming modelers.

- **Event Driven and Asynchronous-** Generally every API on the Node.js platform are nonconcurrent, for instance non-blocking and asynchronous. This basically supposes a Node.js server that doesn't believe that the API will bring data back. The center point server works on action of non-blocking events from Node.js needs the server to give a response from the past API call. Every REST call in the Node.js is stateless.

- **Highly Fast** - Node.js library is fast in code execution, being founded on Google Chrome's V8 JavaScript engine.
- **Single threaded & significantly versatile** - Node.js runs on single threaded approach with non-blocking event. The non blocking event structure impulses the server that it responds in a non-blocking way and increases the server's robustness and outstandingly versatile not in any way like standard servers that set obliged strings to manage expectations. Node.js always uses a single threaded programming system and a comparable simple applications similarly fills in as significantly greater number of requesting than ordinary servers, for instance, HTTP servers.
- **Zero buffering** - Node.js applications don't buffer any data. These applications fundamentally yield the data in bits of memory.

### 3.5 Angular 4+

Saucy is a frontend stage that makes it uncommonly easy to fabricate web applications with the web, having custom templating engines , dependence implantation, all the way tooling , testing and facilitated acknowledged techniques to disentangle exact improvement challenges. Dashing draws in designers to make applications that harp on the web, flexible or work zone.

### 3.6 Bcrypt

Bcrypt is a mystery word or sensitive data hashing module organized and made by a German designer Niels Provos and David Mazieres subject to the bcrypt figure and introduced in USENIX in 1999. Despite adding salt rounds to guarantee against rainbow table attacks, bcrypt library is an adaptable limit: after some time, the excess count can be extended to make it all the more moderate, so it can in like manner be brutal with growing estimation control.

### 3.7 Express.js

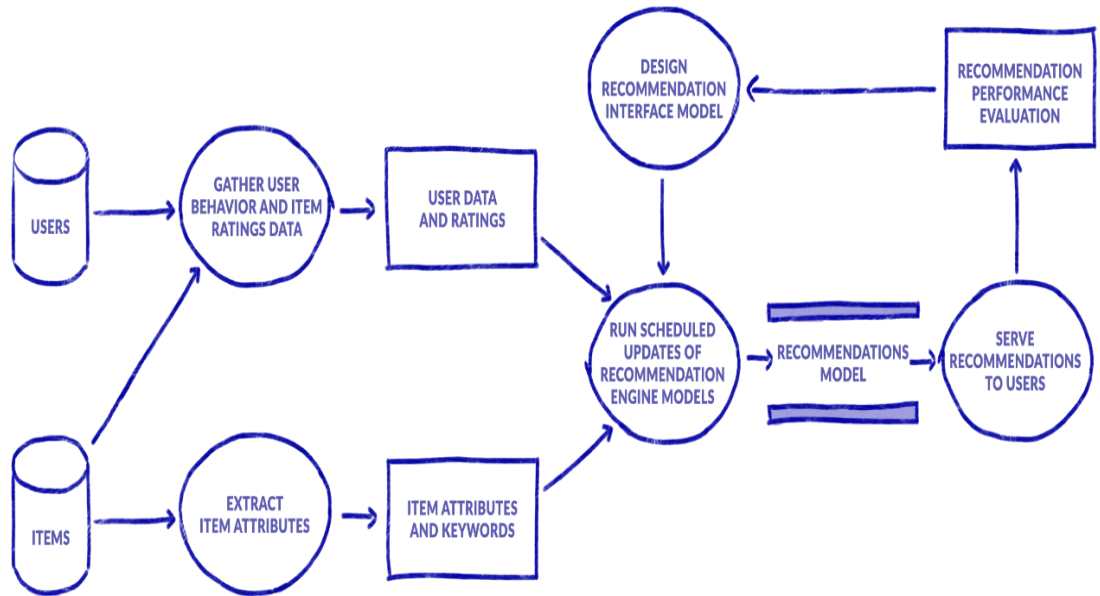
Express.js is a very important and highly versatile Node.js application framework that works on top of Node.js Application and gives best sets to web and flexible applications. It is an open source framework which is made and kept up by the Node.js Founders.

#### Main points of Express.js


























- Node.js makes web application headway speedier and more straightforward.
- Easy to structure and modify.
- Allows you to describe courses to your application subject to HTTP methodologies and URLs.
- There are diverse middleware modules fused that you can play out extra endeavours on requesting and server response.
- It is definitely not hard to organize with various arrangement engines like Jade, Wash, EJS, etc.
- Allows you to portray a misstep while overseeing middleware.
- It is definitely not hard to serve your application's static archives and resources.
- Allows you to make a REST API server.

- Easy to interface with databases like MongoDB, Redis, MySQL.

## Flowchart



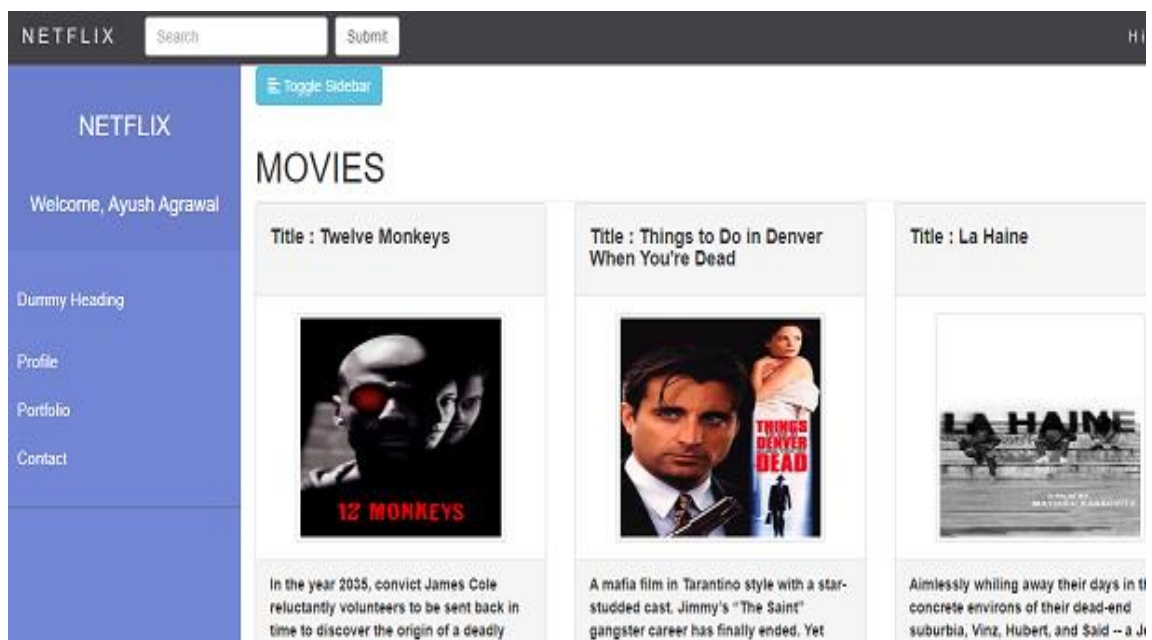
## Utility Matrix

## Predictions

- Pearson Correlation is a mathematical measure that shows how two or more objects can change together. Movies with high intersectionality are movies that are very similar to each other. In our case we will use the Pearson correlation technique. This number will lie between 1 and -1. 1 indicates a straight line alignment while -1 indicates a negative link. 0 shows no direct association. So, movies with zero links are no different.
- In order to facilitate communication between the two data structures we use pandas interaction and functionality. Computer combinations are two combinations of rows or columns of two data objects.

Till Now we have developed the Front End, and applied complete Authentication for the users and applied basic recommender system



NETFLIX

## LOGIN FORM

[Forgot Password?](#)

NETFLIX

## REGISTER FORM



## CHAPTER 4

### PERFORMANCE ANALYSIS

The screenshot shows the PyCharm IDE with a Python script named `main.py`. The code performs a performance analysis by creating a movie matrix, finding correlations, and identifying top movies. The output shows the top 10 movies based on correlation and number of ratings.

```
File Edit View Navigate Code Refactor Run Tools VCS Window Help py - main.py - PyCharm
py main.py
Project D:\py-nodelpy main.py
main.py
movies.csv
ratings.csv
External Libraries
Python 3.7 (py) > C:\Use
Scratches and Consoles
Invalid Python interpreter selected for the project
25
26 # CREATING THE MOVIE MATRIX
27 movie_matrix = data.pivot_table(index='userId', columns='title', values='rating')
28
29 ratings.sort_values('number_of_ratings', ascending=False)
30
31 # FINDING THE CURRENT MOVIE MATRIX FROM MOVIE MATRIX
32 my_movie_rating = movie_matrix[title]
33
34 # FINDING THE CORRELATION BETWEEN CURRENT MOVIE WITH OTHERS HAVING SIMILAR RATINGS
35 similarMovie = movie_matrix.corrwith(my_movie_rating)
36
37 corr_AFO = pd.DataFrame(similarMovie, columns=['correlation'])
38 corr_AFO.dropna(inplace=True)
39 # FINDING THE MOVIES WHICH ARE HAVING NUMBER OF RATINGS > 100
40
41 corr_AFO = corr_AFO.join(ratings['number_of_ratings'])
42 corr_AFO = corr_AFO[corr_AFO['number_of_ratings'] > 100].sort_values(by='correlation', ascending=False)
43
44 # PRINTING THE TOP 10 MOVIES
45
46 print(str(corr_AFO.head()))
47
```

Run: main

↑	Clear and Present Danger (1994)	0.698836	110
↓	Net, The (1995)	0.598322	112
↓	Green Mile, The (1999)	0.574799	111
↓	Firm, The (1993)	0.561304	101
↓	Departed, The (2006)	0.543279	107

Process finished with exit code 0

The screenshot shows the PyCharm IDE interface. The top menu bar includes File, Edit, View, Navigate, Code, Refactor, Run, Tools, VCS, Window, and Help. The project name is 'py - main.py - PyCharm'. The main editor window displays a Python script named 'main.py' with the following code:

```
1 import sys
2 import pandas as pd
3 import numpy as np
4 import os
5
6 import matplotlib.pyplot as plt
7
8 # GETTING THE MOVIE TITLE FROM NODE.JS BACKEND
9 title = sys.argv[1]
10
11 # READING THE MOVIES AND THE RATINGS CSV FILE
12
13 df1 = pd.read_csv(os.path.join(os.path.dirname(__file__), './ratings.csv'))
14
15 df2 = pd.read_csv(os.path.join(os.path.dirname(__file__), './movies.csv'))
16
17 # MERGING THE TWO DATA FRAMES INTO ONE ON THE BASIS OF SIMILAR KEY = movieId
18 data = pd.merge(df1, df2, on='movieId')
19
20 ratings = pd.DataFrame(data.groupby('title')['rating'].mean())
21 ratings['number_of_ratings'] = data.groupby('title')['rating'].count()
22
23
```

The Run window shows the output of the script, displaying a table of movie ratings and the number of ratings for each movie. The process finished with exit code 0.

Movie Title	Rating	Number of Ratings
Clear and Present Danger (1994)	0.698836	110
Net, The (1995)	0.598322	112
Green Mile, The (1999)	0.574799	111
Firm, The (1993)	0.561304	101
Departed, The (2006)	0.543279	107

At the bottom of the Run window, it says "Process finished with exit code 0".

The bottom status bar shows "Packages installed successfully: Installed packages: 'matplotlib==3.1.3' (today 09:14)" and the page number "835".

df - DataFrame

Index	userId	movieId	rating	timestamp
0	1	1	4	964982703
1	1	3	4	964981247
2	1	6	4	964982224
3	1	47	5	964983815
4	1	50	5	964982931
5	1	70	3	964982400
6	1	101	5	964980868
7	1	110	4	964982176
8	1	151	5	964984041
9	1	157	5	964984100
10	1	163	5	964983650
11	1	216	5	964981208
12	1	223	3	964980985
13	1	231	5	964981179

Format    Resize     Background color     Column min/max    OK    Cancel

Index	movieid	title	genres
0	1	Toy Story (1995)	Adventure Animation Ch...
1	2	Jumanji (1995)	Adventure Children Fan...
2	3	Grumpier Old Men (1995)	Comedy Romance
3	4	Waiting to Exhale (1995)	Comedy Drama Romance
4	5	Father of the Bride Part I...	Comedy
5	6	Heat (1995)	Action Crime Thriller
6	7	Sabrina (1995)	Comedy Romance
7	8	Tom and Huck (1995)	Adventure Children
8	9	Sudden Death (1995)	Action
9	10	GoldenEye (1995)	Action Adventure Th...
10	11	American President, T...	Comedy Drama Romance
11	12	Dracula: Dead and Loving I...	Comedy Horror
12	13	Balto (1995)	Adventure Animation Ch...
13	14	Nixon (1995)	Drama

Format    Resize     Background color     Column min/max    **OK**    Cancel

```
df1 = pd.read_csv(os.path.join(os.path.dirname(__file__), './ratings.csv'))
df2 = pd.read_csv(os.path.join(os.path.dirname(__file__), './movies.csv'))

# MERGING THE TWO DATA FRAMES INTO ONE ON THE BASIS OF SIMILAR KEY = movieId
data = pd.merge(df1, df2, on='movieId')
```

df - DataFrame

Index	userId	movieId	rating	timestamp	title
0	1	1	4	964982703	Toy Story (1995)
1	5	1	4	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)
5	18	1	3.5	1455209816	Toy Story (1995)
6	19	1	4	965705637	Toy Story (1995)
7	21	1	3.5	1407618878	Toy Story (1995)
8	27	1	3	962685262	Toy Story (1995)
9	31	1	5	850466616	Toy Story (1995)
10	32	1	3	856736119	Toy Story (1995)
11	33	1	3	939647444	Toy Story (1995)
12	40	1	5	832058959	Toy Story (1995)
13	43	1	5	848993983	Toy Story (1995)
14	44	1	3	869251860	Toy Story (1995)
15	45	1	4	951170182	Toy Story (1995)

Format    Resize     Background color     Column min/max    OK    Cancel

```
In [1]: import pandas as pd
import numpy as np
```

```
In [2]: rating_frame = pd.read_csv("C:/Users/ayaag/Desktop/ml-latest-small/ratings.csv")
rating_frame.head()
```

```
Out[2]:
```

	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

```
In [3]: rating_frame.head(10)
```

```
Out[3]:
```

	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
5	1	70	3.0	964982400
6	1	101	5.0	964980868
7	1	110	4.0	964982176
8	1	151	5.0	964984041
9	1	157	5.0	964984100

```
In [4]: movies_frame = pd.read_csv("C:/Users/ayaag/Desktop/ml-latest-small/movies.csv")
```

```
In [5]: movies_frame.head(10)
```

```
Out[5]:
```

	movieId	title	genres
0	1	Toy Story (1995)	Adventure Animation Children Comedy Fantasy
1	2	Jumanji (1995)	Adventure Children Fantasy
2	3	Grumpier Old Men (1995)	Comedy Romance
3	4	Waiting to Exhale (1995)	Comedy Drama Romance
4	5	Father of the Bride Part II (1995)	Comedy
5	6	Heat (1995)	Action Crime Thriller
6	7	Sabrina (1995)	Comedy Romance
7	8	Tom and Huck (1995)	Adventure Children
8	9	Sudden Death (1995)	Action
9	10	GoldenEye (1995)	Action Adventure Thriller

```
In [6]: data = pd.merge(rating_frame, movie_frame, on='movieId')
```

```
-----
NameError                                Traceback (most recent call last)
<ipython-input-6-c82e27d1afd2> in <module>
----> 1 data = pd.merge(rating_frame, movie_frame, on='movieId')

NameError: name 'movie_frame' is not defined
```

```
In [7]: data = pd.merge(rating_frame, movies_frame, on='movieId')
```

```
In [8]: rec_frame = pd.merge(rating_frame, movies_frame, on='movieId')
```

```
In [9]: rec_frame.head(10)
```

```
In [7]: data = pd.merge(rating_frame, movies_frame, on='movieId')
```

```
In [8]: rec_frame = pd.merge(rating_frame, movies_frame, on='movieId')
```

```
In [9]: rec_frame.head(10)
```

Out[9]:

	userId	movieId	rating	timestamp	title	genres
0	1	1	4.0	964982703	Toy Story (1995)	Adventure Animation Children Comedy Fantasy
1	5	1	4.0	847434962	Toy Story (1995)	Adventure Animation Children Comedy Fantasy
2	7	1	4.5	1106635946	Toy Story (1995)	Adventure Animation Children Comedy Fantasy
3	15	1	2.5	1510577970	Toy Story (1995)	Adventure Animation Children Comedy Fantasy
4	17	1	4.5	1305696483	Toy Story (1995)	Adventure Animation Children Comedy Fantasy
5	18	1	3.5	1455209816	Toy Story (1995)	Adventure Animation Children Comedy Fantasy
6	19	1	4.0	965705637	Toy Story (1995)	Adventure Animation Children Comedy Fantasy
7	21	1	3.5	1407618878	Toy Story (1995)	Adventure Animation Children Comedy Fantasy
8	27	1	3.0	962685262	Toy Story (1995)	Adventure Animation Children Comedy Fantasy
9	31	1	5.0	850466616	Toy Story (1995)	Adventure Animation Children Comedy Fantasy

```
In [10]: ratings = pd.DataFrame(rec_frame.groupby('title')['rating'].mean())
ratings.head(10)
```

Out[10]:

	rating
title	
'71 (2014)	4.000000
'Hellboy: The Seeds of Creation (2004)	4.000000
'Round Midnight (1986)	3.500000
'Salem's Lot (2004)	5.000000
'Til There Was You (1997)	4.000000

title	rating	number_of_ratings
'71 (2014)	4.000000	1
'Hellboy': The Seeds of Creation (2004)	4.000000	1
'Round Midnight (1986)	3.500000	2
'Salem's Lot (2004)	5.000000	1
'Til There Was You (1997)	4.000000	2
'Tis the Season for Love (2015)	1.500000	1
'burbs, The (1989)	3.176471	17
'night Mother (1986)	3.000000	1
(500) Days of Summer (2009)	3.666667	42
'batteries not included (1987)	3.285714	7

In [12]: ratings.sort\_values('number\_of\_ratings', ascending=False)

out[12]:

title	rating	number_of_ratings
Forrest Gump (1994)	4.164134	329
Shawshank Redemption, The (1994)	4.429022	317
Pulp Fiction (1994)	4.197068	307
Silence of the Lambs, The (1991)	4.161290	279
Matrix, The (1999)	4.192446	278
...	...	...
King Solomon's Mines (1950)	3.000000	1
King Solomon's Mines (1937)	2.500000	1
King Ralph (1991)	1.500000	1
King Kong Lives (1986)	2.000000	1
À nous la liberté (Freedom for Us) (1931)	1.000000	1



```
In [13]: ratings.sort_values('number_of_ratings', ascending=False).head(10)
```

Out[13]:

	rating	number_of_ratings
<b>title</b>		
Forrest Gump (1994)	4.164134	329
Shawshank Redemption, The (1994)	4.429022	317
Pulp Fiction (1994)	4.197068	307
Silence of the Lambs, The (1991)	4.161290	279
Matrix, The (1999)	4.192446	278
Star Wars: Episode IV - A New Hope (1977)	4.231076	251
Jurassic Park (1993)	3.750000	238
Braveheart (1995)	4.031646	237
Terminator 2: Judgment Day (1991)	3.970982	224
Schindler's List (1993)	4.225000	220

```
In [14]: movie_matrix = rec_frame.pivot_table(index='userId', columns='title', values='rating')
```

```
In [15]: print(movie_matrix)
```

```
title '71 (2014) 'Hellboy': The Seeds of Creation (2004) \  
userId  
1      NaN      NaN  
2      NaN      NaN  
3      NaN      NaN  
4      NaN      NaN  
5      NaN      NaN  
...     ...     ...  
606    NaN      NaN  
607    NaN      NaN  
608    NaN      NaN  
609    NaN      NaN
```

```
In [16]: my_movie_rating = movie_matrix['Air Force One (1997)']
```

```
In [17]: my_movie_rating.head(10)
```

```
Out[17]: userId
1      NaN
2      NaN
3      NaN
4      NaN
5      NaN
6      NaN
7      NaN
8      NaN
9      NaN
10     NaN
Name: Air Force One (1997), dtype: float64
```

```
In [18]: similarMovie = movie_matrix.corrwith(my_movie_rating)
```

```
C:\Users\ayaag\anaconda3\lib\site-packages\numpy\lib\function_base.py:2526: RuntimeWarning: Degrees of freedom <= 0 for slice
c = cov(x, y, rowvar)
C:\Users\ayaag\anaconda3\lib\site-packages\numpy\lib\function_base.py:2455: RuntimeWarning: divide by zero encountered in true_divide
c *= np.true_divide(1, fact)
```

```
In [19]: corr_AFO = pd.DataFrame(similarMovie, columns=['correlation'])
corr_AFO.dropna(inplace=True)
```

```
In [20]: corr_AFO = corr_AFO.join(ratings['number_of_ratings'])
corr_AFO = corr_AFO[corr_AFO['number_of_ratings'] > 100].sort_values(by='correlation', ascending=False)
```

```
In [21]: corr_AFO.head(10)
```

```
Out[21]:
```

	correlation	number_of_ratings
--	-------------	-------------------

```
In [19]: corr_AFO = pd.DataFrame(similarMovie, columns=['correlation'])
corr_AFO.dropna(inplace=True)
```

```
In [20]: corr_AFO = corr_AFO.join(ratings['number_of_ratings'])
corr_AFO = corr_AFO[corr_AFO['number_of_ratings'] > 100].sort_values(by='correlation', ascending=False)
```

```
In [21]: corr_AFO.head(10)
```

Out[21]:

	correlation	number_of_ratings
title		
Clear and Present Danger (1994)	0.698836	110
Net, The (1995)	0.598322	112
Green Mile, The (1999)	0.574799	111
Firm, The (1993)	0.561304	101
Departed, The (2006)	0.543279	107
Apollo 13 (1995)	0.536136	201
Twister (1996)	0.511892	123
American Pie (1999)	0.501064	103
Truman Show, The (1998)	0.500529	125
Cliffhanger (1993)	0.500267	101

```
In [10]: ratings = pd.DataFrame(rec_frame.groupby('title')['rating'].mean())
ratings.head(10)
```

Out[10]:

	rating
title	
'71 (2014)	4.000000
'Hellboy': The Seeds of Creation (2004)	4.000000
'Round Midnight (1986)	3.500000
'Salem's Lot (2004)	5.000000
'Til There Was You (1997)	4.000000
'Tis the Season for Love (2015)	1.500000
'burbs, The (1989)	3.176471
'night Mother (1986)	3.000000
(500) Days of Summer (2009)	3.666667
*batteries not included (1987)	3.285714

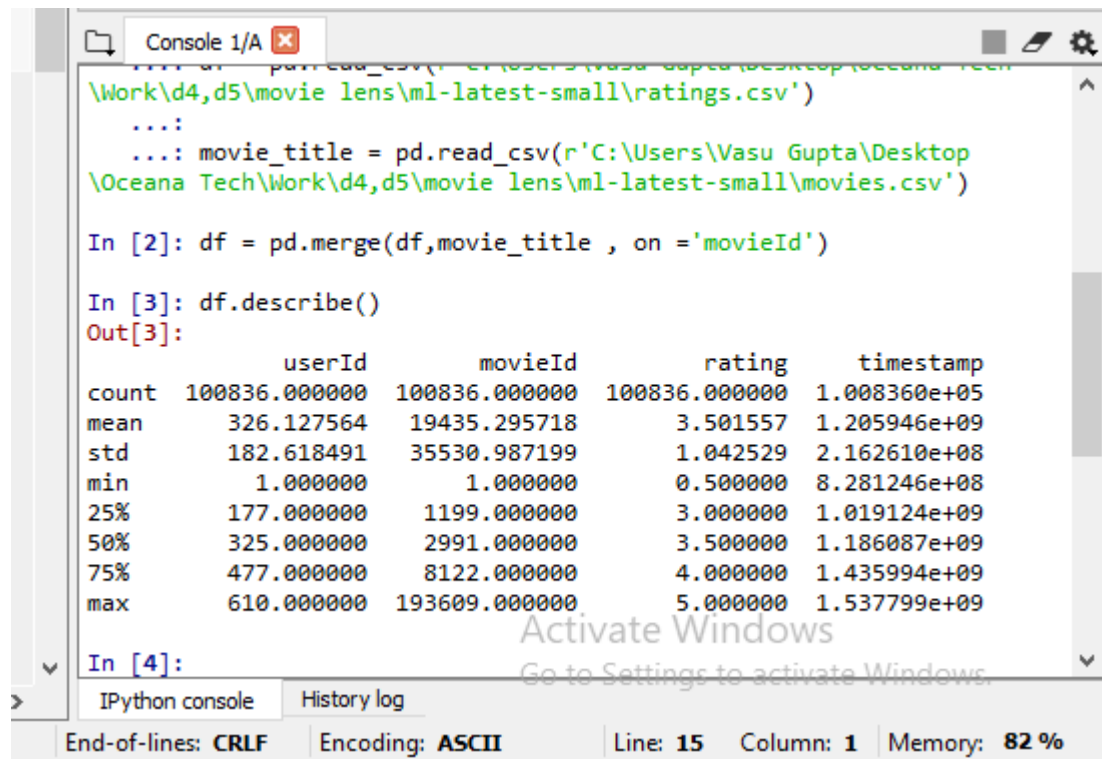
```
In [11]: ratings['number_of_ratings'] = rec_frame.groupby('title')['rating'].count()
ratings.head(10)
```

Out[11]:

	rating	number_of_ratings
title		
'71 (2014)	4.000000	1
'Hellboy': The Seeds of Creation (2004)	4.000000	1
'Round Midnight (1986)	3.500000	2
'Salem's Lot (2004)	5.000000	1
'Til There Was You (1997)	4.000000	2
'Tis the Season for Love (2015)	1.500000	1
'burbs, The (1989)	3.176471	17
'night Mother (1986)	3.000000	1

#describe the datasets mean and total count

df.describe()



```
...: pd.read_csv(r'C:\Users\Vasu Gupta\Desktop\Oceana Tech
\Work\d4,d5\movie lens\ml-latest-small\ratings.csv')
...:
...: movie_title = pd.read_csv(r'C:\Users\Vasu Gupta\Desktop
\Oceana Tech\Work\d4,d5\movie lens\ml-latest-small\movies.csv')

In [2]: df = pd.merge(df,movie_title , on ='movieId')

In [3]: df.describe()
Out[3]:
```

	userId	movieId	rating	timestamp
count	100836.000000	100836.000000	100836.000000	1.008360e+05
mean	326.127564	19435.295718	3.501557	1.205946e+09
std	182.618491	35530.987199	1.042529	2.162610e+08
min	1.000000	1.000000	0.500000	8.281246e+08
25%	177.000000	1199.000000	3.000000	1.019124e+09
50%	325.000000	2991.000000	3.500000	1.186087e+09
75%	477.000000	8122.000000	4.000000	1.435994e+09
max	610.000000	193609.000000	5.000000	1.537799e+09

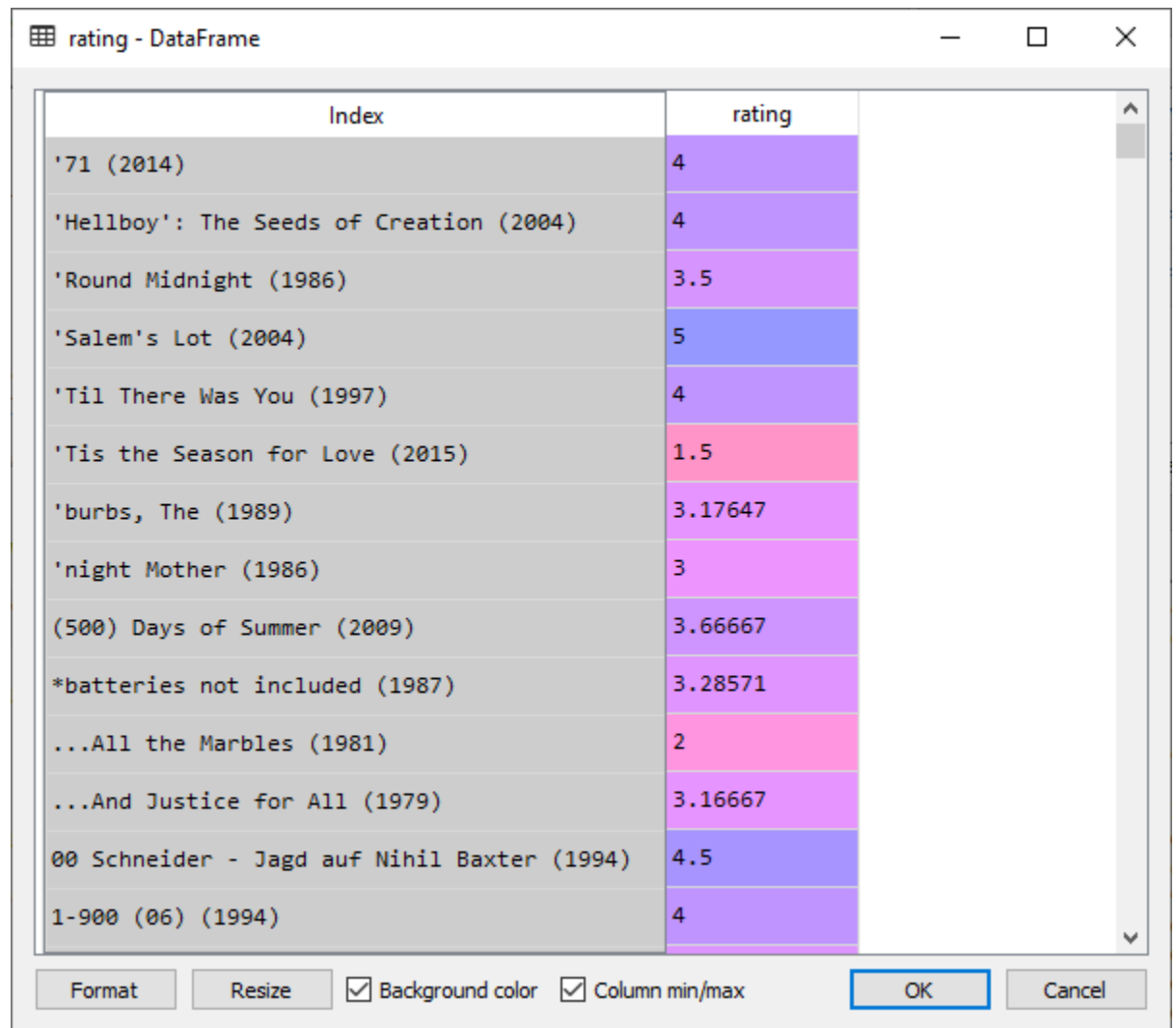
```
In [4]:
```

IPython console History log

End-of-lines: CRLF Encoding: ASCII Line: 15 Column: 1 Memory: 82 %

""Creating Data Frame with average rating of each movie and no of ratings""

So, average rating for every movie



The screenshot shows a window titled "rating - DataFrame" with a table of movie ratings. The table has two columns: "Index" and "rating". The rows are color-coded based on the rating value. The "rating" column contains numerical values, some of which are rounded to two decimal places. The window also features a toolbar at the bottom with buttons for "Format", "Resize", "Background color", "Column min/max", "OK", and "Cancel".

Index	rating
'71 (2014)	4
'Hellboy': The Seeds of Creation (2004)	4
'Round Midnight (1986)	3.5
'Salem's Lot (2004)	5
'Til There Was You (1997)	4
'Tis the Season for Love (2015)	1.5
'burbs, The (1989)	3.17647
'night Mother (1986)	3
(500) Days of Summer (2009)	3.66667
*batteries not included (1987)	3.28571
...All the Marbles (1981)	2
...And Justice for All (1979)	3.16667
00 Schneider - Jagd auf Nihil Baxter (1994)	4.5
1-900 (06) (1994)	4

#adding column -> total no of ratings for each movie

```
rating['number_of_rating'] = df.groupby('title')['rating'].count()
```

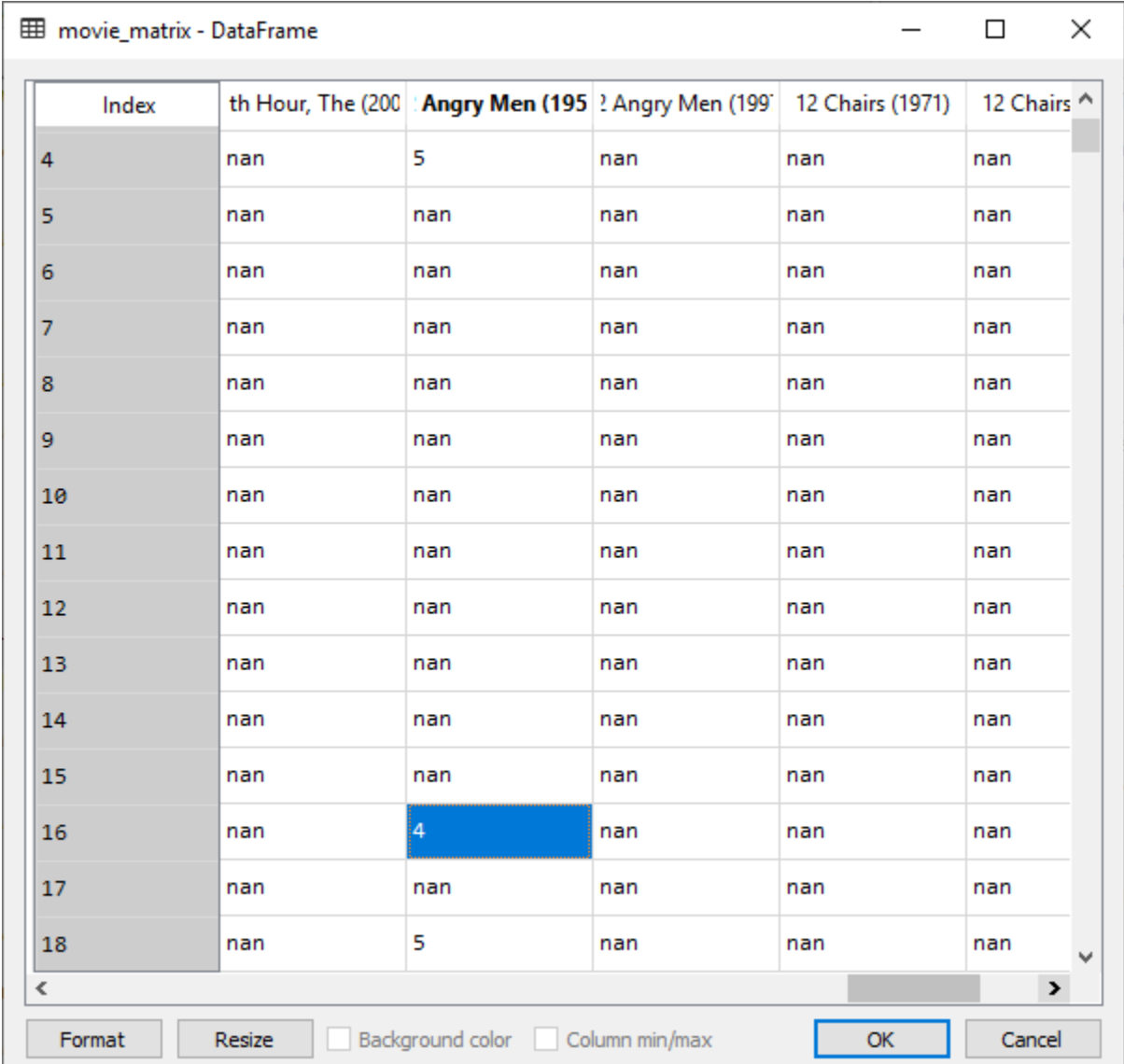
Index	rating	number_of_rating
'71 (2014)	4	1
'Hellboy': The Seeds of Creation (2004)	4	1
'Round Midnight (1986)	3.5	2
'Salem's Lot (2004)	5	1
'Til There Was You (1997)	4	2
'Tis the Season for Love (2015)	1.5	1
'burbs, The (1989)	3.17647	17
'night Mother (1986)	3	1
(500) Days of Summer (2009)	3.66667	42
*batteries not included (1987)	3.28571	7
...All the Marbles (1981)	2	1
...And Justice for All (1979)	3.16667	3
00 Schneider - Jagd auf Nihil Baxter (1994)	4.5	1
1-900 (06) (1994)	4	1

We would be creating a utility matrix with movie names or titles as matrix columns and userId as matrix rows

#each (row i,column j)reperent the rating given by i\_th user to j\_th movie

```
movie_matrix =rec_movies.pivot_table(index='userId',columns='title',values='rating')
```

```
rating.sort_values('number_of_rating',ascending=False)
```



Index	th Hour, The (200	Angry Men (195	? Angry Men (199	12 Chairs (1971)	12 Chairs
4	nan	5	nan	nan	nan
5	nan	nan	nan	nan	nan
6	nan	nan	nan	nan	nan
7	nan	nan	nan	nan	nan
8	nan	nan	nan	nan	nan
9	nan	nan	nan	nan	nan
10	nan	nan	nan	nan	nan
11	nan	nan	nan	nan	nan
12	nan	nan	nan	nan	nan
13	nan	nan	nan	nan	nan
14	nan	nan	nan	nan	nan
15	nan	nan	nan	nan	nan
16	nan	4	nan	nan	nan
17	nan	nan	nan	nan	nan
18	nan	5	nan	nan	nan



```
#finding correlation i.e. movies similar to burbs and movies similar to 500Daysofsummer
```

```
similarToBurbs = movie_matrix.corrwith(burbs_user_rating)
```

```
similarTo500Daysofsummer =  
final_movie_matrix.corrwith(_moviename_customers_rating)
```

```
#dropping the movies with empty rating fields and concatenating
```

```
corrBurbs = pd.DataFrame(similarToBurbs ,columns = ['Correlation'])
```

```
corrBurbs.dropna(inplace=True)
```

```
corr500Daysofsummer =
```

```
pd.DataFrame(similarTo500Daysofsummer,columns = ['Correlation'])
```

```
corr500Daysofsummer.dropna(inplace=True)
```

```
#Setting a threshold on the basis of no of movies in co-realtion dataframes
```

```
corr_500Daysofsummer =
```

```
corr_500Daysofsummer.join(rating['number_of_rating'])
```

```
corr_burbs = corr_burbs.join(rating['number_of_rating'])
```

```
#Extracting the movies similar to 'burbs' with a min of 50 ratings and storing it in a temperory dataframe with high co-realtion movie from top to bottom
```

```
tmp1 =
```

```
corr_burbs[corr_burbs['number_of_rating']>50].sort_values(by='Correlation',ascending=False)
```

```
#Extracting the movies similar to 'burbs' with a min of 50 ratings and storing it in a temperory dataframe with high co-realtion movie from top to bottom
```

```

tmp2 =
corr_500Daysofsummer[corr_500Daysofsummer['number_of_rating']>50].s
ort_values(by='Correlation',ascending=False)

#creating a merged dataframe having co-realtion with burbs as Correlation_x
and corelated to 500Daysofsummer as correlation_y

merged = pd.merge(left=tmp1, left_index=True,right=tmp2,
right_index=True,on = 'number_of_rating',how='inner')

#creating a list with taking mean of corresponding correaltion_x and
correaltion_y values to predict movies based on both the movies watched by
user

i = 0
for i in merged.index:
    l.append((merged['Correlation_x'][i] + merged['Correlation_y'][i])/2)

#Adding the list to the dataframe and displaying top 5 reommended movies
merged['Correlation_xy'] = l
merged.sort_values(by='Correlation_xy',ascending=False).head()

```

## OUTPUT MOVIES HAVING SIMILAR RATINGS

```
correlation  number_of_ratings
title
Clear and Present Danger (1994)    0.698836    110
Net, The (1995)                    0.598322    112
Green Mile, The (1999)             0.574799    111
Firm, The (1993)                   0.561304    101
Departed, The (2006)               0.543279    107

Process finished with exit code 0
```

## Front End Codes

### Login Page

```
<div class="container-fluid">
  <div class="row"><div class="col-md-2"></div>
  <div *ngIf="message" [ngClass]="styles" class="col-md-8">{{message}}</div>
  <div class="col-md-2"></div></div>
  <div class="row text-center">
    <div class="col-md-4"></div>
    <div class="col-md-4 text-center">
      <form (submit)="loginUser()"> <div class="form-group">
        <input type="text" [(ngModel)]="username" name="username" class="form-control" placeholder="Username"></div>
        <div class="form-group">
          <input type="password" [(ngModel)]="password" name="password" class="form-control" placeholder="Password"></div>
        <div class="form-group">
          <input type="submit" value="Login" class="btn btnsuccess"> </div>
      </form> </div> <div class="col-md-4"></div></div>
  <div class="row">
    <div class="col-md-3"></div>
    <div class="col-md-6">
      <div *ngIf="roller" class="jumbotron text-center">Loading...</div>
    </div> </div></div>
```

## Login Controller

```
import { Component, OnInit } from '@angular/core';
import { NgModel } from '@angular/forms';
import { GetDataService } from '../getdata.service';
import { Router } from "@angular/router";
declare var $: any;
@Component({
  selector: 'app-login',
  templateUrl: './login.component.html',
  styleUrls: ['./login.component.css'],
})
export class LoginComponent implements OnInit {
  constructor(private LoginService : GetDataService, private router : Router) {}
  username : String;
  password : String;
  roller : boolean = false;
```

```
  styles : string;
  user_class : string;
  message : string;
  user_message: string;
  ngOnInit() {}
  loginUser(){
    var user = {
      username : this.username, password : this.password
    };
    this.LoginService.loginUser(user).subscribe(res => {
      if(res.status){
        this.LoginService.storeUserData(res.token, res.user);
        this.styles = 'alert alert-success';
        this.message = res.message;
        this.roller = true;
        this.router.navigate(['/dashboard']); }else{
        this.styles = 'alert alert-danger';
        this.message = res.message; }));}}
```

## Register Page

```
<div class="container-fluid">
  <div class="row">
    <div class="col-md-3">
      <form (submit)="regUser()">
        <div class="form-group">
          <input type="text" [(ngModel)]="name" name="name" class="form-control" placeholder="Enter Name">
        </div>
        <div class="form-group">
          <input type="text" [(ngModel)]="username" name="username" class="form-control" placeholder="Enter Username">
        </div>
        <div class="form-group">
          <input type="email" [(ngModel)]="email" name="email" class="form-control" placeholder="Email">
        </div>
        <div class="form-group">
          <input type="password" [(ngModel)]="password" name="password" class="form-control" placeholder="Password">
        </div>
        <div class="form-group">
          <input type="submit" value="Register" class="btn btn-primary">
        </div>
      </form>
    </div>
  </div>
</div>
```

## Register Page Controller

```
import { Component, OnInit } from '@angular/core';
import { NgModel } from '@angular/forms';
import { GetDataService } from '../getdata.service';
import { Router } from '@angular/router';
@Component({
  selector: 'app-register',
  templateUrl: './register.component.html',
  styleUrls: ['./register.component.css']
})
export class RegisterComponent implements OnInit {
  constructor(private RegService : GetDataService, private router : Router)
  {}
  name : String;
  email : String;
  password :String;
  username : String;
  ngOnInit() {}
  regUser(){
    if(this.name == undefined || this.username == undefined || this.email
    == undefined || this.password == undefined){
      console.log('Please Fill All The Credentials');}else{
      var user = {
        username : this.username,
        name: this.name,
        email : this.email,
        password : this.password
      };
      this.RegService.RegisterUser(user).subscribe(res => {
      if(res.status){
        this.RegService.storeUserData(res.token,res.user);
        this.router.navigate(['/dashboard']);
      }});
      this.username = this.name = this.email = this.password = '';
    }
  }
}
```

## User Authentication Codes (Angular Frontend)

```
import { Injectable } from '@angular/core';
import { Http, Response, Headers } from '@angular/http';
import { Router } from '@angular/router';
import 'rxjs/add/operator/map';
import { tokenNotExpired } from 'angular2-jwt';
@Injectable()
export class GetDataService {
  public user: object;
  public token :any;
  constructor(private _http : Http, private router : Router) {}
  getUsername(){
    return JSON.parse(localStorage.getItem('user')).username;
  }
  fileUpload(username,fd){
    return this._http.post('/users/fileUpload/'+ username,fd).map((response : Response) => response.json());
  }
  storeUserData(token, user){
    localStorage.setItem('token', token);
    localStorage.setItem('user', JSON.stringify(user));
    this.user = user;
    this.token = token;
  }
  RegisterUser(user){
    return this._http.post('/users/register', user).map((response : Response) => response.json());
  }
  getCurUser(){
    var token = localStorage.getItem('token');
    let header = new Headers();
    header.append('Authorization',token);
    return this._http.get('/users/me',{ headers : header }).map((response :Response) => response.json()); }
}
```



```
this.token = null;
this.user = null;
localStorage.clear();
}}
```

#### Database Connectivity Code

```
const mysql = require('mysql');
var connection = mysql.createConnection({
  host    : '127.0.0.1',
  user    : 'root',
  password : '',
  database : 'mean'
});
connection.connect(function(err, connect){
  if(err){ console.log(err); }else{
    console.log('connected to database'); }
});
module.exports = connection;
```

#### Server side Code (Node.js)

```
const express = require('express');
const mysql = require('mysql');
const bodyParser = require('body-parser');
const nodemon = require('nodemon');
const path = require('path');
const passport = require('passport');
const nodemailer = require('nodemailer');
const morgan = require('morgan');
const cors = require('cors');
const app = express();
const mysql_con = require('./database/config');
const users = require('./route/api');
app.use(morgan('dev'));
app.use(cors());
app.use(express.static(path.join(__dirname, 'dist')));
app.use(bodyParser.json({uploadDir: './uploads'}));
```

## Server side Code (Node.js)

```
const express = require('express');
const mysql = require('mysql');
const bodyParser = require('body-parser');
const nodemon = require('nodemon');
const path = require('path');
const passport = require('passport');
const nodemailer = require('nodemailer');
const morgan = require('morgan');
const cors = require('cors');
const app = express();
const mysql_con = require('./database/config');
const users = require('./route/api');
app.use(morgan('dev'));
app.use(cors());
app.use(express.static(path.join(__dirname, 'dist')));
app.use(bodyParser.json({uploadDir: './uploads'}));
app.use(bodyParser.urlencoded({ extended : false}));
app.use(passport.initialize());
app.use(passport.session());

app.get('/', (req, res) => { res.send('Working Properly'); });
app.use('/users', users);
require('./auth/passport')(passport);
app.get('*', (req, res) => {
  res.sendFile(path.join(__dirname, 'dist/index.html'));
});
const port = process.env.PORT || 3000;
app.listen(port, function(){
  console.log('Server running On Port 3000');
```

```

public token :any;
  constructor(private _http : Http, private router : Router) {}
  getUsername(){
    return JSON.parse(localStorage.getItem('user')).username;
  }
  fileUpload(username,fd){
    return this._http.post('/users/fileUpload/'+ username,fd).map((response : Response) => response.json());
  }
  storeUserData(token, user){
    localStorage.setItem('token', token);
    localStorage.setItem('user', JSON.stringify(user));
    this.user = user;
    this.token = token;
  }
  RegisterUser(user){
    return this._http.post('/users/register', user).map((response : Response) => response.json());
  }
  getCurUser(){
    var token = localStorage.getItem('token');
    let header = new Headers();
    header.append('Authorization',token);
    return this._http.get('/users/me',{ headers : header }).map((response :Response) => response.json()); }
  loginUser(user){
    return this._http.post('/users/authenticate', user).map((response : Response) => response.json());
  }
  checkUsername(username : String){
    return this._http.get('/users/checkusername/'+ username).map((response : Response) => response.json());}
  loggedIn() {
    return tokenNotExpired(); }
  logout(){

```

## Rest API's Code

```
const express = require('express');
const router = express.Router();
const jwt = require('jsonwebtoken');
const passport = require('passport');
const bodyParser = require('body-parser');
const con = require('../database/config');
const db = require('../app/db');
const bcrypt = require('../app/bcrypt');
const secret = require('../database/main');
const nodemailer = require('nodemailer');
const upload = multer({storage: storage});

router.post('/register', (req, res) => {
  if(req.body.username == '' || req.body.username == undefined || req.body.password == '' || req.body.password == undefined ){
    res.json({ status : false , message : 'Please Provide All The Credentials'})}else{
    var user = {
      username : req.body.username,
      name : req.body.name,
      password : req.body.password,
      email : req.body.email
    }
    db.createUser(user, function(err, result){if(err){
      res.json({ status : false , message : 'User Already Exists'});
    }else{
      var id = result.insertId;
      db.findUserById(id, function(err, user){
const token = jwt.sign({data : user}, secret.secret, { expiresIn: 604800 })
;
      res.json({ status : true, message : 'User Registered', token : 'JWT '+ token, user: { username : user[0].username } }); }); }); }); }); });
router.post('/authenticate', (req, res) => {
  var username = req.body.username;
  var password = req.body.password;
```

```

        }else{ const token = jwt.sign({data : user},secret.secret, { expiresIn: 604800 });
        res.json({ status : true, message : 'User Authenticated', token : 'JWT '+ token , user : { username : user[0].username }});
    } }); } }); });
router.get('/checkusername/:username', (req,res) =>{
    var username = req.params.username;
    if(username == '' || username == undefined){
        res.json({ status : false, message : 'Please Provide Username'});
    }else{
        db.findUserByUsername(username, (err,user) => {
            if(user.length){
                res.json({ status : true, message : 'username already Exist
s'});}else{
                res.json({ status : false});
            } }); } });
router.get('/me', passport.authenticate('jwt', { session : false } ), (req,
res) => {
    let user = req.user[0];
    delete user['password'];
    res.json({ user : user});
});
router.get('/create', (req, res) => {
    res.send('Working Properly');
});
module.exports = router;

```

### Database Querying Code

```

const con = require('../database/config');
const bcrypt = require('bcrypt');
var db = {};
module.exports.createUser = function(user,callback){
    bcrypt.createHash(user.password, function(err, hash){
        let cmd = "insert into users (name, username, email, password)
values ('"+user.name+"', '"+user.username+"', '"+user.email+"', '"+hash+"'");
        con.query(cmd, callback);    });}
module.exports.findUserByUsername = function(username,callback){
    let query = "select * from users where username = '"+ username + "'";
    con.query(query , callback);}
module.exports.findUserById = function(id,callback){
    let cmd = "select * from users where user_id = '"+ id + "'";
    con.query(cmd, callback);}
module.exports.uploadImage= function(username,image, callback){
    let cmd = "update users set image_url = '"+image+"' where username = '"+
username+" "'";
    con.query(cmd,callback);}

```

```

    db.createUser(user, function(err, result){if(err){
    res.json({ status : false , message : 'User Already Exists'});
    }else{
        var id = result.insertId;
        db.findUserById(id, function(err, user){
const token = jwt.sign({data : user}, secret.secret, { expiresIn: 604800 })
;
    res.json({ status : true, message : 'User Registered', token : 'JWT '+ tok
en, user: { username : user[0].username} }); }); }); }); });
router.post('/authenticate', (req, res) => {
    var username = req.body.username;
    var password = req.body.password;
    if(username == '' || username == undefined || password == '' || password
== undefined) {
        res.json({ status : false , message : 'Please Provide All The Crede
ntials' });
    }else{ db.findUserByUsername(username , (err, user) => {
        if(err){ console.log(err); }
        if(!user.length){
            res.json({ status : false , message : 'Username Does Not Ex
});}else{
            bcrypt.comparePass(password, user[0].password, function(err, match){
                if(!match){
                    res.json({ status : false , message : 'Password Does Not Match' });

```

## **CHAPTER 5**

### **CONCLUSIONS**

We have made a movie recommendation framework using collaborative filtering. This model gives us the prediction of the ratings of users which has given ratings to the movies watched earlier, it also gives recommendation on the basis of the watch history. Also, our system gives the nearly accurate movie recommendations.

#### **Future extension:**

- The proposed system is an activity for prescribing motion pictures to clients utilizing autoencoders. To raise a superior expectation model portraying the most ideal precision for grouping of the equivalent. In community sifting, we have an issue of sparsity of information. Not many clients really rate a similar motion picture.
- Connect Python anticipating models to the Node.js for the information rendering to the backend.
- In the half and half approach, we can utilize more highlights to show signs of improvement expectations.

There are a great deal of ways to deal with build up the work done in this assignment. Directly off the bat, the substance based methodology can be stretched out to fuse more criteria to help sort the films. The most clear contemplations is to add features to suggest films with normal on-screen characters, chiefs or creators. Moreover, motion pictures released inside a comparative time period could in like manner get a lift in likelihood for recommendation.

## REFERENCES

- K. Kazuya and M. Yutaka, “**The application of deep learning in recommender system,**” Proceedings of the 28th Annual Conference of the Japanese Society for Artificial Intelligence,
- **Movie Recommender System** by Prateek Sappadla, Yash Sadhwani, Pranit Arora;NYU Courant
- **Algorithms and Methods in Recommender Systems** by Daniar Asanov; Berlin Institute of Technology, Berlin, Germany **A Neural Engine for Movie Recommendation System** by Md. AkterHossain and Mohammed Nazim Uddin ;Department of CSE, East Delta University, Chattogram, Bangladesh, School of Science, Engineering and Technology, East Delta University, Chattogram, Bangladesh
- **Movie Recommendations Using the Deep Learning Approach** by Jeffery Lund and Yiu- Kai Ng, Computer Science Department, Brigham Young University, Provo, Utah 84602, USA
- **Cold Start, Warm Start and Everything in Between: An Autoencoder based Approach to Recommendation** by Anant Jain and Angshul Majumdar, IIT Delhi, New Delhi, India
- **Expanded Autoencoder Recommendation Framework and its Application in Movie Recommendation** by Baolin Yi, Xiaoxuan Shen, Zhaoli Zhang and Jiangbo Shu, Hai Liu, Member, IEEE, national Engineering Research Center for e-Learning, Central China Normal University, Wuhan Hubei, China
- **HDNN-CF: A Hybrid Deep Neural Networks Collaborative Filtering Architecture for Event Recommendation** by Lixin Zou, Yulong Gu, Jiaxing Song, Weidong Liu, Yuan Yao, Department of Computer Science and Technology, Tsinghua University, Beijing, 100084, China





## ORIGINALITY REPORT

10%

SIMILARITY INDEX

5%

INTERNET SOURCES

0%

PUBLICATIONS

8%

STUDENT PAPERS

## PRIMARY SOURCES

1	<a href="https://pdfs.semanticscholar.org">pdfs.semanticscholar.org</a> Internet Source	3%
2	Submitted to Jaypee University of Information Technology Student Paper	3%
3	Submitted to Higher Education Commission Pakistan Student Paper	1%
4	Submitted to University of Mauritius Student Paper	1%
5	Submitted to University of the West Indies Student Paper	<1%
6	Submitted to University of Sheffield Student Paper	<1%
7	Submitted to King's College Student Paper	<1%
8	Submitted to The Hong Kong Polytechnic University Student Paper	<1%

**JAYPEE UNIVERSITY OF INFORMATION TECHNOLOGY, WAKNAGHAT**  
**PLAGIARISM VERIFICATION REPORT**

Date: 16/07/2019

Type of Document (Tick):  PhD Thesis  M.Tech Dissertation/ Report  B.Tech Project Report  Paper

Name: Ayush Agrawal Department: CSE Enrolment No 161279

Contact No. 9805112238 E-mail. ayush21.juit@gmail.com

Name of the Supervisor: Dr. Amit Kumar Jakhur

Title of the Thesis/Dissertation/Project Report/Paper (In Capital letters):  
MOVIE RECOMMENDATION SYSTEM

**UNDERTAKING**

I undertake that I am aware of the plagiarism related norms/ regulations, if I found guilty of any plagiarism and copyright violations in the above thesis/report even after award of degree, the University reserves the rights to withdraw/revoke my degree/report. Kindly allow me to avail Plagiarism verification report for the document mentioned above.

**Complete Thesis/Report Pages Detail:**

- Total No. of Pages = 73
- Total No. of Preliminary pages = 13
- Total No. of pages accommodate bibliography/references = 2

*Ayush Agrawal*  
 (Signature of Student)

**FOR DEPARTMENT USE**

We have checked the thesis/report as per norms and found Similarity index at 10.....(%). Therefore, we are forwarding the complete thesis/report for final plagiarism check. The plagiarism verification report may be handed over to the candidate.

*Juit*  
 (Signature of Guide/Supervisor)

Signature of HOD

**FOR LRC USE**

The above document was scanned for plagiarism check. The outcome of the same is reported below:

Copy Received on	Excluded	Similarity Index (%)	Generated Plagiarism Report Details (Title, Abstract & Chapters)	
Report Generated on	<ul style="list-style-type: none"> <li>• All Preliminary Pages</li> <li>• Bibliography/Images/Quotes</li> <li>• 14 Words String</li> </ul>	Submission ID	Word Counts	
			Character Counts	
			Total Pages Scanned	
			File Size	

Checked by  
 Name & Signature

Librarian

Please send your complete thesis/report in (PDF) with Title Page, Abstract and Chapters in (Word File) through the supervisor at [plagcheck.juit@gmail.com](mailto:plagcheck.juit@gmail.com)