“Hybridizing a Recommendation System which incorporates collaborative and content-based filtering for an online movie streaming platform.”

Author: Rohan Khanolkar

Department of Computer Science and Information Systems,

Birkbeck College, University of London,

2021

Project submitted in partial fulfilment of the requirement for the

MSc. in Data Analytics (Advanced Computing Technologies)

This project is substantially the result of my own work, expressed in my own words except where explicitly indicated in the text. I give my permission for it to be submitted to the plagiarism detection service.

Table of Contents

Abstract:

An algorithm which aims to suggest a user with relevant products/items which could attract a user to click/view/buy/rate the product/item depending on the website or application the user is logged into; this is known as a recommendation system. In today's world, organisations use big data to make relevant recommendations to their customer base to enhance their sales and improve customers loyalty. In some industries, recommendation systems are a crucial part of the organisation which help to generate huge income if this system is extraordinary as compared to their competitors. The basic structure of the recommender system consists of a filtration process according to the user rating/like/dislike. There are various types of algorithm of the recommendation models used in the present day scenarios and we must choose the appropriate model according to the requirement of the business along with analysing the limitations.

In this project, I have built a hybrid filter system which can give personalised recommendations to the user in an online movie streaming platform. This machine learning project involves building a recommender system which can analyse the data and create an indexed memory map to find relevant movies and recommend the user a list of movies which could be of his/her interest. These indexed memory map is built by using filters such as collaborative filter and content-based filter to give accurate prediction. However these filter have a downside of using them individually to obtain recommendations which are discussed in this report, and thus they need to be combined with each other to get better recommended accuracy.

In this project I have individually built collaborative filter with and with-out weighted mean, content-based filter and neural network model, the output prediction are compared using the Root mean squared error and Mean absolute error to understand which output can be combined (create a hybrid) to give the best recommendation results. In conclusion this system shows, how the algorithm gives movie recommendation for an individual user, as well as all the users and give an average prediction accuracy and why the hybrid filter should be considered over others.

1. Introduction

In the year 2000, there was a famous experiment conducted by psychologists from Columbia University and Stanford University which was known as 'The Jam Experiment' [32]. This experiment was part of a field study at a grocery store inside a local food market, in this study there was a food tasting booth which had 24 varieties of jam displayed on one day and on the other day there were 6 varieties of jam displayed on the same counter the next day. This experiment was done to adjudge which kind of booth (24 jams booth or 6 jam booth) will have more sales. Assuming that more variety of jam would generate more sales, more sales could generate more footfall, and both eventually leading to increase in sales. However, the study report observed a different phenomenon, where the booth with 24 variety of jams had more public interest, but the sales conversion was 10 times lower than the booth which had 6 varieties of jams. So, the results of this experiment showed that the booth with 24 varieties of jams had attracted 60% of the shoppers out of which only 3% were sold. On the other hand, the results of this experiment showed that the booth with 6 varieties of jams had attracted 40% of the shoppers out of which only 30% were sold. The conclusion of the experiment showed that a variety of choices only seems appealing but it becomes confusing to choose and indirectly hampers the sales, this is also known as ‘Paradox of Choice’ [32]. This conclusion can also be derived in a an online store which has millions of choices of products available, where a recommendation system can prove to be of good use to filter out products of user’s interest and enhance the website experience of the user and increase the sales.

1.1. Aim of the Project

This Project aims to prove how a recommendation system with a hybrid filter that consists of collaborative and content-based filtering, performs better than neural network and other conventional models, when applied on a movie streaming platform/website [1].

1.2 Background history and problem statement

Since the last 12 years there have been many improvements in using the internet in our day to day life which has been a global phenomenon. In the beginning of the year 2000, it was clearly understood to the developers that the amount of information required for the use of internet/information would increase so much that it would amount to an overload of data which could be irrelevant to most of the users to identify the use of such data [2]. This overload of data was partially segregated by the application of search engines which was beneficial to an extent of finding information but it was not helpful to give personalised choices for the user request. Asanov, D. mentioned the tools of the old recommendation system which were used for filtering and sorting by considering the opinions of fellow users to determine the choices and interest of a user by advising them to fill-up a form which would help to identify user interest [3].

Balabanović, M. and Shoham, Y. (1999) [4] discussed about recommender system which can have 'generic advantages' when it is used in a combination of collaborative filtering setup (where the common taste of user is acquired to recommend) and

content-based filtering setup (where the recommendation is acquired by the user profile/genres). An association of interest could be developed, and it is also discussed in their paper which states that the user scaling and document scaling can be advantageous .

A famous online streaming platform for movies known as Netflix in 2006 was facing issue in making personalization in their recommender system called as 'Cinematch[SM]' and thus they rolled-out a competition with a prize money of $1 million to resolve and identify the issue and increase the efficiency of the model by recognising patterns from their dataset about likes or dislikes of a user and give personalised recognition based on the user history [4]. Three years after the competition was announced in 2009, the "Bellkor's Pragmatic Chaos" team was awarded as the winner of the competition of the Netflix Prize. Their solution for recommendation system was able to improve the overall efficiency of Netflix 'Cinematch[SM]' by 10.06% over its existing efficiency [5]. The "Bellkor's Pragmatic Chaos" research was only focused on content-based filtering and collaborative filtering along with the help of other standard techniques to identify and document user reactions (likes/dislikes) for a given movie and predict user ratings and create a recommendation list that would be personalised for a user according to their past history.

A movie recommendation system usually faces an issue where a list of movies are unwatched or unrated by any user and could give incorrect predictions / recommendations to a user. Thus, the aim for this project is to understand how a hybrid filter (collaborative + content-based filter) could give a better accuracy than the conventional recommender models.

2. Literature Review

2.1. Applications of recommendation system

A recommendation system is a widely used programming algorithm which is used for personalising the preference of a user and matching it with other users to promote various products and to keep the engagement of users with the application/website the user is registered with. Below are some of the applications of recommendation systems :

1) Online streaming platform - To stream music/videos/movies where the system recommends which music/videos/movies to watch next. Examples of such platforms are Apple Music, Spotify Music, Youtube Videos, Netflix movies etc.
2) Online marketplace - There are various websites to buy products, furniture, groceries etc. where the system keeps creates a user profile of past purchases and their taste to recommend similar products for future purchase (product promotion). Examples of such marketplace websites are Amazon, E-bay etc.
3) Social media websites - The system promotes new users, recommends common friends, recommends videos or pictures liked by other users. Examples of such social media websites are Facebook, Instagram, Linked-In etc.
4) Navigation - Websites and applications like google maps, bing maps, apple maps etc. recommends user nearby locations, restaurants, museum etc.
5) Search engine - This is one of the most used systems in the world which gives us recommendation of sentences while typing on the search engine. Examples of such search engine websites are Google, Yahoo, Bing etc.

And much more.

2.2. Associated research in recommender system

E. Çano and M. Morisio. [7] researched 72 papers on recommendation system which had a hybrid filtration process and concluded that most of the filters used in recommender systems are hybrid which consists of weighted average technique for collaborative filtering along with content-based filtering and few others. Their research consisted of an evaluation by comparing the matrix's accuracy of various systems which were similar to each other which were not providing credible evaluation. Their research also found other challenges related to evolving the user context and use test across various domains of recommendation systems. In this project I have used the same weighted average technique to get better accuracy testing results.

2.2.1. Recommender system challenges and solution

Their were many challenges and solutions discussed in E. Çano and M. Morisio.[7] paper out of which there were few challenges which were thoroughly researched by various researchers as mentioned below :

a) Cold Start - Cold start is a problem which occurs when a new user is registered in a system and a system finds it difficult to predict the user choices as their profile does not contain likes/dislikes/reviews; the system does not understand what to suggest for such user [8]. Maneeroj, S. and Takasu, A. [9] found a solution to cold start users by using a probabilistic model which generates accurate ratings by extracting the latent features even when a new user profile is empty (without likes/dislikes/reviews) by showcasing the user with top three products which are best in each category and acquire the features by observing the user behaviour to the provided list of top three products in each category. Chughtai, M. et. al. [10] researched on e-learning domain and tried solving the cold start problem by combining content based filtering along with collaborative filtering with a weighted output which resulted in a rise of accuracy in a situation where new users register. This project is also similarly base on models used by Chughtai, M. et. al. in his research.

b) Sparsity of data - This issue occurs when a user watches a movie but does not review it when the catalogue of movie is huge in number so there is insufficient data to accurately recommend user likings. Coa, J. et.al [11] managed to resolve data sparsity issues by using a factorisation model that relates a user, domain and items. Wang, J. et.al. [12] used a predictor which consists every item rating and user information for every missing ratings, then they found the estimate by merging the rating of items which are similar to other users. After that, they also consider different items rated by same user with the ratings provided by similar user on the similar items. Wang, J. et.al. also provided a substitute solution by interlinking ' Naive Bayes' along with collaborative filtering and obtaining the same results.

c) Accuracy of recommendation - This phenomenon occurs when there is a lack of user input and when the data is sparse; it considerably reduces the ability of the system to recommend accurately. Campos, L. et.al. [13] resolved this accuracy issue by applying the Bayesian network model which has various nodes such as users, features, items i.e. linked with content-based filtering and collaborative filtering to obtain accurate recommendation.

d) Scalability issue - As the number of users and movies are growing, the resource which is required for information processing to give recommendations is debilitating and it can only be resolved by adding more memory space virtually and physically [8], this issue is known as scalability issue. Choi, S.and Jeong,Y.[14] used distance-to-boundary on content-based filtering and Pearson correlation on collaborative filtering which helped them to find the neighbours which are furthest and closest which allowed them to compress the data-set avoiding the sparsity issue and improving computational time. Ghazanfar, M.A. [15], improved scalability issue by using collaborative filtering and combined it with SVM and Naive Bayes.

2.2.2. Building models for machine learning

In the paper of E. Çano and M. Morisio.[7],they had discussed about building machine learning models in the recommendation systems and mentioned that K-Nearest Neighbour (KNN) could be combined with collaborative filtering to analyse and sort nearest neighbours within the users who have similar taste/characteristic/product. They also mentioned the use of Singular Value Decomposition (SVD), Principal Component Analysis (PCA) and Latent Dirichlet Allocation (LDA) can be used for matrix manipulation which can lower the error rate in collaborative filtering. Above methods gained recognition after the results for the Netflix challenge in 2009. On the other hand, Choi, S. and Jeong,Y.[14] used Bayesian Network with the probabilistic reasoning for predicting ratings on collaborative and content-based filtering where the weights were auto-selected which could adapt to a specific condition a model that helps in improving the accuracy of recommendations. This project is building a hybrid filter with KNN and a machine learning model (like Keras Neural Network) to assess if better accuracy is obtained by using this hybrid filter.

3. Description and base concepts of the project

There are many approaches to a modern day recommender system some of them uses context aware, semantic etc. Various technologies are used these days to obtain efficient results for recommending a personalised and specific user requirement [15]. The most popular approaches to a recommender system are content-based filtering and collaborative filtering. These filtering processes are the base concepts of every recommender model which are used in recent times [2].

3.1. Content based filtering

When a new user is registered with the system a profile is created during their registration process, this profile is used to overlook in the recommender system which is functioning on a content-based filtering approach. The user information from this profile which is generated by the registration process gives information like the genre of movies which a user is interested in [2]. In content based filtering process there is a comparison between previously rated movies and unrated movies by the user and the filter tries to find similar movies of the same genre which are not watched by the same user and recommend it to them. Content-based recommendations are purely based on the user information (genres) and not by other user ratings [2].

Movies	The Mask	Gladiator	Troy	Spartacus
Ratings	8	7	9	10

Figure 1: The movies the user has watched

Movies	Genre	Leading actor	Colour	Year
Gladiator	Drama	Russell Crowe	coloured	2000
Robin Hood	Drama	Russell Crowe	coloured	2010
Making A Living	Comedy	Charlie Chaplin	w/b	1914
...	...	...	...	...

Figure 2: The movies list

Figure 1 [2] is an example of movies watched by a user and their ratings given by the same user. Figure 2 [2] is a list of movie names with their details (i.e. genres, leading actors, colours, year). The algorithm of filter based filtering will find movies from the movie based database and create a comparison between the movies watched and rated by the user and accordingly find movies with a similar genre and actor to create a recommendation list for the user to watch.

3.2. Collaborative filtering

The method of finding users who have similar taste and rating of movies is known as collaborative filtering approach [2]. The collaborative filtering classifies the users into various groups of having similar taste, where the users in the group get recommendations based on ratings given by other users of the same group.

Movies Users	Titanic	Gladiator	Black Swan	The Fighter	TRON: Legacy
A	8	7	9	10	-
B	9	7	9	9	10
C	9	8	9	8	9

Figure 3: Collaborative recommender system example

Figure 3 [2] represents three users, who are in the same group, having similar movie tastes. The missing values in the group are unrated movies by the particular user. In the above case this unrated movie by user A from the above group will be recommended the unrated movie (i.e. TRION:Legacy movie). Collaborative filtering

process in the above table focuses primarily on the taste of users in movies unlike other filtering methods which only note the genre of movies. This filtration process is widely used in e-commerce platforms (like Amazon etc.) where product recommendations are given to the user according to the ratings given by other users from the same group. There are further three types of approaches in collaborative filtering: they are user-based, item-based and model-based.

3.2.1. User-based collaborative approach

A University of Minnesota researcher, Herlocker, J in the year 1990 [16], proposed his method of emphasizing on the appreciation by the user to be a key factor for recommending. He further elaborated that the users who have similar preferences in taste could be grouped together and the output recommendation provided to a user can be derived on the basis of other user preferences from the same group. Thus, this approach plays an important role in determining recommendations to a user in a group.

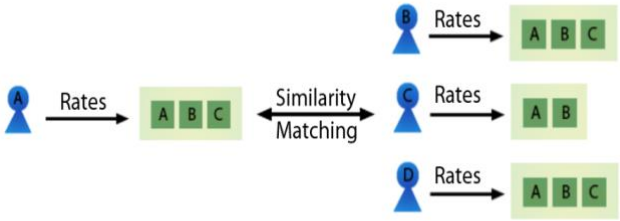


Figure 4: User-based collaboratie recommender system

3.2.2. Item-based collaborative approach

A University of Minnesota researchers, Sawar, B. and Konstan, J in the year 2001 [17] proposed their method which emphasized on changing preference taste of a user in a group. The system will thus generate a recommendation list by analysing the items from the group which the user will prefer to watch.

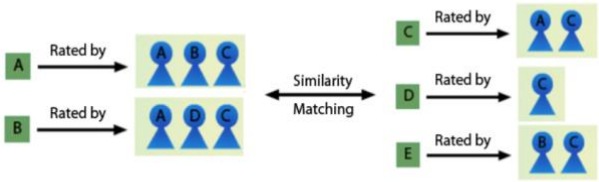


Figure 5: Item-based collaboratie recommender system

3.2.3. Model-based collaborative approach

The model based collaborative approach is best known as a hybrid approach where techniques from content-based filtering and collaborative based filtering are combined together to give accurate results in a recommendation system [2]. This approach is used to avoid limitations and complications which a classic recommender system faces (eg. new user problems or cold start issues) etc. In this project, I will would be implementing a hybrid approach in filtration process using the following method :

a) Separately implementing collaborative and content-based algorithms and combining the results of both the approaches.
b) Using a content-based filtering approach by applying some collaborative filtering rules.

c) Using a collaborative based filtering approach by applying some content-based filtering rules.
d) Configuring the recommender model which jointly combines rules of collaborative and content-based filtering along with a ranking algorithm to rank their outputs.

3.3. About data set

For this project, I was first considering to use the “Netflix Prize data” dataset which is available on Kaggle [18]. This Netflix data set consists of four files; namely combined_data_1.txt, combined_data_2.txt, combined_data_3.txt and combined_data_4.txt; which was approximately 2.6 GB datafile consisting of 100 millions rows of data. Apart from this, there were more than 17,000 movie titles in the file named as movie_titles.csv. Initially I tried to use the entire dataset to create utility matrix for collaborative filtering which creates a user vs movie_id matrix upon the ratings given by the particular users, so to create a matrix out of a dataset which consists of 100 million rows of data was becoming computationally difficult for this project as the kernel get stuck in this process. I also tried using individual datafiles (combined_data_1 , 2 , 3 , 4) to check if the kernel gets stuck while creating a utility matrix, and found that the kernel works perfectly but there are user id's which were found missing from one datafile to another. This means that, dataset consists user id's which are available in one dataset but not in other so I need to use the entire dataset run it on the algorithm and using the entire dataset (2.6 GB) was computationally difficult on my existing computers. To run such dataset, I will have to use spark and run batch jobs on it. This is a classic case of scalability issue with regards to the size of dataset. Thus I took a decision of changing it to a smaller and computationally inexpensive dataset.

For this project, a new dataset is considered from Kaggle known as 'Movie Recommender System Dataset' by Manas Garg [19]; the algorithm is the same the project is only trying to prove the feasibility of the algorithm. This dataset consists of two files known as 'movies.csv' and 'ratings.cvs' which is approximately 3 MB in file size. The 'movies.csv files' consists of 'movieid', 'title' and 'genres' which consists of 9737 rows of data which are individual movie titles classified by their genres (eg. movieid : 1, title: Toy Story (1995), genres : Adventure | Animation | Children | Comedy | Fantasy). The 'ratings.csv files' consists of ‘userid’, 'movieid', 'rating' and 'timestamp' which consists of 101,000 rows of data which are individual movie titles classified by their genres (eg. userid : 1, movieid: 3, rating: 4.0, timestamp: 964981247). So, this data being small it can be easily used on a local computer with sufficient memory space.

This dataset do not have a classification variable (i.e. 0 or 1 / Yes or no, etc.); thus, it is a regression scenario for using machine learning algorithm. The dataset do not have a separate datafile for training and testing; therefore will use train-test split ad divide it into training and testing dataframe. This data do not consist on any empty rows or empty fields nor this data is inconsistent, so do not require any data cleaning for these datasets.

4. Approach for this project:

4.1. Architecture plan for hybrid recommendation plan

In this project, the focus is using a hybrid filter to achieve better recommendation than other convention filtering processes. This approach was improved from the previous approach submitted in the project proposal in April 2021.The new recommendation system architecture is given below which shows various steps involved in giving an optimum recommendation of movies to a particular user along with the recommendation accuracy compared to the ratings provided in the dataset and predicted by the recommender model.

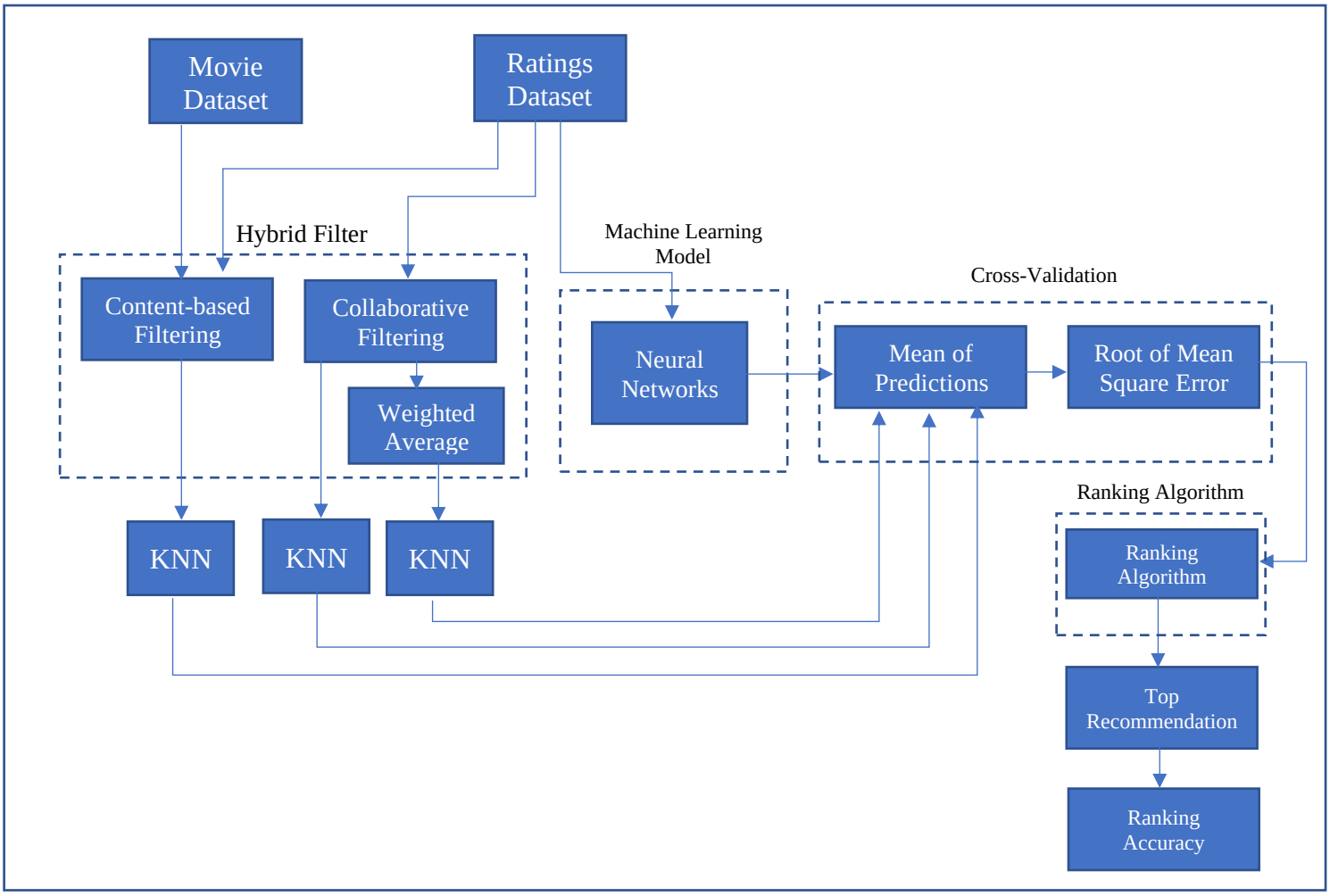


Figure 6. Architecture of recommender system model with hybrid filter

4.1.1. Data Processing

In data pre-processing, I had first loaded both the datasets using 'pd.read_csv' from the Pandas library [20] and checked if there were any blank / empty rows, columns, cell in the dataset I load the data and convert it into a data frame 'user_rating_data' for 'ratings.csv' files and 'generic_additional_data' for 'movies.csv dataset'. After checking for any abnormalities in the dataset, I found that the dataset is perfect and it is ready

to use in the recommender model. In the 'user_rating_data' dataframe and use 'test_train_split' from 'sklearn.model_selection' library [21] where this data is divided into 80% as training data and 20% testing data. The test and frames are now ready to be used for the filtration process.

4.1.2. Hybrid Filter

The hybrid filter is divided in two parts i.e collaborative filtering and content based filtering, where the output of both will be compared using the mean of the output of both filtration process. To begin with, I start with collaborative filtering and first create a utility matrix using that consists of 'user_id' as rows and 'movie_id' as columns and the values inside each cell is represented as the ratings given to a movie by a user. This utility matrix is created by using the pivot function from 'pandas.DataFrame' library [22] where I define index, columns and values to create this matrix. I have noted, there are many 'NaN' values inside the matrix which means that users have not watched and rated those movies. Thus replacing all the 'NaN' values using the 'fillna' function from Pandas library [23] and filled these 'NaN' values with zero.

4.1.2.1. Collaborative Filtering

In collaborative filtering, we have first created a utility matrix as mentioned in the above section. After creating the utility matrix, I found the cosine similarity between the utility matrix created and convert the output of this cosine similarity into the dataframe. This cosine similarity is found using 'cosine_similiarity' from the 'sklearn.metrics' library [24]. Which identifies how close are the users based on the ratings provided for movies using cosine similarity on the utility matrix with itself. From this cosine similarity matrix, generate a user-user similarity matrix by using the 'pd.DataFrame' from Pandas library [25] which converts the cosine similarity matrix into a dataframe by setting the indices to 'user_id'. The matrix which is created shows how much the users are closer to each other according to their pattern of rating the movies. This matrix operation is similar to finding the 'K Nearest Neighbour' (KNN) as it has found its cluster of users with similar taste and ratings.

After finding the nearest neighbour, find the predicted rating for every record in the test dataframe. For better accuracy, I used the predicting function twice for finding the prediction without weighted average and with weighted average.

ALG_CB_UWD (Algorithm for Collaborative Filtering – Without Weighted Average)

INPUT :

- Test DataFrame (test records for which rating has to be predicted , which is passed a parameter to the function)
- User-User Similarity matrix (cosine-similarity between users based on user ratings, which is globally declared)
- User Rating DataFrame (from input data file, , which is globally declared)

OUTPUT:

- Array of predicted ratings for test records

Algorithm:

Step 0 : ***Declare*** *prediction array to be returned*
Step 1 : ***for*** *every record 'r' in test dataframe.*
Step 2 : get r.user_id and r.movie_id
Step 3 : ***if*** *r.user_id exist in user-user similarity matrix.*
Step 4: get cosine-similarity distance between all users and r.user_id.
Step 5: sort users based on similarity distance in the descending order.
Step 6: get user_id's of top K (K=20 users)
Step 7: get input data of top K users from 'User Rating DataFrame' (user_id, movie_id and rating).
Step 8: filter out data for r.movie_id
Step 9: calculate mean of ratings for r.movie_id and append to prediction array
Step 10 : ***else*** *append zero to prediction array.*
Step 11 : ***return*** *prediction array*

ALG_CB_WD (Algorithm for Collaborative Filtering –Weighted Average)

INPUT :

- Test DataFrame (test records for which rating has to be predicted , which is passed a parameter to the function)
- User-User Similarity matrix (cosine-similarity between users based on user ratings, which is globally declared)
- User Rating DataFrame (from input data file, , which is globally declared)

OUTPUT:

- Array of predicted ratings for test records

Algorithm:

Step 0 : ***Declare*** *prediction array to be returned*
Step 1 : ***for*** *every record 'r' in test dataframe.*
Step 2 : get r.user_id and r.movie_id
Step 3 : ***if*** *r.user_id exist in user-user similarity matrix.*
Step 4: get cosine-similarity distance between all users and r.user_id.
Step 5: sort users based on similarity distance in the descending order.
Step 6: get user_id's of top K (K=20 users)
Step 7: get input data of top K users from 'User Rating DataFrame' (user_id, movie_id and rating).
Step 8: filter out data for r.movie_id
Step 9: ***for*** *every record in the filtered data*
*Step 10: Calculate weighted rating (Rating * Similarity distance) and add to the total rating.*
Step 11: ***if*** *filtered data exist*
Step 12: divide total rating / sum of all ratings and append to prediction array.
Step 13: ***else*** *append NaN to prediction array*
Step 14: ***else*** *append zero to prediction array.*
Step 15 : ***return*** *prediction array*

The above function calculates the predicted ratings by first reading the test data frame and iterating each row of the dataframe by to obtain its 'user_id' and 'movie_id'. Now, compare it by checking if the user-user matrix contains user information. Further, the cosine similarity of the test record is calculated with other user id's. After this the entire

output of the cosine similarity is sorted in descending order and only top 20 users (i.e. KNN technique). This is the group of the closest users. Now these top users are considered and only the movies that are rated by them are recorded. The movies which are not rated are discarded. The ratings which are recorded are formed in a list and their mean value is calculated. This value is appended to the prediction array.

The above sequence is done for calculating the predicted ratings without weighted average. Now, for calculating the predicted rating with weighted average (note the highlighted algorithm), I need to further consider the similarity distance. This rating is calculated by adding the product of rating and similarity distance of all users. It is further divided by the sum of all ratings to find the weighted average. This ensures that more weightage is given to the rating by the more similar user. If there exist no records with ratings, then append 'NaN' value, else append the weighted average to the prediction array.

After the above two conditions are satisfied, the function checks if the user is new in the system, if not it will append zero to the prediction array. The prediction array is returned to the calling line of code. After this step both the weighted and unweighted predicted ratings are concatenated to the test dataframe as CB_Pred and CB_Pred2. Now check if there are in any 'NaN' values.

ALG_NAN (Algorithm for to check for NaN values)

INPUT :

- Test DataFrame (test records consisting of predicted ratings, which is passed a parameter to the function)

OUTPUT:

- DataFrame consisting of NaN values.

Algorithm:

Step 0 : get dataframe of NaN values, if present.
Step 1 : get row indices of above dataframe.
Step 2 : extract test data records of those row indices into a datafram
*Step 3 : **return** dataframe*

The above algorithm checks if there are any NaN values present in the test dataframe and returns all the rows which consist of 'NaN' values. This function would help us, to check if the movie for which we need to predict rating is watched by any other user. If nobody has watched the movie i.e. there is no rating data, the predicted rating is assigned a NaN value. If there is a new user in the dataset we predict a zero rating which represents the cold start problem. A Cold start problem arises when there is a new user which is registered in a system. Their profile does not have any information about their likes or dislikes and thus leaving the system clueless about their user choices. Researchers like Maneeroj, S. and Takasu, A. [8] generated a latent features in there probabilistic model which was able to generate ratings even when a new user's profile did not contain any ratings. Other researchers like Chughtai, M. et. al. [9] were able to find a solution to this cold start situation by using a combination of collaborative and content based filtering along with merging their weighted output to find a better accuracy. In this project, I would be also following Chughtai, M. et. al. method to try and find better accuracy in the given situation.

4.1.2.2. Content based filtering

Content based filtering in a recommender system uses vectoriser to convert words into vectors and form a dictionary of these vectorised words and understand the user likes/dislikes and cluster movies with similar movie genres and further predict movies similar to the users taste. This is done using the 'Term Frequency-Inverse Document Frequency' (TF-IDF), the TF-IDF uses a technique to quantify a word in a given document and further computes the weight of to signify its importance in the document [26].The algorithm created for content based filtering is as follows :

ALG_CON (Algorithm for Content based Filtering)

INPUT: TEST DataFrame (passed as parameter)

User Rating Dataframe (user_id, movie_id, rating)

Movies DataFrame (movie_id, genere)

OUTPUT: Prediction Array

Algorithm:

Step 0: ***Declar****e prediction array to be returned*
*Step 1****: for*** *every record r in test dataframe:*
Step 2: Get all movies watched by r.user_id
Step 3: Declare a dictionary such that movie_id is the key and movie_genere is the value
Step 4: ***for*** *every movie watched by r.user_id*
Step 5: Get list of movieID and genere, append it to the dictionary
Step 6: Convert generes to a format acceptable by tf-idf
Step 7: Vectorize the records of genres indexed by movie_ids obtained
Step 8: Repeat Step 3 for a single record of r.movie_id
Step 9: Repeat Step 5, convert genre to tf-idf acceptable format and vectorize the record
Step 10: Calculate cosine similarity between r.movie_id and all other movies watched by user (similarity based on generes)
Step 11: Get movie_ids of top k (k=5) of the most similar user watched movies with respect to r.movie_id
Step 12: Get ratings given by r.user_id to these top k movies
Step 13: Calculate mean of the ratings and append it to the prediction array
Step 14: ***return*** *prediction array*

The above function calculates the predicted rating based on the content preferred by the user. Now, based on the genre of the movie the user has rated. To do this, we need to first iterate through every test record in the test dataframe and get the movies which the user has rated. Get the genres of the rated movies by the user and saved it as a profile of the user. Now for every movie the user has watched, the algorithm finds similar movies in the genres and fills the profile with the value of the genres. After this step, we need to remove the pipe separator (i.e. ' | ') and insert a space so that tf-idf vectorization can be processed on it. After this vectrization create a reference list which consists of movie id's and their value of movie genres into a dataframe. Now for the test records, repeat steps which are similar as done above. Now I calculate and check how close the movie in the test dataframe is from the movies in the user watchlist by using the method of cosine similarity and acquire their values as a separate dictionary. This dictionary is now sorted in order to the user watched movies compared to the closest movies in the test dataset and get the top

five user watched movies and acquire user ratings by calculating the mean of the ratings and append it as a predicted array. The above algorithm takes an approximate time of 2 hours and 30 minutes to process in the given computational speed available for processing. After acquiring the predicted values, concatenate them on the test dataset.

4.1.3. Neural Network Machine Learning model

To create a neural network machine learning model I used the Keras classifier which uses features like input, embedding, flatten, dot, dense from the 'keras.layers' library [27]. The above mentioned features of Keras library are used to create a neural network model where input is used as a point where data is inserted in the neural network, embedding is a layer of neural network which converts the input to a dense vector for processing the output, the flatten feature is a layer which flattens the input from the embedding layer without affecting the badge size but it adds an extra dimension as the data is without a predictor feature. The dense layer of Keras classifier holds the neurons and it serves as the processing memory of the neural network. These dense layers have an activation function which balances the weight of the neural network to give an accurate result. The algorithm for neural network machine learning model as follows :

ALG_NEURALNET (Algorithm for Ranking the predicted rating with neural network algorithm)

INPUT :

- Test DataFrame (test records consisting of user_id and movie_ids

OUTPUT:

- Array of predicted ratings for test records

Algorithm:

Step 0 : create an input layer for 'movie_id'.
Step 1 : embed the input layer with 'movie_id' and use the last id (bottom most number) in the matrix factorisation to minimise error.
Step 2 : create a flatten layer which takes input from embedded layer
Step 3 : create an input layer for 'user_id'.
Step 4 : embed the input layer with 'user_id' and use the last id (bottom most number) in the matrix factorisation to minimise error.
Step 5 : create a flatten layer which takes input from embedded layer
Step 6 : concatenate the vectors from step 2 (for movie id) and step 5 (for user id).
Step 7 : create the neural network Dense layer of 128, 32, 1 neurons.
Step 8 : create a model with one input layer as user id and another as movie id.
Step 9 : compile the model from step 8 with 'adam' optimizer and 'mean_squared_error' as the error function.
Step 10 : fit the features of the train dataframe.
Step 11 : predict on the features of the test dataframe.

The above algorithm predicts the ratings based on the neural network classifier using the Keras library. In this neural network architecture I first create input layers of movie id and embed it with the highest value of the movie id so that it understands the range

of the embedding process. Similarly another input layer for user id is created and embedded with the highest value of user id so that it understands the range of the embedding process. Now combining flatted input of the both (user id and movie id) to prepare it for processing in the neural network layer.

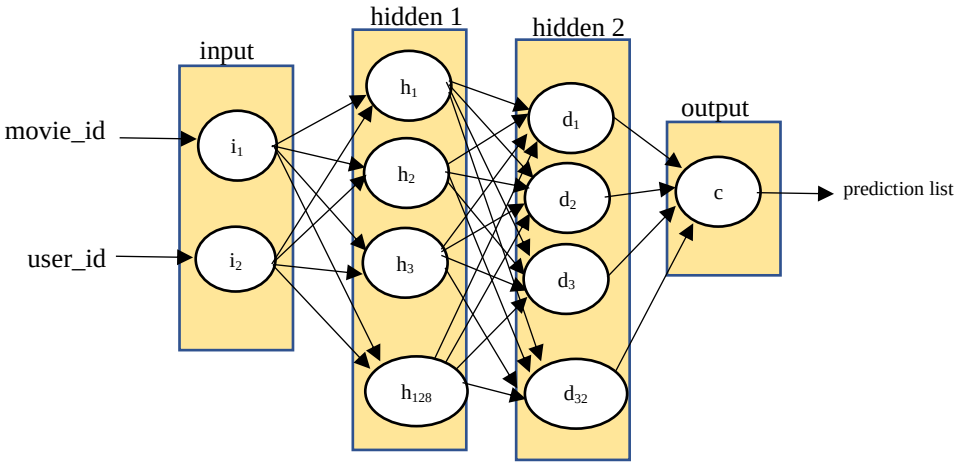


Figure 7. Neural Network Design for the Recommender System

The figure above (fig 7) is a neural network model which consists of 4 layers: input layers with two embedded input (i_1 & i_2), hidden layer 1 (Dense with 128 neurons, h_1 to h_{128}), hidden layer 2 (Dense with 32 neurons, d_1 to d_{32}) and output layer (Dense with 1 neuron). This model consist of an optimiser known as 'adam' [28] and an error function known as 'mean_squared_error' [29]. Apart from this, the flatted input of movie id and user id are inserted to this neural network model. The model is now trained on the training dataset and predicted on the test dataset. The predicted result of this test dataset is concatenated with the test dataframe.

4.1.4. Cross Validation

This layer in the recommender system algorithm consist of a simple algorithm where the predicted output of all the above layers i.e. Hybrid Filter (Content based filtering, weighted collaborative filter and unweighted collaborative filter) and Neural Network Model model in the test dataframe is combined and the mean value of different combinations are observed. These observations are discussed in the trials and observation section of this report. In this section, I first take the mean value of un-weighted collaborative filter and content based filter (i.e. mean of 'CB_Pred' and 'Content_Pred', output as 'CB_CP_Mean_Pred'), then I take the mean of weighted collaborative filter and content based filter(i.e. mean of 'CB_Pred2' and 'Content_Pred', output as 'CB2_CP_Mean_Pred') and a combined mean of weighted collaborative filter, un-weighted collaborative filter, content based filter and neural network (i.e. mean of 'CB_Pred', 'CB_Pred', 'NN_Pred and 'Content_Pred', output as 'Overall_Mean_Pred').

After finding the mean value of various combinations of predicted values; then I find the 'Root-Mean-Squared Error' (RSME) of each predicted column of output to find which one is the best amongst them. The RSME is an approach for measuring error in a model which derives prediction in quantitative data and it is also a better method to estimate the standard deviation of the error distribution [30]. The RMSE is calculated by finding the square root of the difference between predicted value to the

observed value divided by the number of observations for every iteration [30]. The formula can be derived as follows:

$$RMSE = \sqrt{\sum_{i=1}^{n} \frac{(\hat{y}_i - y_i)^2}{n}}$$

$\hat{y}_1, \hat{y}_2, \ldots, \hat{y}_n$ are predicted values
$y_1, y_2, \ldots, y_n$ are observed values
n is the number of observations

The lower the value of RSME in the estimated unit of the predicted values are known to be better predictions [30]. In our case, I will find the the RMSE value for each prediction and each mean prediction i.e. 'CB_Pred', 'CB_Pred', 'NN_Pred', 'Content_Pred', 'CB_CP_Mean_Pred', 'CB2_CP_Mean_Pred' and 'Overall_Mean_Pred' and assess which prediction to be considered for the ranking algorithm to find list of movies according to the predicted values.

After RMSE, I find the ‘Mean Absolute Error’ (MAE) rate of value for each prediction and each mean prediction i.e. 'CB_Pred', 'CB_Pred', 'NN_Pred', 'Content_Pred', 'CB_CP_Mean_Pred', 'CB2_CP_Mean_Pred' and 'Overall_Mean_Pred'. The MAE is a absolute mean value of every prediction on the test dataset for every instance, MAE also shows the distance between the reference point and predicted point [31]. The formula can be derived as follows:

$$\mathbf{MAE} = \frac{\sum_{i=1}^{n} |\boldsymbol{y_i} - \boldsymbol{x_i}|}{\boldsymbol{n}}$$

$\mathbf{MAE}$ = mean absolute error
$\boldsymbol{y_i}$ = prediction
$\boldsymbol{x_i}$ = true value
$\boldsymbol{n}$ = total number of data points

The lower the value of MAE in the estimated unit of the predicted values are known to be better predictions [31].

4.1.5. Ranking algorithm & accuracy

The output of the cross validation layer is feed to the ranking algorithm. This ranking algorithm uses the predicted values of the test records, gets the movie ids and movie names from the movie dataset for recommending movies. In this project I will use this ranking algorithm to compare the predicted movie recommendations and actual recommendations, further calculating the accuracy. These accuracy is only a reference comparison between the actual and the predicted movie names.

ALG_RANKING (Algorithm for Ranking the predicted rating and comparing it with actual ratings)

INPUT :

- Test DataFrame (test records consisting of predicted ratings)
- User_id (to find movie recommendation and accuracy)

OUTPUT:

- Ranked predicted movie names and accuracy.

Algorithm:

Step 0: ***Helper Function*** *Algorithm (with parameter user_id)*
get records of user_id from test dataframe
get predicted recommendation dataframe subset of top 25 records sorted by Hybrid prediction ratings in descending order.
get actual recommendation dataframe subset of top 25 records sorted by actual ratings in descending order
return *both dataframes.*
Step 1: call Helper Function for a given user_id which returns predicted recommendations dataframe and actual recommendation dataframe.
Step 2: get predicted recommendation list of movie_ids and find corresponding movie names.
Step 3: get actual recommendation list of movie_ids and find corresponding movie names.
Step 4: find accuracy by dividing the intersection between the two list by the length of the list.

The above algorithm uses a function to acquire the predicted ratings (i.e.'CB2_CP_Mean_Pred') and the actual ratings (i.e. 'Rating') from the test dataset which was generated in the cross validation layer of the recommendation system architecture. Then the algorithm refers to the 'user_id' and finds top K movie_ids which were rated by the user. The corresponding movie names are obtained from the movies dataframe. The same process happens with the actual movies from the test dataset and the name list of the movies are stored separately. These movies are then displayed and compared with each other to find the accuracies between the actual and the predicted list (i.e. to find the prediction accuracy). After finding the accuracy for single user I now find the accuracy for all the users and the following algorithm shows the process :

ALG_OVERALL_RANKING (Algorithm for finding the overall accuracy)

INPUT :

- Test DataFrame (test records consisting of predicted ratings)

OUTPUT:

- Overall Accuracy.

Algorithm:

Step 0: ***Declare*** *an array for accuracies*
Step 1: ***for*** *every unique user find recommendation accuracy using the* ***ALG_RANKING algorithm.***
Step 2: append accuracy to the array.
Step 3: find mean of values in the array.

The above algorithm calculates the overall accuracy by using the ***ALG_RANKING algorithm*** for every unique user. The overall accuracy is the mean of all accuracies. This value determines the performance of this ranking algorithm.

4.2. Implementation and Technology used

For this project I have used an open-source software called as 'Jupyter Notebook' [33] which uses python language. Jupyter notebook downloaded as a sub-part of the GUI (graphic user interface) called as 'Anaconda Navigator' [34] which has multiple applications like Datalore, IBM Watson Studio Cloud, JupyterLabs, Jupyter Notebook and much more. Anaconda Navigator has some advantages is that I can setup myr own environment with various packages preinstalled in it and other can be installed in just click of a button. One of the advantages using Jupyter Notebook in Anaconda Navigator is that it used the host computer processor and ram for processing and become economical to run computational task with needs more than 6GB RAM memory. As my personal computer has 16 GB RAM memory it had made my processing much quicker.

There were other options like using Google Collab [35] and Google Cloud Platform [36], I had initially tried to run the NETFLIX dataset on Google Collab, but the kernel was freezing because of the large data set (100 million rows of data). Also to buy a higher processing memory was becoming expensive. Then I tried using the Google Cloud Platform where I understood that the NETFLIX dataset can only run on SPARK (installed on Docker extension) which would also be expensive and time consuming. Another Advantage of Jyupter Notebook running on Anaconda Navigator is that, the kernel do not time out even if the computer is idol state or hibernated, the code processing do not stop in the hibernated state of computer.

For this project I have mainly used python as the coding language and the libraries used to implement the recommendation system is 'scikit-learn' [37] and 'Tenserflow' [38] these libraries are easy to use for prediction of data which are non-classification type business problems. The scikit-learn library was majorly used to for cosine similarity, TF-IDF, test-train split and vectorizer, the Tensorflow library was majorly used for importing Keras which is used to build neural network model. The instruction of running the codes are mentioned in the 'MSc_13199041_Project_Code.ipynb'

5. Trials and Observations

5.1. Trials Preformed on Collaborative Filter

I had performed trial on both Collaborative Predictions with unweighted mean (CB_Pred) and Collaborative Predictions with weighted mean(CB2_Pred) by changing the value of 'K' from the KNN and I found the below results:

	RMSE		MAE	
	Un-Weighted	Weighted	Un-Weighted	Weighted
Values of K	CB_RMSE	CB2_MAE	CB_RMSE	CB2_MAE
20	0.7338	0.5325	0.5331	0.3781
50	0.8057	0.6325	0.6029	0.4627
100	0.8421	0.687	0.6364	0.5091
150	0.8546	0.7069	0.6487	0.5261
200	0.8631	0.718	0.655	0.5353
250	0.8681	0.7246	0.6601	0.5403

From the above table I can note that I had randomly tried to change the value of 'K' in an ascending order. Note, that the value of RMSE and MAE are gradually increasing as the number of K (i.e. number of nearest neighbours) increase. But note that, irrespective of the similarities between the two users, equal weightage is given to all the neighbours user ratings while calculating the prediction in the unweighted ratings. In weighted ratings, higher the similarity, more weightage is given to the user rating while calculating the predictions. Note, that the RMSE and MAE value is lowest when 'K = 20' so I use these values for as it is the lowest amongst in the table. I can also understand that, KNN is used to find the nearest neighbours, which means that lower the value of K, the smaller and closer the group of neighbours to each other.

5.2. Trials Preformed on Content-Based Filter

I had performed trial on Content-Based Predictions (Content_Pred) by changing the value of 'K' from the KNN and found the below results:

	RMSE	MAE
Values of K	Content_RMSE	Content_MAE
5	1.0842	0.8397
10	1.1543	0.8991
15	1.2013	0.9021
20	1.2908	0.9387

From the above table, note that I had randomly tried change the value of 'K" in ascending order and started with a lower number from 'K=5' because the computational speed required to conduct this code is high and time taken to

completely run this code is approximately 2hr 30min and as I increase the value of 'K' it goes up-to 6hrs in completely running the code block. Apart from this, note the value of RMSE and MAE is increasing as I increase the value of 'K'. Thus I selected the value of 'K = 5' as the RMSE and MAE value is the lowest amongst the rest. The time taken to run these set of codes is to find similarity of the test record movie with the other movies which will be constant for any value of K. Thus I prefer the value of K for which the RMSE is minimum.

5.3. Trials Preformed on Neural Network Model

I had performed trials on Neural Network Model in the 'Output Dimensions' of the embedding layer of the model and I found the following results:

	RMSE	MAE
Output Dimension	NN_RMSE	NN_MAE
3	0.8632	0.6572
5	0.8727	0.6591
7	0.8723	0.6584
9	0.8731	0.6599
11	0.8757	0.6701

From the above table I can note that I had randomly tried change the value of 'Output Dimensions' in ascending order and stated with that lower number 3 and found that the RMSE and MEA values dropped on 7 as compared to 5 and it is trying to maintain the RMSE and MAE values as much as possible. This means that it had reached its peak value and then trying to maintain it. Thus I considered the 'Output Dimensions' as 7. Apart from this having 3 in the output dimensions in a neural network will be very small to accommodate the sparsity of the data.

5.4. Observations in the Cross-validation Layers

The Predicted output of Un-weighted Collaborative Predictions (CB_Pred), Weighted Collaborative Predictions (CB_Pred2), Content Based Filtering (Content_Pred), Neural Network Model (NN_Pred), Mean of 'CB_Pred' and 'Content_Pred' (CB_CP_Mean), Mean of 'CB2_Pred' and 'Content_Pred' (CB2_CP_Mean) and the Mean of all Predicted outputs (Overall_Mean_Pred) and I found the following results.

Predictions	RMSE	MAE	Overall Prediction Accuracy (in %)
CB_Pred	0.7338	0.5330	89.9081
CB_Pred2	0.5325	0.3781	92.6950
CONTENT_Pred	1.0842	0.8397	84.1967
NN_Pred	0.8668	0.6598	86.9311
CB_CP_Mean_Pred	0.8411	0.6501	89.9081
CB2_CP_Mean_Pred	0.7494	0.5808	92.6885
Overall_Mean_Pred	0.7249	0.5577	90.8721

From the above table I can note that the RMSE and MAE of 'Weighted Collaborative filter' (CB_Pred2) and Mean of 'Weighted Collaborative filter' and 'Content-based filter' (CB2_CP_Mean Pred) are different. Infact the RMSE and MAE value of 'CB2_Pred' is lower than 'CB2_CP_Mean_Pred', but the Overall prediction accuracy (average accuracy across all users) is the same. This is because 'Weighted Collaborative filter' cannot handle cold start issue as well as it cannot able to find the user taste (i.e. Movie Genre), thus the mean of Content based and Collaborative based filtering (i.e. Hybrid Filtering) gives better prediction accuracy of 92.68% . The individual recommendation results of three individual users 432, 288 and 599 (selected randomly) and the output of overall prediction accuracy are given in the (Appendix 1).

6. Conclusion

After critically evaluating the various models to filter movie recommendation in the recommender system I can conclude that the hybrid model is best suited because it had both collaborative and content based filtering. The advantage of a collaborative filter is that, it expands the scope of user suggestion as it depends on the movies that the other similar users have watched and provides the user with a variation in its usual taste of movies. The advantage of content-based filtering is useful in handling the cold start problem where the user only provides with its preferences of movies and there are no similar user profiles due to new data. The hybrid filter uses both advantages of collaborative and content-based filtering where it can identify and recommend movies to a user based on other user ratings and genre of movie the user would be interested to watch.

Creating this recommender system with Hybrid filter was a challenging task and the researching about this was moreover challenging because there were many research papers written over hybrid filters in a recommender system model. Creating the model architecture for this project was tough choice to make as there are so many techniques available to predict better results. For an example, I can use Bayesian Network in the collaborative filter section and then merge it with the content-based filter (to form a hybrid) and get better prediction accuracy.

Sometimes the data can be big in size and the computation speed required to process such big data are expensive and due to which a multiple batch-jobs are required to run such data on high-end cloud servers or physical machines.

The result of this project was satisfactory and the expected out-put of this project was matching the Project proposal. Creating the collaborative and content based filter was a great learning experience, where I understood how weighted output can change the overall outcome of the process and I also experience how 'Term Frequency and Inverse Document Frequency' which are studied in theory in one of the modules of my academic year is implemented in a real-life situation. To complete this Project, the machine learning modules which I had studied in my academic year were helpful and it had helped me clarify many concepts and ideas.

The knowledge I gained by completing this project could be helpful for me in the future as there are many ad-hoc applications of the recommender algorithm for various apps, website etc. and I wish to pursue my future in this sector.

7. References

[1] R. Khanokar, "Hybridizing a Recommendation System which incorporates collaborative and content-based filtering for an online movie streaming platform. (Project Proposal)," Birkbeck, University of London, London , 2021.

[2] D. Asanov, "Algorithms and methods in recommender systems," *Tu-berlin.de*. [Online]. Available: https://www.snet.tuberlin.de/fileadmin/fg220/courses/SS11/snet-project/recommender-systems_asanov.pdf. [Accessed: 26-Sep-2021].

[3] M. Balabanović and Y. Shoham, "Fab: Content-based, collaborative recommendation," *Commun. ACM*, vol. 40, no. 3, pp. 66–72, 1997.

[4] "Netflix prize: Review rules," *Netflixprize.com*. [Online]. Available: https://www.netflixprize.com/rules.html. [Accessed: 26-Sep-2021].

[5] D. Walavalkar, "Assignment for Recommendation System | Submitted to George Mason University," 2020. [Online]. Available: https://github.com/Devikaw?tab=repositories. [Accessed: 08-Aug-2021].

[6] E. Çano and M. Morisio, "Hybrid recommender systems: A systematic literature review," *Intell. Data Anal.*, vol. 21, no. 6, pp. 1487–1524, 2017.

[7] H. Alharthi, D. Inkpen, and S. Szpakowicz, "A survey of book recommender systems," *J. Intell. Inf. Syst.*, vol. 51, no. 1, pp. 139–160, 2018.

[8] S. Maneeroj and A. Takasu, "Hybrid Recommender System Using Latent Features," in *2009 International Conference on Advanced Information Networking and Applications Workshops*, 2009.

[9] M. W. Chughtai, A. Selamat, I. Ghani, and J. J. Jung, "Retracted: E-learning recommender systems based on goal-based hybrid filtering," *Int. J. Distrib. Sens. Netw.*, vol. 10, no. 7, p. 912130, 2014.

[10] L. Hu, J. Cao, G. Xu, L. Cao, Z. Gu, and C. Zhu, “Personalized recommendation via cross-domain triadic factorization,” in *Proceedings of the 22nd international conference on World Wide Web - WWW ’13*, 2013.

[11] J. Wang, A. P. de Vries, and M. J. T. Reinders, “Unifying user-based and item-based collaborative filtering approaches by similarity fusion,” in *Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval - SIGIR ’06*, 2006.

[12] L. M. de Campos, J. M. Fernández-Luna, J. F. Huete, and M. A. Rueda-Morales, “Combining content-based and collaborative recommendations: A hybrid approach based on Bayesian networks,” *Int. J. Approx. Reason.*, vol. 51, no. 7, pp. 785–799, 2010.

[13] S. H. Choi, Y.-S. Jeong, and M. K. Jeong, “A hybrid recommendation method with reduced data for large-scale application,” *IEEE Trans. Syst. Man Cybern. C Appl. Rev.*, vol. 40, no. 5, pp. 557–566, 2010.

[14] A. Khalid, M. A. Ghazanfar, A. Azam, and S. A. Alahmari, “A scalable and practical one-pass clustering algorithm for recommender system,” in *Eighth International Conference on Machine Vision (ICMV 2015)*, 2015.

[15] Y. Kotliarova, “Analysis of methods, models and algorithms of personalization for the recommender systems development,” *ScienceRise*, vol. 12, no. 0, pp. 19–29, 2018.

[16] S. Jung, J. Kim, and J. L. Herlocker, “Applying collaborative filtering for efficient document search,” in *IEEE/WIC/ACM International Conference on Web Intelligence (WI’04)*, 2005.

[17] B. Sarwar, G. Karypis, J. Konstan, and J. Reidl, “Item-based collaborative filtering recommendation algorithms,” in *Proceedings of the tenth international conference on World Wide Web - WWW ’01*, 2001.

[18] Netflix, “Netflix Prize data,” 2019. [Online]. Available: https://www.kaggle.com/netflix-inc/netflix-prize-data. [Accessed: 27-Sep-2021].

[19] M. Garg, “Movie Recommender System Dataset,” 2021. [Online]. Available: https://www.kaggle.com/gargmanas/movierecommenderdataset. [Accessed: 27-Sep-2021].

[20] “pandas.read_csv — pandas 1.3.3 documentation,” *Pydata.org*. [Online]. Available: https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.read_csv.html. [Accessed: 27-Sep-2021].

[21] “Sklearn.Model_selection.Train_test_split,” *Scikit-learn.org*. [Online]. Available: https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.train_test_split.html. [Accessed: 27-Sep-2021].

[22] “pandas.DataFrame.pivot — pandas 1.3.3 documentation,” *Pydata.org*. [Online]. Available: https://pandas.pydata.org/docs/reference/api/pandas.DataFrame.pivot.html. [Accessed: 27-Sep-2021].

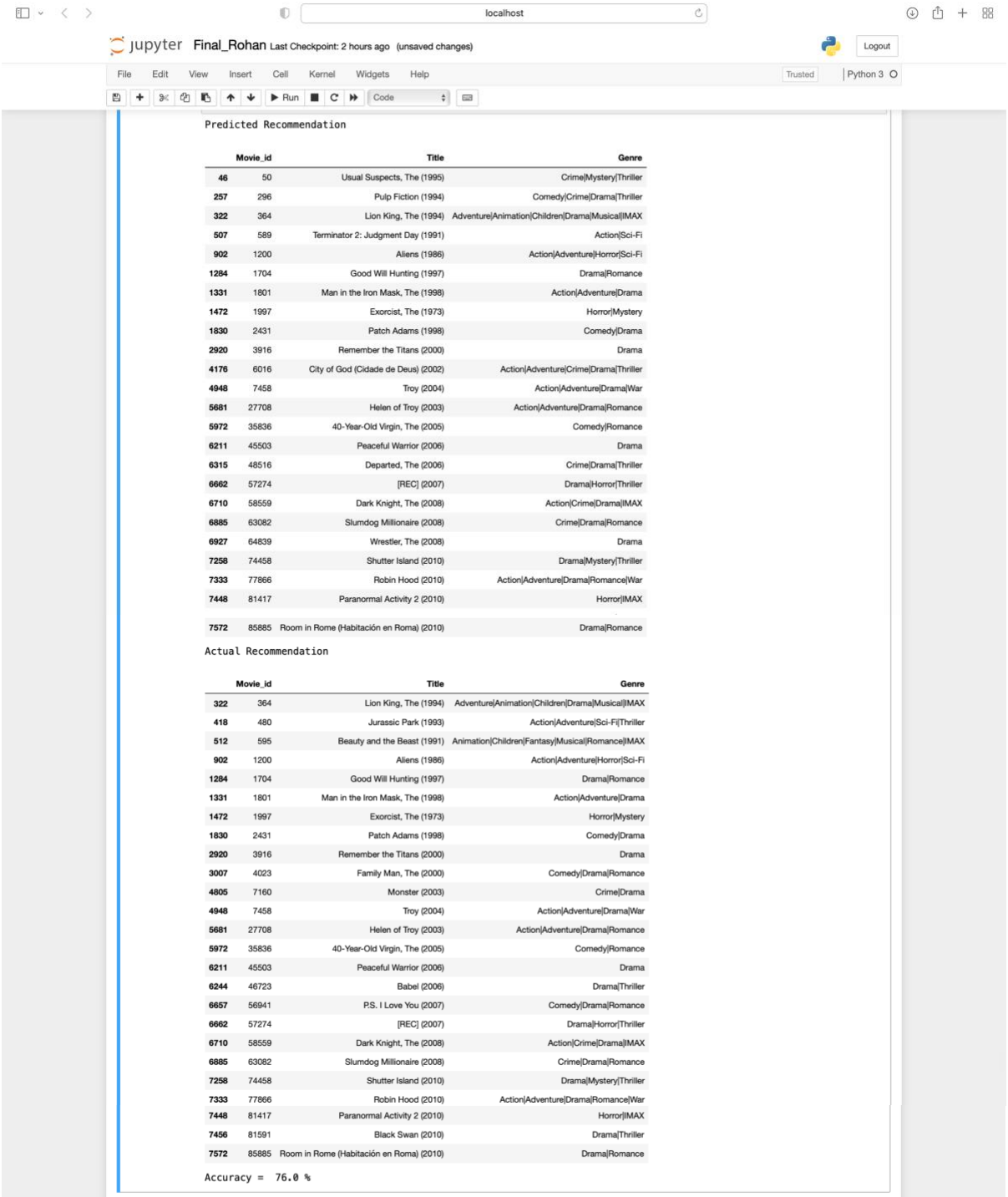
[23] “pandas.DataFrame.fillna — pandas 1.3.3 documentation,” *Pydata.org*. [Online]. Available: https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.DataFrame.fillna.html. [Accessed: 27-Sep-2021].

[24] “Sklearn.Metrics.Pairwise.Cosine_similarity,” *Scikit-learn.org*. [Online]. Available: https://scikit-

learn.org/stable/modules/generated/sklearn.metrics.pairwise.cosine_similarity.html. [Accessed: 27-Sep-2021].
[25] “pandas.DataFrame — pandas 1.3.3 documentation,” *Pydata.org*. [Online]. Available: https://pandas.pydata.org/docs/reference/api/pandas.DataFrame.html. [Accessed: 27-Sep-2021].
[26] W. Scott, “TF-IDF from scratch in python on a real-world dataset,” *Towards Data Science*, 15-Feb-2019. [Online]. Available: https://towardsdatascience.com/tf-idf-for-document-ranking-from-scratch-in-python-on-real-world-dataset-796d339a4089. [Accessed: 27-Sep-2021].
[27] Keras Team, “Keras layers API,” *Keras.io*. [Online]. Available: https://keras.io/api/layers/. [Accessed: 27-Sep-2021].
[28] Keras Team, “Optimizers,” *Keras.io*. [Online]. Available: https://keras.io/api/optimizers/. [Accessed: 27-Sep-2021].
[29] “tf.keras.losses.MeanSquaredError,” *Tensorflow.org*. [Online]. Available: https://www.tensorflow.org/api_docs/python/tf/keras/losses/MeanSquaredError. [Accessed: 27-Sep-2021].
[30] S. Mulani, “RMSE - Root Mean Square Error in Python - AskPython,” *Askpython.com*, 13-Oct-2020. [Online]. Available: https://www.askpython.com/python/examples/rmse-root-mean-square-error. [Accessed: 27-Sep-2021].
[31] E, “Mean absolute error ~ MAE [Machine Learning(ML)],” *Medium*, 02-Feb-2018. [Online]. Available: https://medium.com/@ewuramaminka/mean-absolute-error-mae-machine-learning-ml-b9b4afc63077. [Accessed: 27-Sep-2021].
[32] S. S. Iyengar and M. R. Lepper, “When choice is demotivating: can one desire too much of a good thing?,” *J. Pers. Soc. Psychol.*, vol. 79, no. 6, pp. 995–1006, 2000.
[33] “Project Jupyter,” *Jupyter.org*. [Online]. Available: https://jupyter.org. [Accessed: 29-Sep-2021].
[34] “Anaconda Navigator — Anaconda documentation,” *Anaconda.com*. [Online]. Available: https://docs.anaconda.com/anaconda/navigator/. [Accessed: 29-Sep-2021].
[35] E. Bisong, “Google Colaboratory,” in *Building Machine Learning and Deep Learning Models on Google Cloud Platform*, Berkeley, CA: Apress, 2019, pp. 59–64.
[36] “Cloud Computing Services,” *Google.com*. [Online]. Available: https://cloud.google.com. [Accessed: 29-Sep-2021].
[37] “scikit-learn: machine learning in Python — scikit-learn 0.16.1 documentation,” *Scikit-learn.org*. [Online]. Available: https://scikit-learn.org/. [Accessed: 29-Sep-2021].
[38] “Why TensorFlow,” *Tensorflow.org*. [Online]. Available: https://www.tensorflow.org/about. [Accessed: 29-Sep-2021].

8. Appendix

8.1. Individual Movie recommendation list for user_id no. 432



Predicted Recommendation

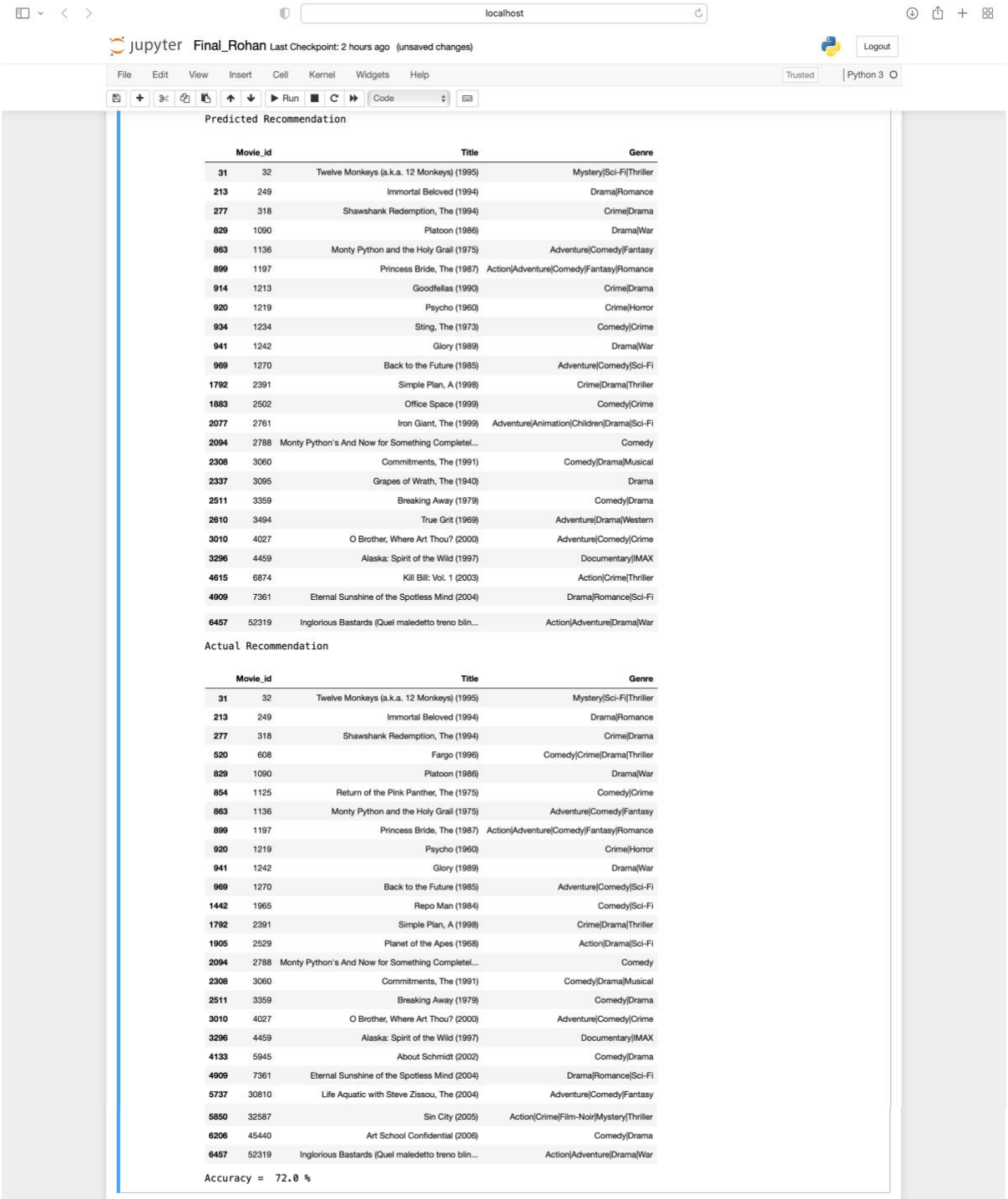
	Movie_id	Title	Genre
46	50	Usual Suspects, The (1995)	Crime|Mystery|Thriller
257	296	Pulp Fiction (1994)	Comedy|Crime|Drama|Thriller
322	364	Lion King, The (1994)	Adventure|Animation|Children|Drama|Musical|IMAX
507	589	Terminator 2: Judgment Day (1991)	Action|Sci-Fi
902	1200	Aliens (1986)	Action|Adventure|Horror|Sci-Fi
1284	1704	Good Will Hunting (1997)	Drama|Romance
1331	1801	Man in the Iron Mask, The (1998)	Action|Adventure|Drama
1472	1997	Exorcist, The (1973)	Horror|Mystery
1830	2431	Patch Adams (1998)	Comedy|Drama
2920	3916	Remember the Titans (2000)	Drama
4176	6016	City of God (Cidade de Deus) (2002)	Action|Adventure|Crime|Drama|Thriller
4948	7458	Troy (2004)	Action|Adventure|Drama|War
5681	27708	Helen of Troy (2003)	Action|Adventure|Drama|Romance
5972	35836	40-Year-Old Virgin, The (2005)	Comedy|Romance
6211	45503	Peaceful Warrior (2006)	Drama
6315	48516	Departed, The (2006)	Crime|Drama|Thriller
6662	57274	[REC] (2007)	Drama|Horror|Thriller
6710	58559	Dark Knight, The (2008)	Action|Crime|Drama|IMAX
6885	63082	Slumdog Millionaire (2008)	Crime|Drama|Romance
6927	64839	Wrestler, The (2008)	Drama
7258	74458	Shutter Island (2010)	Drama|Mystery|Thriller
7333	77866	Robin Hood (2010)	Action|Adventure|Drama|Romance|War
7448	81417	Paranormal Activity 2 (2010)	Horror|IMAX
7572	85885	Room in Rome (Habitación en Roma) (2010)	Drama|Romance

Actual Recommendation

	Movie_id	Title	Genre
322	364	Lion King, The (1994)	Adventure|Animation|Children|Drama|Musical|IMAX
418	480	Jurassic Park (1993)	Action|Adventure|Sci-Fi|Thriller
512	595	Beauty and the Beast (1991)	Animation|Children|Fantasy|Musical|Romance|IMAX
902	1200	Aliens (1986)	Action|Adventure|Horror|Sci-Fi
1284	1704	Good Will Hunting (1997)	Drama|Romance
1331	1801	Man in the Iron Mask, The (1998)	Action|Adventure|Drama
1472	1997	Exorcist, The (1973)	Horror|Mystery
1830	2431	Patch Adams (1998)	Comedy|Drama
2920	3916	Remember the Titans (2000)	Drama
3007	4023	Family Man, The (2000)	Comedy|Drama|Romance
4805	7160	Monster (2003)	Crime|Drama
4948	7458	Troy (2004)	Action|Adventure|Drama|War
5681	27708	Helen of Troy (2003)	Action|Adventure|Drama|Romance
5972	35836	40-Year-Old Virgin, The (2005)	Comedy|Romance
6211	45503	Peaceful Warrior (2006)	Drama
6244	46723	Babel (2006)	Drama|Thriller
6657	56941	P.S. I Love You (2007)	Comedy|Drama|Romance
6662	57274	[REC] (2007)	Drama|Horror|Thriller
6710	58559	Dark Knight, The (2008)	Action|Crime|Drama|IMAX
6885	63082	Slumdog Millionaire (2008)	Crime|Drama|Romance
7258	74458	Shutter Island (2010)	Drama|Mystery|Thriller
7333	77866	Robin Hood (2010)	Action|Adventure|Drama|Romance|War
7448	81417	Paranormal Activity 2 (2010)	Horror|IMAX
7456	81591	Black Swan (2010)	Drama|Thriller
7572	85885	Room in Rome (Habitación en Roma) (2010)	Drama|Romance

Accuracy = 76.0 %

8.2. Individual Movie recommendation list for user_id no. 288



Predicted Recommendation

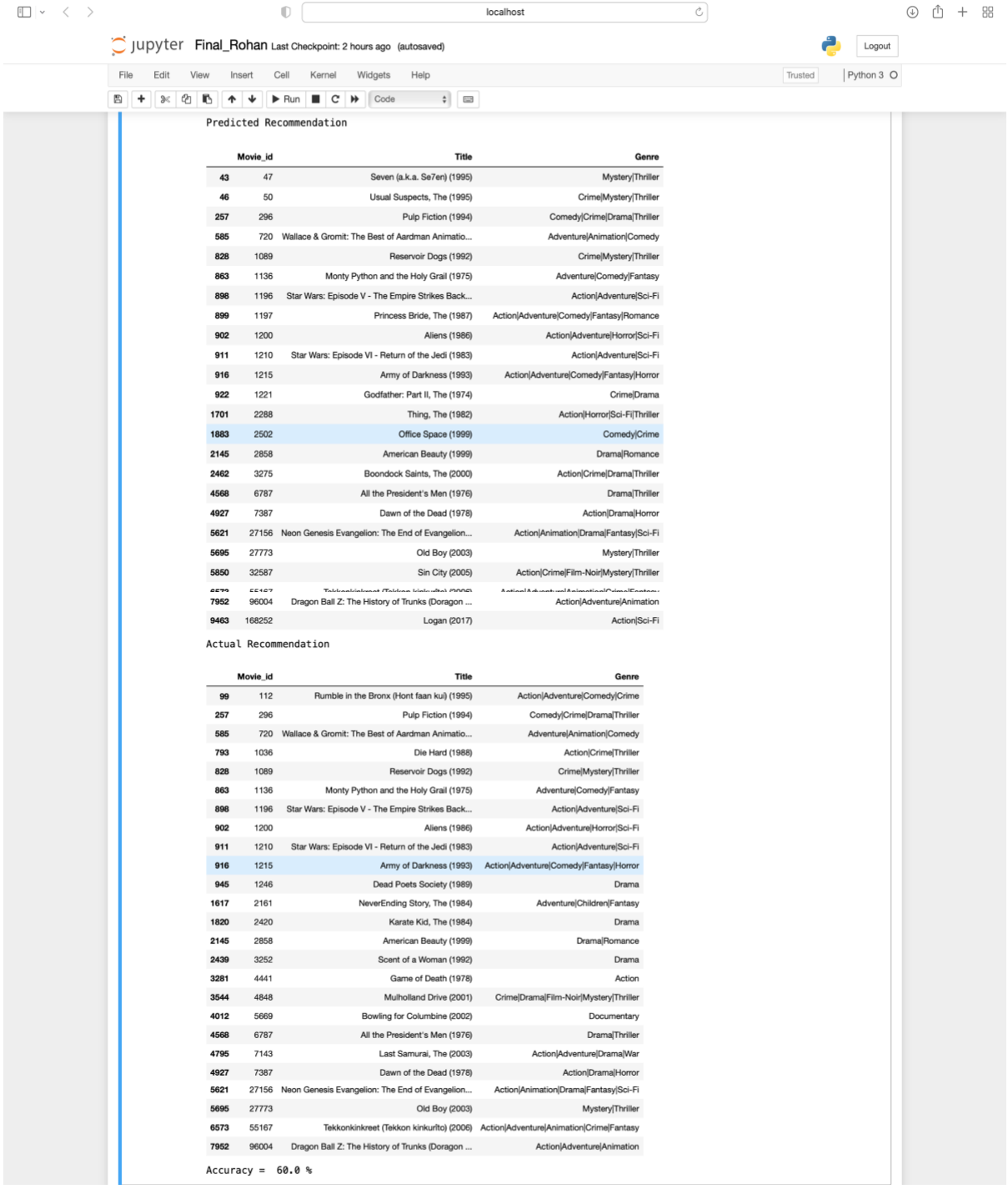
	Movie_id	Title	Genre
31	32	Twelve Monkeys (a.k.a. 12 Monkeys) (1995)	Mystery|Sci-Fi|Thriller
213	249	Immortal Beloved (1994)	Drama|Romance
277	318	Shawshank Redemption, The (1994)	Crime|Drama
829	1090	Platoon (1986)	Drama|War
863	1136	Monty Python and the Holy Grail (1975)	Adventure|Comedy|Fantasy
899	1197	Princess Bride, The (1987)	Action|Adventure|Comedy|Fantasy|Romance
914	1213	Goodfellas (1990)	Crime|Drama
920	1219	Psycho (1960)	Crime|Horror
934	1234	Sting, The (1973)	Comedy|Crime
941	1242	Glory (1989)	Drama|War
969	1270	Back to the Future (1985)	Adventure|Comedy|Sci-Fi
1792	2391	Simple Plan, A (1998)	Crime|Drama|Thriller
1883	2502	Office Space (1999)	Comedy|Crime
2077	2761	Iron Giant, The (1999)	Adventure|Animation|Children|Drama|Sci-Fi
2094	2788	Monty Python's And Now for Something Completel...	Comedy
2308	3060	Commitments, The (1991)	Comedy|Drama|Musical
2337	3095	Grapes of Wrath, The (1940)	Drama
2511	3359	Breaking Away (1979)	Comedy|Drama
2610	3494	True Grit (1969)	Adventure|Drama|Western
3010	4027	O Brother, Where Art Thou? (2000)	Adventure|Comedy|Crime
3296	4459	Alaska: Spirit of the Wild (1997)	Documentary|IMAX
4615	6874	Kill Bill: Vol. 1 (2003)	Action|Crime|Thriller
4909	7361	Eternal Sunshine of the Spotless Mind (2004)	Drama|Romance|Sci-Fi
6457	52319	Inglorious Bastards (Quel maledetto treno blin...	Action|Adventure|Drama|War

Actual Recommendation

	Movie_id	Title	Genre
31	32	Twelve Monkeys (a.k.a. 12 Monkeys) (1995)	Mystery|Sci-Fi|Thriller
213	249	Immortal Beloved (1994)	Drama|Romance
277	318	Shawshank Redemption, The (1994)	Crime|Drama
520	608	Fargo (1996)	Comedy|Crime|Drama|Thriller
829	1090	Platoon (1986)	Drama|War
854	1125	Return of the Pink Panther, The (1975)	Comedy|Crime
863	1136	Monty Python and the Holy Grail (1975)	Adventure|Comedy|Fantasy
899	1197	Princess Bride, The (1987)	Action|Adventure|Comedy|Fantasy|Romance
920	1219	Psycho (1960)	Crime|Horror
941	1242	Glory (1989)	Drama|War
969	1270	Back to the Future (1985)	Adventure|Comedy|Sci-Fi
1442	1965	Repo Man (1984)	Comedy|Sci-Fi
1792	2391	Simple Plan, A (1998)	Crime|Drama|Thriller
1905	2529	Planet of the Apes (1968)	Action|Drama|Sci-Fi
2094	2788	Monty Python's And Now for Something Completel...	Comedy
2308	3060	Commitments, The (1991)	Comedy|Drama|Musical
2511	3359	Breaking Away (1979)	Comedy|Drama
3010	4027	O Brother, Where Art Thou? (2000)	Adventure|Comedy|Crime
3296	4459	Alaska: Spirit of the Wild (1997)	Documentary|IMAX
4133	5945	About Schmidt (2002)	Comedy|Drama
4909	7361	Eternal Sunshine of the Spotless Mind (2004)	Drama|Romance|Sci-Fi
5737	30810	Life Aquatic with Steve Zissou, The (2004)	Adventure|Comedy|Fantasy
5850	32587	Sin City (2005)	Action|Crime|Film-Noir|Mystery|Thriller
6206	45440	Art School Confidential (2006)	Comedy|Drama
6457	52319	Inglorious Bastards (Quel maledetto treno blin...	Action|Adventure|Drama|War

Accuracy = 72.0 %

8.3. Individual Movie recommendation list for user_id no. 599



Predicted Recommendation

	Movie_id	Title	Genre
43	47	Seven (a.k.a. Se7en) (1995)	Mystery|Thriller
46	50	Usual Suspects, The (1995)	Crime|Mystery|Thriller
257	296	Pulp Fiction (1994)	Comedy|Crime|Drama|Thriller
585	720	Wallace & Gromit: The Best of Aardman Animatio...	Adventure|Animation|Comedy
828	1089	Reservoir Dogs (1992)	Crime|Mystery|Thriller
863	1136	Monty Python and the Holy Grail (1975)	Adventure|Comedy|Fantasy
898	1196	Star Wars: Episode V - The Empire Strikes Back...	Action|Adventure|Sci-Fi
899	1197	Princess Bride, The (1987)	Action|Adventure|Comedy|Fantasy|Romance
902	1200	Aliens (1986)	Action|Adventure|Horror|Sci-Fi
911	1210	Star Wars: Episode VI - Return of the Jedi (1983)	Action|Adventure|Sci-Fi
916	1215	Army of Darkness (1993)	Action|Adventure|Comedy|Fantasy|Horror
922	1221	Godfather: Part II, The (1974)	Crime|Drama
1701	2288	Thing, The (1982)	Action|Horror|Sci-Fi|Thriller
1883	2502	Office Space (1999)	Comedy|Crime
2145	2858	American Beauty (1999)	Drama|Romance
2462	3275	Boondock Saints, The (2000)	Action|Crime|Drama|Thriller
4568	6787	All the President's Men (1976)	Drama|Thriller
4927	7387	Dawn of the Dead (1978)	Action|Drama|Horror
5621	27156	Neon Genesis Evangelion: The End of Evangelion...	Action|Animation|Drama|Fantasy|Sci-Fi
5695	27773	Old Boy (2003)	Mystery|Thriller
5850	32587	Sin City (2005)	Action|Crime|Film-Noir|Mystery|Thriller
7952	96004	Dragon Ball Z: The History of Trunks (Doragon ...	Action|Adventure|Animation
9463	168252	Logan (2017)	Action|Sci-Fi

Actual Recommendation

	Movie_id	Title	Genre
99	112	Rumble in the Bronx (Hont faan kui) (1995)	Action|Adventure|Comedy|Crime
257	296	Pulp Fiction (1994)	Comedy|Crime|Drama|Thriller
585	720	Wallace & Gromit: The Best of Aardman Animatio...	Adventure|Animation|Comedy
793	1036	Die Hard (1988)	Action|Crime|Thriller
828	1089	Reservoir Dogs (1992)	Crime|Mystery|Thriller
863	1136	Monty Python and the Holy Grail (1975)	Adventure|Comedy|Fantasy
898	1196	Star Wars: Episode V - The Empire Strikes Back...	Action|Adventure|Sci-Fi
902	1200	Aliens (1986)	Action|Adventure|Horror|Sci-Fi
911	1210	Star Wars: Episode VI - Return of the Jedi (1983)	Action|Adventure|Sci-Fi
916	1215	Army of Darkness (1993)	Action|Adventure|Comedy|Fantasy|Horror
945	1246	Dead Poets Society (1989)	Drama
1617	2161	NeverEnding Story, The (1984)	Adventure|Children|Fantasy
1820	2420	Karate Kid, The (1984)	Drama
2145	2858	American Beauty (1999)	Drama|Romance
2439	3252	Scent of a Woman (1992)	Drama
3281	4441	Game of Death (1978)	Action
3544	4848	Mulholland Drive (2001)	Crime|Drama|Film-Noir|Mystery|Thriller
4012	5669	Bowling for Columbine (2002)	Documentary
4568	6787	All the President's Men (1976)	Drama|Thriller
4795	7143	Last Samurai, The (2003)	Action|Adventure|Drama|War
4927	7387	Dawn of the Dead (1978)	Action|Drama|Horror
5621	27156	Neon Genesis Evangelion: The End of Evangelion...	Action|Animation|Drama|Fantasy|Sci-Fi
5695	27773	Old Boy (2003)	Mystery|Thriller
6573	55167	Tekkonkinkreet (Tekkon kinkurito) (2006)	Action|Adventure|Animation|Crime|Fantasy
7952	96004	Dragon Ball Z: The History of Trunks (Doragon ...	Action|Adventure|Animation

Accuracy = 60.0 %

8.3. Output: Accuracy Per user and Overall Prediction Accuracy

	Accuracy_per_user
432	76.0
288	72.0
599	60.0
42	80.0
75	100.0
...	...
257	100.0
496	100.0
392	100.0
138	100.0
375	100.0

610 rows × 1 columns

```
Overall prediction accuracy :  92.68852459016394
```