CIND-820 Big Data Analytics Capstone Project

A study of developing a Movie Recommendation System using different Machine Learning Algorithms

Shahin Ahmed
Student # 500995601

Ryerson University

Shahin Ahmed
Student # 500995601

Table of Contents

Shahin Ahmed
Student # 500995601

CIND820 Big Data Analytics Project
Ryerson University
Fall 2020

A study of developing a Movie Recommendation System using different Machine Learning Algorithms

Abstract

COVID-19 changes the people lifestyle, due to the lockdown at home, the internet usages are increased, and it impacts on the social life. According to Forbes the initial internet hits increased by between 50% and 70%. Entertainment sites like Netflix, Facebook usages also jumped by 12% approximately.

Considering these changes, Recommendation systems are becoming particularly important in media industry as well as online consumer world. People always prefer to see the items/movies based on their previous search into the system. My project will focus on develop the recommendation system which will help individual to search for content that would be interesting to him/her based on algorithms that sort out all possible choices and create a customize lists of items for that individual.

The dataset I choose for this project is from GroupLens research lab in the University of Minnesota and available in the MovieLens website which contains four files such as movies.csv, ratings.csv, links.csv and tags.csv. The goal of this project is to give a better understanding of User Based Collaborative Filter (**UBCF**) and Item Based Collaborative Filter (**IBCF**) models for hybrid recommender systems by answering the following question:

- With what level of performance can collaborative filtering using UBCF and IBCF models produce movie recommendations based on movies and user's ratings data?

The project will be developed using the R tools.

Dataset link:

MovieLens Latest Datasets

Github Link:

Github Repositories Link

References:

https://www.forbes.com/sites/markbeech/2020/03/25/covid-19-pushes-up-internet-use-70-streaming-more-than-12-first-figures-reveal/#56235ead3104

https://www.nytimes.com/interactive/2020/04/07/technology/coronavirus-internet-use.html

Shahin Ahmed
Student # 500995601

Introduction

In today's competitive online business world, it is important to predicting what item/content a user wants. Netflix, Amazon, YouTube, and Instagram all corporate online business and service provider developed a high standard recommendation system for predicting users new and relevant content. These systems are one of the most valued assets of these companies as demonstrated by the Netflix sponsored competition with a prize of one million dollars to improve on their system [1]. These types of systems are collectively known as recommender systems [2].

Recommender systems usually generate recommendations to users, in one of the following ways:

- **Collaborative filtering algorithms** [3] predict items/products for an active user based on the past data about the other users for same items/products.

- **Content-based algorithms** [4] creates recommendations based on items/products descriptions which could be automatically extracted or manually created, or (and) from user profile which reflects user's interests on that items/products.

- **Knowledge-based algorithms** [5] predicts a recommender system based on specific queries made by the user. It might prompt the user to give a series of rules or guidelines on what the results should look like, or an example of an item. The system then searches through its database of items and returns similar results.

- **Hybrid approaches** [6] generate recommendations by combining several algorithms or recommendation components, which are based on the above three approaches: collaborative filtering and content-based and knowledge-based algorithms.

Table 1 shows some popular sites which are currently using recommendation system for different purpose [7].

Table 1: Popular sites using recommender systems

Site	What is recommended
Amazon	Books/other products
Facebook	Friends/Business/Media
Netflix	Movies
Instagram	Media content

Literature Review

In my project I am going to develop a hybrid recommendation system based on User Based Collaborative Filter (UBCF) and Item Based Collaborative Filter (IBCF). Analysis of Recommendation System is a vast topic, so I studied other's research work closely related to my project work.

In introduction, we classified recommendation system into four types which include memory-based CF, model-based CF, and Hybrid CF ([8],[9],[10]). The memory-based CF techniques explore the entire dataset to find the set of users, which are like the active user ([11],[12]). Thereafter recommendations are made based on the observation of likes and dislikes of the similar users. For similarity computation process, memory-based technique employs various similarity measures which includes PR measure ([13]), Spearman rank correlation (similar to Pearson except rating is rank) ([11]), Kendall's correlation (similar to Spearman rank but instead of rank, relative ranks are used) ([14]). User based CF and Item based CF are classified as Memory-based CF. In users-based CF method ([15],[16]), similar users are found by analyzing other users' preferences against the active user. Other way item-based CF ([17],[18]) approach focuses on finding similarity between items rather than users.

Various machine learning techniques, data mining algorithm are used to develop a user rating model in model-based CF for making predictions. Bayesian model ([19],[20]), clustering models ([21]), rule-based approach ([22]) are some popular algorithms which are used for model-based CF techniques.

The hybrid CF recommendation technique solved the issues find in memory-based and model-based techniques such as cold start, scalability, sparsity and many more. The real-life examples of the recommendation system include Netflix, Amazon, Google, Instagram and many more.

CF is most efficient and widely used recommendation system, still it has some certain limitations, like sparsity, scalability, cold start ([23],[8],[9]). Normally, a user rates only certain limited items and it becomes difficult to find similar users due to sparsity in dataset ([24], [25],[26]). Also, when a new user or new item is added to the system for the first time, the sparsity grows in the dataset. It impacts the quality of recommendations and introduces a problem known as cold-start ([26]). Besides, over time rating grows in millions and computation becomes slow which introduces a serious scalability issue.

To solve the problems of scalability and sparsity in the collaborative filtering, an approach is given in [27] in which personalized recommendation methods joins the user cluster and item cluster. To improve the prediction quality of item-based collaborative filtering, some algorithms take the attributes of items into consideration while predicting the preference of a user ([28]).

There is an attempt to cope with Item cold start using a hybrid method which first clusters items using the rating matrix and then uses the clustering results to build a decision tree to combine novel items with existing ones [29].

Shahin Ahmed
Student # 500995601

Dataset

The dataset used was from MovieLens, and is publicly available at [MovieLens Latest Datasets.](#) In order to keep the recommender simple, I used the smallest dataset available (ml-latest-small.zip), which at the time of download contained 105339 ratings and 6138 tag applications across 10329 movies. These data were created by 668 users between April 03, 1996 and January09, 2016. This dataset was generated on January 11, 2016.(http://grouplens.org/datasets/movielens/latest)

The data are contained in four files: links.csv, movies.csv, ratings.csv and tags.csv. I am going to use the files movies.csv and ratings.csv to build a recommendation system.

glimpse(movies)
Rows: 9,742
Columns: 3
$ movieId <int> 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16,...
$ title <fct> "Toy Story (1995)", "Jumanji (1995)", "Grumpier Old Me...
$ genres <fct> Adventure|Animation|Children|Comedy|Fantasy, Adventure...

summary(movies)

movieId	title	genres
Min. : 1 1st Qu. : 3248 Median : 7300 Mean : 42200 3rd Qu. : 76232 Max. : 193609	Confessions of a Dangerous Mind (2002): 2 Emma (1996) : 2 Eros (2004) : 2 Saturn 3 (1980) : 2 War of the Worlds (2005) : 2 '71 (2014) : 1 (Other) : 9731	Drama : 1053 Comedy : 946 Comedy|Drama : 435 Comedy|Romance: 363 Drama|Romance : 349 Documentary : 339 (Other) :6257

From this initial exploration, we discover that movies have 9,742 observations and 3 attributes:

movieid : integer, Unique ID for the movie
title: factor, movie title (not unique)
genres: factor, genres associated with the movie

glimpse(ratings)
Rows: 100,836
Columns: 4

$ userId <int> 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1...
$ movieId <int> 1, 3, 6, 47, 50, 70, 101, 110, 151, 157, 163, 216, 2...
$ rating <dbl> 4, 4, 4, 5, 5, 3, 5, 4, 5, 5, 5, 5, 3, 5, 4, 5, 3, 3...
$ timestamp <int> 964982703, 964981247, 964982224, 964983815, 96498293...

summary(ratings)

userId	movieId	rating	timestamp
Min. : 1.0	Min. : 1	Min. : 0.500	Min. : 8.281e+08
1st Qu. : 177.0	1st Qu. : 1199	1st Qu. : 3.000	1st Qu. : 1.019e+09
Median : 325.0	Median : 2991	Median : 3.500	Median : 1.186e+09
Mean : 326.1	Mean : 19435	Mean : 3.502	Mean : 1.206e+09
3rd Qu. : 477.0	3rd Qu. : 8122	3rd Qu.: 4.000	3rd Qu. : 1.436e+09
Max. : 610.0	Max. : 193609	Max. : 5.000	Max. : 1.538e+09

From this initial exploration, we discover that ratings have 100,836 observations and 4 attributes:

userid: integer, Unique ID for the user
movieid : integer, Unique ID for the movie
rating: double, a rating between 0 and 5 for the movie
timestamp: discrete, Date and time the rating was given

glimpse(tags)
Rows: 3,683
Columns: 4
$ userId <int> 2, 2, 2, 2, 2, 2, 2, 2, 2, 7, 18, 18, 18, 18, 18, 18...
$ movieId <int> 60756, 60756, 60756, 89774, 89774, 89774, 106782, 10...
$ tag <fct> funny, Highly quotable, will ferrell, Boxing story, ...
$ timestamp <int> 1445714994, 1445714996, 1445714992, 1445715207, 1445...

userId	movieId	tag	timestamp
Min. : 2.0	Min. : 1	In Netflix queue : 131	Min. : 1.137e+09

1st Qu. : 424.0	1st Qu. : 1262	atmospheric : 36	1st Qu. : 1.138e+09
Median : 474.0	Median : 4454	superhero : 24	Median : 1.270e+09
Mean : 431.1	Mean : 27252	thought-provoking : 24	Mean : 1.320e+09
3rd Qu. : 477.0	3rd Qu. : 39263	Disney : 23	3rd Qu. : 1.498e+09
Max. : 610.0	Max. : 193565	funny : 23	Max. : 1.537e+09
		(Other) : 3422	

From this initial exploration, we discover that tags have 3,683 observations and 4 attributes:

userid: integer, Unique ID for the user
movieid : integer, Unique ID for the movie
tag: factor, Tag associated with the movie
timestamp: intger, Date and time the rating was given

glimpse(links)
Rows: 9,742
Columns: 3
$ movieId <int> 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16,...
$ imdbId <int> 114709, 113497, 113228, 114885, 113041, 113277, 114319...
$ tmdbId <int> 862, 8844, 15602, 31357, 11862, 949, 11860, 45325, 909...

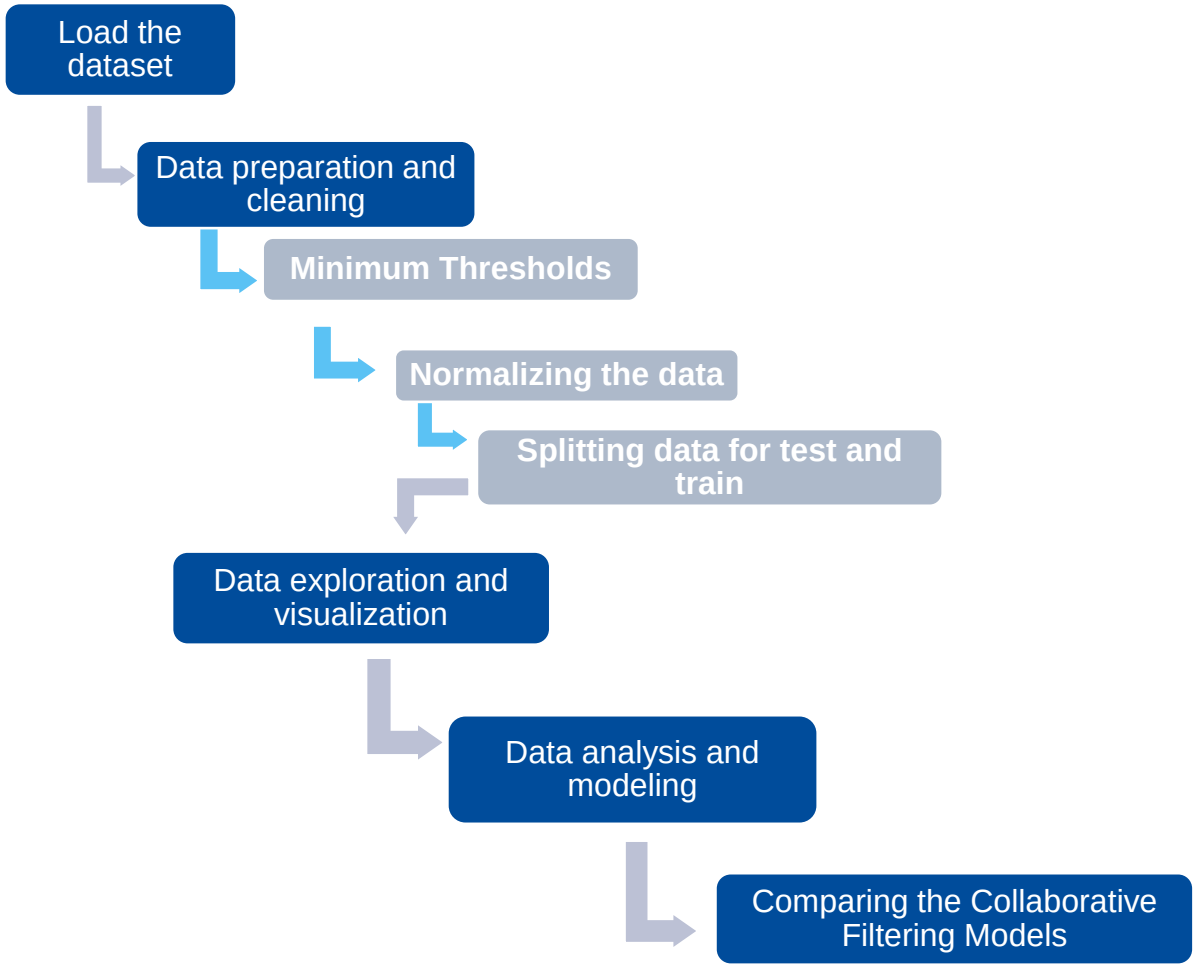
movieId	Imdbid	tmdbid
Min. : 1	Min. : 417	Min. : 2
1st Qu. : 3248	1st Qu. : 95181	1st Qu. : 9666
Median : 7300	Median : 167260	Median : 16529
Mean : 42200	Mean : 677184	Mean : 55162
3rd Qu. : 76232	3rd Qu. : 805568	3rd Qu. : 44206
Max. : 193609	Max. : 8391976	Max. : 525662
		Na's : 8

From this initial exploration, we discover that links have 9,742 observations and 3 attributes:

movieid : integer, Unique ID for the movie
imdbid: integer, IMDB ID for particular movie
tmdbid: integer, TMDB ID for particular movie

Approach

The workflow to build a Movie Recommendation System using the UBCF and IBCF models can be summarized graphically as follows:



Shahin Ahmed
Student # 500995601

Step 1: Load the dataset

The dataset is split into four files- movies.csv, ratings.csv, links.csv and tags.csv. We will iteratively load the files into the workspace reading directly from my GitHub Repository for this project using url and read_csv() function. I load movies.csv and ratings.csv for this projects and explore both data files.

Step 2: Data Preparation and cleaning

In this section we will take the first look at the loaded data frames. We will also perform necessary cleaning and some transformations so that the data better suits our needs.

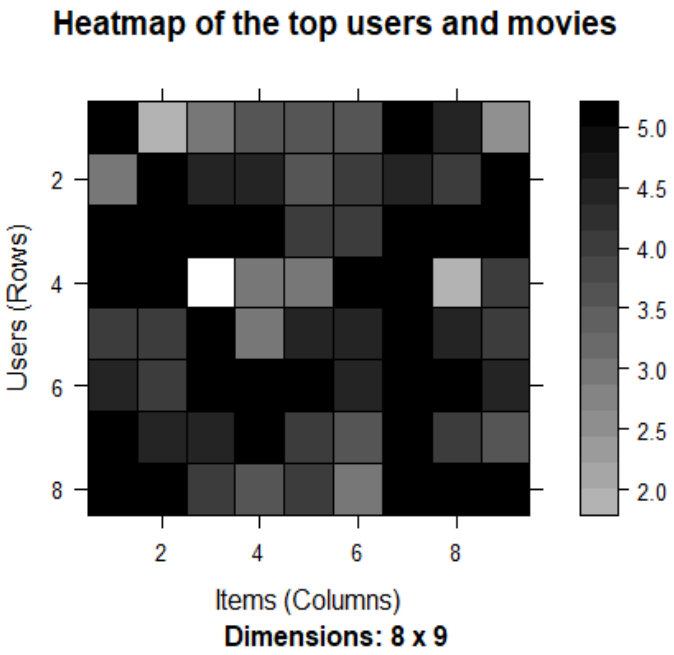
2.1 Minimum Thresholds:

For building a collaborative filtering model we can limit the input data based on minimum thresholds: for example, we may ignore users that have provided too few ratings, and also ignore those movies that have received too few ratings from users.

In order to predict the most relevant data, rating matrix is defined with the minimum number of users per rated movie as 50 and the minimum views number per movie as 50.

Such a selection of the most relevant data contains 378 users and 436 movies, compared to previous 610 users and 9742 movies in the total dataset.

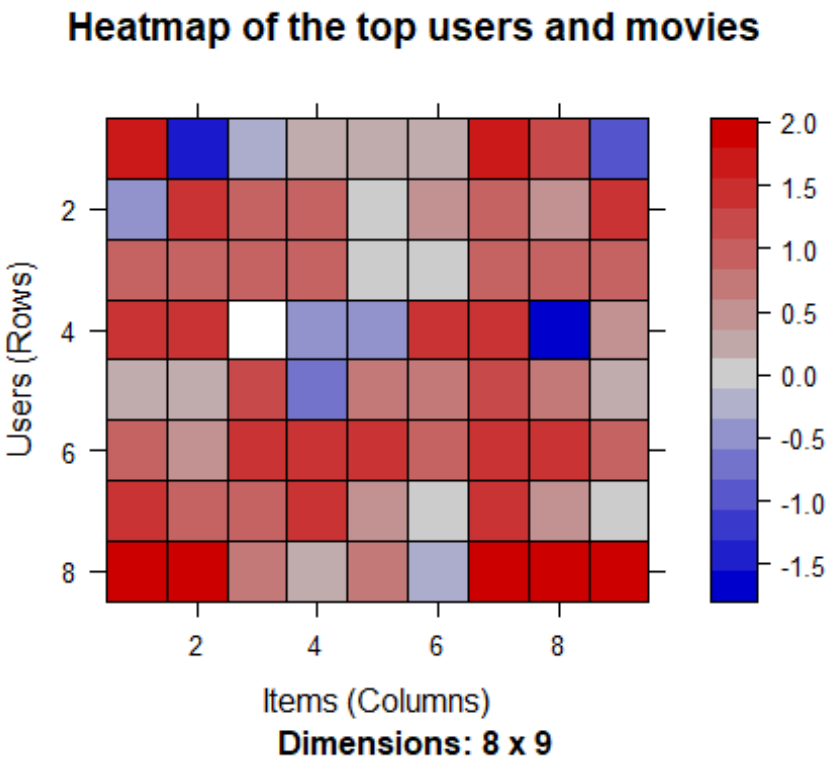
Using the same approach as previously, I visualize the top 2 percent of users and movies in the new matrix of the most relevant data:



2.2 Normalizing the data:

Having users who give high (or low) ratings to all their movies might bias the results. In order to remove this effect, I normalize the data in such a way that the average rating of each user is 0. As a quick check, I calculate the average rating by users, and it is equal to 0, as expected.

Visualize the normalized matrix for the top movies



2.3 Splitting data for test and train:

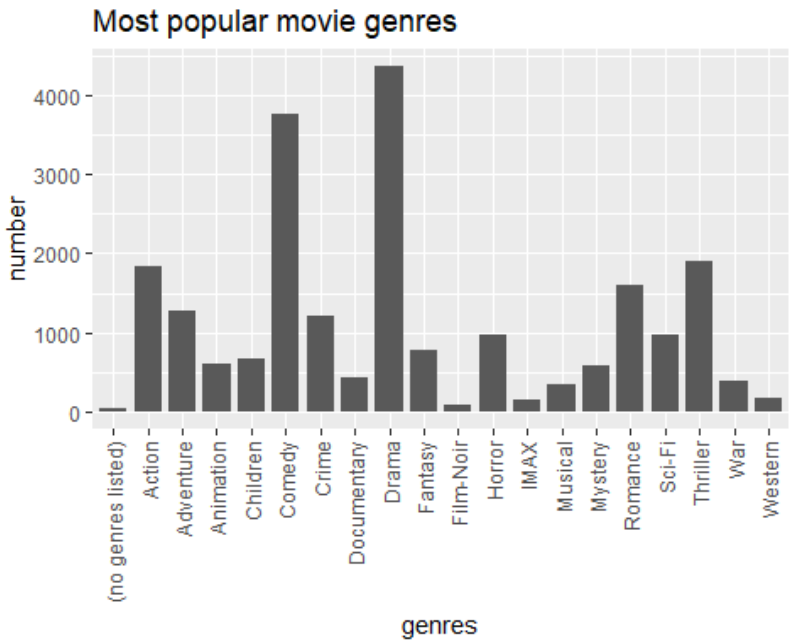
In this step I'm going to split the data into test and train sets, so that we could test the models on test data and later could compare different data models. I build the model using 80% of the whole dataset as a training set, and 20% - as a test set.

Shahin Ahmed
Student # 500995601

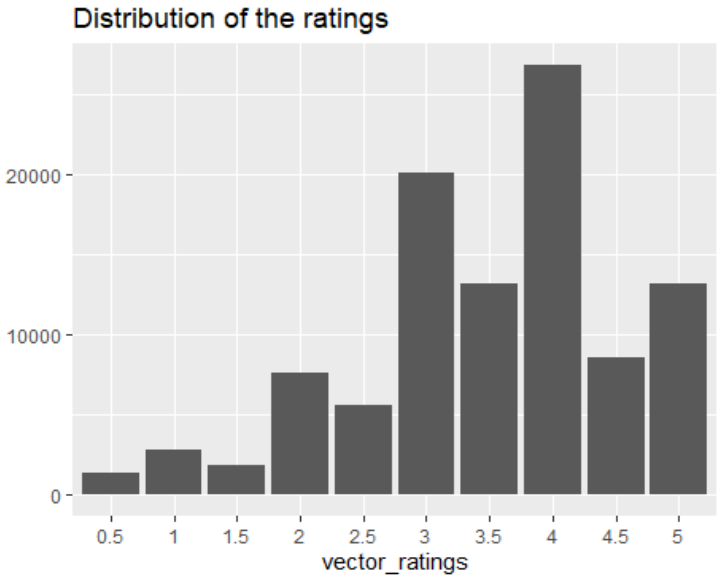
Step 3: Data exploration and visualization

Before start building the model, we need to understand the structure of the data, the distribution of ratings and the movie data. This information will help build a better model.

The number of genres in each movie is listed in this table and hence graph plotted the most popular movie genres

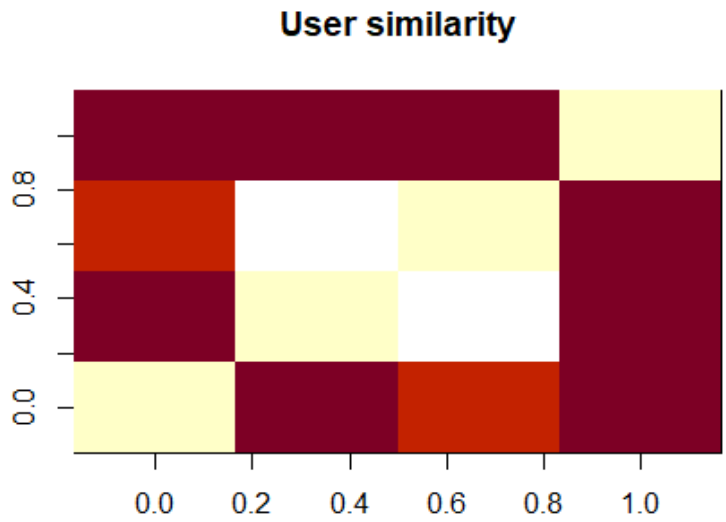


Visualization of rating count

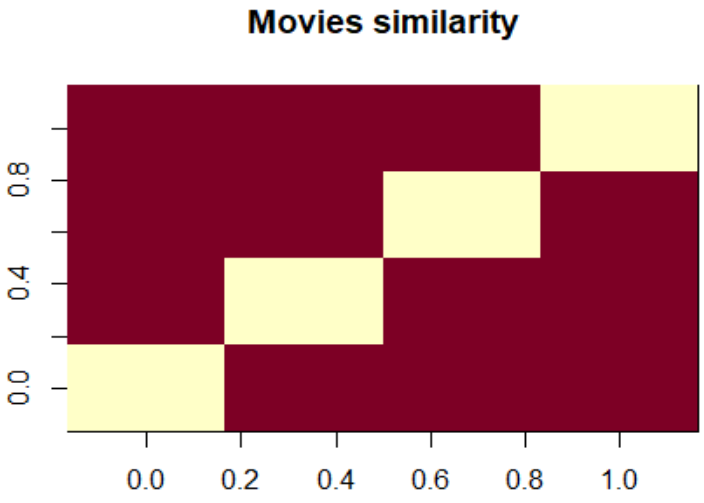


Shahin Ahmed
Student # 500995601

Exploring Similarity Data



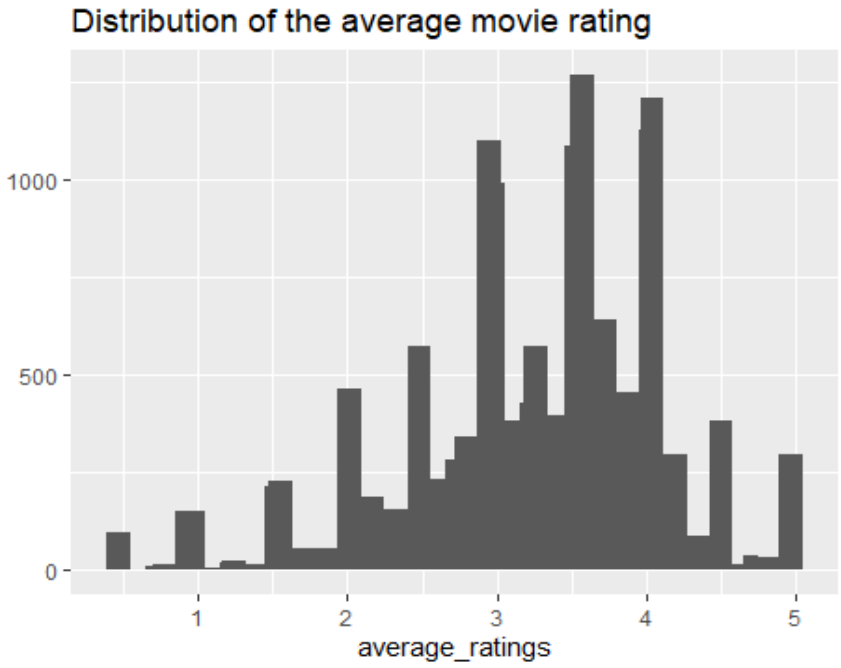
In the given matrix, each row and each column corresponds to a user, and each cell corresponds to the similarity between two users. The more red the cell is, the more similar two users are. Note that the diagonal is red, since it's comparing each user with itself.



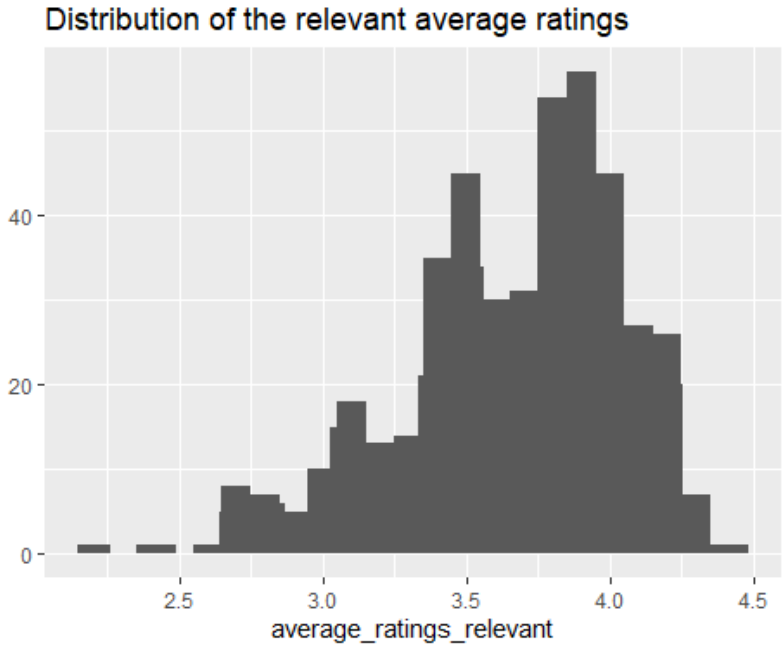
Using the same approach, I compute similarity between the first four movies.

Shahin Ahmed
Student # 500995601

Distribution of the average movie rating



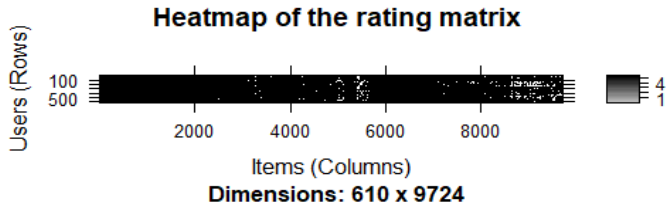
Distribution of the relevant average rating

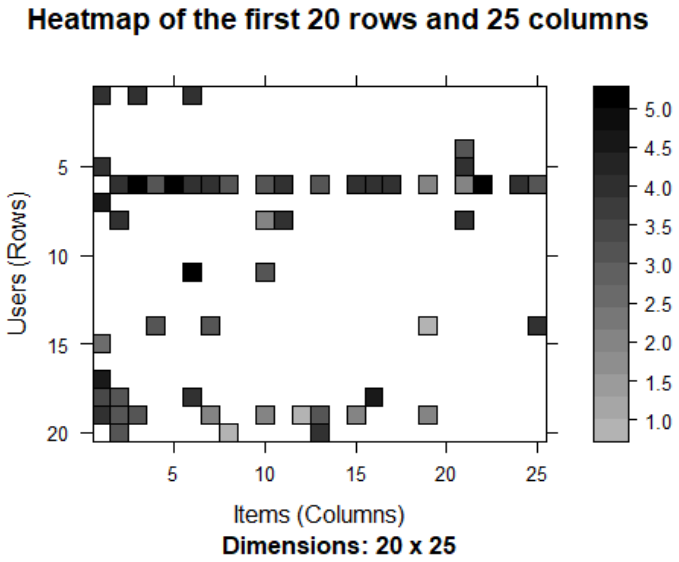


The first image above shows the distribution of the average movie rating. The highest value is around 3, and there are a few movies whose rating is either 1 or 5. Probably, the reason is that these movies received a rating from a few people only, so we shouldn't take them into account.

I remove the movies whose number of views is below a defined threshold of 50, creating a subset of only relevant movies. The second image above shows the distribution of the relevant average ratings. All the rankings are between 2.16 and 4.45. As expected, the extremes were removed. The highest value changes, and now it is around 4.

Heatmap of the rating matrix





Since there are too many users and items, the first chart is hard to read. The second chart is built zooming in on the first rows and columns.

Shahin Ahmed
Student # 500995601

Step 4: Data analysis and modeling

In this part I am going to develop and analysis UBCF and IBCF models.

UBCF Model:

Now, I will use the user-based approach. According to this approach, given a new user, its similar users are first identified. Then, the top-rated items rated by similar users are recommended.

For each new user, these are the steps:

1. Measure how similar each user is to the new one, popular similarity measures are correlation and cosine.
2. Identify the most similar users. The options are:
    - Take account of the top k users (k-nearest_neighbors)
    - Take account of the users whose similarity is above a defined threshold
3. Rate the movies rated by the most similar users. The rating is the average rating among similar users and the approaches are:
    - Average rating
    - Weighted average rating, using the similarities as weights
4. Pick the top-rated movies.

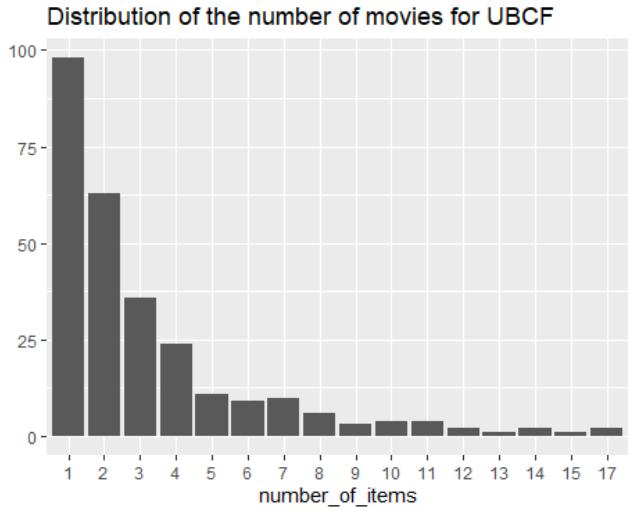
Building the recommendation system:

Again, let's first check the default parameters of UBCF model. Here, nn is a number of similar users, and method is a similarity function, which is cosine by default. I build a recommender model leaving the parameters to their defaults and using the training set.

```
##      [,1]

## [1,] "Dances with Wolves (1990)"

## [2,] "Ulee's Gold (1997)"

## [3,] "Good, the Bad and the Ugly, The (Buono, il brutto, il cattivo, Il) (
1966)"
## [4,] "Cruel Intentions (1999)"

## [5,] "Cemetery Man (Dellamorte Dellamore) (1994)"
```

Applying the recommender model on the test set

Recommendations as 'topNList' with n = 10 for 74 users.



The distribution has a longer tail. This means that there are some movies that are recommended much more often than the others.

Let's take a look at the top titles:

```
##                    Movie title No of items
## 1234         Sting, The (1973)          17
## 1617 L.A. Confidential (1997)          17
## 3897      Almost Famous (2000)          15
## 1394   Raising Arizona (1987)          14
```

IBCF Model

Collaborative filtering is a branch of recommendation that takes account of the information about different users. The word "collaborative" refers to the fact that users collaborate with each other to recommend items. In fact, the algorithms take account of user ratings and preferences.

The starting point is a rating matrix in which rows correspond to users and columns correspond to items. The core algorithm is based on these steps:

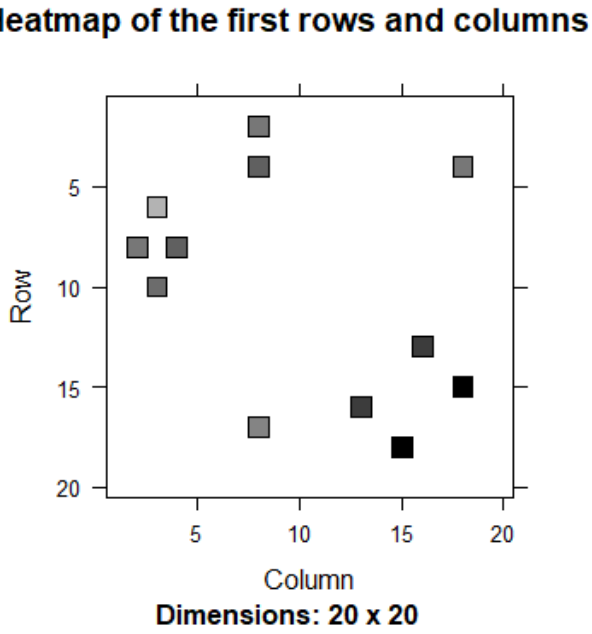
1. For each two items, measure how similar they are in terms of having received similar ratings by similar users
2. For each item, identify the k most similar items
3. For each user, identify the items that are most similar to the user's purchases

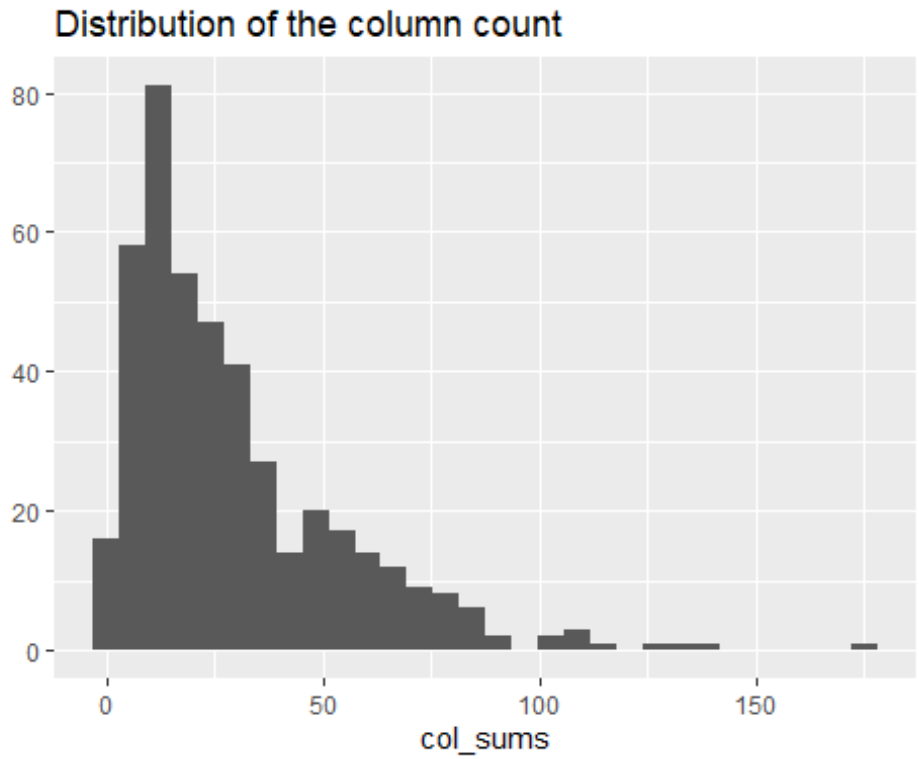
Shahin Ahmed
Student # 500995601

Building the recommendation system:

Let's have a look at the default parameters of IBCF model. Here, k is the number of items to compute the similarities among them in the first step. After, for each item, the algorithm identifies its k most similar items and stores the number. method is a similarity function, which is Cosine by default, may also be pearson. I create the model using the default parameters of method = Cosine and k=30.

Exploring the recommender model:





dgCMatrix is a similarity matrix created by the model. Its dimensions are 436 x 436, which is equal to the number of items. The heatmap of 20 first items show that many values are equal to 0. The reason is that each row contains only k (30) elements that are greater than 0. The number of non-null elements for each column depends on how many times the corresponding movie was included in the top k of another movie. Thus, the matrix is not necessarily symmetric, which is also the case in our model.

The chart of the distribution of the number of elements by column shows there are a few movies that are similar to many others.

Applying recommender system on the dataset:

Now, it is possible to recommend movies to the users in the test set. I define n_recommended equal to 10 that specifies the number of movies to recommend to each user.

For each user, the algorithm extracts its rated movies. For each movie, it identifies all its similar items, starting from the similarity matrix. Then, the algorithm ranks each similar item in this way:

- Extract the user rating of each purchase associated with this item. The rating is used as a weight.
- Extract the similarity of the item with each purchase associated with this item.
- Multiply each weight with the related similarity.
- Sum everything up.

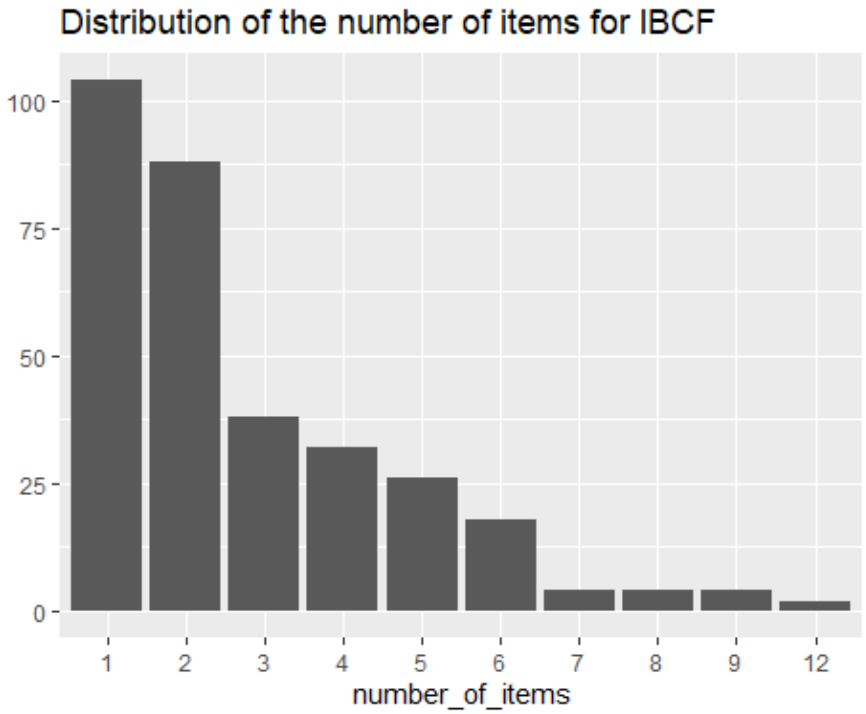
Then, the algorithm identifies the top 10 recommendations:

```
##  [1] "Quiz Show (1994)"
##  [2] "Beverly Hills Cop III (1994)"
##  [3] "True Romance (1993)"
##  [4] "North by Northwest (1959)"
##  [5] "Butch Cassidy and the Sundance Kid (1969)"
##  [6] "Pleasantville (1998)"
##  [7] "Judge Dredd (1995)"
##  [8] "Home Alone (1990)"
##  [9] "RoboCop (1987)"
## [10] "Big (1988)"
```

.I visualize the recommendations for the first four users:

```
[,1] [,2]  [,3] [,4]
##  [1,]  300 4027 71535   50
##  [2,]  420 2080 56367  480
##  [3,]  555 6378  4308  527
##  [4,]  908 3977  3717  780
##  [5,] 1304   32 56174  904
##  [6,] 2321  500  3033 1089
##  [7,]  173 1625  1203 1200
##  [8,]  586 1356  6016 1208
##  [9,] 2985 3623 68157 1222
## [10,] 2797 1517  1307 1265
```

The following image shows the distribution of the number of items for IBCF:



Comparing results of UBCF and IBCF:

Comparing the results of UBCF with IBCF helps find some useful insight on different algorithms. UBCF needs to access the initial data. Since it needs to keep the entire database in memory, it doesn't work well in the presence of a big rating matrix. Also, building the similarity matrix requires a lot of computing power and time.

However, UBCF's accuracy is proven to be slightly more accurate than IBCF, so it's a good option if the dataset is not too big.

Step 5: Comparing the Collaborative Filtering Models

In this section I am going to evaluate the models created in previous part using different similarity parameters.

Evaluating the Different Recommender Systems

There are a few options to choose from when deciding to create a recommendation engine. In order to compare their performances and choose the most appropriate model, I follow these steps:

- Prepare the data to evaluate performance
- Evaluate the performance of some models
- Choose the best performing models
- Optimize model parameters

Using cross-validation to validate models

The k-fold cross-validation approach is the most accurate one, although it's computationally heavier.

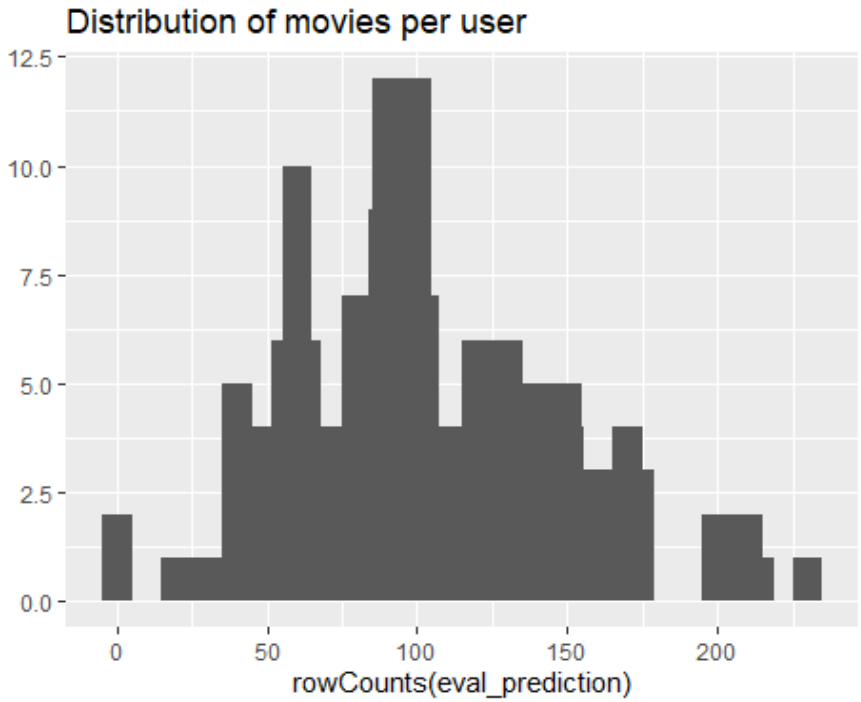
Using this approach, we split the data into some chunks, take a chunk out as the test set, and evaluate the accuracy. Then, we can do the same with each other chunk and compute the average accuracy.

```
## [1] 282 282 282 282
```

Using 4-fold approach, we get four sets of the same size 282

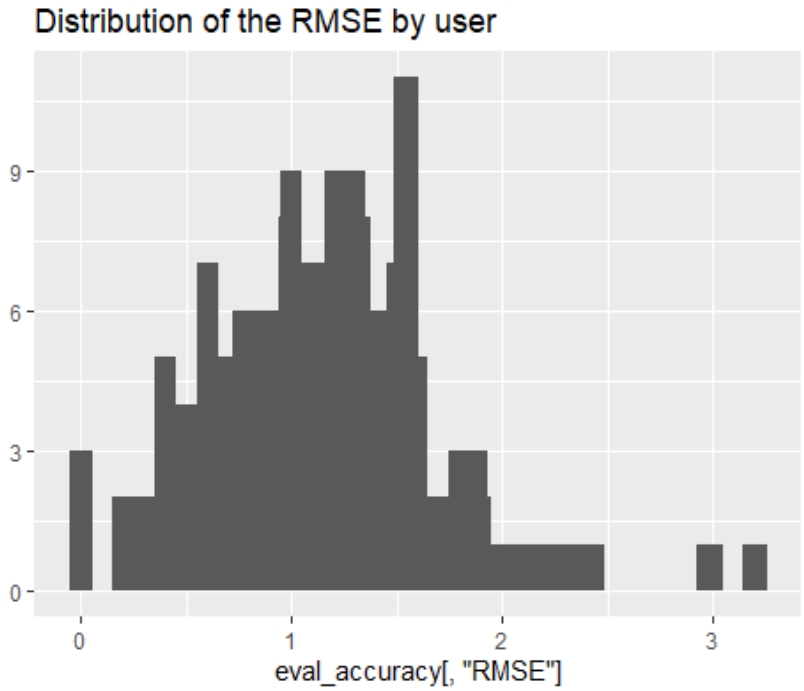
Evaluating the ratings

First, I re-define the evaluation sets, build IBCF model and create a matrix with predicted ratings.



The above image displays the distribution of movies per user in the matrix of predicted ratings.

Now, I compute the accuracy measures for each user. Most of the RMSE's (Root mean square errors) are in the range of 0.8 to 2.4:



Acuracy Matrix IBCF

In order to have a performance index for the whole model, I specify byUser as FALSE and compute the average indices:

```
##     RMSE      MSE      MAE
## 1.340759 1.797633 1.007170
```

Acuracy Matrix IBCF

In order to have a performance index for the whole model, I specify byUser as FALSE and compute the average indices:

```
##     RMSE      MSE      MAE
## 1.105401 1.221911 0.855061
```

The measures of accuracy are useful to compare the performance of different models on the same data.

Model	RMSE
IBCF	1.340759
UBCF	1.105401

From the above table we see UBCF has less RMSE value compare to IBCF which means UBCF model accuracy is better than IBCF model accuracy.

Evaluating the recommendations

Another way to measure accuracy is by comparing the recommendations with the purchases having a positive rating. For this, I can make use of a prebuilt evaluate function in recommenderlab library. The function evaluates the recommender performance depending on the number n of items to recommend to each user. I use n as a sequence n = seq(10, 100, 10).

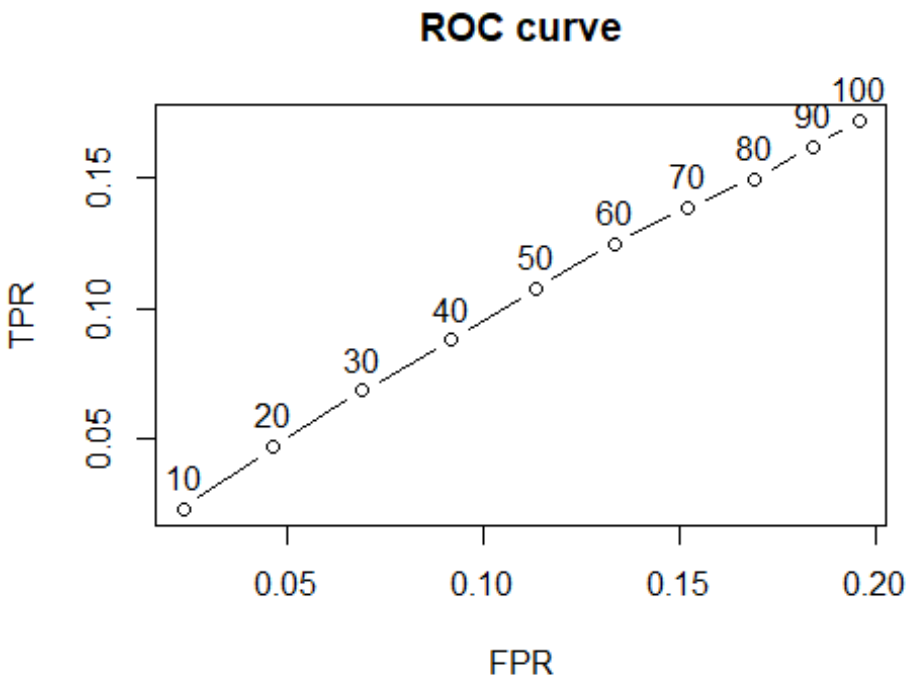
IBCF Model

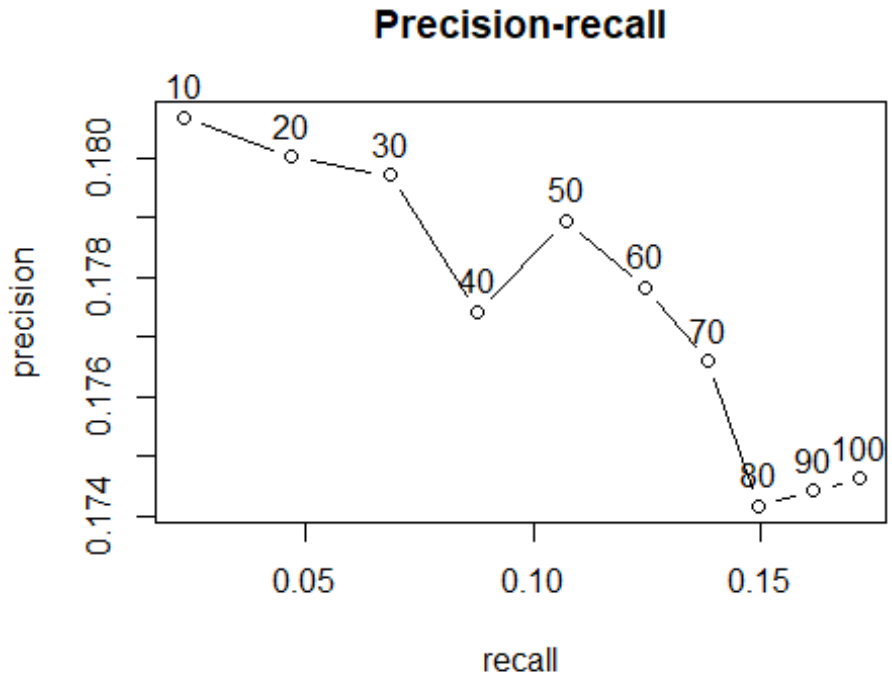
I sum up the indices of columns TP, FP, FN and TN from The first rows of the resulting performance matrix

```
##          TP        FP       FN       TN
## 10  7.09375  32.17708 318.1250 1366.604
## 20 14.10417  64.20833 311.1146 1334.573
## 30 21.02083  95.97917 304.1979 1302.802
## 40 27.46875 127.43750 297.7500 1271.344
## 50 34.21875 157.11458 291.0000 1241.667
## 60 39.98958 185.20833 285.2292 1213.573
```

Shahin Ahmed
Student # 500995601

ROC and the precision/recall curves for IBCF model:





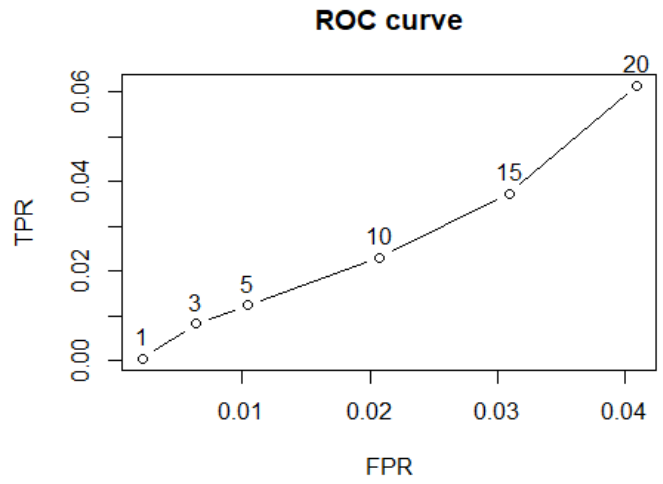
Shahin Ahmed
Student # 500995601

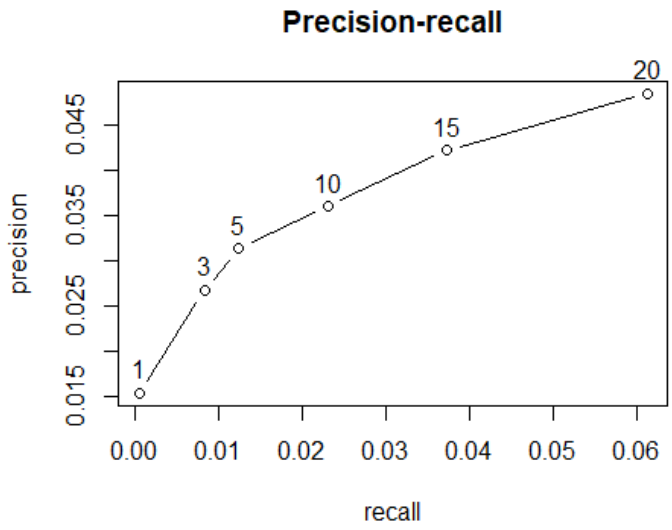
UBCF Model

I sum up the indices of columns TP, FP, FN and TN from The first rows of the resulting performance matrix

```
##             TP        FP       FN       TN
## 1  0.06896552  4.431034 77.32759 2083.172
## 3  0.36206897 13.137931 77.03448 2074.466
## 5  0.70689655 21.793103 76.68966 2065.810
## 10 1.62068966 43.379310 75.77586 2044.224
## 15 2.84482759 64.603448 74.55172 2023.000
## 20 4.34482759 85.517241 73.05172 2002.086
```

ROC and the precision/recall curves for UBCF model:





Shahin Ahmed
Student # 500995601

Comparing different models

We evaluate UBCF and IBCD model, but multiple models can be evaluated and their performance compared. AUC is our benchmark statistic. In order to compare different models, I define them as a following list:

- Item-based collaborative filtering, using the Cosine as the distance function
- Item-based collaborative filtering, using the Pearson correlation as the distance function
- User-based collaborative filtering, using the Cosine as the distance function
- User-based collaborative filtering, using the Pearson correlation as the distance function
- SVD or Matrix Factorization
- Random recommendations to have a base line
-

Confusion matrix, UBCF pearson and SVD similarity

We can explore the performance evaulation, for example, UBCF with pearson distance and SVD with k=100. The true positive ratio increases with the number of recommendations. Note that TPR and Recall are the same.

The following table presents as an example the first rows of the performance evaluation matrix for the UBCF with Pearson distance and SVD with k=100:

UBCF_cor

	precision	recall	TPR	FPR
1	0.1184211	0.000894752	0.000894752	0.002634195
5	0.1842105	0.009687070	0.009687070	0.012236147
10	0.2065789	0.022207315	0.022207315	0.023522541
20	0.2361842	0.050872275	0.050872275	0.044740662
30	0.2425439	0.080498785	0.080498785	0.066709649
40	0.2562500	0.114402972	0.114402972	0.086771719

SVD

	precision	recall	TPR	FPR
1	0.4473684	0.005264316	0.005264316	0.001543526
5	0.3526316	0.020506890	0.020506890	0.009277380
10	0.3092105	0.036358817	0.036358817	0.020157722
20	0.2848684	0.066080358	0.066080358	0.041988253
30	0.2776316	0.096461529	0.096461529	0.063633951
40	0.2690789	0.123931099	0.123931099	0.086062463

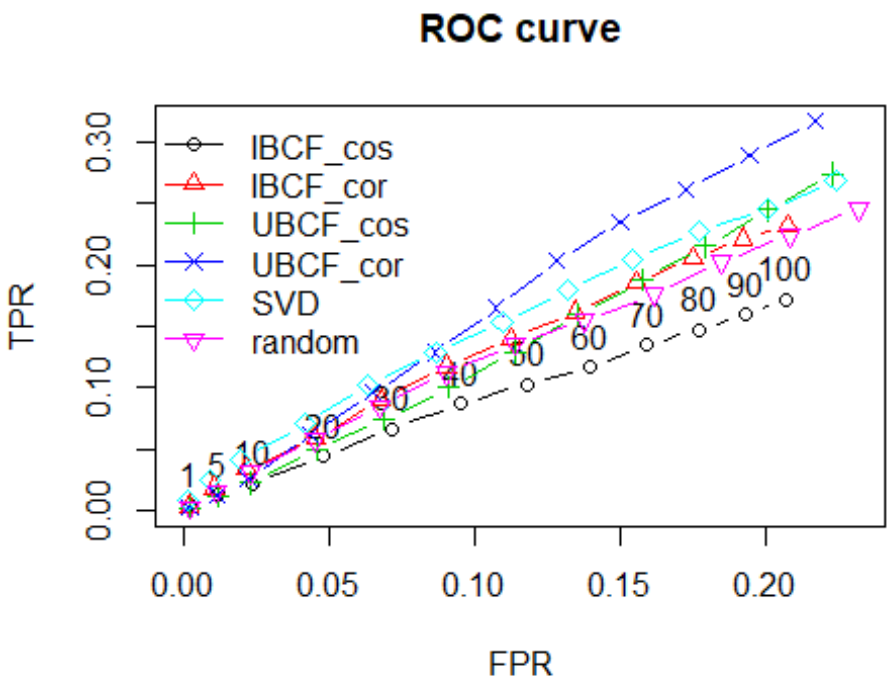
Shahin Ahmed
Student # 500995601

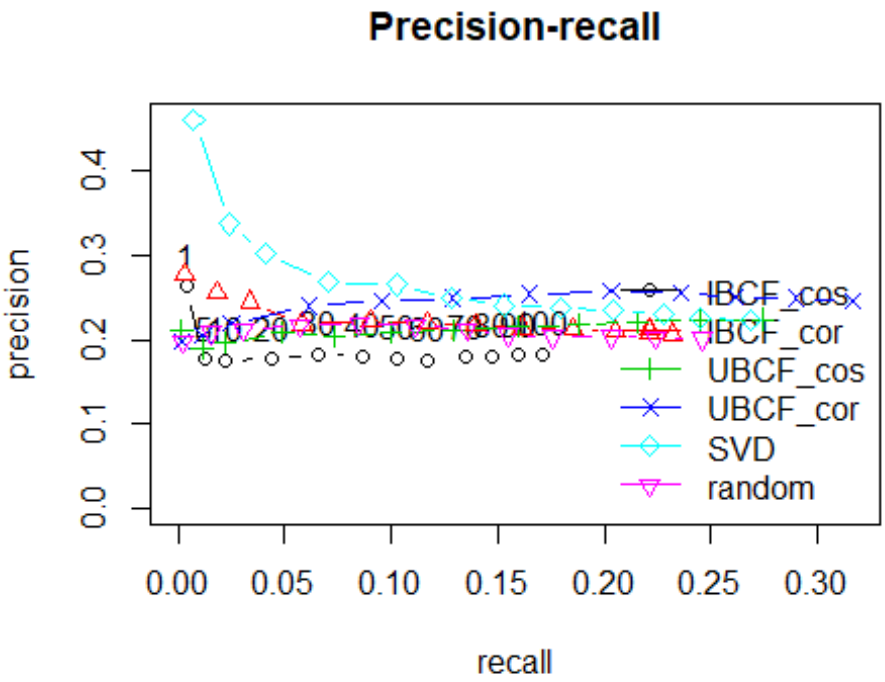
Comparing Models With Varying Values Of Recommendation (RMSE)

Model	RMSE
IBCF_cos	1.297885
IBCF_cor	1.233386
UBCF_cos	1.047406
UBCF_cor	0.983980
SVD	1.007860
Random	1.289798

Identifying the most best performer model

I compare the models by building a chart displaying their ROC curves and Precision/recall curves.





Conclusions and Discussion

In this project, I have developed and evaluated a collaborative filtering recommender (CFR) system for recommending movies. Several models were compared on a reduced dataset. The two top models (SVD & UBCF) were selected to give recommendations on my personal ratings. We see that the best performing model is built by using UBCF and the Pearson correlation as a similarity measure. The model consistently achieves the highest true positive rate for the various false-positive rates and thus delivers the most relevant recommendations.

Let us discuss the strengths and weaknesses of the User-based Collaborative Filtering approach in general.

Strengths:

User-based Collaborative Filtering gives recommendations that can be complements to the item the user was interacting with. This might be a stronger recommendation than what a item-based recommender can provide as users might not be looking for direct substitutes to a movie they had just viewed or previously watched.

Weaknesses:

User-based Collaborative Filtering is a type of Memory-based Collaborative Filtering that uses all user data in the database to create recommendations. Comparing the pairwise correlation of every user in your dataset is not scalable. If there were millions of users, this computation would be very time consuming. Possible ways to get around this would be to implement some form of

dimensionality reduction, such as Principal Component Analysis, or to use a model-based algorithm instead. Also, user-based collaborative filtering relies on past user choices to make future recommendations. The implications of this is that it assumes that a user's taste and preference remain constant over time, which might not be true and makes it difficult to pre-compute user similarities offline.

Future Work:

This report briefly describes simple models that predicts ratings. There are two others widely adopted approaches not discussed here: Content-based Models and collaborative filtering Matrix Factorization Models. The recommenderlab package implements these methods and provides an environment to build and test recommendation systems.

References:

[1] Netflix prize: Home. http://www.netflixprize.com/. (Accessed on03/07/2016).

[2] Francesco Ricci, Lior Rokach, and Bracha Shapira. Introduction to recommender systems handbook. Springer, 2011.

[3] J. B. Schafer, D. Frankowski, J. Herlocker, and S. Sen, ``Collaborative filtering recommender systems," in The Adaptive Web. Berlin, Germany: Springer, 2007, pp. 291_324.

[4] P. Lops, M. de Gemmis, and G. Semeraro, ``Content-based recommender systems: State of the art and trends," in Recommender Systems Handbook. Boston, MA, USA: Springer, 2011, pp. 73_105.

[5] S. Trewin, ``Knowledge-based recommender systems," Encyclopedia Library Inf. Sci., vol. 69, no. 32, p. 180, 2000.

[6] R. Burke, ``Hybrid recommender systems: Survey and experiments," User Model. User-Adapted Interaction, vol. 12, no. 4, pp. 331_370, 2002.

[7] Linyuan Lua, Matus Medo, Chi Ho Yeung, Yi-Cheng Zhang, Zi-Ke Zhang,Tao Zhou, et al. "Recommender systems" Physics Reports 519.1 (2012): 1-49.

[8] Bobadilla, J., Ortega, F., Hernando, A. and Gutierrez, A. (2013) 'Recommender systems survey', *Knowledge-Based Systems*, July, Vol. 46, pp.109–132.

[9] Su, X. and Khoshgoftaar, T.M. (2009) 'A survey of collaborative filtering techniques', *Advances in Artificial Intelligence*, Vol. 3, pp.1–20.

[10] Pennock, D.M., Horvitz, E., Lawrence, S. and Giles, C.L. (2000) 'Collaborative filtering by personality diagnosis: a hybrid memory and model-based approach', *Proc. 16th Conf. Uncertainty in Artificial Intelligence*, pp.473–480.

[11] Herlocker, J.L., Konstan, J.A., Terveen, L.G. and Riedl, J.T. (2004) 'Evaluating collaborative filtering recommender systems', *ACM Transactions on Information Systems*, Vol. 22, No. 1, pp.5–53.

[12] Candillier, L., Meyer, F. and Boull, M. (2007) 'Comparing state-of-the-art collaborative filtering systems', *LNAI*, Vol. 4571, pp.548–562.

[13] Resnick, P., Iakovou, N., Sushak, M., Bergstrom, P. and Riedl, J. (1994) 'GroupLens: an open architecture for collaborative filtering of netnews', *Proc. 1994 Computer Supported Cooperative Work Conf.*, pp.175–186.

[14] Kendall, M. and Gibbons, J.D. (1990) *Rank Correlation Methods*, 5th ed., Charles Griffin, Edward Arnold, London.

[15] Zhao, Z.D. and Shang, M.S. (2010) 'User-based collaborative-filtering recommendation algorithms on Hadoop', in *International Workshop on Knowledge Discovery and Data Mining*, pp.478–481.

[16] Wang, J., De Vries, A.P. and Reinders, M.J.T. (2006) 'Unifying user-based and item-based collaborative filtering approaches by similarity fusion', *Proc. 29th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR'06)*, ACM, New York, NY, pp.501–508.

[17] Sarwar, B., Karypis, G., Konstan, J. and Riedl, J. (2001) 'Item-based collaborative filtering recommendation algorithms', *Proc. 10th ACM International World Wide Web Conference*, pp.285–295.

[18] Treerattnapitak, K. and Juruskulchai, C. (2009) 'Items based fuzzy c-mean clustering for collaborative filtering', *Information Tech. J.*, Vol. 5, No. 10, pp.30–34.

[19] Pennock, D.M., Horvitz, E., Lawrence, S. and Giles, C.L. (2000) 'Collaborative filtering by personality diagnosis: a hybrid memory and model-based approach', *Proc. 16th Conf. Uncertainty in Artificial Intelligence*, pp.473–480.

[20] Chen, Y.H. and George, E.I. (1993) 'A Bayesian model for collaborative filtering', *Proc. 7th International Workshop on Artificial*.

[21] Kim, T.H., Park, S.I. and Yang, S.B. (2008) 'Improving prediction quality in collaborative filtering based on clustering', *Proc. IEEE/WIC/ACM International Conference*, pp.704–710.

[22] Tyagi, S. and Bharadwaj, K.K. (2012) 'A hybrid recommender system using rule-based and case-based reasoning', *International Journal of Information and Electronics Engineering*, Vol. 2, No. 4, pp.586–590.

[23] Adomavicius, G., Tuzhilin, A. and Zheng, R. (2011) 'REQUEST: a query language for customizing recommendations', *Information System Research*, Vol. 22, No. 1, pp.99–117.

[24] Anand, D. and Bharadwaj, K.K. (2011) 'Utilizing various sparsity measures for enhancing accuracy of collaborative recommender systems based on local and global similarities', *Expert Syst. Appl.*, Vol. 38, No. 5, pp.5101–5109.

[25] Bergholz, A. (2003) 'Coping with sparsity in a recommender system', *Lecture Notes Comput. Sci*., Vol. 2703, pp.86–99.

[26] Hyung, J.A. (2008) 'A new similarity measure for collaborative filtering to alleviate the new user coldstarting problem', *Information Sciences: an International Journal*, January, Vol. 178, No. 1, pp.37–51.

[27] Gong, S. (2010). A collaborative filtering recommendation algorithm based on user clustering and item clustering. Journal of Software, 5(7), 745-752.

[28] Puntheeranurak, S., & Chaiwitooanukool, T. (2011, July). An Item-based collaborative filtering method using Item-based hybrid similarity. In Software Engineering and Service Science (ICSESS), 2011 IEEE 2nd International Conference on (pp. 469-472). IEEE.

[29] Sun, D., Luo, Z., & Zhang, F. (2011, October). A novel approach for collaborative filtering to alleviate the new item cold-start problem. In Communications and Information Technologies (ISCIT), 2011 11th International Symposium on (pp. 402-406). IEEE.