CIND820 Big Data Analytics

Developing a Movie Recommendation System Study with machine learning

Gopal Narasimhaiah

040703878

Ceni Babaoglu, Ph.D

Table of Contents

[Abstract 3](#_Toc100004138)

[Introduction 4](#_Toc100004139)

[Literature Review 5](#_Toc100004140)

[Dataset 6](#_Toc100004141)

[System/Approach 11](#_Toc100004142)

[Step 1: Load the Dataset 12](#_Toc100004143)

[Step 2: Data Preparation and Cleaning 12](#_Toc100004144)

[Exploring Similarity Data 15](#_Toc100004145)

[Heatmap of the rating matrix 17](#_Toc100004146)

[Step 4: Data Analysis and Modelling 18](#_Toc100004147)

[Step 5: Comparison of Collaborative Filtering Methods 23](#_Toc100004148)

[Conclusion 32](#_Toc100004149)

[References 33](#_Toc100004150)

# Abstract

The pandemic has made the human population stay at home longer which has opened the number of available hours to them to use and consume. One activity that has been accessible is the internet and the usage has increased over time due to the lack of alternative activities and the choice to participate and be stimulated by the offerings of the Internet. From Statistics Canada - Almost half of Canadians (48%) streamed video content, such as Netflix, Crave, news, concerts or fitness videos, more often since the start of the pandemic. New York Times reported 15-17% increase in Netflix and YouTube usage and Forbes states there has been a increase of 12%.

With this trend, there is a place where people will want access to items, products, videos faster and tailored to them. The system that takes this space is called the Recommender systems which has been prevalent these days and avoidable in our daily journey. My project will focus on developing a recommender system which will help individuals retrieve content based on their interests through algorithms that will create choices and customized lists based on that user  
  
Dataset: From GroupsLens (University of Minnesota)

* Available at MovieLens site
  + Movies.csv
  + Ratings.csv
  + Links.csv
  + Tags.csv

Objective: To attain a greater understanding of Collaborative Filters and Contest Based Filter Models for a hybrid model to answer the following question

What performance can collaborative filtering produce movie recommendations based on the related dataset.

DataSet  
<https://grouplens.org/datasets/movielens/>  
  
Github  
<https://github.com/gnarasim311/cind820capstone>

References   
<https://www150.statcan.gc.ca/n1/pub/45-28-0001/2021001/article/00027-eng.htm>

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

<https://towardsdatascience.com/introduction-to-recommender-systems-6c66cf15ada>

Introduction  
  
In todays evolved Internet Age of Business, having a system in place to predict what item/content/story a user wants is a paramount to organization’s success. The Online Giants of Youtube, Netflix, Amazon, Instagram all have highly manicured recommendation systems in which they predict new content that is relevant to their customer base. Netflix which has become a normal web service in a media consumer lives opened up their recommendation system strategy in a form of competition. On 6 October 2006, Netflix, Inc., launched the Netflix Prize, a contest offering US$1m to the first individual or team to develop a recommendation system capable of predicting movie ratings with at least 10% greater accuracy than Cinematch, the company’s existing system [1]. Comparable recommendation systems belonging to Amazon, Facebook, Google,Match, Microsoft, Twitter, and other technology-driven companies tend to operate similarly, their inner workings “wired shut” with patent and trade secret laws, non-disclosure agreements, non-compete clauses, and other legal instruments. [1].

**Recommendation Systems** – use the opinion of members of the community to help the individual within the community identify the information of products most likely to be interesting to them or relevant to their needs [2]

**Recommendation Systems Techniques**Collaborative Filtering (CF) - produces recommendations according to the similarities between the active user and other users, or between the target items and similar items that have been rated by the active users [3]

Content Based Filtering (CBF) - , produces recommendations aimed at discovering product attributes and relations between products and between customers [3]  
  
Knowledge Based Approach – produces recommendations based on specific queries that are 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. [4]

Hybrid Approach – a blend of all approaches above to generate recommendations. [3]

Literature Review  
  
In this project, the goal is developing a hybrid recommendation system that is based on User Based Collaborative Filter (UBCF) and Item Based Collaborative Filter (IBCF). Recommendations Systems are what make many Online Businesses function.

The recommendation techniques were listed in the introduction and separated into 4 categories. Collaborative filtering (CF) permitting the exploitation of information about user interaction and transactions, such as product ratings and orders. [5]Content Based Filtering is aimed at discovering product attributes and relations between products and between customers. Knowledge Based Filtering is enabling the generation of advice based on explicit human knowledge about the item assortment, user preferences, and recommendation criteria [5] Hybrid Recommendation is based on everything above[5]. In the context of this project of developing a system, we need to understand UBCF and IBCF. User-based collaborative filtering (UBCF) engine follows the “people like you” logic which recommends to a user an item that similar users liked before[6][7][8]. Item-based collaborative filtering (IBCF) follows the logic “if you like this, you might also like that also” and it recommends items that are similar to the ones you previously liked. [6][7][8].

The algorithms that are used but not limited to are based on clustering, the probability of collaborative filtering algorithm, collaborative filtering based on neural networks ,collaborative filtering matrix decomposition based on a variety of models such as the probability model, Bayes model abstraction, maximum entropy model, Gibbs abstract and linear regression.[9][10][11] Even with these algorithms here still exists problems of sparsity, early rater and cold start.[9]  
  
In recommendation engines, cold start refers to the condition when the recommendation system is not yet optimal to generate the best results because of data sparsity i.e. problems in finding an ample number of similar users since in general the active users only scored a small fraction of items.[9][10][11]

To solve the problems of scalability and sparsity in the collaborative filtering, this paper proposed a personalized recommendation approach joins the user clustering technology and item clustering technology[12][13]. Users are clustered based on users’ ratings on items, and each users cluster has a cluster center. Based on the similarity between target user and cluster centers, the nearest neighbors of target user can be found and smooth the prediction where necessary. Then, the proposed approach utilizes the item clustering collaborative filtering to produce the recommendations. The recommendation joining user clustering and item clustering collaborative filtering is more scalable and more accurate than the traditional one. [13][14][15]  
  
To potentially solve the cold start problem using a hybrid method which first cluster items using the rating matrix and then uses the clustering results to build a decision tree to combine items with existing ones. [13][14][15]

Dataset  
  
The dataset that will be used will be MovieLens and is available at MovieLens website which is publicly available.   
  
Dataset – For the interest of this project – used the smallest set available and at the time of the download contains 100836 ratings and 3683 tag applications across 9742 movies. These data were created by 610 users between March 29, 1996 and September 24, 2018. This dataset was generated on September 26, 2018. Movies and Ratings datasets will be used in building a recommender system.

**Load the datasets**

movies <- read.csv("movies.csv")  
ratings <- read.csv("ratings.csv")  
links <- read.csv("links.csv")  
tags <- read.csv("tags.csv")

**Movie summary**

dim(movies)

## [1] 9742 3

There are 9742 observations and 3 attributes in the movies data-set.

head(movies)

## movieId title  
## 1 1 Toy Story (1995)  
## 2 2 Jumanji (1995)  
## 3 3 Grumpier Old Men (1995)  
## 4 4 Waiting to Exhale (1995)  
## 5 5 Father of the Bride Part II (1995)  
## 6 6 Heat (1995)  
## genres  
## 1 Adventure|Animation|Children|Comedy|Fantasy  
## 2 Adventure|Children|Fantasy  
## 3 Comedy|Romance  
## 4 Comedy|Drama|Romance  
## 5 Comedy  
## 6 Action|Crime|Thriller

str(movies)

## 'data.frame': 9742 obs. of 3 variables:  
## $ movieId: int 1 2 3 4 5 6 7 8 9 10 ...  
## $ title : chr "Toy Story (1995)" "Jumanji (1995)" "Grumpier Old Men (1995)" "Waiting to Exhale (1995)" ...  
## $ genres : chr "Adventure|Animation|Children|Comedy|Fantasy" "Adventure|Children|Fantasy" "Comedy|Romance" "Comedy|Drama|Romance" ...

summary(movies)

## movieId title genres   
## Min. : 1 Length:9742 Length:9742   
## 1st Qu.: 3248 Class :character Class :character   
## Median : 7300 Mode :character Mode :character   
## Mean : 42200   
## 3rd Qu.: 76232   
## Max. :193609

**Rating summary**

dim(ratings)

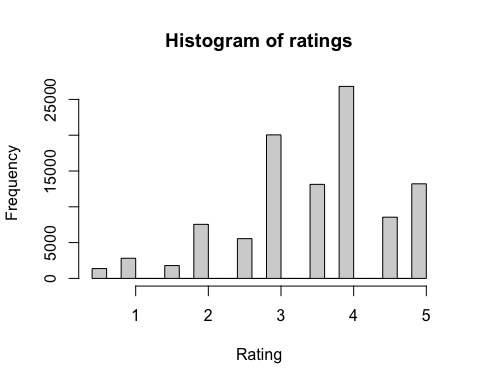
## [1] 100836 4

There are 100836 observations and 4 attributes in the ratings data-set.

head(ratings)

## userId movieId rating timestamp  
## 1 1 1 4 964982703  
## 2 1 3 4 964981247  
## 3 1 6 4 964982224  
## 4 1 47 5 964983815  
## 5 1 50 5 964982931  
## 6 1 70 3 964982400

hist(ratings$rating, main = "Histogram of ratings", xlab = "Rating")



str(ratings)

## 'data.frame': 100836 obs. of 4 variables:  
## $ userId : int 1 1 1 1 1 1 1 1 1 1 ...  
## $ movieId : int 1 3 6 47 50 70 101 110 151 157 ...  
## $ rating : num 4 4 4 5 5 3 5 4 5 5 ...  
## $ timestamp: int 964982703 964981247 964982224 964983815 964982931 964982400 964980868 964982176 964984041 964984100 ...

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

**Links summary**

dim(links)

## [1] 9742 3

There are 9742 observations and 3 attributes in the link data-set.

head(links)

## movieId imdbId tmdbId  
## 1 1 114709 862  
## 2 2 113497 8844  
## 3 3 113228 15602  
## 4 4 114885 31357  
## 5 5 113041 11862  
## 6 6 113277 949

str(links)

## 'data.frame': 9742 obs. of 3 variables:  
## $ movieId: int 1 2 3 4 5 6 7 8 9 10 ...  
## $ imdbId : int 114709 113497 113228 114885 113041 113277 114319 112302 114576 113189 ...  
## $ tmdbId : int 862 8844 15602 31357 11862 949 11860 45325 9091 710 ...

summary(links)

## 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

**Tags summary**

dim(tags)

## [1] 3683 4

There are 3683 observations and 4 attributes in the tag data-set.

head(tags)

## userId movieId tag timestamp  
## 1 2 60756 funny 1445714994  
## 2 2 60756 Highly quotable 1445714996  
## 3 2 60756 will ferrell 1445714992  
## 4 2 89774 Boxing story 1445715207  
## 5 2 89774 MMA 1445715200  
## 6 2 89774 Tom Hardy 1445715205

str(tags)

## 'data.frame': 3683 obs. of 4 variables:  
## $ userId : int 2 2 2 2 2 2 2 2 2 7 ...  
## $ movieId : int 60756 60756 60756 89774 89774 89774 106782 106782 106782 48516 ...  
## $ tag : chr "funny" "Highly quotable" "will ferrell" "Boxing story" ...  
## $ timestamp: int 1445714994 1445714996 1445714992 1445715207 1445715200 1445715205 1445715054 1445715051 1445715056 1169687325 ...

summary(tags)

## userId movieId tag timestamp   
## Min. : 2.0 Min. : 1 Length:3683 Min. :1.137e+09   
## 1st Qu.:424.0 1st Qu.: 1262 Class :character 1st Qu.:1.138e+09   
## Median :474.0 Median : 4454 Mode :character Median :1.270e+09   
## Mean :431.1 Mean : 27252 Mean :1.320e+09   
## 3rd Qu.:477.0 3rd Qu.: 39263 3rd Qu.:1.498e+09   
## Max. :610.0 Max. :193565 Max. :1.537e+09

# System/Approach

Below is the framework to build a recommendation system shown graphically

Load the data

Data Prep/Cleaning

Min Thresholds

Normalizing the Data

Splitting the Data Test/Train

Data Exploration/Visualization

Data Analysis and Modelling

Comparing the Collaborative Filtering Models

Step 1: Load the Dataset  
Loading dataset is split into four files- movies.csv, ratings.csv, links.csv and tags.csv and will be using R language to load and be saved in rmd format.

Step 2: Data Preparation and Cleaning

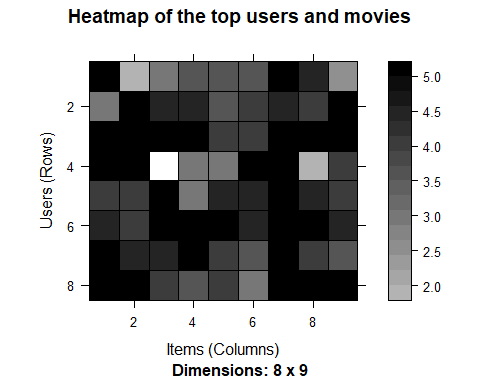
*2.1 Minimum Thresholds* - limit the input data based on minimum thresholds:

For this project we restrict the model training to those users who have rated at least 50 movies, and those movies that have been rated by at least 100 users

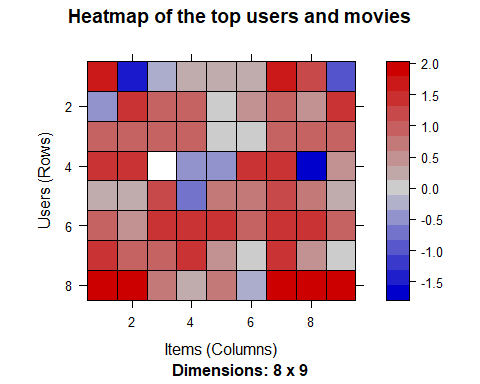
In order to predict the most relevant data, rating matrix is defined with the minimum number of users per rated movie as 50 and the minimum views number per movie as 50:

378 x 436 rating matrix of class ‘realRatingMatrix’ with 36214 ratings.

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



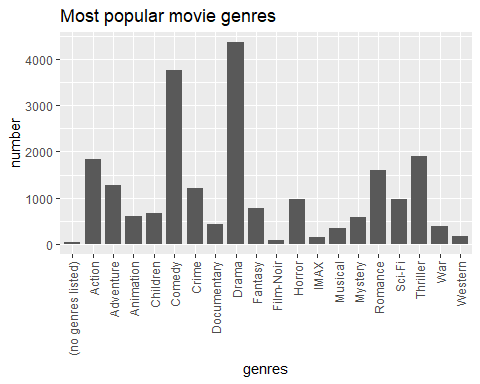
*Normalizing the Data –* to eliminate bias in data will assign the average rating given by each user to 0 to remove user who give high and low ratings consistently.



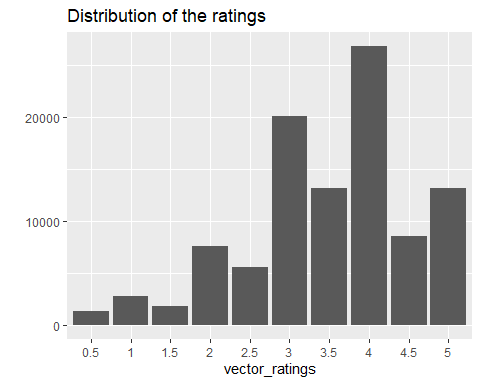
*Split the Data for Test/Train*   
  
To split the data into test and train sets, so that we could test the models on test data and later could compare different data models. I build the model using 80% of the whole dataset as a training set, and 20% - as a test set.

**Step 3 : Data Exploration/Visualization**Exploring the dataset and deciphering the movie business from this dataset

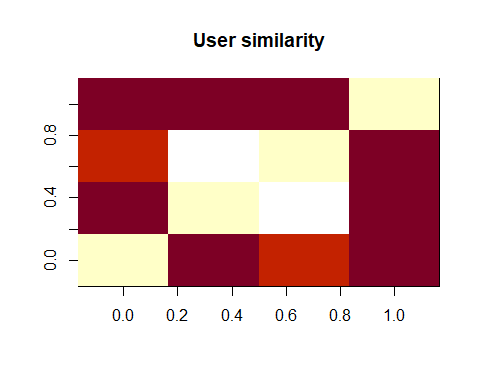
**The number of genres in each movie is listed in this table**



**Ratings Count**

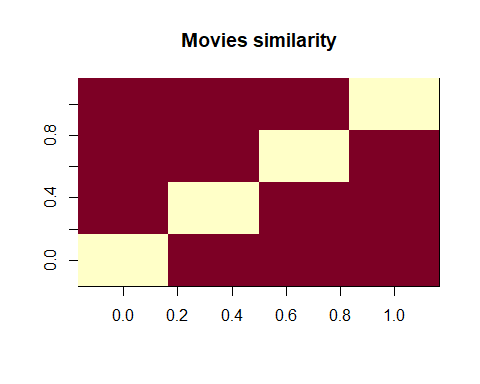


## Exploring Similarity Data

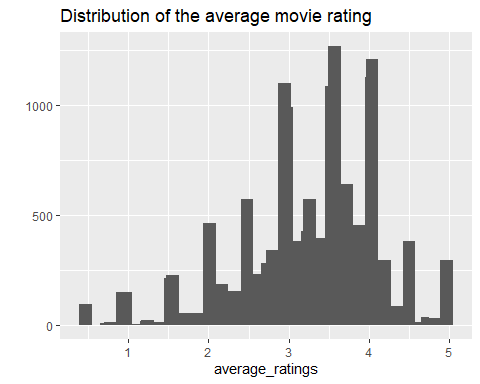


Each row and each column corresponds to a user, and each cell corresponds to the similarity between two users.

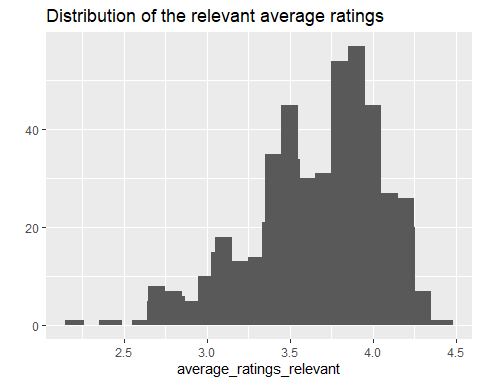
Using the same approach, We computed between the first four movies.



**Average movie rating**



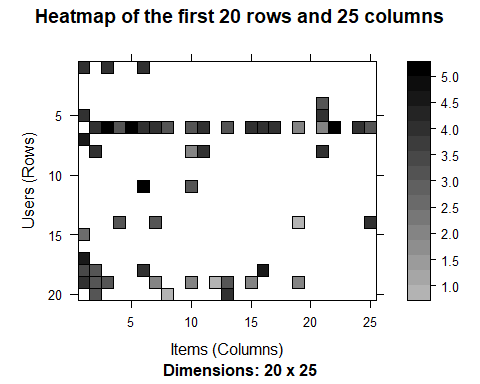
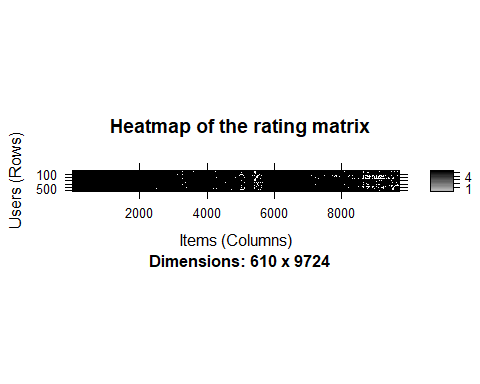
**Average relevant rating**



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.

The second image above shows the distribution of the relevant average ratings. All the rankings are between 2.16 and 4.45. The extremes were removed. The highest value changes, and now it is around 4.

### Heatmap of the rating matrix



Step 4: Data Analysis and Modelling   
Analysis of UBCF and IBCF models

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

**UBCF Model:**

Utilizing 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:

* the top k users (k-nearest\_neighbors)
* the users whose similarity is above a defined threshold

1. 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

1. Pick the top-rated movies.

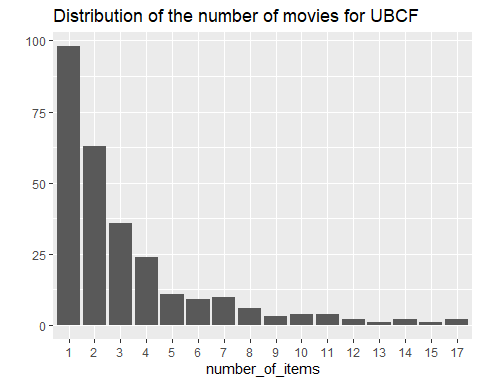
**Building the recommendation system:**

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. Generated 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 equals 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..

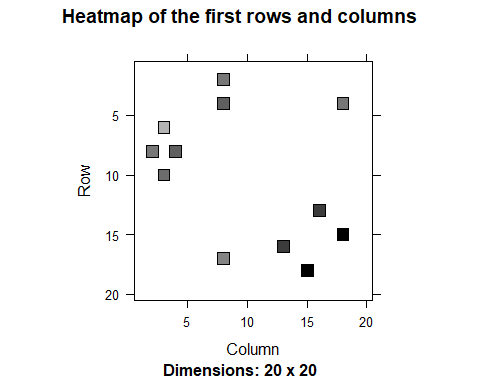
The starting point is a rating matrix in which rows correspond to users and columns correspond to items. The core algorithm is based on these steps:

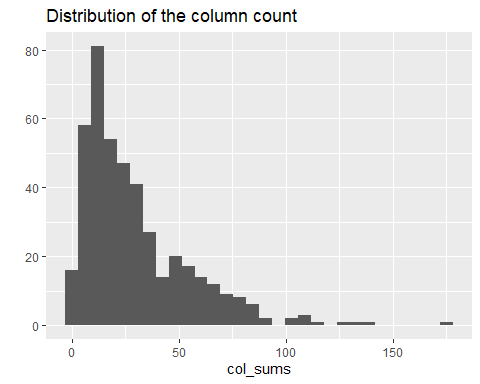
1. each two items, measured how similar they are in terms of having received similar ratings by similar users
2. each item, identify the k most similar items
3. each user, identify the items that are most similar to the user’s purchases

**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. 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. 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.

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:**

It is now possible to make movie recommendations to the users in the test group. I set the value of n recommended to 10 to indicate the number of movies to recommend to each user.

The programme extracts each user's rated movies. Starting with the similarity matrix, it detects all of the movie's similar things. The system then scores each similar item as follows:

* Get the user rating for each purchase that is linked to this item. A weight is assigned to the rating
* Calculate the item's resemblance to each purchase associated with it
* Extract the similarity of the item with each purchase associated with this item.
* Multiply each weight by the similarity factor.
* Put it all together.

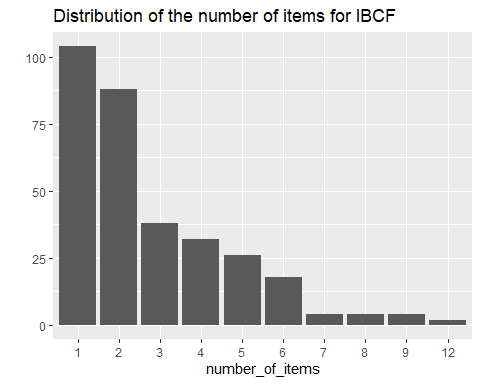
Then, the algorithm identifies the top 10 recommendations:

## [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 ] 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

**Distribution of the number of items for IBCF:**



## Step 5: Comparison of Collaborative Filtering Methods

Evaluating collaborative filtering methods based on parameters to determine their effectiveness

**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

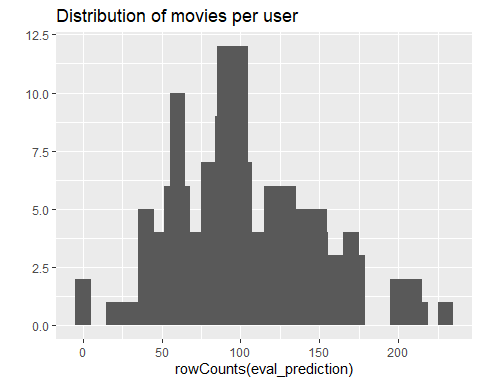
**Using cross-validation to validate models**

The k-fold cross-validation method is the most accurate, but it is also the most computationally intensive.

Using this method, we divide the data into pieces, select one as the test set, and assess the correctness. Then we may repeat the process for each chunk and calculate the average accuracy.## [1] 282 282 282 282

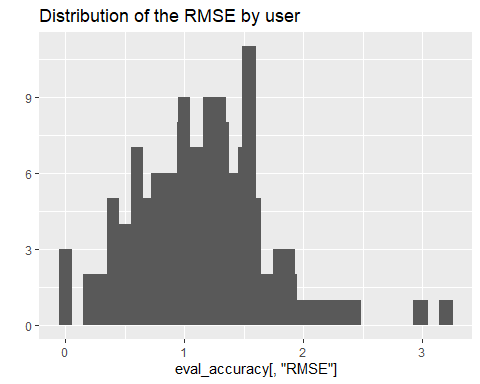
Using 4-fold approach, we get four sets of the same size 282

**Evaluating the ratings**



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

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 |

.

**Evaluating the recommendations**

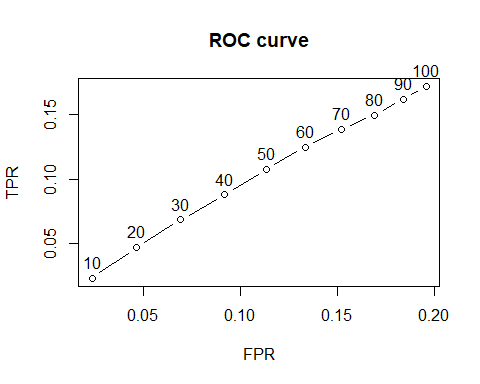
Another way to measure accuracy is by comparing the recommendations with the purchases having a positive rating. For this, I can make use of a prebuilt evaluate function in recommenderlab library. The function evaluates the recommender performance depending on the number n of items to recommend to each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

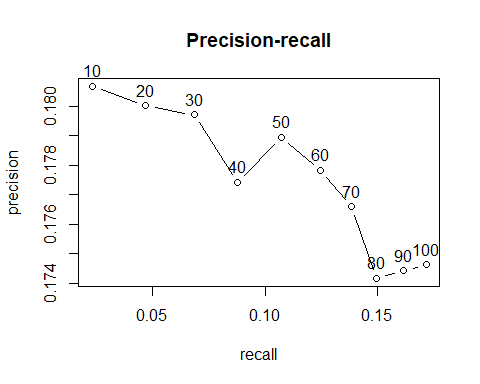
**IBCF Model**

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

ROC and the precision/recall curves for IBCF model:



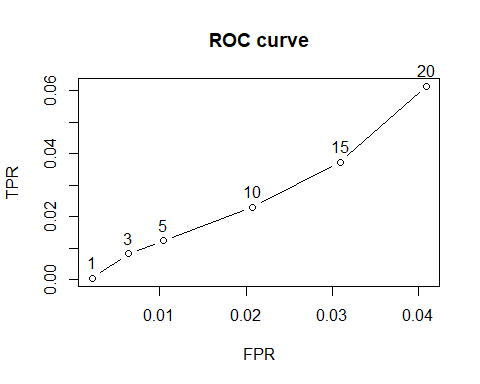


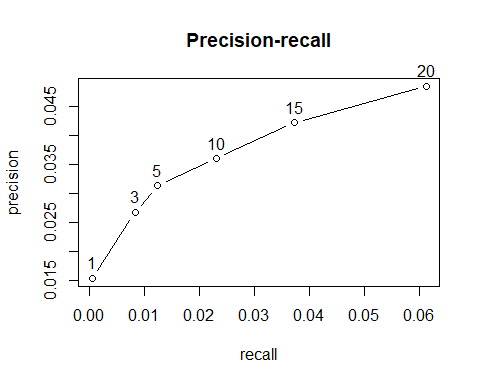
**UBCF Model**

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:





**Comparing different models**

We assess the performance of the UBCF and IBCD models, although different models can be reviewed and compared. Our benchmark metric is the AUC. To make it easier to compare different models, I've created a 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

Confusion matrix, UBCF pearson and SVD similarity

We can investigate performance evaluations using UBCF with pearson distance and SVD with k=100, for example. With the number of suggestions, the genuine positive ratio rises. TPR and Recall are the same thing.

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

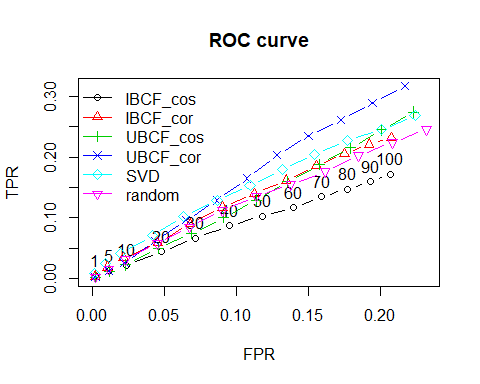
30 0.2776316 0.096461529 0.096461529 0.063633951

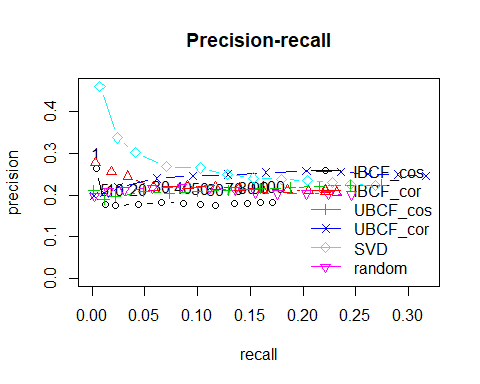
40 0.2690789 0.123931099 0.123931099 0.086062463

# 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





## 

Conclusion  
  
I created and tested a collaborative filtering recommender (CFR) system for movie recommendations as part of this project. On a smaller dataset, several models were compared. The top two models (SVD and UBCF) were chosen to make suggestions based on my personal preferences. We can see that utilising UBCF and the Pearson correlation as a similarity metric produces the highest performing model. For each false-positive rate, the model consistently obtains the greatest true positive rate, resulting in the most relevant suggestions.

Let's look at the advantages and disadvantages of the User-based Collaborative Filtering method in general.

**Advantages:**

User-based Collaborative Filtering provides suggestions that can be useful additions to the item with which the user was interacting. Users may not be seeking for straight equivalents to a movie they had just seen or previously watched, hence this may be a greater recommendation than what an item-based recommender can deliver.

**Disadvantages:**

User-based Collaborative Filtering is a form of Memory-based Collaborative Filtering that generates recommendations based on all user data in the database. It is not scalable to compare the pairwise correlation of every user in your dataset. This computation would take a long time if there were millions of users. Implementing some sort of dimensionality reduction, such as Principal Component Analysis, or using a model-based technique instead could be a way to get around this. Furthermore, user-based collaborative filtering makes future recommendations based on previous user decisions. This has the effect of assuming that a user's taste and preference remain consistent over time, which may or may not be the case, making it impossible to compute user similarities offline.

# References

[1] Hallinan, B., & Striphas, T. (2016). Recommended for you: The Netflix Prize and the production of algorithmic culture. New Media & Society, 18(1), 117–137. https://doi.org/10.1177/1461444814538646

[2] Shardanand, U., Maes, P. 1995. Social information filtering: Algorithms for automating "Word of Mouth". In Proceedings of the Conference on Human Factors in Computing Systems (CHI'95). ACM, New York, 210--217.

[3] Ali A. Amer, Hassan I. Abdalla, Loc Nguyen,Enhancing recommendation systems performance using highly-effective similarity measures,Knowledge-Based Systems,Volume 217,2021, <https://doi.org/10.1016/j.knosys>

[4] Burke, R. (2000). Knowledge-based recommender systems. *Encyclopedia of library and information systems*, *69*(Supplement 32), 175-186

[5] Min Dong, Xianyi Zeng, Ludovic Koehl, Junjie Zhang,An interactive knowledge-based recommender system for fashion product design in the big data environment,Information Sciences,Volume 540,2020,Pages 469-488

[6] Deng, Z., & Wang, J. (2013). Collaborative Filtering Algorithm Based on User Clustering. Applied Mechanics and Materials, 411-414, 1044. <http://dx.doi.org/10.4028/www.scientific.net/AMM.411-414.1044>

[7]Ahmad A. Kardan, Mahnaz Ebrahimi,A novel approach to hybrid recommendation systems based on association rules mining for content recommendation in asynchronous discussion groups,Information Sciences,Volume 219,2013,Pages 93-110,

[8]Xiaofeng Zhang, Huijie Liu, Xiaoyun Chen, Jingbin Zhong, Di Wang,A novel hybrid deep recommendation system to differentiate user’s preference and item’s attractiveness,Information Sciences,Volume 519,2020,Pages 306-316

[9] Ajaegbu, C. An optimized item-based collaborative filtering algorithm. J Ambient Intell Human Comput 12, 10629–10636 (2021). <https://doi-org.ezproxy.lib.ryerson.ca/10.1007/s12652-020-02876-1>

[10] Chen, W., Niu, Z., Zhao, X. et al. A hybrid recommendation algorithm adapted in e-learning environments. World Wide Web 17, 271–284 (2014). <https://doi-org.ezproxy.lib.ryerson.ca/10.1007/s11280-012-0187-z>

[11] Alberto Huertas Celdrán, Manuel Gil Pérez, Félix J. García Clemente, Gregorio Martínez Pérez,Design of a recommender system based on users’ behavior and collaborative location and tracking,Journal of Computational Science,Volume 12,2016,Pages 83-94,

[12]Tahmasebi, F., Meghdadi, M., Ahmadian, S. et al. A hybrid recommendation system based on profile expansion technique to alleviate cold start problem. Multimed Tools Appl 80, 2339–2354 (2021). <https://doi-org.ezproxy.lib.ryerson.ca/10.1007/s11042-020-09768-8>

[13] Schafer J.B., Frankowski D., Herlocker J., Sen S. (2007) Collaborative Filtering Recommender Systems. In: Brusilovsky P., Kobsa A., Nejdl W. (eds) The Adaptive Web. Lecture Notes in Computer Science, vol 4321. Springer, Berlin, Heidelberg. https://doi.org/10.1007/978-3-540-72079-9\_9

[14] Cite: SongJie Gong, "A Collaborative Filtering Recommendation Algorithm Based on User Clustering and Item Clustering," Journal of Software vol. 5, no. 7, pp. 745-752, 2010.

[15] 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.