Movie Recommendation System using ML in R

Vatsal Desai

January 29 2021

Abstract

The main goal of this machine learning project is to build a recommendation engine that recommends movies to users. We will be developing an Item Based Collaborative Filter using recommenderlab package.

Contents

Introduction **2**

What is a Recommendation System 2

Dataset and libraries **2**

Data pre-processing **4**

Choosing recommendation model **6**

Exploring similar data **7**

Most viewed movies visualization **10**

Heatmap of movie ratings **11**

Data preparation **12**

Finding useful data 12

Data normalization 14

Data binarization 15

Collaborative filtering system **16**

Building recommendation system **16**

Summary **21**

Introduction

What is a Recommendation System

A recommendation system provides suggestions to the users through a filtering process that is based on user preferences and browsing history. The information about the user is taken as an input. The information is taken from the input that is in the form of browsing data. This information reflects the prior usage of the product as well as the assigned ratings. A recommendation system is a platform that provides its users with various contents based on their preferences and likings. A recommendation system takes the information about the user as an input. The recommendation system is an implementation of the machine learning algorithms.
A recommendation system also finds a similarity between the different products. For example, Netflix Recommendation System provides you with the recommendations of the movies that are similar to the ones that have been watched in the past. Furthermore, there is a collaborative content filtering that provides you with the recommendations in respect with the other users who might have a similar viewing history or preferences. There are two types of recommendation systems – Content-Based Recommendation System and Collaborative Filtering Recommendation. In this project of recommendation system in R, we will work on a collaborative filtering recommendation system and more specifically, ITEM based collaborative recommendation system.

Dataset and libraries

In order to build our recommendation system, we have used the *MovieLens* Dataset. You can find the *movies.csv* and *ratings.csv* file that we have used in our Recommendation System Project **here** This data consists of 105339 ratings applied over 10329 movies.

```
library(recommenderlab)
```

```
Loading required package: Matrix
```

```
Loading required package: arules
```

```

Attaching package: 'arules'
```

```
The following objects are masked from 'package:base':

    abbreviate, write
```

```
Loading required package: proxy
```

```

Attaching package: 'proxy'
```

```
The following object is masked from 'package:Matrix':

    as.matrix
```

```
The following objects are masked from 'package:stats':

    as.dist, dist
```

```
The following object is masked from 'package:base':

    as.matrix

Loading required package: registry

Registered S3 methods overwritten by 'registry':
  method               from
  print.registry_field proxy
  print.registry_entry proxy
```

```
library(ggplot2)
library(data.table)
library(reshape2)
```

```
Attaching package: 'reshape2'

The following objects are masked from 'package:data.table':

    dcast, melt
```

```
movie_data <- read.csv("movies.csv", stringsAsFactors = FALSE)
rating_data <- read.csv("ratings.csv")
str(movie_data)
```

```
'data.frame':   10329 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|Roma
```

We can overview the summary of the movies using the *summary()* function. We will also use the *head()* function to print the first six lines of *movie_data*

```
summary(movie_data)
```

```
    movieId          title              genres
 Min.   :     1   Length:10329       Length:10329
 1st Qu.:  3240   Class :character   Class :character
 Median :  7088   Mode  :character   Mode  :character
 Mean   : 31924
 3rd Qu.: 59900
 Max.   :149532
```

```
head(movie_data)
```

```
  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
```

Similarly, we can output the summary as well as the first six lines of the 'rating_data' dataframe

```
summary(rating_data)
```

```
     userId         movieId           rating        timestamp
 Min.   :  1.0   Min.   :     1   Min.   :0.500   Min.   :8.286e+08
 1st Qu.:192.0   1st Qu.:  1073   1st Qu.:3.000   1st Qu.:9.711e+08
 Median :383.0   Median :  2497   Median :3.500   Median :1.115e+09
 Mean   :364.9   Mean   : 13381   Mean   :3.517   Mean   :1.130e+09
 3rd Qu.:557.0   3rd Qu.:  5991   3rd Qu.:4.000   3rd Qu.:1.275e+09
 Max.   :668.0   Max.   :149532   Max.   :5.000   Max.   :1.452e+09
```

```
head(rating_data)
```

```
  userId movieId rating  timestamp
1      1      16    4.0 1217897793
2      1      24    1.5 1217895807
3      1      32    4.0 1217896246
4      1      47    4.0 1217896556
5      1      50    4.0 1217896523
6      1     110    4.0 1217896150
```

Data pre-processing

From the above table, we observe that the *userId* column, as well as the *movieId* column, consist of integers. Furthermore, we need to convert the genres present in the *movie_data* dataframe into a more usable format by the users. In order to do so, we will first create a *one-naught encoding* to create a matrix that comprises of corresponding genres for each of the films.

```
movie_genre <- as.data.frame(movie_data$genres,stringsAsFactors = FALSE)
movie_genre2 <- as.data.frame(tstrsplit(movie_genre[,1], '[|]', type.convert = TRUE),
                              stringAsFactors = FALSE)
colnames(movie_genre2) <- c(1:10)

list_genre <- c("Action", "Adventure", "Animation", "Children",
                "Comedy", "Crime","Documentary", "Drama", "Fantasy",
                "Film-Noir", "Horror", "Musical", "Mystery","Romance",
                "Sci-Fi", "Thriller", "War", "Western")
genre_mat1 <- matrix(0,10330,18)
genre_mat1[1,] <- list_genre
```

```
colnames(genre_mat1) <- list_genre

for (index in 1:nrow(movie_genre2)) {
  for (col in 1:ncol(movie_genre2)) {
    gen_col = which(genre_mat1[1,] == movie_genre2[index,col])
    genre_mat1[index+1,gen_col] <- 1
}
}

genre_mat2 <- as.data.frame(genre_mat1[-1,], stringsAsFactors=FALSE)
for (col in 1:ncol(genre_mat2)) {
  genre_mat2[,col] <- as.integer(genre_mat2[,col])
}
str(genre_mat2)
```

```
'data.frame':   10329 obs. of  18 variables:
 $ Action     : int  0 0 0 0 0 1 0 0 1 1 ...
 $ Adventure  : int  1 1 0 0 0 0 0 1 0 1 ...
 $ Animation  : int  1 0 0 0 0 0 0 0 0 0 ...
 $ Children   : int  1 1 0 0 0 0 0 1 0 0 ...
 $ Comedy     : int  1 0 1 1 1 0 1 0 0 0 ...
 $ Crime      : int  0 0 0 0 0 1 0 0 0 0 ...
 $ Documentary: int  0 0 0 0 0 0 0 0 0 0 ...
 $ Drama      : int  0 0 0 1 0 0 0 0 0 0 ...
 $ Fantasy    : int  1 1 0 0 0 0 0 0 0 0 ...
 $ Film-Noir  : int  0 0 0 0 0 0 0 0 0 0 ...
 $ Horror     : int  0 0 0 0 0 0 0 0 0 0 ...
 $ Musical    : int  0 0 0 0 0 0 0 0 0 0 ...
 $ Mystery    : int  0 0 0 0 0 0 0 0 0 0 ...
 $ Romance    : int  0 0 1 1 0 0 1 0 0 0 ...
 $ Sci-Fi     : int  0 0 0 0 0 0 0 0 0 0 ...
 $ Thriller   : int  0 0 0 0 0 1 0 0 0 1 ...
 $ War        : int  0 0 0 0 0 0 0 0 0 0 ...
 $ Western    : int  0 0 0 0 0 0 0 0 0 0 ...
```

In the next step of Data Pre-processing of data, we will create a *'search matrix'* that will allow us to perform an easy search of the films by specifying the genre present in our list.

```
SearchMatrix <- cbind(movie_data[,1:2], genre_mat2[])
head(SearchMatrix)
```

```
  movieId                              title Action Adventure Animation
1       1                  Toy Story (1995)      0         1         1
2       2                    Jumanji (1995)      0         1         0
3       3           Grumpier Old Men (1995)      0         0         0
4       4          Waiting to Exhale (1995)      0         0         0
5       5 Father of the Bride Part II (1995)      0         0         0
6       6                         Heat (1995)      1         0         0
  Children Comedy Crime Documentary Drama Fantasy Film-Noir Horror Musical
1        1      1     0           0     0       1         0      0       0
2        1      0     0           0     0       1         0      0       0
3        0      1     0           0     0       0         0      0       0
```

```
4        0      1     0           0     1       0         0      0       0
5        0      1     0           0     0       0         0      0       0
6        0      0     1           0     0       0         0      0       0
  Mystery Romance Sci-Fi Thriller War Western
1       0       0      0        0   0       0
2       0       0      0        0   0       0
3       0       1      0        0   0       0
4       0       1      0        0   0       0
5       0       0      0        0   0       0
6       0       0      0        1   0       0
```

There are movies that have several genres, for example, Toy Story, which is an animated film also falls under the genres of Comedy, Fantasy, and Children. This applies to the majority of the films.
For our movie recommendation system to make sense of our ratings through *recommenderlabs*, we have to convert our matrix into a sparse matrix one. This new matrix is of the class *'realRatingMatrix'*. This is performed as follows:

```
ratingMatrix <- dcast(rating_data, userId~movieId, value.var = "rating", na.rm=FALSE)
ratingMatrix <- as.matrix(ratingMatrix[,-1])
ratingMatrix <- as(ratingMatrix, "realRatingMatrix")
ratingMatrix
```

```
668 x 10325 rating matrix of class 'realRatingMatrix' with 105339 ratings.
```

Choosing recommendation model

Let us now overview some of the important parameters that provide us various options for building recommendation systems for movies

```
recommendation_model <- recommenderRegistry$get_entries(dataType = "realRatingMatrix")
names(recommendation_model)
```

```
 [1] "HYBRID_realRatingMatrix"       "ALS_realRatingMatrix"
 [3] "ALS_implicit_realRatingMatrix" "IBCF_realRatingMatrix"
 [5] "LIBMF_realRatingMatrix"        "POPULAR_realRatingMatrix"
 [7] "RANDOM_realRatingMatrix"       "RERECOMMEND_realRatingMatrix"
 [9] "SVD_realRatingMatrix"          "SVDF_realRatingMatrix"
[11] "UBCF_realRatingMatrix"
```

```
lapply(recommendation_model, "[[", "description")
```

```
$HYBRID_realRatingMatrix
[1] "Hybrid recommender that aggegates several recommendation strategies using weighted averages."

$ALS_realRatingMatrix
[1] "Recommender for explicit ratings based on latent factors, calculated by alternating least squares a

$ALS_implicit_realRatingMatrix
[1] "Recommender for implicit data based on latent factors, calculated by alternating least squares alg
```

```
$IBCF_realRatingMatrix
[1] "Recommender based on item-based collaborative filtering."

$LIBMF_realRatingMatrix
[1] "Matrix factorization with LIBMF via package recosystem (https://cran.r-project.org/web/packages/re

$POPULAR_realRatingMatrix
[1] "Recommender based on item popularity."

$RANDOM_realRatingMatrix
[1] "Produce random recommendations (real ratings)."

$RERECOMMEND_realRatingMatrix
[1] "Re-recommends highly rated items (real ratings)."

$SVD_realRatingMatrix
[1] "Recommender based on SVD approximation with column-mean imputation."

$SVDF_realRatingMatrix
[1] "Recommender based on Funk SVD with gradient descend (https://sifter.org/~simon/journal/20061211.htm

$UBCF_realRatingMatrix
[1] "Recommender based on user-based collaborative filtering."
```

Out of all these available models, we shall apply *Item based Collaborative Filtering* model to our data here.

```
recommendation_model$IBCF_realRatingMatrix$parameters
```

```
$k
[1] 30

$method
[1] "Cosine"

$normalize
[1] "center"

$normalize_sim_matrix
[1] FALSE

$alpha
[1] 0.5

$na_as_zero
[1] FALSE
```

Exploring similar data

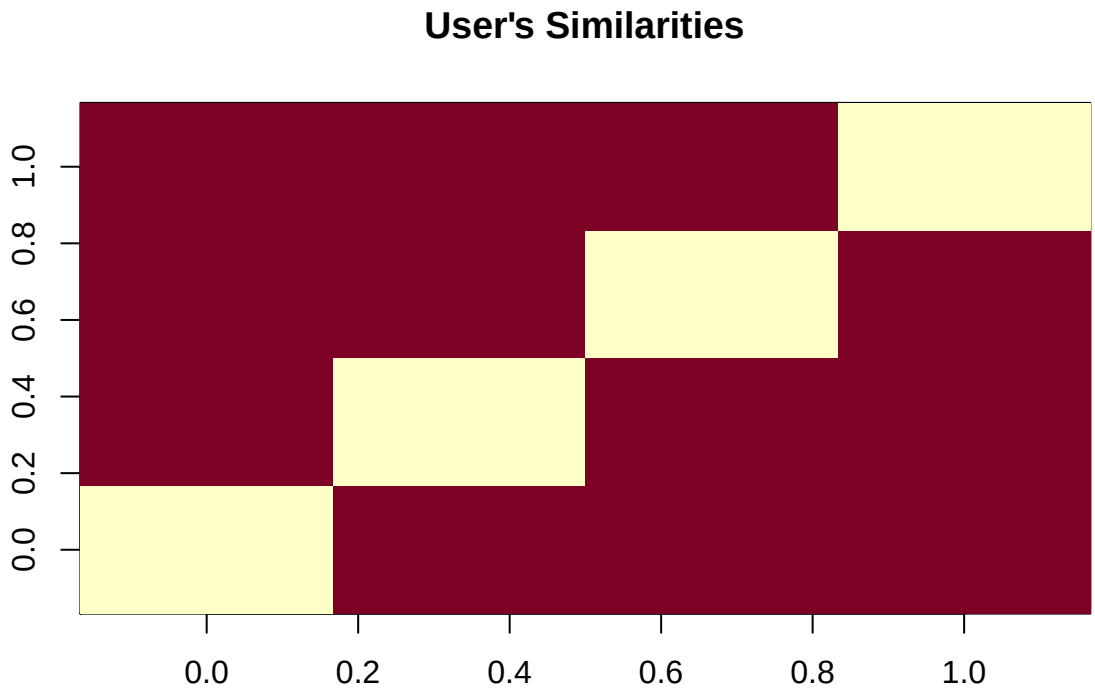
Collaborative Filtering involves suggesting movies to the users that are based on collecting preferences from many other users. For example, if a user A likes to watch action films and so does user B, then the movies that the user B will watch in the future will be recommended to A and vice-versa. Therefore, recommending movies is dependent on creating a relationship of similarity between the two users. With the

help of *recommenderlab*, we can compute similarities using various operators like cosine, pearson as well as jaccard.

```
similarity_mat <- similarity(ratingMatrix[1:4, ],
                             method = "cosine",
                             which = "users")
as.matrix(similarity_mat)
```

```
          1         2         3         4
1 0.0000000 0.9760860 0.9641723 0.9914398
2 0.9760860 0.0000000 0.9925732 0.9374253
3 0.9641723 0.9925732 0.0000000 0.9888968
4 0.9914398 0.9374253 0.9888968 0.0000000
```

```
image(as.matrix(similarity_mat), main = "User's Similarities")
```

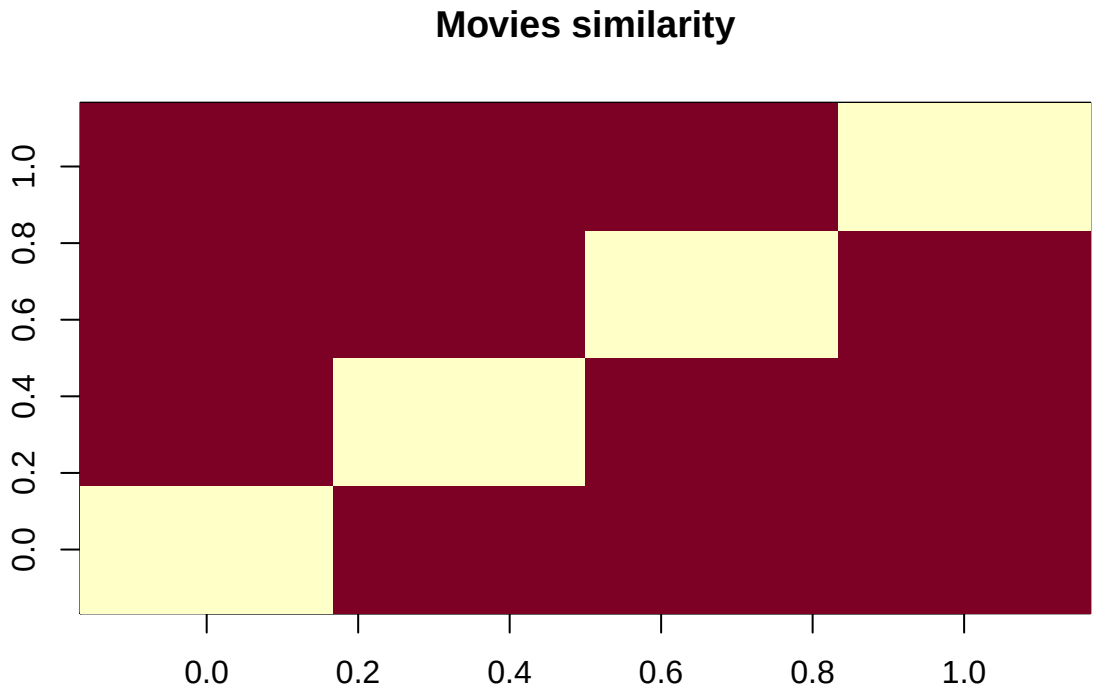


In the above matrix, each row and column represents a user. We have taken four users and each cell in this matrix represents the similarity that is shared between the two users.
Now, we delineate the similarity that is shared between the films

```
movie_similarity <- similarity(ratingMatrix[, 1:4], method =
                               "cosine", which = "items")
as.matrix(movie_similarity)
```

```
          1         2         3         4
1 0.0000000 0.9669732 0.9559341 0.9101276
2 0.9669732 0.0000000 0.9658757 0.9412416
3 0.9559341 0.9658757 0.0000000 0.9864877
4 0.9101276 0.9412416 0.9864877 0.0000000
```

```
image(as.matrix(movie_similarity), main = "Movies similarity")
```



Let us now extract the most unique ratings

```
rating_values <- as.vector(ratingMatrix@data)
unique(rating_values)
```

```
 [1] 0.0 5.0 4.0 3.0 4.5 1.5 2.0 3.5 1.0 2.5 0.5
```

Now, we will create a table of ratings that will display the most unique ratings.

```
Table_of_Ratings <- table(rating_values)
Table_of_Ratings
```

```
rating_values
      0     0.5       1     1.5       2     2.5       3     3.5       4     4.5
6791761    1198    3258    1567    7943    5484   21729   12237   28880    8187
      5
  14856
```

Most viewed movies visualization

In this section of the machine learning project, we will explore the most viewed movies in our dataset. We will first count the number of views in a film and then organize them in a table that would group them in descending order.

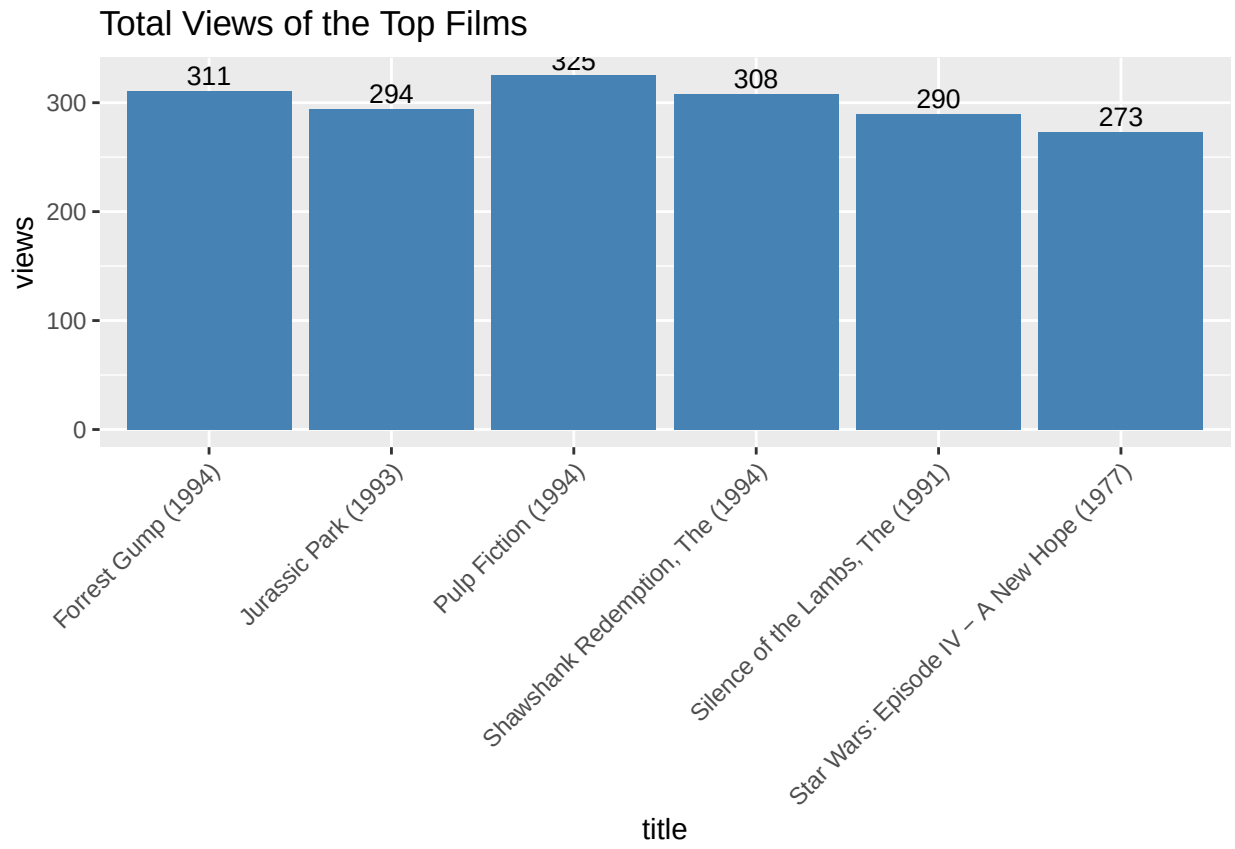
```
movie_views <- colCounts(ratingMatrix)
table_views <- data.frame(movie = names(movie_views), views = movie_views)
table_views <- table_views[order(table_views$views, decreasing = TRUE), ]
table_views$title <- NA
for (index in 1:10325){
  table_views[index,3] <- as.character(subset(movie_data,
                                        movie_data$movieId == table_views[index,1])$title)
}
table_views[1:6,]
```

```
    movie views                                       title
296   296   325                        Pulp Fiction (1994)
356   356   311                        Forrest Gump (1994)
318   318   308          Shawshank Redemption, The (1994)
480   480   294                      Jurassic Park (1993)
593   593   290          Silence of the Lambs, The (1991)
260   260   273 Star Wars: Episode IV - A New Hope (1977)
```

Now, we will visualize a bar plot for the total number of views of the top films. We will carry this out using *ggplot2*

```
ggplot(table_views[1:6, ], aes(x = title, y = views)) +
  geom_bar(stat="identity", fill = 'steelblue') +
  geom_text(aes(label=views), vjust=-0.3, size=3.5) +
  theme(axis.text.x = element_text(angle = 45, hjust = 1)) +

  ggtitle("Total Views of the Top Films")
```

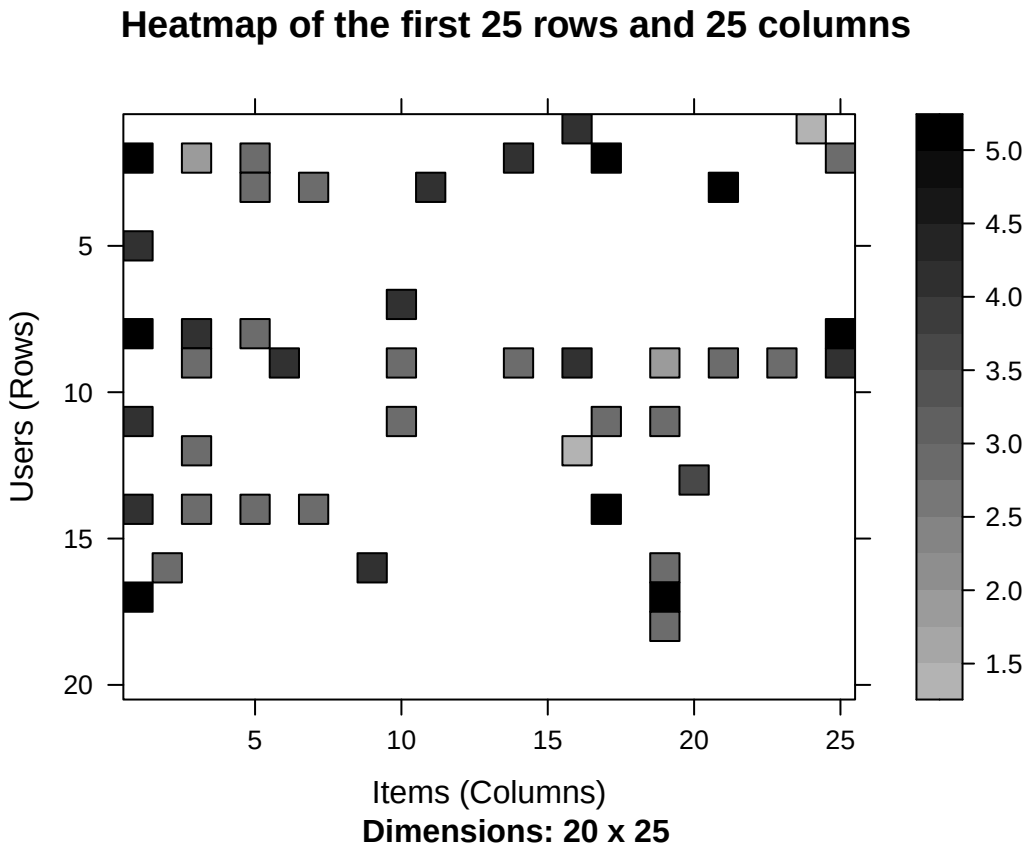


From the above bar-plot, we observe that Pulp Fiction is the most-watched film followed by Forrest Gump.

Heatmap of movie ratings

Now, we shall visualize a heatmap of the movie ratings. This heatmap will contain first 25 rows.

```
image(ratingMatrix[1:20, 1:25], axes = FALSE, main = "Heatmap of the first 25 rows and 25 columns")
```



Data preparation

We shall conduct the data preparation in the following 3 steps.
- Selecting useful data
- Normalizing data
- Binarizing the data

Finding useful data

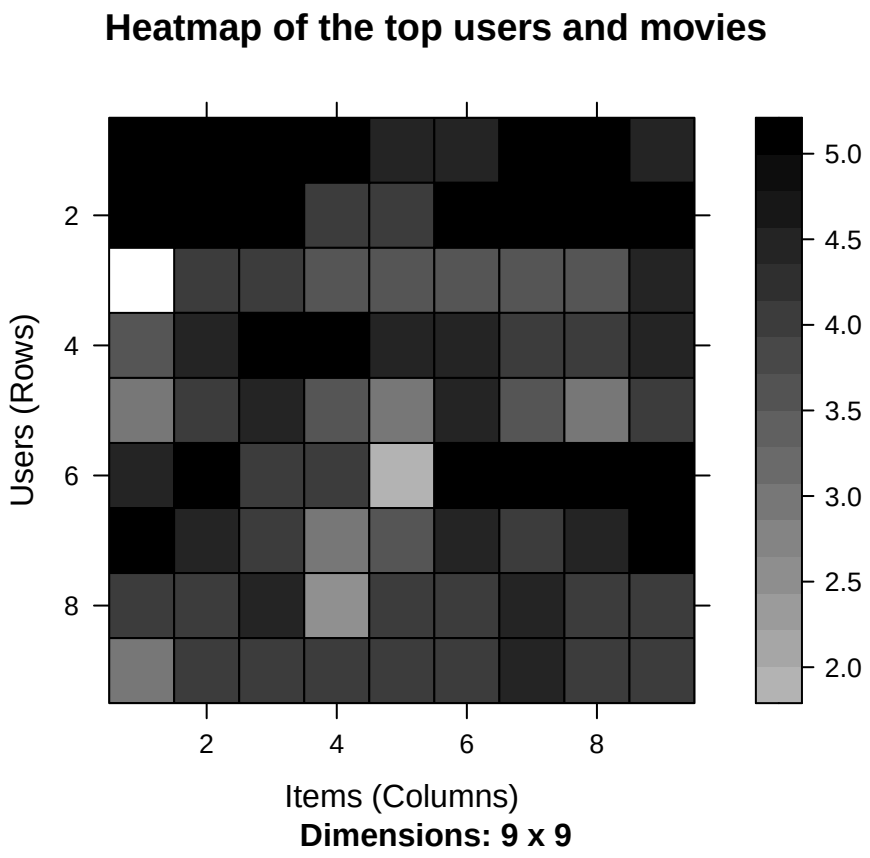
For finding useful data in our dataset, we have set the threshold for the minimum number of users who have rated a film as 50. This is also same for minimum number of views that are per film. This way, we have filtered a list of watched films from least-watched ones.

```
movie_ratings <- ratingMatrix[rowCounts(ratingMatrix) > 50,
                              colCounts(ratingMatrix) > 50]
movie_ratings
```

```
420 x 447 rating matrix of class 'realRatingMatrix' with 38341 ratings.
```

From the above output of 'movie_ratings', we observe that there are 420 users and 447 films as opposed to the previous 668 users and 10325 films. We can now delineate our matrix of relevant users as follows:

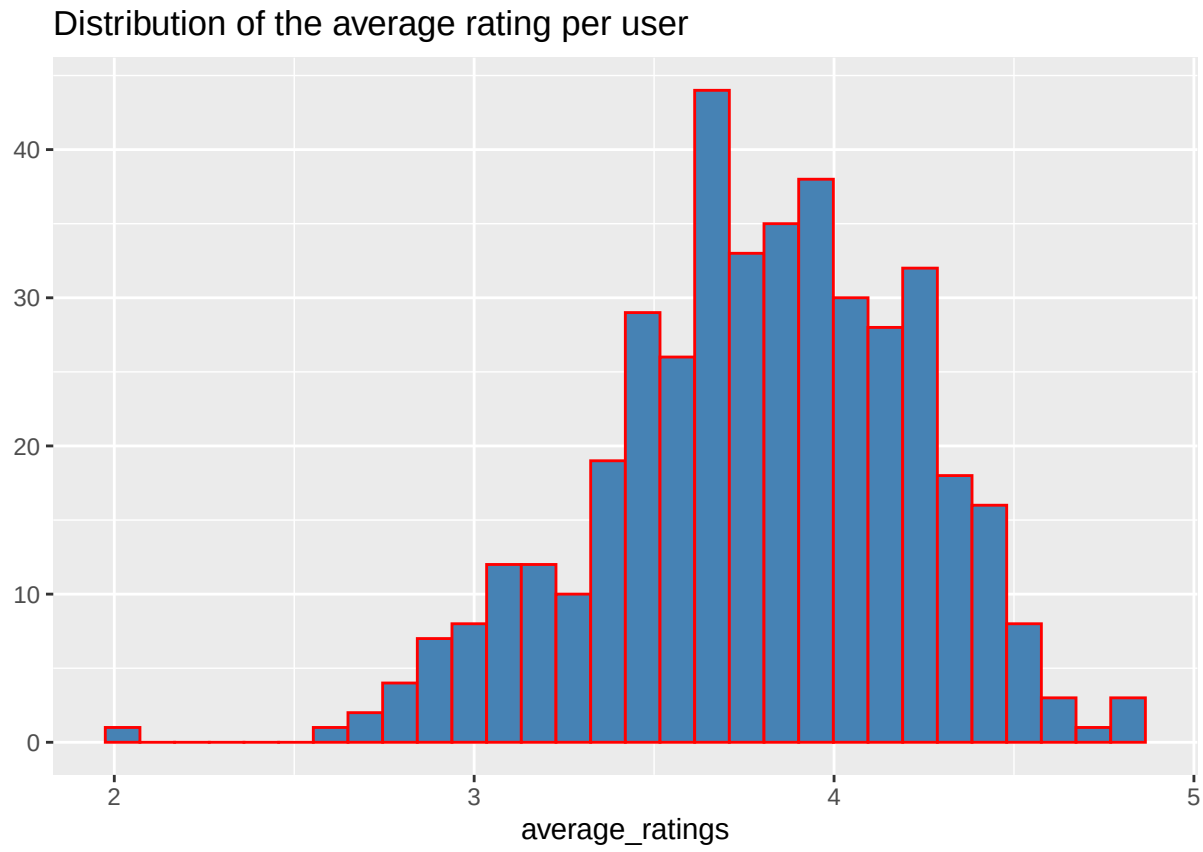
```
minimum_movies<- quantile(rowCounts(movie_ratings), 0.98)
minimum_users <- quantile(colCounts(movie_ratings), 0.98)
image(movie_ratings[rowCounts(movie_ratings) > minimum_movies,
                    colCounts(movie_ratings) > minimum_users],
main = "Heatmap of the top users and movies")
```



Now, we will visualize the distribution of the average ratings per user.

```
average_ratings <- rowMeans(movie_ratings)
qplot(average_ratings, fill=I("steelblue"), col=I("red")) +
  ggtitle("Distribution of the average rating per user")
```

```
`stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



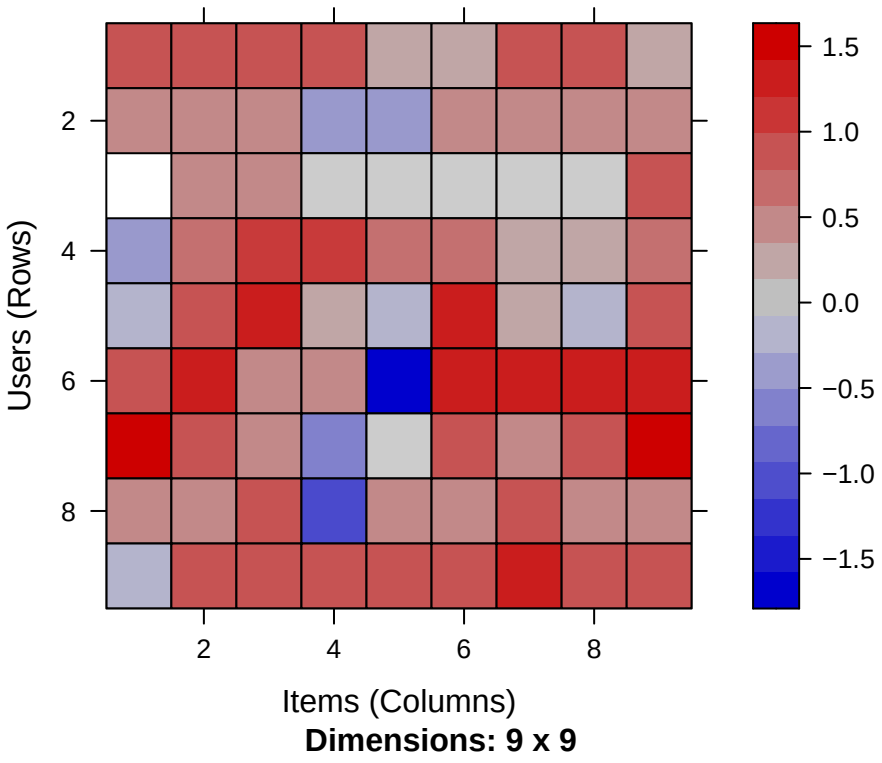
Data normalization

In the case of some users, there can be high ratings or low ratings provided to all of the watched films. This will act as a bias while implementing our model. In order to remove this, we normalize our data. Normalization is a data preparation procedure to standardize the numerical values in a column to a common scale value. This is done in such a way that there is no distortion in the range of values. Normalization transforms the average value of our ratings column to 0. We then plot a heatmap that delineates our normalized ratings.

```
normalized_ratings <- normalize(movie_ratings)
sum(rowMeans(normalized_ratings) > 0.00001)
```

```
[1] 0
```

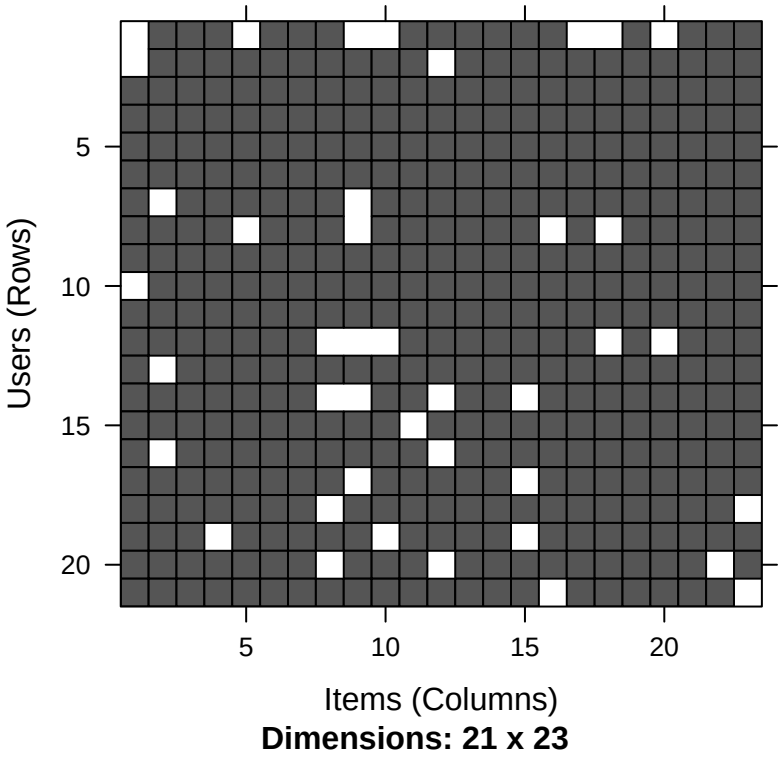
```
image(normalized_ratings[rowCounts(normalized_ratings) > minimum_movies,
                         colCounts(normalized_ratings) > minimum_users],
main = "Normalized Ratings of the Top Users")
```



Data binarization

In the final step of our data preparation, we will binarize our data. Binarizing the data means that we have two discrete values 1 and 0, which will allow our recommendation systems to work more efficiently. We will define a matrix that will consist of 1 if the rating is above 3 and otherwise it will be 0.

```
binary_minimum_movies <- quantile(rowCounts(movie_ratings), 0.95)
binary_minimum_users <- quantile(colCounts(movie_ratings), 0.95)
good_rated_films <- binarize(movie_ratings, minRating = 3)
image(good_rated_films[rowCounts(movie_ratings) > binary_minimum_movies,
colCounts(movie_ratings) > binary_minimum_users],
main = "Heatmap of the top users and movies")
```



Collaborative filtering system

Next we shall develop *Item Based Collaborative Filtering* system. This type of collaborative filtering finds similarity in the items based on the people's ratings of them. The algorithm first builds a similar-items table of the customers who have purchased them into a combination of similar items. This is then fed into the recommendation system.
We will build this filtering system by splitting the dataset into 80% training set and 20% test set.

```
sampled_data<- sample(x = c(TRUE, FALSE),
                      size = nrow(movie_ratings),
                      replace = TRUE,
                      prob = c(0.8, 0.2))
training_data <- movie_ratings[sampled_data, ]
testing_data <- movie_ratings[!sampled_data, ]
```

Building recommendation system

We will now explore the various parameters of our Item Based Collaborative Filter. These parameters are default in nature. In the first step, k denotes the number of items for computing their similarities. Here, k is equal to 30. Therefore, the algorithm will now identify the k most similar items and store their number. We use the cosine method which is the default one but, one can also use pearson method.

```
recommendation_system <- recommenderRegistry$get_entries(dataType ="realRatingMatrix")
recommendation_system$IBCF_realRatingMatrix$parameters
```

```
$k
[1] 30

$method
[1] "Cosine"

$normalize
[1] "center"

$normalize_sim_matrix
[1] FALSE

$alpha
[1] 0.5

$na_as_zero
[1] FALSE
```

```
recommen_model <- Recommender(data = training_data,
                              method = "IBCF",
                              parameter = list(k = 30))
recommen_model
```

```
Recommender of type 'IBCF' for 'realRatingMatrix'
learned using 358 users.
```

```
class(recommen_model)
```

```
[1] "Recommender"
attr(,"package")
[1] "recommenderlab"
```

Let us now explore our recommendation system model as follows:
Using the *getModel()* function, we will retrieve the *recommen_model*. We will then find the class and dimensions of our similarity matrix that is contained within *model_info*. Finally, we will generate a heatmap, that will contain the top 20 items and visualize the similarity shared between them.

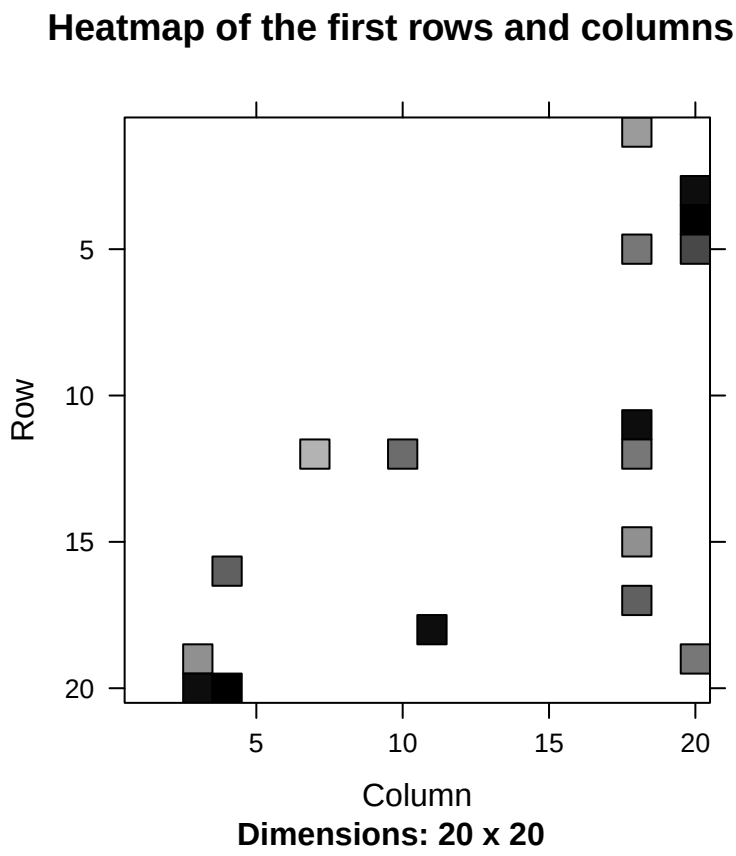
```
model_info <- getModel(recommen_model)
class(model_info$sim)
```

```
[1] "dgCMatrix"
attr(,"package")
[1] "Matrix"
```

```
dim(model_info$sim)
```

```
[1] 447 447
```

```
top_items <- 20
image(model_info$sim[1:top_items, 1:top_items],
   main = "Heatmap of the first rows and columns")
```



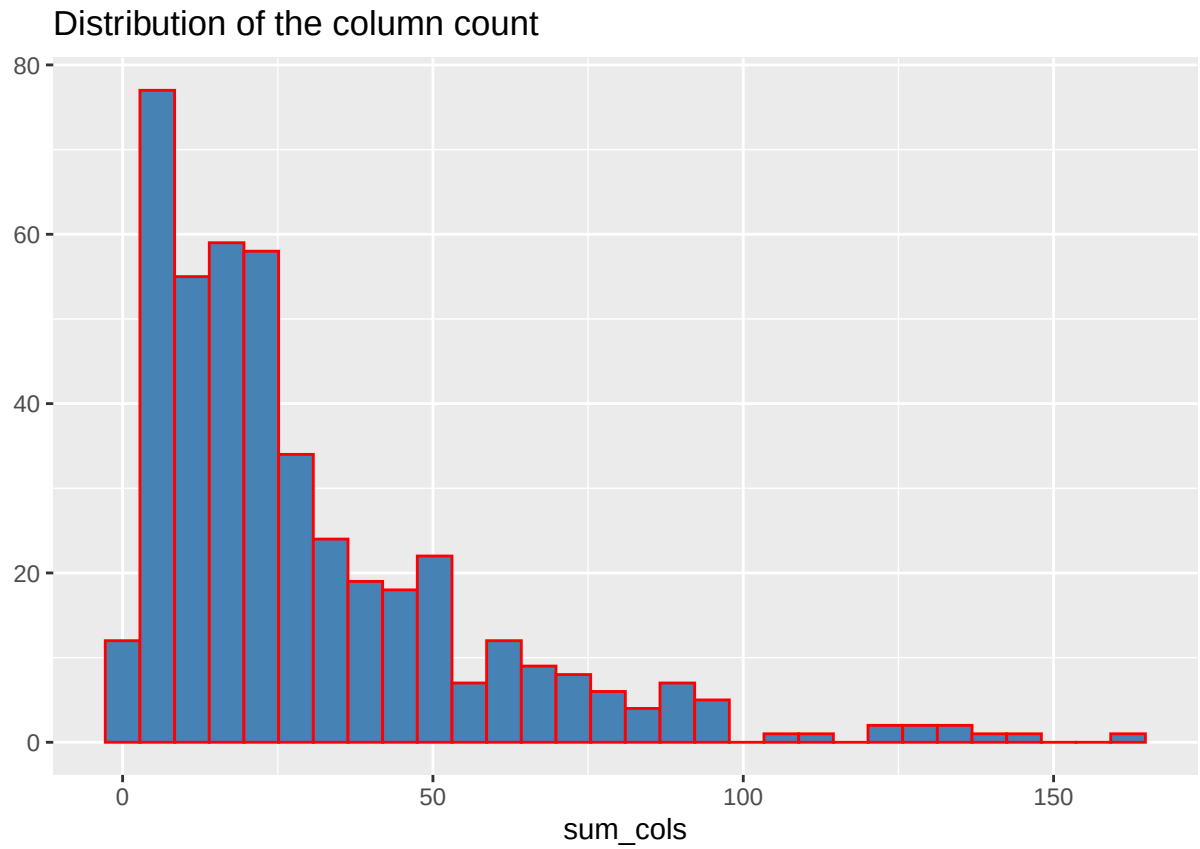
Next, we shall carry out the sum of rows and columns with the similarity of the objects above 0. We will visualize the sum of columns through a distribution as follows:

```
sum_rows <- rowSums(model_info$sim > 0)
table(sum_rows)
```

```
sum_rows
 30
447
```

```
sum_cols <- colSums(model_info$sim > 0)
qplot(sum_cols, fill=I("steelblue"), col=I("red"))+ ggtitle("Distribution of the column count")
```

```
`stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



We will create a *top_recommendations* variable which will be initialized to 10, specifying the number of films to each user. We will then use the *predict()* function that will identify similar items and will rank them appropriately. Here, each rating is used as a weight. Each weight is multiplied with related similarities. Finally, everything is added in the end.

```
top_recommendations <- 10
predicted_recommendations <- predict(object = recommen_model,
                          newdata = testing_data,
                          n = top_recommendations)
predicted_recommendations
```

```
Recommendations as 'topNList' with n = 10 for 62 users.
```

```
user1 <- predicted_recommendations@items[[1]]
movies_user1 <- predicted_recommendations@itemLabels[user1]
movies_user2 <- movies_user1
for (index in 1:10){
  movies_user2[index] <- as.character(subset(movie_data,
                                        movie_data$movieId == movies_user1[index])$title)
}
movies_user2
```

```
 [1] "GoldenEye (1995)"
 [2] "Twelve Monkeys (a.k.a. 12 Monkeys) (1995)"
```

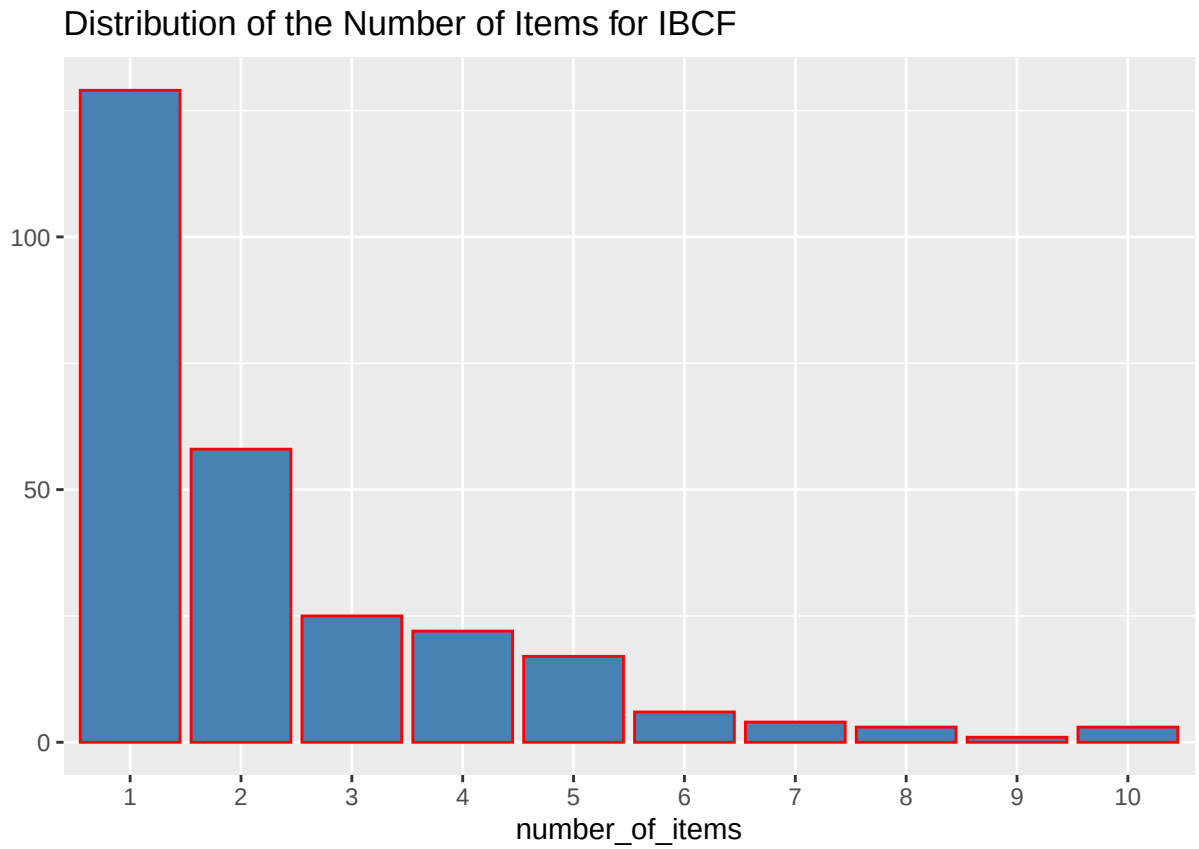
```
 [3] "Birdcage, The (1996)"
 [4] "Rob Roy (1995)"
 [5] "Casper (1995)"
 [6] "Crimson Tide (1995)"
 [7] "Judge Dredd (1995)"
 [8] "Dumb & Dumber (Dumb and Dumber) (1994)"
 [9] "French Kiss (1995)"
[10] "Legends of the Fall (1994)"
```

```
recommendation_matrix <- sapply(predicted_recommendations@items,
                      function(x){ as.integer(colnames(movie_ratings)[x]) })
recommendation_matrix[,1:4]
```

```
      [,1]  [,2] [,3]  [,4]
 [1,]   10 48780  969  1641
 [2,]   32  2542 2947  5816
 [3,]  141  3421 3671 54286
 [4,]  151  8961 3753   410
 [5,]  158  1275 2324  1356
 [6,]  161   594  594   161
 [7,]  173 54286 4720   551
 [8,]  231   353 2701   733
 [9,]  236   784 7147   780
[10,]  266 70286 2599  1047
```

This also allows us to see how many items are matching to each recommendation in our model and also we can visualizing which of the movies match most of the things in the sample we are working with and how many items they match with based on rating and genre.

```
number_of_items <- factor(table(recommendation_matrix))
chart_title <- "Distribution of the Number of Items for IBCF"
qplot(number_of_items, fill=I("steelblue"), col=I("red")) + ggtitle(chart_title)
```




```
number_of_items_sorted <- sort(number_of_items, decreasing = TRUE)
number_of_items_top <- head(number_of_items_sorted, n = 4)
table_top <- data.frame(as.integer(names(number_of_items_top)), number_of_items_top)
for(i in 1:4) {
  table_top[i,1] <- as.character(subset(movie_data,movie_data$movieId == table_top[i,1])$title)
}
colnames(table_top) <- c("Movie Title", "No. of Items")
head(table_top)
```

```
                Movie Title No. of Items
3   Grumpier Old Men (1995)           10
7            Sabrina (1995)           10
39          Clueless (1995)           10
168     First Knight (1995)            9
```

Summary

Recommendation Systems are the most popular type of machine learning applications; that are used in all sectors. They are an improvement over the traditional classification algorithms as they can take many classes of input and provide similarity ranking based algorithms to provide the user with accurate results. These recommendation systems have evolved over time and have incorporated many advanced machine learning techniques to provide the users with the content that they want.

To recreate any of the above performed operations here; the list of all packages and environment conditions are given below.

```
sessionInfo()
```

```
R version 4.0.3 (2020-10-10)
Platform: x86_64-w64-mingw32/x64 (64-bit)
Running under: Windows 10 x64 (build 19042)

Matrix products: default

locale:
[1] LC_COLLATE=English_India.1252  LC_CTYPE=English_India.1252
[3] LC_MONETARY=English_India.1252 LC_NUMERIC=C
[5] LC_TIME=English_India.1252

attached base packages:
[1] stats     graphics  grDevices utils     datasets  methods   base

other attached packages:
[1] reshape2_1.4.4       data.table_1.13.2    ggplot2_3.3.2
[4] recommenderlab_0.2-6 registry_0.5-1       proxy_0.4-24
[7] arules_1.6-6         Matrix_1.2-18

loaded via a namespace (and not attached):
 [1] Rcpp_1.0.5       plyr_1.8.6       compiler_4.0.3   pillar_1.4.6
 [5] tools_4.0.3      digest_0.6.26    evaluate_0.14    lifecycle_0.2.0
 [9] tibble_3.0.4     gtable_0.3.0     lattice_0.20-41  pkgconfig_2.0.3
[13] rlang_0.4.8      yaml_2.2.1       xfun_0.19        withr_2.3.0
[17] stringr_1.4.0    dplyr_1.0.2      knitr_1.30       generics_0.1.0
[21] vctrs_0.3.4      grid_4.0.3       tidyselect_1.1.0 glue_1.4.2
[25] R6_2.4.1         recosystem_0.4.4 rmarkdown_2.6    irlba_2.3.3
[29] farver_2.0.3     purrr_0.3.4      magrittr_1.5     scales_1.1.1
[33] htmltools_0.5.0  ellipsis_0.3.1   colorspace_1.4-1 labeling_0.4.2
[37] stringi_1.5.3    munsell_0.5.0    crayon_1.3.4
```