MINI PROJECT

ON

Project Entitled: **“ Movie Recommendations By Data Analysis using R”**

Submitted by

1. Kunal Temkar

ABSTRACT

A recommendation system is a system that provides suggestions to users for certain resources like books, movies, songs, etc., based on some data set. Movie recommendation systems usually predict what movies a user will like based on the attributes present in previously liked movies. Such recommendation systems are beneficial for organizations that collect data from large amounts of customers, and wish to effectively provide the best suggestions possible. A lot of factors can be considered while designing a movie recommendation system like the genre of the movie, actors present in it or even the director of the movie. The systems can recommend movies based on one or a combination of two or more attributes. In this project, the recommendation system has been built on the type of genres that the user might prefer to watch. The approach adopted to do so is item-based collaborative filtering.The data analysis tool used is R.

Introduction

Now-a-days, there are various popular online streaming platform like Netflix, Amazon Prime, etc. So when a person watches a movie and after some time, that platform starts recommending us different movies and TV shows. Due to this, a question might arise that how the movie streaming platform could suggest us content that appealed to us. In today's age, a majority of organizations implement recommendation systems for fulfilling customer requirements. LinkedIn, Amazon, and Netflix are just a few to name as mentioned above. LinkedIn recommends relevant connections of the people the user might know among the millions that are subscribed on the portal. This way, the user does not have to run extensive searches for people manually. Amazon recommendation systems work such that they suggest correlated items that the customers can purchase. If a certain customer prefers buying books from the shopping portal, Amazon provides suggestions related to any new arrivals in previously preferred categories. In a very similar way, Netflix takes into account the types of shows that a customer watches, and provides recommendations similar to those.

Collaborative filtering analyses the user's previous experiences and ratings and correlates it with other users. Based on the ones that have the most similarity, recommendations are made.

Libraries used.

In this project, I have used these four packages – *'recommenderlab', 'ggplot2', 'data.table'* and *'reshape2.*

```
> 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
```

Fig 1.1 Library(recommender lab)

```
library(ggplot2)                         #Author DataFlair
```

```
## Registered S3 methods overwritten by 'ggplot2':
##   method         from
##   [.quosures     rlang
##   c.quosures     rlang
##   print.quosures rlang
```

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

```
##
## Attaching package: 'reshape2'
```

```
## The following objects are masked from 'package:data.table':
##
##     dcast, melt
```

Fig 1.2 library(ggplot2, data.table, reshape2)

DataSets:

1	movieId	title	title
2	1	Toy Story (1995)	Adventure|Animation|Children|Comedy|Fantasy
3	2	Jumanji (1995)	Adventure|Children|Fantasy
4	3	Grumpier Old Men (1995)	Comedy|Romance
5	4	Waiting to Exhale (1995)	Comedy|Drama|Romance
6	5	Father of the Bride Part II (199	Comedy
7	6	Heat (1995)	Action|Crime|Thriller
8	7	Sabrina (1995)	Comedy|Romance
9	8	Tom and Huck (1995)	Adventure|Children
10	9	Sudden Death (1995)	Action
11	10	GoldenEye (1995)	Action|Adventure|Thriller
12	11	American President, The (1995	Comedy|Drama|Romance
13	12	Dracula: Dead and Loving It (1	Comedy|Horror
14	13	Balto (1995)	Adventure|Animation|Children
15	14	Nixon (1995)	Drama
16	15	Cutthroat Island (1995)	Action|Adventure|Romance
17	16	Casino (1995)	Crime|Drama
18	17	Sense and Sensibility (1995)	Drama|Romance
19	18	Four Rooms (1995)	Comedy
20	19	Ace Ventura: When Nature Ca	Comedy
21	20	Money Train (1995)	Action|Comedy|Crime|Drama|Thriller
22	21	Get Shorty (1995)	Comedy|Crime|Thriller
23	22	Copycat (1995)	Crime|Drama|Horror|Mystery|Thriller
24	23	Assassins (1995)	Action|Crime|Thriller
25	24	Powder (1995)	Drama|Sci-Fi
26	25	Leaving Las Vegas (1995)	Drama|Romance
27	26	Othello (1995)	Drama
28	27	Now and Then (1995)	Children|Drama
29	28	Persuasion (1995)	Drama|Romance

Fig 1.3 Dataset “movies.csv”

1	userId	movieId	rating	timestamp
2	1	16	4	1.22E+09
3	1	24	1.5	1.22E+09
4	1	32	4	1.22E+09
5	1	47	4	1.22E+09
6	1	50	4	1.22E+09
7	1	110	4	1.22E+09
8	1	150	3	1.22E+09
9	1	161	4	1.22E+09
10	1	165	3	1.22E+09
11	1	204	0.5	1.22E+09
12	1	223	4	1.22E+09
13	1	256	0.5	1.22E+09
14	1	260	4.5	1.22E+09
15	1	261	1.5	1.22E+09
16	1	277	0.5	1.22E+09
17	1	296	4	1.22E+09
18	1	318	4	1.22E+09
19	1	349	4.5	1.22E+09
20	1	356	3	1.22E+09
21	1	377	2.5	1.22E+09
22	1	380	3	1.22E+09
23	1	457	4	1.22E+09
24	1	480	3.5	1.22E+09
25	1	527	4.5	1.22E+09
26	1	589	3.5	1.22E+09
27	1	590	3.5	1.22E+09
28	1	592	2.5	1.22E+09
29	1	593	5	1.22E+09

Fig 1.4 Dataset “ratings.csv”

Retrieving the Data and Exploring it:

Here, I had retrieved the data from movies.csv into movies_data dataframe and ratings.csv into rating_data. I had used the str() function to display information about the movies_data dataframe.

```
> movies_data<-read.csv("kunal/movies.csv",stringsAsFactors = FALSE)
> rating_data<-read.csv("kunal/ratings.csv")
> str(movies_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|Romance" "Comedy|Drama|R
omance" ...
> summary(movies_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(movies_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
```

Fig 1.4 summary(movies_data)

Here is the overview of the summary of the movies using the summary() function. Also, I had used the head() function to print the first six lines of movie_data.

```
> 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
```

Fig 1.5 summary(ratings_data)

Similarly, this is the summary as well as the first six lines of the 'rating_data' dataframe –

Data Pre-processing

From fig 1.5, we observe that the userId column, as well as the movieId column, consist of integers. Furthermore, it is needed to convert the genres present in the movies_data dataframe into a more usable format by the users. In order to do so, it is required to create a one-hot encoding to create a matrix that comprises of corresponding genres for each of the films.

Code:

```r
movie_genre <- as.data.frame(movies_data$genres, stringsAsFactors=FALSE)
library(data.table)
movie_genre2 <- as.data.frame(tstrsplit(movie_genre[,1], '[|]',
                                        type.convert=TRUE),
                              stringsAsFactors=FALSE) #DataFlair
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]) #Author DataFlair
    genre_mat1[index+1,gen_col] <- 1
  }
}
genre_mat2 <- as.data.frame(genre_mat1[-1,], stringsAsFactors=FALSE) #remove first row, which was the genre list
for (col in 1:ncol(genre_mat2)) {
  genre_mat2[,col] <- as.integer(genre_mat2[,col]) #convert from characters to integers
}
str(genre_mat2)
```

Fig 1.6 Code to create matrix

Output:

```
> source('~/.active-rstudio-document')
'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 ...
```

Further, a 'search matrix' is created that will allow us to perform an easy search of the films by specifying the genre present in our list.

Code & output:

```
> SearchMatrix <- cbind(movies_data[,1:2], genre_mat2[])
> head(SearchMatrix)
  movieId                              title Action Adventure Animation Children Comedy Crime Documentary Drama Fantasy
1       1                   Toy Story (1995)      0         1         1        1      1     0           0     0       1
2       2                     Jumanji (1995)      0         1         0        1      0     0           0     0       1
3       3            Grumpier Old Men (1995)      0         0         0        0      1     0           0     0       0
4       4           Waiting to Exhale (1995)      0         0         0        0      1     0           0     1       0
5       5 Father of the Bride Part II (1995)      0         0         0        0      1     0           0     0       0
6       6                        Heat (1995)      1         0         0        0      0     1           0     0       0
  Film-Noir Horror Musical Mystery Romance Sci-Fi Thriller War Western
1         0      0       0       0       0      0        0   0       0
2         0      0       0       0       0      0        0   0       0
3         0      0       0       0       1      0        0   0       0
4         0      0       0       0       1      0        0   0       0
5         0      0       0       0       0      0        0   0       0
6         0      0       0       0       0      0        1   0       0
> 
```

Fig. 1.7 SearchMatrix

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 the 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=F
ALSE)
ratingMatrix <- as.matrix(ratingMatrix[,-1]) #remove userIds
#Convert rating matrix into a recommenderlab sparse matrix
ratingMatrix <- as(ratingMatrix, "realRatingMatrix")
ratingMatrix
```

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

Fig. 1.8 ratingMatrix to sparseMatrix

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")
> 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")
> |
```

Fig 1.9 similarityMatrix

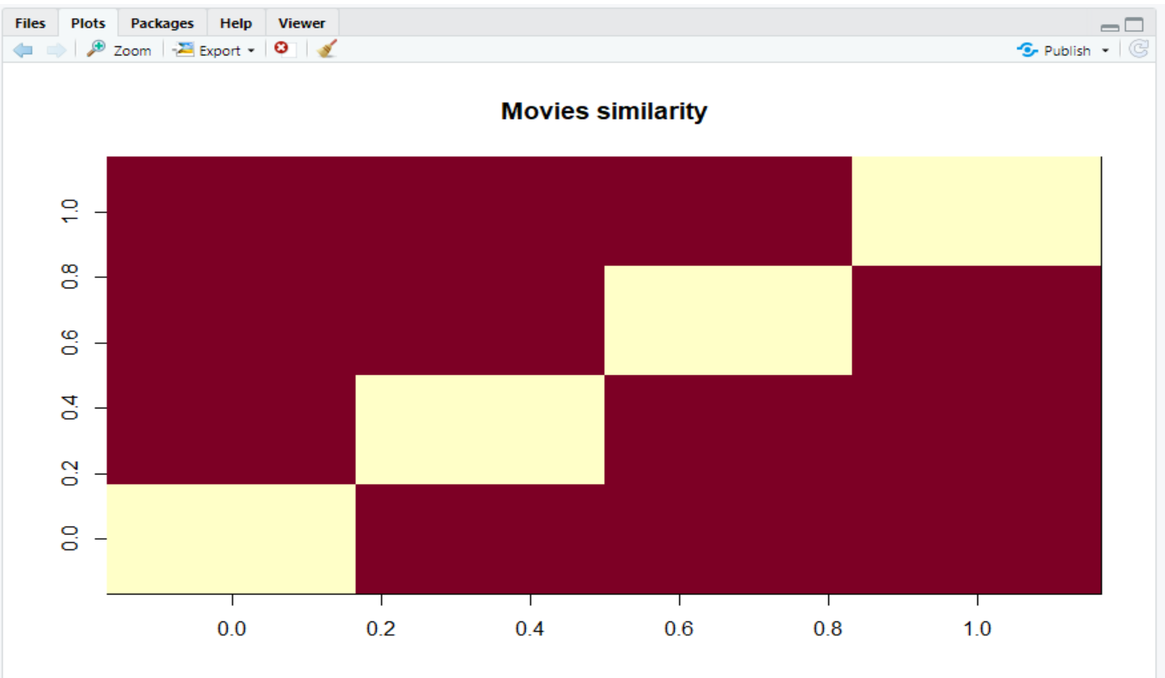

Fig 1.10 Graph

Most Viewed Movies Visualization

In this section, the exploration of the most viewed movies in our dataset is done. Before that, the number of views in a film is counted and then organized them in a table that would group them in descending order.

Code:

```
Table_of_Ratings <- table(rating_values) # creating a count of movie ratings
Table_of_Ratings



library(ggplot2)
movie_views <- colCounts(ratingMatrix) # count views for each movie
table_views <- data.frame(movie = names(movie_views),
                          views = movie_views) # create dataframe of views
table_views <- table_views[order(table_views$views,
                                 decreasing = TRUE), ] # sort by number of views
table_views$title <- NA
for (index in 1:10325){
  table_views[index,3] <- as.character(subset(movies_data,
                                        movies_data$movieId == table_views[index,1])$title)
}
table_views[1:6,]



```

Output:

```
> 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)
> 
```

Fig 2.1 No.Of moviesView

Now,visualization of a bar plot for the total number of views of the top films is shown here using ggplot2.

Code:

```
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")
```

Output:

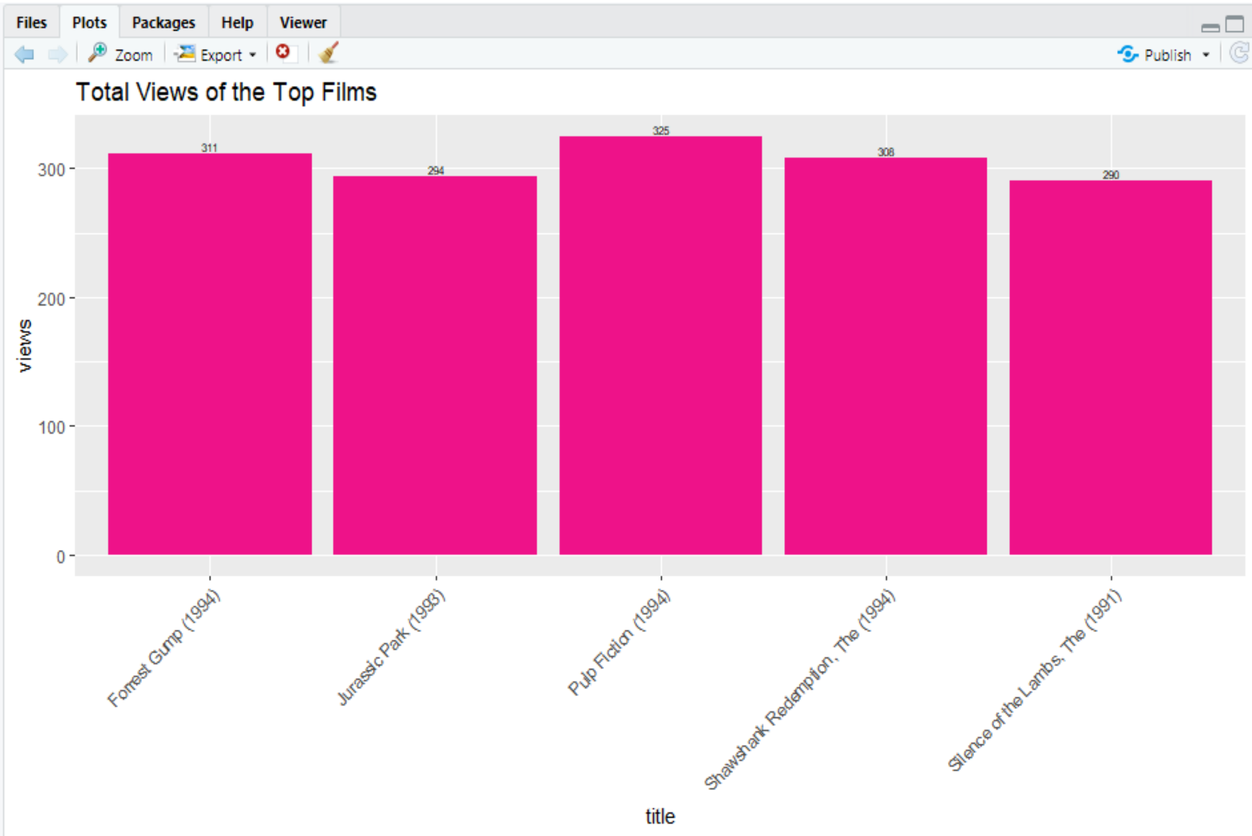

Fig 2.3 Graph (Top viewed films)

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

Heatmap of Movie Ratings

Now, in this, visualization of a heatmap of the movie ratings is done. This heatmap will contain first 25 rows and 25 columns as follows –

Code:

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



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



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")
```

Output:

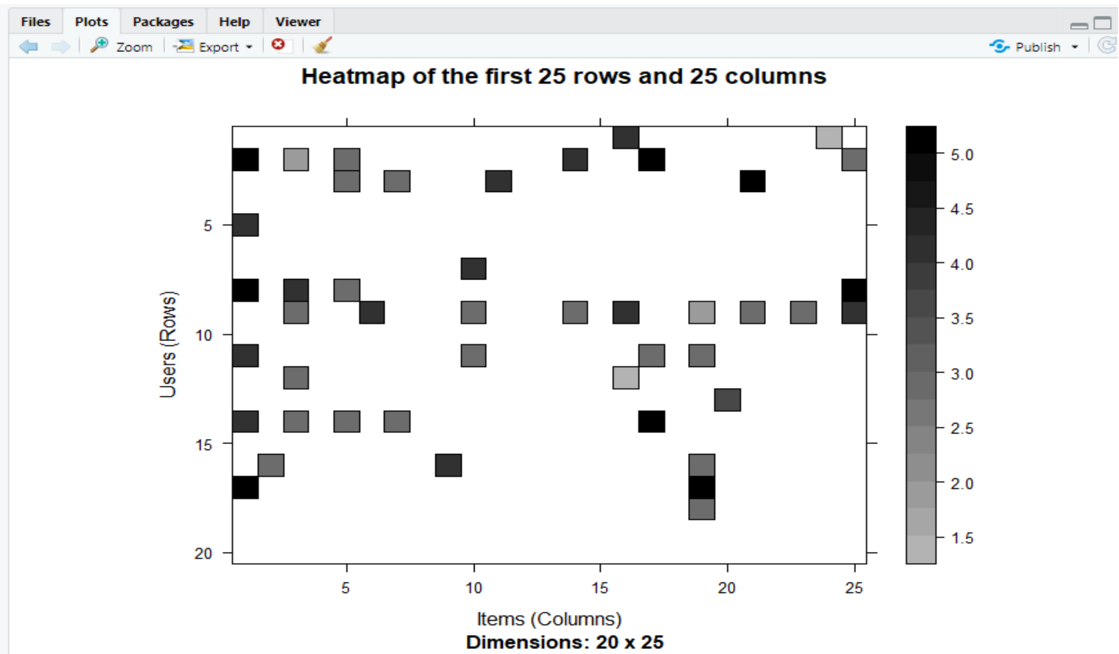

Fig 2.4 HeatMap_1

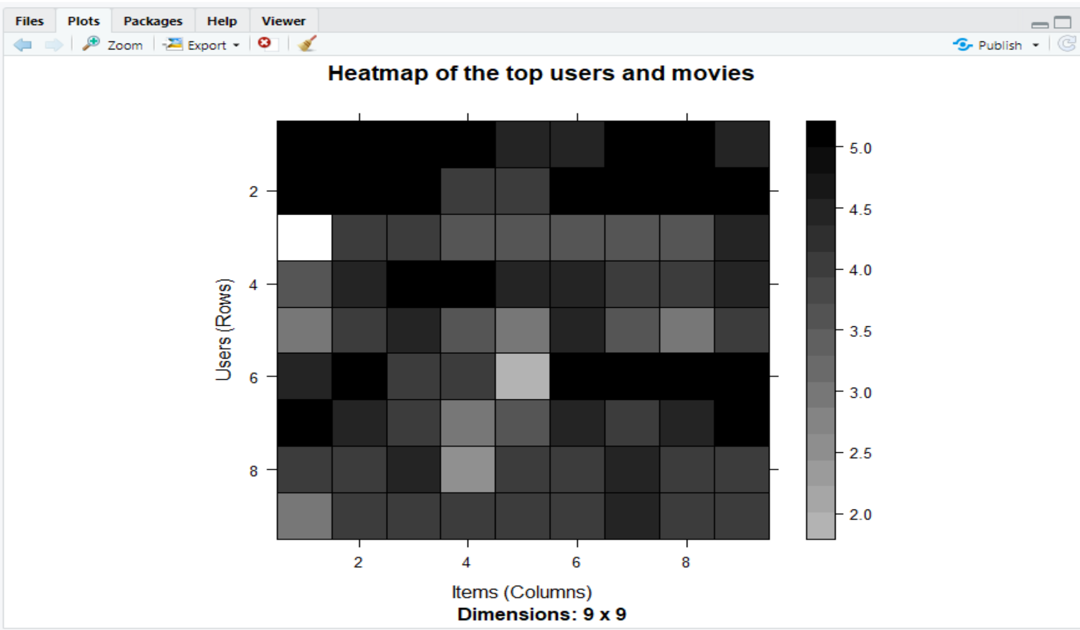

Fig 2.5 HeatMap_2

Visualization of Distribution of the average ratings per user.

Code:

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

Output:

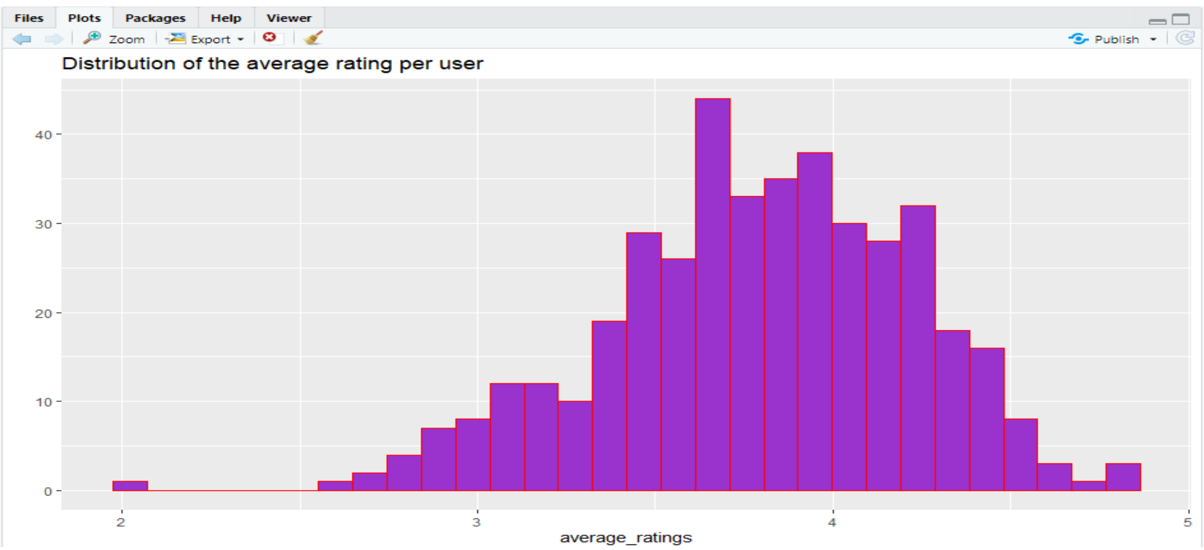

Fig 2.6 (Avg_ratings per user)

Collaborative Filtering System

In this section, Item Based Collaborative Filtering System is developed. 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.

The similarity between single products and related products can be determined with the following algorithm –

- For each Item i1 present in the product catalog, purchased by customer C.
- And, for each item i2 also purchased by the customer C.
- Create record that the customer purchased items i1 and i2.
- Calculate the similarity between i1 and i2.

We will build this filtering system by splitting the dataset into 80% training set and 20% test set.

Code:

```
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, ]
```

Recommender System on dataset

Code:

```
top_recommendations <- 10 # the number of items to recommend to each user
predicted_recommendations <- predict(object = recommen_model,
newdata = testing_data,
n = top_recommendations)
predicted_recommendations
```

Output:

```
> top_recommendations <- 10 # the number of items to recommend to each user
> predicted_recommendations <- predict(object = recommen_model,
+                                      newdata = testing_data,
+                                      n = top_recommendations)
> predicted_recommendations
Recommendations as 'topNList' with n = 10 for 94 users.
> 
```

Fig. 2.7

Code:

```
user1 <- predicted_recommendations@items[[1]] # recommendation for the first user
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
```

Output:

```
##  [1] "Broken Arrow (1996)"
##  [2] "Species (1995)"
##  [3] "Mask, The (1994)"
##  [4] "Executive Decision (1996)"
##  [5] "Annie Hall (1977)"
##  [6] "Little Miss Sunshine (2006)"
##  [7] "Pan's Labyrinth (Laberinto del fauno, El) (2006)"
##  [8] "Hangover, The (2009)"
##  [9] "Mrs. Doubtfire (1993)"
## [10] "Leaving Las Vegas (1995)"
```

Fig 2.8

Conclusion:

The recommendation system implemented in this paper aims at providing movie recommendation based on the genres of the movies. If a user highly rates a movie of a particular genre, movies containing similar genres will be recommended to him. Recommendation systems are widely used in today’s era of Web 2.0 for searching for reliable and relevant information.