Movie Recommendation project

Presented by:-

Valiveti Manikanta Bhuvanesh-19BCD7088

Tarini Guttula -19BCE7758

Popuri Harsha Vardhan -19BCI7039

G.Balasubramanyam -19BCE7681

Rokkam Anirudh -19BCD7081

Guided by:-

Dr.GopiKrishnan

VIT-AP UNIVERSITY

ABSTRACT

- If you're a fan of Amazon, Amazon Prime, or Netflix, you're presumably aware that these services use "recommendation engines." As the name implies, the main aim of a recommendation engine is to "propose" relevant things to customers — while Amazon recommends merchandise, Prime and Netflix recommend material to users based on their previous purchase or watch history. This R project's main purpose is to create a recommendation system that will recommend movies to users. We will develop an Item Based Collaborative Filter in this project.

INTRODUCTION

ABOUT DATASET

- The dataset used for this project is MovieLens dataset. This data includes 105339 ratings for over 10329 movies.

DATASET

movieId	title	genres
1	Toy Story (1995)	Adventure|Animation|Children|Comedy|Fantasy
2	Jumanji (1995)	Adventure|Children|Fantasy
3	Grumpier Old Men (1995)	Comedy|Romance
4	Waiting to Exhale (1995)	Comedy|Drama|Romance
5	Father of the Bride Part II	Comedy
6	Heat (1995)	Action|Crime|Thriller
7	Sabrina (1995)	Comedy|Romance
8	Tom and Huck (1995)	Adventure|Children
9	Sudden Death (1995)	Action
10	GoldenEye (1995)	Action|Adventure|Thriller
11	American President, The (1	Comedy|Drama|Romance
12	Dracula: Dead and Loving	Comedy|Horror
13	Balto (1995)	Adventure|Animation|Children
14	Nixon (1995)	Drama
15	Cutthroat Island (1995)	Action|Adventure|Romance

userId	movieId	rating	timestamp
1	16	4	1.22E+09
1	24	1.5	1.22E+09
1	32	4	1.22E+09
1	47	4	1.22E+09
1	50	4	1.22E+09
1	110	4	1.22E+09
1	150	3	1.22E+09
1	161	4	1.22E+09
1	165	3	1.22E+09
1	204	0.5	1.22E+09
1	223	4	1.22E+09
1	256	0.5	1.22E+09
1	260	4.5	1.22E+09
1	261	1.5	1.22E+09
1	277	0.5	1.22E+09
1	296	4	1.22E+09
1	318	4	1.22E+09
1	349	4.5	1.22E+09
1	356	3	1.22E+09

PROCEDURE

Step1:Loading dataset

- We import datasets containing movies and ratings

```
> movie_data <- read.csv("movies.csv",stringsAsFactors=FALSE)
> rating_data <- read.csv("ratings.csv")
> head(movie_data)
  movieId                               title                                        genres
1       1                    Toy Story (1995) Adventure|Animation|Children|Comedy|Fantasy
2       2                      Jumanji (1995)                  Adventure|Children|Fantasy
3       3             Grumpier Old Men (1995)                              Comedy|Romance
4       4            Waiting to Exhale (1995)                        Comedy|Drama|Romance
5       5 Father of the Bride Part II (1995)                                      Comedy
6       6                           Heat (1995)                     Action|Crime|Thriller
> 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
>
```

Step 2 :Data Pre-Processing

- We can see that the userId and movieId columns are both made up of numbers. Furthermore, we must change the genres included in the movie data dataframe into a more user-friendly format. To do so, we will first develop a one-hot encoding to generate a matrix of associated genres for each of the films. We constructed a'search matrix' that will allow us to easily search the films in our list by choosing the genre. We must turn our matrix into a sparse matrix in order to receive ratings from recommenderlabs. This new matrix belongs to the'realRatingMatrix' class. Item Based Collaborative Filtering was implemented.

```
> movie_genre <- as.data.frame(movie_data$genres, stringsAsFactors=FALSE
> head(movie_genre)
                              movie_data$genres
1 Adventure|Animation|Children|Comedy|Fantasy
2                  Adventure|Children|Fantasy
3                            Comedy|Romance
4                     Comedy|Drama|Romance
5                                   Comedy
6                      Action|Crime|Thriller
> library(data.table)
> movie_genre2 <- as.data.frame(tstrsplit(movie_genre[,1], '[|]',
+                                         type.convert=TRUE),
+                               stringsAsFactors=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 ...
```

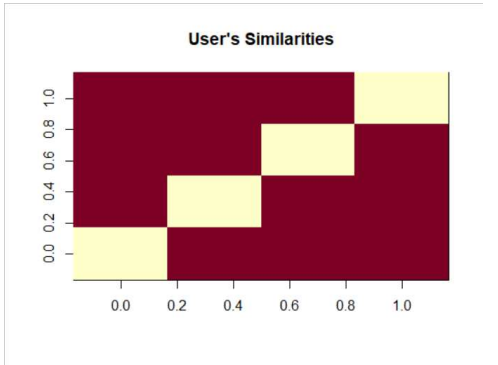
```
> head(genre_mat2)
  Action Adventure Animation Children Comedy Crime Documentary Drama Fantasy Film-Noir Horror Musical Mystery
1      0         1         1        1      1     0           0     0       1         0      0       0       0
2      0         1         0        1      0     0           0     0       1         0      0       0       0
3      0         0         0        0      1     0           0     0       0         0      0       0       0
4      0         0         0        0      1     0           0     1       0         0      0       0       0
5      0         0         0        0      1     0           0     0       0         0      0       0       0
6      1         0         0        0      0     1           0     0       0         0      0       0       0
  Romance Sci-Fi Thriller War Western
1       0      0        0   0       0
2       0      0        0   0       0
3       1      0        0   0       0
4       1      0        0   0       0
5       0      0        0   0       0
6       0      0        1   0       0
> SearchMatrix <- cbind(movie_data[,1:2], genre_mat2[])
> head(SearchMatrix)
  movieId                              title Action Adventure Animation Children Comedy Crime Documentary Drama
1       1                  Toy Story (1995)      0         1         1        1      1     0           0     0
2       2                    Jumanji (1995)      0         1         0        1      0     0           0     0
3       3           Grumpier Old Men (1995)      0         0         0        0      1     0           0     0
4       4          Waiting to Exhale (1995)      0         0         0        0      1     0           0     1
5       5 Father of the Bride Part II (1995)      0         0         0        0      1     0           0     0
6       6                       Heat (1995)      1         0         0        0      0     1           0     0
  Fantasy Film-Noir Horror Musical Mystery Romance Sci-Fi Thriller War Western
1       1         0      0       0       0       0      0        0   0       0
2       1         0      0       0       0       0      0        0   0       0
3       0         0      0       0       0       1      0        0   0       0
4       0         0      0       0       0       1      0        0   0       0
5       0         0      0       0       0       0      0        0   0       0
6       0         0      0       0       0       0      0        1   0       0
> ratingMatrix <- dcast(rating_data, userId~movieId, value.var = "rating", na.rm=FALSE)
> ratingMatrix <- as.matrix(ratingMatrix[,-1])
> ratingMatrix <- as(ratingMatrix, "realRatingMatrix")
> head(ratingMatrix)
6 x 10325 rating matrix of class 'realRatingMatrix' with 468 ratings.
> 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
> recommendation_model <- recommenderRegistry$get_entries(dataType = "realRatingMatrix")
> names(recommendation_model)
 [1] "HYBRID_realRatingMatrix"       "ALS_realRatingMatrix"          "ALS_implicit_realRatingMatrix"
 [4] "IBCF_realRatingMatrix"         "LIBMF_realRatingMatrix"        "POPULAR_realRatingMatrix"
 [7] "RANDOM_realRatingMatrix"       "RERECOMMEND_realRatingMatrix"  "SVD_realRatingMatrix"
[10] "SVDF_realRatingMatrix"         "UBCF_realRatingMatrix"
> lapply(recommendation_model, "[[", "description")
$HYBRID_realRatingMatrix
[1] "Hybrid recommender that aggegates several recommendation strategies using weighted averages."

$ALS_realRatingMatrix
```

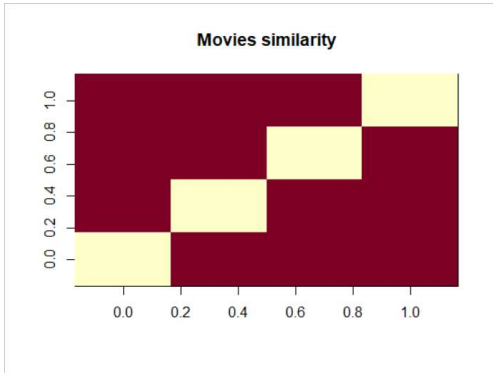
Step 3:Data Exploration

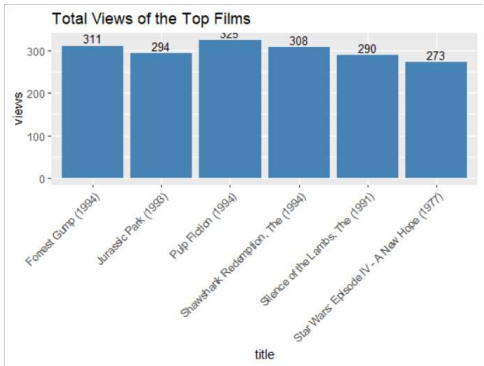
- In this step, we are able to discover the information this is contained withinside the datasets. We will use the str() function to display information about the movie_data and rating_data dataframes . We will use summary() function to summarize 2 datasets.

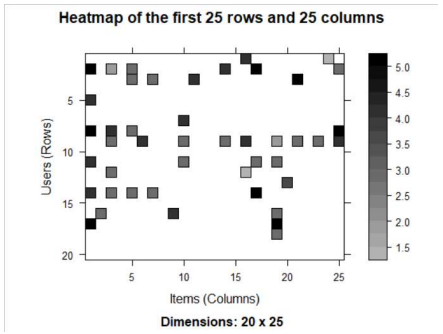
```
> str(movie_data)
'data.frame':   10329 obs. of  3 variables:
 $ movieId: int  1 2 3 4 5 6 7 8 9 10 ...
 $ title  : chr  "Toy Story (1995)" "Jumanji (1995)" "Grumpier Old Men (1995)" "Waiting to Exhale
 $ genres : chr  "Adventure|Animation|Children|Comedy|Fantasy" "Adventure|Children|Fantasy" "Comed
Drama|Romance" ...
> 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
> str(rating_data)
'data.frame':   105339 obs. of  4 variables:
 $ userId   : int  1 1 1 1 1 1 1 1 1 1 ...
 $ movieId  : int  16 24 32 47 50 110 150 161 165 204 ...
 $ rating   : num  4 1.5 4 4 4 4 3 4 3 0.5 ...
 $ timestamp: int  1217897793 1217895807 1217896246 1217896556 1217896523 1217896150 1217895940 12
 1217895786 ...
> 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
>
```

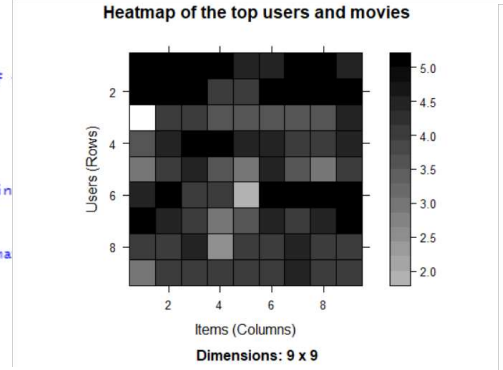
```
> similarity_mat <- similarity(ratingMatrix[1:4, ],method = "cosine",which = "users")
> as.matrix(similarity_mat)
          1         2         3         4
1 0.0000000 0.9760860 0.9641723 0.9914398
2 0.9760860 0.0000000 0.9925732 0.9374253
3 0.9641723 0.9925732 0.0000000 0.9888968
4 0.9914398 0.9374253 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")
> 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
> 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       5
6791761    1198    3258    1567    7943    5484   21729   12237   28880    8187   14856
> library(ggplot2)
> 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)
> 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")
> 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
420 x 447 rating matrix of class 'realRatingMatrix' with 38341 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
> 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`.
> 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
> 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],ma
```

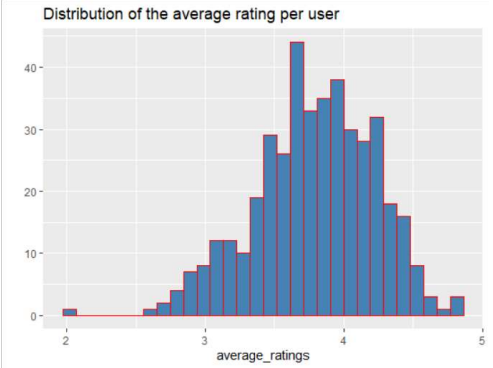


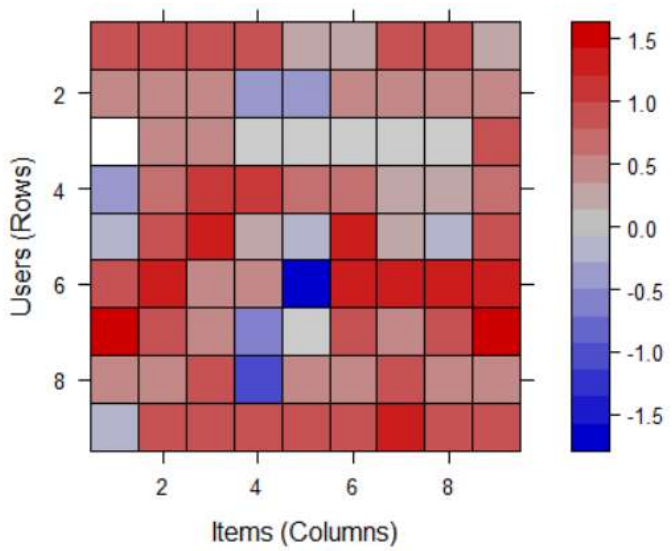


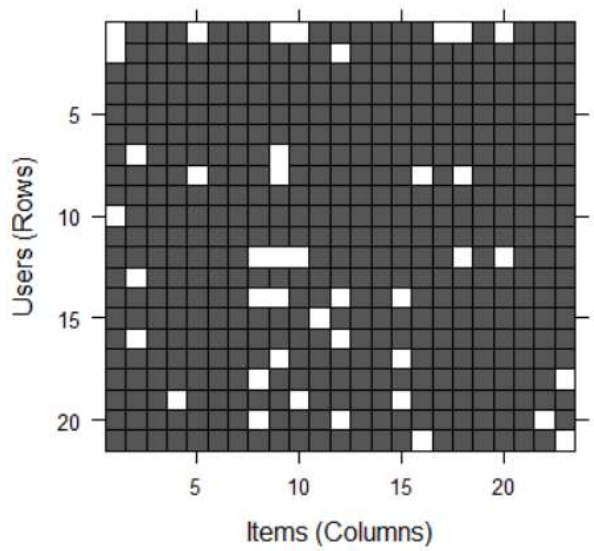









Normalized Ratings of the Top Users
Users (Rows)
Items (Columns)
2
4
6
8
1.5
1.0
0.5
0.0
-0.5
-1.0
-1.5
Dimensions: 9 x 9


Heatmap of the top users and movies
Users (Rows)
Items (Columns)
5
10
15
20
Dimensions: 21 x 23

```
> df$Amount=scale(df$Amount)
> data=df[,-c(1)]
> head(data)
         V1          V2        V3         V4          V5          V6          V7          V8         V9         V10        V11         V12        V13
1 -1.3598071 -0.07278117 2.5363467  1.3781552 -0.33832077  0.46238778  0.23959855  0.09869790  0.3637870  0.09079417 -0.5515995 -0.61780086 -0.9913898
2  1.1918571  0.26615071 0.1664801  0.4481541  0.06001765 -0.08236081 -0.07880298  0.08510165 -0.2554251 -0.16697441  1.6127267  1.06523531  0.4890950
3 -1.3583541 -1.34016307 1.7732093  0.3797796 -0.50319813  1.80049938  0.79146096  0.24767579 -1.5146543  0.20764287  0.6245015  0.06608369  0.7172927
4 -0.9662717 -0.18522601 1.7929933 -0.8632913 -0.01030888  1.24720317  0.23760894  0.37743587 -1.3870241 -0.05495192 -0.2264873  0.17822823  0.5077569
5 -1.1582331  0.87773675 1.5487178  0.4030339 -0.40719338  0.09592146  0.59294075 -0.27053268  0.8177393  0.75307443 -0.8228429  0.53819555  1.3458516
6 -0.4259659  0.96052304 1.1411093 -0.1682521  0.42098688 -0.02972755  0.47620095  0.26031433 -0.5686714 -0.37140720  1.3412620  0.35989384 -0.3580907
         V14        V15        V16         V17         V18         V19         V20          V21          V22         V23         V24         V25
1 -0.3111694  1.4681770 -0.4704005  0.20797124  0.02579058  0.40399296  0.25141210 -0.018306778  0.277837576 -0.11047391  0.06692807  0.1285394
2 -0.1437723  0.6355581  0.4639170 -0.11480466 -0.18336127 -0.14578304 -0.06908314 -0.225775248 -0.638671953  0.10128802 -0.33984648  0.1671704
3 -0.1659459  2.3458649 -2.8900832  1.10996938 -0.12135931 -2.26185710  0.52497973  0.247998153  0.771679402  0.90941226 -0.68928096 -0.3276418
4 -0.2879237 -0.6314181 -1.0596472 -0.68409279  1.96577500 -1.23262197 -0.20803778 -0.108300452  0.005273597 -0.19032052 -1.17557533  0.6473760
5 -1.1196698  0.1751211 -0.4514492 -0.23703324 -0.03819479  0.80348692  0.40854236 -0.009430697  0.798278495 -0.13745808  0.14126698 -0.2060096
6 -0.1371337  0.5176168  0.4017259 -0.05813282  0.06865315 -0.03319379  0.08496767 -0.208253515 -0.559824796 -0.02639767 -0.37142658 -0.2327938
         V26          V27         V28      Amount Class
1 -0.1891148  0.133558377 -0.02105305  0.24496383     0
2  0.1258945 -0.008983099  0.01472417 -0.34247394     0
3 -0.1390966 -0.055352794 -0.05975184  1.16068389     0
4 -0.2219288  0.062722849  0.06145763  0.14053401     0
5  0.5022922  0.219422230  0.21515315 -0.07340321     0
6  0.1059148  0.253844225  0.08108026 -0.33855582     0
```

Step 4 :Data Modeling

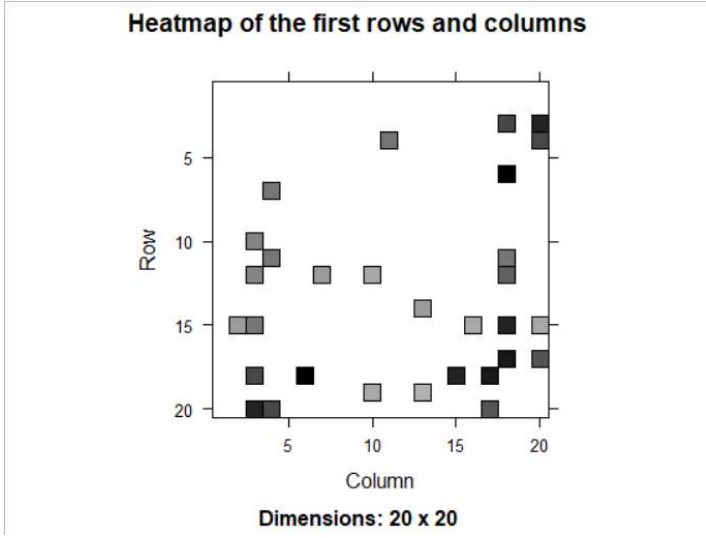
- Once we have standardized the dataset, we will divide our dataset into training sets and test sets with a divide ratio of 0.80. This means that 80% of our data will be attributed to train data while 20% will be attributed to test data.

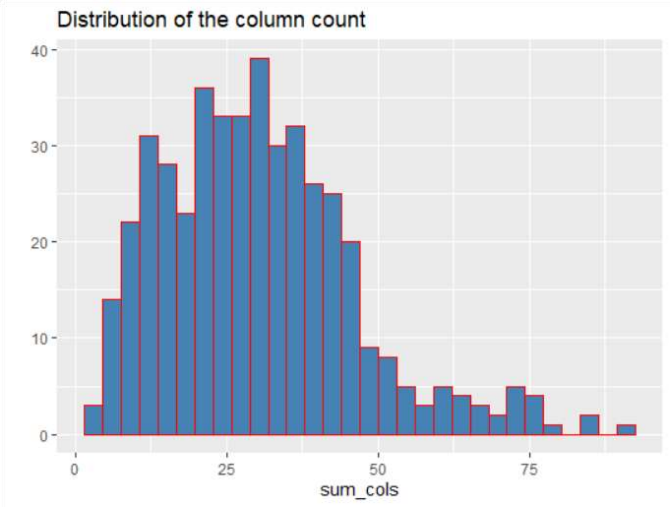
```
> sampled_data<- sample(x = c(TRUE, FALSE),size = nrow(movie_ratings),replace = TRUE,prob = c(0.8, 0.2))
> training_data <- movie_ratings[sampled_data, ]
> testing_data <- movie_ratings[!sampled_data, ]
> 
```

Build Recommender System

- We employ an Item Based Collaborative Filter. These parameters are of the default type. We utilise the cosine approach by default, but you can alternatively use the Pearson method. We will acquire the recommen model using the getModel() function. The class and dimensions of our similarity matrix, which are contained under model info, will then be determined. Finally, we will create a heatmap with the top 20 items and illustrate the similarity between them. We will compute the sum of rows and columns where the similarity of the objects is greater than 0. A distribution will be used to visualise the total of columns. We generated a top recommendations variable, which will be set to 10, indicating the amount of films recommended to each user. The predict() function will then be used to find comparable items and rank them appropriately. Each rating is treated as a weight in this case.

```
> recommen_model <- Recommender(data = training_data,method = "IBCF",parameter = list(k = 30))
> recommen_model
Recommender of type 'IBCF' for 'realRatingMatrix'
learned using 344 users.
> class(recommen_model)
[1] "Recommender"
attr(,"package")
[1] "recommenderlab"
> 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$sim[1:top_items, 1:top_items],main = "Heatmap of the first rows and columns")
> sum_rows <- rowSums(model_info$sim > 0)
> table(sum_rows)
sum_rows
 30
447
> sum_cols <- colSums(model_info$sim > 0)
> qplot(sum_cols, fill=I("steelblue"), col=I("red"))+ ggtitle("Distribution of the column count")
`stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
> top_recommendations <- 10
> predicted_recommendations <- predict(object = recommen_model,
+                                      newdata = testing_data,
+                                      n = top_recommendations)
> predicted_recommendations
Recommendations as 'topNList' with n = 10 for 76 users.
> user1 <- predicted_recommendations@items[[1]]
> movies_user1 <- predicted_recommendations@itemLabels[user1]
> movies_user2 <- movies_user1
> for (index in 1:10){
+   movies_user2[index] <- as.character(subset(movie_data,
+                                              movie_data$movieId == movies_user1[index])$title)
+ }
> movies_user2
 [1] "Snow White and the Seven Dwarfs (1937)"
 [2] "Field of Dreams (1989)"
 [3] "Cape Fear (1991)"
 [4] "Good Will Hunting (1997)"
 [5] "Trading Places (1983)"
 [6] "Big Fish (2003)"
 [7] "Lock, Stock & Two Smoking Barrels (1998)"
 [8] "Usual Suspects, The (1995)"
 [9] "WALLÂ·E (2008)"
[10] "Amelie (Fabuleux destin d'AmÃ©lie Poulain, Le) (2001)"
> recommendation_matrix <- sapply(predicted_recommendations@items,
+                                 function(x){ as.integer(colnames(movie_ratings)[x]) }) # matrix
>
> recommendation_matrix[,1:5]
      [,1] [,2]  [,3]  [,4] [,5]
 [1,]  594    7 49272   913  785
 [2,] 1302   47 54286  3147 4720
 [3,] 1343  161  1225   150  919
 [4,] 1704  265  1127 55820 3114
 [5,] 3039  355  1610   551  661
 [6,] 7147  474   509  3471   25
 [7,] 2542  497 33794  1221 3418
 [8,]   50  590  1527  1266 1641
 [9,] 60069  708   293  1997 2278
[10,] 4973  745  1249 48516 2700
```





Summary

- Finally, we learned how to use machine learning to develop our movie recommended model. We implemented this model using ML algorithm and plotted some visualizations of data and predictions. We learned how to analyse and visualise data in order to recommend movies to the users.