MovieLens Project, HarvardX

Ghodsieh Ghanbari

4/20/2021

Contents

Overview

The current project is related to the MovieLens project of the Capstone course of Harvardx: Data Science. A movie recommendation system will be created by using MovieLens data set. We divide the data set into the train set (edx), and test set (validation). We make an algorithm based on the data in edx set, then we use this algorithm to predict the ratings of the movies in the validation set.

The Root Mean Square Error is employed to evaluate the accuracy of the method and determine the closeness of the predicted values to the true ratings in validation set. The following formula will be used:

$$RMSE = \sqrt{\frac{1}{N}\sum_{u,i}(\hat{y}_{u,i} - y_{u,i})^2}$$

We will try several methods to find an algorithm with RMSE less than 0.86490.

Downloading the data set and building the train and test data sets

```
##########################################################
# Create edx set, validation set (final hold-out test set)
##########################################################
```

```r
# Note: this process could take a couple of minutes

if(!require(tidyverse)) install.packages("tidyverse", repos = "http://cran.us.r-project.org")
```

```
## Loading required package: tidyverse

## -- Attaching packages --------------------------------------- tidyverse 1.3.0 --

## v ggplot2 3.3.3     v purrr   0.3.4
## v tibble  3.1.0     v dplyr   1.0.5
## v tidyr   1.1.3     v stringr 1.4.0
## v readr   1.4.0     v forcats 0.5.1

## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```r
if(!require(caret)) install.packages("caret", repos = "http://cran.us.r-project.org")
```

```
## Loading required package: caret

## Loading required package: lattice

##
## Attaching package: 'caret'

## The following object is masked from 'package:purrr':
##
##     lift
```

```r
if(!require(data.table)) install.packages("data.table", repos = "http://cran.us.r-project.org")
```

```
## Loading required package: data.table

##
## Attaching package: 'data.table'

## The following objects are masked from 'package:dplyr':
##
##     between, first, last

## The following object is masked from 'package:purrr':
##
##     transpose
```

```r
library(dplyr)
library(caret)
library(data.table)
library(tinytex)
```

```
## Warning: package 'tinytex' was built under R version 4.0.5
```

```r
library(latexpdf)
library(ggplot2)

# MovieLens 10M dataset:
# https://grouplens.org/datasets/movielens/10m/
# http://files.grouplens.org/datasets/movielens/ml-10m.zip

dl <- tempfile()
download.file("http://files.grouplens.org/datasets/movielens/ml-10m.zip", dl)

ratings <- fread(text = gsub("::", "\t", readLines(unzip(dl, "ml-10M100K/ratings.dat"))),
                 col.names = c("userId", "movieId", "rating", "timestamp"))

movies <- str_split_fixed(readLines(unzip(dl, "ml-10M100K/movies.dat")), "\\::", 3)
colnames(movies) <- c("movieId", "title", "genres")


# if using R 4.0 or later:
movies <- as.data.frame(movies) %>% mutate(movieId = as.numeric(movieId),
                                           title = as.character(title),
                                           genres = as.character(genres))


movielens <- left_join(ratings, movies, by = "movieId")

# Validation set will be 10% of MovieLens data

set.seed(1, sample.kind="Rounding") # if using R 3.5 or earlier, use `set.seed(1)`
```

```
## Warning in set.seed(1, sample.kind = "Rounding"): non-uniform 'Rounding' sampler
## used
```

```r
test_index <- createDataPartition(y = movielens$rating, times = 1, p = 0.1, list = FALSE)
edx <- movielens[-test_index,]
temp <- movielens[test_index,]

# Make sure userId and movieId in validation set are also in edx set
validation <- temp %>%
      semi_join(edx, by = "movieId") %>%
      semi_join(edx, by = "userId")

# Add rows removed from validation set back into edx set
removed <- anti_join(temp, validation)
```

```
## Joining, by = c("userId", "movieId", "rating", "timestamp", "title", "genres")
```

```r
edx <- rbind(edx, removed)

rm(dl, ratings, movies, test_index, temp, movielens, removed)
```

Analysis and methods

Data analysis

In this part, in order to know more about the data we see the first rows of the train set. It is seen that there are 6 features in the edx data set which are userId, movieId, rating, timestamp, title, and genres.

```
##    userId movieId rating timestamp                         title
## 1:      1     122      5 838985046              Boomerang (1992)
## 2:      1     185      5 838983525               Net, The (1995)
## 3:      1     292      5 838983421               Outbreak (1995)
## 4:      1     316      5 838983392               Stargate (1994)
## 5:      1     329      5 838983392 Star Trek: Generations (1994)
## 6:      1     355      5 838984474       Flintstones, The (1994)
##                           genres
## 1:                Comedy|Romance
## 2:         Action|Crime|Thriller
## 3:  Action|Drama|Sci-Fi|Thriller
## 4:       Action|Adventure|Sci-Fi
## 5: Action|Adventure|Drama|Sci-Fi
## 6:       Children|Comedy|Fantasy
```

The number of distinct movies, users, and genres are as follows:

```
##   number_of_distinct_users number_of_distinct_movies number_of_distinct_genres
## 1                    69878                     10677                       797
```

We transform timestamp column to the year the movie was rated and extract the year each movie was released. So, we get the following data set:

```
##
## Attaching package: 'lubridate'

## The following objects are masked from 'package:data.table':
##
##     hour, isoweek, mday, minute, month, quarter, second, wday, week,
##     yday, year

## The following objects are masked from 'package:base':
##
##     date, intersect, setdiff, union

##    userId movieId rating timestamp                         title
## 1:      1     122      5 838985046              Boomerang (1992)
## 2:      1     185      5 838983525               Net, The (1995)
## 3:      1     292      5 838983421               Outbreak (1995)
## 4:      1     316      5 838983392               Stargate (1994)
## 5:      1     329      5 838983392 Star Trek: Generations (1994)
## 6:      1     355      5 838984474       Flintstones, The (1994)
##                           genres rating_year
## 1:                Comedy|Romance        1996
## 2:         Action|Crime|Thriller        1996
## 3:  Action|Drama|Sci-Fi|Thriller        1996
## 4:       Action|Adventure|Sci-Fi        1996
## 5: Action|Adventure|Drama|Sci-Fi        1996
## 6:       Children|Comedy|Fantasy        1996
```

```
##    userId movieId rating                         title
## 1:      1     122      5              Boomerang (1992)
## 2:      1     185      5               Net, The (1995)
## 3:      1     292      5               Outbreak (1995)
## 4:      1     316      5               Stargate (1994)
## 5:      1     329      5 Star Trek: Generations (1994)
## 6:      1     355      5       Flintstones, The (1994)
##                           genres rating_year released_year
## 1:                Comedy|Romance        1996          1992
## 2:         Action|Crime|Thriller        1996          1995
## 3:  Action|Drama|Sci-Fi|Thriller        1996          1995
## 4:       Action|Adventure|Sci-Fi        1996          1994
## 5: Action|Adventure|Drama|Sci-Fi        1996          1994
## 6:       Children|Comedy|Fantasy        1996          1994
```

Calculating age of the movies at the time of rating and adding a new column called age_of_movie to the data set:

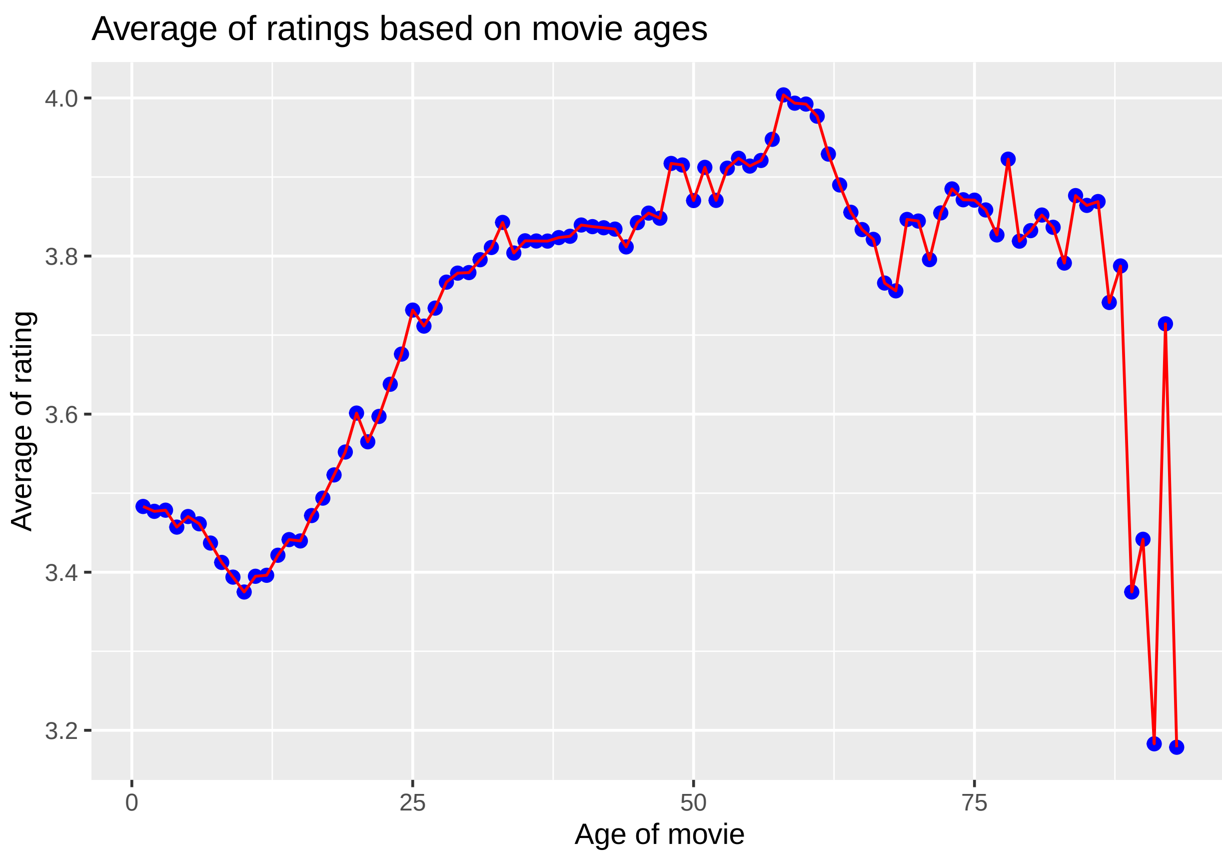
```
##    userId movieId rating                         title
## 1:      1     122      5              Boomerang (1992)
## 2:      1     185      5               Net, The (1995)
## 3:      1     292      5               Outbreak (1995)
## 4:      1     316      5               Stargate (1994)
## 5:      1     329      5 Star Trek: Generations (1994)
## 6:      1     355      5       Flintstones, The (1994)
##                           genres rating_year released_year age_of_movie
## 1:                Comedy|Romance        1996          1992            4
## 2:         Action|Crime|Thriller        1996          1995            1
## 3:  Action|Drama|Sci-Fi|Thriller        1996          1995            1
## 4:       Action|Adventure|Sci-Fi        1996          1994            2
## 5: Action|Adventure|Drama|Sci-Fi        1996          1994            2
## 6:       Children|Comedy|Fantasy        1996          1994            2
```

We explore if the age of movies has correlation with ratings. First, we find the unique ages of movies.

```
## [1] "unique values of age of movies:"
```

```
##  [1]  4  1  2  3  5 59 26 20  7 14 11 10 12 16 31 38 21 13  8  9  6 17 24 15 25
## [26] 39 55 58 56 47 46 18 42 35 37 23 30 29 32  0 51 45 44 62 64 57 63 49 41 54
## [51] 43 19 70 72 36 34 40 81 28 52 60 22 50 53 27 33 66 67 61 69 68 48 65 74 75
## [76] 71 77 73 79 76 83 78 85 80 84 91 82 90 89 88 86 87 -1 93 92 -2
```

```
## # A tibble: 6 x 2
##   age_of_movie age_ave_rating
##          <dbl>          <dbl>
## 1           58           4.00
## 2           59           3.99
## 3           60           3.99
## 4           61           3.98
## 5           57           3.95
## 6           62           3.93
```

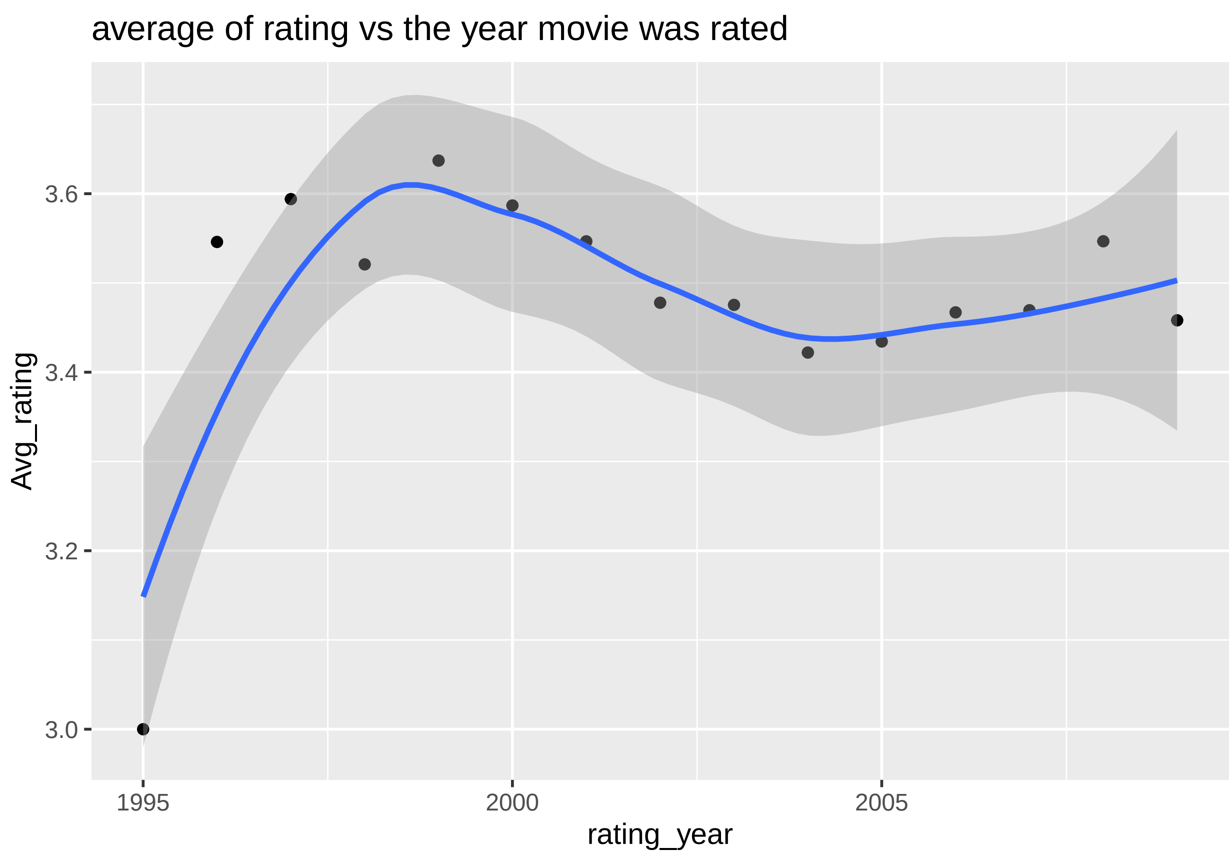


From the above graph we can see that the older is the movie the higher is its rating.

Determining if the year in which the movie was rated has correlation with rating:

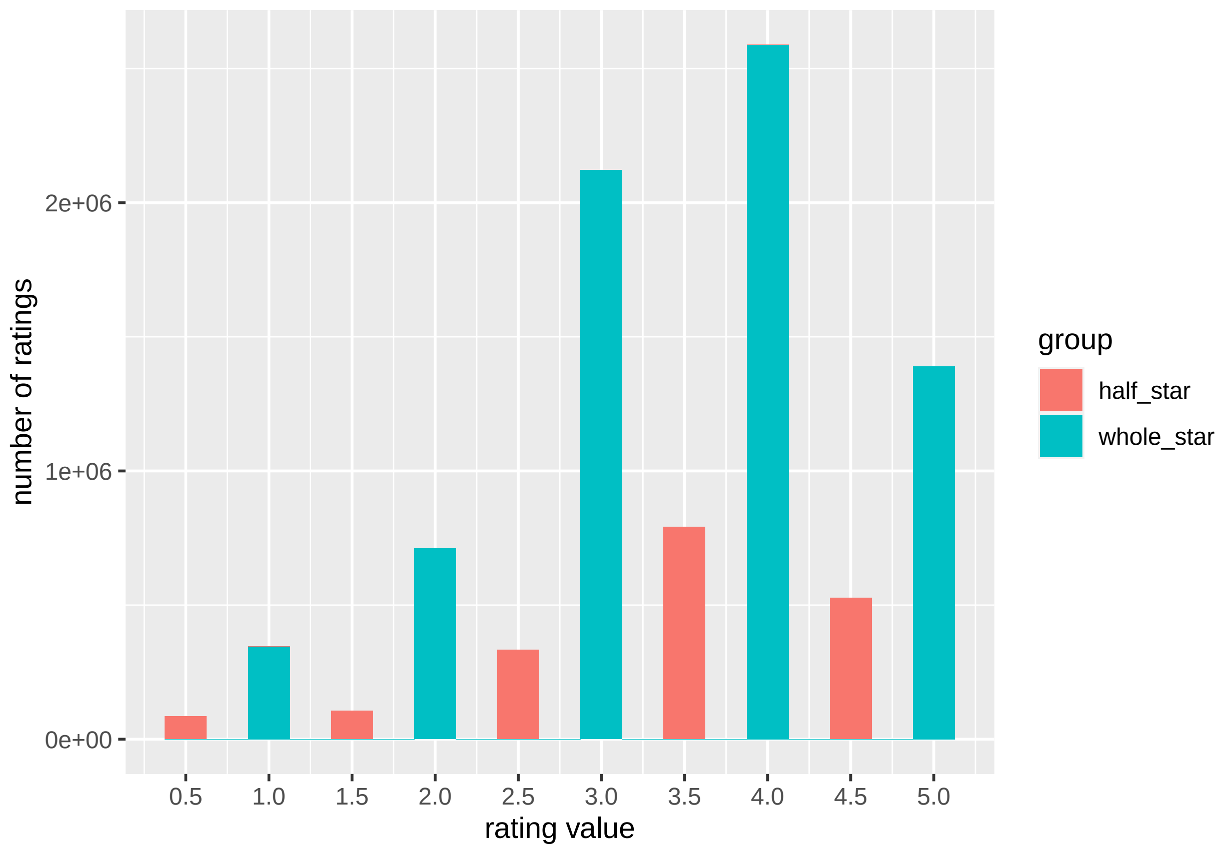
```
## # A tibble: 6 x 2
##   rating_year Avg_rating
##         <dbl>      <dbl>
## 1        1999       3.64
## 2        1997       3.59
## 3        2000       3.59
## 4        2008       3.55
## 5        2001       3.55
## 6        1996       3.55
```

```
## `geom_smooth()` using method = 'loess' and formula 'y ~ x'
```



It is seen that between 1996 and 2002 almost the highest ratings were given to the movies.

Now, we explore the variability of number of ratings for whole star and half star rating values.



From the above histogram we can observe that no one has gives 0 as a rating. Also, whole stars are more common compared to half stars. The most common ratings are 4,3, 5, 3.5.

Exploring the top 10 movies based on average of ratings:

```
## # A tibble: 12 x 3
##    title                                                     n avg_rate
##    <chr>                                                 <int>    <dbl>
##  1 Blue Light, The (Das Blaue Licht) (1932)                  1     5
##  2 CJ7 (Cheung Gong 7 hou) (2008)                            1     5
##  3 Constantine's Sword (2007)                                2     4.75
##  4 Fighting Elegy (Kenka erejii) (1966)                      1     5
##  5 Hellhounds on My Trail (1999)                             1     5
##  6 Human Condition II, The (Ningen no joken II) (1959)       4     4.75
##  7 Human Condition III, The (Ningen no joken III) (1961)     4     4.75
##  8 Miss Pettigrew Lives for a Day (2008)                     2     4.75
##  9 Satan's Tango (SÃ¡tÃ¡ntangÃ³) (1994)                      2     5
## 10 Shadows of Forgotten Ancestors (1964)                     1     5
## 11 Sun Alley (Sonnenallee) (1999)                            1     5
## 12 Who's Singin' Over There? (a.k.a. Who Sings Over There) (Ko t~     4     4.75
```

We can see that the highest ratings belong to movies which were rated a few times!

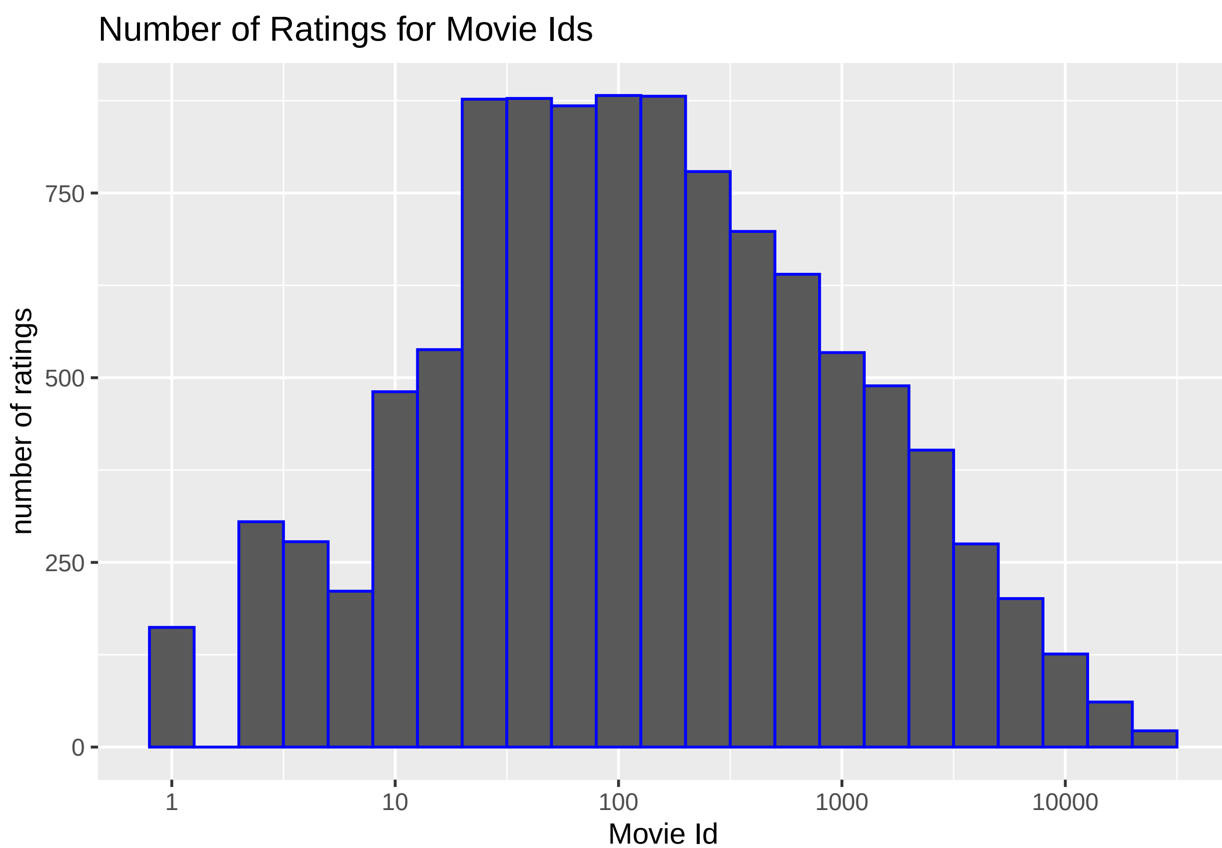
We explore which movies had the highest number of ratings:

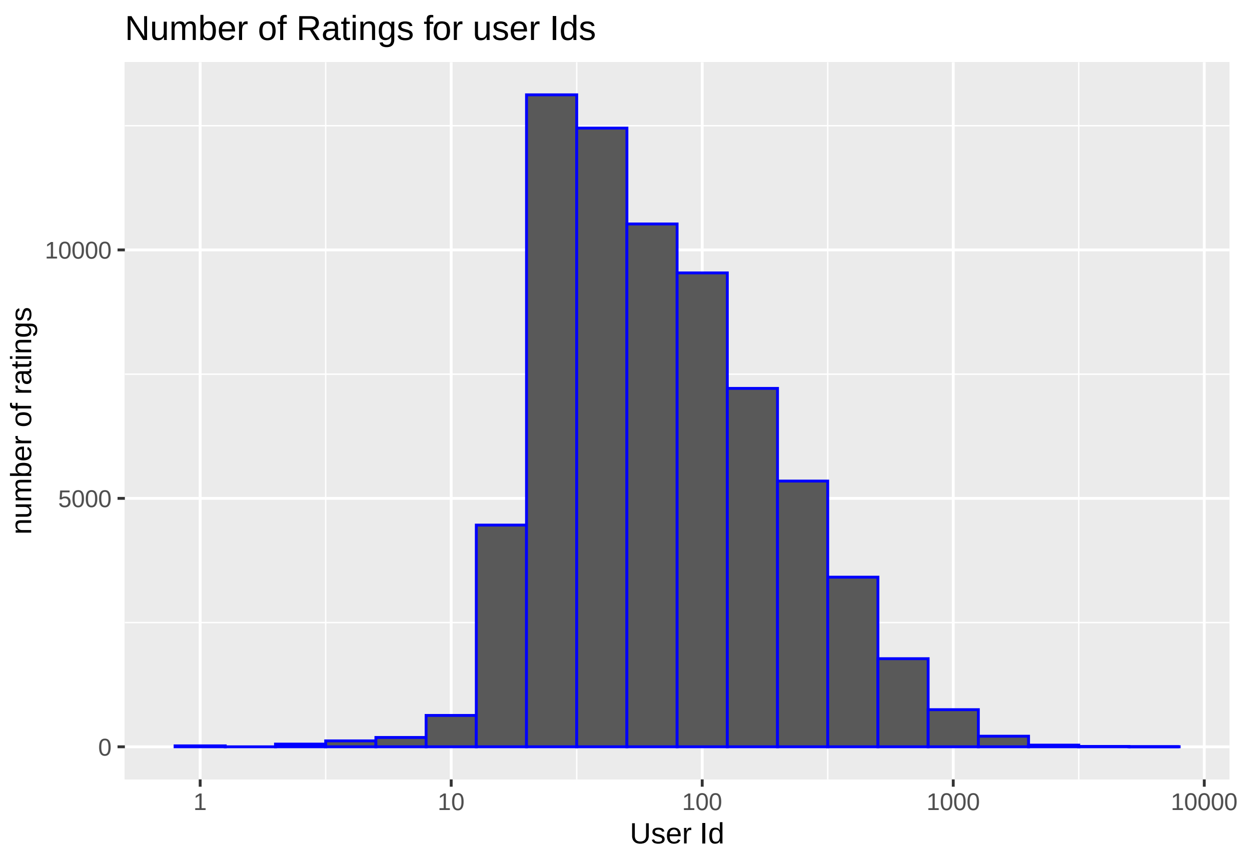
```
## # A tibble: 10,587 x 2
##    title                                                   n
```

```
##    <chr>                                                    <int>
##  1 Pulp Fiction (1994)                                      31362
##  2 Forrest Gump (1994)                                      31079
##  3 Silence of the Lambs, The (1991)                         30382
##  4 Jurassic Park (1993)                                     29360
##  5 Shawshank Redemption, The (1994)                         28015
##  6 Braveheart (1995)                                        26212
##  7 Fugitive, The (1993)                                     25998
##  8 Terminator 2: Judgment Day (1991)                        25984
##  9 Star Wars: Episode IV - A New Hope (a.k.a. Star Wars) (1977) 25672
## 10 Apollo 13 (1995)                                         24284
## # ... with 10,577 more rows
```

The movie Pulp Fiction had the highest number of ratings, 31362 times.

Histograms of number of rating by movieId and userId:





Based on the above histograms, since some movies get more number of ratings and some users give more number of ratings, we should consider movie and user effects.

Exploring the relation between the genre of movies and movie ratings. I extract genre and analyze to see what are the effects of genres on movie ratings.

```
## [1] 20000
```

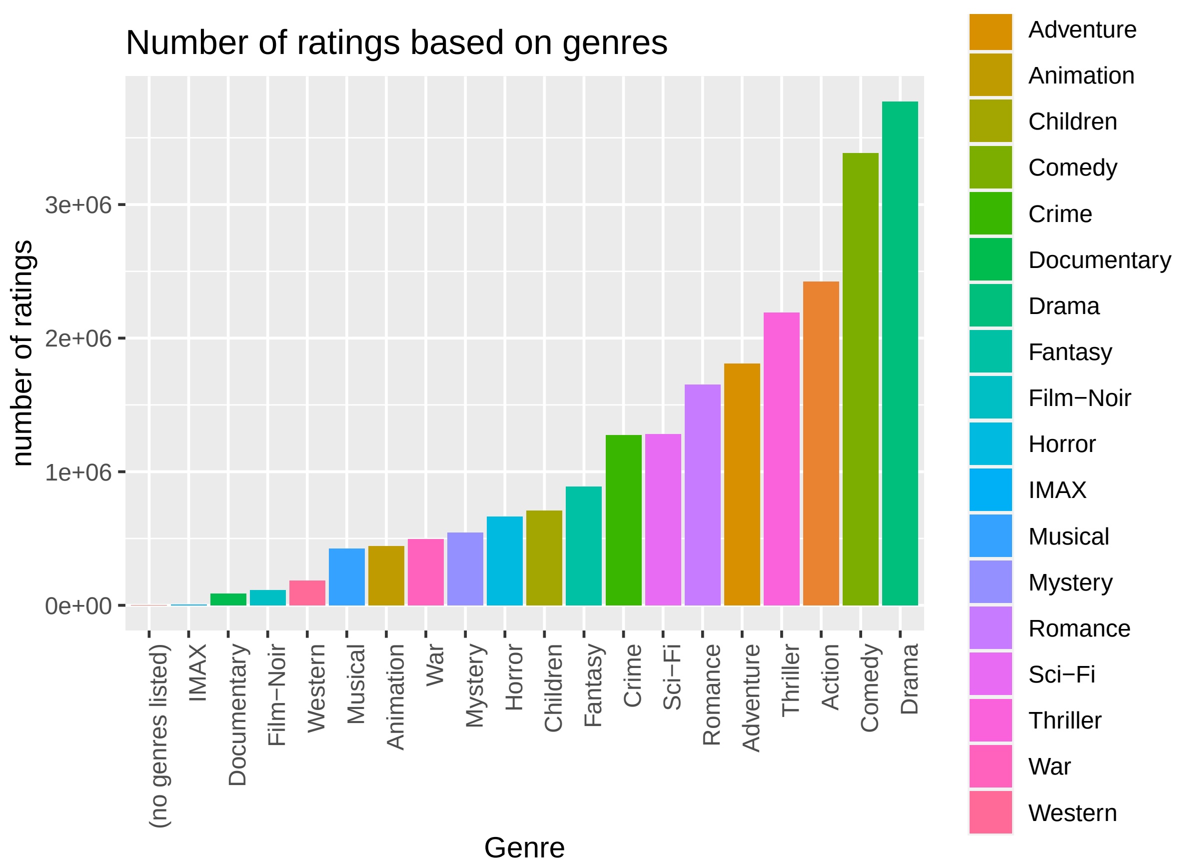
```
## # A tibble: 6 x 5
##   genres               count avg_rating_genre distinctMovies distinctUserId
##   <chr>                <int>            <dbl>          <int>          <int>
## 1 (no genres listed)       7             3.64              1              7
## 2 Action             2423024             3.42           1465          69214
## 3 Adventure          1808971             3.49           1021          69114
## 4 Animation           444420             3.60            280          58167
## 5 Children            708993             3.42            525          63463
## 6 Comedy             3385808             3.44           3672          69798
```

It is seen that there are 19 type of genres. We have one movie with no genre which was rated by 7 users.

Arranging the genres based on the number of ratings in each genre

```
## # A tibble: 20 x 5
##    genres               count avg_rating_genre distinctMovies distinctUserId
##    <chr>                <int>            <dbl>          <int>          <int>
##  1 Drama              3772120             3.68           5302          69788
```

```
##   2 Comedy             3385808          3.44        3672        69798
##   3 Action             2423024          3.42        1465        69214
##   4 Thriller           2193086          3.51        1694        69236
##   5 Adventure          1808971          3.49        1021        69114
##   6 Romance            1652625          3.56        1677        69321
##   7 Sci-Fi             1281377          3.39         752        67870
##   8 Crime              1275413          3.67        1113        68302
##   9 Fantasy             889119          3.50         540        66360
##  10 Children            708993          3.42         525        63463
##  11 Horror              666093          3.28        1000        59775
##  12 Mystery             544948          3.68         504        60399
##  13 War                 496075          3.79         503        63882
##  14 Animation           444420          3.60         280        58167
##  15 Musical             423064          3.56         431        58345
##  16 Western             186647          3.56         274        47284
##  17 Film-Noir           116011          4.02         148        30767
##  18 Documentary          87551          3.78         470        23725
##  19 IMAX                  5017          3.54          29         4247
##  20 (no genres listed)       7          3.64           1            7
```



We can observe that which genres have the most number of movies and ratings. They are Drama, Comedy, Action, etc.

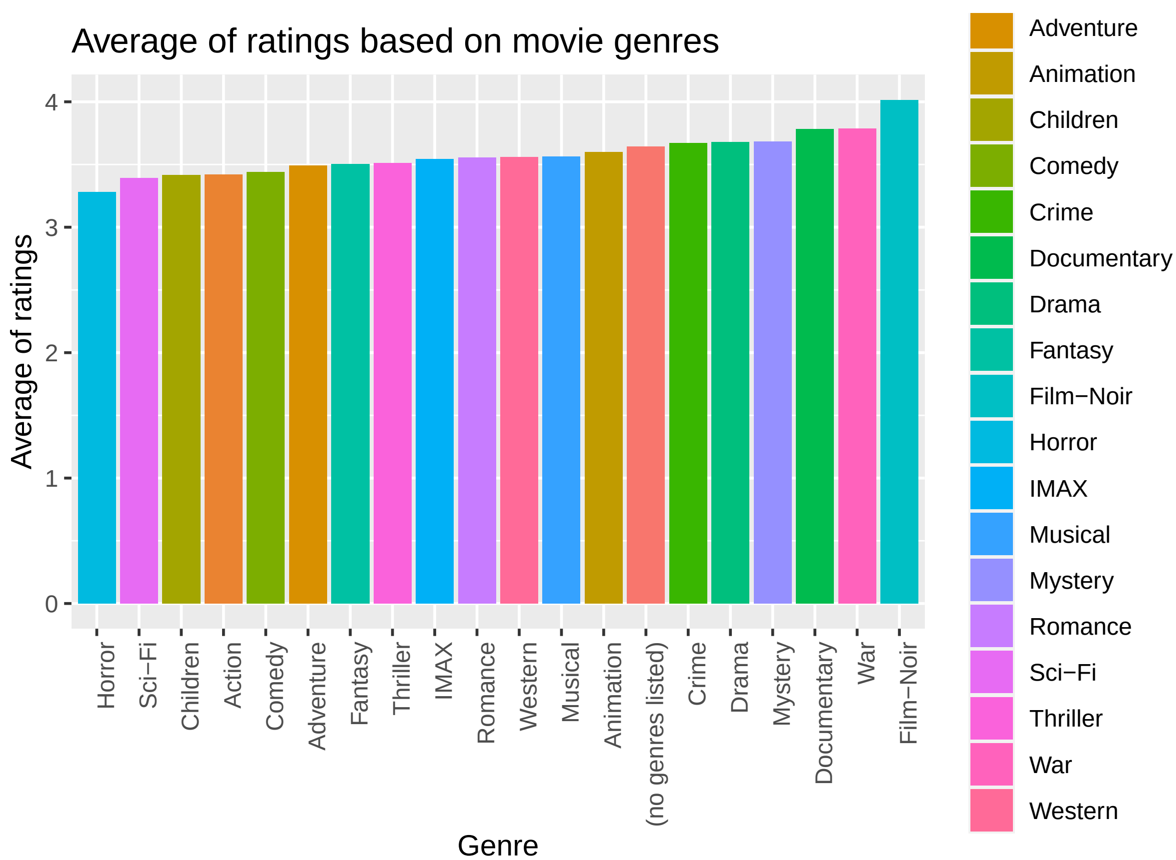
Arranging the genres based on the average of rating:

```
## # A tibble: 20 x 5
```

```
##    genres                count avg_rating_genre distinctMovies distinctUserId
##    <chr>                 <int>            <dbl>          <int>          <int>
##  1 Film-Noir            116011             4.02            148          30767
##  2 War                  496075             3.79            503          63882
##  3 Documentary           87551             3.78            470          23725
##  4 Mystery              544948             3.68            504          60399
##  5 Drama               3772120             3.68           5302          69788
##  6 Crime               1275413             3.67           1113          68302
##  7 (no genres listed)        7             3.64              1              7
##  8 Animation            444420             3.60            280          58167
##  9 Musical              423064             3.56            431          58345
## 10 Western              186647             3.56            274          47284
## 11 Romance             1652625             3.56           1677          69321
## 12 IMAX                   5017             3.54             29           4247
## 13 Thriller            2193086             3.51           1694          69236
## 14 Fantasy              889119             3.50            540          66360
## 15 Adventure           1808971             3.49           1021          69114
## 16 Comedy              3385808             3.44           3672          69798
## 17 Action              2423024             3.42           1465          69214
## 18 Children             708993             3.42            525          63463
## 19 Sci-Fi              1281377             3.39            752          67870
## 20 Horror               666093             3.28           1000          59775
```

It is seen that some genres with low number of ratings have high average rating.

We graph the average of rating based on genre of movies:



The above graph shows that which genres have highest average ratings. Film-Noir has the highest average rating with 116011 ratings.

Methods

In this part we evaluate several methods to see which method gives the desired RMSE (< 0.86490). We create following models: 1) naive model, just using the average ratings. 2) Movie effect model. 3) Movie and user effects model. 4) Regularized Movie and user effects model.

The following formula is used to compute RMSE:

$$RMSE = \sqrt{\frac{1}{N}\sum_{u,i}(\hat{y}_{u,i} - y_{u,i})^2}$$

where N is the number of user and movie combinations. The lower is RMSE, the better is the model.

The following function is used for calculating RMSE:

```
#RMSE function:

RMSE <- function(true_ratings, predicted_ratings){
     sqrt(mean((true_ratings - predicted_ratings)^2))}
```

Naive model:

$$Y_{u,i} = \mu + \epsilon_{u,i}$$

```
#Naive model:
  mu<- mean(edx$rating)
naive_rmse <- RMSE(validation$rating, mu)
naive_rmse
```

```
## [1] 1.061202
```

Table of result:

```
## Warning: `data_frame()` was deprecated in tibble 1.1.0.
## Please use `tibble()` instead.
```

Method	RMSE
Just the average	1.061202

Movie effect model:

$$Y_{u,i} = \mu + b_i + \epsilon_{u,i}$$

```
# Model with Movie effect:

movie_avgs <- edx %>%
     group_by(movieId) %>%
     summarize(b_i = mean(rating - mu))
```

```
predicted_ratings <- mu + validation %>%
     left_join(movie_avgs, by='movieId') %>% .$b_i

model_b_i_rmse <- RMSE(validation$rating, predicted_ratings)

rmse_results <- bind_rows(rmse_results,
                          data_frame(Method="Movie Effect Model",
                                     RMSE = model_b_i_rmse ))
```

Table of results:

Method	RMSE
Just the average	1.0612018
Movie Effect Model	0.9439087

Movie and user effects model:

$$Y_{u,i} = \mu + b_i + b_u + \epsilon_{u,i}$$

```
# Model with Movie and User effect:

user_avgs <- edx %>%
     left_join(movie_avgs, by='movieId') %>%
     group_by(userId) %>%
     summarize(b_u = mean(rating - mu - b_i))

predicted_ratings <- validation %>%
     left_join(movie_avgs, by='movieId') %>%
     left_join(user_avgs, by='userId') %>%
     mutate(pred = mu + b_i + b_u) %>%
     .$pred

model_b_i_u_rmse <- RMSE(validation$rating, predicted_ratings)

rmse_results <- bind_rows(rmse_results,
                          data_frame(Method="Movie + User Effects Model",
                                     RMSE = model_b_i_u_rmse ))
```

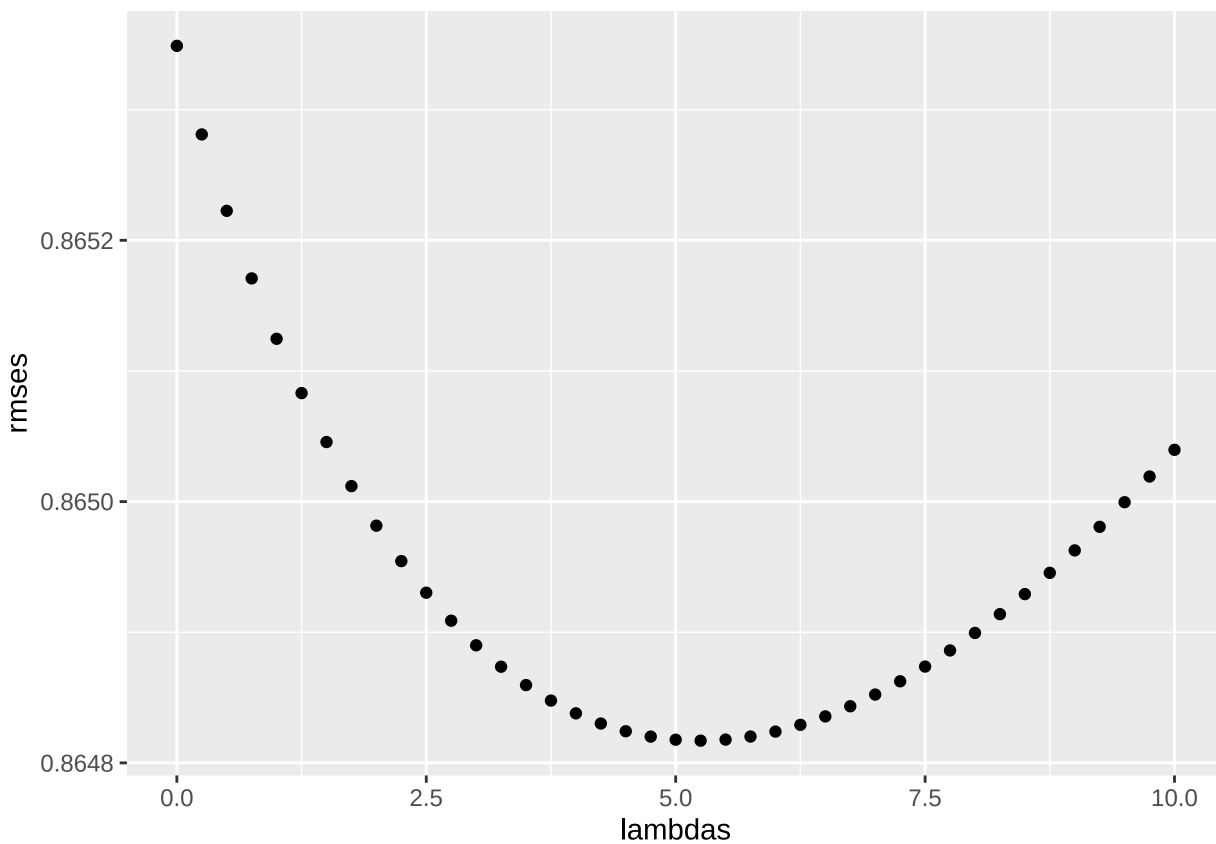
Table of results:

Method	RMSE
Just the average	1.0612018
Movie Effect Model	0.9439087
Movie + User Effects Model	0.8653488

Now, since we had movies with few number of ratings and users that rated a few movies, we apply Regularization for movie and user effects.

```r
lambdas <- seq(0, 10, 0.25)

rmses <- sapply(lambdas, function(l){
     mu <- mean(edx$rating)

     b_i <- edx %>%
          group_by(movieId) %>%
          summarize(b_i = sum(rating - mu)/(n()+l))
     b_u <- edx %>%
          left_join(b_i, by="movieId") %>%
          group_by(userId) %>%
          summarize(b_u = sum(rating - b_i - mu)/(n()+l))

     predicted_ratings <- validation %>%
          left_join(b_i, by = "movieId") %>%
          left_join(b_u, by = "userId") %>%
          mutate(pred = mu + b_i + b_u) %>%
          .$pred

     return(RMSE(validation$rating, predicted_ratings))
})
```

Plot of RMSE versus lambda values:



```
lambdas[which.min(rmses)]
```

```
## [1] 5.25
```

It is seen that 5.25 gives the minimum RMSE. Therefore, we use lambda = 5.25 for our final model.

Results

The table below shows the RMSE values obtained by different methods. Based on the table Reguralized movie and user effects model gives the lowest RMSE. Thus, the final RMSE is 0.8648170.

Method	RMSE
Just the average	1.0612018
Movie Effect Model	0.9439087
Movie + User Effects Model	0.8653488
Regularized Movie + User Effects Model	0.8648170

Conclusion

In this project we built a recommendation system for the MovieLens data set. We created several models based on the data in the train set and applied these models to predict ratings in the test sat (validation data set). It was shown that applying regularization on movie and user effects will improve our model and will result in more accurate predicting based on the RMSE values.