Movie Lens Project Final Report

HarvardX: PH125.9x; Data Science Capstone

Marwa J. Nafakh

7-1-2020

Report Outline

Introduction 2
I. Dataset Description 2
II. Goal of the Project 2
Methods and Analysis 3
I. Data cleaning 4
II. Data Exploration and Visualization 4
III. Insights Gained 15
IV. Modeling Approach 15
Results 22
Conclusion 23
I. Summary of the report 23
II. Limitation and Future Work 23

Introduction

The methodology of Recommendation systems is aimed at predicting recommendations based on a built-in algorithm that considers various factors to generate the desirable outcome for users. Building the right algorithm has a major impact on users' satisfaction and hence a tangible consequence in generating revenue and business growth in relevant sectors.

In this project, a machine learning algorithm was developed based on considering average predictors, user, movie and genres predictors, and then tuning the later model through a tuning factor (lambda) to improve the Root Mean Squared Error in which its target value was lower than 0.86490.

I. Dataset Description

In the MovieLens (10M) dataset, data has been collected for a period of 14 years for random movie viewers and released in 2009 (data collection started in 1995 to 2003). Ratings of movies was based on a 5-point scale until 2003 where the half star rating was introduced.

For the purpose of this course work the MovieLens huge dataset (10M) has been subdivided into smaller datasets to ease the process of analysis and modelling.

1. The edx dataset (training subset) has been created for the purpose of finding the optimal recommendation system algorithm representing 90% of the MovieLens dataset. "edx" dataset consists of 9000055 observations with 6 factors/variables.
2. The validation dataset (testing subset) representing 10% of the MovieLens dataset, used for validation purposes only acted upon by the generated data models to find the optimal model.

II. Goal of the Project

This project aims at building a movie recommendation system based on users' ratings and achieve an RMSE lower than "0.86490" through the implementation and testing of various data prediction and modelling algorithms. Validation of the selected machine learning algorithm is ensured through the validation dataset. The evaluation criterion of the generated models is the RMSE metrics; root mean squared error, and calculated through the below formula:

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

- N is the number of user-movie-genres combination.
- $y_{u,i,g}$ is the rating for movie 'i' by user 'u' and genres 'g'.
- $\hat{y}_{u,i,g}$ is the predicted ratings.

Methods and Analysis

MovieLens dataset was downloaded through the provided code on the EDX learning platform. ‘edx’ dataset serving as the training dataset; 90% of the MovieLens dataset, has been used to train all the generated models. “validation” dataset; 10% of the full MovieLens dataset used for model validation purposes assuming that this dataset is unknown to measure the accuracy and select the best model based on the lowest returned RMSE value. Through cross validation method of the predicted ratings against the actual ratings in the validation test set.

The following steps were performed on the ‘edx’ dataset leading to proper definition of relationships between predictors and hence gaining an insight for developing an effective algorithm with the lowest RMSE for the recommendation system.

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

MovieLens 10M dataset Source:

- https://grouplens.org/datasets/movielens/10m/
- http://files.grouplens.org/datasets/movielens/ml-10m.zip

```r
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")

movies <- as.data.frame(movies) %>% mutate(movieId = as.numeric(movieId),
                                           title = as.character(title),
                                           genres = as.character(genres))
```

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

# Validation set will be 10% of MovieLens data
set.seed(1, sample.kind="Rounding")

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)
edx <- rbind(edx, removed)

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

I. Data cleaning

The edx dataset has been checked for any missing value and returned FALSE ensuring there were no missed values that might have an impact on the accuracy of the built model. The data set was tidy since each row has one observation and columns stated the features for each observation.

```
any(is.na(edx))
```

```
## [1] FALSE
```

In this project the needed factors to predict the rating for a given user includes movie, user, and genres. Therefore, these were the factors selected from the data sets (edx and validation) to perform the analysis on.

```
# First the used factors will be selected from both edx and validation datasets
edx <- edx %>% select(userId, movieId, rating, title, genres)
validation <- validation  %>% select(userId, movieId, rating, title, genres)
```

II. Data Exploration and Visualization

In order to better understand the edx dataset, the relationships between the various factors, and gain a clear insight of how to develop an effective recommendation system algorithm, various data exploration methods took place.

1. Exploring the structure of the ‘edx’ dataset

```
str(edx)
```

```
## Classes 'data.table' and 'data.frame':   9000055 obs. of  6 variables:
##  $ userId   : int  1 1 1 1 1 1 1 1 1 1 ...
##  $ movieId  : num  122 185 292 316 329 355 356 362 364 370 ...
##  $ rating   : num  5 5 5 5 5 5 5 5 5 5 ...
##  $ timestamp: int  838985046 838983525 838983421 838983392 838983392 838984474 838983653 838984885 838983707 8
38984596 ...
##  $ title    : chr  "Boomerang (1992)" "Net, The (1995)" "Outbreak (1995)" "Stargate (1994)" ...
##  $ genres   : chr  "Comedy|Romance" "Action|Crime|Thriller" "Action|Drama|Sci-Fi|Thriller" "Action|Adventure|S
ci-Fi" ...
##  - attr(*, ".internal.selfref")=<externalptr>
```

```
head(edx)
```

```
##    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
```

```
class(edx)
```

```
## [1] "data.table" "data.frame"
```

```
summary(edx)
```

```
##      userId         movieId          rating        timestamp
##  Min.   :    1   Min.   :    1   Min.   :0.500   Min.   :7.897e+08
##  1st Qu.:18124   1st Qu.:  648   1st Qu.:3.000   1st Qu.:9.468e+08
##  Median :35738   Median : 1834   Median :4.000   Median :1.035e+09
##  Mean   :35870   Mean   : 4122   Mean   :3.512   Mean   :1.033e+09
##  3rd Qu.:53607   3rd Qu.: 3626   3rd Qu.:4.000   3rd Qu.:1.127e+09
##  Max.   :71567   Max.   :65133   Max.   :5.000   Max.   :1.231e+09
##     title              genres
##  Length:9000055     Length:9000055
##  Class :character   Class :character
##  Mode  :character   Mode  :character
##
##
##
```

The ‘edx’ dataset is of class: data.frame and consists of 9000055 observations of 6 variables.

2. Understanding Collected Data

After that, the various factors included in the dataset were further explored giving an insight on how to predict movie ratings for users. Time, movie, user, genres, and rating behavior of users were analyzed and summarized using various graphs and tables.

a. Time

By adding a new column to the data.frame; year_of_rating, that makes it easier to later visualize all time related data, the original dataset has the timestamp measured in seconds since Jan, 1970.

```r
library(lubridate)
# Convert the timestamp factor (measured in seconds) into more readable date format by adding a new
# column with the year of rating
edx <- mutate(edx, year_of_rating = year(as_datetime(timestamp)))

# View edx data.frame after new column addition
head(edx)
```

```
##    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 year_of_rating
## 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
```

Also, the duration of collected data was calculated:

```r
# Calculate the duration of collected data

tibble('First Date' = min(edx$year_of_rating),
       'Final Date' = max(edx$year_of_rating))%>%
        mutate(Duration = (max(edx$year_of_rating) -
                            min(edx$year_of_rating)))
```

```
## # A tibble: 1 x 3
##   `First Date` `Final Date` Duration
##          <dbl>        <dbl>    <dbl>
## 1         1995         2009       14
```

b. Users

Users were also explored, and their characteristics were summarized for further analysis.

```r
# Counting and displaying the number of unique users

paste('There are ', n_distinct(edx$userId), 'different users')
```

```
## [1] "There are  69878 different users"
```

```
# Understanding raters statistics

edx_users <- edx %>% group_by(userId) %>%
                    summarize(num=n()) %>%
                    arrange(desc(num))
summary(edx_users)
```

```
##      userId           num
##  Min.   :    1   Min.   :  10.0
##  1st Qu.:17943   1st Qu.:  32.0
##  Median :35799   Median :  62.0
##  Mean   :35782   Mean   : 128.8
##  3rd Qu.:53620   3rd Qu.: 141.0
##  Max.   :71567   Max.   :6616.0
```

```
# Maximum and minimum number of ratings for users

paste('The most raters of users rated ', max(edx_users$num),
      'movies, and the least raters rated only ', min(edx_users$num))
```

```
## [1] "The most raters of users rated  6616 movies, and the least raters rated only  10"
```

```
# Average number of rating for users

paste('Users in average rated around ', round(mean(edx_users$num)), ' movies.')
```

```
## [1] "Users in average rated around  129  movies."
```

The number of unique users in the edx dataset was 69878.

The variable edx_users was created to filter the unique users and learn about their characteristics. It has been found that the minimum number of ratings per user was 10, while the maximum number of ratings per user was 6616 ratings, and on average each user rated around 129 movies.

c. Movies

```
# Counting and displaying the number of unique movies
paste('There are ', n_distinct(edx$movieId), 'different movies')
```

```
## [1] "There are  10677 different movies"
```

```
# Calculating and displaying the number of movies with 0 ratings
paste('There are ', edx %>% filter(rating == 0) %>% tally(), 'movies with 0 ratings')
```

```
## [1] "There are  0 movies with 0 ratings"
```

```
# Calculating and displaying the number of movies with ratings = 3
paste('There are ', edx %>% filter(rating == 3) %>% tally(), 'movies with 3 stars ratings')
```

```
## [1] "There are  2121240 movies with 3 stars ratings"
```

```r
# Counting and displaying the different number of movies in each genres
paste('Drama has ', edx %>% filter(str_detect(genres,"Drama")) %>% nrow(), 'movies')
```

```
## [1] "Drama has  3910127 movies"
```

```r
paste('Comedy has ', edx %>% filter(str_detect(genres,"Comedy")) %>% nrow(), 'movies')
```

```
## [1] "Comedy has  3540930 movies"
```

```r
paste('Thriller has ', edx %>% filter(str_detect(genres,"Thriller")) %>% nrow(), 'movies')
```

```
## [1] "Thriller has  2325899 movies"
```

```r
paste('Romance has ', edx %>% filter(str_detect(genres,"Romance")) %>% nrow(), 'movies')
```

```
## [1] "Romance has  1712100 movies"
```

```r
# Selecting the movie that has the greatest number of ratings
edx %>% group_by(movieId, title) %>%
        summarize(number = n()) %>%
        arrange(desc(number))
```

```
## # A tibble: 10,677 x 3
## # Groups:   movieId [10,677]
##    movieId title                                                        number
##      <dbl> <chr>                                                         <int>
##  1     296 Pulp Fiction (1994)                                           31362
##  2     356 Forrest Gump (1994)                                           31079
##  3     593 Silence of the Lambs, The (1991)                              30382
##  4     480 Jurassic Park (1993)                                          29360
##  5     318 Shawshank Redemption, The (1994)                              28015
##  6     110 Braveheart (1995)                                             26212
##  7     457 Fugitive, The (1993)                                          25998
##  8     589 Terminator 2: Judgment Day (1991)                             25984
##  9     260 Star Wars: Episode IV - A New Hope (a.k.a. Star Wars) (1977)  25672
## 10     150 Apollo 13 (1995)                                              24284
## # ... with 10,667 more rows
```

```
# Selecting the movie genres that has the greatest number of ratings
edx %>% group_by(movieId, genres) %>%
        summarize(number = n()) %>%
        arrange(desc(number))
```

```
## # A tibble: 10,677 x 3
## # Groups:   movieId [10,677]
##    movieId genres                              number
##      <dbl> <chr>                                <int>
##  1     296 Comedy|Crime|Drama                   31362
##  2     356 Comedy|Drama|Romance|War             31079
##  3     593 Crime|Horror|Thriller                30382
##  4     480 Action|Adventure|Sci-Fi|Thriller  29360
##  5     318 Drama                                28015
##  6     110 Action|Drama|War                     26212
##  7     457 Thriller                             25998
##  8     589 Action|Sci-Fi                        25984
##  9     260 Action|Adventure|Sci-Fi              25672
## 10     150 Adventure|Drama                      24284
## # ... with 10,667 more rows
```

It has been found that all movies in the dataset has a rating, and there were 10677 unique movies. Various genres were explored, and it has been found that the most popular genres was "Comedy|Crime|Drama" as it has the most ratings. The movie "Pulp Fiction" was on top of the list.

d. Genres

```
# Total number of genres, it is noted that some movies are tagged with more than one genres

edx %>% group_by(genres) %>%
  summarize(number=n()) %>%
  arrange(desc(number))
```

```
## # A tibble: 797 x 2
##    genres                    number
##    <chr>                      <int>
##  1 Drama                     733296
##  2 Comedy                    700889
##  3 Comedy|Romance            365468
##  4 Comedy|Drama              323637
##  5 Comedy|Drama|Romance      261425
##  6 Drama|Romance             259355
##  7 Action|Adventure|Sci-Fi   219938
##  8 Action|Adventure|Thriller 149091
##  9 Drama|Thriller            145373
## 10 Crime|Drama               137387
## # ... with 787 more rows
```

```r
# Count the sum number of genres per each movie

edx %>% group_by(count=str_count(genres, fixed("|")), genres=genres) %>%
        summarize(number= n()) %>%
        arrange(desc(count))
```

```
## # A tibble: 797 x 3
## # Groups:   count [8]
##     count genres                                                    number
##     <int> <chr>                                                      <int>
##  1      7 Action|Adventure|Comedy|Drama|Fantasy|Horror|Sci-Fi|Thriller    256
##  2      6 Adventure|Animation|Children|Comedy|Crime|Fantasy|Mystery     10975
##  3      6 Adventure|Animation|Children|Comedy|Drama|Fantasy|Mystery       355
##  4      6 Adventure|Animation|Children|Comedy|Fantasy|Musical|Romance     515
##  5      5 Action|Adventure|Animation|Children|Comedy|Fantasy              187
##  6      5 Action|Adventure|Animation|Children|Comedy|IMAX                  66
##  7      5 Action|Adventure|Animation|Children|Comedy|Sci-Fi               600
##  8      5 Action|Adventure|Animation|Drama|Fantasy|Sci-Fi                 239
##  9      5 Action|Adventure|Children|Comedy|Fantasy|Sci-Fi                2832
## 10      5 Action|Adventure|Children|Crime|Mystery|Thriller                 62
## # ... with 787 more rows
```

In exploring the genres variable, it has been noted that some movies were tagged with more than one genre, therefore it was better understood to summarize the movies by all its tagged genres. In which the top of this summary included:

- Action|Adventure|Comedy|Drama|Fantasy|Horror|Sci-Fi|Thriller

e. Ratings

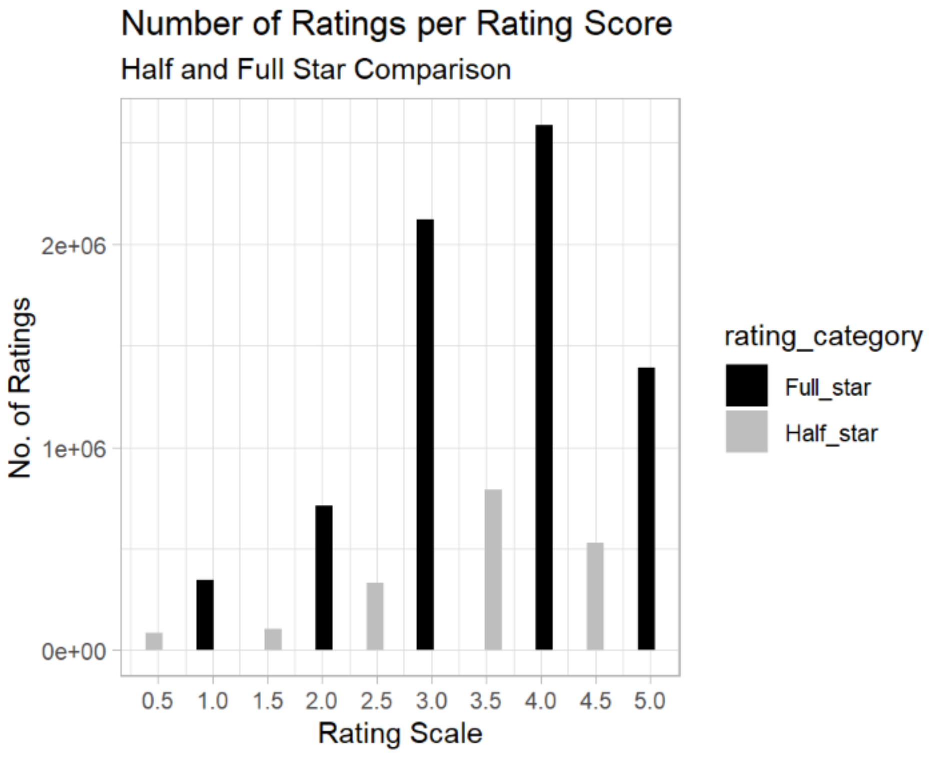
```r
# Sorting the five most given ratings in the edx dataset
head(sort(-table(edx$rating)),5)
```

```
##
##        4        3        5      3.5        2
## -2588430 -2121240 -1390114  -791624  -711422
```

```r
# Numerically comparing the half and full star ratings
table(edx$rating)
```

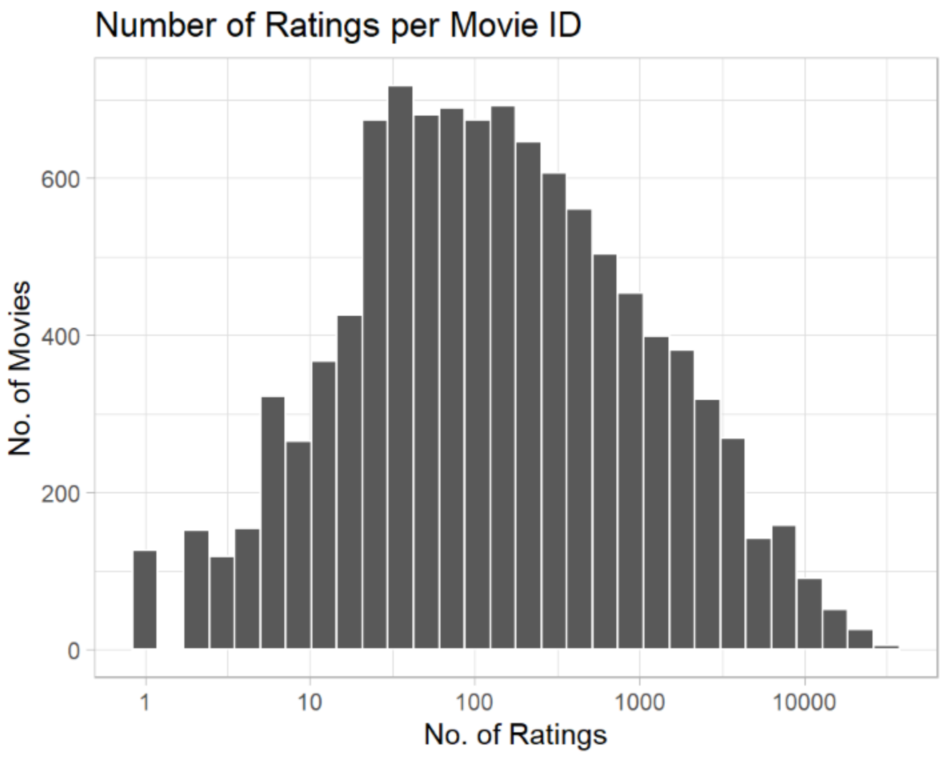
```
##
##     0.5       1     1.5       2     2.5       3     3.5       4     4.5       5
##   85374  345679  106426  711422  333010 2121240  791624 2588430  526736 1390114
```

The difference between full and half star ratings were summarized in the below graph:



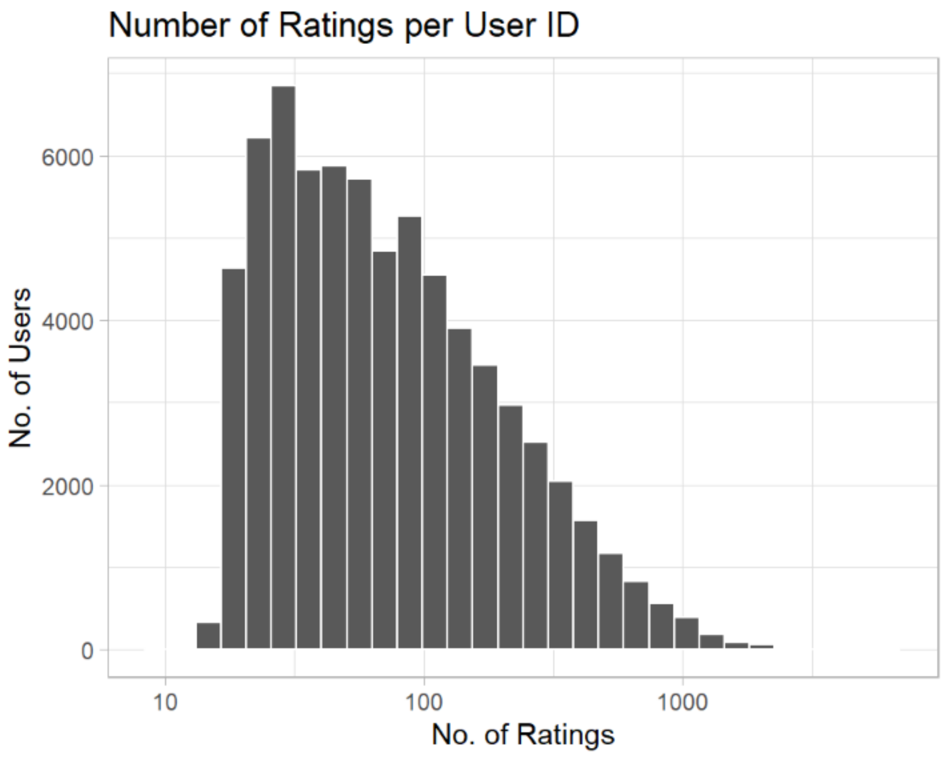
It is noted that full star ratings were much more common among users than half star ratings.

In the next graph, number of ratings was plotted against number of movies:

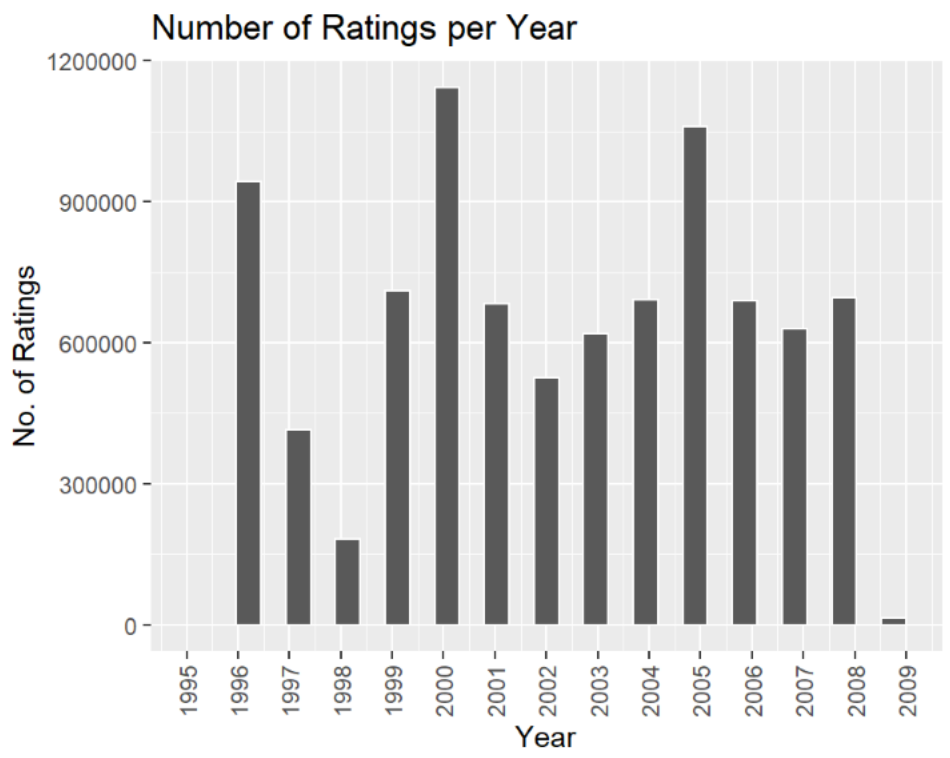


It is apparent that some movies had more ratings than others. Interestingly, around 120 movies had only 1 rating.

Next, number of ratings was plotted against number of users:

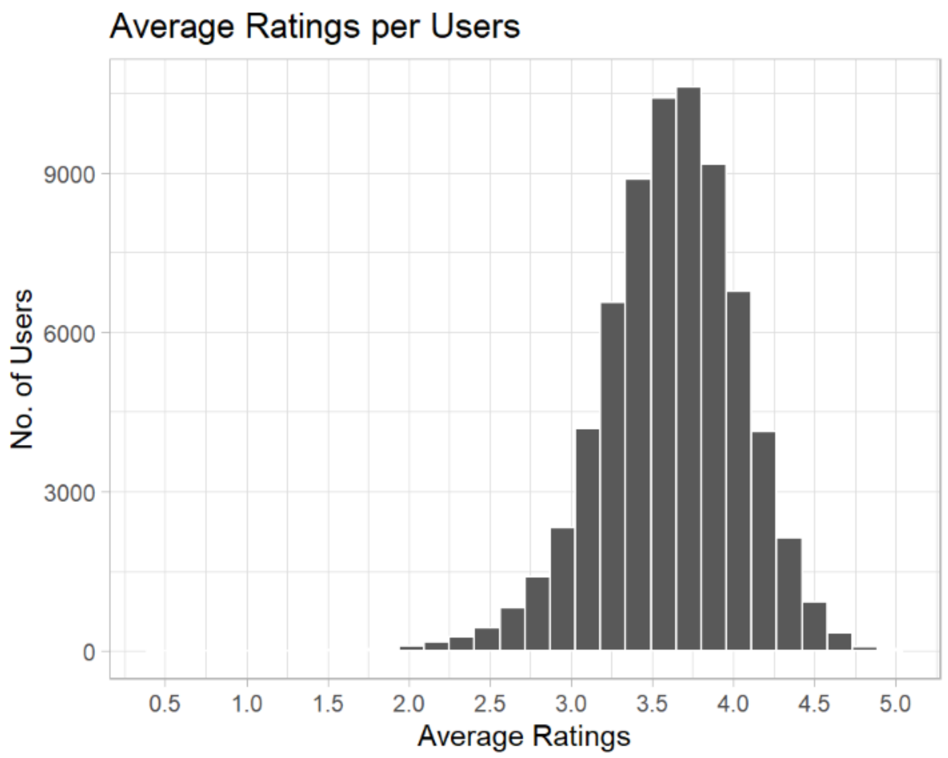


In the above graph it is noted that most raters rated around 30 to 100 movies.



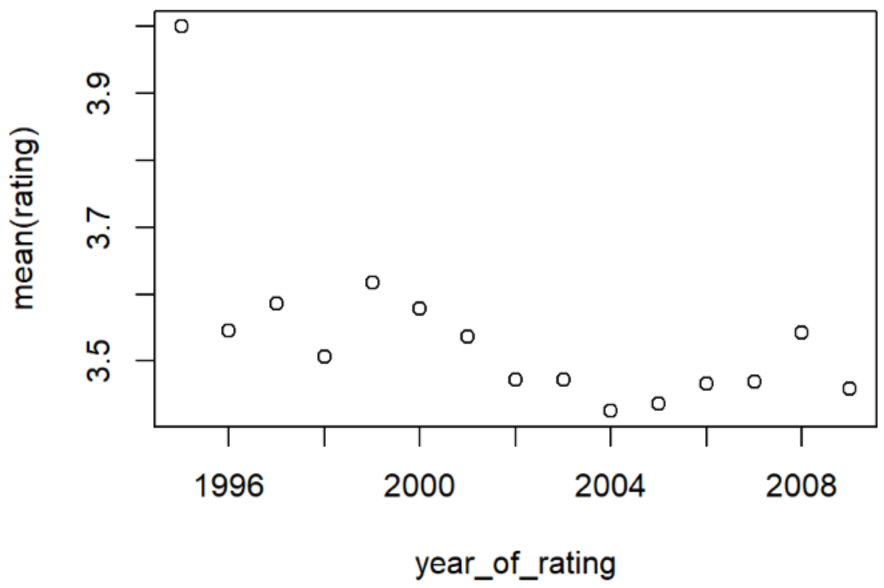
Also, number of ratings per year was visualized but it has been noted that there is no specific behavior for this distribution as it was varying over the years.

Next, the average number of ratings was plotted against the number of users:



It is noted that on average majority of raters gave a score of between 3 and 4 stars ratings for the movies they watched. However, it is a normally distributed graph.

Finally, the average ratings per year was visualized:



It is noted that in the early years of the data collection, average ratings were not remarkably close this might be because rating behavior was not common among users at that period.

III. Insights Gained

Based on the performed exploration strategies and the concluded data properties, some observations were interesting:

- Some movies were more rated than others.
- Rating scores by users in average was 3 and 4 starts ratings.
- Year of rating did not have a major impact of the number of ratings observed.
- Most raters rated between 30 to 100 movies.
- Around 125 movies had only one rating.
- Full star ratings were more common than half stars, however it is noted that half star rating were introduced in 2003.
- Some movies were tagged in more than one genre therefore the filter applied considered the total number of genres per each movie.
- Average ratings were more volatile in the early years of data collection, it may be due to lack of rating platforms or the uncommon behavior of movie ratings per users.

Due to limited resources to perform adequate analysis taking into consideration all the factors available in this data set, it has been decided to pick three factors that have visibly impacted the distribution of ratings. In this case, movies, users, and genre were selected to build data models and predictions in order to be considered in the final machine learning algorithm.

IV. Modeling Approach

Data modelling was based on three various modelling approaches:

1. Simple prediction based on average ratings

The average prediction model is based on the formula:

$$\hat{y} = \mu + \epsilon_{\mathrm{u,i}}$$

```
avg <- mean(edx$rating)
rmse_simple_model <- RMSE(validation$rating, avg)
rmse_simple_model
```

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

```
# Check this model results and save it in the data.frame
rmse_results <- data_frame(Model_Method = "Average Prediction Model",
                           RMSE = rmse_simple_model)

# Generate RMSE Result Table using kable() function
rmse_results %>% knitr::kable()
```

RMSE Value:

Model_Method	RMSE
Average Prediction Model	1.061202

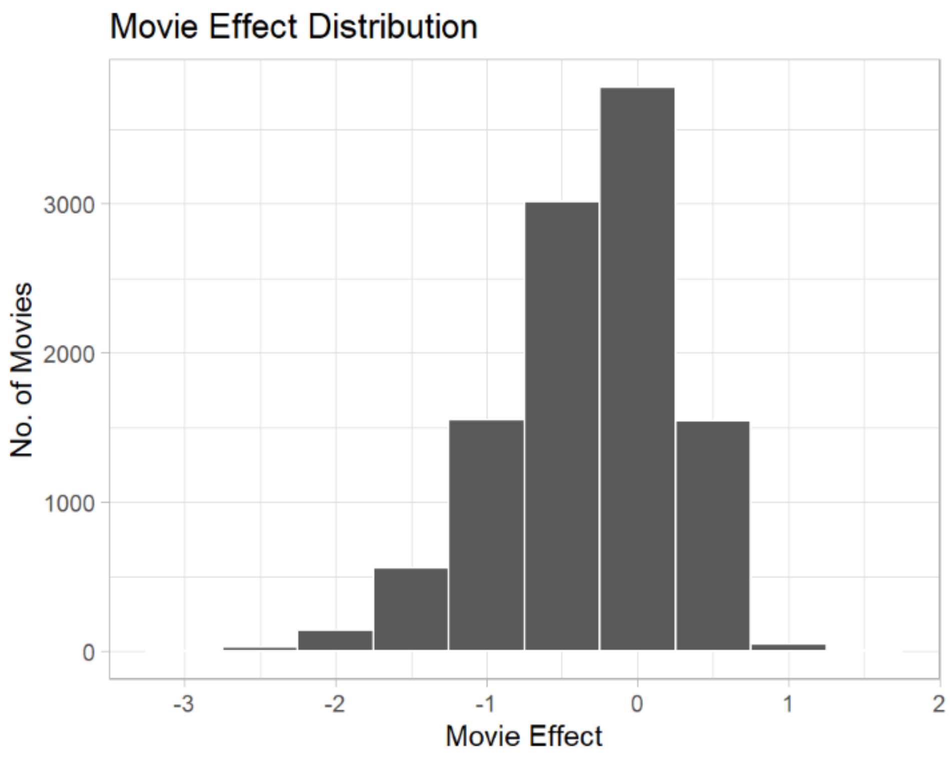
RMSE value considered not acceptable. Therefore, an enhanced model is necessary.

2. Linear model considering movie effect

The movie effect model is based on the formula:

$$\hat{y} = \mu + \boldsymbol{b_i} + \epsilon_{\mathrm{u,i}}$$

```r
bi <- edx %>%
      group_by(movieId) %>%
      summarize(b_i = mean(rating - avg))

bi %>% ggplot(aes(x = b_i)) +
       geom_histogram(bins=10, col = I("white")) +
       labs(title="Movie Effect Distribution",
            x="Movie Effect",
            y="No. of Movies") +
        theme_light()
```



```
# Check this model results and save it in the dataframe

predicted_ratings <- avg +  validation %>%
                    left_join(bi, by='movieId') %>%
                    pull(b_i)

rmse_movie_model <- RMSE(predicted_ratings, validation$rating)

rmse_results <- bind_rows(rmse_results,
                          data_frame(Model_Method="Movie Effect Model",
                                     RMSE = rmse_movie_model ))

# Update RMSE Result Table
rmse_results %>% knitr::kable()
```

RMSE Value:

Model_Method	RMSE
Average Prediction Model	1.0612018
Movie Effect Model	0.9439087

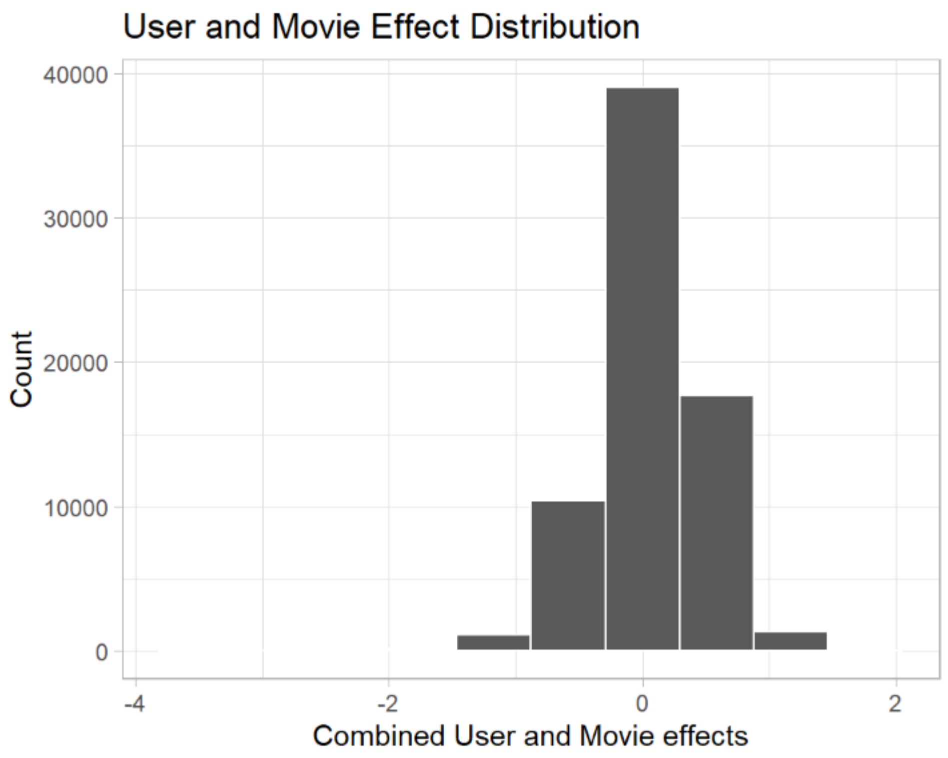
RMSE value was improved, but still we need a better model to achieve the target of this project.

3. Linear model considering both movie and user effects

The movie-user effect model is based on the formula:

$$\hat{y} = \mu + \boldsymbol{b_i} + \boldsymbol{b_u} + \epsilon_{\mathrm{u,i}}$$

```
bu <- edx %>%
      left_join(bi, by='movieId') %>%
      group_by(userId)%>%
      summarize(b_u = mean(rating - avg - b_i))

bu %>% ggplot(aes(x = b_u)) +
  geom_histogram(bins=10, col = I("white")) +
  labs(title="User and Movie Effect Distribution",
       x="Combined User and Movie effects",
       y="Count") +
  theme_light()
```



```r
# Check and store model results
predicted_ratings <- validation %>%
                     left_join(bi, by='movieId') %>%
                     left_join(bu, by='userId') %>%
                     mutate(pred = avg + b_i + b_u) %>%
                     pull(pred)

rmse_movie_user_model <- RMSE(predicted_ratings, validation$rating)

rmse_results <- bind_rows(rmse_results,
                          data_frame(Model_Method="Movie and User Effect model",
                                     RMSE = rmse_movie_user_model))

# Update RMSE Result Table
rmse_results %>% knitr::kable()
```

RMSE Value:

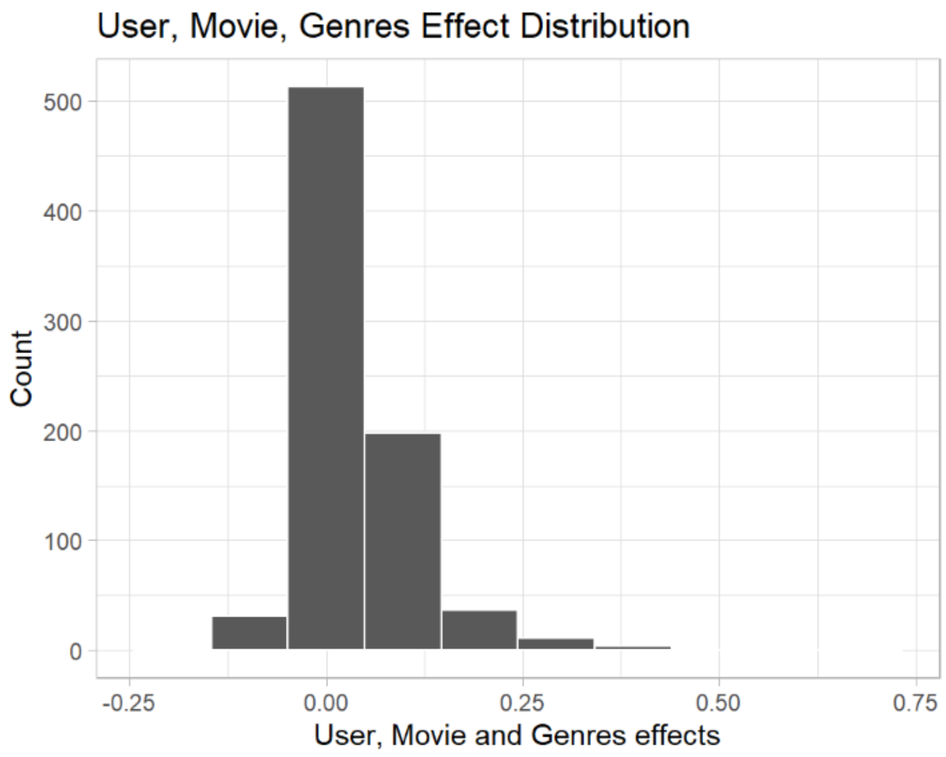
Model_Method	RMSE
Average Prediction Model	1.0612018
Movie Effect Model	0.9439087
Movie and User Effect model	0.8653488

RMSE value was improved after combining two predictors (user and movie effects), as per the exploration of the dataset genres had an impact on the overall rating therefore it will be considered in the next model.

4. Linear model considering movie, user and genres effects

$$\hat{y} = \mu + \boldsymbol{b_i} + \boldsymbol{b_u} + \boldsymbol{b_g} + \epsilon_{\mathrm{u,i}}$$

```r
bg <- edx %>%
  left_join(bi, by='movieId') %>%
  left_join(bu, by='userId') %>%
  group_by(genres) %>%
  summarize(b_g = mean(rating - avg - b_i - b_u))

bg %>% ggplot(aes(x = b_g)) +
  geom_histogram(bins=10, col = I("white")) +
  labs(title="User, Movie, Genres Effect Distribution",
       x="User, Movie and Genres effects",
       y="Count") +
  theme_light()
```



```r
# Check and store model results
predicted_ratings <- validation %>%
  left_join(bi, by='movieId') %>%
  left_join(bu, by='userId') %>%
  left_join(bg, by='genres') %>%
  mutate(pred = avg + b_i + b_u + b_g) %>%
  pull(pred)

rmse_movie_user_genre_model <- RMSE(predicted_ratings, validation$rating)

rmse_results <- bind_rows(rmse_results,
                          data_frame(Model_Method="Movie, User and Genres Effect model",
                                     RMSE = rmse_movie_user_genre_model))

# Update RMSE Result Table
rmse_results %>% knitr::kable()
```

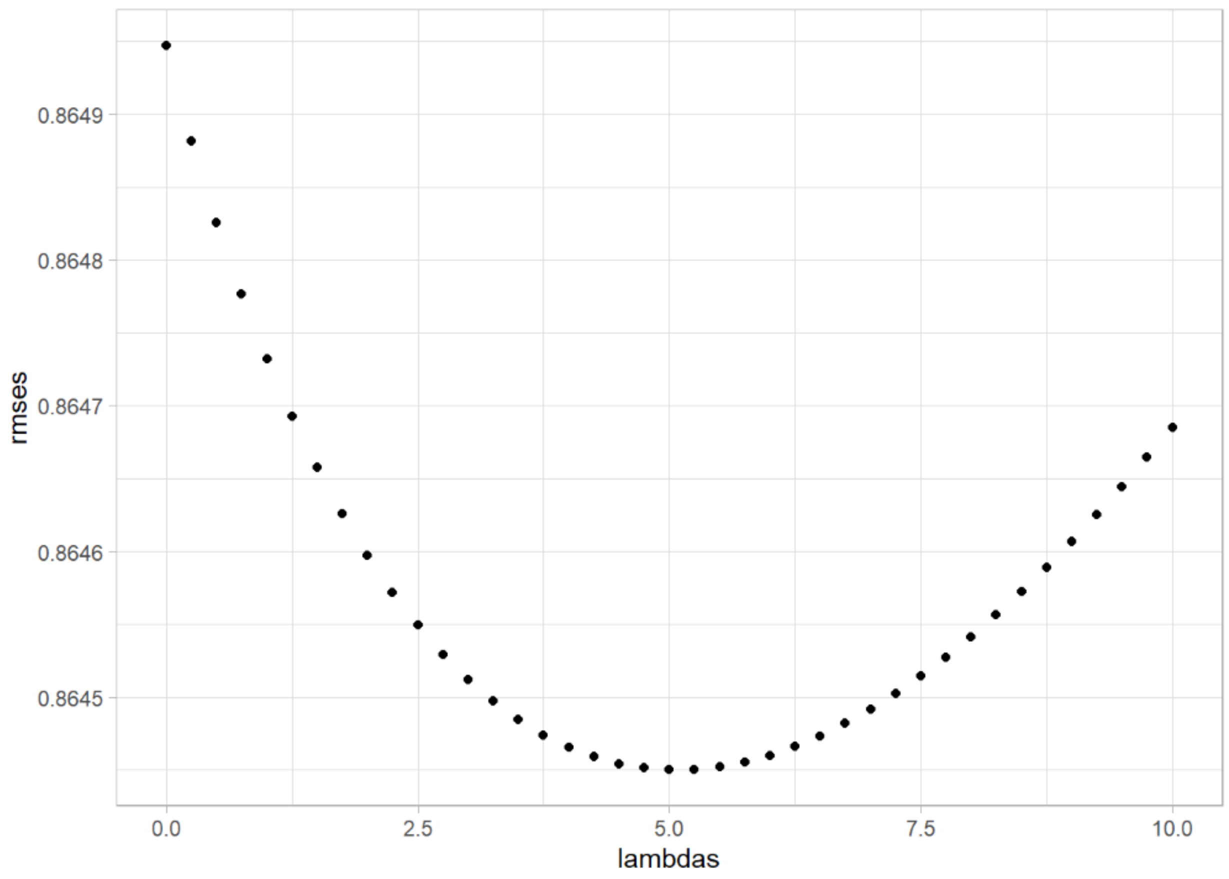
Model_Method	RMSE
Average Prediction Model	1.0612018
Movie Effect Model	0.9439087
Movie and User Effect model	0.8653488
Movie, User and Genres Effect model	0.8649469

RMSE was improved after adding the genres effect. In the next step an account for the total variability of the effect sizes was considered through penalizing large estimates that come from small sample in the considered factors.

5. Model Regularization

The models generated previously has been regularized through a tuning factor (lambda) in order to improve the RMSE value, in defining lambda a sequence of values was considered and then the optimal lambda minimizing RMSE to lowest possible value was chosen and applied on the validation data set as a final step.

```r
## Considering adding a tuning factor for regularization (lambda)
## to the Movie, User and Genres Model to get a better reduced RMSE value

# lambda is a tuning parameter
# Use cross-validation to choose it.
lambdas <- seq(0, 10, 0.25)

# For each lambda,find b_i, b_u and b_g followed by rating prediction & testing
# Using cross validation method

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))

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

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

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

# Plot rmses vs lambdas to select the optimal lambda
qplot(lambdas, rmses)+
  theme_light()
```



```
# Find optimal lambda
lambda <- lambdas[which.min(rmses)]
lambda
```

```
## [1] 5
```

```
# Check and store model results
rmse_results <- bind_rows(rmse_results,
                          data_frame(Model_Method="Regularized Movie, User and Genres Effect Model",
                                     RMSE = min(rmses)))
# Update RMSE Result Table
rmse_results %>% knitr::kable()
```

Model_Method	RMSE
Average Prediction Model	1.0612018
Movie Effect Model	0.9439087
Movie and User Effect model	0.8653488
Movie, User and Genres Effect model	0.8649469
Regularized Movie, User and Genres Effect Model	0.8644501

Results

The evaluation of all the generated data models considered a loss function: RMSE to pick the best performing model. RMSE was calculated through the formula:

$$\sqrt{\frac{1}{\mathrm{N}}\sum_{\mathrm{u,i,g}}(\hat{\mathrm{y}}_{\mathrm{u,i,g}}\ \text{-}\ \mathrm{y}_{\mathrm{u,i,g}})^2}$$

The three different approaches of the developed models are summarized in the RMSE result table below:

Model_Method	RMSE
Average Prediction Model	1.0612018
Movie Effect Model	0.9439087
Movie and User Effect model	0.8653488
Movie, User and Genres Effect model	0.8649469
Regularized Movie, User and Genres Effect Model	0.8644501

As shown, the best model was the regularized movie, user and genres effect, resulting in the lowest RMSE = 0.8644501. The optimal tuning factor value was '5' in which it minimized the RMSE value to the lowest possible value.

The final data model formula was:

$$\hat{\mathrm{y}} = \mu + \boldsymbol{b_i} + \boldsymbol{b_u} + \ \boldsymbol{b_g} + \epsilon_{\mathrm{u,i}}$$

Conclusion

I. Summary of the report

In conclusion, based on the available resources the best machine algorithm for this project took into consideration three different factors of the MovieLens dataset: Users, Movies, and Genres, and used a regularization strategy to enhance the data model overall performance to build an effective recommendation system that predict movie ratings for users based on movie, user, and movie genre.

II. Limitation and Future Work

This algorithm may be further enhanced to achieve better results. More complex algorithms can be generated and evaluated through better processing power, and other factors in dataset may also lead to better results. Due to available processing power only three variables were selected and evaluated.