HarvardX PH125_9x Data Science Capstone Report for Prediction of User rating by using MovieLens Dataset

Kingsley Sam

7 April 2021

Contents

1. Introduction
   This report submission is part of Edx Harvardx PH125.9 Data Science Capstone for the Movielens Project. In the streaming platform, movie recommendation system is used to retain their customers on the platform by keeping them watching the videos or movies. When a favourite or relevant movie was recommended to the customer, the customer was likely to watch for a longer time, to spend more time on the streaming platform, so that the platform can enhance customer loyalty and usage frequency. Therefore prediction performance of a movie recommendation system is important to a streaming platform. In this report, we try to build a prediction algorithm for the user rating by using the dataset and initial codes provided in the following.

1.1 Data Overview
In the dataset, there are 10677 movies rated by 69878 users. 9000055 observations in the dataset with 6 variables including "userId", "movieId", "rating", "timestamp", "title", and "genres".
For the "rating", users could rate the movie from 0.5 to 4.0 with 0.5 interval. The data was briefly overviewed by using the following codes.

```
# 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  3.0.6     v dplyr   1.0.3
## v tidyr   1.1.2     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(tidyverse)
library(caret)
library(data.table)

# 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")

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
```

```
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")
```

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

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

edx %>% summarize(n_movies = n_distinct(movieId))
```

```
##   n_movies
## 1    10677
```

```
#load(file='Workspace_Capstone_03_final.RData')
paste('The edx dataset has',nrow(edx),'rows and',ncol(edx),'columns.')
```

```
## [1] "The edx dataset has 9000055 rows and 6 columns."
```

```
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 83
##  $ title    : chr  "Boomerang (1992)" "Net, The (1995)" "Outbreak (1995)" "Stargate (1994)" ...
##  $ genres   : chr  "Comedy|Romance" "Action|Crime|Thriller" "Action|Drama|Sci-Fi|Thriller" "Action|A
##  - attr(*, ".internal.selfref")=<externalptr>
```

```
#load(file='Workspace_Capstone_03_final.RData')
paste('The edx dataset has',nrow(edx),'rows and',ncol(edx),'columns.')
```

```
## [1] "The edx dataset has 9000055 rows and 6 columns."
```

```
paste(sum(edx$rating == 0), 'ratings with 0 were given and',
      sum(edx$rating == 3),'ratings with 3')
```

```
## [1] "0 ratings with 0 were given and 2121240 ratings with 3"
```

```
drama <- edx %>% filter(str_detect(genres,"Drama"))
comedy <- edx %>% filter(str_detect(genres,"Comedy"))
thriller <- edx %>% filter(str_detect(genres,"Thriller"))
romance <- edx %>% filter(str_detect(genres,"Romance"))

paste('Drama has',nrow(drama),'movies')
```

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

```
paste('Comedy has',nrow(comedy),'movies')
```

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

```
paste('Thriller has',nrow(thriller),'movies')
```

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

```
paste('Romance has',nrow(romance),'movies')
```

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

```
edx %>% group_by(title) %>% summarise(number = n()) %>%
  arrange(desc(number))
```

```
## # A tibble: 10,676 x 2
##    title                                                     number
##    <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 10,666 more rows
```

```
edx %>% group_by(rating) %>% summarise(number = n()) %>%
  arrange(desc(number))
```

```
## # A tibble: 10 x 2
##    rating  number
##     <dbl>   <int>
##  1    4   2588430
##  2    3   2121240
##  3    5   1390114
##  4    3.5  791624
##  5    2    711422
##  6    4.5  526736
##  7    1    345679
##  8    2.5  333010
##  9    1.5  106426
## 10    0.5   85374
```

```
head(sort(-table(edx$rating)),5)
```

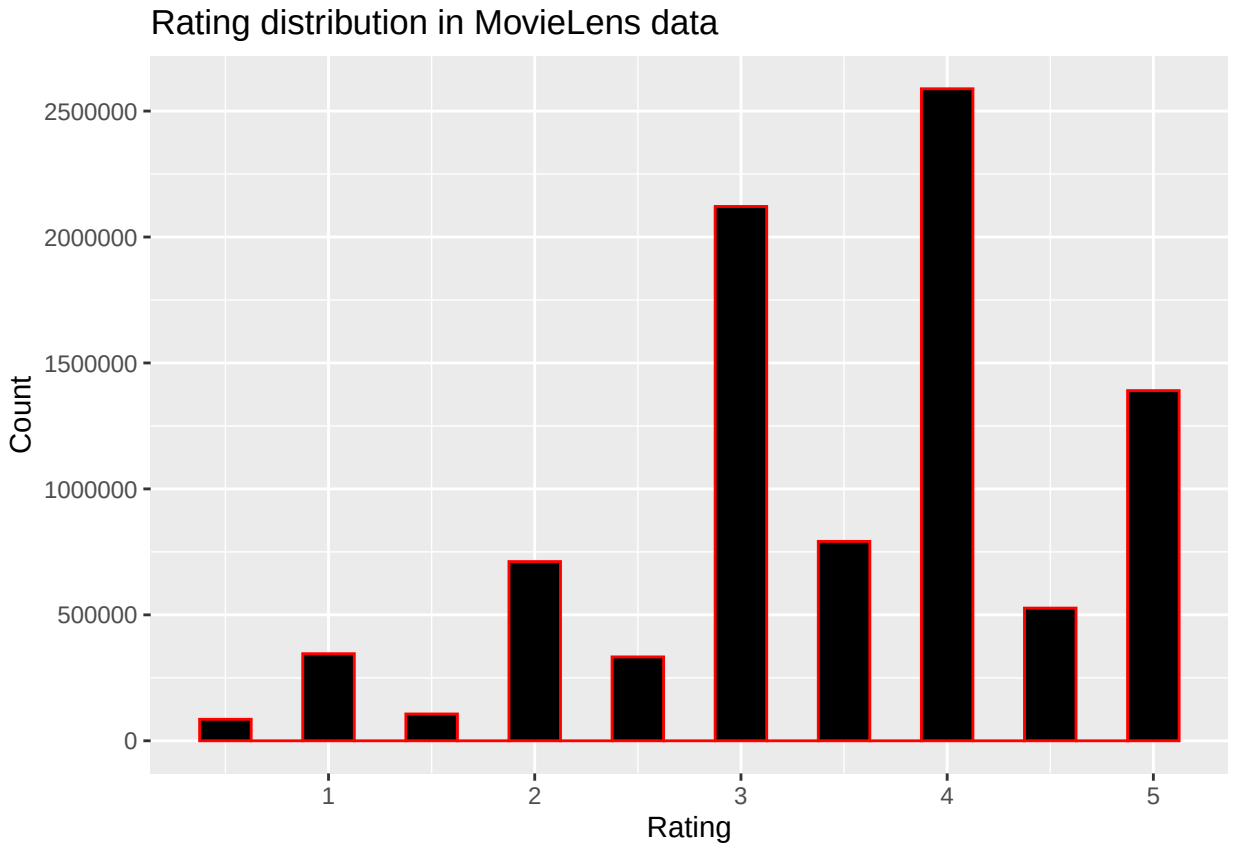
```
##
##        4        3        5      3.5        2
## -2588430 -2121240 -1390114  -791624  -711422
```

1.2 Data distribution

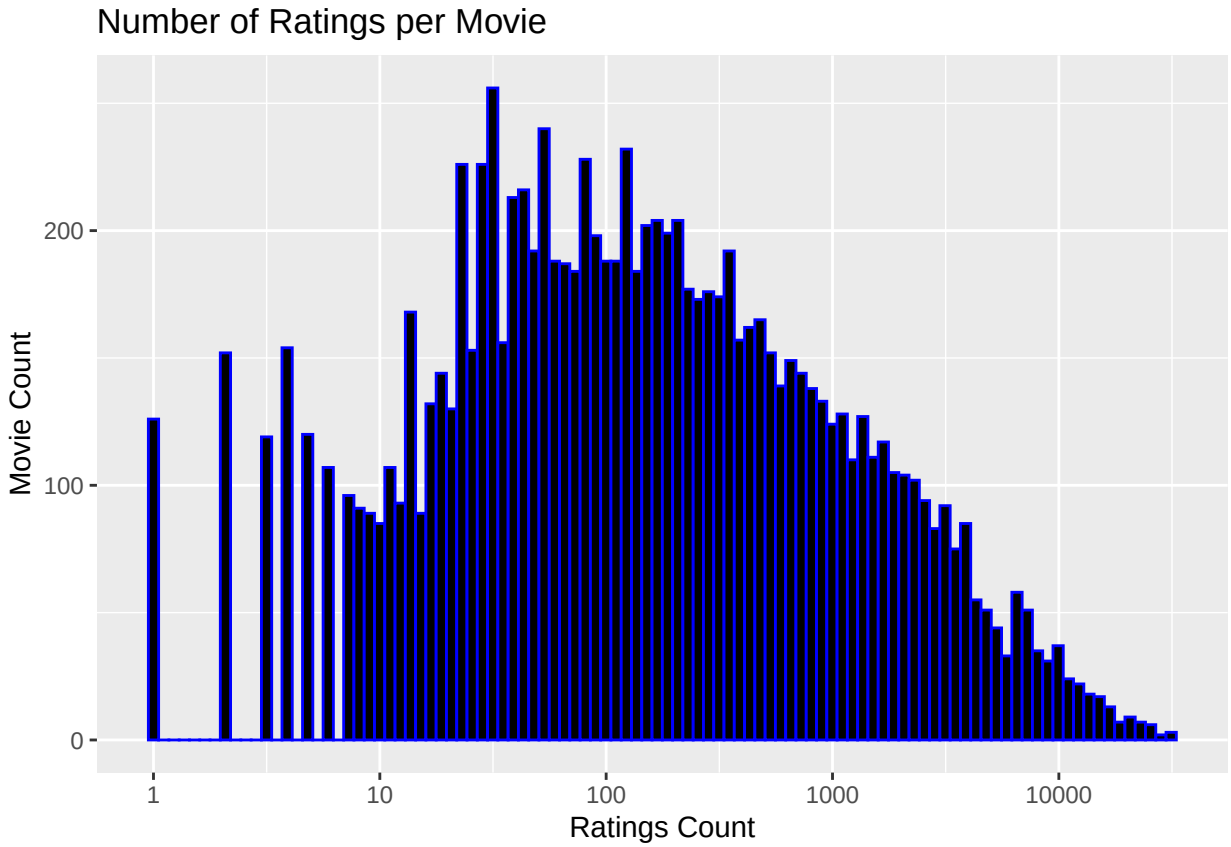
To further analyse the dataset, the distribution of variables were plotted by using ggplot function. As illustrated in the following histogram 1 created by ggplot function, ratings with‘.5” including 0.5, 1.5, 2.5, 3.5, and 4.5 are much less than integer rate e.g 1, 2, 3,4 and 5. This reveals the tendency of users’ rating towards

integral options. Moreover, the rating distribution is skewed to the right, and more users tend to give ratings with "4" in general.

```
#Histogram 1:
# Ratings distribution in MovieLens data
edx %>%
  ggplot(aes(rating)) +
  geom_histogram(binwidth = 0.25, color = "red", fill = "black") +
  xlab("Rating") +
  scale_y_continuous(breaks = c(seq(0, 3000000, 500000))) +
  ylab("Count") +
  ggtitle("Rating distribution in MovieLens data")
```

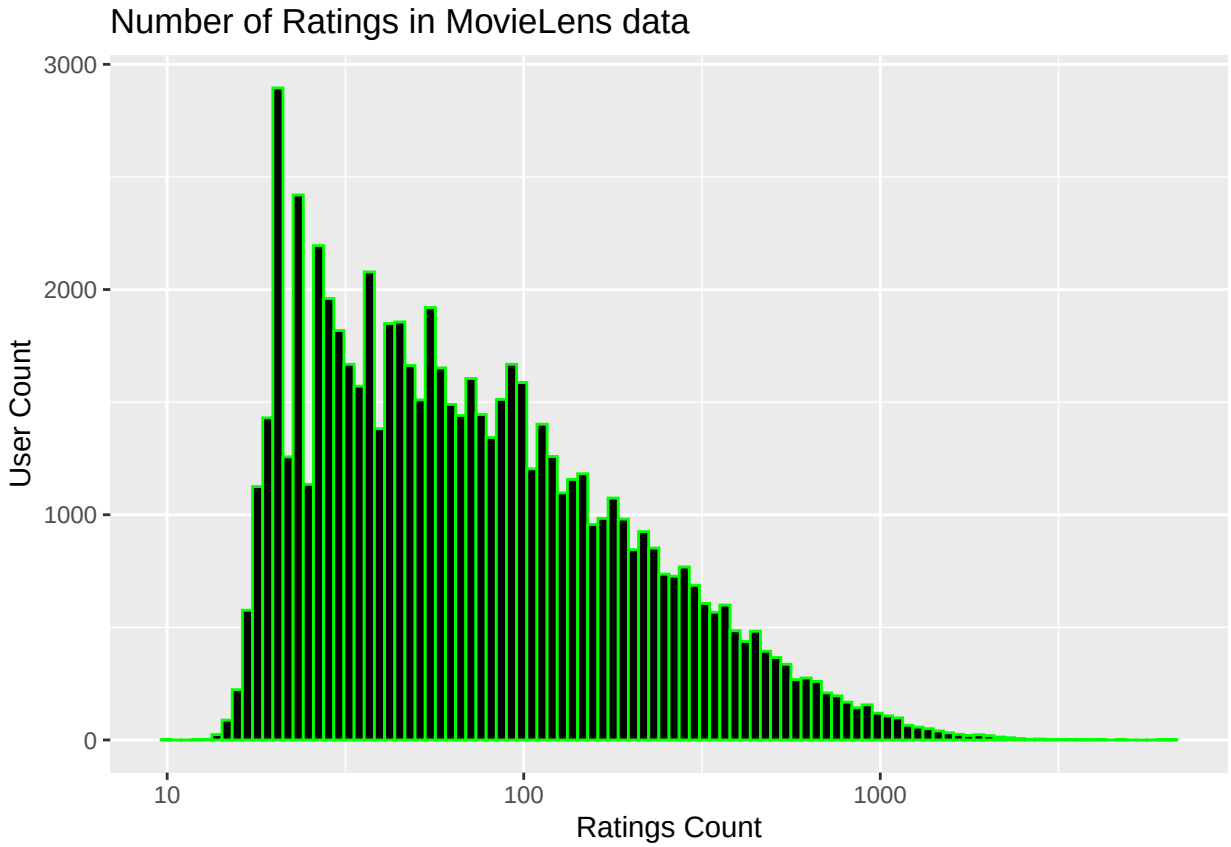



```
#Histogram 2:
  # Number of ratings per Movie in MovieLens data
edx %>%
  count(movieId) %>%
  ggplot(aes(n)) +
  geom_histogram(bins = 100, color = "blue", fill = "black") + xlab("Ratings Count") +
  scale_x_log10() +
  ylab("Movie Count") +
  ggtitle("Number of Ratings per Movie")
```



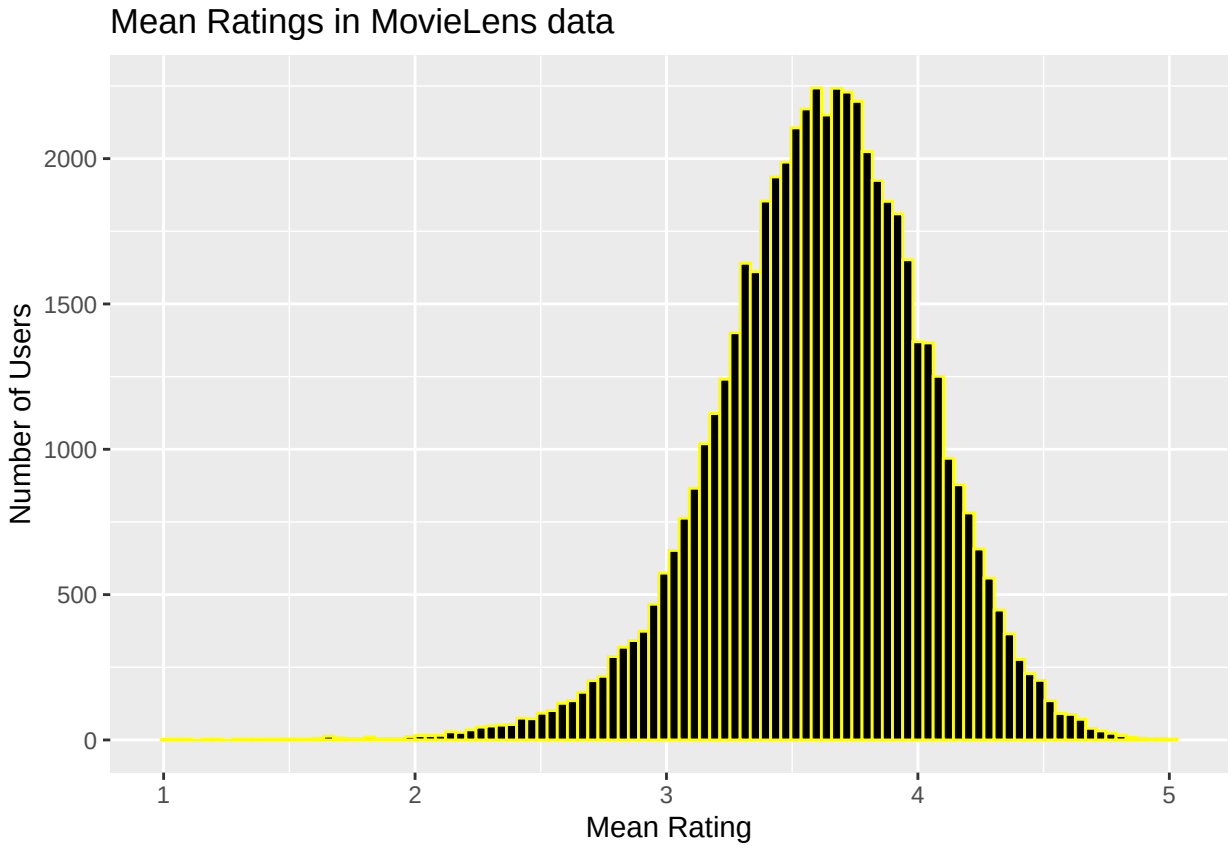
The Histogram 2 illustrated that distribution of movie count against ratings count is also not even. Some movies have a higher ratings count which could be more than 10000, while some movies were seldom rated whose ratings count could be less than 10. This distribution may be attributed to the popularity and advertisement of the movie which made some movies famous. Intuitively, famous movies were watched by more users, so the famous movies would have a higher chance to be rated by more users.

```
#Histogram 3:
# Number of ratings given by users in MovieLens data
edx %>%
  count(userId) %>%
  ggplot(aes(n)) +
  geom_histogram(bins = 100, color = "green", fill = "black") + xlab("Ratings Count") +
  scale_x_log10() +
  ylab("User Count") +
  ggtitle("Number of Ratings in MovieLens data")
```



In histogram 3, it also illustrates a right skewed distribution of users count against rating counts. Some users tend to rate more frequently on the streaming platform than other users. To interpret the distribution, it shows that not every user contributes the same influence to the rating count.

```
#Histogram 4:
# Mean ratings given by user in MovieLens data
edx %>%
  group_by(userId) %>%
  filter(n() >= 30) %>%
  summarize(mean_rating = mean(rating)) %>% ggplot(aes(mean_rating)) +
  geom_histogram(bins = 100, color = "yellow", fill = "black") + xlab("Mean Rating") +
  ylab("Number of Users") +
  ggtitle("Mean Ratings in MovieLens data")
```



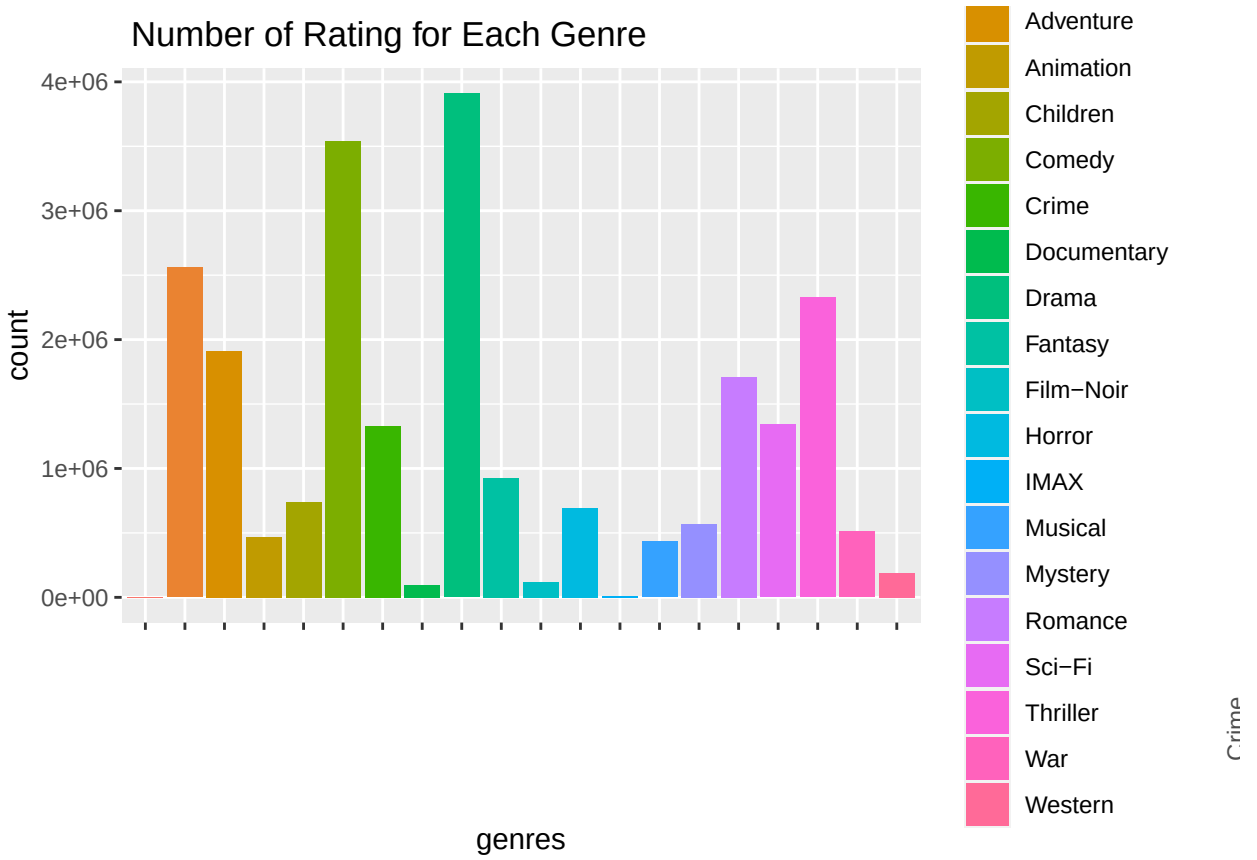

```
summary(edx)
```

```
##      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
##
##
##
```

In the histogram 4, the distribution of mean ratings given by users normally distributed and the mean rating is around 3.5 to 4.5 which may be one of the important factors in the prediction model to be analysed.

```
#Histogram 5:
#number of rating for each movie genres
edx %>% separate_rows(genres, sep = "\\|") %>%
  group_by(genres) %>%
  summarize(count = n()) %>%
  arrange(desc(count)) %>% ggplot(aes(genres,count)) +
  geom_bar(aes(fill =genres),stat = "identity")+
  labs(title = " Number of Rating for Each Genre")+
```

```
  theme(axis.text.x  = element_text(angle= 90, vjust = 50 ))
```



2. Data Analysis

2.1 Methods

The algorithms will use the Root Mean Squared Error (RMSE) to evaluate performance. It is a way to measure the difference between the value observed to the value predicted, and the goal is to get it as low as possible.The target RMSE for this project is lower than 0.86490 according to the course instruction.
There are 6 variables in the dataset, some of them may affect the rating. Initially, I created the prediction algorithm from the simplest model, which is Naive Model by containing the mean of the ratings. Afterward, I investigated different variables one by one by adding them into the model and compared the RMSE in order to understand their effects on the accuracy of model prediction. Therefore, we can try to obtain a final model with the lowest RMSE by multiple combinations of the variables.

```
##Data Partitioning
##partition the edx data set into 20 % for test set and 80% for the training set.
set.seed(1)
test_index <- createDataPartition(y = edx$rating, times = 1, p = 0.2, list = FALSE)
train_set <- edx[-test_index,]
test_set <- edx[test_index,]
```

2.2 Modelling

Model building and RMSE calculation

```
RMSE <- function(true_ratings, predicted_ratings){
  sqrt(mean((true_ratings - predicted_ratings)^2, na.rm = TRUE))
}
```

Model 1: Navie model

```
## Model 1: Navie model
Mu_1 <- mean(train_set$rating)
Mu_1
```

```
## [1] 3.512482
```

```
## Testing RMSE of Model 1 : Navie Model
Mu_1_rmse <- RMSE(validation$rating, Mu_1)
Mu_1_rmse
```

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

According to the data distribution as seen in Histogram 4, Mean ratings seems to be a fundamental factor to affect the prediction model as discussed previously.

Now, the Naive Model only consists of mean rating for the first trial. Mean rating was found to be 3.512482 out of 5. It is found that the RMSE of Naive model is 1.061202.

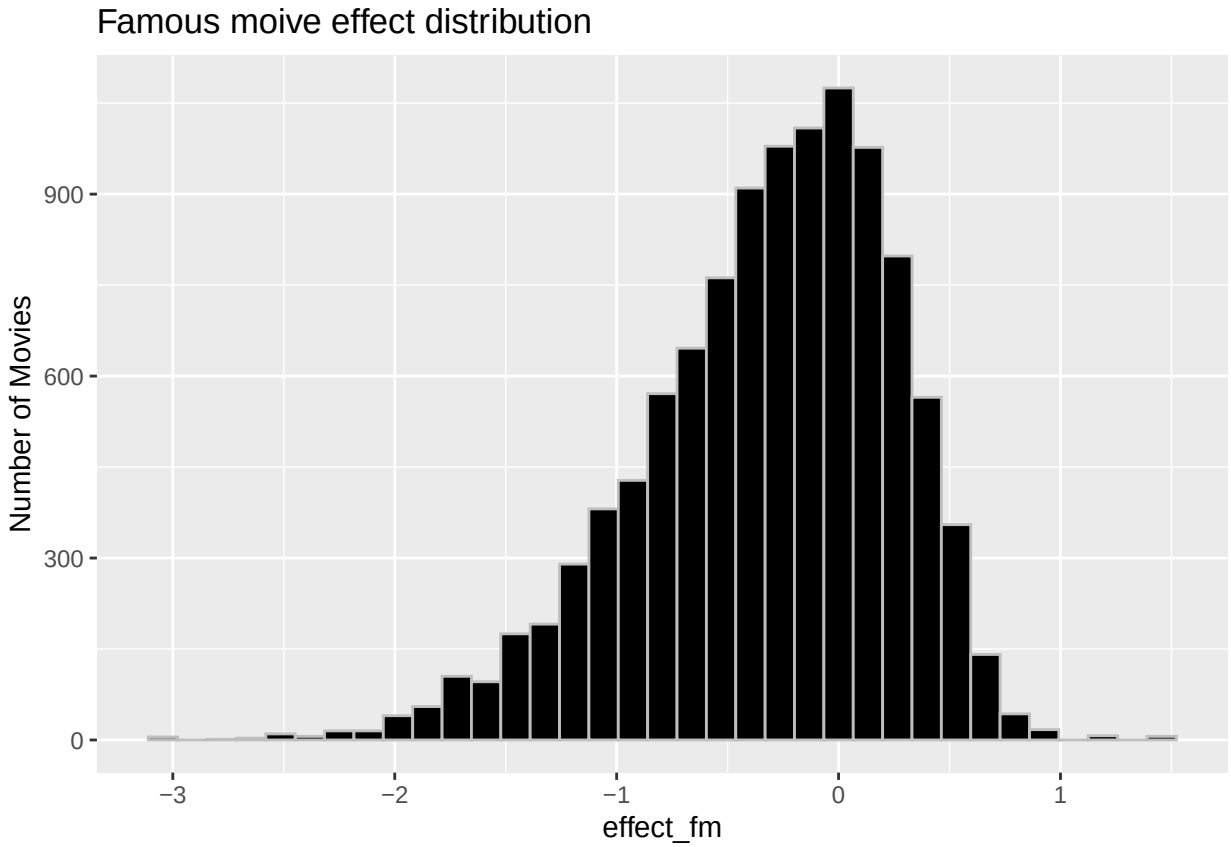
Model 2 : Famous Movie effect Model

According to the data distribution as seen in Histogram 2, some famous movies got higher rating count, which might be affected by how famous the movie was. Therefore, this famous movie effect ("effect_fm") may deviate the mean rating of each movie from the total mean rating of all movies. To assess the estimated deviation due to effect_fm, following code was applied. The left-skewed histogram in the following implied the negative effect exerted by the famous movie effect.

```
## Model 2 : Famous Movie effect Model

famous_movie_effect <- edx %>%
  group_by(movieId) %>%
  summarize(effect_fm = mean(rating - Mu_1))

famous_movie_effect %>%
  ggplot(aes(effect_fm)) +
  geom_histogram(bins = 35, color = "grey", fill = "black") + ylab("Number of Movies") +
  ggtitle("Famous moive effect distribution")
```




```
## Testing RMSE of Model 2 : Famous Movie effect Model
movie_effect_predictions <- validation %>%
  left_join(famous_movie_effect, by = "movieId")%>%
  mutate(prediction = Mu_1 + effect_fm)
Mu_2_rmse <- RMSE(validation$rating, movie_effect_predictions$prediction)
Mu_2_rmse
```

```
## [1] 0.9439087
```

Then, we add the famous movie effect to the naive model to calculate the RMSE as follows. The RMSE was improved compared to the Naive Model and it became 0.9439087.

Model 3: Famous Movie and User Effect Model

In the histogram 3, it is not a normal distribution. The histogram was also skewed to right which implied some users were more willing to rate. In the model 3, I add the User effect on the model 2 to assess the RMSE as follows. The RMSE was improved compared to the Model 2 and became 0.8653488.

```
## Model 3 : Famous movie and User effect Model
user_effect <- edx %>%
  left_join(famous_movie_effect, by = "movieId") %>% group_by(userId) %>%
  summarize(effect_u = mean(rating - Mu_1 - effect_fm))

## Testing RMSE of Model 3 : Famous movie and User effect Model
user_effect_predictions <- validation %>% left_join(famous_movie_effect, by = "movieId")%>%
  left_join(user_effect, by = "userId")%>%
  mutate(prediction = Mu_1 + effect_fm + effect_u)
Mu_3_rmse <- RMSE(validation$rating, user_effect_predictions$prediction)
Mu_3_rmse
```

```
## [1] 0.8653488
```

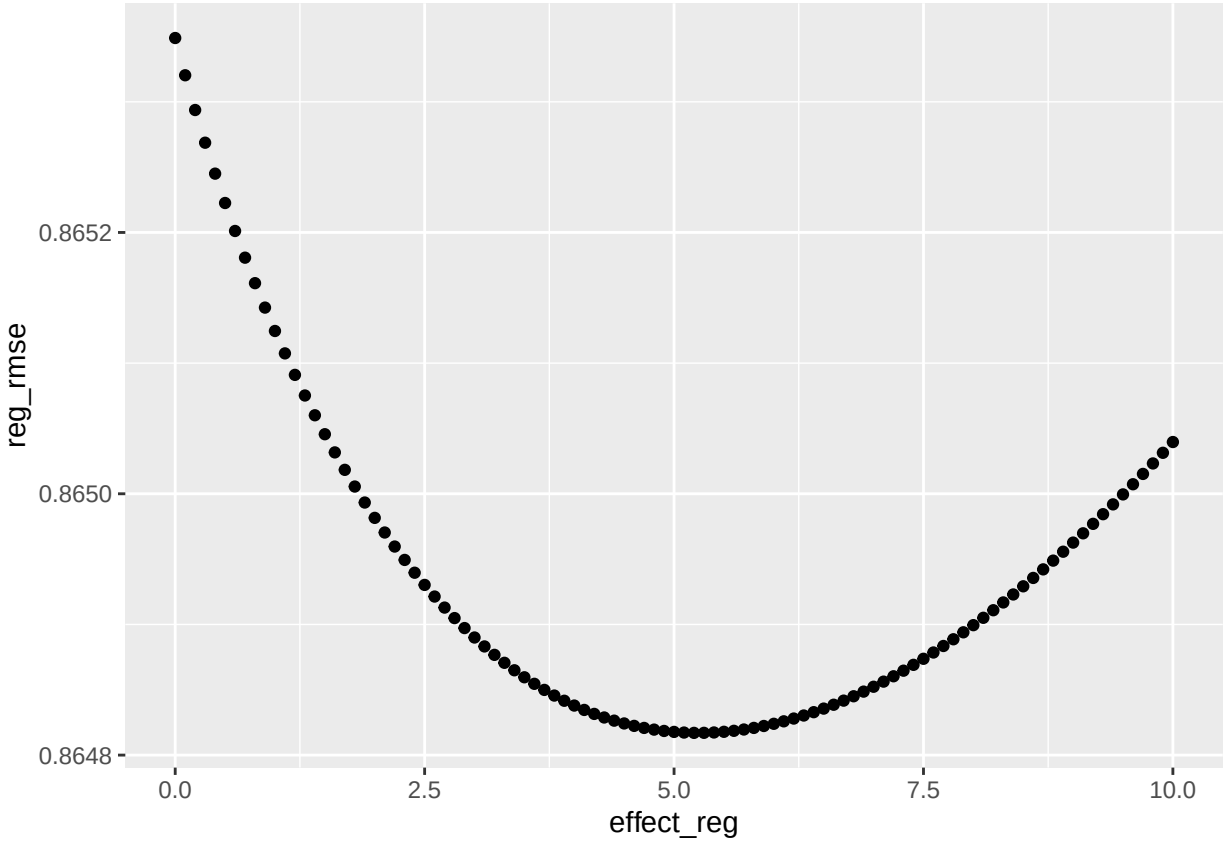
Model 4: Regularized Model

In the Model 4, I tried to remove the deviation effect due to outliers in the dataset. There could be extremely high or low predictions derived by outliers values. By adding a tuning parameter, named "effect_reg", we assessed the lowest value of RMSE by the effect_reg to regularise the model 3.

```
## Model 4: Regularized Model
# Let the effect_reg be the tuning parameter
effect_reg <- seq(0,10,0.1)

reg_rmse <- sapply(effect_reg, function(effect_reg){
  Mu_1 <- mean(edx$rating)
  effect_fm <- edx %>%group_by(movieId) %>%
    summarize(effect_fm = sum(rating - Mu_1) / (n() + effect_reg))
  effect_u <- edx %>%
    left_join(effect_fm, by = "movieId") %>%
    group_by(userId) %>%
    summarize(effect_u = sum(rating - effect_fm - Mu_1) / (n() + effect_reg))
  predictions <- validation %>%
    left_join(effect_fm, by = "movieId") %>%
    left_join(effect_u, by = "userId") %>%
    mutate(prediction = Mu_1 + effect_fm + effect_u) %>% .$prediction
  RMSE(validation$rating, predictions) })

# Plot the effect_reg against the regularised RMSEs to visualize what is the best effect_reg
tibble(effect_reg = effect_reg, reg_rmse = reg_rmse) %>%
  ggplot(aes(effect_reg, reg_rmse)) + geom_point()
```




```
# Find the effect_reg with minimal reg_rmse
effect_reg <- effect_reg[which.min(reg_rmse)]
effect_reg
```

```
## [1] 5.2
```

```
# Find the lowest rmse in the model based on the minimal effect_reg
min(reg_rmse)
```

```
## [1] 0.864817
```

3. Result
   The Model 4 Regularized Model achieved the lowest RMSE which can provide a better prediction performance.
   The model was built on the ground of mean rating, famous movie effect, user effect, and it is finally refined by the regularised effect which prevented the effect of outlier data(too high rating or too low rating).

```
results <- data_frame(Model=c("Model 1:Navie Model",
                              "Model 2:Famous Movie effect Model",
                              "Model 3:Famous movie and User effect Model",
                              "Model 4:Regularized Model"),
                      RMSE=c(Mu_1_rmse,Mu_2_rmse,Mu_3_rmse,min(reg_rmse)))
```

```
## Warning: `data_frame()` is deprecated as of tibble 1.1.0.
## Please use `tibble()` instead.
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_warnings()` to see where this warning was generated.
```

```
results
```

```
## # A tibble: 4 x 2
##   Model                                      RMSE
##   <chr>                                     <dbl>
## 1 Model 1:Navie Model                       1.06
## 2 Model 2:Famous Movie effect Model         0.944
## 3 Model 3:Famous movie and User effect Model 0.865
## 4 Model 4:Regularized Model                 0.865
```

```
signif(results$RMSE, digits=6)
```

```
## [1] 1.061200 0.943909 0.865349 0.864817
```

4. Conclusion

Model 4: Regularized Model has the lowest RMSE= 0.864817, which is < 0.86490. The Model 4 Regularized Model is the machine learning algorithm built for prediction of user rating so as to enhance the movie recommendation function of streaming media. When we built the algorithm, step-wise approach was applied to investigate how the variable affects the prediction performance. The dataset can further explore the effect of other variables e.g. genre and year of release and different combinations of the variables in order to further improve the algorithm.