Updated MovieLens Project

Viwen Suresh Kumar

1/9/2022

Introduction

The MovieLens dataset is being generated using the following code shown below. The movie ratings can be predicted in validation set. The code is used to analyse MovieLens dataset using validation set. Companies like Netflix use recommendation systems which use the ratings based on the items that were given to the users by making specific recommendations. These companies use this system to help them to sell many products like movies to the customers based on their recommendation and permits them to give rating on their products which can be predicted for their specific items. Thus, recommendations systems are one of the most frequent models in machine learning algorithms. Companies like Netflix and Amazon becomes successful due to their strong recommendation system.

Aim

The aim of this project is to investigate whether machine learning algorithm which is recommendation system that can predict the user ratings with range of 0.5 to 5 using the inputs of provided MovieLens dataset (from eDx) to predict movie ratings in a given validation set. Root Mean Square Root also known as RMSE is the value which is used to evaluate the machine learning algorithm performance. RMSE is mostly used in measuring the differences between predicted value by the model and observed value. RMSE defined as a measure of the error of a model in predicting quantative data. There were 4 models that were being developed to be compared using the resulting RMSE in order to improve the quality. The evaluation model for this algorithm is expected RMSE to be lower than 0.8775 which being shown below:

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

Methods & Analysis

Dataset

The validation sets must be generated before the code can be used to analyse the dataset of MovieLens. The code for the validation set for MovieLens dataset is shown below.

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

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

```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")

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")

# if using R 3.6 or earlier:
movies <- as.data.frame(movies) %>% mutate(movieId = as.numeric(levels(movieId))[movieId],
                                           title = as.character(title),
                                           genres = as.character(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)`
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)

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

Data Analysis

Dataset Dimensions

```
##   userId movieId rating timestamp                         title
## 1      1     122      5 838985046              Boomerang (1992)
## 2      1     185      5 838983525               Net, The (1995)
## 4      1     292      5 838983421               Outbreak (1995)
## 5      1     316      5 838983392               Stargate (1994)
## 6      1     329      5 838983392 Star Trek: Generations (1994)
## 7      1     355      5 838984474       Flintstones, The (1994)
##                         genres
## 1                Comedy|Romance
## 2          Action|Crime|Thriller
## 4  Action|Drama|Sci-Fi|Thriller
## 5       Action|Adventure|Sci-Fi
## 6 Action|Adventure|Drama|Sci-Fi
## 7       Children|Comedy|Fantasy
```

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

Unique Users

Following the questions from MovieLens quiz, there are 10677 unique movies and 69878 unique users which can be found using code shown below

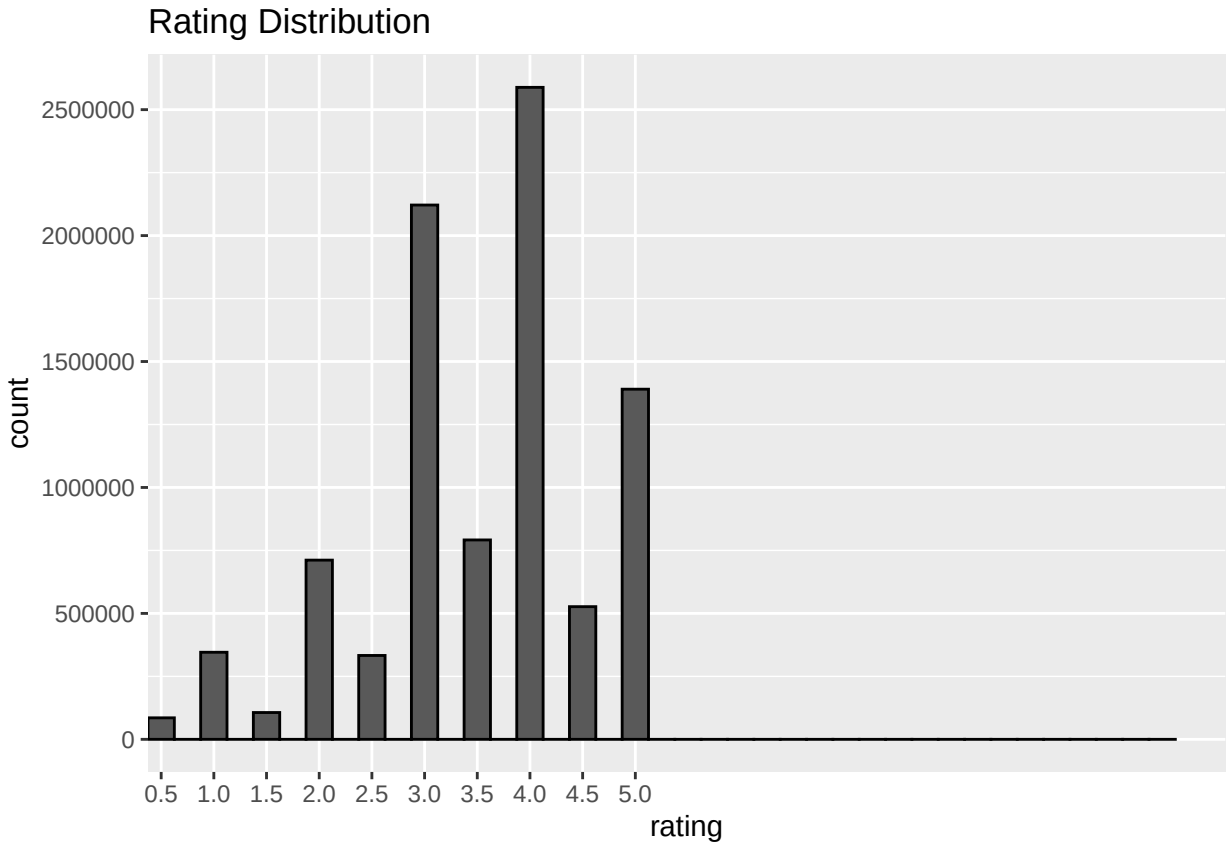
```
##   n_users n_movies
## 1   69878    10677
```

Ratings

The information of movielens rating being analysed using the code shown below

Rating Distribution

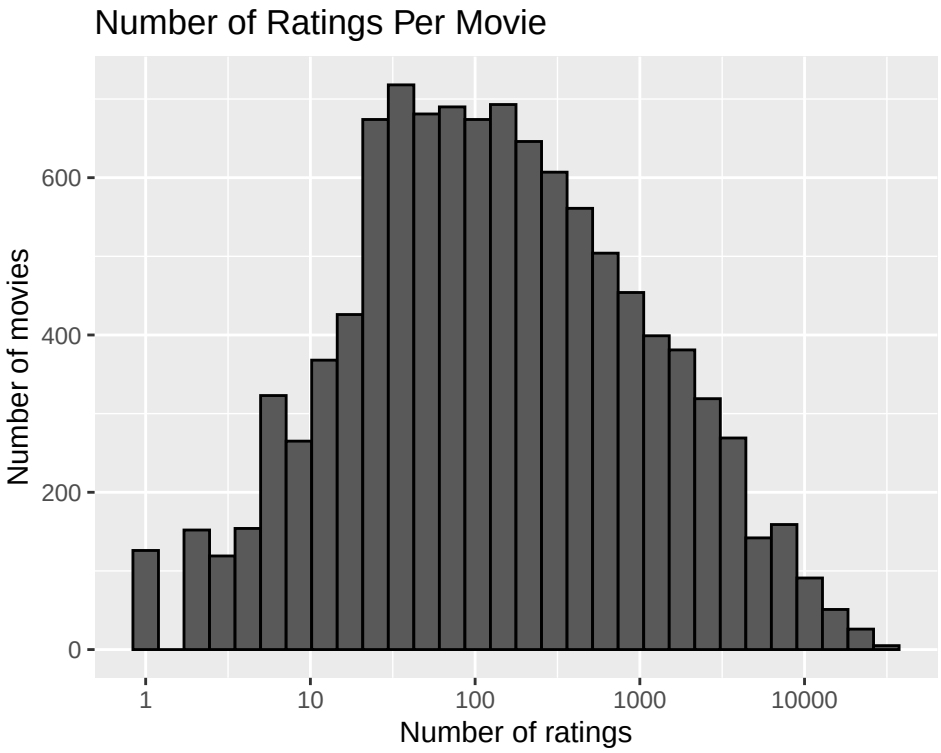
Based on the analysis of the movie ratings, it was shown that Pulp Fiction has the greatest number of ratings which is 31362 among 10667 movies in the dataset. Additionally, using the code below can determine the arranged order of the movie ratings which proven that rating 4 which is arranged as the highest



Based on the plot above, it is shown that we can observe that some movies have been much rated than the other movies while some of them have few ratings or at least one. Thus, it is shown that regularization is a technique used for tuning the function by adding an additional penalty term in the error function which can be applied in the model.

Number of Ratings

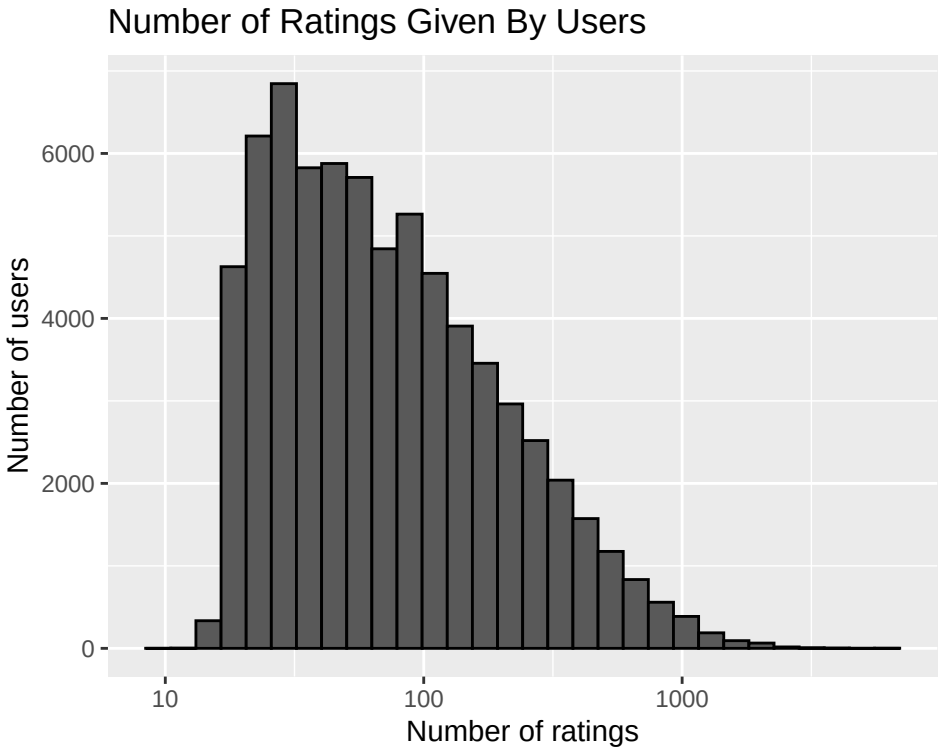
```
edx %>%
count(movieId) %>%
ggplot(aes(n)) +
geom_histogram(bins = 30, color = "black") +
scale_x_log10() +
xlab("Number of ratings") +
  ylab("Number of movies") +
ggtitle("Number of Ratings Per Movie")
```



Following the graph shown above, it is shown that number of movies is the highest when the number of ratings close to 50. However, the number of movies had been dropping when the number of ratings is higher than 500. Finally, the number of the movies is close to 0 after the it hits after 10000 ratings.

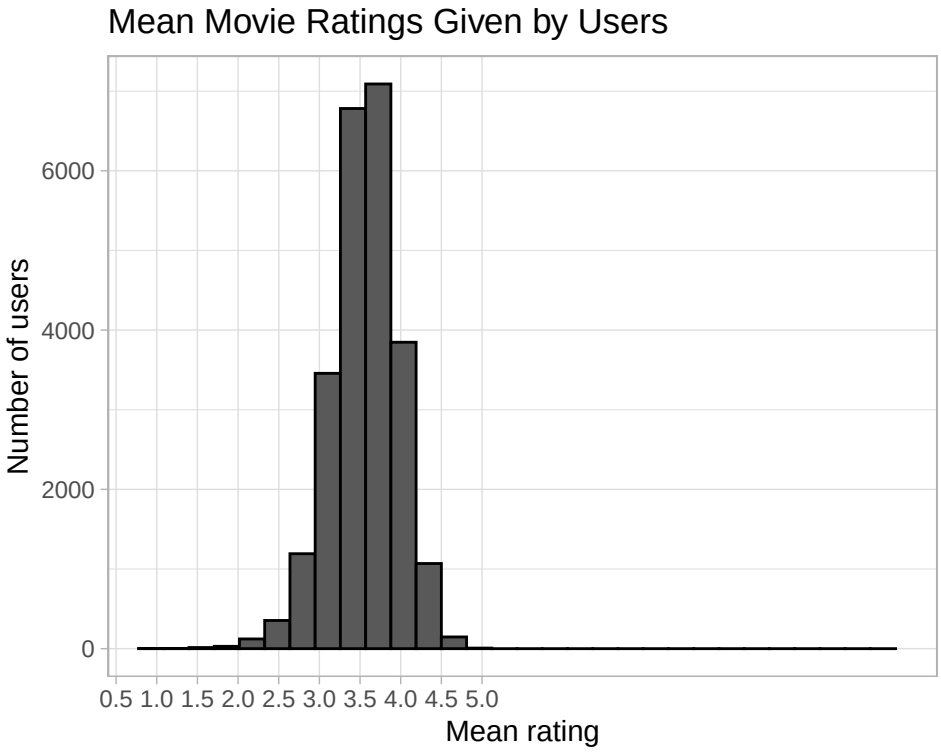
Based on the graph shown below, it is shown that the majority of users have rated between 30 and 100 movies.Therefore, a user penalty term must be included later in this model

```
edx %>%
count(userId) %>%
ggplot(aes(n)) +
geom_histogram(bins = 30, color = "black") +
scale_x_log10() +
xlab("Number of ratings") +
ylab("Number of users") +
ggtitle("Number of Ratings Given By Users")
```



Futhermore, there are some users that tend to give lower star ratings while there are users do give higher star ratings. Thus, this shows that users differ vastly in how critical they are with their ratings. The visualisation is shown below based on the code analysis

```
edx %>%
  group_by(userId) %>%
  filter(n() >= 100) %>%
  summarize(b_u = mean(rating)) %>%
  ggplot(aes(b_u)) +
  geom_histogram(bins = 30, color = "black") +
  xlab("Mean rating") +
  ylab("Number of users") +
  ggtitle("Mean Movie Ratings Given by Users") +
  scale_x_discrete(limits = c(seq(0.5,5,0.5))) +
  theme_light()
```


Mean Movie Ratings Given by Users
Number of users
6000
4000
2000
0
0.5 1.0 1.5 2.0 2.5 3.0 3.5 4.0 4.5 5.0
Mean rating

Modelling Approach

Following analysis from the MovieLens dataset and the graphs from the above, the formula of loss-function which computed the RMSE which defined as follows:

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

With N represent as number of user/movie combinations and the sum that occurs over all these combinations. Thus, RMSE also represent as measure of model accuracy. RMSE can be intepreted same as the standard deviation which is the typical error of movie rating predictions. Furthermore, the typical error will be larger than one star if the result is larger than 1 which not a good result.

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

A. Average movie rating model

Following the formula shown below, this predicted the same rating for all movies, so the dataset's mean rating can be computed. The expected rating of underlying data set situated between 3 and 4. Furthermore, the simplest possible recommender system can be build to predict the same rating for all movies despite of users who give it. The formula which is displayed below is based on approaching the assumes the same rating for all movie with all differences explained by random variation.

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

with $\epsilon_{u,i}$ being independent error sample from the same distribution which centred at 0 and μ represents as true rating for all movies. Thus, it is estimated that the minimize of RMSE is the least square estimate of $Y_{u,i}$ which is the average of all ratings.

```
mu <- mean(edx$rating)
mu
```

```
## [1] 3.512465
```

Additionally, the 1st naive RMSE can be obtained if all unknown ratings with μ being predicted.

```
naive_rmse <- RMSE(validation$rating, mu)
naive_rmse
```

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

Next, the results table is represented with the 1st RMSE:

```
rmse_results <- data_frame(method = "Average movie rating model", RMSE = naive_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.
```

```
rmse_results %>% knitr::kable()
```

method	RMSE
Average movie rating model	1.061202

Thus, this produce the baseline of RMSE to compare with the next modelling approaches.

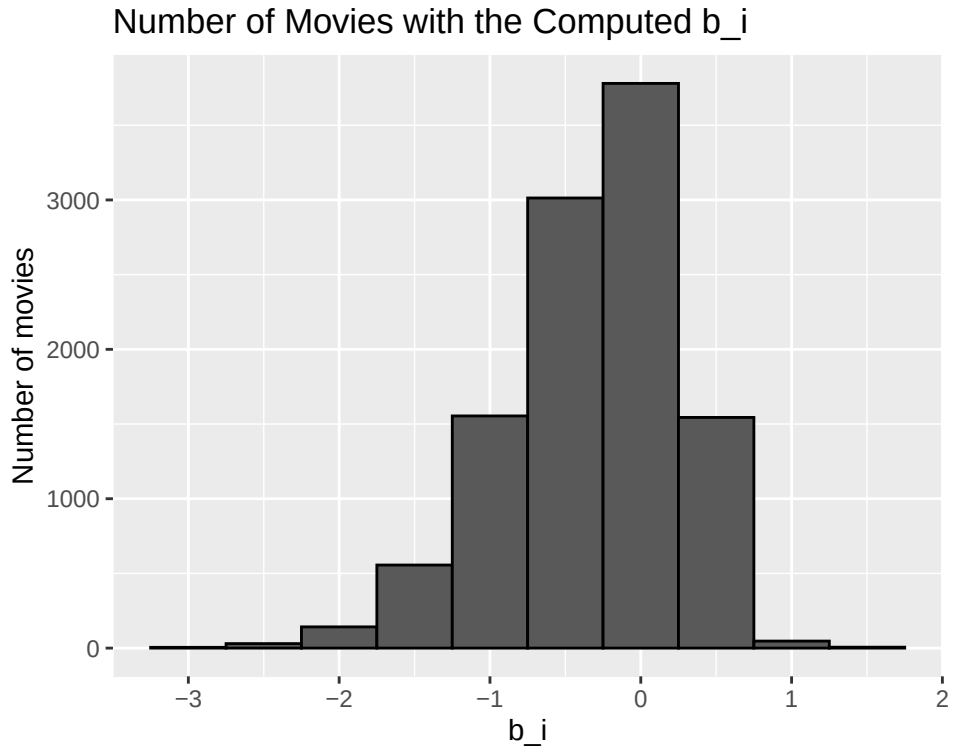
B.Movie effect model

The model from the above can be improved since it is known that some movies are generally rated higher than other movies. Higher ratings means higher popularity movies among the users while lower ratings means lower popularity of the movies. The estimated deviation can be computed for each movies' mean rating from the total mean of all movies μ. Thus, the resulting variable is called "b" (known as bias) for each movie "i" where b_i represent as average ranking for movie i:

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

This implied that more movies have negative effects since the histogram seems asymmetrical. Therefore, this is known as penalty term movie effect

```
movie_avgs <- edx %>%
  group_by(movieId) %>%
  summarize(b_i = mean(rating - mu))
movie_avgs %>% qplot(b_i, geom ="histogram", bins = 10, data = ., color = I("black"),
ylab = "Number of movies", main = "Number of Movies with the Computed b_i")
```



Using this model, this can improve our predictions of movie ratings.

```
predicted_ratings <- mu +  validation %>%
  left_join(movie_avgs, by='movieId') %>%
  pull(b_i)
model_1_rmse <- RMSE(predicted_ratings, validation$rating)
rmse_results <- bind_rows(rmse_results,
                          data_frame(method="Movie effect model",
                                     RMSE = model_1_rmse ))
rmse_results %>% knitr::kable()
```

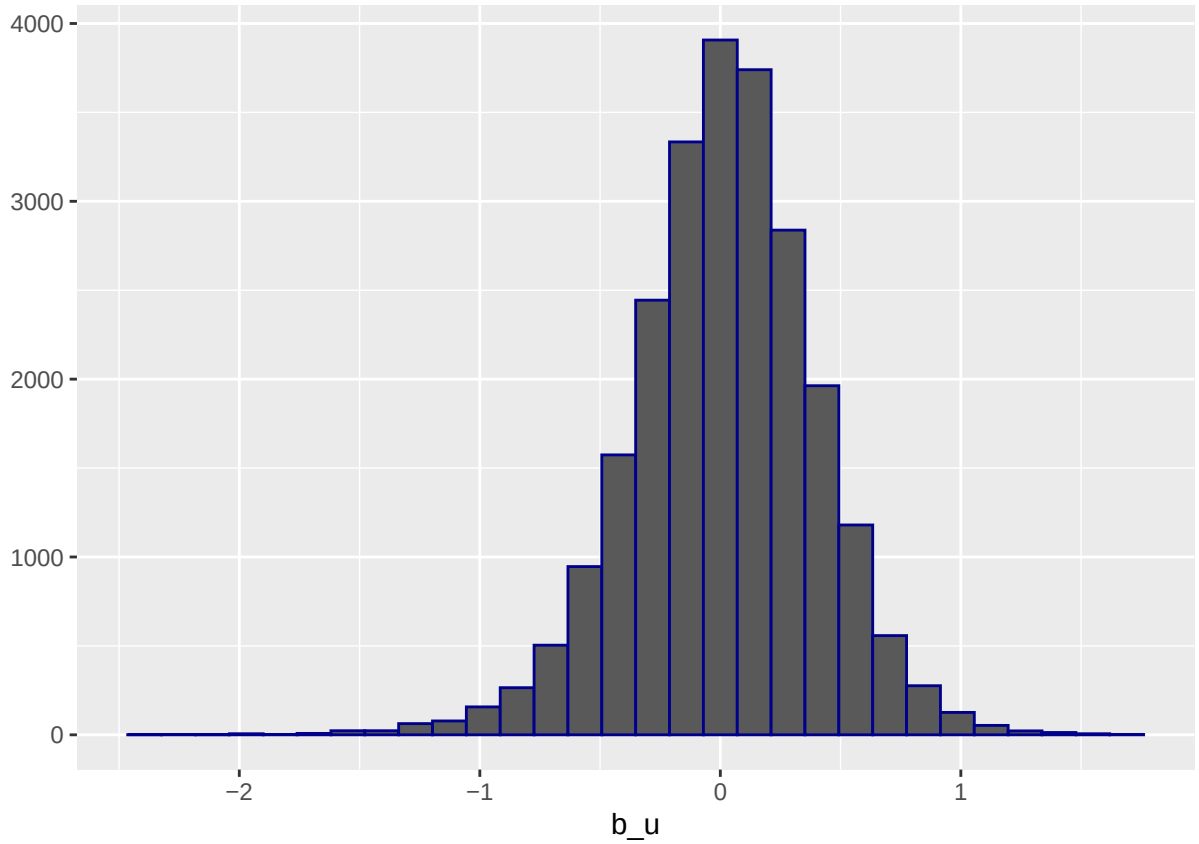
method	RMSE
Average movie rating model	1.0612018
Movie effect model	0.9439087

Thus, the movie rating can be predicted based on the fact that movies are rated differently by combining the computed b_i to μ. It is predicted that movie rating is lower than μ by b_i if an average of individual movie is rated worse than the average rating of all movies μ with difference of the individual movie average from the total average. Eventhough there is an improvement in this model, but this model cannot consider the individual user rating effect.

C. Movie & user effect model

The average rating for user μ can computed for those that have rated over 100 movies. Thus, the users can affect the ratings positively or negatively

```
user_avgs<- edx %>%
  left_join(movie_avgs, by='movieId') %>%
  group_by(userId) %>%
  filter(n() >= 100) %>%
  summarize(b_u = mean(rating - mu - b_i))
user_avgs%>% qplot(b_u, geom ="histogram", bins = 30, data = ., color = I("dark blue"))
```



Based on the graph shown above, there was substantial variability where there are some users do not like the every movies while others love every movie. Thus, it was hinted that there was the further improvement

in this model which was shown below:

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

where b_u represent as user-specific effect. If irritable user (negative b_u) rates a great movie (positive b_i), this effects will confict each other and it can be predicted correctly that the user gave this great movie a 3 rather than 5. we can compute μ and b_i and estimate b_u which can be used to compute an approximation, as the average of

$$Y_{u,i} - \mu - b_i$$

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

Now the predictors can be constructed

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

model_2_rmse <- RMSE(predicted_ratings, validation$rating)
rmse_results <- bind_rows(rmse_results,
                          data_frame(method="Movie and user effect model",
                                     RMSE = model_2_rmse))
rmse_results %>% knitr::kable()
```

method	RMSE
Average movie rating model	1.0612018
Movie effect model	0.9439087
Movie and user effect model	0.8653488

The RMSE got reduced further by the rating predictions. However, there were some mistakes in the first model where the suppose 'best' and 'worst' movies were rated by few users. Since there were more uncertainty, this shows that these movies were mostly unclear. Therefore,larger estimates of b_i are more likely to produce large errors which increase the RMSE.

The standard error and constructed confidence intervals to account for different levels of uncertainty. However, one number and one prediction to make the predictions excluding the interval. Hence, the concept of regularization is introduced which permits to penalize large estimates that comes from small sample sizes. The general idea of minimizing the sum of squares equation is to add a penalty for large b_i value. This proves that the larger the value of b_i, the harder is to minimize sum of squares equation. Regulariztion is a method of adding information in order to reduce the effect of overfitting.

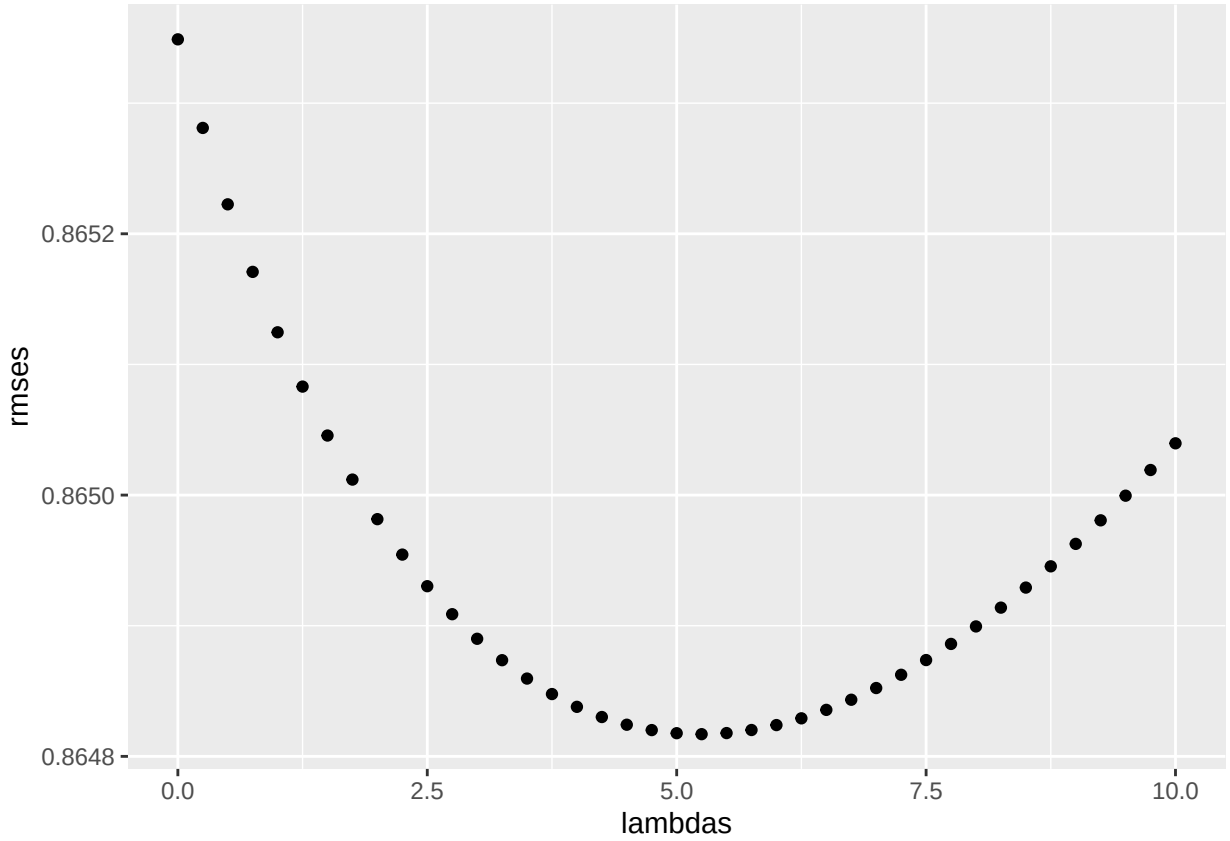
D. Regularized movie & user effect model

Movies with few ratings and in some users that only rated a very small number of movies shich caused the estimates of b_i and b_u. Thus, the rating prediction can be strongly influenced by this. Regularization is also used to permit to penalize these aspects. The value of lambda, λ (as turning parameter) must be determined in order to minimize RMSE which makes the b_i and b_u smaller in case of small number of ratings.

```
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) %>%
    pull(pred)

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

RMSE vs lambdas is plotted to determine the optimal lambda, λ_o

```
qplot(lambdas, rmses)
```



The optimal lambda, λ_o for full model is 5.25 based on the data analysis of the code shown below.

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

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

The new result table is shown below along with the code:

```
rmse_results <- bind_rows(rmse_results,
                          data_frame(method="Regularized movie and user effect model",
                                     RMSE = min(rmses)))
rmse_results %>% knitr::kable()
```

method	RMSE
Average movie rating model	1.0612018
Movie effect model	0.9439087
Movie and user effect model	0.8653488
Regularized movie and user effect model	0.8648170

method	RMSE

Results

The RMSE values of all the represented models are shown in the result table below:

method	RMSE
Average movie rating model	1.0612018
Movie effect model	0.9439087
Movie and user effect model	0.8653488
Regularized movie and user effect model	0.8648170

Theresore, the lowest value of RMSE is 0.8648170

Discussion

Finally, the final model for this project is confirmed which is shown below:

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

This proves that this model work smoothly if the average user does not rate popular movie with a large positive b_i by disliking the particular movie.

Conclusion

In conclusion, it was finally stated that machine learning algorithm was being built to predict movie ratings using MovieLens dataset. The lower RMSE value characterized the regularized model which includes the effect of the user and optimal model was used for the present project. The final value of lowest RMSE is 0.8648170 with improvements in optimal model which was lower than the value from initial evaluation model that produce 0.8775. We can deduce that RMSE can be improved by adding other effect (e.g: genre/year/age). Thus, this proves that the movie rating prediction made by the user is certain.