HarvardX: PH125.9x Data Science
MovieLens Movie Rating Prediction

Battula Damodhar

June 22, 2021

Contents

1 Overview

Recommendation System is an implementation of Machine Learning Algorithms and one of the most successful application of Machine Learning Technologies in business. In this project we will use and combine several machine learning strategies to construct movie recommendation system using "MovieLens" dataset. The present report start with a general idea of the project and by representing its objective.Then the given dataset will be prepared and setup. An exploratory analysis is carried out in order to develop a machine learning algorithm that could predict movie ratings until a final model. Results will be explained. Finally the report ends with some concluding remarks.

1.1 Introduction

Recommendation System is an important class of Machine Learning Techniques that offers "relevant" suggestions to users. It provides suggestions to users through a filtering process based on user browsing history and preferences.

In this project we will be creating a movie recommendation system using the MovieLens dataset. The actual dataset we can find at: https://grouplens.org/datasets/movielens/latest/ is much larger with millions of ratings. But in this project we will use '10M' version of the MovieLens dataset instead of actual one. The data will be split into 90% as 'edx' set and 10% as 'validation' set. We will split the edx data into separate training and test sets to design and test our algorithm. We will use 'edx' set to develop algorithm and 'validation' set for final prediction and to get the RMSE value. First we build a baseline prediction models on movie, user effect then will apply regularization using these effects. Also will add year effect in regularization to see improvement.

We will use Root Mean Square Error (RMSE) in this project to evaluate algorithm performance. Root mean square (RMSE) is the standard deviation of the residuals (estimated errors). RMSE is mostly used to measure the differences between values predicted by a model and the values observed. We will stop our prediction if we get expected RMSE else will go with matrix factorization technique for further prediction.

1.2 Aim and Objective

The aim of this project is to develop and train a machine learning algorithm using the inputs of a provided edx dataset to predict the movie ratings on provided validation set. The objective of this project is to develop a model to get RMSE expected to be lower than 0.86490.

2 Dataset downloading and preperation

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

#Installing required packages:
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")
```

2.1 Used Libraries

```
library(tidyverse)
library(caret)
library(data.table)
library(ggplot2)
```

2.2 Used Dataset

The MovieLens dataset is automatically downloaded

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

2.3 Data Loading

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

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: Using 4.1.0
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")
```

2.4 Data Pre-processing

```
#if using R 3.6 or earlier
#set.seed(1)

#if using R 4.0 or later:- using R 4.1.0
set.seed(1,sample.kind="Rounding")

# Split Raw data into Train and Test sets:
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)
```

Now we have two dataset, 'edx' to develope algorithm and 'validation' set for final prediction and to evaluate the RMSE value.

Extracting year as a column from title in the edx & validation dataset.

```
edx <- edx %>% mutate(year = as.numeric(str_sub(title,-5,-2)))
validation <- validation %>% mutate(year = as.numeric(str_sub(title,-5,-2)))
```

3 Method and Analysis

3.1 Data Analysis and Exploration

To get familiar with data, let's have a look on general overview of dataset.

```
dim(edx)
```

```
## [1] 9000055       7
```

```
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
## 1:                Comedy|Romance 1992
## 2:         Action|Crime|Thriller 1995
## 3:  Action|Drama|Sci-Fi|Thriller 1995
## 4:       Action|Adventure|Sci-Fi 1994
## 5: Action|Adventure|Drama|Sci-Fi 1994
## 6:       Children|Comedy|Fantasy 1994
```

```
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               year
##  Length:9000055     Length:9000055     Min.   :1915
##  Class :character   Class :character   1st Qu.:1987
##  Mode  :character   Mode  :character   Median :1994
##                                        Mean   :1990
##                                        3rd Qu.:1998
##                                        Max.   :2008
```

We can see, edx set contains 9000055 records and 6 variables 'userId', 'movieId', 'rating', 'timestamp', 'title', 'genres' and 'year'(extracted from column 'title'). Each row represents a single rating of a user for each movie. The summary of the set confirms that there are no missing values.

Now summarize the number of unique movies and users in the edx dataset.

```
edx %>%
  summarize(n_users = n_distinct(userId),
            n_movies = n_distinct(movieId))
```

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

There are 69878 distinct users and 10677 distinct movies in the edx set. If every user rated on every movie then there would be 746087406 (i.e. 69878 X 10677) possible ratings. But we have only 9000055 ratings, which is nearly 1/83 times of the possible ratings.

Top 10 Movies ranked in order of the number of Ratings :

```
edx %>% group_by(movieId, title) %>%
  summarize(count = n()) %>%
  arrange(desc(count))
```

```
## # A tibble: 10,677 x 3
## # Groups:   movieId [10,677]
##    movieId title                                                        count
##      <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
```

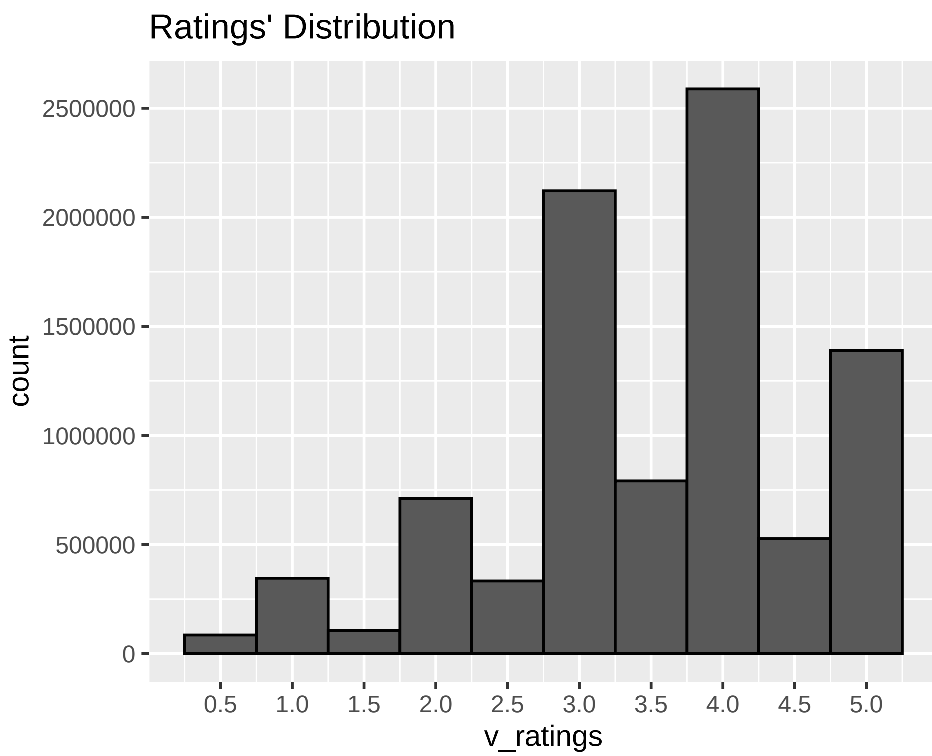
Ratings' Distribution:

The ratings' of movies are in range of 0 to 5 and users have a preference to rate movies rather higher than lower. Lets have a look into unique movie ratings' given by user.

```
v_ratings <- as.vector(edx$rating)
unique(v_ratings)
```

```
##  [1] 5.0 3.0 2.0 4.0 4.5 3.5 1.0 1.5 2.5 0.5
```

```
edx %>% ggplot(aes(v_ratings)) +
  geom_histogram(bins = 10, color = "black") +
  scale_x_continuous(breaks = c(seq(0.5,5,0.5))) +
  scale_y_continuous(breaks = c(seq(0,3000000,500000))) +
  ggtitle("Ratings' Distribution")
```

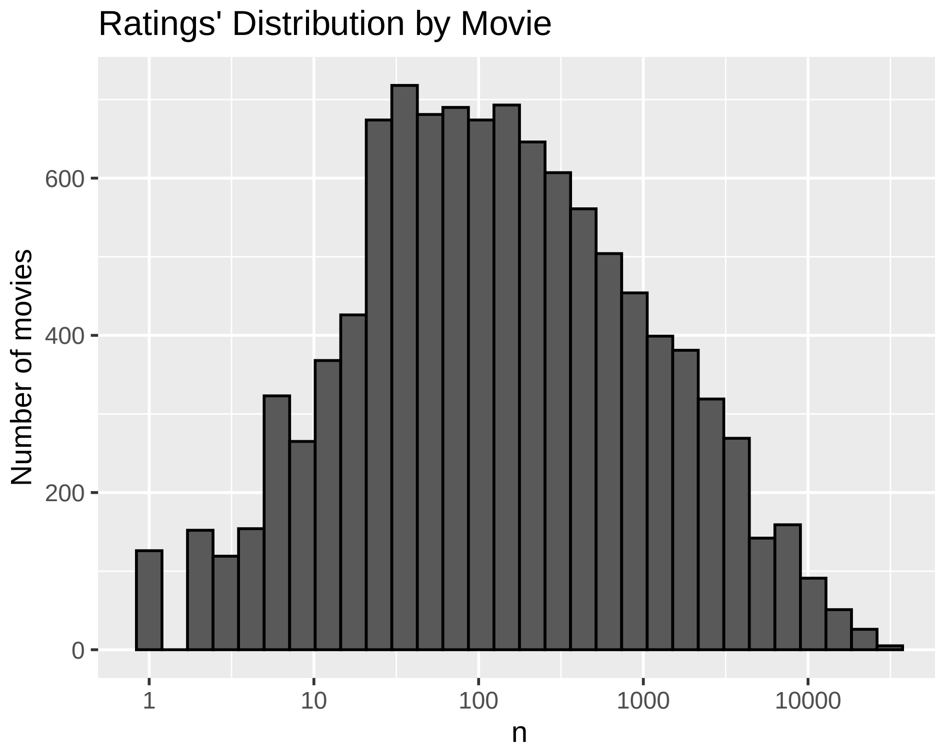


We can see 4 is the most common rating followed by 3, 5, 3.5, 2 and so on. 0.5 is the least common rating.

Ratings' Distribution by Movie:

Number of times majority of movies reviewed by user.

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



The histograms represent that the majority of movies have been reviewed between 30 and 1000 times. There are 125 movies that have been viewed only once.

Top 20 Movies which rated only once by User:

```
edx %>% group_by(movieId) %>%
  summarize(count = n()) %>%
  filter(count == 1) %>%
  left_join(edx, by = "movieId") %>%
  group_by(title) %>%
  summarize(rating = rating, n_rating = count) %>%
  slice(1:20) %>%
  knitr::kable()
```

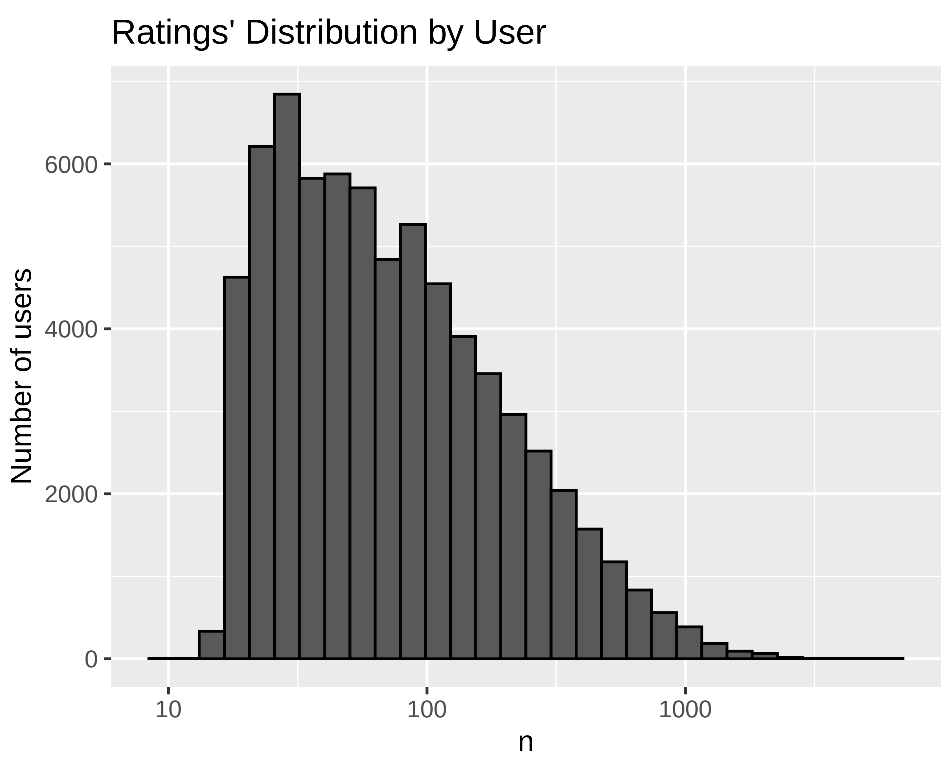
title	rating	n_rating
1, 2, 3, Sun (Un, deuz, trois, soleil) (1993)	2.0	1
100 Feet (2008)	2.0	1
4 (2005)	2.5	1
Accused (Anklaget) (2005)	0.5	1
Ace of Hearts (2008)	2.0	1
Ace of Hearts, The (1921)	3.5	1
Adios, Sabata (Indio Black, sai che ti dico: Sei un gran figlio di…) (1971)	1.5	1
Africa addio (1966)	3.0	1
Aleksandra (2007)	3.0	1
Bad Blood (Mauvais sang) (1986)	4.5	1
Battle of Russia, The (Why We Fight, 5) (1943)	3.5	1
Bellissima (1951)	4.0	1
Big Fella (1937)	3.0	1

title	rating	n_rating
Black Tights (1-2-3-4 ou Les Collants noirs) (1960)	3.0	1
Blind Shaft (Mang jing) (2003)	2.5	1
Blue Light, The (Das Blaue Licht) (1932)	5.0	1
Borderline (1950)	3.0	1
Brothers of the Head (2005)	2.5	1
Chapayev (1934)	1.5	1
Cold Sweat (De la part des copains) (1970)	2.5	1

Ratings' Distribution by User:

Number of movies rated by majority of users.

```
edx %>% count(userId) %>%
  ggplot(aes(n)) +
  geom_histogram(bins = 30, color = "black") +
  scale_x_log10() +
  ylab("Number of users") +
  ggtitle("Ratings' Distribution by User")
```



The histogram represents that the majority of users have been rated below 100 movies, but also above 30 movies.

3.2 Modelling Approach

Function to calculate RMSE value:

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

3.2.1 Average Rating Model

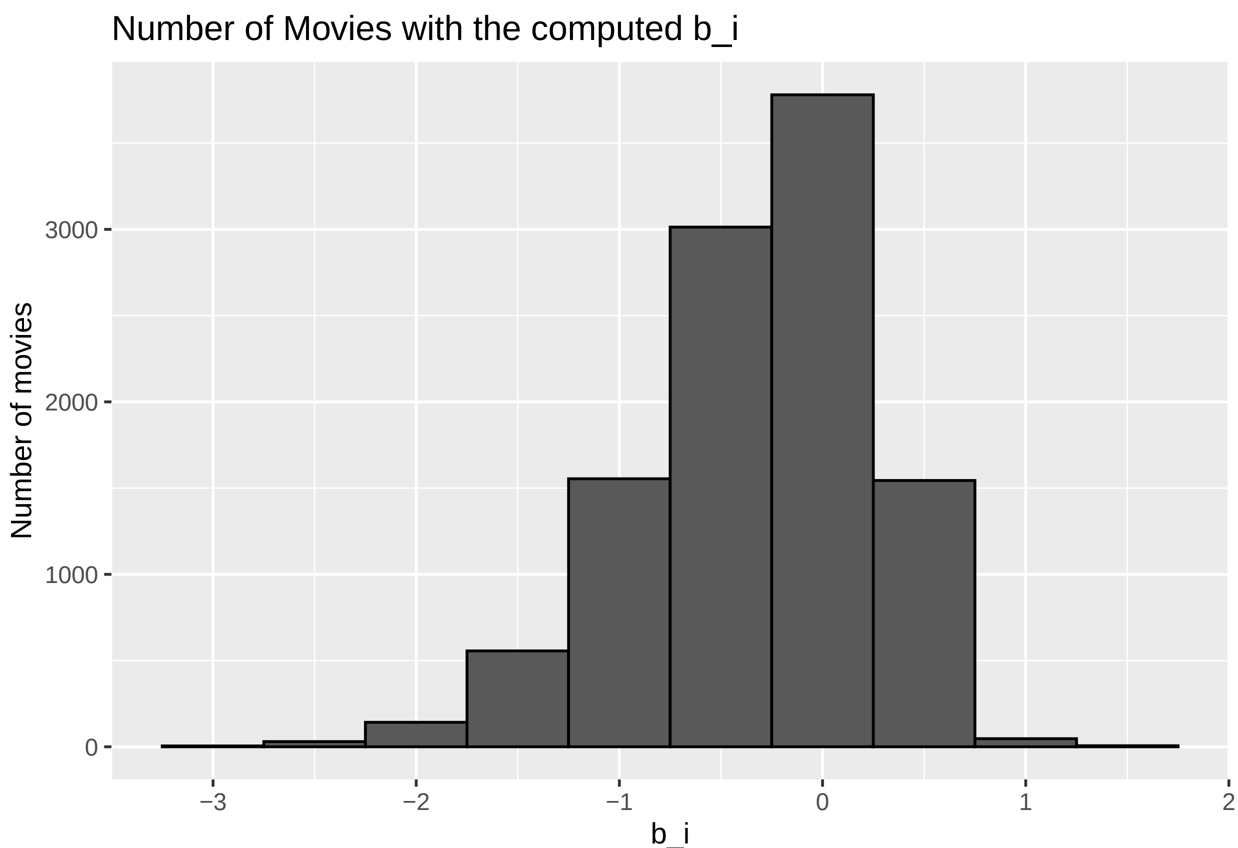
```
# Mean rating estimation
mu <- mean(edx$rating)

# Using Mean only
model_1_rmse <- RMSE(validation$rating, mu)

# Save RMSE result in data frame
rmse_result <- data.frame(Model = "Average Rating Model", RMSE = model_1_rmse)

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

Model	RMSE
Average Rating Model	1.061202

3.2.2 Movie Effect Model

Adding movie effect to achieve improvement in RMSE.

```
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")
```




```r
predicted_ratings_2 <- mu + validation %>%
  left_join(movie_avgs, by='movieId') %>%
  pull(b_i)

model_2_rmse <- RMSE(validation$rating, predicted_ratings_2)

# Save RMSE result in data frame
rmse_result <- bind_rows(rmse_result,
                         tibble(Model="Movie Effect Model",
                                RMSE = model_2_rmse))

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

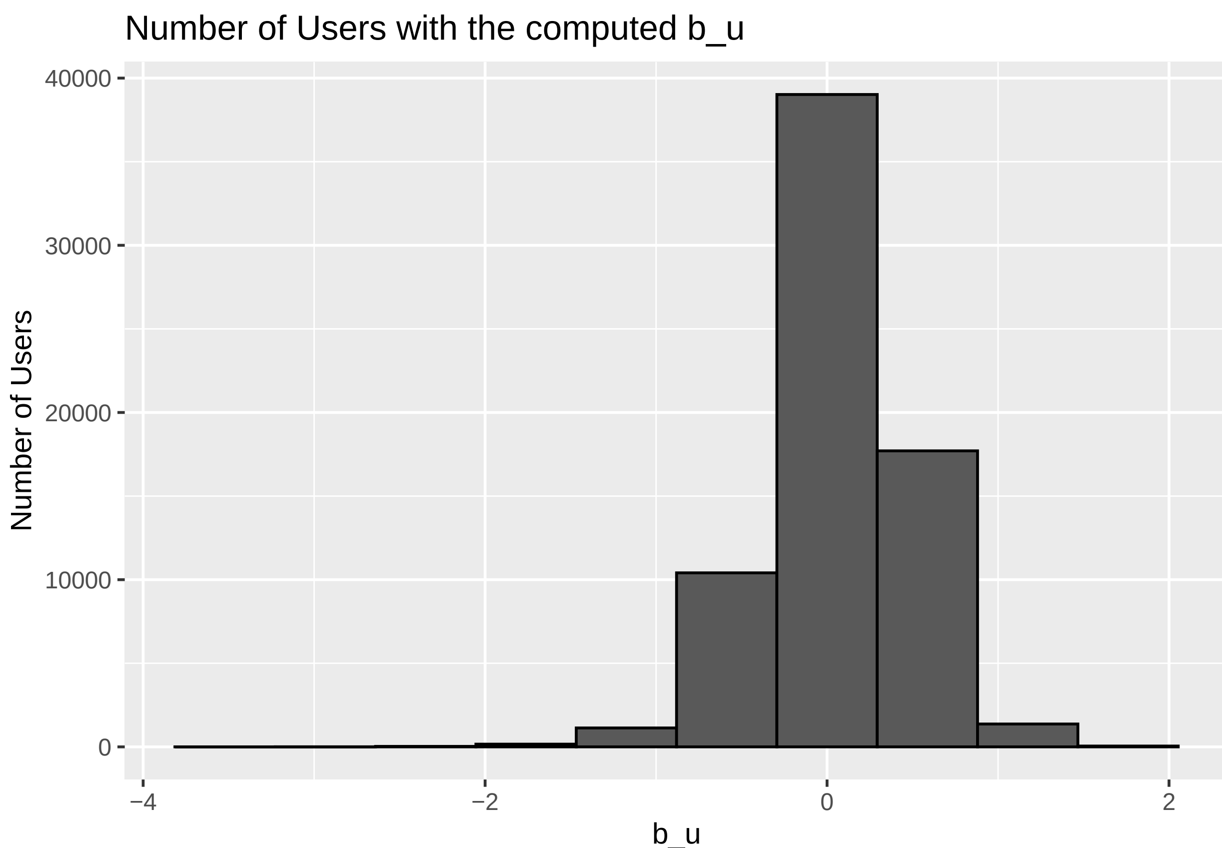
Model	RMSE
Average Rating Model	1.0612018
Movie Effect Model	0.9439087

We can see RMSE is improved by adding the Movie Effect.

3.2.3 User Effect Model

Adding user effect in above model to achieve further improvement in RMSE.

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

user_avgs %>% qplot(b_u, geom ="histogram", bins = 10, data = .,
                    color = I("black"), ylab = "Number of Users",
                    main = "Number of Users with the computed b_u")
```




```
predicted_ratings_3 <- validation %>%
  left_join(movie_avgs, by='movieId') %>%
  left_join(user_avgs, by='userId') %>%
  mutate(pred = mu + b_i + b_u) %>%
  pull(pred)

model_3_rmse <- RMSE(validation$rating, predicted_ratings_3)

# Save RMSE result in data frame
rmse_result <- bind_rows(rmse_result,
                         tibble(Model = "Movie and User Effects Model",
                                RMSE = model_3_rmse))

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

Model	RMSE
Average Rating Model	1.0612018
Movie Effect Model	0.9439087
Movie and User Effects Model	0.8653488

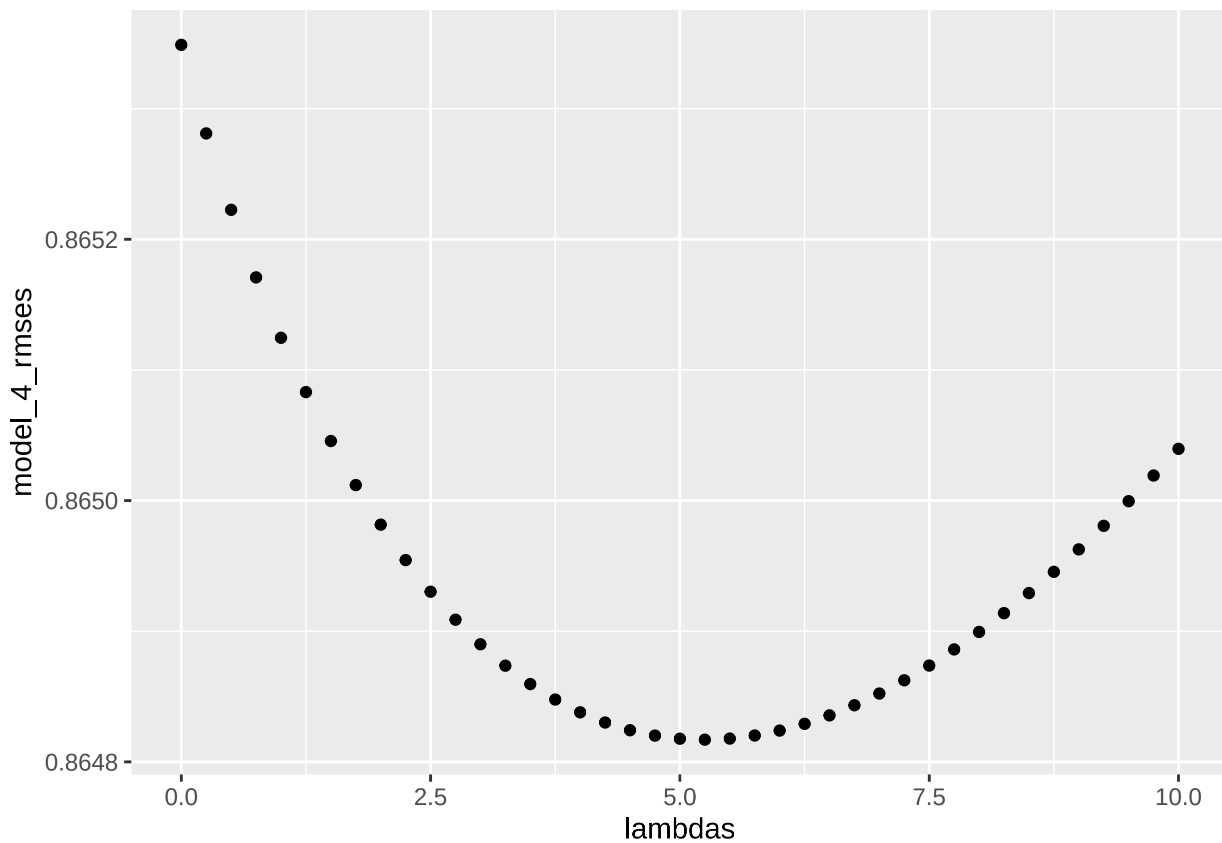
Achieved further improvement by adding the user effect. Now apply regularization on above model for further improvement.

3.2.4 Regularized Movie and User Effect Model

The Regularized model implements the concept of regularization which helps to account for the effect of lower ratings' numbers. We use Regularization method to reduce the effect of Overfitting.

Here we use cross validation to chose it and lambda as a tuning parameter. For each lambda, will find b_i and b_u followed by rating prediction and testing.

```r
lambdas = seq(0, 10, 0.25)

# Note: Below function could take a couple of minutes
model_4_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_4 <- validation %>%
    left_join(b_i, by = "movieId") %>%
    left_join(b_u, by = "userId") %>%
    mutate(pred = mu + b_i + b_u) %>%
    pull(pred)

  return(RMSE(validation$rating, predicted_ratings_4))
})

# Plot rmse vs lambda to select the optimal lambda
qplot(lambdas, model_4_rmses)
```




```
lambda <- lambdas[which.min(model_4_rmses)]
lambda
```

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

```
# Save RMSE result in data frame
rmse_result <- bind_rows(rmse_result,
                         tibble(Model = "Regularized Movie and User Effect Model",
                                RMSE = min(model_4_rmses)))

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

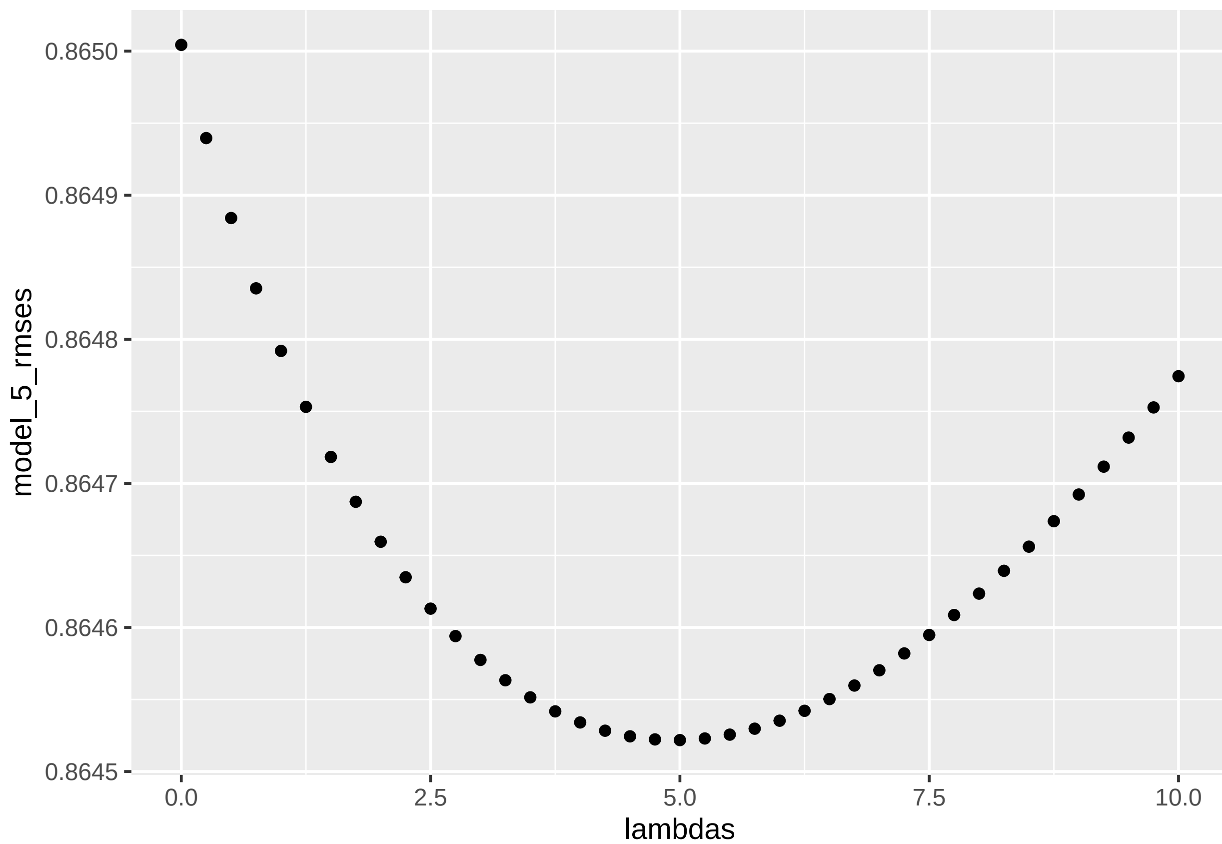
Model	RMSE
Average Rating Model	1.0612018
Movie Effect Model	0.9439087
Movie and User Effects Model	0.8653488
Regularized Movie and User Effect Model	0.8648170

The RMSE 0.8648170 is just fulfilled the criteria. Now add year effect in the above regularization model for further improvement on RMSE to get better results.

3.2.5 Regularized Movie, User and Year Effect Model

Add Year effect b__y in the above regularized model. For each lambda, will find b__i, b__u and b__y followed by rating prediction and testing like before.

```
lambdas = seq(0, 10, 0.25)

# Note: Below function could take a couple of minutes
model_5_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_y <- edx %>%
    left_join(b_i, by='movieId') %>%
    left_join(b_u, by='userId') %>%
    group_by(year) %>%
    summarize(b_y = sum(rating - mu - b_i - b_u)/(n()+l))

  predicted_ratings_5 <- validation %>%
    left_join(b_i, by='movieId') %>%
    left_join(b_u, by='userId') %>%
    left_join(b_y, by = 'year') %>%
    mutate(pred = mu + b_i + b_u + b_y) %>%
    pull(pred)

  return(RMSE(validation$rating, predicted_ratings_5))
})

# Plot rmse vs lambda to select the optimal lambda
qplot(lambdas, model_5_rmses)
```




```
lambda <- lambdas[which.min(model_5_rmses)]
lambda
```

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

```
# Save RMSE result in data frame
rmse_result <- bind_rows(rmse_result,
                         tibble(Model = "Regularized Movie, User and Year Effect Model",
                                RMSE = min(model_5_rmses)))

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

Model	RMSE
Average Rating Model	1.0612018
Movie Effect Model	0.9439087
Movie and User Effects Model	0.8653488
Regularized Movie and User Effect Model	0.8648170
Regularized Movie, User and Year Effect Model	0.8645218

4 Results

The RMSE values of all represented models are the following:

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

Model	RMSE
Average Rating Model	1.0612018
Movie Effect Model	0.9439087
Movie and User Effects Model	0.8653488
Regularized Movie and User Effect Model	0.8648170
Regularized Movie, User and Year Effect Model	0.8645218

We achieved the lowest value of RMSE that is 0.8645218.

5 Conclusion

The regularized model including the effect of movie, user and year results the lowest RMSE value of 0.8645218 and it is the optimal model for this project. This RMSE is lower than the initial evolution criteria (0.86490) given as the goal of this project.

Further improvement in this model could be achieved by adding other effects (e.g. genres, age). Other machine learning models (e.g. Matrix factorization) could also improve the results.

6 Appendix - Enviroment

Operating System: Microsoft Windows 10

R Version details:

```
version
```

```
##                _
## platform       x86_64-w64-mingw32
## arch           x86_64
## os             mingw32
## system         x86_64, mingw32
## status
## major          4
## minor          1.0
## year           2021
## month          05
## day            18
## svn rev        80317
## language       R
## version.string R version 4.1.0 (2021-05-18)
## nickname       Camp Pontanezen
```