HarvardX: PH125.9x Data Science

Mangalam Khare

06th January 2021

Introduction

This project is part of the HarvardX course Capstone project. The objective of this project is to develop Machine learning algorithm using the "10 M version of MovieLens dataset.Several machine learning algorithm has been used and results have been compared to get maximum possible accuracy. RMSE, Root Mean Square Error is used to evaluate the algorithm performance, It is the one of the most used measure of the diffrerences between predicted values and actual/observed values. Smaller the RMSE the better a model is able to fit the data

This Report has a problem statement section, data set preparation, Data pre-processing and exploratory analysis, Modelling and analysis of various models, results and conclusion

Problem Statement

The objective of this project is to use machine learning algorithms that predicts user ratings using the inputs present in the MovieLens dataset (trainset : edx) and validate them with tests et (Validation_set) predicts the movie rating by a user based on users past rating of movies. As mentioned in the Introduction section the aim is to get the smallest RMSE possible

The dataset used for this purpose can be found in the following links ? [MovieLens 10M dataset] https://grouplens.org/datasets/movielens/10m/ ? [MovieLens 10M dataset - zip file] http://files.grouplens.org/datasets/movielens/ml-10m.zip

Dataset Preparation

```r
################################################################
# Create edx set, validation set, and submission file
################################################################

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:
```

```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")

# 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))


ratings_tab <- as.data.frame(ratings) %>%
                    mutate(movieId =as.numeric(movieId),
                    userId = as.numeric(userId),
                    rating = as.numeric(rating),
                    timestamp = as.numeric(timestamp))

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

head(movielens)
```

```
##   userId movieId rating timestamp                        title
## 1      1     122      5 838985046             Boomerang (1992)
## 2      1     185      5 838983525              Net, The (1995)
## 3      1     231      5 838983392         Dumb & Dumber (1994)
## 4      1     292      5 838983421              Outbreak (1995)
## 5      1     316      5 838983392              Stargate (1994)
## 6      1     329      5 838983392 Star Trek: Generations (1994)
##                          genres
## 1                Comedy|Romance
## 2         Action|Crime|Thriller
## 3                        Comedy
## 4  Action|Drama|Sci-Fi|Thriller
## 5       Action|Adventure|Sci-Fi
## 6 Action|Adventure|Drama|Sci-Fi
```

```r
# Split the dataset iinto training and test tests
# 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)

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

Data pre-processing and exploratory analysis

Machine learning algorith will be developed using edx set and validation set will be used to test the algorith

Additional Libraries

```
library(ggplot2)
library(lubridate)
library(dslabs)
```

Check few rows of data set

Check few rows of the edx data set to get familiar with the data It contains 6 columns "userID", "movieID,"rating",",”timestamp",”title" and "generes". Each row represents data for rating for a movie

```
head(edx)
```

```
##   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
```

Check Dimensions and Summary stats

Check for the dimensions of the data set to get total no of rows and columns and Summary stats

```
# Rows Columns
dim(edx)
```

```
## [1] 9000055       6
```

```r
# Data set Summary
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  
##                                       
##                                       
## 
```

```r
# check for 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
```

```r
# The total no of unique users are approx 69878 and 10677 movies
```

Define the function for RMSE

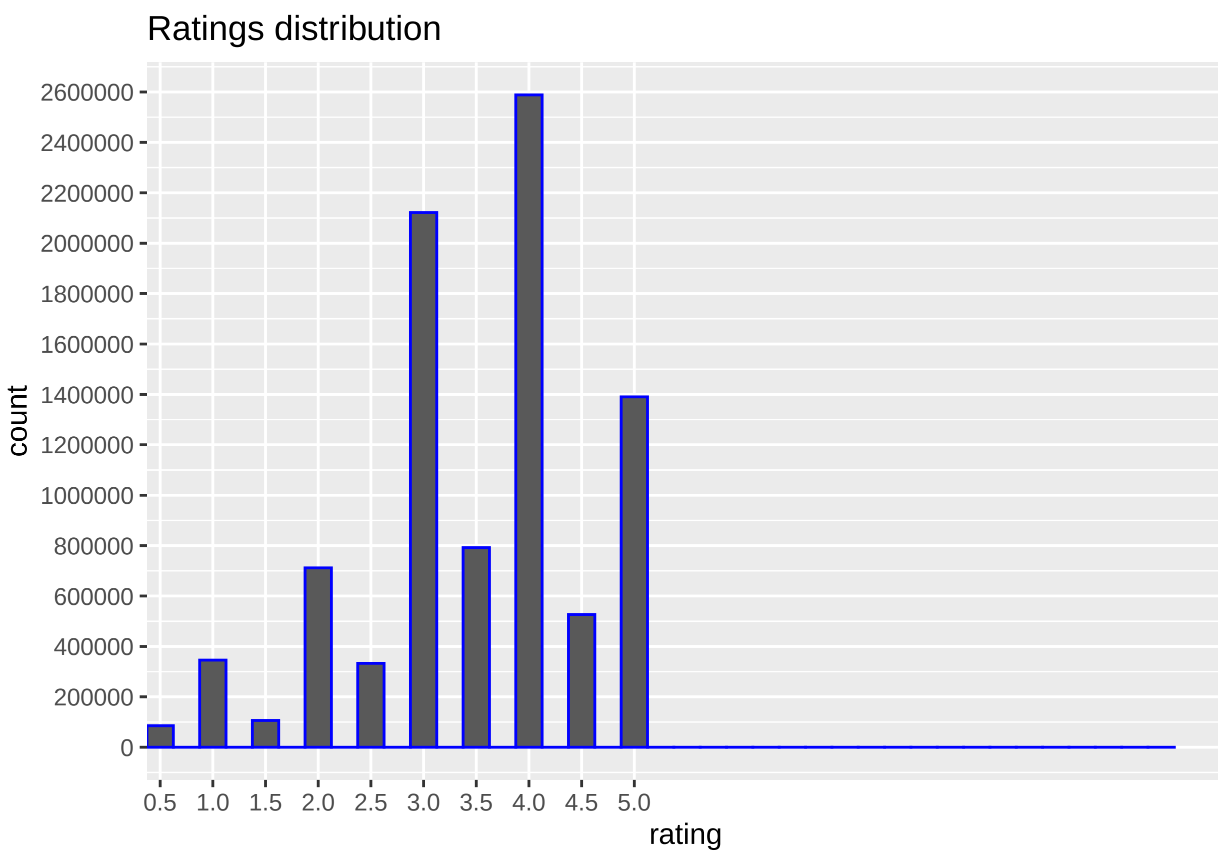
As discussed earlier we will be training the machine learning algorithm and RMSE, Root Mean Sqaure Error will be used to measure the accuracy, we will make frequent use of RMSE so lets define a function for it

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

Ratings distribution

The rating distribution shows that users have mostly rated the movies between 3 and 4. Some movies are rated much often then other while some have very few ratings. We should furthere explore the effect of different features to make a beter prediction.

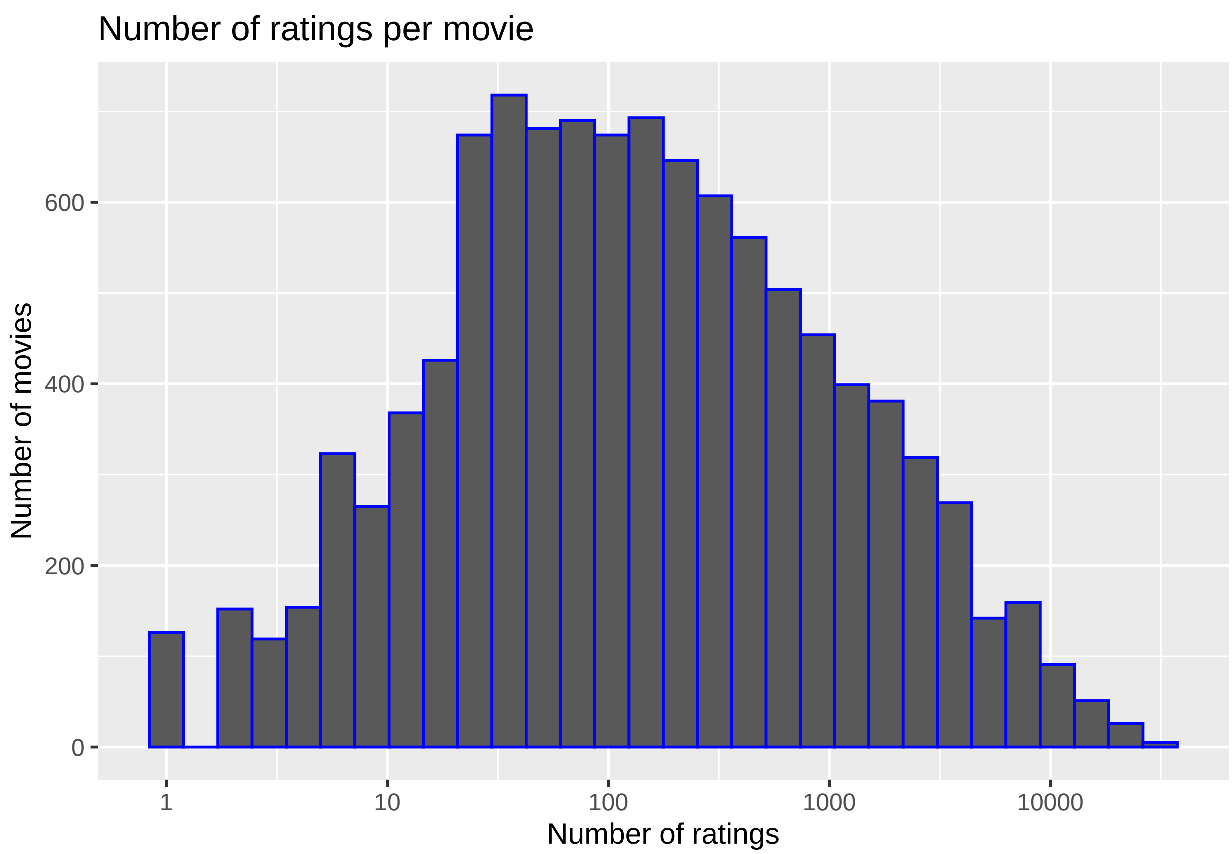
```r
edx %>%
 ggplot(aes(rating)) +
 geom_histogram(binwidth = 0.25, color = "blue") +
 scale_x_discrete(limits = c(seq(0.5,5,0.5))) +
 scale_y_continuous(breaks = c(seq(0, 3000000, 200000))) +
 ggtitle("Ratings distribution")
```



Movies distribution

We can see that some movies are rated more often then others this may lead to movie bias this needs to be incorporated in our model

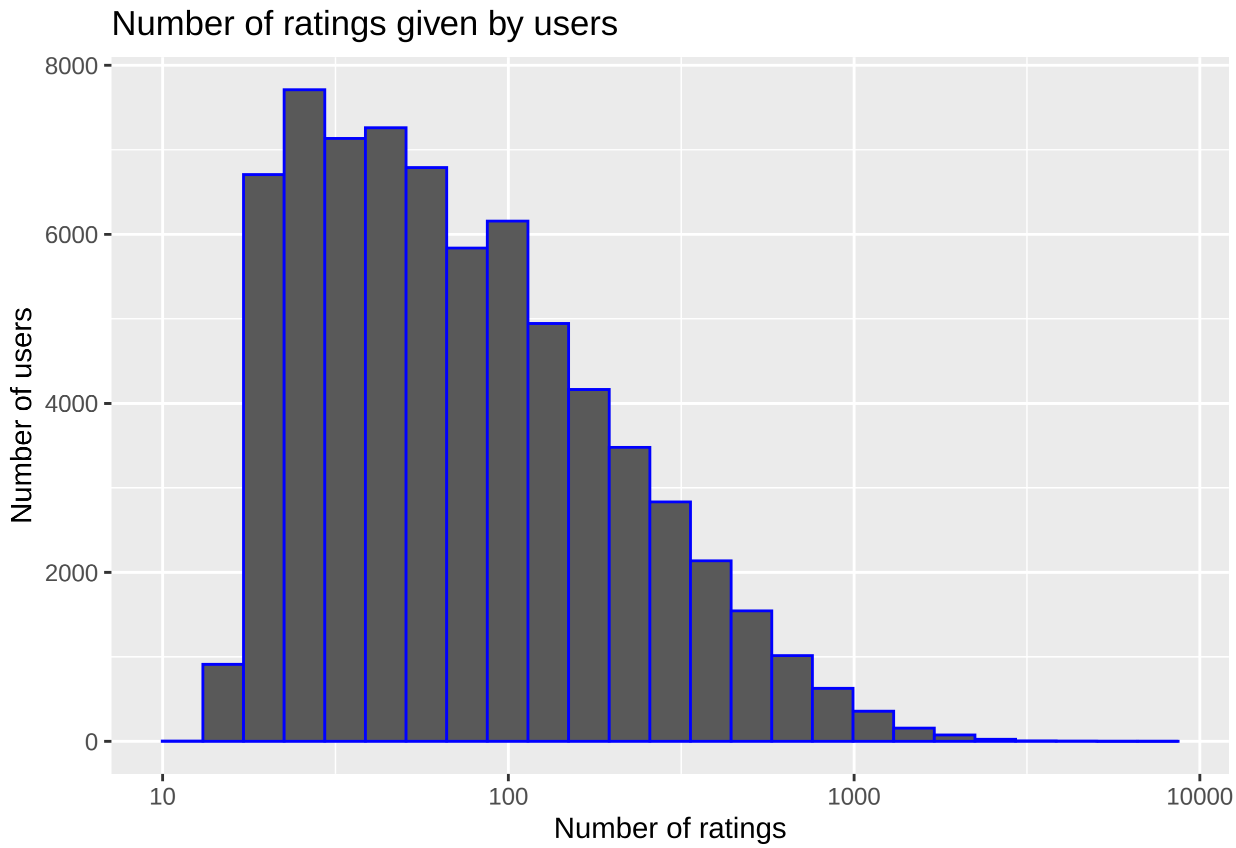
```
edx %>%
  count(movieId) %>%
  ggplot(aes(n)) +
  geom_histogram(bins = 30, color = "blue") +
  scale_x_log10() +
  xlab("Number of ratings") +
  ylab("Number of movies") +
  ggtitle("Number of ratings per movie")
```



User's Distribution

As we can see from the graph users differ the way in which they give ratings some users give much lower ratings and some users give higher ratings. this is user bias and need to be incorporated in the model.

```
edx %>% count(userId) %>%
  ggplot(aes(n)) +
  geom_histogram(bins = 25, color = "blue") +
  scale_x_log10() +
  xlab("Number of ratings") +
  ylab("Number of users") +
  ggtitle("Number of ratings given by users")
```



Modelling and Analysis

Simple : Average movie rating model

In this model we will Compute the mean rating from the edx data set Dataset's mean rating is used to predict the same rating for all movies, regardless of the user and movie. This simple model assumes that all the diferences in movie ratings are explained bt random variable alone. Based on this model the expected rating of the data set is between 3 and 4

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

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

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

Check results Save prediction in data frame

```
# Test results based on simple prediction


rmse_results <- data_frame(method = " model using mean only",
                           RMSE = simple_rmse)
rmse_results %>% knitr::kable()
```

method	RMSE
model using mean only	1.061202

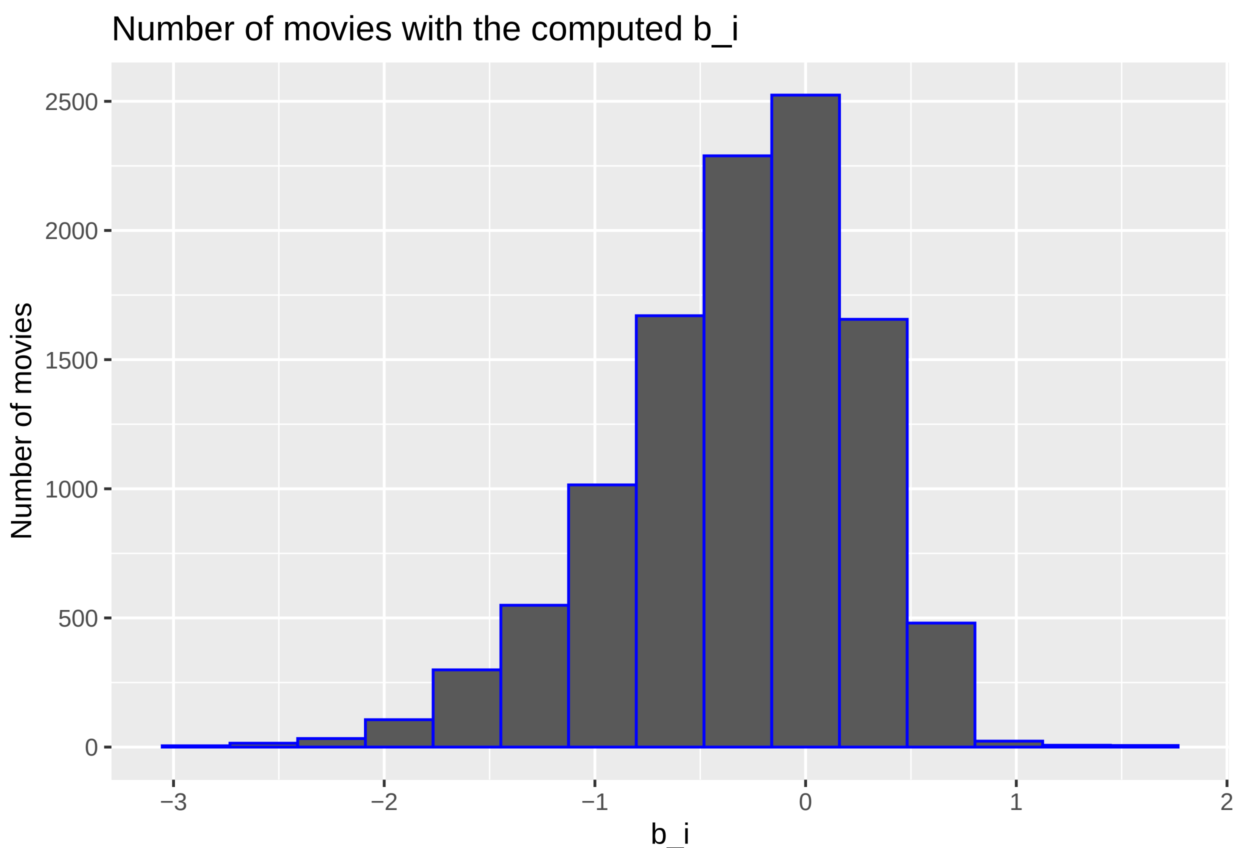
This is baseline RMSE to be compared with next models

Movie effect model

As discussed in data preparation and exploratory data analysis different movies rae rated diffferently some are rated more often and some less then others which leads to movie bias. We will calculate movie effect : penalty term (b_i)

Plot number of movies with the computed b_i

```
movie_avgs <- edx %>%
  group_by(movieId) %>%
  summarize(b_i = mean(rating - mu))
movie_avgs %>% qplot(b_i, geom ="histogram", bins = 15,
                     data = ., color = I("blue"),
                     ylab = "Number of movies",
                     main = "Number of movies with the computed b_i")
```




```
# Test and save rmse results
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))
# Check results
rmse_results %>% knitr::kable()
```

method	RMSE
model using mean only	1.0612018
Movie effect model	0.9439087

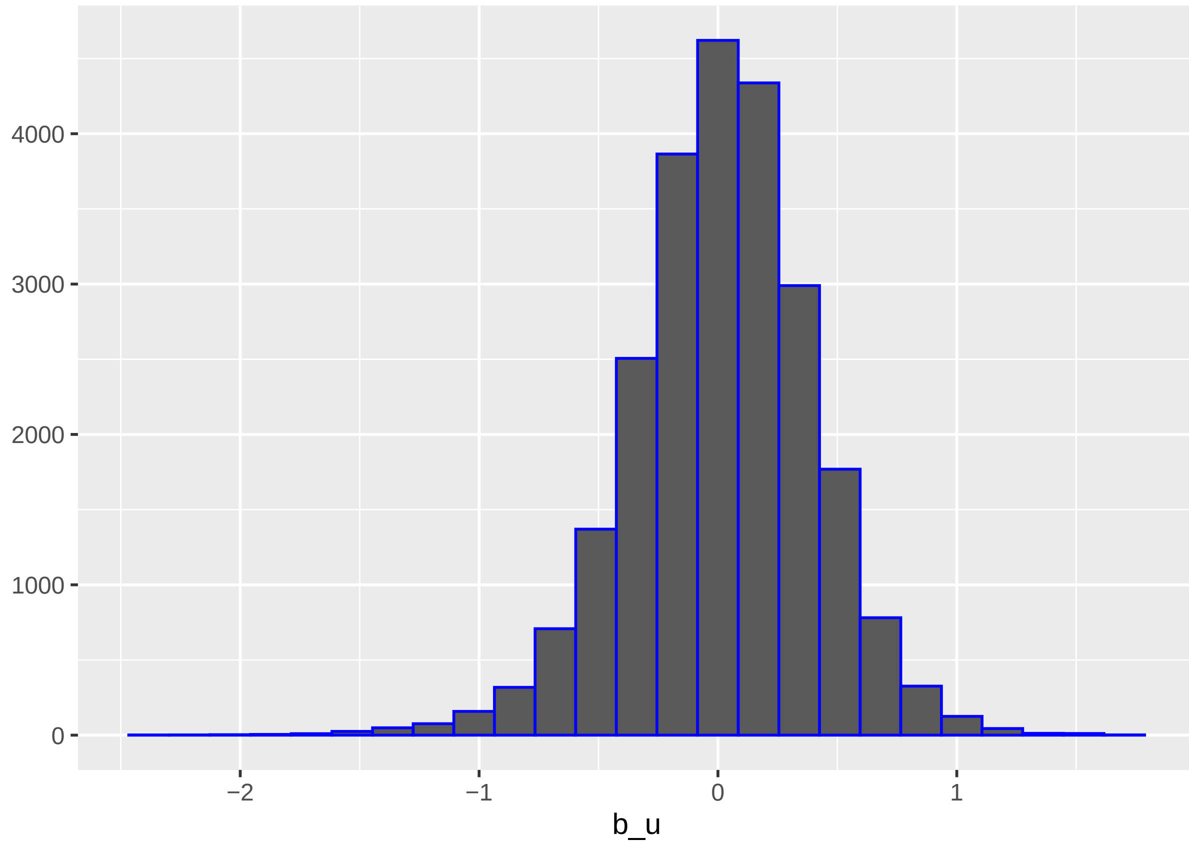
We can see some improvment in the prediction, we can further improve by taking into account User effect as well

Movie and user effect model

As discussed in data preparation and exploratory data analysis users differ the way in which they give ratings some users give much lower ratings and some users give higher ratings which leads to user bias We will calculate penalty user effect penalty term (b_u)

Since movie and user bias those affect the prediction of movie rating a further improvement is achieved by adding the user effect

```
# Plot penaly term user effect(b_u)
# join movie averages & user averages

predicted_ratings_user_avgs<- edx %>%
  left_join(movie_avgs, by='movieId') %>%
  group_by(userId) %>%
  filter(n() >= 100) %>%
  summarize(b_u = mean(rating - mu - b_i))
predicted_ratings_user_avgs %>% qplot(b_u, geom ="histogram", bins = 25, data = ., color = I("blue"))
```



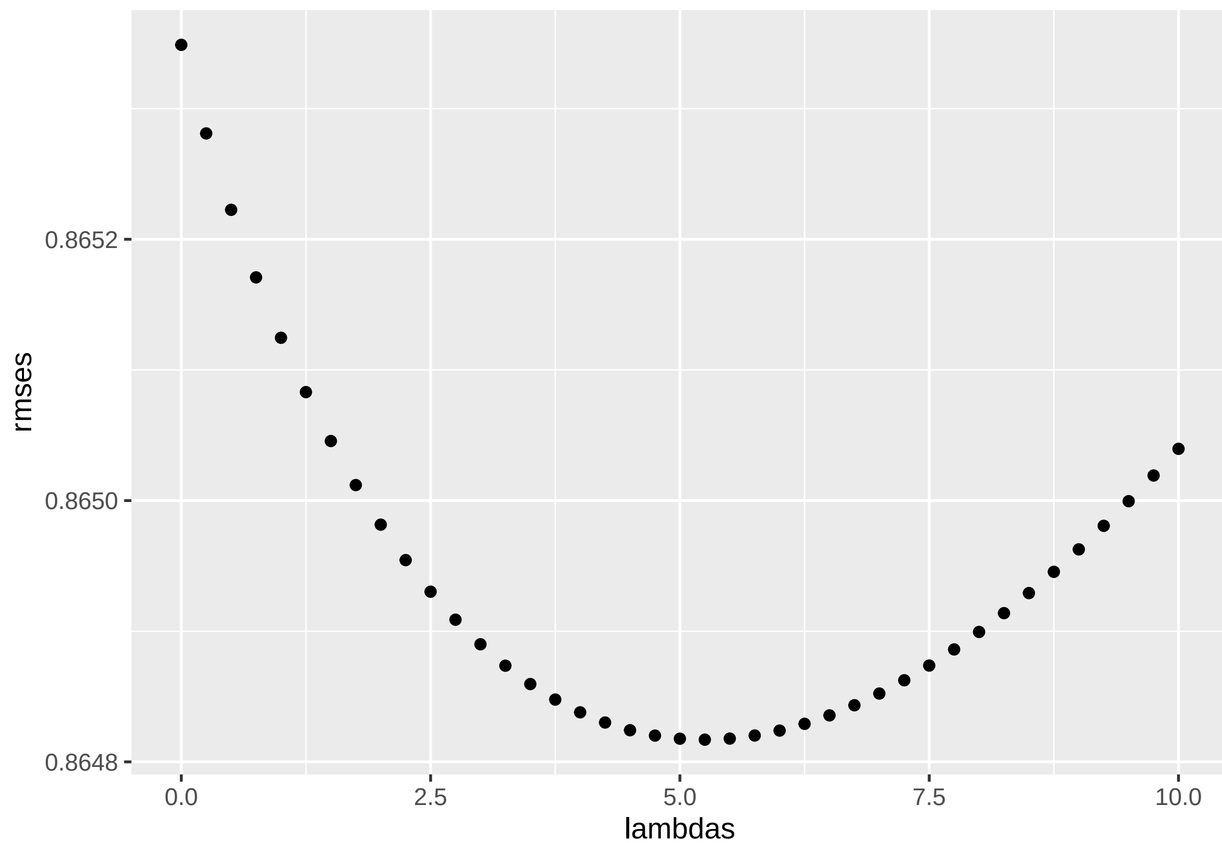

```
predicted_ratings_user_avgs <- edx %>%
  left_join(movie_avgs, by='movieId') %>%
  group_by(userId) %>%
  summarize(b_u = mean(rating - mu - b_i))


# Test and save rmse results
predicted_ratings <- validation%>%
  left_join(movie_avgs, by='movieId') %>%
  left_join(predicted_ratings_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",

# Check result
rmse_results %>% knitr::kable()
```

method	RMSE
model using mean only	1.0612018
Movie effect model	0.9439087
Movie and user effect model	0.8653488

We can see that we have further reduced the RMSE. Till now we have computed the RMSE based on different levels of uncertainty. we have noticed in the exploratory analysis that some movies are highly rated and some are not that actively rated may very few times Also some users actively give ratings and some do not. We will now add Regularization effect to reduce the effect of overfitting in our estimations in the movie and user effect model. We will find the value of lambda (tunning parameter) which will further minimise the RMSE

Regularized Movie and user effect model

```
# We need to start by taking a few lambda values starting from zero
lambdas <- seq(0, 10, 0.25)


# For each lambda,find b_i & b_u, followed by rating prediction & testing

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))
})


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




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

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

```
# Test and save results
rmse_results <- bind_rows(rmse_results,
                          data_frame(method="Regularized movie and user effect model",
                                     RMSE = min(rmses)))

# Check results
rmse_results %>% knitr::kable()
```

method	RMSE
model using mean only	1.0612018
Movie effect model	0.9439087
Movie and user effect model	0.8653488
Regularized movie and user effect model	0.8648170

```
rmse_results
```

```
## # A tibble: 4 x 2
```

```
##   method                                      RMSE
##   <chr>                                      <dbl>
## 1 " model using mean only"                   1.06 
## 2 "Movie effect model"                       0.944
## 3 "Movie and user effect model"              0.865
## 4 "Regularized movie and user effect model" 0.865
```

Results

The RMSE values of all the represented models are the following: method RMSE Average movie rating model 1.0612018 Movie effffect model 0.9439087 Movie and user effffect model 0.8653488 Regularized movie and user effffect model 0.8648170

We therefore found the lowest value of RMSE that is 0.8648170

Conclusion

Based on various models as explained in the Modeling section we have developed a machine learning algorithm to predict ratings using MovieLens dataset. we achived to get a lower RMSE by taking into consideration movie effect and user effect bias. we further optimised it by applying Regularization.

The Final Model produced an RMSE of 0.8648170