Movielens Project Report

Mangalam Khare

23rd July 2021

Introduction

This project is part of the HarvardX course Capstone Movielens project. The objective of this project is to develop Machine learning algorithm using the "10 M version of MovieLens dataset.Several models has been used and results have been compared to get the smallest RMSE possible as a measure of evaluating model performance

RMSE, Root Mean Square Error is the measure of the differences between predicted values and actual/observed values.

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

Problem Statement

The objective of this project is to use machine learning techniques that predicts user ratings using the data present in the MovieLens dataset (trainset : edx) and validate them with tests set (Validation). 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

```
#############################################################
# Create edx set, validation set
#############################################################

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)

library(ggplot2)
```

```r
library(lubridate)
library(dslabs)

# MovieLens 10M dataset:

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")
```

```r
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

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. The Title Column is combination of the Title of the movie and the year. We will split this column to take out year part to add one more feature for the analysis

```r
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

```r
# 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  
##                                       
##                                       
## 
```

Add a new column year to the edx and validation data set by splitting Title column - to add one more feature for the analysis

```r
#add a new column year to the edx and validation data set by splitting Title column
edx <- edx %>% mutate(year = as.numeric(str_sub(title,-5,-2)))
validation <- validation %>% mutate(year = as.numeric(str_sub(title,-5,-2)))

dim(edx)
```

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

```r
# check for the number of unique movies and users in the edx dataset
# The total no of unique users are approx 69878 and 10677 movies

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

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

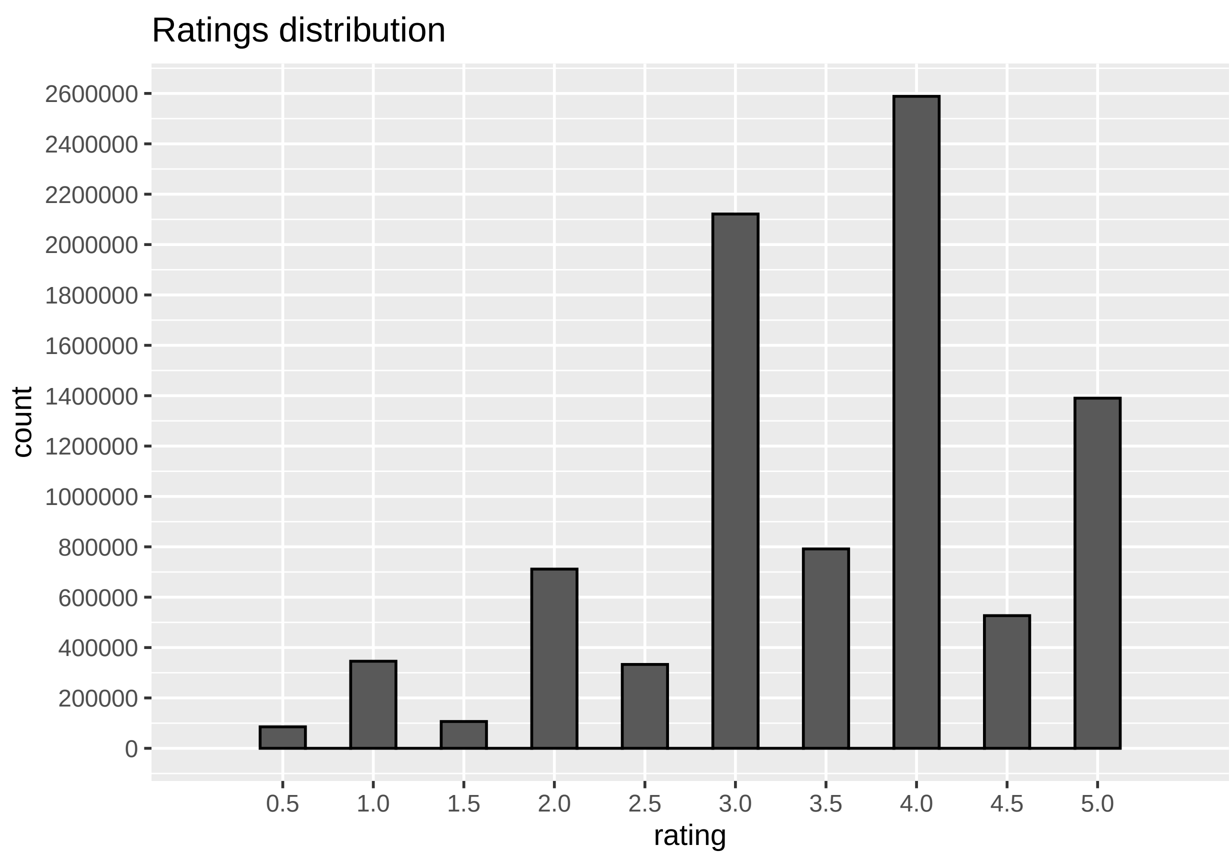
Define the function for RMSE

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

Ratings distribution

The rating distribution shows that most of the ratings are between 3 and 4. Some movies are rated much often then other while some have very few ratings. we will check diffrent features to see the effects of them on rating

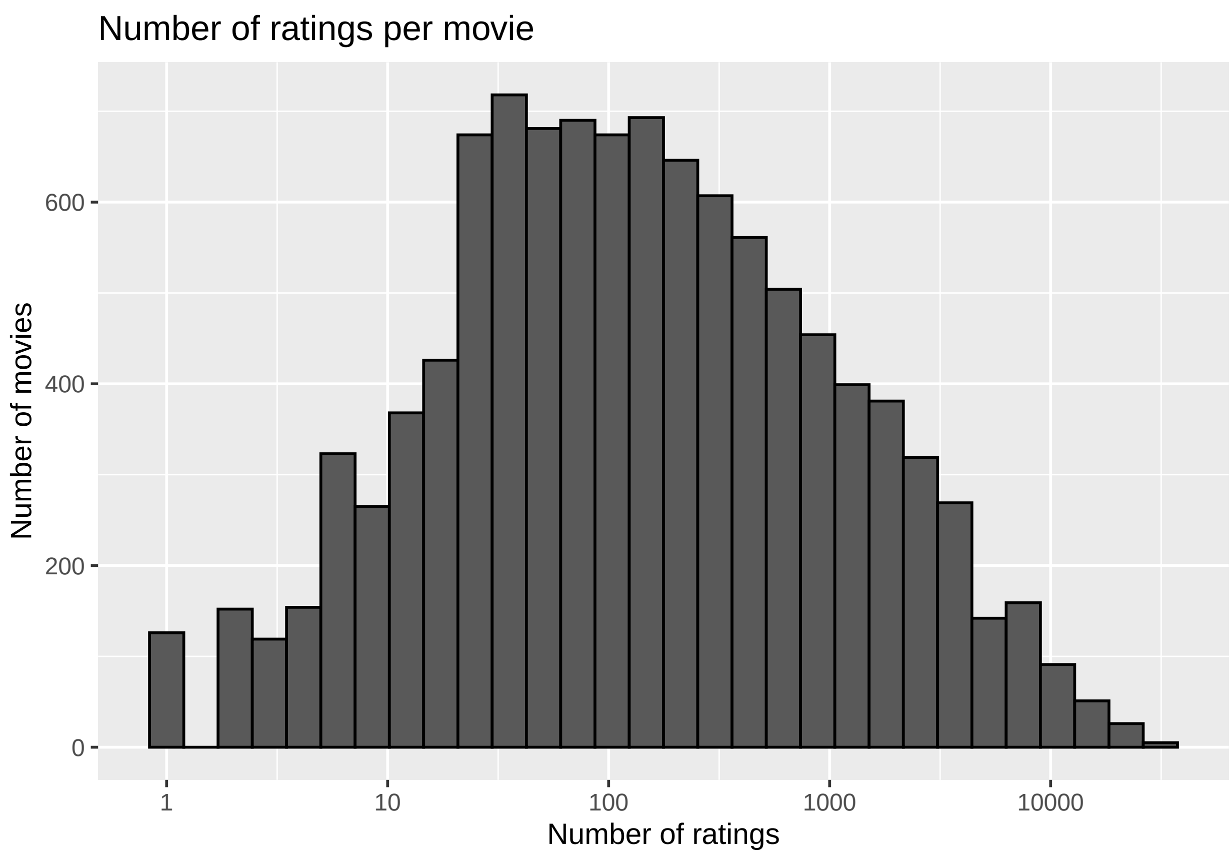
```r
# Ratings distribution
edx %>%
  ggplot(aes(rating)) +
  geom_histogram(binwidth = 0.25, color = "black") +
  scale_x_discrete(limits = c(seq(0.5,5,0.5))) +
  scale_y_continuous(breaks = c(seq(0, 3000000, 200000))) +
  ggtitle("Ratings distribution")
```



Movies distribution

We can see that some movies are rated more often then others this may lead to movie bias.

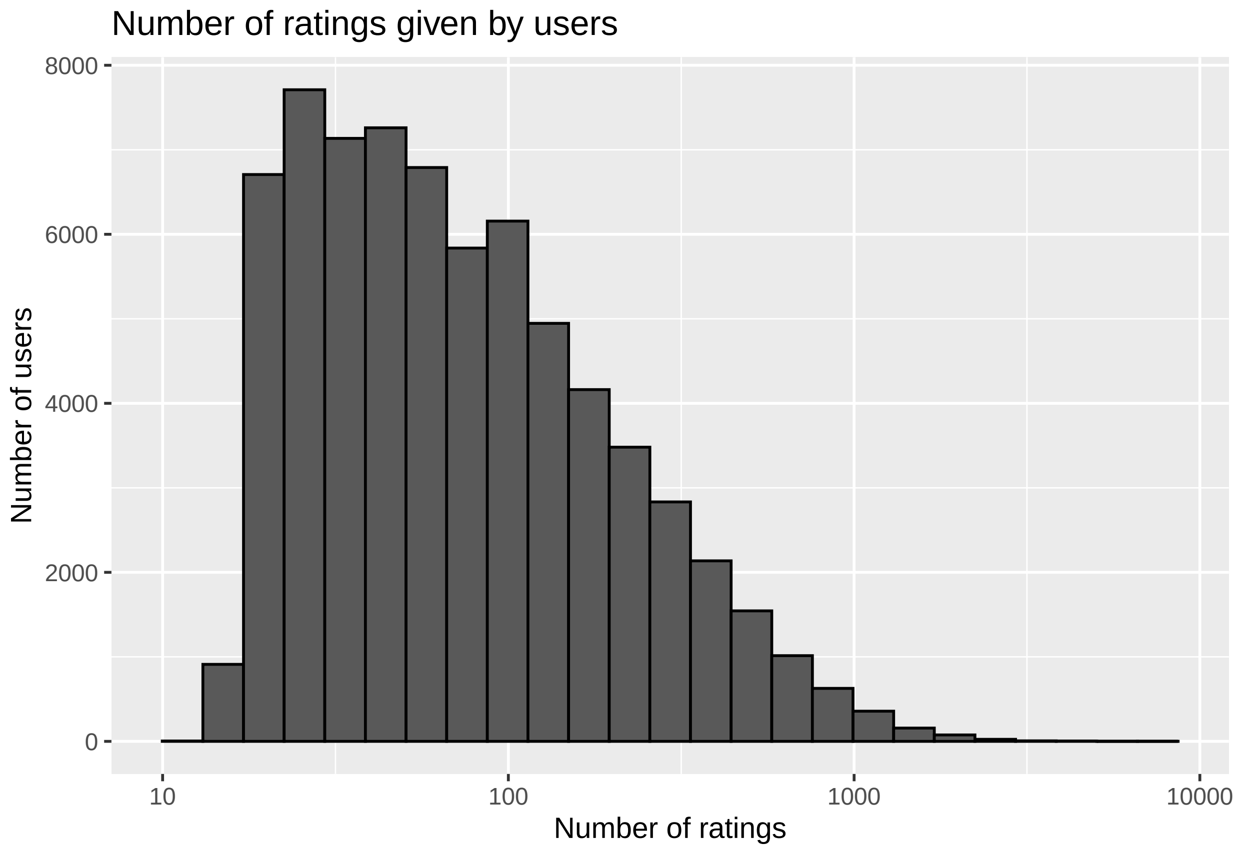
```r
# Movies distribution

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")
```



User's Distribution

As we can see from the graph different users have given ratings differently some have given low some have given higher. This may lead to user bias.

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



Modelling and Analysis

We will first run various models using edx set and validate final model with validation set. First we will split edx set into train and test set to build model and then run the final model on full edx and validation set to get final RMSE

```
## Split the edx dataset into training and test tests
## test set will be 10% of edx data

set.seed(1, sample.kind="Rounding")
test_index <- createDataPartition(y = edx$rating, times = 1, p = 0.1, list = FALSE)
edx_train <- edx[-test_index,]
edx_temp <- edx[test_index,]


# Make sure userId and movieId in test set are also in train set

edx_test <- edx_temp %>%
  semi_join(edx_train, by = "movieId") %>%
  semi_join(edx_train, by = "userId")

# Add rows removed from validation set back into edx set

removed <- anti_join(edx_temp, edx_test)
```

```r
edx_train <- rbind(edx_train, removed)

rm(test_index, edx_temp,removed)

head(edx_train)
```

```
##   userId movieId rating timestamp                         title
## 1      1     122      5 838985046              Boomerang (1992)
## 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)
## 8      1     356      5 838983653           Forrest Gump (1994)
##                           genres year
## 1                 Comedy|Romance 1992
## 4  Action|Drama|Sci-Fi|Thriller 1995
## 5       Action|Adventure|Sci-Fi 1994
## 6 Action|Adventure|Drama|Sci-Fi 1994
## 7       Children|Comedy|Fantasy 1994
## 8      Comedy|Drama|Romance|War 1994
```

```r
head(edx_test)
```

```
##    userId movieId rating  timestamp
## 2       1     185      5  838983525
## 25      2     260      5  868244562
## 28      2     590      5  868245608
## 37      2    1049      3  868245920
## 39      2    1210      4  868245644
## 47      3    1148      4 1133571121
##                                                         title
## 2                                                Net, The (1995)
## 25 Star Wars: Episode IV - A New Hope (a.k.a. Star Wars) (1977)
## 28                                     Dances with Wolves (1990)
## 37                              Ghost and the Darkness, The (1996)
## 39             Star Wars: Episode VI - Return of the Jedi (1983)
## 47                   Wallace & Gromit: The Wrong Trousers (1993)
##                                 genres year
## 2             Action|Crime|Thriller 1995
## 25          Action|Adventure|Sci-Fi 1977
## 28          Adventure|Drama|Western 1990
## 37                 Action|Adventure 1996
## 39          Action|Adventure|Sci-Fi 1983
## 47 Animation|Children|Comedy|Crime 1993
```

```r
dim(edx_train)
```

```
## [1] 8100065       7
```

```r
dim(edx_test)
```

```
## [1] 899990      7
```

Simple : Average movie rating model

In this model we will predict the same rating for all movies using mean.

```
#Simple : Average movie rating model
mu <- mean(edx_train$rating)
mu
```

```
## [1] 3.512456
```

Test results based on simple prediction

```
rmse <- RMSE(edx_test$rating, mu)
rmse
```

```
## [1] 1.060054
```

Check results, Save prediction in data frame

```
rmse_results <- data_frame(Method = " Average movie rating model",
                           RMSE = rmse)
rmse_results %>% knitr::kable()
```

Method	RMSE
Average movie rating model	1.060054

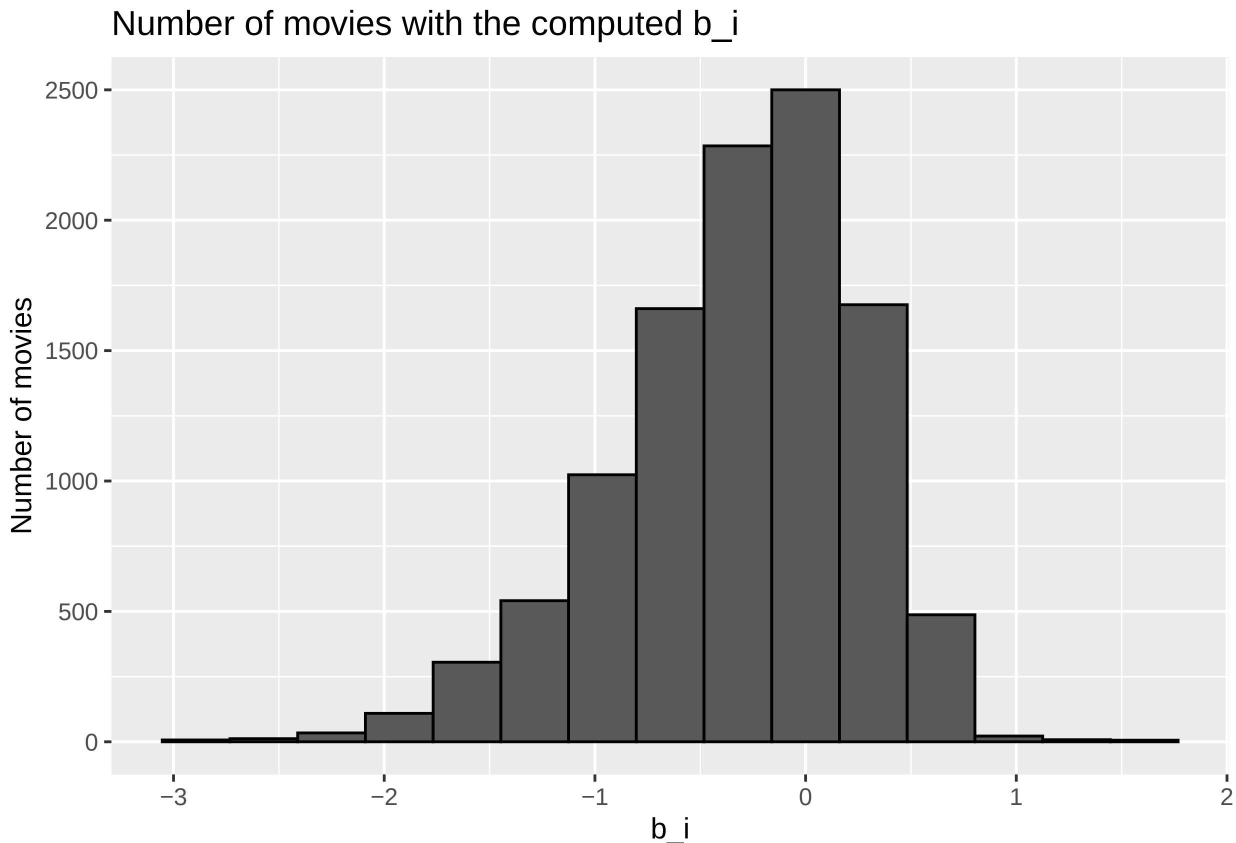
This is baseline RMSE to be compared with next models

Movie effect model

As discussed in data preparation and exploratory data analysis different movies are rated differently some are rated more often and some less then others. Also some movies are generally rated higher, which leads to movie bias. We will calculate movie effect : b_i

Plot number of movies with the computed b_i

```
# Number of movies with the computed b_i

movie_avgs <- edx_train %>%
  group_by(movieId) %>%
  summarize(b_i = mean(rating - mu))

qplot(b_i,
      bins = 15,
      data = movie_avgs, color = I("black"),
      ylab = "Number of movies",
      main = "Number of movies with the computed b_i")
```




```
# Test and save rmse results

predicted_ratings <- mu +  edx_test %>%
  left_join(movie_avgs, by='movieId') %>%
  pull(b_i)

rmse <- RMSE(predicted_ratings, edx_test$rating)

rmse_results <- bind_rows(rmse_results,
                          data_frame(Method="Movie effect model",
                                     RMSE = rmse))
# Check results
rmse_results %>% knitr::kable()
```

Method	RMSE
Average movie rating model	1.0600537
Movie effect model	0.9429615

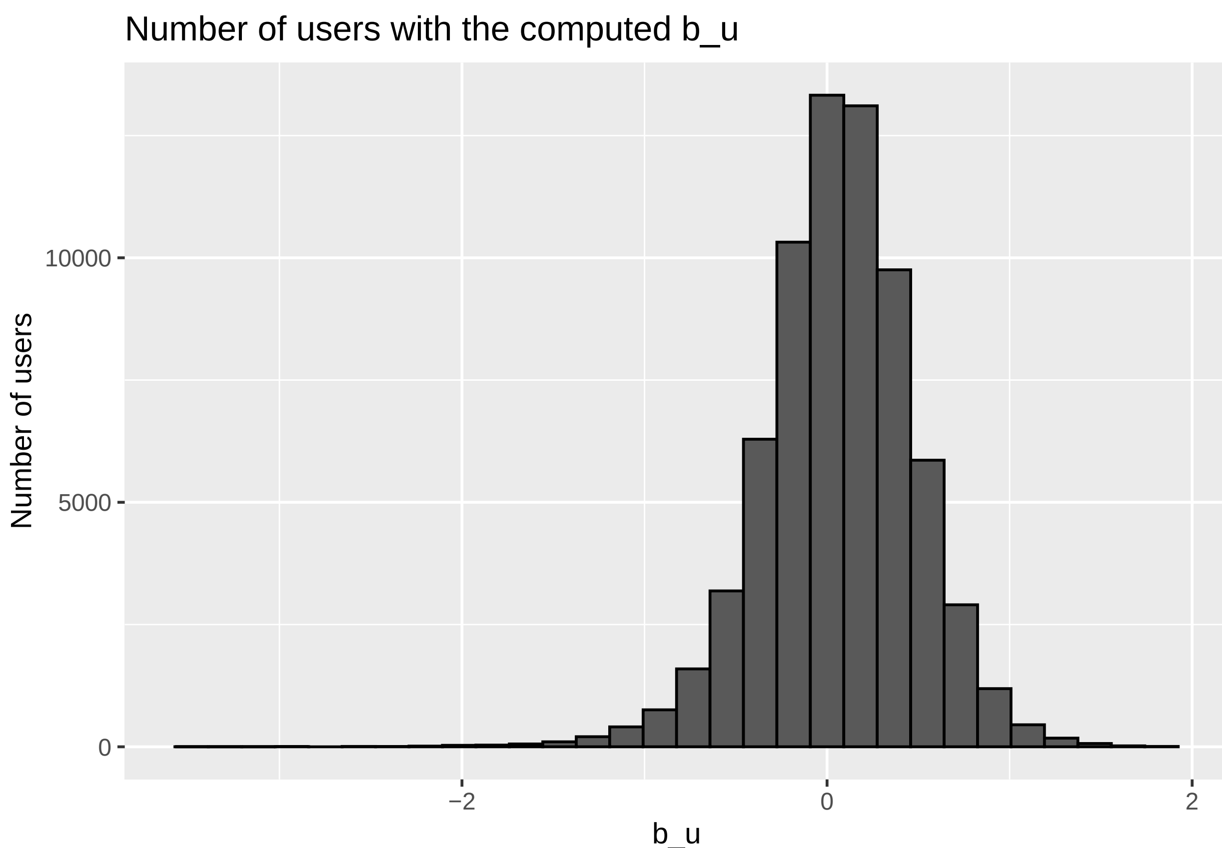
We can see some improvement in the RMSE, we can further improve by taking into account User effect

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)

A further improvement is achieved by adding the user effect

```
# Model considering Movie effect , user effect

# Plot user effect

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

  qplot(b_u,
      bins = 30,
      data = user_avgs, color = I("black"),
      ylab = "Number of users",
      main = "Number of users with the computed b_u")
```




```
# Test and save rmse results

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

```r
rmse <- RMSE(predicted_ratings, edx_test$rating)
rmse_results <- bind_rows(rmse_results,
                          data_frame(Method="Movie and user effect model",
                                     RMSE = rmse))

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

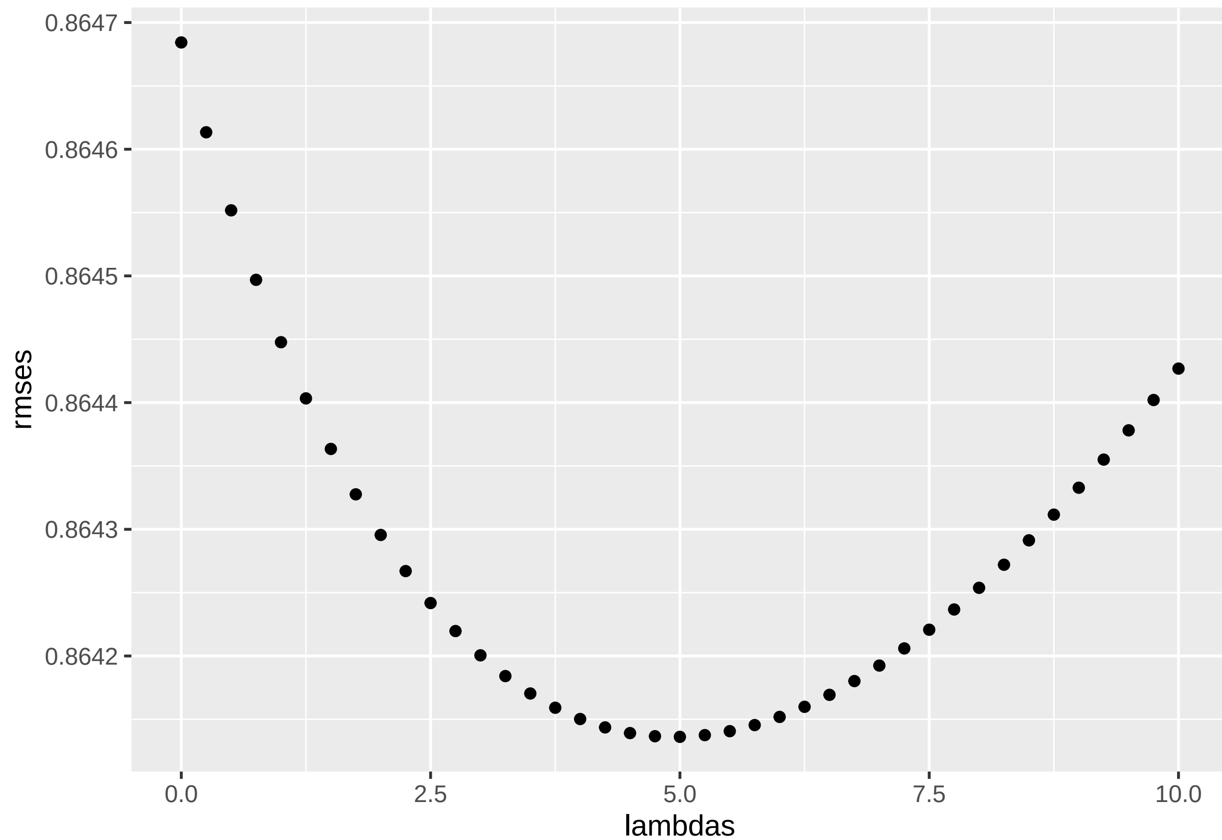
Method	RMSE
Average movie rating model	1.0600537
Movie effect model	0.9429615
Movie and user effect model	0.8646843

WE can see that we have further reduced the RMSE. Till now we have computed the RMSE based on different predictors and considering the bias based on that. we have noticed in the exploratory analysis that some movies are highly rated and some are not that actively rated , same is the case for users. This variability of the size effect is taken into consideration by using Regularization We will now add Regularization effect to the estimations in the movie and user effect model by using Lambda a tuning parameter to see if we are able to further reduce the RMSE

Regularized Movie and user effect model

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

rmses <- sapply(lambdas, function(l){

  mu <- mean(edx_train$rating)

  b_i <- edx_train %>%
    group_by(movieId) %>%
    summarize(b_i = sum(rating - mu)/(n()+l))

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

  predicted_ratings <-
    edx_test %>%
    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, edx_test$rating))
})
```

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




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

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

```r
# 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()
```

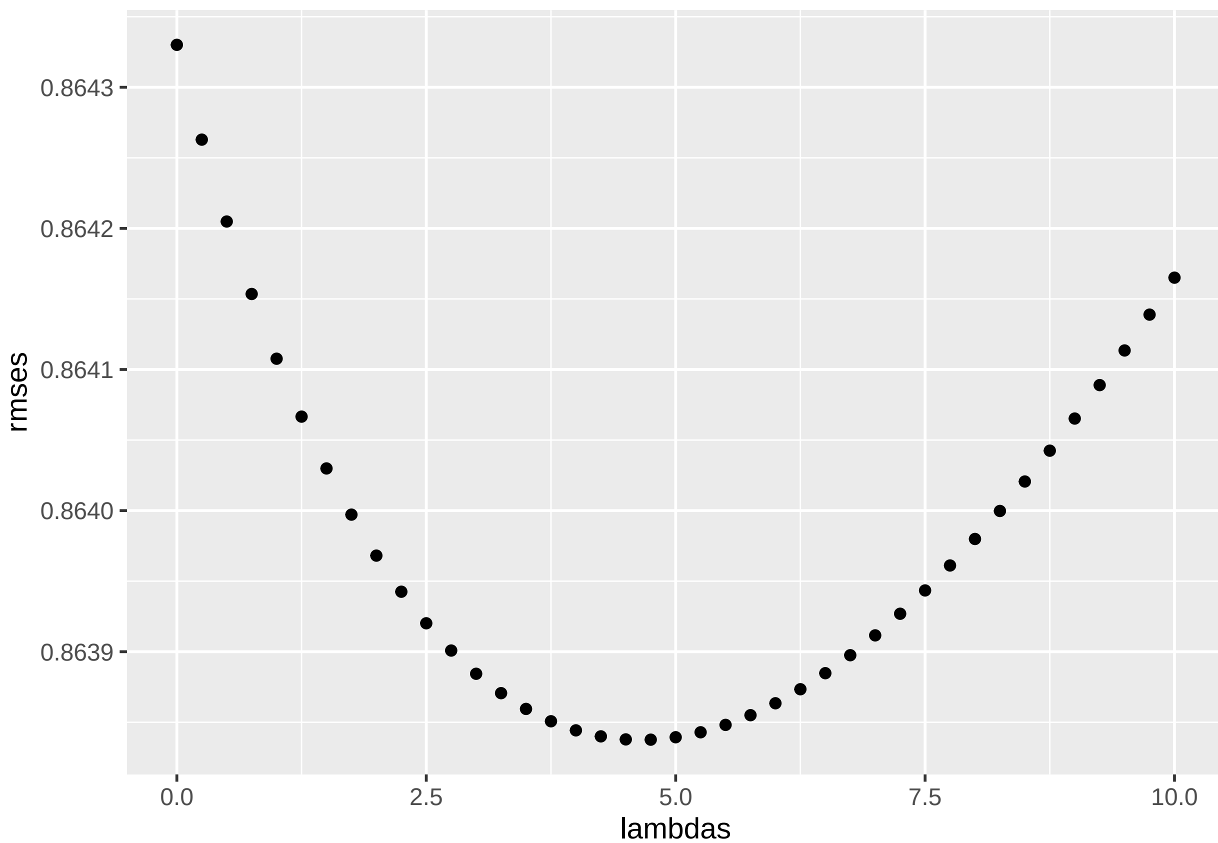
Method	RMSE
Average movie rating model	1.0600537
Movie effect model	0.9429615
Movie and user effect model	0.8646843
Regularized movie and user effect model	0.8641362

We can see that we have further reduced the value of RMSE

We will try to further reduce the RMSE by considering another predictor year (newly added column by splitting the title)

Regularized Movie and user effect model by ading new predictor year

```
# Regularization with Year effect

lambdas <- seq(0, 10, 0.25)

rmses <- sapply(lambdas, function(l){

  mu <- mean(edx_train$rating)

  b_i <- edx_train %>%
    group_by(movieId) %>%
    summarize(b_i = sum(rating - mu)/(n()+l))

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

  b_y <- edx_train %>%
    left_join(b_i, by='movieId') %>%
    left_join(b_u, by='userId') %>%
    group_by(year) %>%
    summarize(b_y = sum(rating - b_i - mu - b_u)/(n()+l))

  predicted_ratings <-
    edx_test %>%
    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(predicted_ratings, edx_test$rating))
})


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




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

```
## [1] 4.75
```

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

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

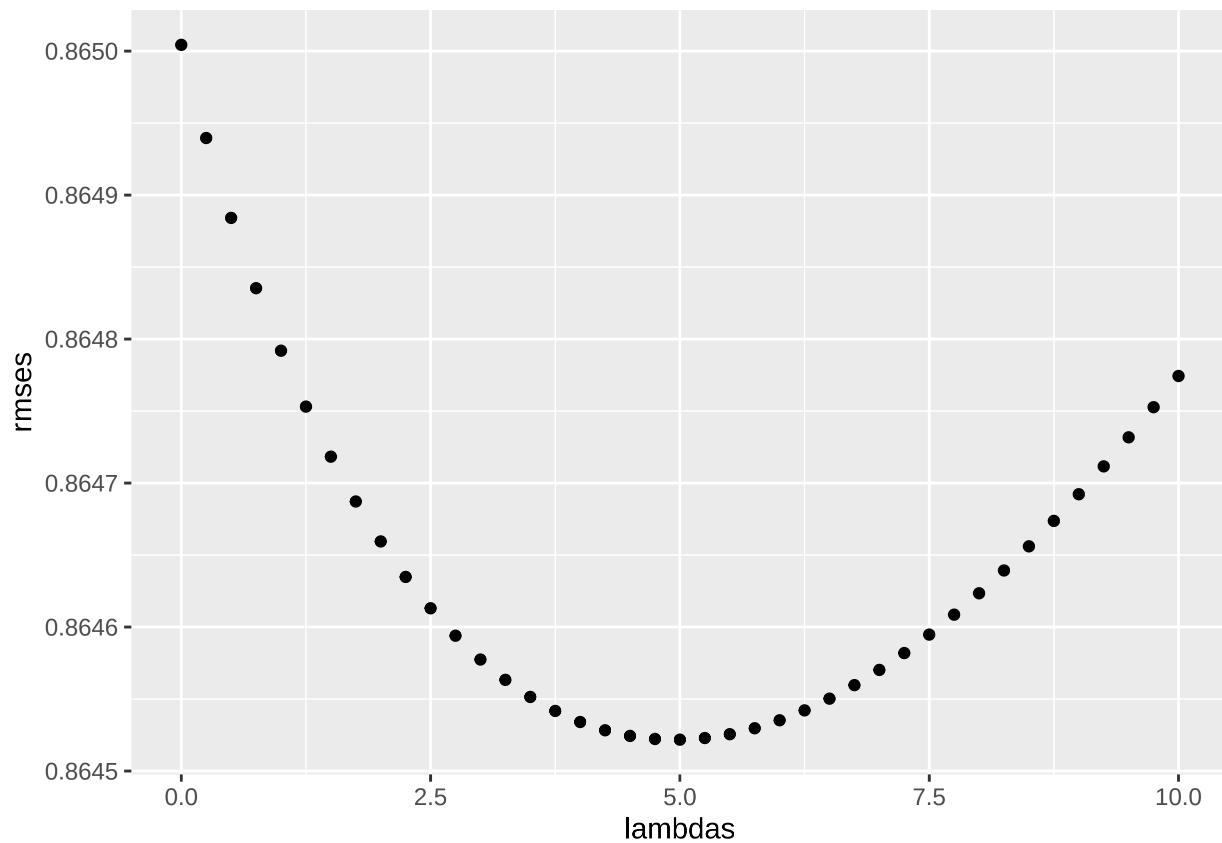
Method	RMSE
Average movie rating model	1.0600537
Movie effect model	0.9429615
Movie and user effect model	0.8646843
Regularized movie and user effect model	0.8641362
Regularized movie, user and year effect model	0.8638377

As we can see form the results that we were further able to reduce RMSE by adding a new predictor year.

This will be our final model and we will run this model with full edx and validation set to get the final RMSE

Final Model on Full edx and validation set

```
# Final Model ( Movie + User Effect+year + Regularization) on full edx and Validation test

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))

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

  predicted_ratings <-
    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(predicted_ratings, validation$rating))
})


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




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

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

```r
# Test and save results
rmse_results <- bind_rows(rmse_results,
                          data_frame(Method="Final model on full edx set and validation set",
                                     RMSE = min(rmses)))

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

Method	RMSE
Average movie rating model	1.0600537
Movie effect model	0.9429615
Movie and user effect model	0.8646843
Regularized movie and user effect model	0.8641362
Regularized movie, user and year effect model	0.8638377
Final model on full edx set and validation set	0.8645218

Results

The RMSE values of all the represented models are the following:

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

Method	RMSE
Average movie rating model	1.0600537
Movie effect model	0.9429615
Movie and user effect model	0.8646843
Regularized movie and user effect model	0.8641362
Regularized movie, user and year effect model	0.8638377
Final model on full edx set and validation set	0.8645218

Conclusion

We have used various models as explained in the modelling section to develop machine learning algorithm to predict ratings using Movie lens data set. The RMSE's kept reducing as we added different predictors in the analysis. It improved further by adding Regularization. The RMSE's was reduced further when we added new predictor year to the Regularization model. We then used this model on the full edx and validation set to get final RMSE

The Final RMSE value obtained was -

```
min(rmses)
```

```
## [1] 0.8645218
```

Future work

We can try to further reduce the RMSE by adding another predictor for genre column. It's the combination of various genres in the Movie lens data set we can separate them into multiple rows to have one genre per row and try to predict ratings by adding genre predictor in our analysis.