# Movielens Project

Kuntal Bhar

## INTRODUCTIONS

Build a machine learning alogrithm to provide movie recommendation system. The goal is to understand the data structure, visualize the data and creating a movie recommendation system that build the best model that can predict predict movie ratings for users in a large moveilens collected by GroupLens Research dataset with accuracy. This has 4 main section and its subsection which include Introduction & Objectives where we presented the problem and its objectives, Dataset where it analyze the data, Methods that contained modes/implement applied, Conclusion section share result summary.

#### Objective

The objective of this project is to train a linear model alogrithim that predicts user ratings and calculate Root Mean Square Error (RMSE) of the predicted ratings versus the actual ratings. We train machine language alogrithm on traning dataset (edx as provided) to predict movie ratings in test dataset (final_holdout_test as provided). We develop four methods and compare their resulting RMSE, then best resulting method will be used to predict the movie ratings.

## DATASET ANALYSIS

Provided dataset is created. It create training set named ad 'edx' and test set named as 'final_holdout_test'

```r
# **** Create edx set, final_holdout_test 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")
library(tidyverse)
library(caret)

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

options(timeout = 120)

dl <- "ml-10M100K.zip"
if(!file.exists(dl))
  download.file("https://files.grouplens.org/datasets/movielens/ml-10m.zip", dl)

ratings_file <- "ml-10M100K/ratings.dat"
if(!file.exists(ratings_file))
  unzip(dl, ratings_file)
```

```r
movies_file <- "ml-10M100K/movies.dat"
if(!file.exists(movies_file))
  unzip(dl, movies_file)

ratings <- as.data.frame(str_split(read_lines(ratings_file), fixed("::"), simplify = TRUE),
                         stringsAsFactors = FALSE)
colnames(ratings) <- c("userId", "movieId", "rating", "timestamp")
ratings <- ratings %>%
  mutate(userId = as.integer(userId),
         movieId = as.integer(movieId),
         rating = as.numeric(rating),
         timestamp = as.integer(timestamp))

movies <- as.data.frame(str_split(read_lines(movies_file), fixed("::"), simplify = TRUE),
                        stringsAsFactors = FALSE)
colnames(movies) <- c("movieId", "title", "genres")
movies <- movies %>%
  mutate(movieId = as.integer(movieId))

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

# Final hold-out test set will be 10% of MovieLens data
set.seed(1, sample.kind="Rounding") # if using R 3.6 or later
# set.seed(1) # if using R 3.5 or earlier
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 final hold-out test set are also in edx set
final_holdout_test <- temp %>%
  semi_join(edx, by = "movieId") %>%
  semi_join(edx, by = "userId")

# Add rows removed from final hold-out test set back into edx set
removed <- anti_join(temp, final_holdout_test)
edx <- rbind(edx, removed)

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

## Data Analysis

We use Movie Lens Dataset with 10 million data with over 10,000 movie. *Pulp Fiction* (1994) has highest rating and over 100 movies rated atleast once.

The dataset is split 90-10 on train and test sets respectively.

```r
# Most rated films
edx %>% group_by(title) %>%
  summarize(topn_ratings = n()) %>%
  arrange(desc(topn_ratings))
```

```
## # A tibble: 10,676 x 2
```

```
##    title                                                          topn_ratings
##    <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
```

```
# Number of movies rated once
edx %>% group_by(title) %>%
  summarize(topn_ratings = n()) %>%
  filter(topn_ratings==1) %>%
  count() %>% pull()
```

```
## [1] 126
```

The training set (edx) has 9,000,055 entries with 6 columns. The subset contain the six columns "userID", "movieID", "rating", "timestamp", "title", and "genres".

```
## 'data.frame':    9000055 obs. of  6 variables:
##  $ userId   : int  1 1 1 1 1 1 1 1 1 1 ...
##  $ movieId  : int  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 8
##  $ 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
```

The test set (final_holdout_test) has 999,999 entries and 6 columns same as edx

```
## 'data.frame':    999999 obs. of  6 variables:
##  $ userId   : int  1 1 1 2 2 2 3 3 4 4 ...
##  $ movieId  : int  231 480 586 151 858 1544 590 4995 34 432 ...
##  $ rating   : num  5 5 5 3 2 3 3.5 4.5 5 3 ...
##  $ timestamp: int  838983392 838983653 838984068 868246450 868245645 868245920 1136075494 1133571200
##  $ title    : chr  "Dumb & Dumber (1994)" "Jurassic Park (1993)" "Home Alone (1990)" "Rob Roy (1995)
##  $ genres   : chr  "Comedy" "Action|Adventure|Sci-Fi|Thriller" "Children|Comedy" "Action|Drama|Roman
```

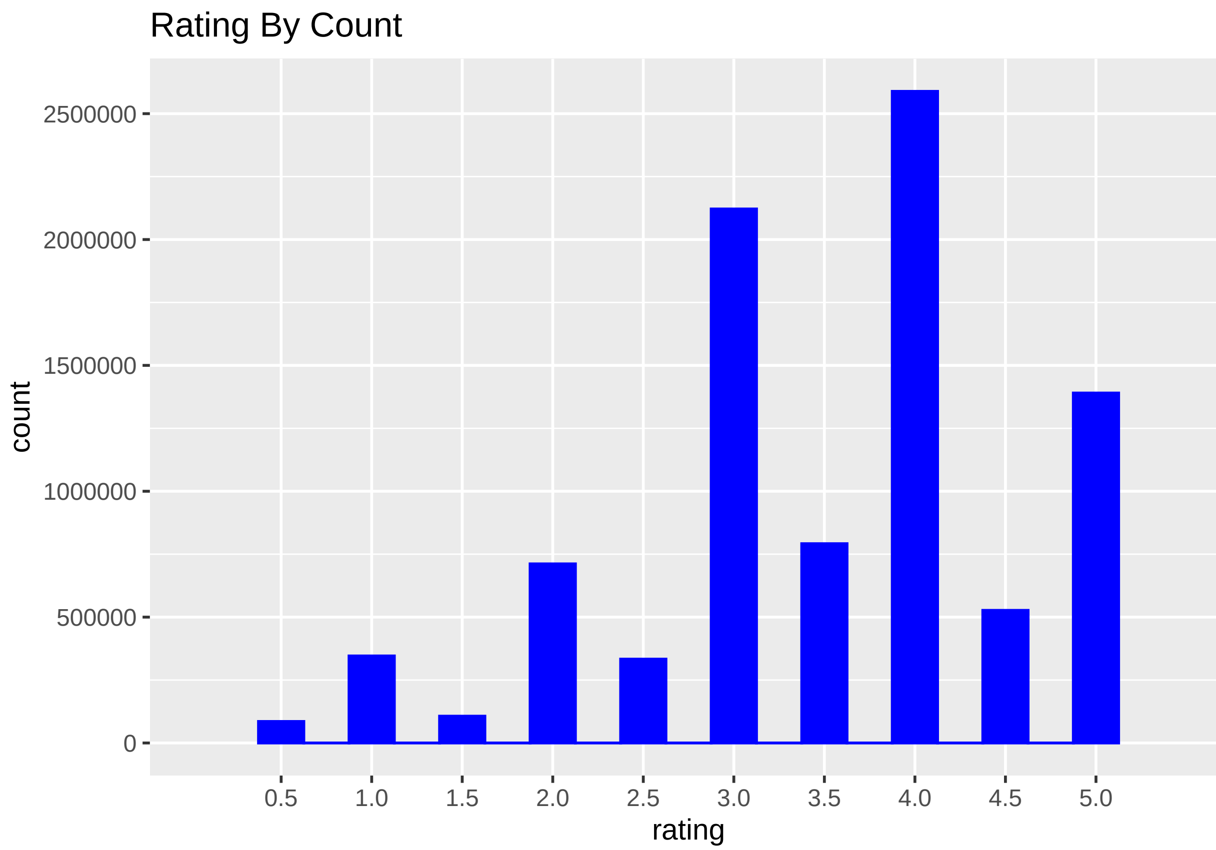
The total of unique movies and users in the edx subset is about 69,878 unique users and about 10,677 different movies:

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

A summary of the subset confirms that there are no missing values.

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

User Rating preference shown below. Half rating are fewer than whole star ratings. 4 rating being highest and




0.5 being lowest

# ANALYSIS/METHODS

We will use various methods to improve result step by step. As mentioned above will compute RMSE for accuracy. RMSE formula as defined as

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

where N is the observation of user/movie and sum of all combination

RMSE is the standard deviation of the residuals (prediction errors) when predcting movie rating. It is always non-negative, and a value of 0 (almost never achieved in practice) would indicate a perfect fit to the data. In general, a lower RMSD is better than a higher one.

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

### Method 1: Average movie rating

This method uses simple approach where it averages across every user and every movie of our predicted ratings. Represent by formula

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

where $Y_{u,i}$ is the predicted rating of user $u$ and movie $i$ and $\mu$ is the average rating across training data (edx).

Calculate mean/average

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

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

RMSE #Calulate RMSE on test data final_holdout_test

```
avg_rmse<-RMSE(final_holdout_test$rating, mu)
##Print RMSE
cat('RMSE for Average Rating is', avg_rmse)
```

```
## RMSE for Average Rating is 1.061202
```

### Method 2: Rating by including Movie Bias

Movie Bias is when movies gets extreme rateing due to like and dislike. Therefore taking this into considration and to minimise the extreme rating effect we added movie bias (b as bias) to our previous method. Formula which represent this is

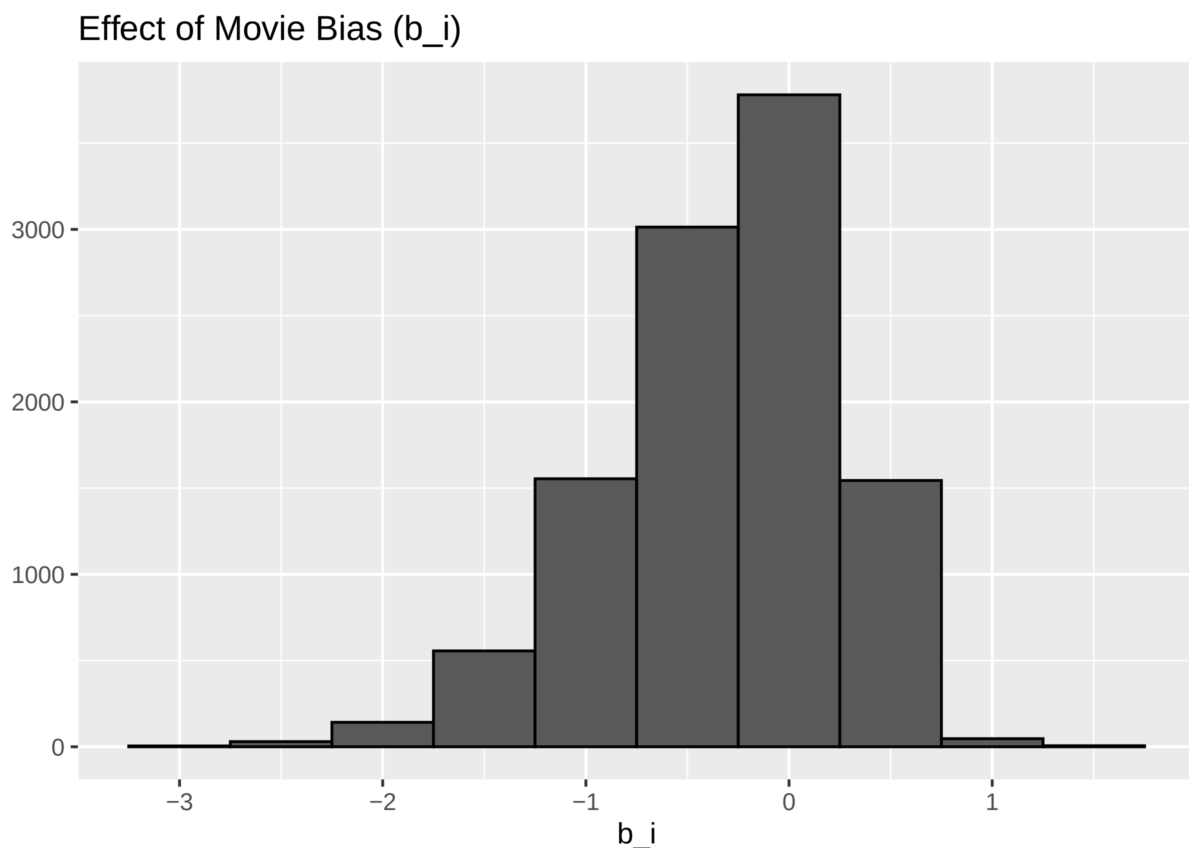
$$Y_{u,i} = \mu + b_i + \epsilon_{u,i}$$

where $b_i$ as bias (b_i) for each movie $\mu$ total mean of all movies

Histrogram showing negative effect of all movie bias

```
# add movie bias b_i
b_i <- edx %>%
  group_by(movieId) %>%
  summarize(b_i = mean(rating - mu))

#draw histogram of training data (edx)
b_i %>% qplot(b_i, geom ="histogram", bins = 10, data = ., color = I("black"),
              main = "Effect of Movie Bias (b_i)")
```




As one can see there is some biasness or effect.

Predict improvement by adding movie bias.

```
# predict rating considering movie bias on test data final_holdout_test
prediction <- final_holdout_test %>%
  left_join(b_i, by='movieId') %>%
  mutate(pred = mu + b_i) %>%
  pull(pred)

# calculate RMSE
movie_rmse <-RMSE(final_holdout_test$rating, prediction)
##Print RMSE
cat('RMSE for Rating that include Movie Bias is', movie_rmse)
```

```
## RMSE for Rating that include Movie Bias is 0.9439087
```

### Method 3: Rating now including User Bias

User bias is when user give extreme rating basesd on their liking and disliking. This method improve further by adding user bias to previous method. Therefore taking this considration and to minimise the extreme user effect we added User Bias to the formula which represent as

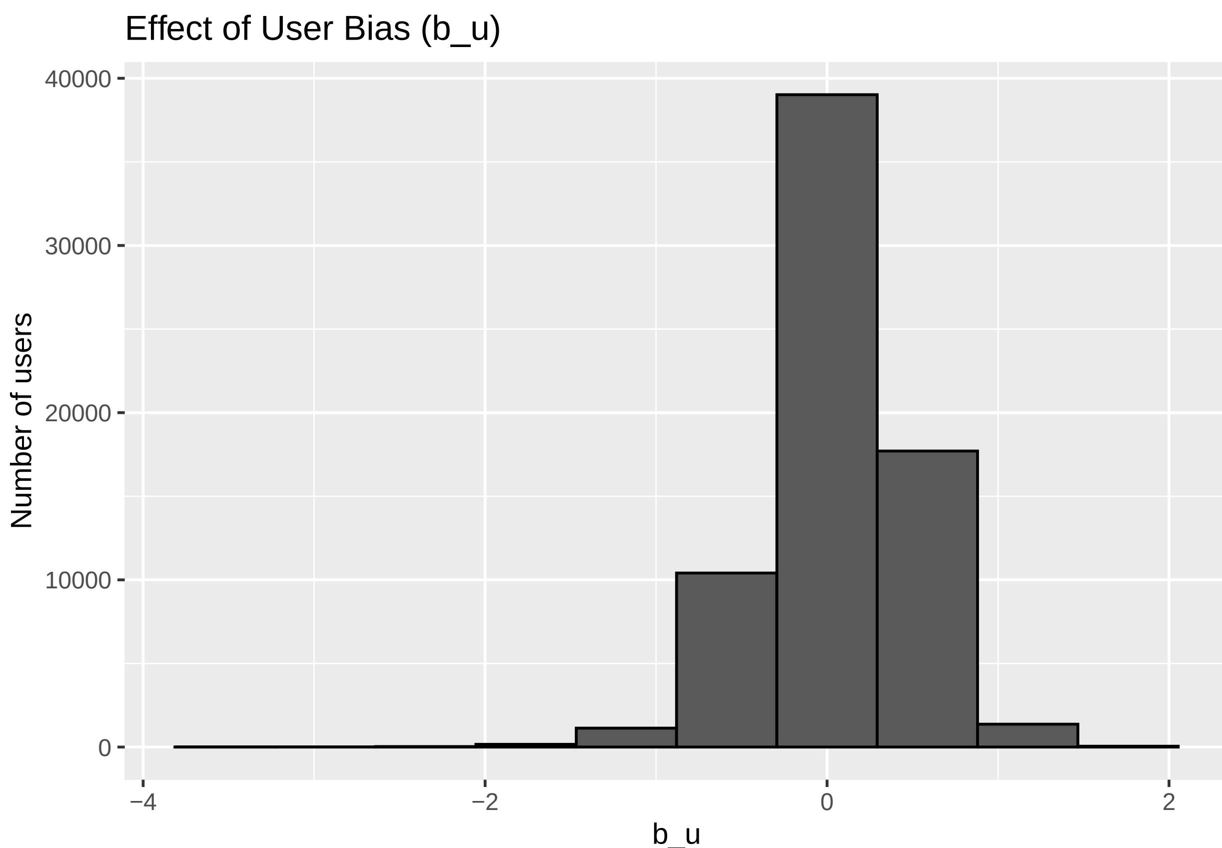
$$Y_{u,i} = \mu + b_i + b_u + \epsilon_{u,i}$$

where $b_u$ as user bias for each movies, $b_i$ as bias (b_i) for each movie $\mu$ total mean of all movies

Histrogram showing negative effect of user bias

```
# add movie bias b_i
b_u <- edx %>%
  left_join(b_i, by='movieId') %>%
  group_by(userId) %>%
  summarize(b_u = mean(rating - mu - b_i))

#draw histogram of training data (edx)
 b_u %>% qplot(b_u, geom ="histogram", bins = 10, data = ., color = I("black"),
ylab = "Number of users", main = "Effect of User Bias (b_u)")
```




As one can see there is some biasness or effect.

Predict improvements by adding user bias.

```
# predict rating considering user bias on test data final_holdout_test
prediction <- final_holdout_test %>%
  left_join(b_i, by='movieId') %>%
  left_join(b_u, by='userId') %>%
  mutate(pred = mu + b_i + b_u) %>%
  pull(pred)

# calculate RMSE
user_rmse <-RMSE(final_holdout_test$rating, prediction)
##Print RMSE
cat('RMSE for Rating that include Movie and User Bias is', user_rmse)
```

```
## RMSE for Rating that include Movie and User Bias is 0.8653488
```

### Method 4: Regularization

As we now apply regularization to reduces incorrect estimates or large errors in our predictions that come from small sizes. Here we uses regularization on movie bais $b_i$ to reduce the large abnormility in movie ratings. Same for $b_u$ to reduce abnormility in rating given by users.

Regularization has the same goal as confidence intervals except you are able to predict a single number instead of an interval. Formula for this model represent

$$\frac{1}{N}\sum_{u,i}(Y_{u,i} - \mu - b_i - b_u)^2 + \lambda(\sum_i b_i^2 + \sum_u b_u^2)$$

where first part is our previous least squares equation and the last part $\lambda(\sum_i b_i^2 + \sum_u b_u^2)$ is the penalty with large bias terms. To Minimize the biases we use a $\lambda$ as goal to our model shown above.

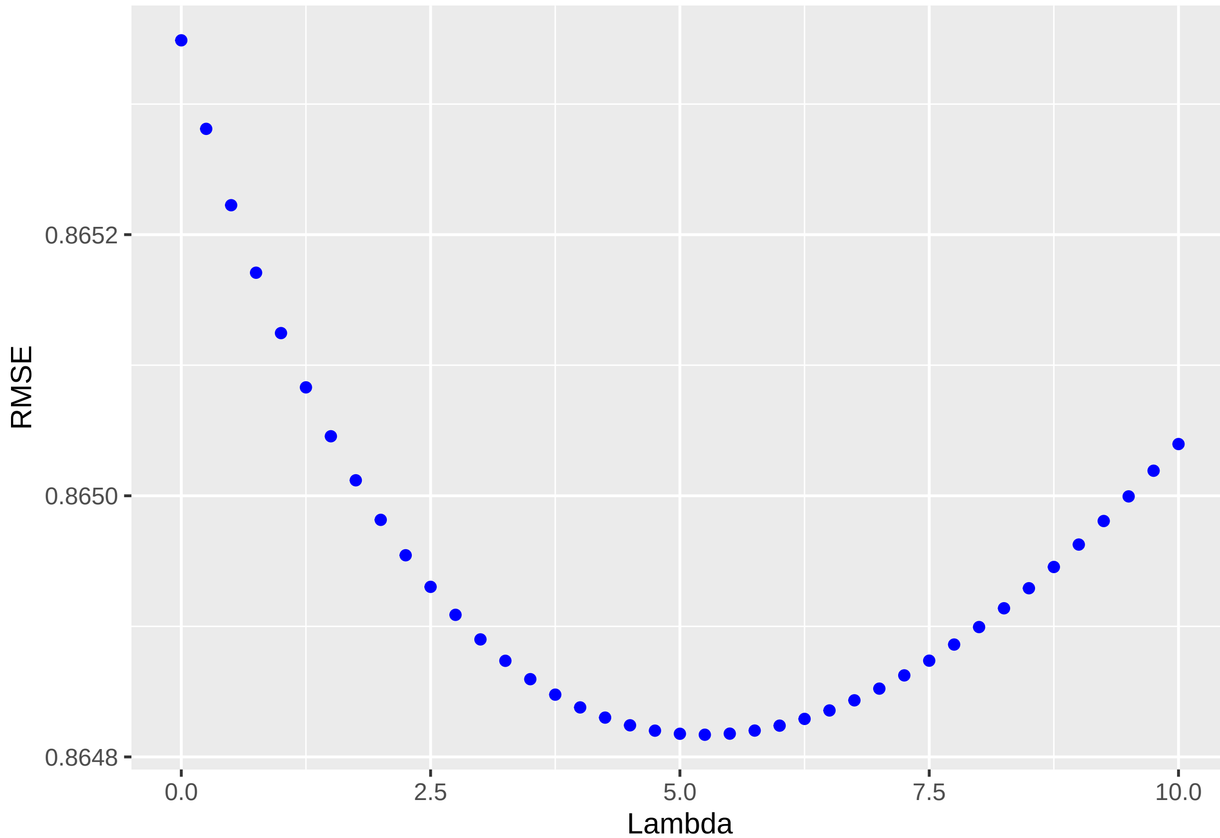
We will use 2 steps here

1. First get the series RMSE from the Lambda sequence seq(from=0, to=10, by=0.25)
2. Get the best lambda (minimum RMSE) from the generated RMSE series and apply it to the final method

We test `lamda <- seq(from=0, to=10, by=0.25)` and plot the results below:

Plot shows RMSE vs. Lambda

```
qplot(lambdas, rmse_series, colour = I("blue"),xlab = "Lambda", ylab="RMSE")
```




Get best Lambda $\lambda$ is

```
best_lambda<-lambdas[which.min(rmse_series)]
best_lambda
```

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

**Method 4.1: Applying the best Lambda to our final method**

```
# Use the best lambda on movie bias (b_i) on training set (edx)
b_i <- edx %>%
  group_by(movieId) %>%
  summarize(b_i = sum(rating - mu)/(n()+best_lambda))

# Use the best lambda on user bias (b_u) on training set (edx)
b_u <- edx %>%
  left_join(b_i, by="movieId") %>%
  group_by(userId) %>%
  summarize(b_u = sum(rating - b_i - mu)/(n()+best_lambda))

# predict using best lambda applied above for regularization on test data final_holdout_test
prediction <- final_holdout_test %>%
  left_join(b_i, by = "movieId") %>%
  left_join(b_u, by = "userId") %>%
```

```
  mutate(pred = mu + b_i + b_u) %>%
  pull(pred)

# output RMSE of our final model
# calculate RMSE
regularization_rmse <-RMSE(final_holdout_test$rating, prediction)
##Print RMSE
cat('RMSE for Rating that include best Lambda regularization in Movie and User Bias is', regularization_
```

```
## RMSE for Rating that include best Lambda regularization in Movie and User Bias is 0.864817
```

# RESULT

Summary Table

Here is the summary result of various method we implemented and improved by considering bias and regularization. Below table shows the RMSE improvent with each methods.

```
#build the column values
c1<-c("Average",
      "Effect included Movie Bias",
      "Effect included Movie and User Bias",
      "Effect included regulerization on Movie and User Bias")
c2<-c(avg_rmse,
      movie_rmse,
      user_rmse,
      regularization_rmse )
#add it to dataframe
df <- data.frame(c1,c2)

#name the heading
names(df) <- c('Method Description', 'RMSE')

#print the result data frame
head(df)
```

```
##                                      Method Description      RMSE
## 1                                               Average 1.0612018
## 2                            Effect included Movie Bias 0.9439087
## 3                   Effect included Movie and User Bias 0.8653488
## 4 Effect included regulerization on Movie and User Bias 0.8648170
```

As we can see the regularized model including the effect of user and movie is has lowest RMSE (0.8648170) value which is lowers than the the initial evaluation criteria (0.8775) and is hence the optimal model use for the present project.

# CONCLUSION

We have built the efficient machine learning algorithm for predecting movie rating for MovieLens Dataset. As we added bias and employ regularizating, our result has improved. We can further improve the result by adding more attribues effect like gener, year etc to our machine language modal. But due to hardware constrain we have used only user and movie effect.