

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

Mano Krishnan

March 09, 2020

Contents

1	Introduction	3
1.0.1	Dataset and Data Loading	3
1.0.2	Libraries	3
1.0.3	Aim & Objectives	4
2	Methodology & Analysis	4
2.1	Data Pre-processing	4
2.1.1	Evaluation of Predicted Ratings using RMSE	4
2.1.2	Split Raw Data: Train and Test Sets	4
2.1.3	Modifying the Year	5
2.2	Data Visualization and Data Exploration	5
2.2.1	General Data Information	5
2.2.2	Distribution of Ratings	6
2.2.3	Ratings per movie	7
2.2.4	Number of ratings by Number of Users	8
2.2.5	Mean movie ratings by users	10
2.3	Data Analysis and modelling	10
2.3.1	Sample estimate- mean	11
2.3.2	Movie Effect - Penalty	11
2.3.3	User Effect - Penalty	11
2.3.4	Naive Model : just the mean	12
2.3.5	Movie Effect Model	13
2.3.6	Movie and User Effect Model	13
2.3.7	Regularized Movie and User Effect Model	14
2.3.8	Regularized Movie, User and year Effect Model	16

3	Results	19
4	Conclusion	19
5	Appendix - Enviroment	20

1 Introduction

The main objective here is to produce product recommendations of an analytical system by applying statistical techniques. We are taking 'movielens' database to predict ratings of the movies.

The 10M version of the dataset is available in the grouplens website. Based on the different statistical models, we will build a rating predictor system.

1.0.1 Dataset and Data Loading

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

Data Loading

```
tinytex::install_tinytex()
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")

# Kindly download and keep the files movies.dat and ratings.dat in the project folder - ml-10M100K.

library(readr)
movies <- read_delim("ml-10M100K/movies.dat",
                    "::", escape_double = FALSE, col_names = FALSE,
                    trim_ws = TRUE)
View(movies)
ratings <- read_delim("ml-10M100K/ratings.dat",
                    "::", escape_double = FALSE, col_names = FALSE,
                    trim_ws = TRUE)

movies <- movies %>% select(-X2,-X4)
colnames(movies) <- c("movieId", "title", "genres")

ratings <- ratings %>% select(-X2,-X4,-X6)
colnames(ratings) <- c("userId", "movieId", "rating", "timestamp")

movies <- as.data.frame(movies) %>% mutate(movieId = as.numeric(movieId)[movieId],
                                          title = as.character(title),
                                          genres = as.character(genres))

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

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

1.0.2 Libraries

The following libraries were used in this report:

```
library(ggplot2)
library(lubridate)
library(caret)
library(tidyverse)
```

1.0.3 Aim & Objectives

The provided dataset is divided into training set and validation set. We are training the first set with the machine learning algorithms and to predict movie ratings in the validation set.

Data visualization and data exploration is used to find the interesting trends and the factors affecting the users' ratings. We are creating four models based on their resulting RMSE and finalizing the optimal model to predict the movie ratings.

2 Methodology & Analysis

2.1 Data Pre-processing

2.1.1 Evaluation of Predicted Ratings using RMSE

The root-mean-square deviation (RMSD) or root-mean-square error (RMSE) is a frequently used measure of the differences between values (sample or population values) predicted by a model or an estimator and the values observed. Here, we are using the RMSE value to evaluate each model.

The function that computes the RMSE for vectors of ratings and their corresponding predictors will be the following:

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

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

2.1.2 Split Raw Data: Train and Test Sets

We can partition the movielens dataset into 2 sets. One set is used for building the algorithm and the second set are used for the validation of the model. The 10% of the movielens data represents the validation set.

```
set.seed(1)
test_index <- createDataPartition(y = movielens$rating, times = 1, p = 0.1, list = FALSE)
edx <- movielens[-test_index,]
temp <- movielens[test_index,]
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)
```

```
## Joining, by = c("userId", "movieId", "rating", "timestamp", "title", "genres")
```

```
edx <- rbind(edx, removed)
validation_CM <- validation
validation <- validation %>% select(-rating)
```

2.1.3 Modifying the Year

The title column is merged with name of the movie and the year of release. So, we are splitting the title column into name and the year column. So that we can find the dependencies between years of release and rating.

```
# Modify the year as a column in the edx & validation datasets
edx <- edx%>%separate(title,c("name", "year"), "\\s*\\((?=\\d+\\))$|\\)$")
```

```
## Warning: Expected 2 pieces. Additional pieces discarded in 8027054 rows [1, 2,
## 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 22, ...].
```

```
## Warning: Expected 2 pieces. Missing pieces filled with `NA` in 391681 rows [20,
## 21, 39, 47, 129, 184, 189, 255, 282, 329, 338, 348, 355, 377, 381, 449, 485,
## 489, 490, 499, ...].
```

```
validation <- validation%>%separate(title,c("name", "year"), "\\s*\\((?=\\d+\\))$|\\)$")
```

```
## Warning: Expected 2 pieces. Additional pieces discarded in 891667 rows [1, 2, 4,
## 5, 6, 7, 9, 10, 11, 12, 13, 14, 15, 17, 18, 19, 21, 22, 23, 24, ...].
```

```
## Warning: Expected 2 pieces. Missing pieces filled with `NA` in 43471 rows [3, 8,
## 16, 20, 34, 65, 106, 127, 147, 165, 190, 194, 197, 204, 240, 252, 277, 283, 292,
## 301, ...].
```

2.2 Data Visualization and Data Exploration

2.2.1 General Data Information

```
# The 1st rows of the edx are presented below:
head(edx)
```

```
##   userId movieId rating timestamp          name year
## 1      1      122      5 838985046 Boys of St. Vincent, The 1992
## 2      1      185      5 838983525          Nadjia 1994
## 3      1      231      5 838983392    Death and the Maiden 1994
## 4      1      292      5 838983421    Once Were Warriors 1994
## 5      1      316      5 838983392 Secret of Roan Inish, The 1994
## 6      1      329      5 838983392    To Live (Huozhe) 1994
##
##              genres
## 1              Drama
## 2              Drama
## 3      Drama|Thriller
## 4      Crime|Drama
## 5 Children|Drama|Fantasy|Mystery
## 6              Drama
```

```
# Summary Statistics of edx
summary(edx)
```

```
##      userId      movieId      rating      timestamp
## Min.   :    1   Min.   :    1   Min.   :0.500   Min.   :7.897e+08
## 1st Qu.:18122   1st Qu.:   648   1st Qu.:3.000   1st Qu.:9.468e+08
## Median :35743   Median :  1834   Median :4.000   Median :1.035e+09
## Mean   :35869   Mean   :  4120   Mean   :3.512   Mean   :1.033e+09
## 3rd Qu.:53602   3rd Qu.:  3624   3rd Qu.:4.000   3rd Qu.:1.127e+09
## Max.   :71567   Max.   :65133   Max.   :5.000   Max.   :1.231e+09
##      name      year      genres
## Length:9000061   Length:9000061   Length:9000061
## Class :character Class :character Class :character
## Mode  :character Mode  :character Mode  :character
##
##
##
```

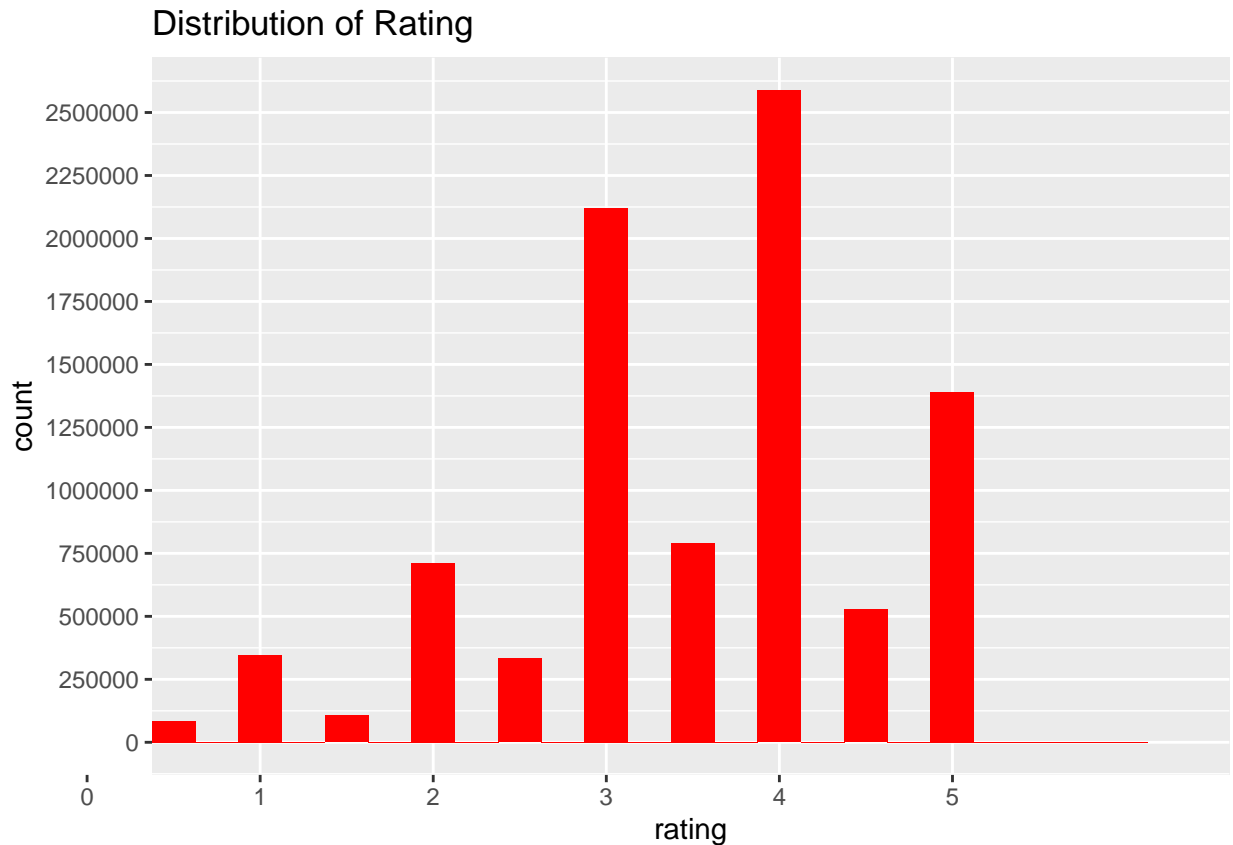
```
# Number of unique movies and users in the edx dataset
edx %>% summarize(n_users = n_distinct(userId), n_movies = n_distinct(movieId))
```

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

2.2.2 Distribution of Ratings

Most common ratings are 3 and 4 compared to other ratings. Then half star ratings are less popular to whole star ratings. The preference of the users is with the higher ratings than lower ratings.

```
edx %>%
  ggplot(aes(rating)) + geom_histogram(binwidth=0.25, fill = "red") +
  scale_x_discrete(limits = c(seq(0,5,1))) +
  scale_y_continuous(breaks = c(seq(0, 3000000, 250000))) +
  ggtitle("Distribution of Rating")
```

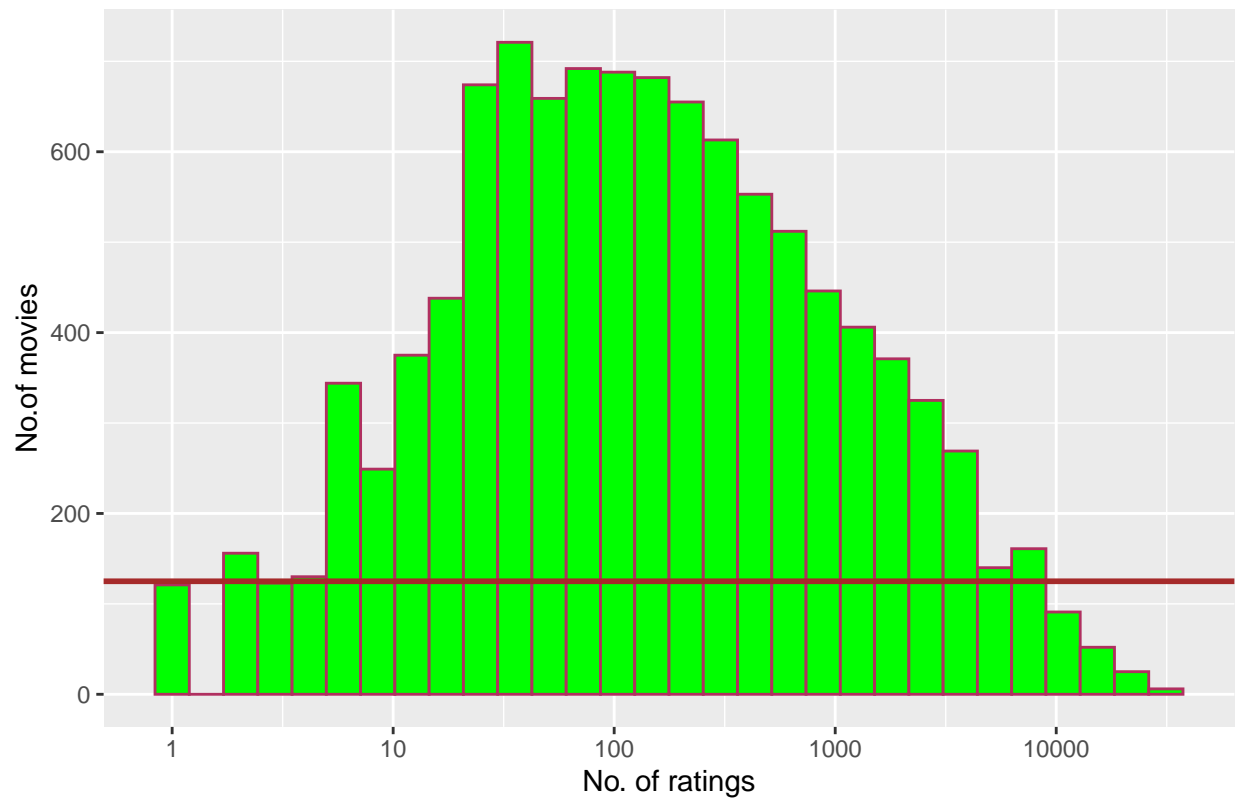


2.2.3 Ratings per movie

The majority of the movies have been rated between 50 and 1000 times. Another interesting fact shows that around 125 movies have been rated only once. These scenarios pushed us to add a penalty term in the model preparation.

```
edx %>%
  count(movieId) %>%
  ggplot(aes(n)) +
  geom_histogram(bins = 30, color = "maroon", fill = "green") +
  scale_x_log10() +
  geom_hline(yintercept=125, color = "brown",size=1) +
  xlab("No. of ratings") +
  ylab("No. of movies") +
  ggtitle("No. ratings per movie")
```

No. ratings per movie

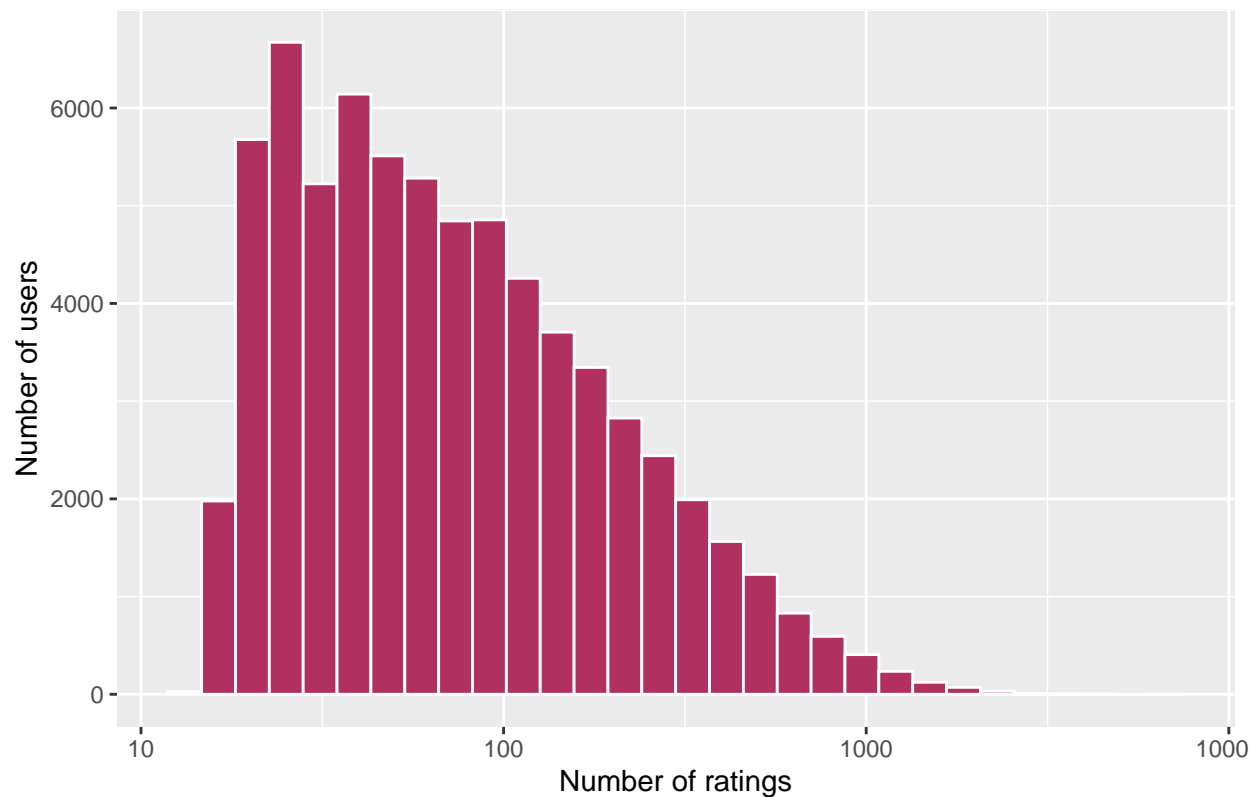


2.2.4 Number of ratings by Number of Users

The majority people have rated below 100 movies and above 30 movies. So a penalty term would be added for this.

```
edx %>%
  count(userId) %>%
  ggplot(aes(n)) +
  geom_histogram(bins = 30, color = "white", fill = "maroon") +
  scale_x_log10() +
  xlab("Number of ratings") +
  ylab("Number of users") +
  ggtitle("Number of ratings given by users")
```


Number of ratings given by users



Total movie ratings per genre

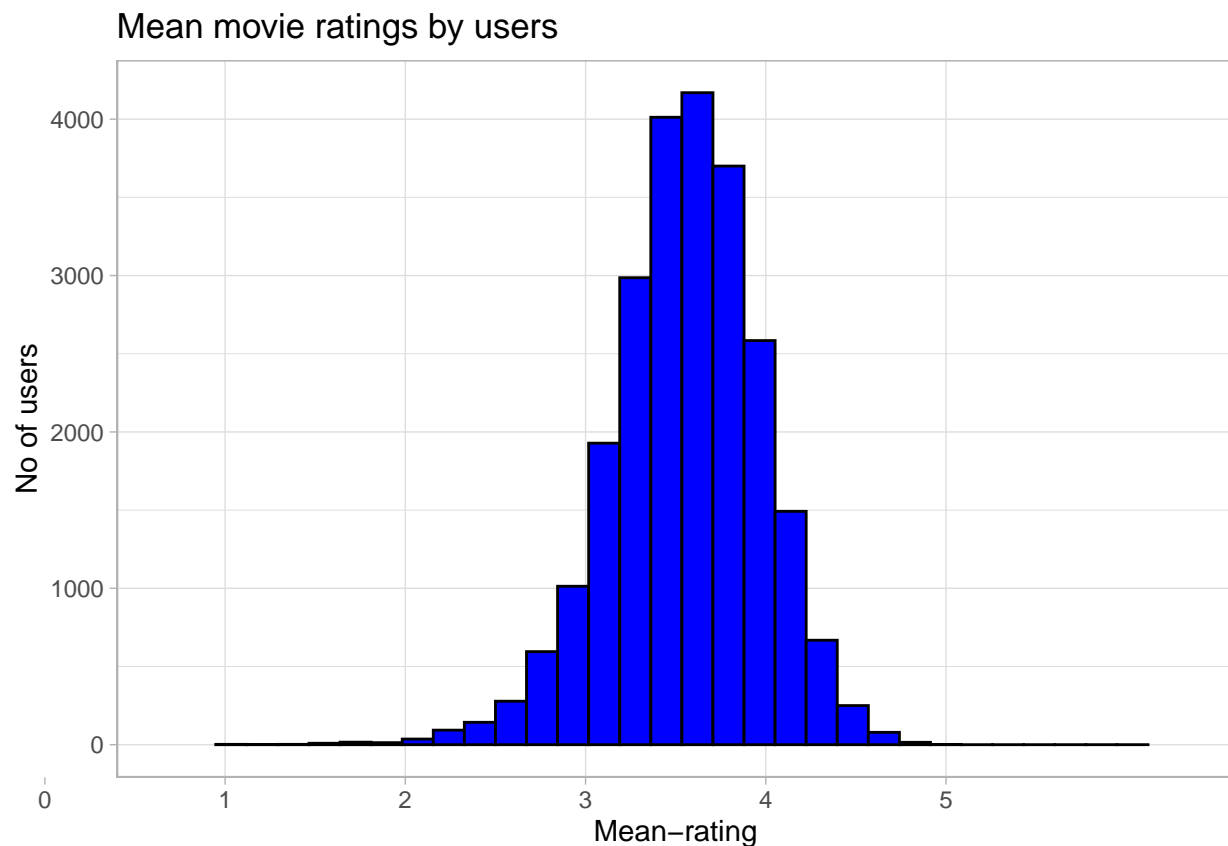
This shows the genre details of the movielens dataset.

```
edx%>%
  group_by(genres) %>%
  summarize(count = n()) %>%
  arrange(desc(count))
```

```
## # A tibble: 629 x 2
##   genres                count
##   <chr>                 <int>
## 1 Drama                1370890
## 2 <NA>                 965190
## 3 Comedy              867118
## 4 Comedy|Drama        397149
## 5 Comedy|Romance      347771
## 6 Drama|Romance       306098
## 7 Comedy|Drama|Romance 208686
## 8 Horror              182946
## 9 Drama|Thriller      182596
## 10 Documentary        167660
## # ... with 619 more rows
```

2.2.5 Mean movie ratings by users

```
edx %>%
group_by(userId) %>%
filter(n() >= 100) %>%
summarize(b = mean(rating)) %>%
ggplot(aes(b)) +
geom_histogram(bins = 30, fill = "blue", color = "black") +
xlab("Mean-rating") +
ylab("No of users") +
ggtitle("Mean movie ratings by users") +
scale_x_discrete(limits = c(seq(0,5,1))) +
theme_light()
```



2.3 Data Analysis and modelling

```
#Initiate RMSE results to compare various models
rmse_results <- data_frame()
```

```
## Warning: `data_frame()` is deprecated, use `tibble()`.
## This warning is displayed once per session.
```

2.3.1 Sample estimate- mean

The initial step is to compute the dataset's mean rating.

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

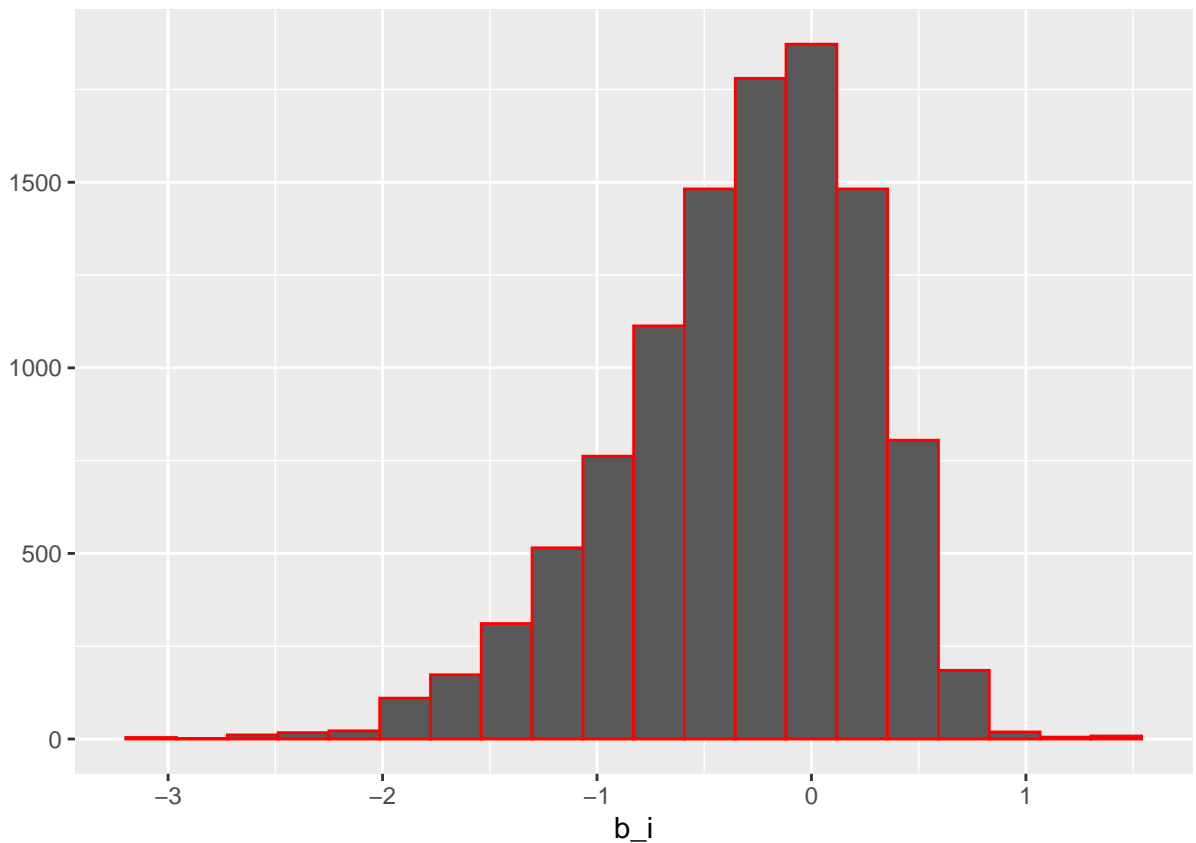
```
## [1] 3.512464
```

2.3.2 Movie Effect - Penalty

Popular movies have higher rating mostly and unpopular movies have low ratings. The histogram is left skewed and it shows that more movies have negative effects.

```
movie_av <- edx %>%
  group_by(movieId) %>%
  summarize(b_i = mean(rating - mu))

movie_av %>% qplot(b_i, geom = "histogram", bins = 20, data = ., color = I("red"))
```



2.3.3 User Effect - Penalty

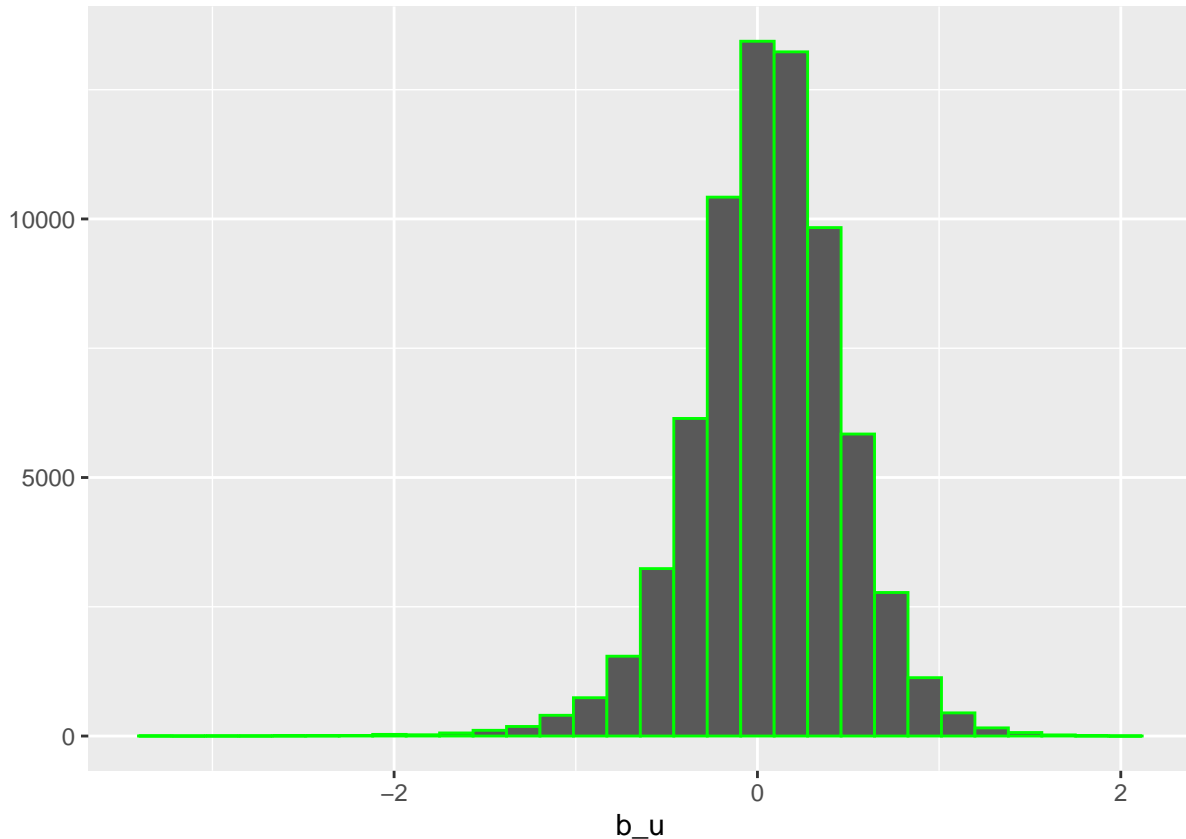
Some users can also affect the ratings either positively (by giving higher ratings) or negatively.

```

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

user_av %>% qplot(b_u, geom="histogram", bins = 30, data = ., color = I("green"))

```



Model Creation ##### Model Validation

The quality of the model will be assessed by the RMSE (the lower the better).

2.3.4 Naive Model : just the mean

This model uses the sample mean represents the initial simplest model. This implies that prediction is the sample average. The resulting RMSE is quite high with this model.

```

# Naive Model -- mean only
naive_rmse <- RMSE(validation_CM$rating,mu)
## Test results based on simple prediction
naive_rmse

```

```
## [1] 1.060651
```

```

## Check results
rmse_results <- data_frame(method = "Using mean only", RMSE = naive_rmse)
rmse_results

```

```
## # A tibble: 1 x 2
##   method      RMSE
##   <chr>      <dbl>
## 1 Using mean only 1.06
```

```
## Save prediction in data frame
```

2.3.5 Movie Effect Model

The RMSE improvisation can be done by adding the movie effect.

```
predrat_movie_norm <- validation %>%
  left_join(movie_av, by='movieId') %>%
  mutate(pred = mu + b_i)

model_1_rmse <- RMSE(validation_CM$rating, predrat_movie_norm$pred)

rmse_results <- bind_rows(rmse_results,
  data_frame(method="Movie Effect Model",
    RMSE = model_1_rmse ))

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

method	RMSE
Using mean only	1.0606506
Movie Effect Model	0.9437046

```
rmse_results
```

```
## # A tibble: 2 x 2
##   method      RMSE
##   <chr>      <dbl>
## 1 Using mean only 1.06
## 2 Movie Effect Model 0.944
```

2.3.6 Movie and User Effect Model

Next improvisation is achieved by adding the user effect.

```
# Use test set, join movie averages & user averages
# Prediction equals the mean with user effect b_u & movie effect b_i
predrat_user_norm <- validation %>%
  left_join(movie_av, by='movieId') %>%
  left_join(user_av, by='userId') %>%
  mutate(pred = mu + b_i + b_u)

# test and save rmse results

model_2_rmse <- RMSE(validation_CM$rating, predrat_user_norm$pred)
rmse_results <- bind_rows(rmse_results,
```

```
data_frame(method="Movie and User Effect Model",
            RMSE = model_2_rmse ))
rmse_results %>% knitr::kable()
```

method	RMSE
Using mean only	1.0606506
Movie Effect Model	0.9437046
Movie and User Effect Model	0.8655329

```
rmse_results
```

```
## # A tibble: 3 x 2
##   method          RMSE
##   <chr>          <dbl>
## 1 Using mean only    1.06
## 2 Movie Effect Model 0.944
## 3 Movie and User Effect Model 0.866
```

2.3.7 Regularized Movie and User Effect Model

This model adds the concept of regularization to account for the effect of low ratings' numbers for movies and users. This regularization used to reduce the effect of overfitting.

```
# lambda is a tuning parameter
# Use cross-validation to choose it.
lambdas <- seq(0, 10, 0.25)
# For each lambda, find b_i & b_u, followed by rating prediction & testing
# note: the below code could take some time
rmses <- sapply(lambdas, function(l){

  mu <- mean(edx$rating)

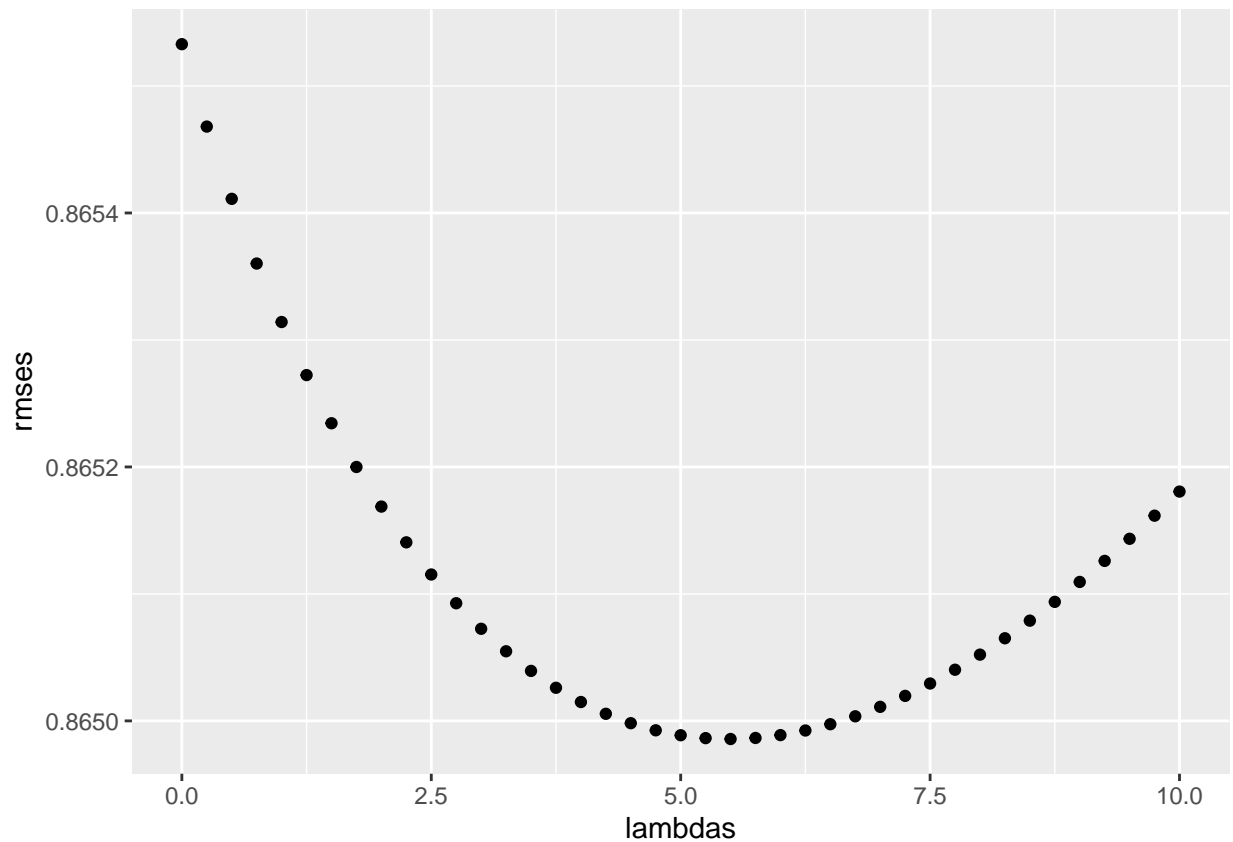
  b_i <- edx %>%
    group_by(movieId) %>%
    summarize(b_i = sum(rating - mu)/(n()+1))

  b_u <- edx %>%
    left_join(b_i, by="movieId") %>%
    group_by(userId) %>%
    summarize(b_u = sum(rating - b_i - mu)/(n()+1))

  predrat <-
    validation %>%
    left_join(b_i, by = "movieId") %>%
    left_join(b_u, by = "userId") %>%
    mutate(pred = mu + b_i + b_u) %>%
    .$pred
  return(RMSE(predrat, validation_CM$rating))

})
```

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



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

# Compute regularized estimates of b_i using lambda
movie_av_reg <- edx %>%
  group_by(movieId) %>%
  summarize(b_i = sum(rating - mu)/(n()+lambda), n_i = n())

# Compute regularized estimates of b_u using lambda

user_av_reg <- edx %>%
  left_join(movie_av_reg, by='movieId') %>%
  group_by(userId) %>%
  summarize(b_u = sum(rating - mu - b_i)/(n()+lambda), n_u = n())

# Predict ratings

predrat_reg <- validation %>%
  left_join(movie_av_reg, by='movieId') %>%
  left_join(user_av_reg, by='userId') %>%
  mutate(pred = mu + b_i + b_u) %>%
  .$pred
```

```
# Test and save results
```

```
model_3_rmse <- RMSE(validation_CM$rating, predrat_reg)
rmse_results <- bind_rows(rmse_results,
                          data_frame(method="Regularized Movie and User Effect Model",
                                     RMSE = model_3_rmse ))
rmse_results %>% knitr::kable()
```

method	RMSE
Using mean only	1.0606506
Movie Effect Model	0.9437046
Movie and User Effect Model	0.8655329
Regularized Movie and User Effect Model	0.8649857

```
rmse_results
```

```
## # A tibble: 4 x 2
##   method          RMSE
##   <chr>          <dbl>
## 1 Using mean only    1.06
## 2 Movie Effect Model 0.944
## 3 Movie and User Effect Model 0.866
## 4 Regularized Movie and User Effect Model 0.865
```

2.3.8 Regularized Movie, User and year Effect Model

This model adds the concept of regularization to account for the effect of low ratings' numbers for year in-addition to the movie and user. This regularization used to reduce the effect of overfitting compared to earlier model.

```
# lambda is a tuning parameter
# Use cross-validation to choose it.
lambdas <- seq(0, 10, 0.25)

# For each lambda, find b_i1, b_u1 & b_y1, followed by rating prediction & testing

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

  mu <- mean(edx$rating)

  b_i1 <- edx %>%
    group_by(movieId) %>%
    summarize(b_i1 = sum(rating - mu)/(n()+1))

  b_u1 <- edx %>%
    left_join(b_i1, by="movieId") %>%
    group_by(userId) %>%
    summarize(b_u1 = sum(rating - b_i1 - mu)/(n()+1))
```



```

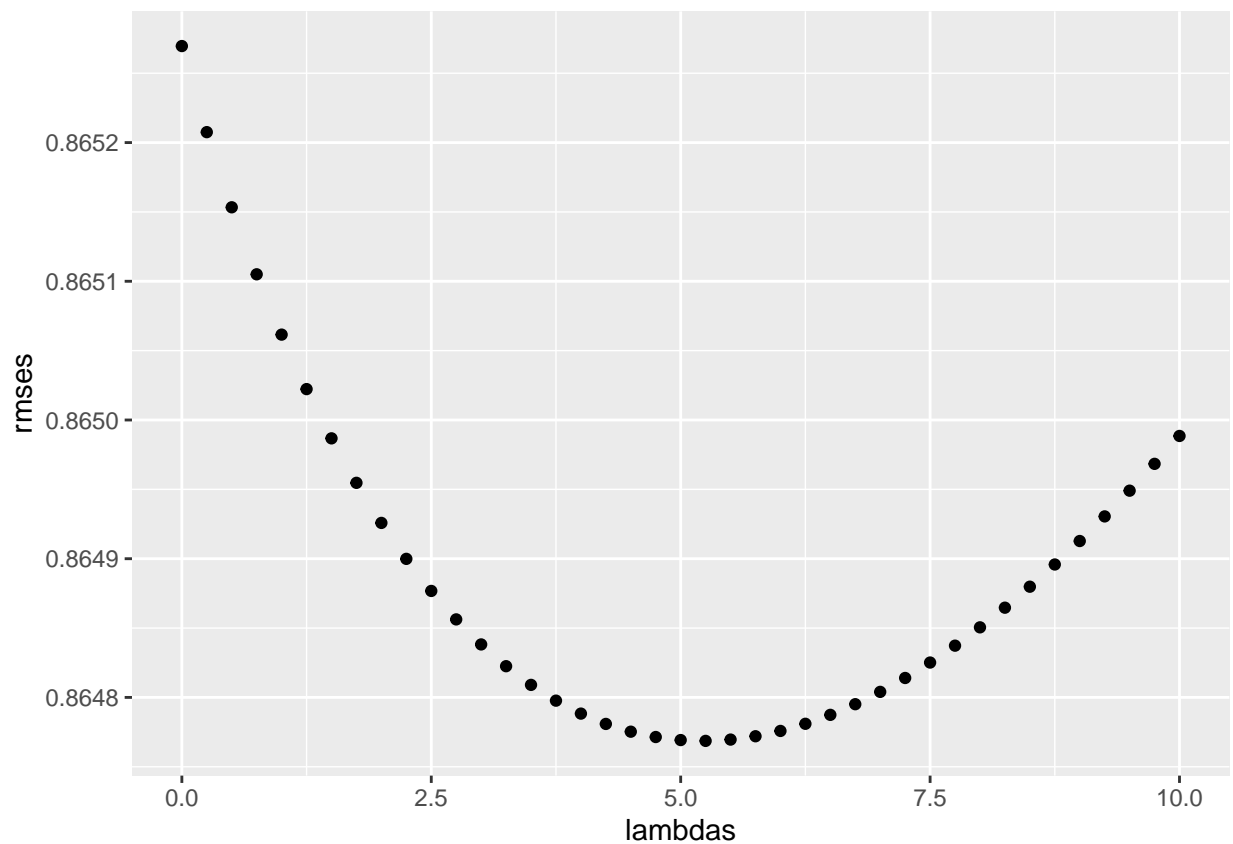
b_y1 <- edx %>%
  left_join(b_i1, by="movieId") %>%
  left_join(b_u1, by="userId") %>%
  group_by(year) %>%
  summarize(b_y1 = sum(rating - b_u1 - b_i1 - mu)/(n()+1))

predrat1 <-
  validation %>%
  left_join(b_i1, by = "movieId") %>%
  left_join(b_u1, by = "userId") %>%
  left_join(b_y1, by = "year") %>%
  mutate(pred = mu + b_i1 + b_u1 + b_y1) %>%
  pull(pred)

  return(RMSE(predrat1, validation_CM$rating))
})

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

```



```

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

# Compute regularized estimates of b_i1 using lambda
movie_av_reg1 <- edx %>%

```

```

group_by(movieId) %>%
  summarize(b_i1 = sum(rating - mu)/(n()+lambda), n_i1 = n())

# Compute regularized estimates of b_u1 using lambda

user_av_reg1 <- edx %>%
  left_join(movie_av_reg1, by='movieId') %>%
  group_by(userId) %>%
  summarize(b_u1 = sum(rating - mu - b_i1)/(n()+lambda), n_u1 = n())

# Compute regularized estimates of b_y1 using lambda

year_av_reg1 <- edx %>%
  left_join(movie_av_reg1, by='movieId') %>%
  left_join(user_av_reg1, by='userId') %>%
  group_by(year) %>%
  summarize(b_y1 = sum(rating - mu - b_i1 - b_u1)/(n()+lambda), n_y1 = n())

# Predict ratings

predrat_reg1 <- validation %>%
  left_join(movie_av_reg1, by='movieId') %>%
  left_join(user_av_reg1, by='userId') %>%
  left_join(year_av_reg1, by = 'year') %>%
  mutate(pred = mu + b_i1 + b_u1 + b_y1) %>%
  .$pred

# Test and save results

model_4_rmse <- RMSE(validation_CM$rating, predrat_reg1)
rmse_results <- bind_rows(rmse_results,
  data_frame(method="Regularized Movie User and Year Effect Model",
    RMSE = model_4_rmse ))

rmse_results %>% knitr::kable()

```

method	RMSE
Using mean only	1.0606506
Movie Effect Model	0.9437046
Movie and User Effect Model	0.8655329
Regularized Movie and User Effect Model	0.8649857
Regularized Movie User and Year Effect Model	0.8647687

```
rmse_results
```

```

## # A tibble: 5 x 2
##   method          RMSE
##   <chr>          <dbl>
## 1 Using mean only 1.06
## 2 Movie Effect Model 0.944
## 3 Movie and User Effect Model 0.866

```

```
## 4 Regularized Movie and User Effect Model      0.865
## 5 Regularized Movie User and Year Effect Model 0.865
```

3 Results

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

```
rmse_results
```

```
## # A tibble: 5 x 2
##   method          RMSE
##   <chr>          <dbl>
## 1 Using mean only      1.06
## 2 Movie Effect Model    0.944
## 3 Movie and User Effect Model 0.866
## 4 Regularized Movie and User Effect Model 0.865
## 5 Regularized Movie User and Year Effect Model 0.865
```

The primitive model was with the mean only model which gave the rmse as 1.0606. Then movie effect model has shown a very little improvisation. The user effect model in-addition to the movie effect has improvised the rmse to 0.8655. Furthermore, we have introduced the regularized movie and user effect model by interating the lambda value. This model gave the rmse as 0.8649. Finally, we added year effect to the regularized movie and user effect model and we have achieved the desired rmse which is below 0.86490. As per above details, we found the lowest value of RMSE is 0.8647687.

4 Conclusion

We have built a machine learning algorithm with different models to predict movie ratings. The regularised model including the effect of movie, user and year is characterized by the lower RMSE value. Hence, this is the optimal model for this project. This model RMSE value is 0.8647687 which is lower than the evaluation criteria (0.86490). We can also improve the RMSE by adding other effects like genre and age with limitation to the configuration of the regular computers.

5 Appendix - Enviroment

```
print("Operating System:")
```

```
## [1] "Operating System:"
```

```
version
```

```
##  
## platform      x86_64-w64-mingw32  
## arch          x86_64  
## os            mingw32  
## system        x86_64, mingw32  
## status  
## major         3  
## minor         6.3  
## year          2020  
## month         02  
## day           29  
## svn rev       77875  
## language      R  
## version.string R version 3.6.3 (2020-02-29)  
## nickname      Holding the Windsock
```