HarvardX: PH125.9x - Data Science MovieLens Project Report

Aparajithan Rajendran

February 3, 2019

Introduction

This is a capstone project report generated as part of the course 'Data Science' (HarvardX: PH125.9x) at Harvard University in collaboration with edx with the intent of building recommendation engine from the dataset of MovieLens with 10 Million records by using R programming language.

Goal

The idea is to train a machine learning algorithm using the inputs in one subset with 9 Million records to predict movie ratings in the validation set of 1 Million. In this specific project, the goal is to achieve RMSE less than 0.8775 using regularization techniques.

Key Steps

There are four key steps performed in order to build recommendation system that could predict ratings on the validation set.

1. Create edx(training) set, validation(test) set - provided by the instructor
2. Data Cleaning - performed by the student
3. Data Exploration - performed by the student
4. Prediction Modeling - performed by the student
5. Run prediction on validation set - performed by the student

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

# Note: this process could take a couple of minutes

if(!require(tidyverse)) install.packages("tidyverse", repos = "http://cran.us.r-project.org")
```

```
## Loading required package: tidyverse
```

```
## -- Attaching packages --------------------------------------------------------------- tidyverse 1.2.1 --
```

```
## v ggplot2 3.1.0     v purrr   0.2.5
## v tibble  2.0.1     v dplyr   0.7.8
## v tidyr   0.8.2     v stringr 1.3.1
## v readr   1.3.1     v forcats 0.3.0
```

```
## -- Conflicts ------------------------------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```
if(!require(caret)) install.packages("caret", repos = "http://cran.us.r-project.org")
```

```
## Loading required package: caret
```

```
## Loading required package: lattice
```

```
##
## Attaching package: 'caret'
```

```
## The following object is masked from 'package:purrr':
##
##     lift
```

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

dl <- tempfile()
download.file("http://files.grouplens.org/datasets/movielens/ml-10m.zip", dl)

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

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

# Validation set will be 10% of MovieLens data

set.seed(1)
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)
```

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

```
edx <- rbind(edx, removed)
```

Methodology

To achieve the RMSE less than 0.87750, the following methodology is applied.

Data Cleaning

As part of Data Cleaning process the following steps are performed:

- Understanding features

```
#----------Data Cleaning starts
names(edx)
```

```
## [1] "userId"    "movieId"   "rating"    "timestamp" "title"     "genres"
```

- Removing unused data
    - This filters timestamp out.

```
# Since we want to predict ratings irrespective of timestamp, remove timestamp
edx_subset <- edx %>% select(-timestamp)
names(edx_subset)
```

```
## [1] "userId"  "movieId" "rating"  "title"   "genres"
```

- Transforming string into factored numeric values
    - This transforms genres values into factored numbers.

```
# Since genres is type of string and there are only number of genres repeated, we replace the string with numbers
edx_subset <- edx_subset %>% mutate(genres = as.numeric(as.factor(genres)))
```

- Handling collinearity
    - This filters title out.

```
# Since the column title that is unique and has a collinearity only with movieId we treat it as overhead
# Before removing, let's capture them for future reference
movie_titles <- unique(edx_subset %>% select(movieId, title))
edx_subset <- edx_subset %>% select(-title)
```

- Understanding variable importance
    - This explains the importance of the features movieId & userId.

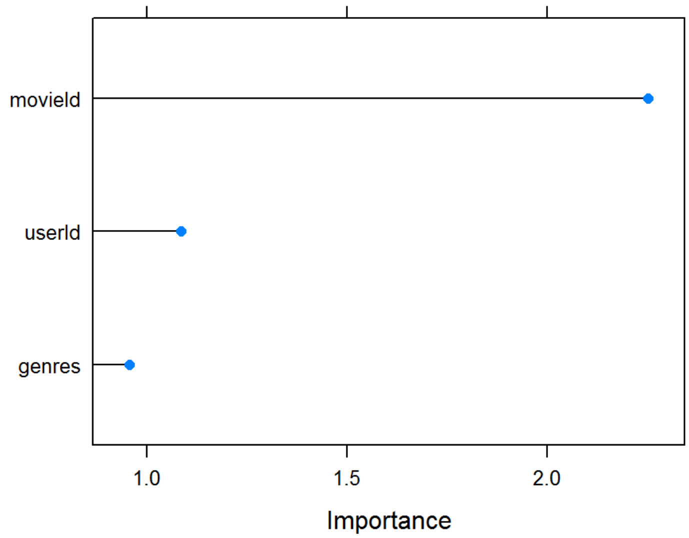
```
# Prepare sampling for 5 & 4.5 ratings
train_n <- 100
train_1 <- edx_subset %>% filter(rating == 5) %>% mutate(rating = 'A')
train_2 <- edx_subset %>% filter(rating == 4.5) %>% mutate(rating = 'B')
set.seed(12345)
train_i_1 <- sample(1:nrow(train_1), train_n, replace = FALSE)
train_i_2 <- sample(1:nrow(train_2), train_n, replace = FALSE)
train <- rbind(train_1[train_i_1, ], train_2[train_i_2, ])

# We have 3 features to predict rating
names(train)
```

```
## [1] "userId"  "movieId" "rating"  "genres"
```

```
# Train model for binomial logistic regression to see how the overall variable importance is between the given features
control <- trainControl(method="repeatedcv", number=10, repeats=3)
model <- train(as.factor(rating)~., data=train, method="glm", preProcess="scale", trControl=control)

# estimate variable importance
importance <- varImp(model, scale=FALSE)
# summarize importance
print(importance)
```

```
## glm variable importance
##
##         Overall
## movieId  2.2524
## userId   1.0851
## genres   0.9545
```

```
# plot importance
plot(importance)
```



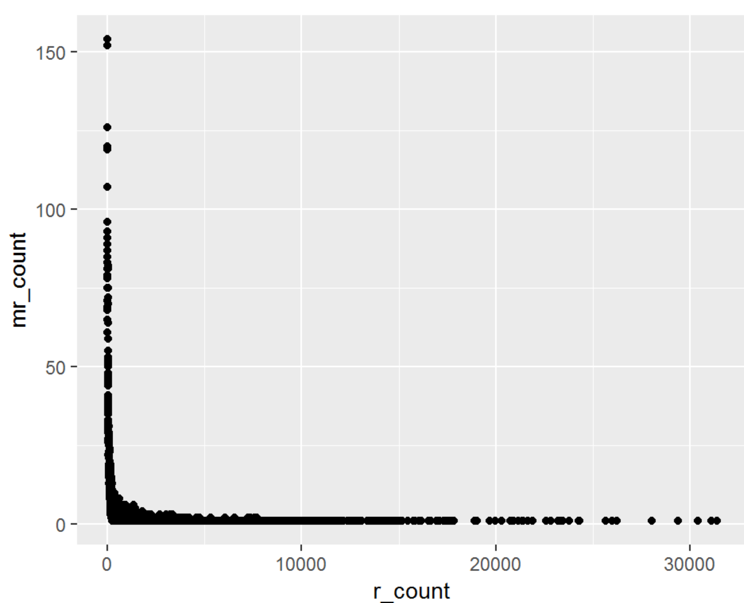

```
# Let's start understanding the variances by each feature - movieId, userId, & genre
#----------Data Cleaning ends
```

Data Exploration

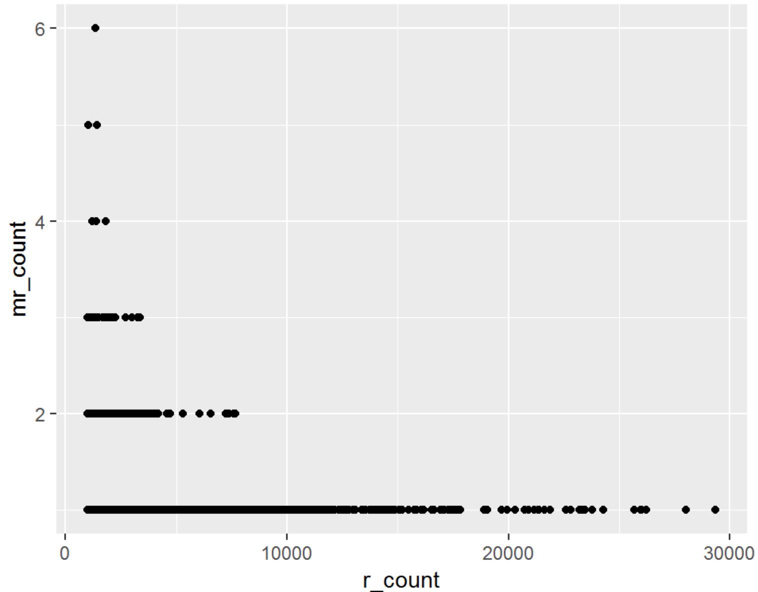
As part of Data Exploration process the following steps are performed:

- Understanding movies-rating
    - Histogram on the summarized movie-rating count explains the variance & distribution of how the movies are rated.

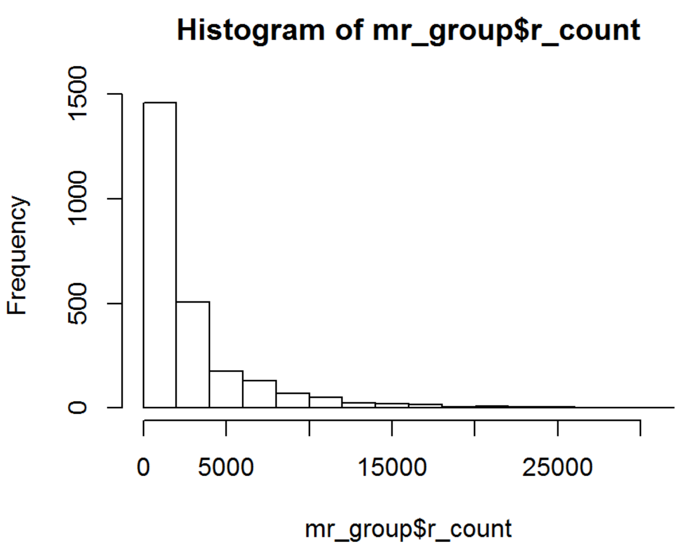
```
#----------Data Exploration starts
#----------Understanding movies-rating distribution
m_group <- edx_subset %>% select(movieId, rating) %>% group_by(movieId) %>% dplyr::summarise(r_count = n())
mr_group <- m_group %>% group_by(r_count) %>% dplyr::summarise(mr_count = n())
qplot(r_count, mr_count, data = mr_group, color = I("black"))
```



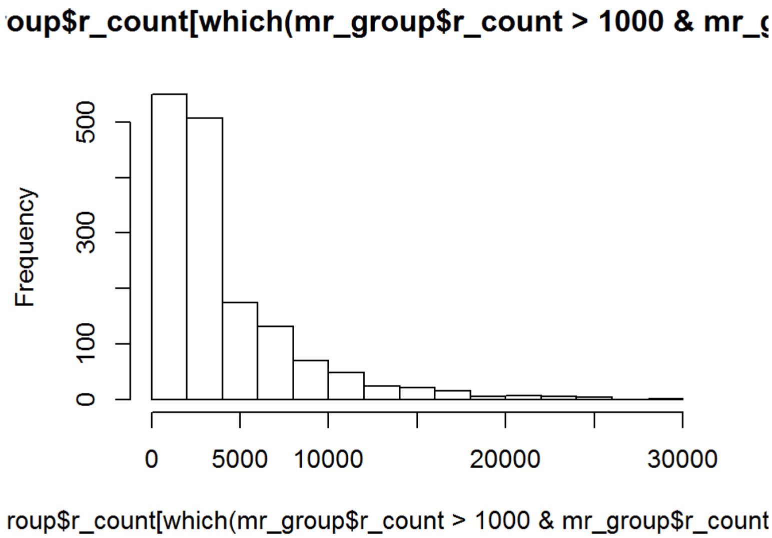

```
qplot(r_count, mr_count, data = mr_group[which(mr_group$r_count > 1000 & mr_group$r_count < 30000),], color = I("black"))
```




```
hist(mr_group$r_count)
```



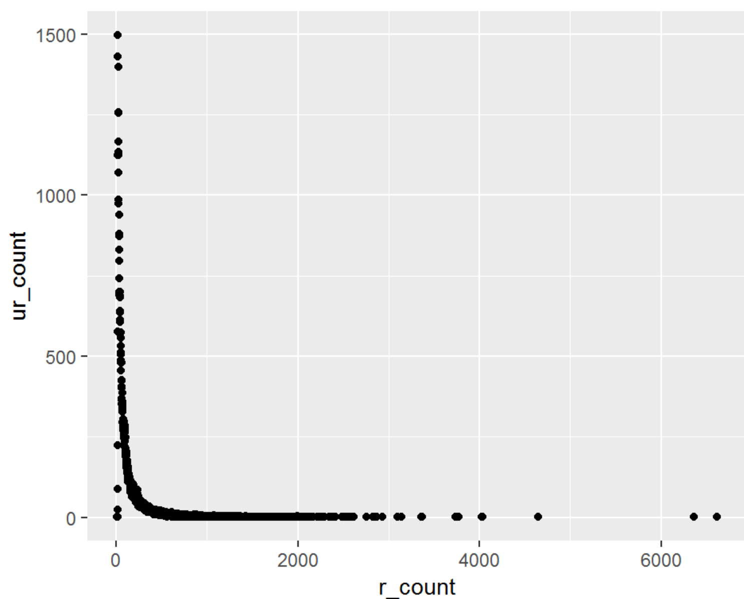

```
# To understand better relative distribution, let's remove the movies which has less than 1000 ratings & greater than 30000
 ratings
hist(mr_group$r_count[which(mr_group$r_count > 1000 & mr_group$r_count < 30000)])
```



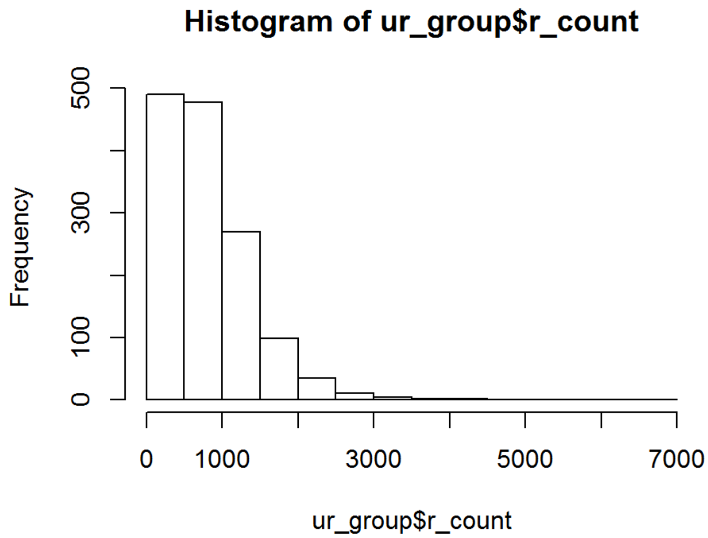

```
# Most of the movies have recieved less 5000 ratings, given the max number of ratings 30000
```

- Understanding Users-rating
    - Histogram on the summarized user-rating count explains the variance & distribution of how the users have rated.

```
#----------Understanding Users-rating distribution
u_group <- edx_subset %>% select(userId, rating) %>% group_by(userId) %>% dplyr::summarise(r_count = n())
ur_group <- u_group %>% group_by(r_count) %>% dplyr::summarise(ur_count = n())
qplot(r_count, ur_count, data = ur_group, color = I("black"))
```



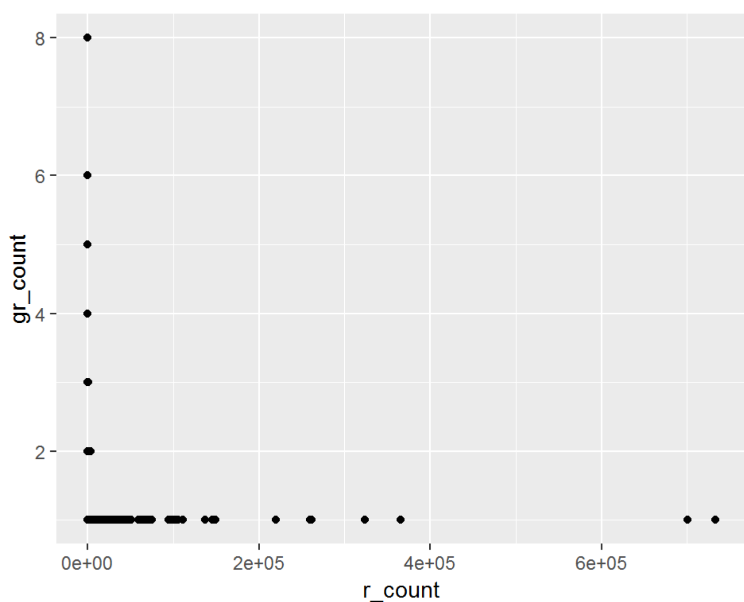

```
hist(ur_group$r_count)
```



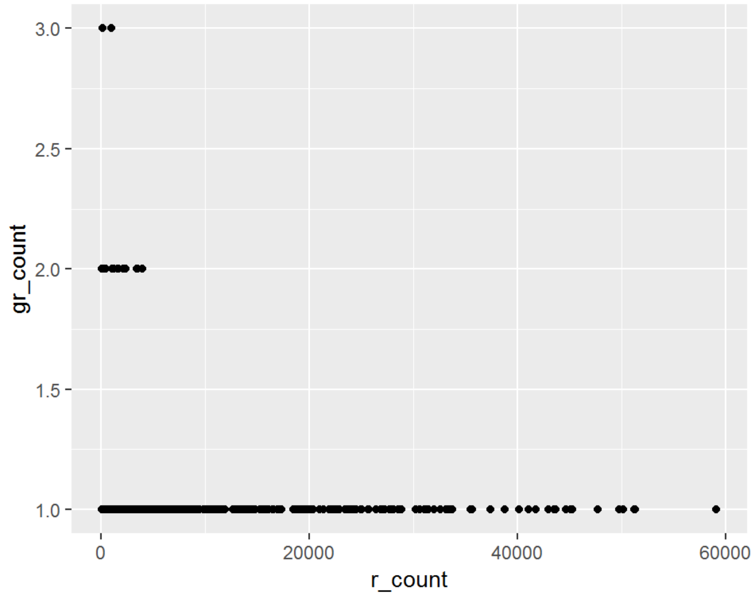

```
# Most of the users have rated less 2000 ratings, given the max number of ratings 4500
```

- Understanding Genres-rating
  - Histogram on the summarized genres-rating count explains the variance & distribution of how the genres are rated.

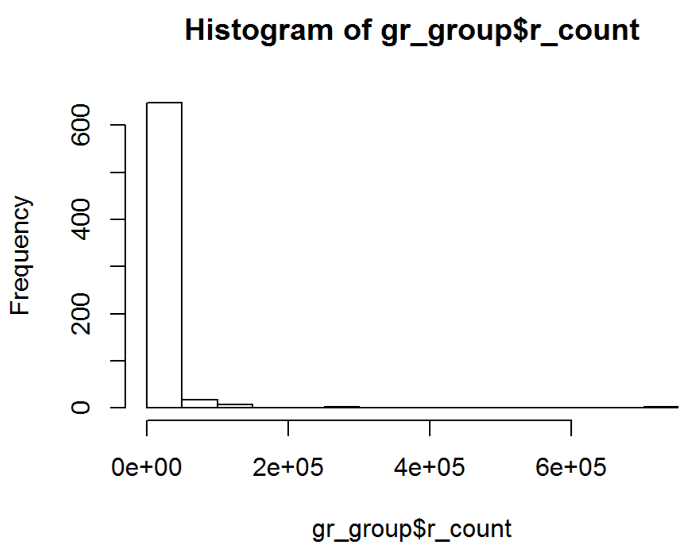
```
#----------Understanding Genres-rating distribution
g_group <- edx_subset %>% select(genres, rating) %>% group_by(genres) %>% dplyr::summarise(r_count = n())
gr_group <- g_group %>% group_by(r_count) %>% dplyr::summarise(gr_count = n())
qplot(r_count, gr_count, data = gr_group, color = I("black"))
```



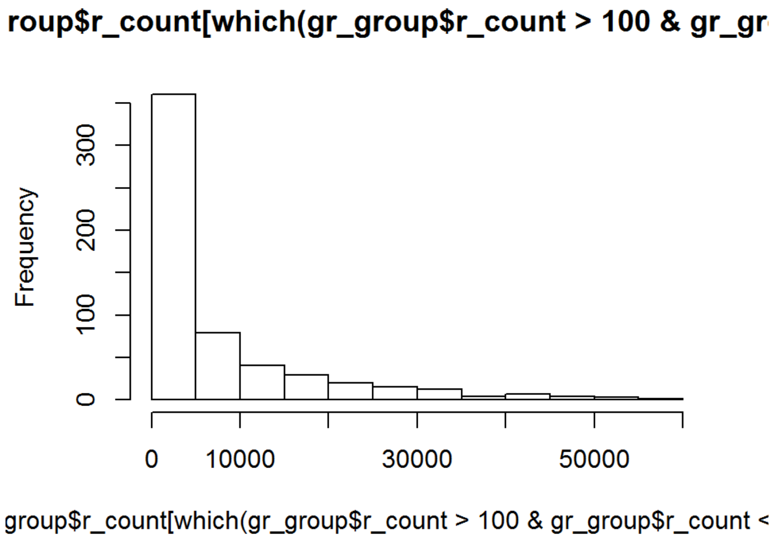

```
qplot(r_count, gr_count, data = gr_group[which(gr_group$r_count > 100 & gr_group$r_count < 60000),], color = I("black"))
```




```
hist(gr_group$r_count)
```




```
# To understand better relative distribution, let's remove the genres which has less than 100 ratings & greater than 60000
 ratings
hist(gr_group$r_count[which(gr_group$r_count > 100 & gr_group$r_count < 60000)])
```




```
# Most of the genres have recieved less 5000 ratings, given the max number of ratings 60000
#----------Data Exploration ends
```

Prediction Modeling

As part of Prediction Modeling process, a step-by-step approach is taken to achieve the RMSE optimization or reduction:

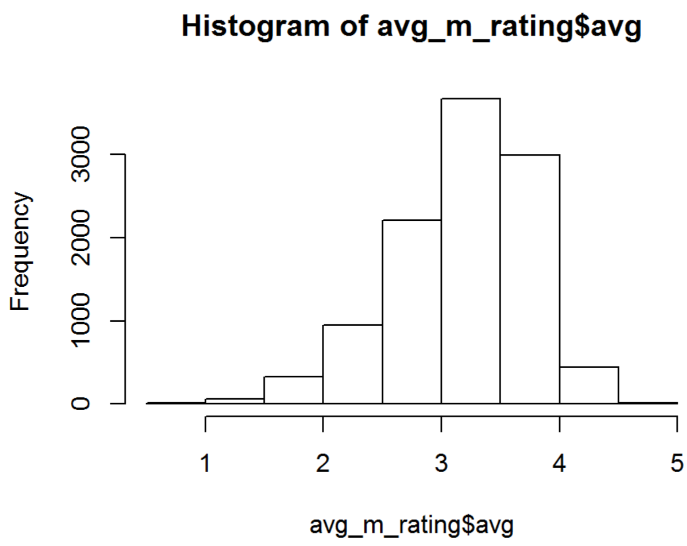
```
#----------Prediction Modeling starts
# Define function as If RMSE > 1, not a good prediction
RMSE <- function(true_ratings, predicted_ratings) {
    sqrt(mean((true_ratings - predicted_ratings) ^ 2))
}

# We compute this average on the training data.
mu_hat <- mean(edx_subset$rating)

# Initialize results frame to store RMSEs as calculated & improvised through out the modeling process
rmse_results <- data.frame()
```

- Model By Overall Average
  - First model to start with overall mean of rating, no bias i.e. without taking any feature effects into account.

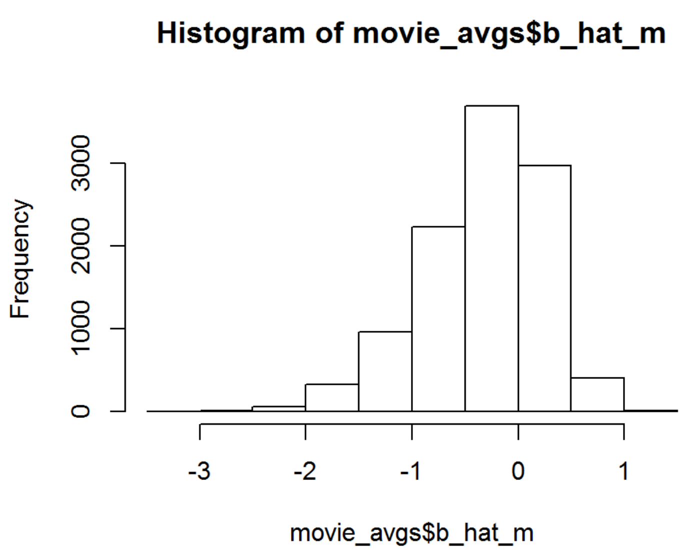
```
#----------1. By Overall Average
# We compute the residual mean squared error (naive_rmse) on the test set data by overall average.
predictions <- mu_hat # This assignment is performed just for readability purposes
rmse_1 <- RMSE(edx_subset$rating, predictions)
rmse_results <- bind_rows(rmse_results, tibble(method = "By overall average", RMSE = round(rmse_1, digits = 3)))
```

- Model By Movie Effects
  - Second model to consider the overall mean of rating along with the bias introduced by the movie effects.

```
#----------2. By Movie Effects
# Each movie is rated differently -> can be seen by finding the average of ratings for each movie
avg_m_rating <- edx_subset %>% select(movieId, rating) %>% group_by(movieId) %>% dplyr::summarise(avg = mean(rating))
hist(avg_m_rating$avg)
```




```
# Most of the average ratings are between 2.5 & 4.0

# Hence, Y_hat u,m = mu_hat + b_hat_m + E u,m -> where b_hat_m (bias) is the average rating bias for the movie m
# b_hat_m = avg(y_hat u,m - mu_hat)
movie_avgs <- edx_subset %>% select(movieId, rating) %>% group_by(movieId) %>% dplyr::summarise(b_hat_m = mean(rating - mu_
hat))

# b_hat_m values vary substantially
hist(movie_avgs$b_hat_m)
```




```
# Relationship btw mu_hat & b_hat_m -> Y_hat u,m = mu_hat + b_hat_m (3.5 overall mean + 1.5 bias for a 5 star rated movie)
predictions <- mu_hat + (edx_subset %>% left_join(movie_avgs, by = "movieId") %>% .$b_hat_m)
rmse_2 <- RMSE(edx_subset$rating, predictions)
rmse_results <- bind_rows(rmse_results, tibble(method = "By movie based average", RMSE = round(rmse_2, digits = 3)))

# top 10 best movies by movie averages
edx_subset %>% count(movieId) %>% left_join(movie_avgs, by = "movieId") %>% left_join(movie_titles, by = "movieId") %>% arr
ange(desc(b_hat_m)) %>% select(title, b_hat_m, n) %>% slice(1:10) %>% knitr::kable()
```

title	b_hat_m	n
Hellhounds on My Trail (1999)	1.487535	1

title	b_hat_m	n
Satan's Tango (SÃ¡tÃ¡ntangÃ³) (1994)	1.487535	2
Shadows of Forgotten Ancestors (1964)	1.487535	1
Fighting Elegy (Kenka erejii) (1966)	1.487535	1
Sun Alley (Sonnenallee) (1999)	1.487535	1
Blue Light, The (Das Blaue Licht) (1932)	1.487535	1
Who's Singin' Over There? (a.k.a. Who Sings Over There) (Ko to tamo peva) (1980)	1.237535	4
Human Condition II, The (Ningen no joken II) (1959)	1.237535	4
Human Condition III, The (Ningen no joken III) (1961)	1.237535	4
Constantine's Sword (2007)	1.237535	2

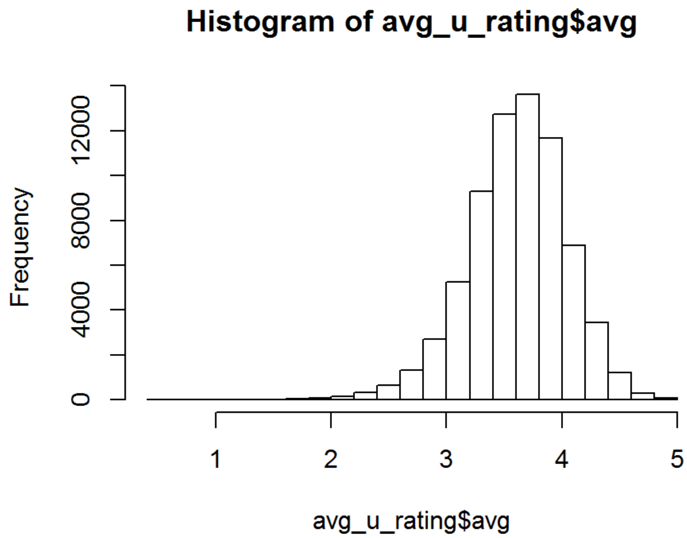
```
# top 10 worst movies by movie averages
edx_subset %>% count(movieId) %>% left_join(movie_avgs, by = "movieId") %>% left_join(movie_titles, by = "movieId") %>% arr
ange(b_hat_m) %>% select(title, b_hat_m, n) %>% slice(1:10) %>% knitr::kable()
```

title	b_hat_m	n
Besotted (2001)	-3.012465	2
Hi-Line, The (1999)	-3.012465	1
Accused (Anklaget) (2005)	-3.012465	1
Confessions of a Superhero (2007)	-3.012465	1
War of the Worlds 2: The Next Wave (2008)	-3.012465	2
SuperBabies: Baby Geniuses 2 (2004)	-2.717822	56
Hip Hop Witch, Da (2000)	-2.691037	14
Disaster Movie (2008)	-2.653090	32
From Justin to Kelly (2003)	-2.610455	199
Criminals (1996)	-2.512465	2

```
# Most of the movies, which are chosen as the best and worst, have received very few ratings. Most of them are obscure.
# We should not trust these noisy estimates and so need to use regularization
# However, before we go for regularization, we will try the user effects too
```

- Model By Movie + User Effects
    - Third model to take both biases introduced by movie and user effects into account in order to improvise the RMSE.

```
#----------3. By Movie + User Effects
# Each user rated differently -> can be seen by finding the average of ratings for each user
avg_u_rating <- edx_subset %>% select(userId, rating) %>% group_by(userId) %>% dplyr::summarise(avg = mean(rating))
hist(avg_u_rating$avg)
```




```
# Most of the average ratings are between 3 & 4.0

# As realized, there are user-specific effect (Happy Users, Cranky Users, Reasonable Users), we introduce b_hat_u
# Y_hat u,m = mu_hat + b_hat_m + b_hat_u + E u,m -> where b_hat_u (bias) is the average rating bias by the user u
user_avgs <- edx_subset %>% left_join(movie_avgs, by="movieId") %>% select(b_hat_m, userId, rating) %>% group_by(userId) %
>% dplyr::summarise(b_hat_u = mean(rating - mu_hat - b_hat_m))

# Relationship btw mu_hat, b_hat_m & b_hat_u -> Y_hat u,m = mu_hat + b_hat_m + b_hat_u
predictions <- edx_subset %>% left_join(movie_avgs, by = "movieId") %>% left_join(user_avgs, by = "userId") %>% mutate(pred
 = mu_hat + b_hat_m + b_hat_u) %>% .$pred
rmse_3 <- RMSE(edx_subset$rating, predictions)
rmse_results <- bind_rows(rmse_results, tibble(method = "By movie & user based average", RMSE = round(rmse_3, digits = 3)))

# top 10 best movies by user averages
user_avgs %>% left_join(edx_subset, by = "userId") %>% left_join(movie_titles, by = "movieId") %>% arrange(desc(b_hat_u)) %
>% select(title, b_hat_u) %>% slice(1:10) %>% knitr::kable()
```

title	b_hat_u
Johnny Mnemonic (1995)	1.890564
Waterworld (1995)	1.890564
Demolition Man (1993)	1.890564
Jurassic Park (1993)	1.890564
Independence Day (a.k.a. ID4) (1996)	1.890564
Nutty Professor, The (1996)	1.890564
Mars Attacks! (1996)	1.890564
Lost World: Jurassic Park, The (Jurassic Park 2) (1997)	1.890564
Starship Troopers (1997)	1.890564
Armageddon (1998)	1.890564

```
# top 10 worst movies by user averages
user_avgs %>% left_join(edx_subset, by = "userId") %>% left_join(movie_titles, by = "movieId") %>% arrange(b_hat_u) %>% sel
ect(title, b_hat_u) %>% slice(1:10) %>% knitr::kable()
```

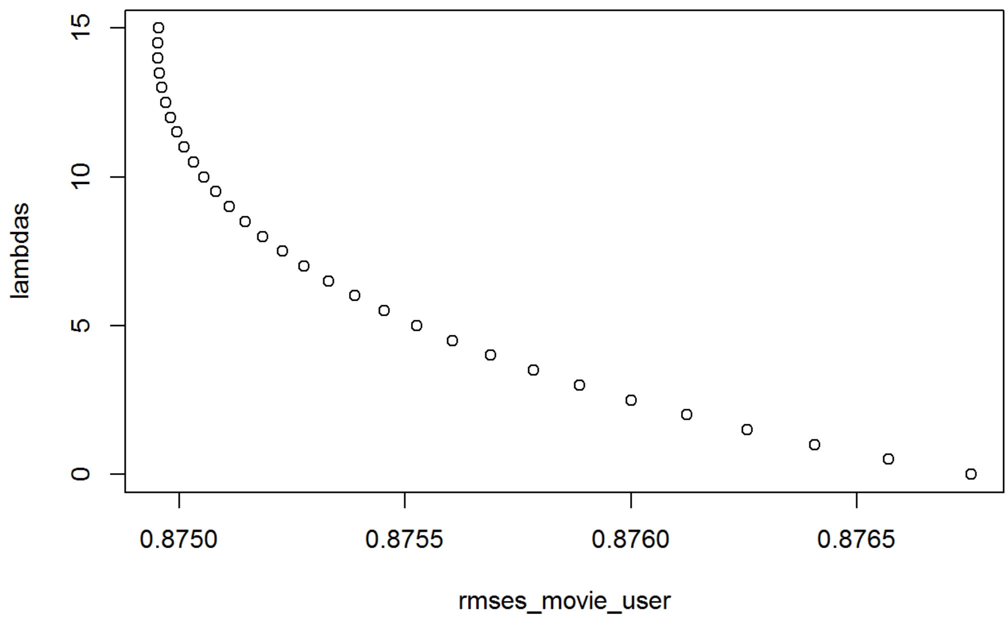
title	b_hat_u
Shawshank Redemption, The (1994)	-3.390564

title	b_hat_u
Philadelphia (1993)	-3.390564
Rudy (1993)	-3.390564
One Flew Over the Cuckoo's Nest (1975)	-3.390564
Amadeus (1984)	-3.390564
Raging Bull (1980)	-3.390564
Fried Green Tomatoes (1991)	-3.390564
Field of Dreams (1989)	-3.390564
Boogie Nights (1997)	-3.390564
Truman Show, The (1998)	-3.390564

```
# Most of the positively biased movies are popular/blockbuster movies vs. negatively biased ones are less popular around th
e world
```

- Model By Regularized Movie + Regularized User Effects
  - Our last model to apply regularization in order to further reduce the RMSE caused by both movie & user effects on the training set.
  - Using sapply function to determine RMSE for a range of lambdas on the cross validation set in order to determine the optimal lambda for the combined effects of movie & user
  - Optimal lambda is determined as 14.5.

```
#----------4. By Regularized Movie + Regularized User Effects
lambdas <- seq(0, 15, 0.5)

# Recalculate Lambda
just_sum_m <- edx_subset %>% group_by(movieId)%>% dplyr::summarise(s_m = sum(rating - mu_hat), n_hat_m = n())
just_sum_u <- edx_subset %>% group_by(userId)%>% dplyr::summarise(s_u = sum(rating - mu_hat), n_hat_u = n())
rmses_movie_user <- sapply(lambdas, function(l) {
    predicted_ratings <- edx_subset %>% left_join(just_sum_m, by = "movieId") %>% mutate(b_hat_m = s_m / (l + n_hat_m)) %>%
 left_join(just_sum_u, by = "userId") %>% mutate(b_hat_u = s_u / (l + n_hat_u)) %>% mutate(pred = (mu_hat + b_hat_m + b_hat
_u)) %>% .$pred
    RMSE(edx_subset$rating, predicted_ratings)
}
)
plot(rmses_movie_user, lambdas)
```




```
lambda_movie_user <- lambdas[which.min(rmses_movie_user)]

print(lambda_movie_user)
```

```
## [1] 14.5
```

```
# Revise movie_user_avgs with the regularized biases
movie_reg_avgs <- just_sum_m %>% mutate(b_hat_m_reg = s_m / (lambda_movie_user + n_hat_m))
user_reg_avgs <- edx_subset %>% left_join(movie_reg_avgs, by = "movieId") %>% select(b_hat_m_reg, userId, rating) %>% group
_by(userId) %>% dplyr::summarise(b_hat_u_reg = mean(rating - mu_hat - b_hat_m_reg))

# Relationship btw mu_hat, b_hat_m_reg & b_hat_u_reg -> Y_hat u,m = mu_hat + b_hat_m_reg + b_hat_u_reg
predictions <- edx_subset %>% left_join(movie_reg_avgs, by = "movieId") %>% left_join(user_reg_avgs, by = "userId") %>% mut
ate(pred = mu_hat + b_hat_m_reg + b_hat_u_reg) %>% .$pred
rmse_4 <- RMSE(edx_subset$rating, predictions)
rmse_results <- bind_rows(rmse_results, tibble(method = "By reg. movie & reg. user based average", RMSE = round(rmse_4, dig
its = 3)))

# top 10 best movies with regularized bias - impact by movie
edx_subset %>% count(movieId) %>% left_join(movie_reg_avgs, by = "movieId") %>% left_join(movie_titles, by = "movieId") %>%
 arrange(desc(b_hat_m_reg)) %>% select(title, b_hat_m_reg, n) %>% slice(1:10) %>% knitr::kable()
```

title	b_hat_m_reg	n
Shawshank Redemption, The (1994)	0.9421783	28015
Godfather, The (1972)	0.9021637	17747
Usual Suspects, The (1995)	0.8528172	21648
Schindler’s List (1993)	0.8504964	23193
Casablanca (1942)	0.8069169	11232
Rear Window (1954)	0.8047158	7935
Sunset Blvd. (a.k.a. Sunset Boulevard) (1950)	0.7994472	2922
Third Man, The (1949)	0.7950749	2967

title	b_hat_m_reg	n
Double Indemnity (1944)	0.7930136	2154
Seven Samurai (Shichinin no samurai) (1954)	0.7920656	5190

```
# top 10 worst movies with regularized bias - impact by movie
edx_subset %>% count(movieId) %>% left_join(movie_reg_avgs, by = "movieId") %>% left_join(movie_titles, by = "movieId") %>%
 arrange(b_hat_m_reg) %>% select(title, b_hat_m_reg, n) %>% slice(1:10) %>% knitr::kable()
```

title	b_hat_m_reg	n
From Justin to Kelly (2003)	-2.433164	199
PokÃ©mon Heroes (2003)	-2.245596	137
Glitter (2001)	-2.241091	339
Gigli (2003)	-2.216493	313
Pokemon 4 Ever (a.k.a. PokÃ©mon 4: The Movie) (2002)	-2.177912	202
Barney's Great Adventure (1998)	-2.173451	208
SuperBabies: Baby Geniuses 2 (2004)	-2.158838	56
Turbo: A Power Rangers Movie (1997)	-2.076894	394
Son of the Mask (2005)	-2.030957	165
House of the Dead, The (2003)	-2.025079	209

```
# top 10 best movies with regularized bias - impact by user
user_reg_avgs %>% left_join(edx_subset, by = "userId") %>% left_join(movie_titles, by = "movieId") %>% arrange(desc(b_hat_u
_reg)) %>% select(title, b_hat_u_reg) %>% slice(1:10) %>% knitr::kable()
```

title	b_hat_u_reg
Johnny Mnemonic (1995)	1.889827
Waterworld (1995)	1.889827
Demolition Man (1993)	1.889827
Jurassic Park (1993)	1.889827
Independence Day (a.k.a. ID4) (1996)	1.889827
Nutty Professor, The (1996)	1.889827
Mars Attacks! (1996)	1.889827
Lost World: Jurassic Park, The (Jurassic Park 2) (1997)	1.889827
Starship Troopers (1997)	1.889827
Armageddon (1998)	1.889827

```
# top 10 worst movies with regularized bias - impact by user
user_reg_avgs %>% left_join(edx_subset, by = "userId") %>% left_join(movie_titles, by = "movieId") %>% arrange(b_hat_u_reg)
 %>% select(title, b_hat_u_reg) %>% slice(1:10) %>% knitr::kable()
```

title	b_hat_u_reg
Shawshank Redemption, The (1994)	-3.389866
Philadelphia (1993)	-3.389866
Rudy (1993)	-3.389866

title	b_hat_u_reg
One Flew Over the Cuckoo's Nest (1975)	-3.389866
Amadeus (1984)	-3.389866
Raging Bull (1980)	-3.389866
Fried Green Tomatoes (1991)	-3.389866
Field of Dreams (1989)	-3.389866
Boogie Nights (1997)	-3.389866
Truman Show, The (1998)	-3.389866

```
# Though it seems regularization didn't make a huge improvements for user effects, it did for movie effects as we see the m
ovies like Godfather, The (1972) and Shawshank Redemption, The (1994) are grouped together with similar ranks

#----------Prediction Modeling ends
```

- Apply model on validation set
  - Finally, apply the best model (Fourth/Last model in our case) on the validation set and capture the RMSE.

```
#----------Prediction on Validation starts
validation_subset <- validation %>% select(movieId, userId)
validation_ratings <- validation$rating

predictions <- validation_subset %>% left_join(movie_reg_avgs, by = "movieId") %>% left_join(user_reg_avgs, by = "userId")
 %>% mutate(pred = mu_hat + b_hat_m_reg + b_hat_u_reg) %>% .$pred
rmse_5 <- RMSE(validation_ratings, predictions)
rmse_results <- bind_rows(rmse_results, tibble(method = "Reg. effects on validation set", RMSE = round(rmse_5, digits = 3
)))
#----------Prediction on Validation ends
```

Results

Below is the captured RMSE results by each model on the training set and the final one on the test set.

```
# The summary of RMSE results would show how the RMSE has been reduced through out this process
rmse_results
```

```
##                                    method  RMSE
## 1                        By overall average 1.060
## 2                    By movie based average 0.942
## 3             By movie & user based average 0.857
## 4 By reg. movie & reg. user based average 0.857
## 5            Reg. effects on validation set 0.865
```

Conclusion

Running the final model with the regulaized averages and biases introduced by both movie and user effects on the validation set of 1 Million records (unknown ratings), the predicted ratings express that the RMSE 0.865 tends to be a little higher than the one achieved on the training set (known ratings) 0.856 which is normal.

The RMSE/accuracy can be further optimized using Principle Component Analysis by identifying the principal components. However, this report concludes with the RMSE 0.865 < 0.8775, as stated under the goal.

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

End