MovieLens Recommender System

Harvard Data Science Capstone Project

Contents

1. Executive Summary

This is Harvard Data Science Capstone project for creating a recommender system using MovieLens dataset. The movielens dataset used for this final assignment contains approximately 10 Millions of movie ratings, then divided to **9 Million** for training and **one Million** for validation. It is a small subset of a much larger dataset with several millions of ratings. Due to our laptop limitation in RAM and testing time, we use the small training dataset, there are approximately **70.000 users** and **10.000 different movies** divided in 20 genres such as Action, Adventure, Horror, Drama, Thriller and more.
After the initial data exploration, the recommender systems built on this dataset are evaluated and chosen based on the RMSE - Root Mean Squared Error that should be at least lower than **0.8775**

RMSE =`sqrt(mean((true_ratings - predicted_ratings)^2))`

For accomplishing this goal, the **Regularized Movie+User+Genre Model** is capable to reach the lowest RMSE,

2. Exploratory Data Analysis

1. Data description

List of feature description for dataset
- **userId** : the unique identification number for each user.
- **movieId:** the unique identification number for each movie.
- **rating** : the rating of one movie by an user. Ratings are made on a 5-Star scale with half-star increments.
- **timestamp** : the timestamp for one specific rating provided by one user.
- **title** : the title of each movie including the year of release.
- **genres** : a list of pipe-separated of genre of each movie.

Let’s look at the data structure

```
# Data type glimpse
glimpse(edx_dat)
```

```
## Rows: 9,000,061
## Columns: 6
## $ userId    <int> 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1, 1, ~
## $ movieId   <dbl> 122, 185, 231, 292, 316, 329, 355, 356, 362, 364,
370, 377, ~
## $ rating    <dbl> 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5,
5, 5, ~
## $ timestamp <int> 838985046, 838983525, 838983392, 838983421,
838983392, 83898~
## $ title     <chr> "Boomerang (1992)", "Net, The (1995)", "Dumb & Dumber
(1994)~
## $ genres    <chr> "Comedy|Romance", "Action|Crime|Thriller", "Comedy",
"Action~
```

```
# First 6 rows of data
```

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

2. Check numbers for distinct users and movies, Avg, Sd

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

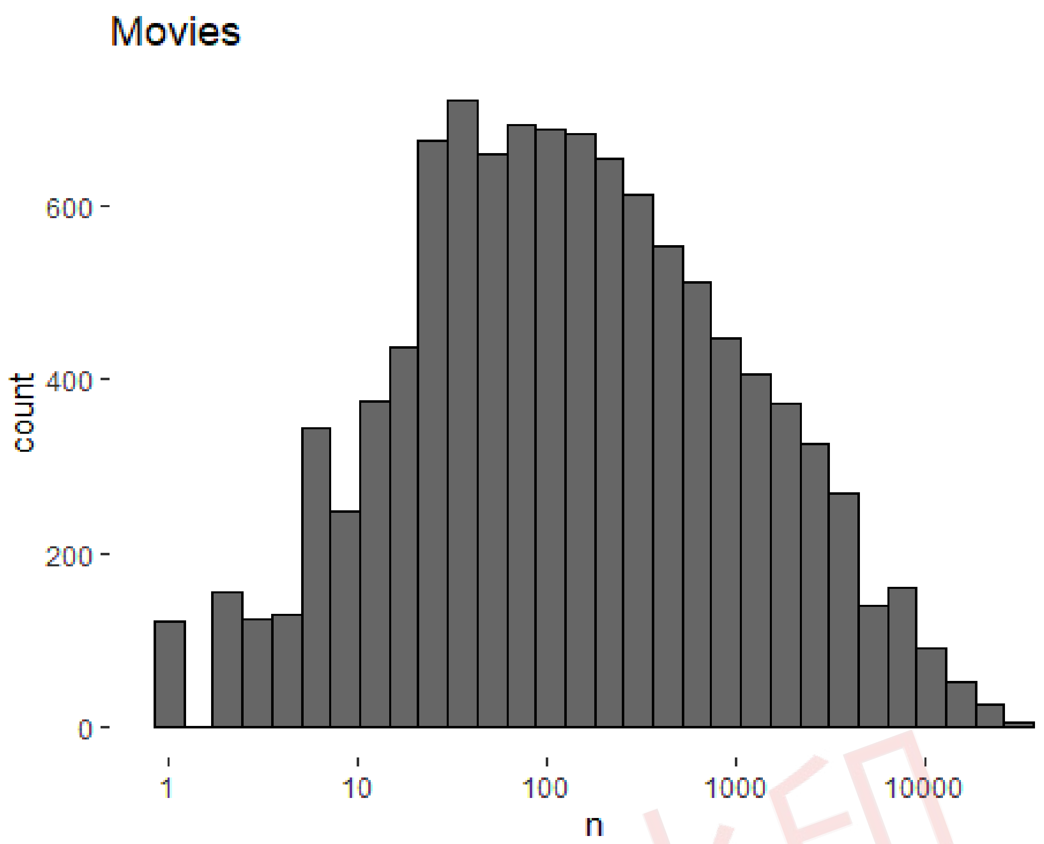
```
##   n_users n_movies
## 1   69878    10677
```

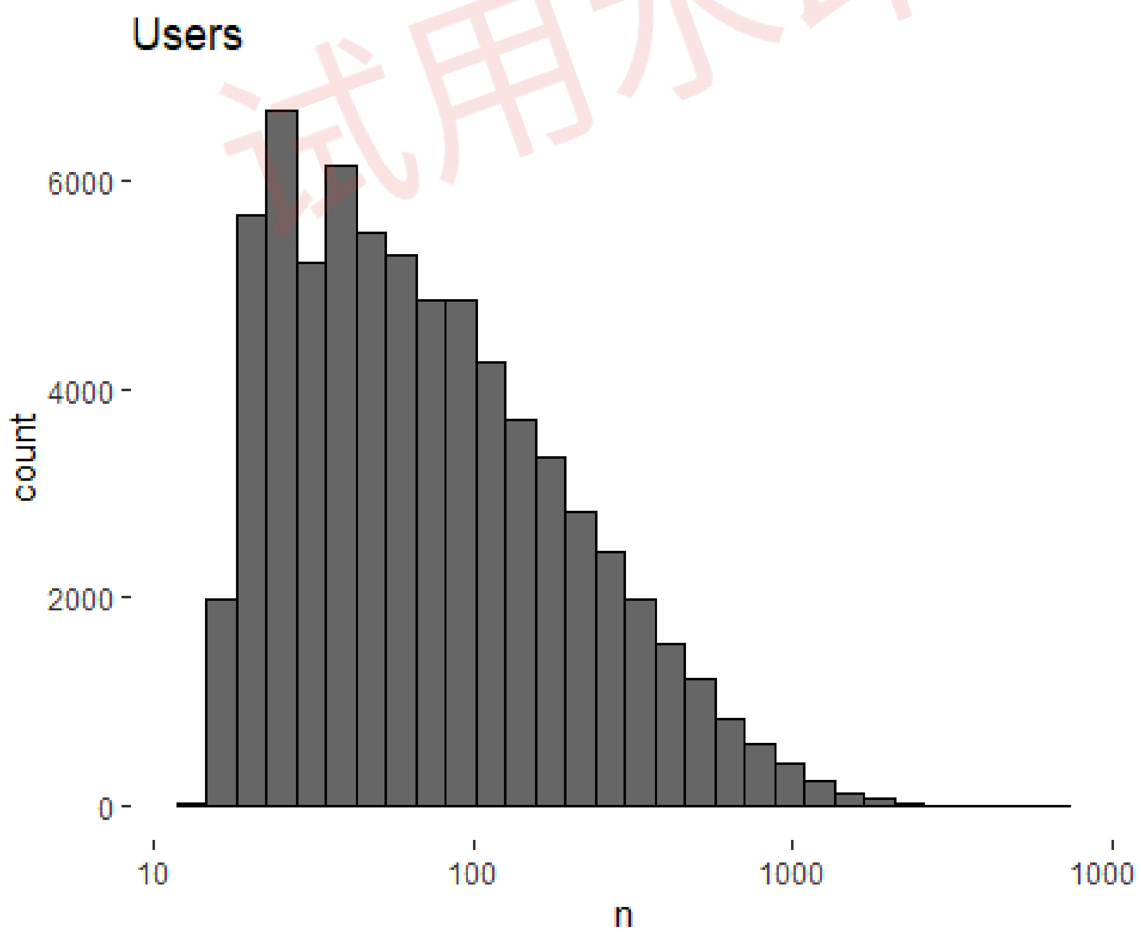
From below calculation we see movies in training has average score **3.51** with SD +/-1.06, movie rating is approximate in range of **2.45 – 4.57**

```
# Average Ratings for all movies in training
round(mean(edx_dat$rating),4)
## [1] 3.5125
# Standard deviation for all movies in training
round(sd(edx_dat$rating),4)
## [1] 1.0604
```

3. Plot Movie ratings and Users ratings

Some movies have more ratings than others, some users rated more movies than others


Movies
count
600
400
200
0
1
10
100
1000
10000
n


Users
count
6000
4000
2000
0
10
100
1000
1000
n

4. Top movies and top users

```r
# Top 5 most-rated movies
top5 <- edx_dat %>%
   dplyr::count(movieId) %>%
   top_n(5) %>%
   pull(movieId)
top5
## [1] 296 318 356 480 593
```

```r
# Top 5 frequent users
top5_u <- edx_dat %>%
   dplyr::count(userId) %>%
   top_n(5) %>%
   pull(userId)
top5_u
## [1] 14463 27468 59269 67385 68259
```

```r
# Ratings of Top5 Movies given by top5 users
tab <- edx_dat %>%
   filter(movieId %in% top5) %>%
   filter(userId %in% top5_u) %>%
   select(userId, title, rating) %>%
   spread(title, rating)
print(tab)
##   userId Forrest Gump (1994) Jurassic Park (1993) Pulp Fiction (1994)
## 1  14463                 3.0                    4                 4.0
## 2  27468                 5.0                    5                 5.0
## 3  59269                 4.0                    3                 4.5
## 4  67385                 3.5                    4                 5.0
## 5  68259                 4.0                    4                 5.0
##   Shawshank Redemption, The (1994) Silence of the Lambs, The (1991)
## 1                              4.0                                4
## 2                              5.0                                5
## 3                              4.0                                4
## 4                              4.5                                5
## 5                              5.0                                5
```

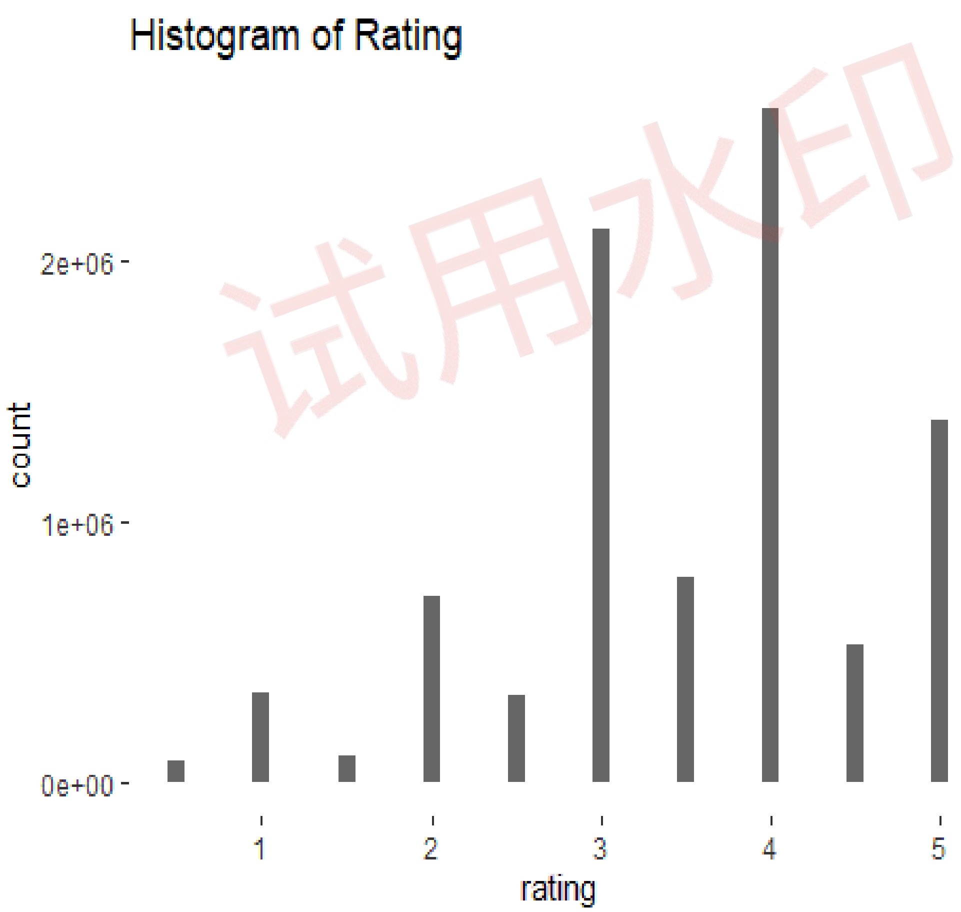
```r
# Top 20 most rated movies and their Titles, many are popular movies
t<-edx_dat %>%
   group_by(title) %>%
   summarise(count = n()) %>%
   arrange(desc(count)) %>%
   head(n=20)
print.data.frame(t)
```

```
##                                                           title count
## 1                                           Pulp Fiction (1994) 31336
## 2                                           Forrest Gump (1994) 31076
## 3                              Silence of the Lambs, The (1991) 30280
## 4                                          Jurassic Park (1993) 29291
## 5                              Shawshank Redemption, The (1994) 27988
## 6                                             Braveheart (1995) 26258
## 7                             Terminator 2: Judgment Day (1991) 26115
```

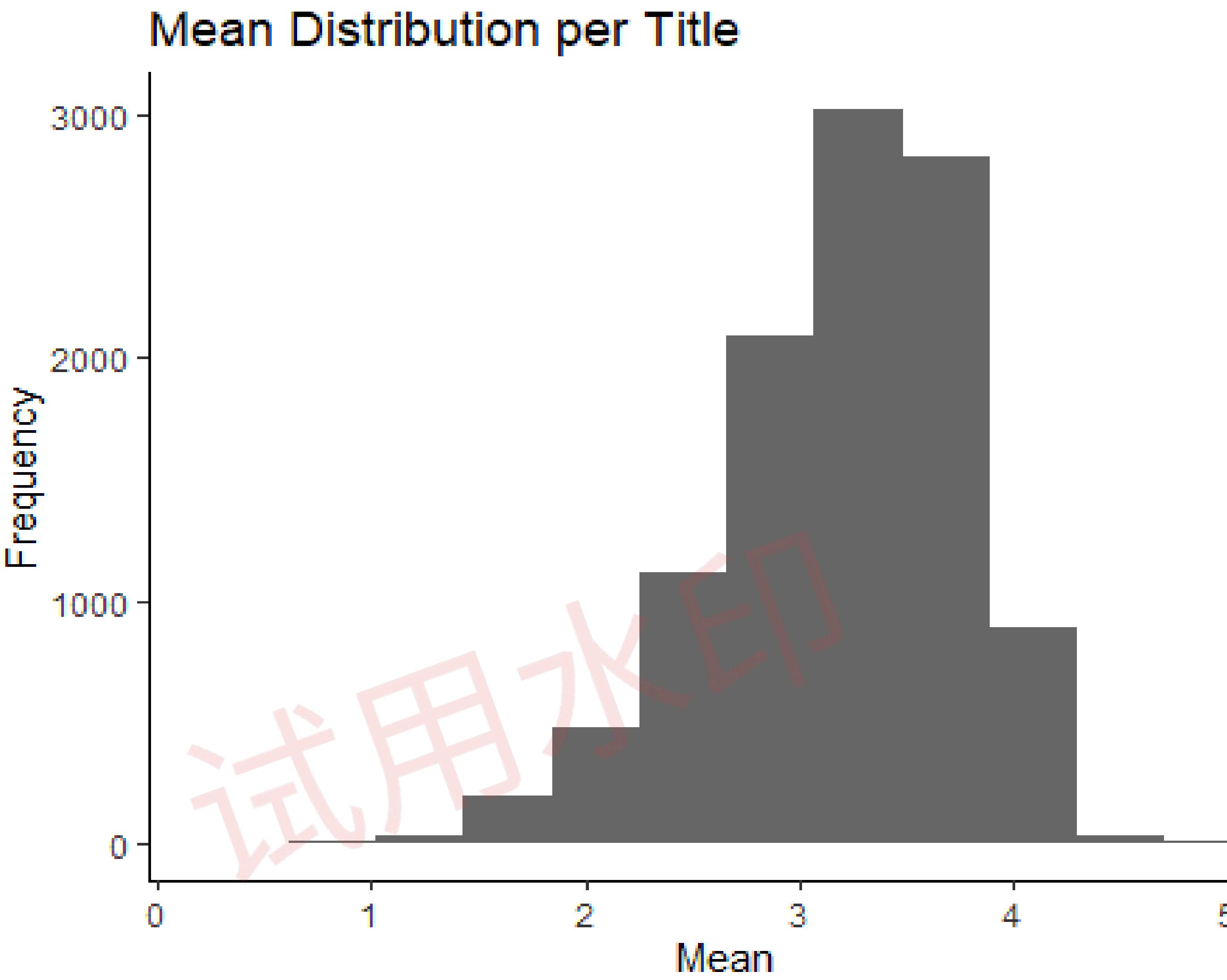
```
## 8                                               Fugitive, The (1993) 26050
## 9  Star Wars: Episode IV - A New Hope (a.k.a. Star Wars) (1977) 25809
## 10                                                     Batman (1989) 24343
## 11                                                  Apollo 13 (1995) 24277
## 12                                                  Toy Story (1995) 23826
## 13                              Independence Day (a.k.a. ID4) (1996) 23360
## 14                                         Dances with Wolves (1990) 23312
## 15                                           Schindler's List (1993) 23234
## 16                                                 True Lies (1994) 22786
## 17              Star Wars: Episode VI - Return of the Jedi (1983) 22629
## 18                             12 Monkeys (Twelve Monkeys) (1995) 21959
## 19                                      Usual Suspects, The (1995) 21533
## 20                                                      Speed (1994) 21384
```

5. Rating distribution

Below histogram shows that there are a small amount of votes below 3. Half-Star votes are less common than “Full-Star” in each category of from 1 to 5.



From below it shows most movies rating is distributed around 2.5 – 4.5, by title.



6. The list of unique genres for all movies

```
# View of all unique genres
unique_genres_list <- str_extract_all(unique(edx_dat$genres), "[^|]+") %>%
  unlist() %>%unique()
unique_genres_list
```

```
##  [1] "Comedy"              "Romance"             "Action"
##  [4] "Crime"               "Thriller"            "Drama"
##  [7] "Sci-Fi"              "Adventure"           "Children"
## [10] "Fantasy"             "War"                 "Animation"
## [13] "Musical"             "Western"             "Mystery"
```

```
## [16] "Film-Noir"          "Horror"             "Documentary"
## [19] "IMAX"               "(no genres listed)"
```

3. Analysis – Model Building and Evaluation

```
options(digits = 4)
# The RMSE function that will be used in this project is:
RMSE <- function(true_ratings = NULL, predicted_ratings = NULL) {
    sqrt(mean((true_ratings - predicted_ratings)^2))
}
```

1. Native Mean Avg Model

The simplest model that can build is a Naive Model that predict ALWAYS the mean. We use it as our base-line model.

```
# Calculate the average of all movies
mu_hat <- mean(edx_dat$rating)
mu_hat  #3.512
```

```
## [1] 3.512
```

```
# Predict the RMSE on the validation set
rmse_mean <- RMSE(validation_dat$rating, mu_hat) #1.06065
```

```
# Creating a results dataframe
results <- data.frame(model="Naive Mean Avg Model", RMSE=rmse_mean)
print.data.frame(results)
```

```
##                  model  RMSE
## 1 Naive Mean Avg Model 1.061
```

2. Movie Avg Model

The first Non-Naive Model we use, in this case the movie-model which takes into account of movie can have a '**movie bias**' score that are rated higher or lower respect to the average score. It gives the RMSE to 0.94 below 1, but still far from our target 0.87 below.

```
# Calculate the average by movie
movie_avgs <- edx_dat %>%
   group_by(movieId) %>%
   summarize(b_i = mean(rating - mu_hat))
```

```
# Compute the predicted ratings on validation dataset
rmse_movie_model <- validation_dat %>%
   left_join(movie_avgs, by='movieId') %>%
   mutate(pred = mu_hat + b_i) %>%
   pull(pred)
```

```
rmse_movie_model_result <- RMSE(validation_dat$rating, rmse_movie_model)
```

```r
# Adding row to the results
results <- results %>% add_row(model="Movie-Based Model",
RMSE=rmse_movie_model_result)
print.data.frame(results)
```

```
##                  model   RMSE
## 1 Naive Mean Avg Model 1.0607
## 2    Movie-Based Model 0.9437
```

3. Movie User Avg Model

The second Non-Naive Model takes into account that the users have different preference and rate differently, so it introduced the ‘**user-bias score**’. The RMSE on validation set is 0.86, it’s very good. This model archives desired performance, but still have room to improve

```r
# Calculate the average by user
user_avgs <- edx_dat %>%
   left_join(movie_avgs, by='movieId') %>%
   group_by(userId) %>%
   summarize(b_u = mean(rating - mu_hat - b_i))

# Compute the predicted ratings on validation dataset
rmse_movie_user_model <- validation_dat %>%
   left_join(movie_avgs, by='movieId') %>%
   left_join(user_avgs, by='userId') %>%
   mutate(pred = mu_hat + b_i + b_u) %>%
   pull(pred)

rmse_movie_user_model_result <- RMSE(validation_dat$rating,
rmse_movie_user_model)

# Adding row to the results
results <- results %>% add_row(model="Movie+User Based Model",
RMSE=rmse_movie_user_model_result)
print.data.frame(results)
```

```
##                    model   RMSE
## 1   Naive Mean Avg Model 1.0607
## 2      Movie-Based Model 0.9437
## 3 Movie+User Based Model 0.8655
```

4. Movie User Genre Avg Model

This model introduced a measure for ‘**genre-bias**’**s** score. By using B_i in below calculation it shows it archived desired performance of below 0.87, improved a little bit.

```r
# calculate genre bias
genre_avgs <- edx_dat %>%
   left_join(movie_avgs, by='movieId') %>%
   left_join(user_avgs, by='userId') %>%
   group_by(genres) %>%
```

```
  summarize(b_u_g = mean(rating - mu_hat - b_i - b_u))


# Compute the predicted ratings on validation dataset
rmse_movie_user_genre_model <- validation_dat %>%
  left_join(movie_avgs, by='movieId') %>%
  left_join(user_avgs, by='userId') %>%
  left_join(genre_avgs, by='genres') %>%
  mutate(pred = mu_hat + b_i + b_u + b_u_g) %>%
  pull(pred)

rmse_movie_user_genre_model_result <-
RMSE(validation_dat$rating,rmse_movie_user_genre_model)

# Adding row to the results
results <- results %>% add_row(model="Movie+User+Genre Based Model",
RMSE=rmse_movie_user_genre_model_result)
print.data.frame(results)
```

```
##                          model   RMSE
## 1          Naive Mean Avg Model 1.0607
## 2             Movie-Based Model 0.9437
## 3        Movie+User Based Model 0.8655
## 4 Movie+User+Genre Based Model 0.8652
```

5. Regularized Movie Model

The regularization method allows us to add a penalty λ (lambda) to penalize movies with large estimates from a small sample size.

```
set.seed(1)
lambdas <- seq(0, 10, 0.1)

# Compute the predicted ratings on validation dataset using different
lambda
rmses <- sapply(lambdas, function(lambda) {

  # Calculate the average by user
  b_i <- edx_dat %>%
    group_by(movieId) %>%
    summarize(b_i = sum(rating - mu_hat) / (n() + lambda))

  # Compute the predicted ratings on validation dataset
  predicted_ratings <- validation_dat %>%
    left_join(b_i, by='movieId') %>%
    mutate(pred = mu_hat + b_i) %>%
    pull(pred)

  # Predict the RMSE on the validation set
    return(RMSE(validation_dat$rating, predicted_ratings))
})

# Get the lambda value that minimize the RMSE
```

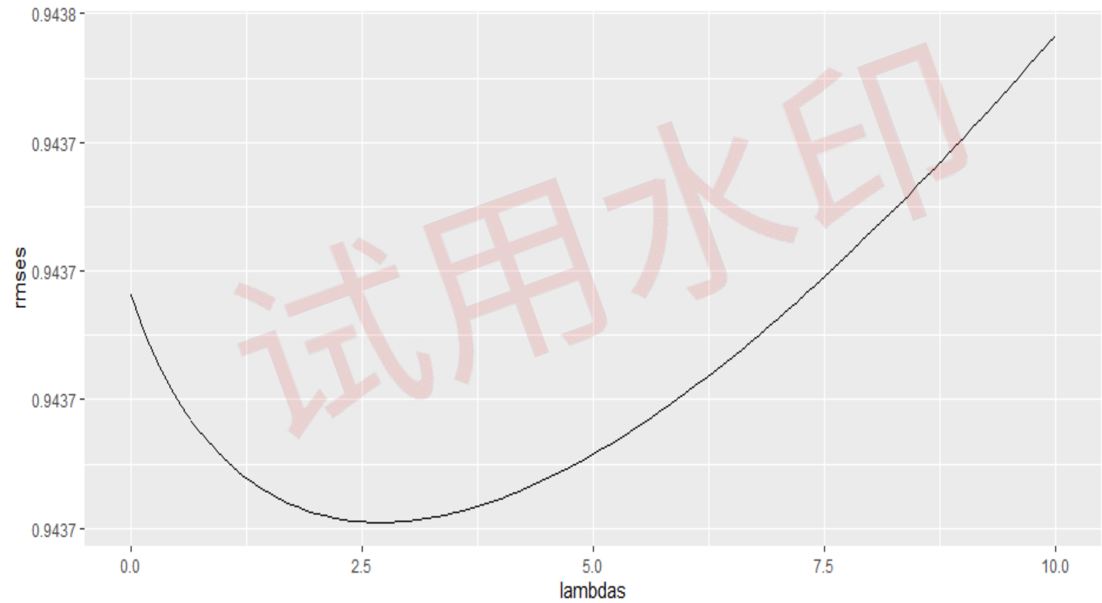
```
min_lambda <- lambdas[which.min(rmses)]

# RMSE
rmse_regularized_movie_model <- min(rmses)

# Adding row to the results
results <- results %>% add_row(model="Regularized Movie-Based Model", 
RMSE=rmse_regularized_movie_model)
print.data.frame(results)
```

```
##                           model   RMSE
## 1           Naive Mean Avg Model 1.0607
## 2              Movie-Based Model 0.9437
## 3          Movie+User Based Model 0.8655
## 4    Movie+User+Genre Based Model 0.8652
## 5 Regularized Movie-Based Model 0.8637
```

```
# Plot the RMSEs vs Lambda - Movie Model
```



6. Regularized Movie + User Model

this step took more than an hour at my laptop with 4G RAM and windows 10

```
set.seed(1)

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

  # Calculate the average by movie
  b_i <- edx_dat %>%
     group_by(movieId) %>%
```

```
    summarize(b_i = sum(rating - mu_hat) / (n() + lambda))

  # Calculate the average by user
  b_u <- edx_dat %>%
    left_join(b_i, by='movieId') %>%
    group_by(userId) %>%
    summarize(b_u = sum(rating - b_i - mu_hat) / (n() + lambda))

  # Compute the predicted ratings on validation dataset
  predicted_ratings <- validation_dat %>%
    left_join(b_i, by='movieId') %>%
    left_join(b_u, by='userId') %>%
    mutate(pred = mu_hat + b_i + b_u) %>%
    pull(pred)

  # Predict the RMSE on the validation set
  return(RMSE(validation_dat$rating, predicted_ratings))
})

# Get the lambda value that minimize the RMSE
min_lambda <- lambdas[which.min(rmses)]

# Predict the RMSE on the validation set
rmse_regularized_movie_user_model <- min(rmses)

# Adding the results to the results dataset
results <- results %>% add_row(model="Regularized Movie+User Based Model",
RMSE=rmse_regularized_movie_user_model)
print.data.frame(results)
##                                model   RMSE
## 1            Naive Mean Avg Model 1.0607
## 2              Movie-Based Model 0.9437
## 3          Movie+User Based Model 0.8655
## 4    Movie+User+Genre Based Model 0.8652
## 5    Regularized Movie-Based Model 0.8637
## 6 Regularized Movie+User Based Model 0.8628
```


Note: the regularized movie+user+genre model runs crashed my RStudio, tried to run twice over 12+ hours but not able to finish. I include the code in the appendix.

4. Conclusion

From above trained different models results, it shows that movieId and userId contribute more than the genre predictor. Without regularization, the model can achieves desired RMSE of approximate/below 0.87, but after applying regularization the results makes a good improvement in either movie predictor or user predictors, suggestion regulation is a good method solving this problem, and there could be other unknown factors not included in this dataset, that makes it possible to reach a final RSME of 0.86 or even better for the trained models.

From the data insights, it also shows no data ‘perfect’, data pre-processing or cleaning is very important for later result; data sizes Do matter, usually the larger the better, however, it could be limited by computer resources in MEM/CPU and crash your computer, so pick a small dataset for training, test your algorithm appropriately, validate at the larger dataset is a better way. There are other possible data not included in this dataset, for example, a user could rate movies influenced by his/her family opinion so the rate given maybe not his/her original thought; another example like the timing or seasonal effect, like the media propaganda in certain period of time for a specific movie then it could make other movies in same genre scoring up.

It’s been a great pleasure with Harvard on edx in the past for this Data Science program, from start to finish, from a total beginner with R to be a good player with R, added with knowledge gain by quiz/exercise/midterm/final, I really appreciate edx and Harvard provide such opportunity to us, so we the working adults can have a chance to learn Data Science! Thank you very much, warmest memory and learning experience!

5. Appendix

1. code provided by edx

```
#################################################################
# Create edx set, validation set
#################################################################
# Note: this process could take a couple of minutes
# use readRDS to open saved files previously processed

library(tidyverse)
library(caret)
library(data.table)
library(dplyr)
library(tidyverse)
library(kableExtra)
library(tidyr)
library(stringr)
library(forcats)
library(ggplot2)

# set working dir
setwd("C:\\Users\\user\\Desktop\\Harvard_R\\CapStone\\MovieLensRecommen
dation")

# 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"))
# Save ratings object at my laptop file size around 46MB
#saveRDS(ratings, "ratings.rda")

#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(movieId),
#                                            title =
as.character(title),
#                                            genres =
as.character(genres))
#saveRDS(movies, "movies.rda")

#movielens <- left_join(ratings, movies, by = "movieId")
#saveRDS(movielens, "movielens.rda")

# Validation set will be 10% of MovieLens data
#set.seed(1)
#movielens_model<-readRDS("movielens.rda")
#test_index <- createDataPartition(y = movielens_model$rating, times =
1, p = 0.1, list = FALSE)
#edx <- movielens_model[-test_index,]
#temp <- movielens_model[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)

#saveRDS(edx, "edx.rda")
#saveRDS(validation, "validation.rda")

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


2. my movielens R code

```
# Note: use readRDS to open saved files previously processed

library(tidyverse)
library(caret)
```

```
library(data.table)
library(dplyr)
library(tidyverse)
library(kableExtra)
library(tidyr)
library(stringr)
library(forcats)
library(ggplot2)

# set working dir
setwd("C:\\Users\\user\\Desktop\\Harvard_R\\CapStone\\MovieLensRecommen
dation")


#####################################################################
# MovieLens Reccomender System Project
edx_dat <- readRDS("edx.rda")
validation_dat <-readRDS("validation.rda")

# --- Descriptive Analysis ----
# data type glimpse
glimpse(edx_dat)
head(edx_dat)

# Distinct number of user and movieId
edx_dat %>%
   summarize(n_users = n_distinct(userId),
             n_movies = n_distinct(movieId))

# Avg rating and SD
round(mean(edx_dat$rating), digits = 4)
round(sd(edx_dat$rating), digits = 4)

# Some movies have more ratings than others
edx_dat %>%
   dplyr::count(movieId) %>%
   ggplot(aes(n)) +
   geom_histogram(bins = 30, color = "black") +
   scale_x_log10() +
   ggtitle("Movies")

# Some users rate more frequently than others
edx_dat %>%
   dplyr::count(userId) %>%
   ggplot(aes(n)) +
   geom_histogram(bins = 30, color = "black") +
   scale_x_log10() +
   ggtitle("Users")

#  Top 5 most rated movies
top5 <- edx_dat %>%
   dplyr::count(movieId) %>%
   top_n(5) %>%
   pull(movieId)
top5

# Top 5 frequent users
top5_u <- edx_dat %>%
```

```
   dplyr::count(userId) %>%
   top_n(5) %>%
   pull(userId)
top5_u

# Top 5 mostly rated Movies rating given by top 5 frequent users
tab <- edx_dat %>%
   filter(movieId %in% top5) %>%
   filter(userId %in% top5_u) %>%
   select(userId, title, rating) %>%
   spread(title, rating)
print(tab)

# Top 20 most rated movies and their Titles
t<-edx_dat %>%
   group_by(title) %>%
   summarise(count = n()) %>%
   arrange(desc(count)) %>%
   head(n=20)
print.data.frame(t)


# ratings distribution histogram
edx_dat %>%
   ggplot(aes(rating)) +
   geom_histogram(binwidth = 0.1)+
   labs(title="Histogram of Rating")


# mean rating distribution per title
edx_dat %>%
   group_by(title) %>%
   summarise(mean = mean(rating)) %>%
   ggplot(aes(mean)) +
   theme_classic() +
   geom_histogram(bins=12) +
   labs(title = "Mean Distribution per Title",
        x = "Mean",
        y = "Frequency")

# View of all unique genres
unique_genres_list <- str_extract_all(unique(edx_dat$genres), "[^|]+")
%>%
   unlist() %>%unique()
unique_genres_list


# --- Analysis part ---
options(digits = 4)
# The RMSE function that will be used in this project is:
RMSE <- function(true_ratings = NULL, predicted_ratings = NULL) {
    sqrt(mean((true_ratings - predicted_ratings)^2))
}

# Split into train 0.75 and test set 0.25 of dataset
set.seed(1)
test_index <- createDataPartition(edx_dat$rating, times = 1, p = 0.25,
list = FALSE)
```

```r
test <- edx_dat[test_index,]
train <- edx_dat [-test_index,]

# --- 1. Native Mean Avg Model ---
# Calculate the average of all movies
mu_hat <- mean(edx_dat$rating)
mu_hat  #3.512

# Predict the RMSE on the validation set
rmse_mean <- RMSE(validation_dat$rating, mu_hat) #1.06065

# Creating a results dataframe
results <- data.frame(model="Naive Mean Avg Model", RMSE=rmse_mean)
print.data.frame(results)

# --- 2. Movie Avg Model ---
# Calculate the average by movie
movie_avgs <- train %>%
   group_by(movieId) %>%
   summarize(b_i = mean(rating - mu_hat))

# Compute the predicted ratings on validation dataset
rmse_movie_model <- test  %>%
   left_join(movie_avgs, by='movieId') %>%
   mutate(pred = mu_hat + b_i) %>%
   pull(pred)

rmse_movie_model_result <- RMSE(test$rating, rmse_movie_model)

# Adding row to the results
results <- results %>% add_row(model="Movie-Based Model",
RMSE=rmse_movie_model_result)
print.data.frame(results)


# --- 3. Movie User Avg Model ---
# Calculate the average by user
user_avgs <- train %>%
   left_join(movie_avgs, by='movieId') %>%
   group_by(userId) %>%
   summarize(b_u = mean(rating - mu_hat - b_i))

# Compute the predicted ratings on validation dataset
rmse_movie_user_model <- testt %>%
   left_join(movie_avgs, by='movieId') %>%
   left_join(user_avgs, by='userId') %>%
   mutate(pred = mu_hat + b_i + b_u) %>%
   pull(pred)

rmse_movie_user_model_result <- RMSE(test$rating,
rmse_movie_user_model)

# Adding row to the results
results <- results %>% add_row(model="Movie+User Based Model",
RMSE=rmse_movie_user_model_result)
print.data.frame(results)
```

```
# --- 4 Movie User Genre Avg Model ---
# calculate genre bias
genre_avgs <- train %>%
   left_join(movie_avgs, by='movieId') %>%
   left_join(user_avgs, by='userId') %>%
   group_by(genres) %>%
   summarize(b_u_g = mean(rating - mu_hat - b_i - b_u))


# Compute the predicted ratings on validation dataset
rmse_movie_user_genre_model <- test %>%
   left_join(movie_avgs, by='movieId') %>%
   left_join(user_avgs, by='userId') %>%
   left_join(genre_avgs, by='genres') %>%
   mutate(pred = mu_hat + b_i + b_u + b_u_g) %>%
   pull(pred)

rmse_movie_user_genre_model_result <-
RMSE(test$rating,rmse_movie_user_genre_model)

# Adding row to the results
results <- results %>% add_row(model="Movie+User+Genre Based Model",
RMSE=rmse_movie_user_genre_model_result)
print.data.frame(results)


# --- 5 Regularized Movie Model  ---
set.seed(1)
lambdas <- seq(0, 10, 0.1)

# Compute the predicted ratings on validation dataset using different
lambda
rmses <- sapply(lambdas, function(lambda) {

  # Calculate the average by user
   b_i <- train %>%
      group_by(movieId) %>%
      summarize(b_i = sum(rating - mu_hat) / (n() + lambda))

   # Compute the predicted ratings on validation dataset
   predicted_ratings <- test %>%
      left_join(b_i, by='movieId') %>%
      mutate(pred = mu_hat + b_i) %>%
      pull(pred)

   # Predict the RMSE on the validation set
      return(RMSE(test$rating, predicted_ratings))
})

# Get the lambda value that minimize the RMSE
min_lambda <- lambdas[which.min(rmses)]

# RMSE
rmse_regularized_movie_model <- min(rmses)

# Adding row to the results
results <- results %>% add_row(model="Regularized Movie-Based Model",
RMSE=rmse_regularized_movie_model)
```

```
print.data.frame(results)

# Plot RMSE vs Lambda
data.frame(lam = lambdas, rmse=rmses)%>%
   ggplot(aes(x=lambdas, y=rmses)) +
     geom_line()
     labs(title = "RMSEs vs Lambdas - Regularized Movie Model")


# --- 6 Regularized Movie + User Model ---
set.seed(1)

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

   # Calculate the average by movie
   b_i <- edx_dat %>%
      group_by(movieId) %>%
      summarize(b_i = sum(rating - mu_hat) / (n() + lambda))

   # Calculate the average by user
   b_u <- edx_dat %>%
      left_join(b_i, by='movieId') %>%
      group_by(userId) %>%
      summarize(b_u = sum(rating - b_i - mu_hat) / (n() + lambda))

   # Compute the predicted ratings on validation dataset
   predicted_ratings <- validation_dat %>%
      left_join(b_i, by='movieId') %>%
      left_join(b_u, by='userId') %>%
      mutate(pred = mu_hat + b_i + b_u) %>%
      pull(pred)

   # Predict the RMSE on the validation set
   return(RMSE(validation_dat$rating, predicted_ratings))
})

# Get the lambda value that minimize the RMSE
min_lambda <- lambdas[which.min(rmses)]

# Predict the RMSE on the validation set
rmse_regularized_movie_user_model <- min(rmses)

# Adding the results to the results dataset
results <- results %>% add_row(model="Regularized Movie+User Based
Model", RMSE=rmse_regularized_movie_user_model)
print.data.frame(results)


# # --- 7 Regularized Movie User Genres Model ---
# set.seed(1)
# lambdas <- seq(0, 10, 0.1)
#
# # Compute the predicted ratings on validation dataset using different
values of lambda
# rmses <- sapply(lambdas, function(lambda) {
#
#    # Calculate the average by movie
#    b_i <- edx_dat %>%
```

```
#        group_by(movieId) %>%
#        summarize(b_i = sum(rating - mu_hat) / (n() + lambda))
#
#     # Calculate the average by user
#     b_u <- edx_dat %>%
#        left_join(b_i, by='movieId') %>%
#        group_by(userId) %>%
#        summarize(b_u = sum(rating - b_i - mu_hat) / (n() + lambda))
#     # Calculate genre bias
#      b_g <- edx_dat %>%
#        left_join(b_i, by='movieId') %>%
#        left_join(b_u, by='userId') %>%
#        group_by(genres) %>%
#        summarize(b_g = sum(rating - b_i - mu_hat - b_u) / (n() +
lambda))
#
#     # Compute the predicted ratings on validation dataset
#     predicted_ratings <- validation_dat s%>%
#        left_join(b_i, by='movieId') %>%
#        left_join(b_u, by='userId') %>%
#        left_join(b_g, by='genres') %>%
#        mutate(pred = mu_hat + b_i + b_u + b_g) %>%
#        pull(pred)
#
#     # Predict the RMSE on the validation set
#     return(RMSE(validation_dat$rating, predicted_ratings))
# })
#
# # Get the lambda value that minimize the RMSE
# min_lambda <- lambdas[which.min(rmses)]
#
# # Predict the RMSE on the validation set
# rmse_regularized_movie_user_genre_model <- min(rmses)
#
# # Adding the results to the results dataset
# results <- results %>% add_row(model="Regularized Movie+User+Genre
Based Model", RMSE=rmse_regularized_movie_user_genre_model)
# print.data.frame(results)
```