MovieLens Rating Prediction

HarvardX PH125.9x Data Science: Capstone Project

Farah Fauzia

October, 2020

Contents

Abstract

The aim of this project was to develop algorithm model to predict movie ratings for recommendation system. A 10 Million version of MovieLens dataset was used and several algorithm candidates were developed based on normalization of global effect and regularization approach. Model performances were evaluated based on root mean squared error (RMSE) loss function, and final model of regularized after removing movie, user, and genre-specific effects was validated resulting RMSE value of 0.8648196.

1 Introduction

This report is compiled as part of the Capstone course assignment of Data Science Professional Certificate program from HarvardX (1). This project is based on the developed machine learning algorithm for movie recommendation system that win Netflix challenge, which has been used as main example from previous Machine Learning course.

This report starts with general idea of the project, objectives summary, and key steps that were performed. Next, dataset exploration and visualization are carried out through explanatory data analysis. After that, the modeling approach is explained in model development, testing and analysis. Then, the modeling results and performance is discussed. Finally, some concluding remarks will end this report. All code used in this project is attached in the Appendix.

1.1 Problem Overview

Recommendation systems use ratings that users have given to make specific recommendation (2), which leads to development of machine learning algorithm for the wide-range application of product recommendation system. This project focus on the movie recommendation topic which based on the Netflix Prize challenge. Netflix has been using recommendation system to predict how many stars a user will give a specific movie. The Netflix Prize was an open competition for improving the collaborative filtering algorithm to predict user ratings for films, based on previous ratings without any other information about the users or films which was held on October 2006 (3). In this project, the approach of algorithm development would be based on the strategies used by the winning team, which has been previously introduced in the Data Science: Machine Learning course from HarvardX. The dataset used in this project was a smaller subset of MovieLens dataset from GroupLens research lab (4), which contains 10 million ratings for 10,000 movies by 72,000 users.

1.2 Objectives

The main objective of this project was to develop recommendation system using the given MovieLens dataset input by training algorithms that predict movie ratings from user to be applied outside our control. For evaluation, the obtained predicted rating value would be compared to 'actual' values in the validation dataset by using loss function, which measures the difference between these values. For this assignment, Root Mean Squared Error (RMSE) method for accuracy measurement would be used, with the evaluation criteria of RMSE value lower than 0.86490.

1.3 Key Steps

Several steps that will be carried out to achieve the goals include:

1. Data Preparation: First, necessary R packages would be loaded and the 10 Million version of MovieLens dataset would be downloaded. Next, the dataset would be split into two subsets: a 'train' subset (called `edx`) to train the algorithm and `validation` subset for final validation. The `validation` set would be treated as a whole separate data as if we have no access to it and would only be used for validating the final algorithm.

2. Explanatory Data Analysis: The `edx` subset data would be explored and visualization would be carried out to find important characteristics for developing the model algorithm.

3. Model Development, Testing & Analsis: The insights gained from previous step would be translated into model algorithm which would be tested to find the best model that fulfill the objective's evaluation criteria. The entire development and testing would be based on further subset of `edx` sub dataset. Therefore, the `edx` subset would be further split into both 'train' and 'test' set to evaluate the algorithm candidates throughout the model development.

4. Model Validation & Performance Summary: The final model that fulfill the criteria on the testing dataset would be tested using the `validation` dataset and the final performance would be evaluated.

2 Explanatory Data Analysis

2.1 Data Preparation

10 Million version of MovieLens dataset from GroupLens research lab (3) was prepared after necessary pre-processing. This dataset was then be split into two subsets: `edx` sub-dataset, for the purpose of training the algorithm model, and `validation` sub-dataset for final validation. The `validation` sub-dataset would be treated as a whole separate data as if we have no access to it and would only be used for evaluating the very final algorithm. All data exploration and visualization would be carried out using `edx` sub-dataset. In this preparation step, it was also confirmed that both `edx` and `validation` sub-dataset contains no missing values.

2.2 Data Exploration & Visualization

2.2.1 Initial Exploration

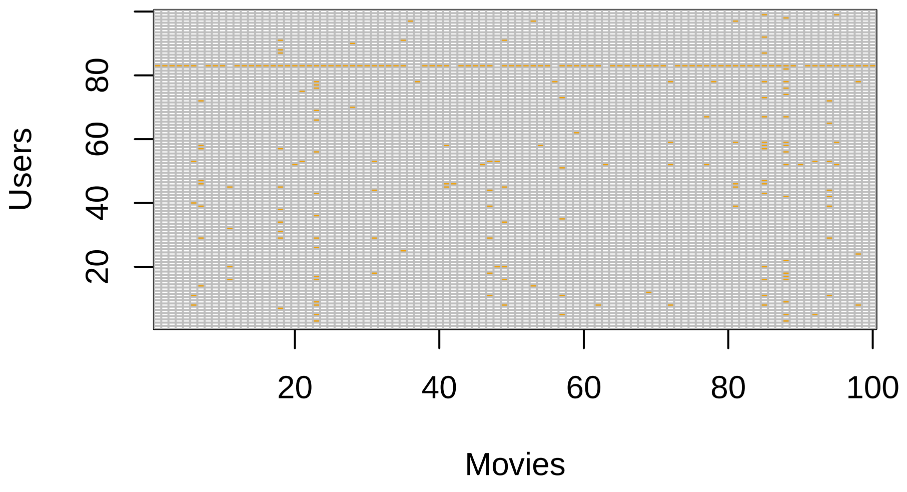
This table summarizes the first 6 lines of `edx` sub-dataset:

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	292	5	838983421	Outbreak (1995)	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
1	355	5	838984474	Flintstones, The (1994)	Children|Comedy|Fantasy

The `edx` sub-dataset consists of approximately 9 million observation with 6 column, with 10,677 different movies, 69,878 unique users, and 797 unique genres combinations:

```
## Rows: 9,000,055
## Columns: 6
## $ userId    <int> 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, ...
## $ movieId   <dbl> 122, 185, 292, 316, 329, 355, 356, 362, 364, 370, 377, 42...
## $ rating    <dbl> 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, ...
## $ timestamp <int> 838985046, 838983525, 838983421, 838983392, 838983392, 83...
## $ title     <chr> "Boomerang (1992)", "Net, The (1995)", "Outbreak (1995)",...
## $ genres    <chr> "Comedy|Romance", "Action|Crime|Thriller", "Action|Drama|...
```

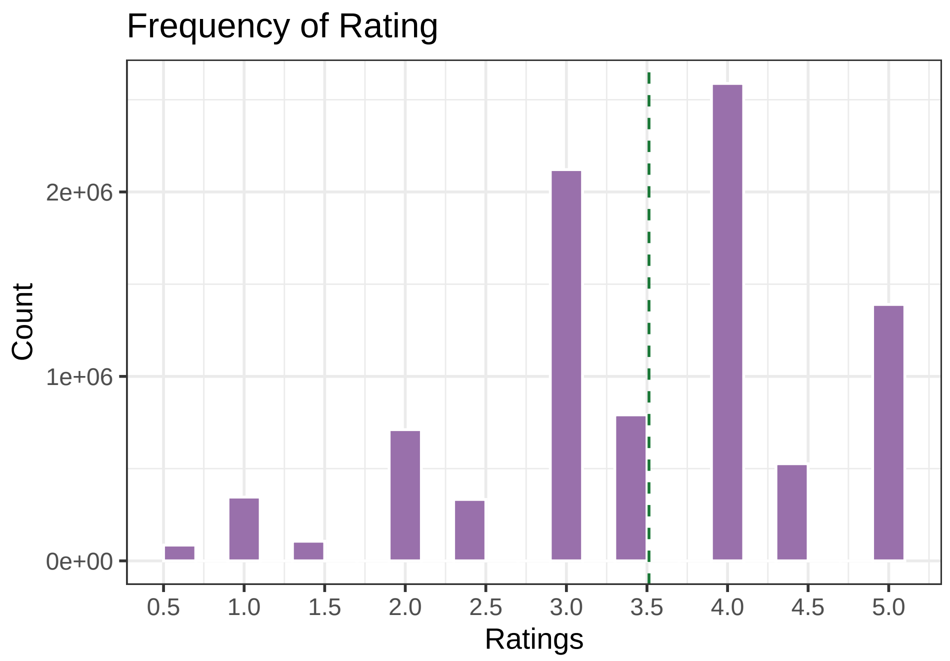
This sub-dataset is very sparse, with about 98.79% missing observation when assuming each user would rate every movie. This could be visualized through generating random samples of 100 users and 100 movies on the following figure:



This project can be seen as building model algorithm for filling these missing observation with appropriate predicted ratings.

2.2.2 "Rating"

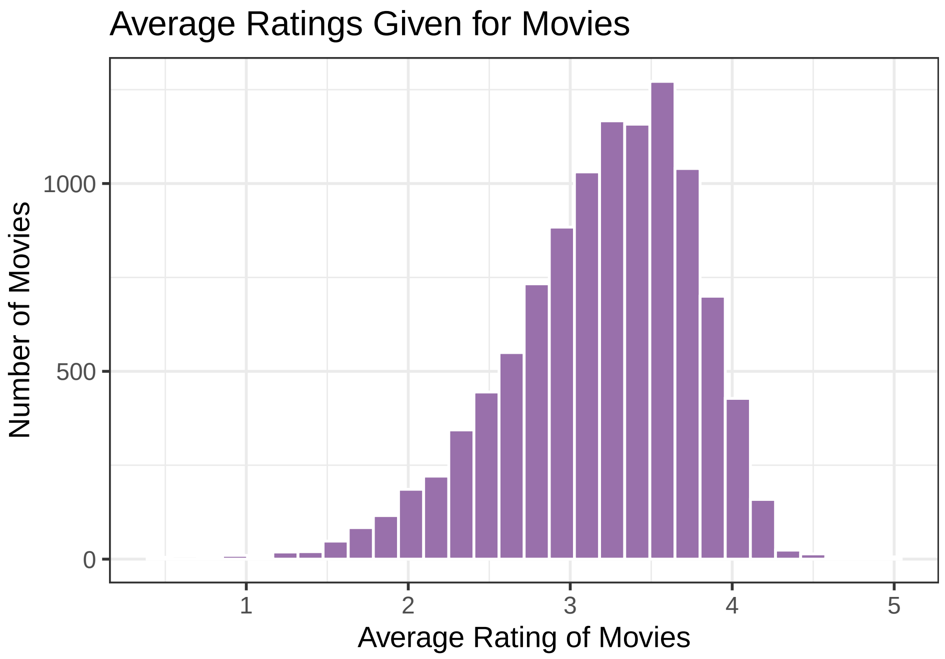
`rating` column contains the value of rating given by certain user for certain movie, which made on 5-star scale with half-star increments (higher rating indicates better evaluation). Average rating in this sub-dataset was 3.5, while the frequency of rating could be seen in the following figure:



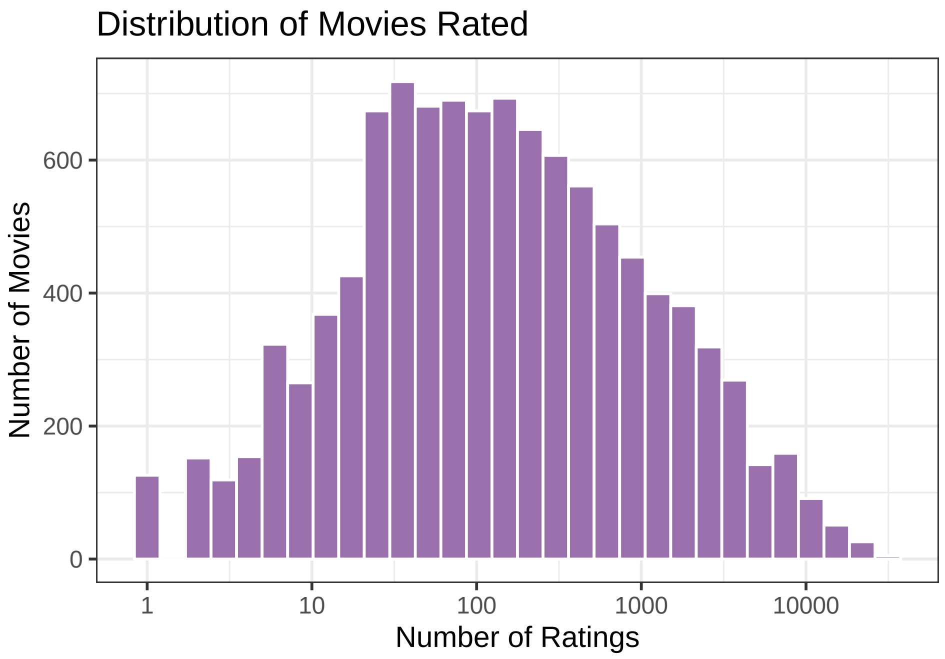
From the above plot, it can be seen that whole-star ratings are more common than half-star ratings, with 4 star is the most common. Higher frequency of high rating indicates the preference of user, where a user may tend to give rating only for the movies they prefer.

2.2.3 "Movies"

`movieId` column contains unique identification number for each movie that has been rated by certain user. `edx` sub-dataset contains 10,677 unique movie title. Average ratings given for movies can be seen in the following figure:



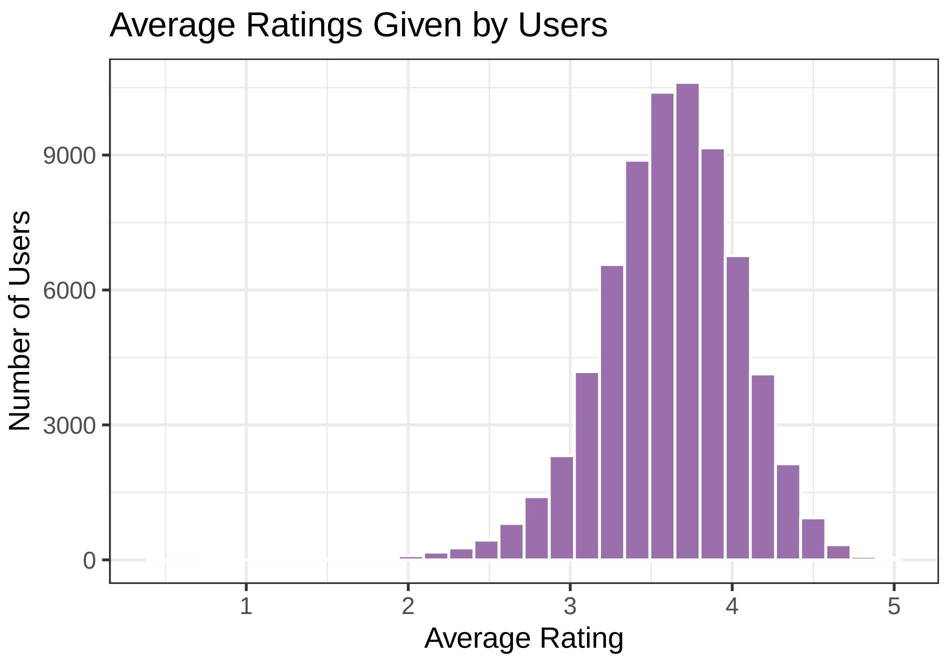
From the above plot, it can be seen that some movies are rated higher than others, which indicating the presence general effect of specific movies. Therefore, it is worth to consider this as movie-specific bias, where different movies are rated differently. Next, distribution of movies rated can be seen in the following figure:



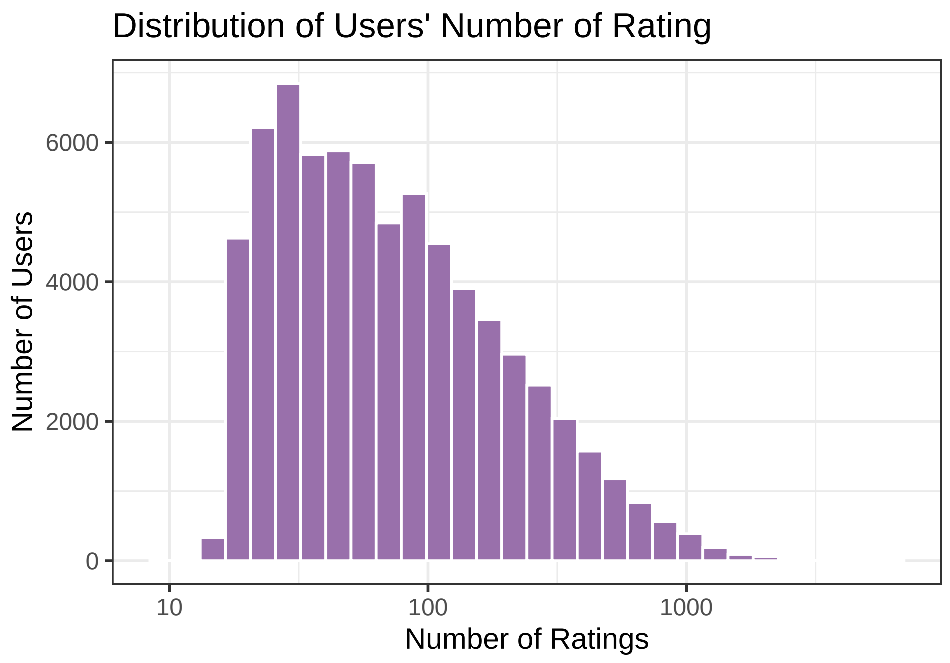
From the above plot, it is indicated that there is variation of rating distribution for movies (some movies get rated more often than others). There are even some movies which have very few ratings, which can influence the model. Therefore, it is necessary to apply the regularization when considering this model. Regularization allow penalization of large estimates which formed by relatively small sample sizes.

2.2.4 "Users"

`userId` column contains unique identification number for each user that have given rating for certain movies. `edx` sub-dataset contains 69,878 unique user. Average ratings given by users can be seen in the following figure:



From the above plot, it can be seen that some users tend to give higher or lower rating than average users, which indicating the presence of general effect of specific user. Therefore, it is worth to consider this as user-specific bias, where different users are giving rating differently. Next, distribution of user' number of rating can be seen in the following figure:



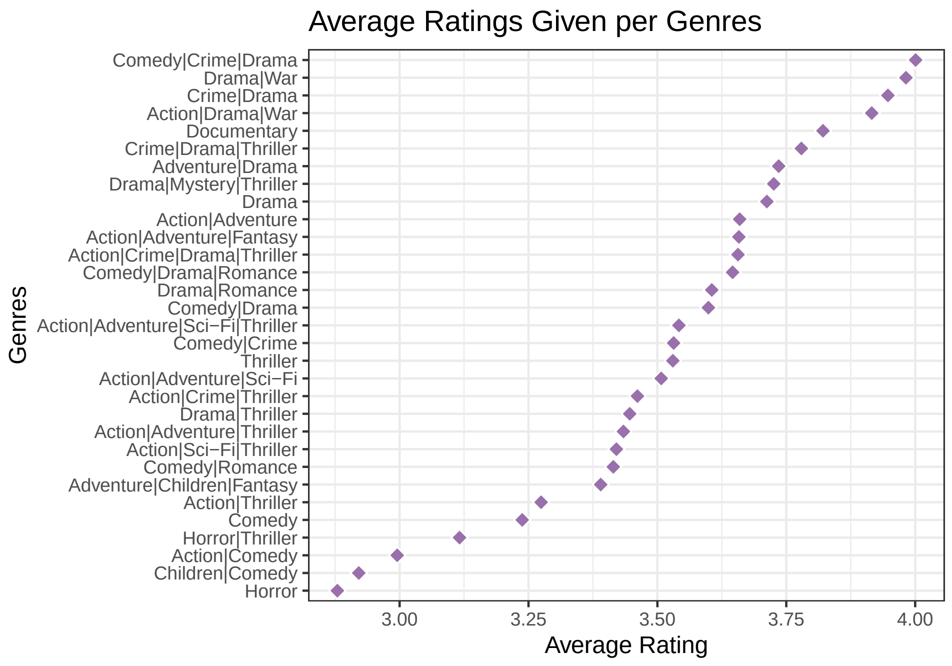
From the above plot, it was indicated that some users are generally more active than others at giving rating for movies. Therefore it is also necessary to apply regularization approach for user penalty when considering this model.

2.2.5 "Genres"

`genres` column contains a list of genre, pipe-separated, of each movies being rated. `edx` sub-dataset contains 797 unique genre combinations. These combinations were consist of 20 separate genres:

```
##  [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)"
```

Average ratings given per genre combination (for genres which have more than total 50,000 rating) can be seen in the following figure:



From the above plot, it is indicated that average rating varies up to more than 1 star difference between certain genre combinations. Therefore, it may worth to consider this as genre-specific bias, where different genre combination are rated differently.

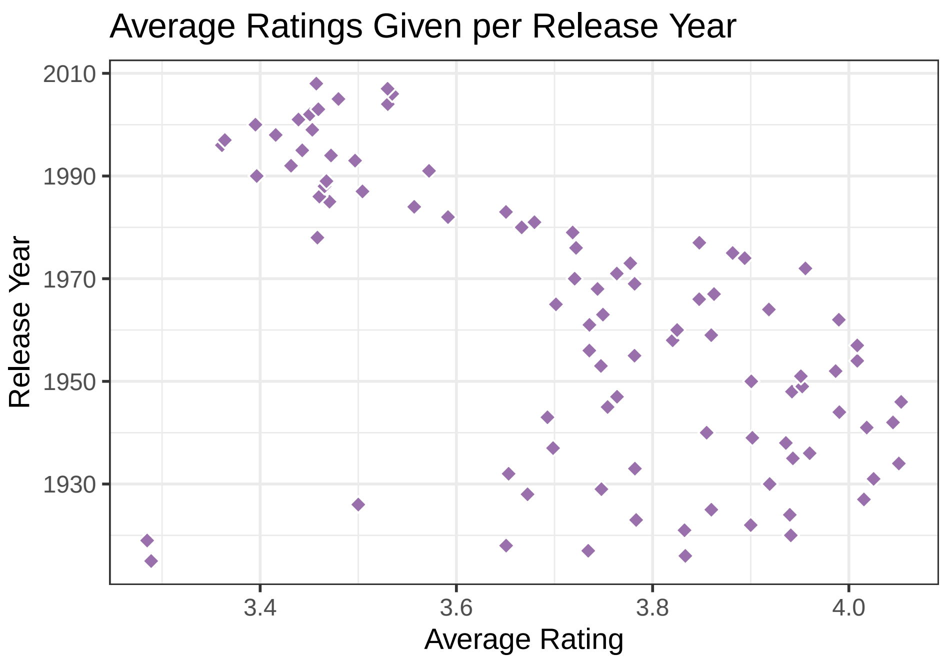
While it is possible to extract the genre for each movies, it is not appropriate to multiply the same observation just by differentiating the listed genre. Therefore, when considering genre in model algorithm later, it will be incorporated as the combination.

2.2.6 "Year"

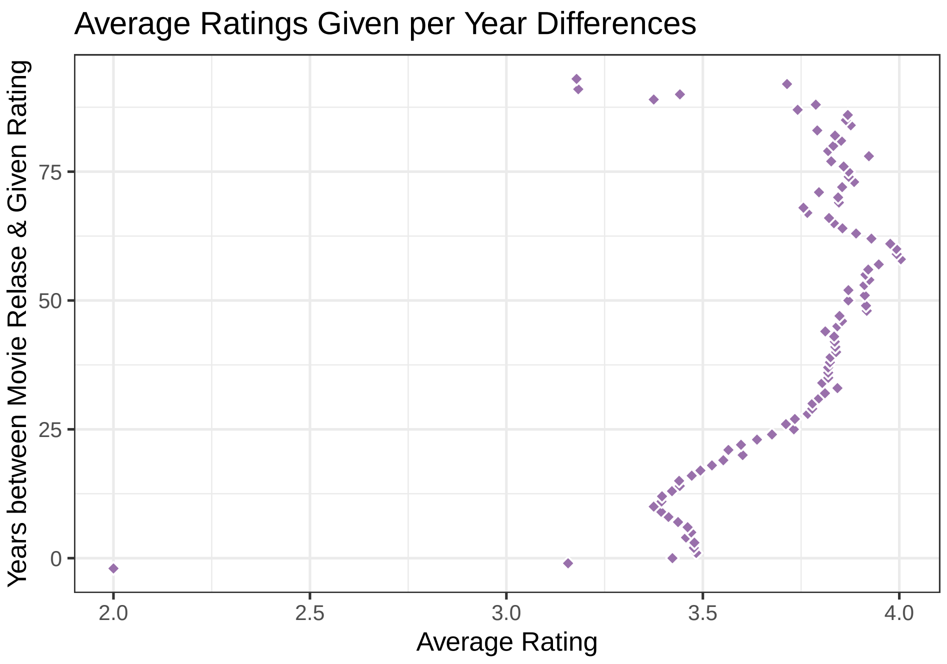
`title` column also contains the release year of each movie, while `timestamp` column contains the time when certain movie get rated by certain user. To explore this time variable, the release year of the movie was extracted into separate column called `year`, while the `timestamp` was converted into date & time format. A new column called `year_diff` was generated, which indicates the year difference between release year and the year when the rating was given. The preview of `edx` sub-dataset after processing can be seen as follow:

userId	movieId	rating	timestamp	title	genres	year	year_diff
1	122	5	1996-08-02 11:24:06	Boomerang (1992)	Comedy|Romance	1992	4
1	185	5	1996-08-02 10:58:45	Net, The (1995)	Action|Crime|Thriller	1995	1
1	292	5	1996-08-02 10:57:01	Outbreak (1995)	Action|Drama|Sci-Fi|Thriller	1995	1
1	316	5	1996-08-02 10:56:32	Stargate (1994)	Action|Adventure|Sci-Fi	1994	2
1	329	5	1996-08-02 10:56:32	Star Trek: Generations (1994)	Action|Adventure|Drama|Sci-Fi	1994	2
1	355	5	1996-08-02 11:14:34	Flintstones, The (1994)	Children|Comedy|Fantasy	1994	2

Average ratings given per movies from certain release year can be seen in the following figure:



From the above plot, it is indicated that movies released before 1980 tends to get rated slightly higher than recent movies. As additional confirmation, the average ratings given per movies from certain year difference value can be seen in the following plot:



Above plot indicates that while it is true that older movies (larger year difference value between release year and rating given) have higher rating, the trend is non-linear and the difference is rather slight (mostly differ by around 0.5 stars). Based on this result, year parameters would not be considered in the model algorithm.

3 Model Development, Testing & Analysis

3.1 Sub-dataset Preparation

The entire model algorithm development and testing would be carried out on `edx` sub-dataset. Therefore, it was necessary to further split this sub-dataset into 'train' set (called `edx_train`) and 'test' set (called `edx_test`) for evaluating the model algorithm candidates throughout the development process. The splitting ratio would mimic the original MovieLens dataset splitting, where 90% of `edx` sub-dataset will be used as train set and the remaining 10% for test set.

3.2 RMSE as Evaluation Parameter

Basic evaluation of machine learning algorithms is comparing the obtained predicted value with the actual outcome, which can be measured using loss function or the difference between these values. In the Netflix challenge, the winner was decided based on the residual mean squared error (RMSE) as loss function, where the interpretation of its value is similar to standard deviation (meaning that if the number is larger than 1, the typical error is larger than one star, which is not a good result) (2). In this report, RMSE was used as sole evaluation parameter, with specific objective of RMSE lower than 0.86490.

RMSE can be defined by the formula:

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

Where Y is the rating for movie i by user u, $\hat{Y}$ is the predicted rating, and N is the number of user/movie combinations and the sum occurring over all the combination (2). Since this evaluation would be carried out throughout the model development, a function to compute RMSE for vectors of ratings and their corresponding predictors would be defined first:

```
RMSE <- function(predicted_ratings, true_ratings){
  sqrt(mean((predicted_ratings - true_ratings)^2, na.rm = TRUE))  }
```

3.3 Normalization of Global Effects Approach

The basic approach used in this report was by normalizing the global effects, or removing several predictors biases. While simple, this approach was turned to be important and useful enough as the strategy used by Netflix Prize winners (5). In other words, this approach would predict specific interaction between certain user and movie rating after removing the effect of these predictor biases.

3.3.1 Baseline Model

In this simplest model, the approach is to assume that all users will give the same rating to all movies regardless of other effects and the movie differences is explained by random variation, which can be defined as:

$$\hat{Y}_{u,i} = \mu + \epsilon_{i,u}$$

Where $\hat{Y}$ is the predicted rating, μ is the 'true' rating for all movies), and $\epsilon_{i,u}$ is the independent error distribution (2). For this base estimation, to minimize the RMSE, all unknown ratings could be predicted by using the mean of all ratings on the observed data.

model	RMSE
Average Rating	1.060054

From this baseline model, additional predictors effect that were gained from previous data exploration and visualization would be applied to improve the model and lower the RMSE. The term of bias, b, for each predictors effect would be introduced. For the algorithm, while the linear model can be used, it is not recommended to apply this model for such big dataset. Instead, the least square estimates of the b could be carried out manually by averaging the observed rating $Y_(u,i)$ and the mean $\hat{\mu}$ for each movie i.

3.3.2 Movie-specific Effect Model

It has been explored that there is variation rating distribution for different movies, or some movies are generally rated higher or lower than others. In this approach, the movie bias (b_i) would be removed for each movie i, and this could be incorporated to the previous model by:

$$\hat{Y}_{u,i} = \mu + b_i + \epsilon_{i,u}$$

Where the least square estimate of movie-specific effect $\hat{b}_i$ can be defined as the average of $Y_(u,i)$ - $\hat{\mu}$ for each movie i (2). The prediction was improved greatly after considering this effect:

model	RMSE
Average + Movie Effect	0.9429615

3.3.3 User-specific Effect Model

It has also been explored that different users would have different rating distribution, some users may tend to rate movies lower or higher than the average user. Therefore, it was necessary to remove the specific bias for each user u (b_u), which could be incorporated to the previous model by:

$$\hat{Y}_{u,i} = \mu + b_i + b_u + \epsilon_{i,u}$$

Where the least square estimate of user-specific effect $\hat{b}_u$ can be defined as the average of $Y_(u,i)$ - $\hat{\mu}$ - $\hat{b}_i$ for each movie i (2). The prediction after considering this additional effect was greatly improved:

model	RMSE
Average + Movie + User Effects	0.8646843

3.3.4 Genre-specific Effect Model

The model could also be further improved by considering genre effect, as it has been explored that some genre combinations tend to be rated lower or higher than others. In this approach, each genre combinations would be treated as unique, which was can be considered as similar to the approach of weighing each movie by containing certain genre or not. The specific bias for each genre combination g (b_g)would be removed and could be incorporated to the previous model by:

$$\hat{Y}_{u,i} = \mu + b_i + b_u + b_g + \epsilon_{i,u}$$

Where the least square estimate of genre-specific effect $\hat{b}_g$ can be defined as the average of $Y_(u,i)$ - $\hat{\mu}$ - $\hat{b}_i$ - $\hat{b}_u$ for each movie i. The prediction after considering genre effect only improved RMSE slightly:

model	RMSE
Average + Movie + User + Genre Effects	0.8643241

While this may happened because the model treated each combinations of the genres uniquely, it is not appropriate to consider the genre independently or multiplying the same movies to be treated as different observation per genre. Therefore, the genre effect was incorporated as it is.

Additional baseline predictors such as considering time effect can also be carried out. However from the previous data exploratory and visualization result, the year effect will not be incorporated to this model. Instead, further improvement of the model would be carried out by considering regularization approach.

3.4 Regularization Approach

From previous data exploratory, it has been shown that there are some movies which have very small number of ratings, and some users who rate only very few movies. For these movies and users, the sample is very small, which may lead to larger estimated error. Therefore, it is necessary to introduce regularization, or penalization of large estimates which formed by relatively small sample sizes. In addition, since the objective is to predict the future unknown ratings, it is necessary to be caution to avoid over-fitting the observed data which could also be achieved by regularizing the parameters (6).

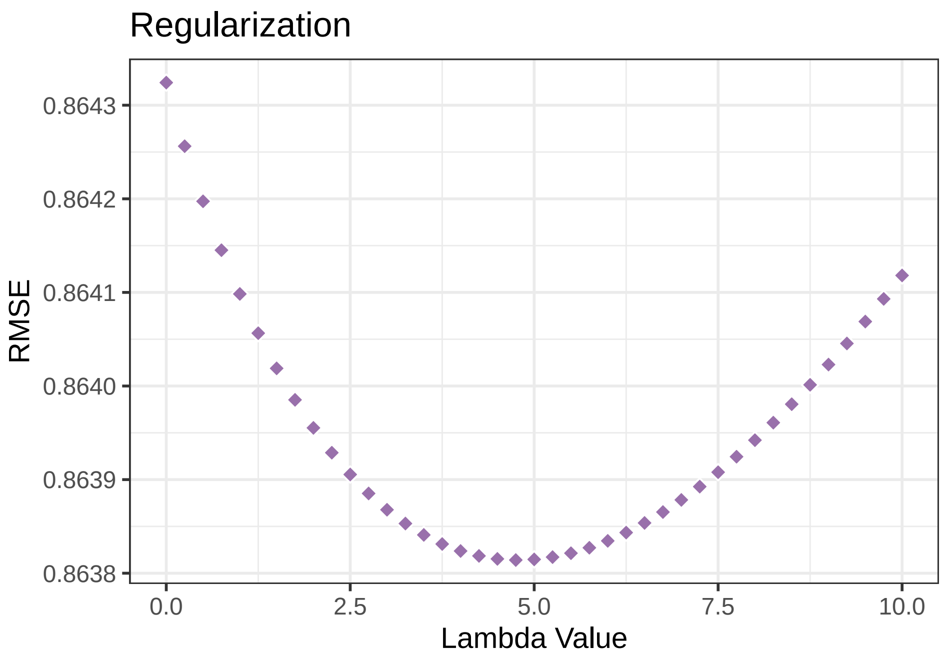
This approach would introduce a factor, λ, that penalizes small sample sizes that can be considered as noisy estimates. Therefore for the previously developed model, the estimated value of movie, user, and genre-specific effect could be improved after adding λ, which could be defined as:

$$\hat{b}_i = \frac{1}{n_i + \lambda} \sum_{u=1}^{n_i} (y_{u,i} - \hat{\mu})$$

$$\hat{b}_u = \frac{1}{n_u + \lambda} \sum_{i=1}^{n_u} (y_{u,i} - \hat{b}_i - \hat{\mu})$$

$$\hat{b}_g = \frac{1}{n_g + \lambda} \sum_{i=1}^{n_g} (y_{u,i} - \hat{b}_i - \hat{b}_u - \hat{\mu})$$

The tuning parameter λ would be estimated to find the value that minimize the RMSE, which would shrinks these $\hat{b}_i$, $\hat{b}_u$, and $\hat{b}_g$ in the previous model, in case of small number of rating samples. These sequences values of λ were tested and incorporated into previous model which considered the removal of average, movie, user, and genre-specific effect, and the RMSE resulted from each λ can be visualized as follow:



The obtained optimal λ value that give the lowest RMSE was 4.75. Thus, using this tuned parameters from regularization, the rating predictions could be calculated and the final RMSE for the regularized model could be obtained:

model	RMSE
Regularized Average + Movie + User + Genre Effect	0.8641357

This further lower the previous RMSE from non-regularized model and already reached the objective of lower than 0.86490. This regularized model was selected as the final model and was subjected into final validation.

4 Results

The final algorithm model based on regularization after removing movie, user, and genre-specific effect was selected, which can be expressed as:

$$\hat{Y}_{u,i} = \mu + b_i + b_u + b_g + \epsilon_{i,u}$$

This final model was then subjected for final validation, which would be trained using the original `edx` sub-dataset and be tested using `validation` sub-dataset.

The final validated RMSE result was 0.8648196. Although this value is higher than the predicted RMSE during development, it is still achieve the assignment objective of being lower than 0.86490.

The following table summarize the RMSE improvement throughout the algorithm model development and validation of final model:

model	RMSE
Average Rating	1.0600537
Average + Movie Effect	0.9429615
Average + Movie + User Effects	0.8646843
Average + Movie + User + Genre Effects	0.8643241
Regularized Average + Movie + User + Genre Effect	0.8641357
Final Regularization (Validation)	0.8648196

Using the validated final model, the predicted best and worst movies can be shown. The following table show best 5 movies:

title
Usual Suspects, The (1995)
Shawshank Redemption, The (1994)
Shawshank Redemption, The (1994)
Shawshank Redemption, The (1994)
Eternal Sunshine of the Spotless Mind (2004)

While the worst 5 movies can be seen as follows:

title
Battlefield Earth (2000)
Police Academy 4: Citizens on Patrol (1987)
Karate Kid Part III, The (1989)

title
Pok<U+FF83><U+FF69>mon Heroes (2003)
Turbo: A Power Rangers Movie (1997)

5 Conclusion

Several algorithm models to predict movie ratings based on MovieLens dataset were developed based on normalization of global effect and regularization approach. Final model of regularized after removing movie, user, and genre-specific effects resulted in RMSE value of 0.8648196 on validation test, which successfully achieve the objective of RMSE value lower than 0.86490.

This project was carried out under limited computational performance, and therefore could only produced limited data exploration/visualization and simple modeling approaches. Global bias effect considered in this work also only involved removing three bias predictors: movie, user, and genre-specific.

Future improvement may include: 1) adding several predictors such as time bias (i.e. number of days since a user give rating or number of days since the movie first rated by anyone), user-time bias (i.e. user may tend to change their baseline ratings over time, so also including user bias in time function), and frequencies (i.e. the number of ratings a user gave on specific day); and 2) using more sophisticated approach, such as: Neighborhood Models, Matrix Factorization, or Ensemble Methods (5, 6). Another perspective of improvement is including user's gender bias, as there were indication of gender stereotypes about movie preferences (7), which may be useful consideration when building recommendation system for possibly similar dataset which include gender information.

6 References

(1) https://www.edx.org/professional-certificate/harvardx-data-science
(2) Irizarry RA. Introduction to Data Science - Data Analysis and Prediction Algorithms with R. Available at: https://rafalab.github.io/dsbook/large-datasets.html#recommendation-systems (Accessed on 2020 Sept 29)
(3) https://en.wikipedia.org/wiki/Netflix_Prize
(4) https://grouplens.org/datasets/movielens/latest/
(5) Chen E. "Winning the Netflix Prize: A Summary". Available at: http://blog.echen.me/2011/10/24/winning-the-netflix-prize-a-summary/ (Accessed on 2020 Sept 29)
(6) Koren Y. The BellKor Solution to the Netflix Grand Price. Published 2009 Aug. Available at: http://www.netflixprize.com/assets/GrandPrize2009_BPC_BellKor.pdf
(7) Wühr P, Lange BP, Schwarz S. Tears or Fears? Comparing Gender Stereotypes about Movie Preferences to Actual Preferences. Front Psychol. 2017;8:428. Published 2017 Mar 24. doi:10.3389/fpsyg.2017.00428

7 Appendix

7.1 Environment

```
## R version 4.0.2 (2020-06-22)
## Platform: x86_64-w64-mingw32/x64 (64-bit)
## Running under: Windows 10 x64 (build 18363)
##
## Matrix products: default
##
## Random number generation:
##  RNG:     Mersenne-Twister
##  Normal:  Inversion
##  Sample:  Rounding
##
## locale:
## [1] LC_COLLATE=English_United States.1252
## [2] LC_CTYPE=English_United States.1252
## [3] LC_MONETARY=English_United States.1252
## [4] LC_NUMERIC=C
## [5] LC_TIME=English_United States.1252
## system code page: 932
##
## attached base packages:
## [1] stats     graphics  grDevices utils     datasets  methods   base
##
## other attached packages:
##  [1] lubridate_1.7.9   scales_1.1.1      data.table_1.13.0 caret_6.0-86
##  [5] lattice_0.20-41   forcats_0.5.0     stringr_1.4.0     dplyr_1.0.2
##  [9] purrr_0.3.4       readr_1.3.1       tidyr_1.1.2       tibble_3.0.3
## [13] ggplot2_3.3.2     tidyverse_1.3.0
##
## loaded via a namespace (and not attached):
##  [1] httr_1.4.2           jsonlite_1.7.1       splines_4.0.2
##  [4] foreach_1.5.0        prodlim_2019.11.13   modelr_0.1.8
##  [7] assertthat_0.2.1     highr_0.8            stats4_4.0.2
## [10] blob_1.2.1           cellranger_1.1.0     yaml_2.2.1
## [13] ipred_0.9-9          pillar_1.4.6         backports_1.1.10
## [16] glue_1.4.2           pROC_1.16.2          digest_0.6.25
## [19] rvest_0.3.6          colorspace_1.4-1     recipes_0.1.13
## [22] htmltools_0.5.0      Matrix_1.2-18        plyr_1.8.6
## [25] timeDate_3043.102    pkgconfig_2.0.3      broom_0.7.1
## [28] haven_2.3.1          gower_0.2.2          lava_1.6.8
## [31] farver_2.0.3         generics_0.0.2       ellipsis_0.3.1
## [34] withr_2.3.0          nnet_7.3-14          cli_2.0.2
## [37] survival_3.1-12      magrittr_1.5         crayon_1.3.4
## [40] readxl_1.3.1         evaluate_0.14        fs_1.5.0
## [43] fansi_0.4.1          nlme_3.1-148         MASS_7.3-51.6
## [46] xml2_1.3.2           class_7.3-17         tools_4.0.2
## [49] hms_0.5.3            lifecycle_0.2.0      munsell_0.5.0
## [52] reprex_0.3.0         compiler_4.0.2       rlang_0.4.7
## [55] grid_4.0.2           iterators_1.0.12     rstudioapi_0.11
## [58] labeling_0.3         rmarkdown_2.4        gtable_0.3.0
## [61] ModelMetrics_1.2.2.2 codetools_0.2-16     DBI_1.1.0
## [64] reshape2_1.4.4       R6_2.4.1             knitr_1.30
## [67] utf8_1.1.4           stringi_1.5.3        Rcpp_1.0.5
## [70] vctrs_0.3.4          rpart_4.1-15         dbplyr_1.4.4
## [73] tidyselect_1.1.0     xfun_0.18
```

7.2 Code Used in this Report

```
# ------------------------------------------------------------------------------
#
# 1. DATA PREPARATION
#
# ------------------------------------------------------------------------------
# Install / load necessary packages
# ------------------------------------------------------------------------------
if(!require(tidyverse)) install.packages("tidyverse", repos = "http://cran.us.r-project.org")
if(!require(caret)) install.packages("caret", repos = "http://cran.us.r-project.org")
if(!require(data.table)) install.packages("data.table", repos = "http://cran.us.r-project.org")
library(tidyverse)
library(caret)
library(data.table)
library(scales)
library(lubridate)
# ------------------------------------------------------------------------------
# Download MovieLens data & tidying (provided by HarvardX)
# ------------------------------------------------------------------------------
dl <- tempfile()
download.file("http://files.grouplens.org/datasets/movielens/ml-10m.zip", dl)

ratings <- fread(text = gsub("::", "\t", readLines(unzip(dl, "ml-10M100K/ratings.dat"))),
                 col.names = c("userId", "movieId", "rating", "timestamp"))

movies <- str_split_fixed(readLines(unzip(dl, "ml-10M100K/movies.dat")), "\\::", 3)
colnames(movies) <- c("movieId", "title", "genres")

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

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

```r
# --------------------------------------------------------------------------------
# Creating `train` ("edx") and "validation" subset dataset (provided by HarvardX)
# --------------------------------------------------------------------------------
# Validation set will be 10% of MovieLens dataset
set.seed(1, sample.kind="Rounding")
test_index <- createDataPartition(y = movielens$rating, times = 1, p = 0.1, list = FALSE)
edx <- movielens[-test_index,]
temp <- movielens[test_index,]

# Make sure userId and movieId in validation set are also in edx set
validation <- temp %>%
  semi_join(edx, by = "movieId") %>%
  semi_join(edx, by = "userId")

# Add rows removed from validation set back into edx set
removed <- anti_join(temp, validation)
edx <- rbind(edx, removed)

rm(dl, ratings, movies, test_index, temp, movielens, removed)
# --------------------------------------------------------------------------------
# Checking any missing data
# --------------------------------------------------------------------------------
any(is.na(edx))
any(is.na(validation))
# --------------------------------------------------------------------------------
#
# 2. EXPLANOTARY DATA ANALYSIS
#
# --------------------------------------------------------------------------------
# 2.1. INITIAL EXPLORATION
# --------------------------------------------------------------------------------
# --------------------------------------------------------------------------------
# Checking general overview of dataset
# --------------------------------------------------------------------------------
# Display first 6 lines of dataset
head(edx)

# Display glimpse of dataset
glimpse(edx)

# Check how many distinct movie, user, and genre
n_movies <- n_distinct(edx$movieId)
n_users <- n_distinct(edx$userId)
n_genres <- n_distinct(edx$genres)
# --------------------------------------------------------------------------------
# Checking how sparse the dataset
# --------------------------------------------------------------------------------
# Calculate the missing observation rate:
mo <- (1-(nrow(edx)/(n_movies*n_users)))*100

# Visualize random samples of 100 users and 100 movies
users <- sample(unique(edx$userId), 100)
edx %>% filter(userId %in% users) %>% select(userId, movieId, rating) %>%
    mutate(rating = 1) %>% spread(movieId, rating) %>%
    select(sample(ncol(.), 100)) %>% as.matrix() %>% t(.) %>%
    image(1:100, 1:100,. , xlab="Movies", ylab="Users")
abline(h=0:100+0.5, v=0:100+0.5, col = "grey")

# ----------------------------------------------------------------------------------
# 2.2. "RATING"
# ----------------------------------------------------------------------------------
# Mean of rating in dataset
rating <- mean(edx$rating)

# Plotting frequency for each rating
edx %>% ggplot(aes(rating)) +
    geom_histogram(bins = 30, binwidth = 0.2, color = "white", fill = "#9970AB") +
    scale_x_continuous(breaks = seq(0.5, 5, 0.5)) +
    geom_vline(xintercept = rating, col = "#1B7837", linetype = "dashed") +
    labs(title = "Frequency of Rating",
    x = "Ratings", y = "Count") + theme_bw()
# ----------------------------------------------------------------------------------
# 2.3. "MOVIES"
# ----------------------------------------------------------------------------------
# Plotting average ratings given for movies
edx %>% group_by(movieId) %>% summarize(avg_i = mean(rating)) %>%
    ggplot(aes(avg_i)) + geom_histogram(bins = 30, color = "white", fill = "#9970AB") +
    labs(title = "Average Ratings Given for Movies",
    x = "Average Rating of Movies", y = "Number of Movies") + theme_bw()

# Plotting distribution of movies being rated
edx %>% count(movieId) %>% ggplot(aes(n)) +
    geom_histogram(bins = 30, color = "white", fill = "#9970AB") +
    scale_x_log10() + labs(title = "Distribution of Movies Rated",
    x = "Number of Ratings", y = "Number of Movies") + theme_bw()
# ----------------------------------------------------------------------------------
# 2.3. "USERS"
# ----------------------------------------------------------------------------------
# Plotting average ratings given by users
edx %>% group_by(userId) %>% summarize(avg_u = mean(rating)) %>%
    ggplot(aes(avg_u)) + geom_histogram(bins = 30, color = "white", fill = "#9970AB") +
    labs(title = "Average Ratings Given by Users",
    x = "Average Rating", y = "Number of Users") + theme_bw()

# Plotting distribution of users' number of rating
edx %>% count(userId) %>% ggplot(aes(n)) +
    geom_histogram(bins = 30, color = "white", fill = "#9970AB") +
    scale_x_log10() + labs(title = "Distribution of Users' Number of Rating",
    x = "Number of Ratings", y = "Number of Users") + theme_bw()
# ----------------------------------------------------------------------------------
# 2.4. "GENRES"
# ----------------------------------------------------------------------------------
# Generating unique genres
unique_genres <- str_extract_all(unique(edx$genres), "[^|]+") %>%
  unlist() %>% unique()
unique_genres
```

```
# Plotting average ratings given per genres
edx %>% group_by(genres) %>% summarize(n=n(), avg_g = mean(rating)) %>%
    filter(n >= 50000) %>% mutate(genres = reorder(genres, avg_g)) %>%
    ggplot(aes(y = genres, x = avg_g)) +
    geom_point(size=2, shape=23, color="white", fill="#9970AB") +
    labs(title = "Average Ratings Given per Genres",
    x = "Average Rating", y = "Genres") + theme_bw(base_size = 9)
# --------------------------------------------------------------------------------
# 2.5. "YEAR"
# --------------------------------------------------------------------------------
# Extracting release year of movie
edx_time <- edx %>%
    mutate(timestamp = as_datetime(timestamp)) %>%
    mutate(year = substring(title, nchar(title) - 6)) %>%
    mutate(year = as.numeric(substring(year, regexpr("\\(", year) + 1,
                                    regexpr("\\)", year) - 1))) %>%
    mutate(year_diff = as.numeric(year(timestamp)) - year)
head(edx_time)

# Plotting average ratings given per release year
edx_time %>% group_by(year) %>% summarize(n=n(), avg_y = mean(rating)) %>%
    ggplot(aes(y = year, x = avg_y)) +
    geom_point(size=2, shape=23, color="white", fill="#9970AB") +
    labs(title = "Average Ratings Given per Release Year",
    x = "Average Rating", y = "Release Year") + theme_bw()

# Plotting average ratings given per year differences
edx_time %>% group_by(year_diff) %>% summarize(n=n(), avg_yd = mean(rating)) %>%
    ggplot(aes(y = year_diff, x = avg_yd)) +
    geom_point(size=1.5, shape=23, color="white", fill="#9970AB") +
    labs(title = "Average Ratings Given per Year Differences",
    x = "Average Rating", y = "Years between Movie Relase & Given Rating") +
    theme_bw(base_size = 10)
# --------------------------------------------------------------------------------
#
# 3. MODEL DEVELOPMENT, TESTING, & ANALYSIS
#
# --------------------------------------------------------------------------------
# 3.1. SUB-DATASET PREPARATION
# --------------------------------------------------------------------------------
# --------------------------------------------------------------------------------
# Creating `train` and `test` subset from "edx" sub dataset
# --------------------------------------------------------------------------------
# The training set will be 90% of "edx" data and the test set will be the remaining 10%
set.seed(1, sample.kind="Rounding")
test_index <- createDataPartition(y = edx$rating, times = 1, p = 0.1, list = FALSE)
edx_train <- edx[-test_index,]
temp2 <- edx[test_index,]

# Make sure userId and movieId in test set are also in train set
edx_test <- temp2 %>%
  semi_join(edx_train, by = "movieId") %>%
  semi_join(edx_train, by = "userId")
```

```r
# Add rows removed from test set back into train set
removed <- anti_join(temp2, edx_test)
edx_train <- rbind(edx_train, removed)

rm(test_index, temp2, removed)
# ----------------------------------------------------------------------------------
# 3.2. RMSE AS EVALUATION PARAMETER
# ----------------------------------------------------------------------------------
# Defining loss function RMSE as evaluation parameter
RMSE <- function(predicted_ratings, true_ratings){
  sqrt(mean((predicted_ratings - true_ratings)^2, na.rm = TRUE))}
# ----------------------------------------------------------------------------------
# 3.3. NORMALIZATION OF GLOBAL EFFECTS APPROACH
# ----------------------------------------------------------------------------------
# ----------------------------------------------------------------------------------
# 3.3.1 (Model 1) Baseline Model
# ----------------------------------------------------------------------------------
# Predict the rating
mu <- mean(edx_train$rating)

# Calculate RMSE for model 1
avg_rmse <- RMSE(mu, edx_test$rating)
rmse_results <- tibble(model = "Average Rating", RMSE = avg_rmse)
rmse_results
# ----------------------------------------------------------------------------------
# 3.3.2 (Model 2) Movie-specific Effect Introduction
# ----------------------------------------------------------------------------------
# Calculate movie-specific effect (b_i)
movie_eff <- edx_train %>%
  group_by(movieId) %>%
  summarize(b_i = mean(rating - mu))

# Predict the rating after including b_i
predicted_ratings <- mu + edx_test %>%
  left_join(movie_eff, by = 'movieId') %>%
  pull(b_i)

# RMSE for Model 2
model_2_rmse <- RMSE(predicted_ratings, edx_test$rating)
rmse_results <- bind_rows(rmse_results,
                          tibble(model = "Average + Movie Effect",
                                 RMSE = model_2_rmse))
rmse_results[2,]
# ----------------------------------------------------------------------------------
# 3.3.3 (Model 3) User-specific Effect Introduction
# ----------------------------------------------------------------------------------
# Calculate user-specific effect (b_u)
user_eff <- edx_train %>%
  left_join(movie_eff, by = 'movieId') %>%
  group_by(userId) %>%
  summarize(b_u = mean(rating - mu - b_i))

# Predict the rating after including b_u
```

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

# RMSE for Model 3
model_3_rmse <- RMSE(predicted_ratings, edx_test$rating)
rmse_results <- bind_rows(rmse_results,
                          tibble(model = "Average + Movie + User Effects",
                                 RMSE = model_3_rmse))
rmse_results[3,]
# --------------------------------------------------------------------------------
# 3.3.4 (Model 4) Genre-specific Effect Introduction
# --------------------------------------------------------------------------------
# Calculate genre-specific effect (b_g)
genres_eff <- edx_train %>%
  left_join(movie_eff, by = 'movieId') %>%
  left_join(user_eff, by = 'userId') %>%
  group_by(genres) %>%
  summarize(b_g = mean(rating - mu - b_i - b_u))

# Predict the rating after including b_g
predicted_ratings <- edx_test %>%
  left_join(movie_eff, by = 'movieId') %>%
  left_join(user_eff, by = 'userId') %>%
  left_join(genres_eff, by = c('genres')) %>%
  mutate(pred = mu + b_i + b_u + b_g) %>%
  pull(pred)

# RMSE for Model 4
model_4_rmse <- RMSE(predicted_ratings, edx_test$rating)
rmse_results <- bind_rows(rmse_results,
                          tibble(model = "Average + Movie + User + Genre Effects",
                                 RMSE = model_4_rmse))
rmse_results[4,]
# --------------------------------------------------------------------------------
# 3.4. REGULARIZATION APPROACH
# --------------------------------------------------------------------------------
# --------------------------------------------------------------------------------
# Choosing tuning parameter lambda
# --------------------------------------------------------------------------------
# Define set of lambdas to be tested
lambdas <- seq(0, 10, 0.25)

# Calculate RMSEs for the defined lambdas
rmses <- sapply(lambdas, function(l){

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

  predicted_ratings <- edx_test %>%
    left_join(b_i, by = "movieId") %>%
    left_join(b_u, by = "userId") %>%
    left_join(b_g, by = "genres") %>%
    mutate(pred = mu + b_i + b_u + b_g) %>% pull(pred)

  return(RMSE(predicted_ratings, edx_test$rating))
})

# Visualize lambda values and RMSE obtained for each of it
tibble(lambda = lambdas, RMSE = rmses) %>% ggplot(aes(lambda, RMSE)) +
   geom_point(size=2, shape=23, color="white", fill="#9970AB") +
   labs(title = "Regularization", x = "Lambda Value", y = "RMSE") + theme_bw()

# Choose lambda value that returns the lowest RMSE
lambda <- lambdas[which.min(rmses)]
# --------------------------------------------------------------------------------
# (Model 5) Regularization Introduction
# --------------------------------------------------------------------------------
# Recalculate the biases, while including best lambda value from regularization
mu <- mean(edx_train$rating)
movie_eff_reg <- edx_train %>% group_by(movieId) %>%
  summarize(b_i = sum(rating - mu)/(n()+lambda))
user_eff_reg <- edx_train %>%
  left_join(movie_eff_reg, by = 'movieId') %>% group_by(userId) %>%
  summarize(b_u = sum(rating - mu - b_i)/(n()+lambda))
genres_eff_reg <- edx_train %>% left_join(movie_eff_reg, by = 'movieId') %>%
  left_join(user_eff_reg, by = 'userId') %>% group_by(genres) %>%
  summarize(b_g = mean(rating - mu - b_i - b_u)/(n()+lambda))

# Predict the rating after introducing regularization
predicted_ratings_reg <- edx_test %>%
  left_join(movie_eff_reg, by = "movieId") %>%
  left_join(user_eff_reg, by = "userId") %>%
  left_join(genres_eff_reg, by = "genres") %>%
  mutate(pred = mu + b_i + b_u + b_g) %>% pull(pred)

# RMSE for Model 5
model_5_rmse <- RMSE(predicted_ratings_reg, edx_test$rating)
rmse_results <- bind_rows(rmse_results,
                          tibble(model = "Regularized Average + Movie + User +
                                 Genre Effect",
                                 RMSE = model_5_rmse ))
rmse_results[5,]
# --------------------------------------------------------------------------------
#
# 4. FINAL VALIDATION & PERFORMANCE SUMMARY
#
```

```r
# -------------------------------------------------------------------------------
# Perform validation on MovieLens dataset
# -------------------------------------------------------------------------------
# Train the final model using "edx" sub-dataset
mu_edx <- mean(edx$rating)

movie_eff_edx <- edx %>% group_by(movieId) %>%
  summarize(b_i = sum(rating - mu_edx)/(n()+lambda))

user_eff_edx <- edx %>%
  left_join(movie_eff_edx, by = 'movieId') %>% group_by(userId) %>%
  summarize(b_u = sum(rating - mu_edx - b_i)/(n()+lambda))

genres_eff_edx <- edx %>% left_join(movie_eff_edx, by = 'movieId') %>%
  left_join(user_eff_edx, by = 'userId') %>% group_by(genres) %>%
  summarize(b_g = mean(rating - mu_edx - b_i - b_u)/(n()+lambda))

# Test the final model using "validation" sub-dataset
valid_predicted_ratings <- validation %>%
  left_join(movie_eff_edx, by = "movieId") %>%
  left_join(user_eff_edx, by = "userId") %>%
  left_join(genres_eff_edx, by = "genres") %>%
  mutate(pred = mu_edx + b_i + b_u + b_g) %>% pull(pred)

# RMSE for Final Validation
model_final <- RMSE(valid_predicted_ratings, validation$rating)
rmse_results <- bind_rows(rmse_results, tibble(model = "Final Regularization
                                              (Validation)", RMSE = model_final))
rmse_results[6,]
# -------------------------------------------------------------------------------
# Show RMSE improvement throughout development & validation
# -------------------------------------------------------------------------------
rmse_results
# -------------------------------------------------------------------------------
# Summarizing performance of final model
# -------------------------------------------------------------------------------
# Show 5 best movies using final model
validation %>%
  left_join(movie_eff_edx, by = "movieId") %>%
  left_join(user_eff_edx, by = "userId") %>%
  left_join(genres_eff_edx, by = "genres") %>%
  mutate(pred = mu_edx + b_i + b_u + b_g) %>%
  arrange(-pred) %>% group_by(title) %>%
  select(title) %>% head(5)
# Show 5 worst movies using final model
validation %>%
  left_join(movie_eff_edx, by = "movieId") %>%
  left_join(user_eff_edx, by = "userId") %>%
  left_join(genres_eff_edx, by = "genres") %>%
  mutate(pred = mu_edx + b_i + b_u + b_g) %>%
  arrange(pred) %>% group_by(title) %>%
  select(title) %>% head(5)
```