edx Capstone : MovieLens

Taisuke Matsuki

13 May 2024

Introduction

This project is related to the MovieLens project in HervardX: PH125.9x Data Science and is a capstone course.

Next, you will prepare and set up a given dataset. An exploratory data analysis will be conducted to develop a machine learning algorithm to predict movie ratings. The results obtained by applying the machine learning algorithm will be defined. Finally, the project will conclude with a conclusion.

The theme of this project is "Movie Recommendation System Algorithm Creation". The given dataset is the "Movie Lens 10M version," which is the background that the edX management has taken into account to lighten the computational load for us.

After the algorithm, i.e., the machine learning model, is created and validated, it is specified that its accuracy is evaluated by RMSE (Root Mean Squared Error).

There are several ranks of evaluation, but the best score RMSE should be less than 0.86490. I will go through some modeling to achieve my best score.

Supplemental information on RMSE

Before I explain RMSE, let me explain MSE (Mean Squared Error). MSE is the mean of the squared difference between the predicted and measured values. Squaring absorbs the pluses and minuses of deviations and at the same time reflects the greater impact of large errors and the smaller impact of small errors.

On the other hand, MSE is not often used because the squared values are on a different scale from the measured and predicted values, and it is difficult to intuitively understand what the magnitude of the value represents.

As an alternative, the RMSE, which is the square root of the MSE, is used. It is one of the most popular accuracy indices for forecasting. It is frequently used in Japanese business practice.

Recommendation Algorithm Creation Process

1.Data Set loading

2.Exploratory data analysis (EDA)

3.Feature engineering as needed

4.Build several models to increase accuracy

5.Accuracy evaluation

Data Set loading

```
##########################################################
# Create edx and final_holdout_test sets
##########################################################

# Note: this process could take a couple of minutes

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

library(tidyverse)
library(caret)

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

options(timeout = 120)

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

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

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

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

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

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

# Final hold-out test set will be 10% of MovieLens data
set.seed(1, sample.kind="Rounding") # if using R 3.6 or later
# set.seed(1) # if using R 3.5 or earlier
test_index <- createDataPartition(y = movielens$rating, times = 1, p = 0.1, list = FALSE)
edx <- movielens[-test_index,]
temp <- movielens[test_index,]

# Make sure userId and movieId in final hold-out test set are also in edx set
final_holdout_test <- temp %>%
  semi_join(edx, by = "movieId") %>%
  semi_join(edx, by = "userId")

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

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

Exploratory data analysis (EDA) & Feature engineering as needed

```
# Loading of necessary libraries
library(scales)

# Exploratory Data Analysis (EDA) is performed to understand the data and determine the direction of
# First, characterize the data set from various angles
head(edx)
```

```
##   userId movieId rating timestamp                         title
## 1      1     122      5 838985046              Boomerang (1992)
## 2      1     185      5 838983525               Net, The (1995)
## 4      1     292      5 838983421               Outbreak (1995)
## 5      1     316      5 838983392               Stargate (1994)
## 6      1     329      5 838983392 Star Trek: Generations (1994)
## 7      1     355      5 838984474       Flintstones, The (1994)
##                          genres
## 1                Comedy|Romance
## 2         Action|Crime|Thriller
## 4  Action|Drama|Sci-Fi|Thriller
## 5       Action|Adventure|Sci-Fi
## 6 Action|Adventure|Drama|Sci-Fi
## 7       Children|Comedy|Fantasy
```

```
str(edx)
```

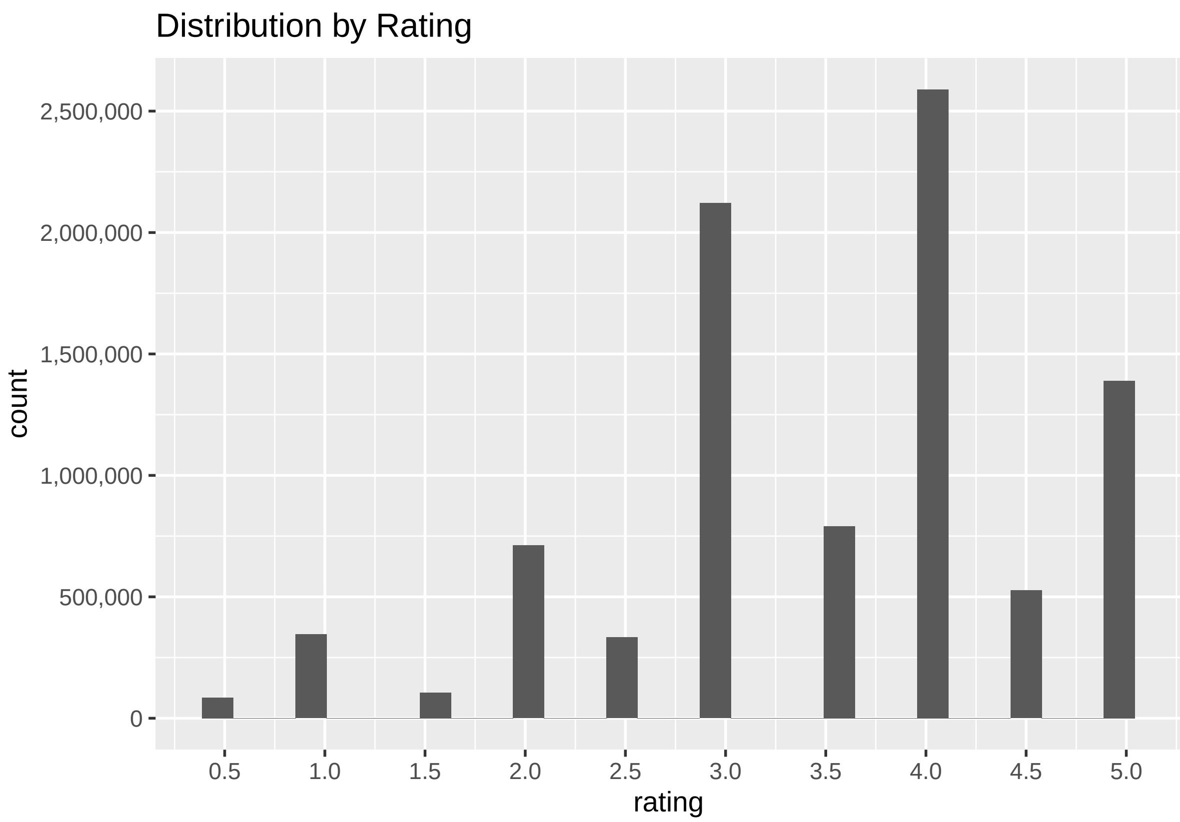
```
## 'data.frame':    9000055 obs. of  6 variables:
##  $ userId   : int  1 1 1 1 1 1 1 1 1 1 ...
##  $ movieId  : int  122 185 292 316 329 355 356 362 364 370 ...
##  $ rating   : num  5 5 5 5 5 5 5 5 5 5 ...
##  $ timestamp: int  838985046 838983525 838983421 838983392 838983392 838984474 838983653 83898488
##  $ title    : chr  "Boomerang (1992)" "Net, The (1995)" "Outbreak (1995)" "Stargate (1994)" ...
##  $ genres   : chr  "Comedy|Romance" "Action|Crime|Thriller" "Action|Drama|Sci-Fi|
Thriller" "Action|Adventure|Sci-Fi" ...
```

```
# Add a year column
edx <- edx %>%
  mutate(Movie_year = as.numeric(str_extract(str_extract(title, "[/(]\\d{4}[/)]$"), regex("\\d{4}"))
```

```r
# See how the whole thing is going.
edx %>%
  summarise(Movies_n = n(),
          Avg_Rating = mean(rating))
```

```
##   Movies_n Avg_Rating
## 1  9000055   3.512465
```

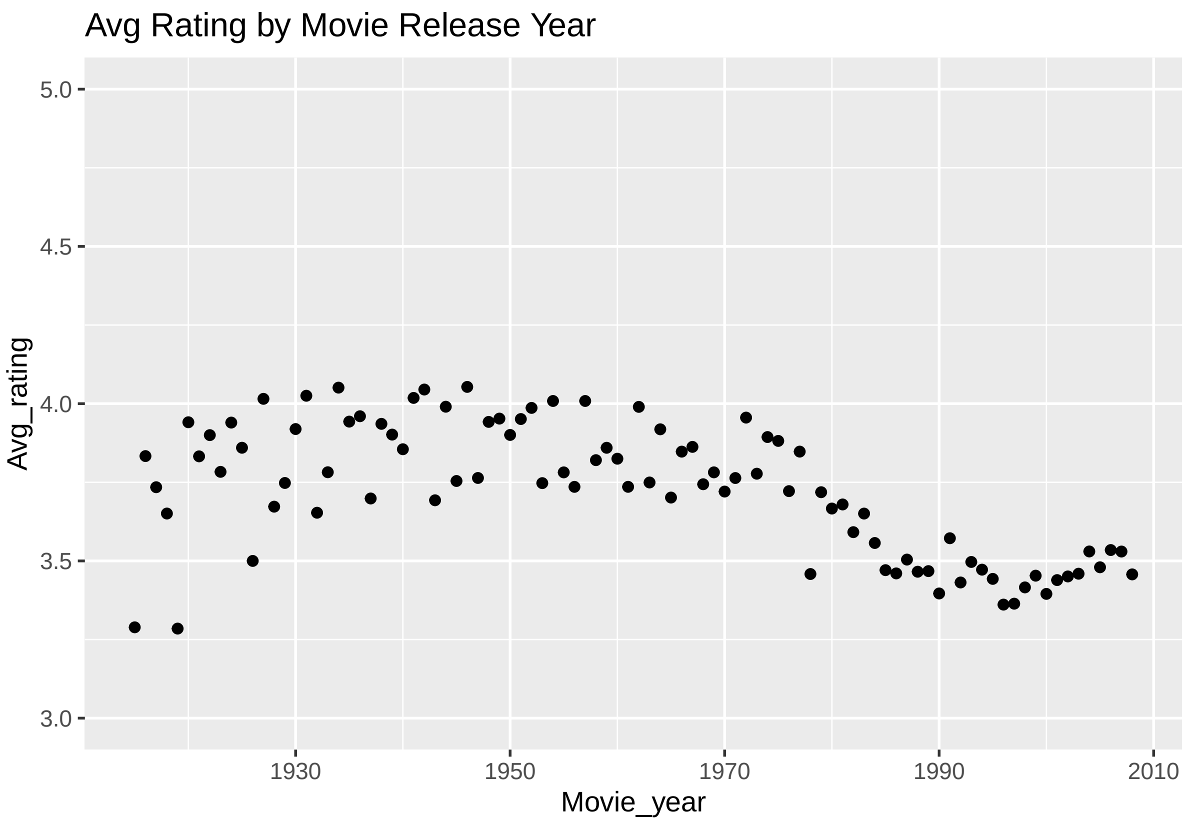
```r
# Check the distribution of ratings
edx %>% ggplot(aes(rating))+
  geom_histogram()+
  scale_x_continuous(breaks = seq(0.0,5.0,0.5))+
  scale_y_continuous(breaks = seq(0,30000000,500000),labels = label_comma())+
  labs(title = "Distribution by Rating")
```




```r
# Check the latest and oldest release years
summary(edx$Movie_year)
```

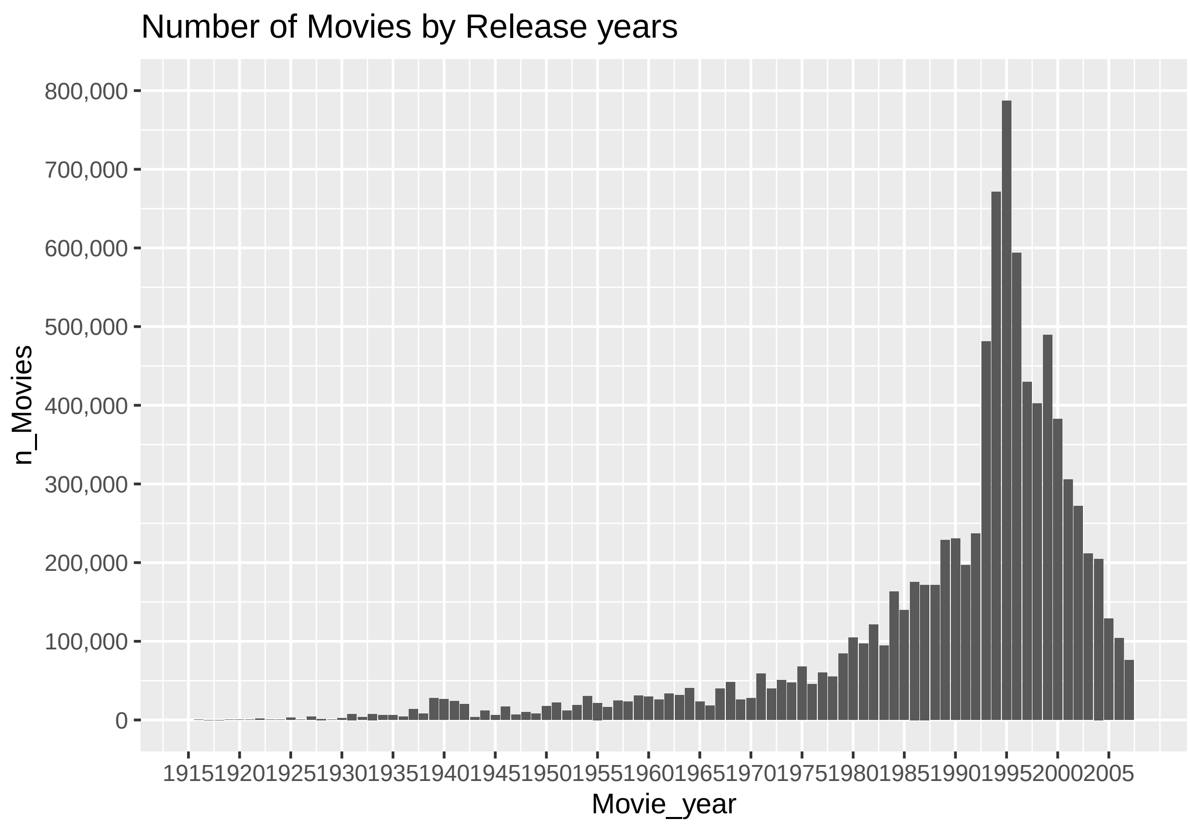
```
##     Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##     1915    1987    1994    1990    1998    2008
```

```r
# Check ratings by release year
edx %>%
  group_by(Movie_year) %>%
  summarise(Avg_rating = mean(rating)) %>%
  ggplot(aes(Movie_year,Avg_rating))+
  geom_point()+
  labs(title = "Avg Rating by Movie Release Year")+
  scale_y_continuous(breaks = seq(3.0,5.0,0.5),limits = c(3.0,5.0))
```




```r
# Check the number of movies by release years
edx %>%
  group_by(Movie_year) %>%
  summarise(n_Movies = n()) %>%
  ggplot(aes(Movie_year,n_Movies))+
  geom_bar(stat = "identity", position = "dodge")+
  labs(title = "Number of Movies by Release years")+
```

```
  scale_y_continuous(breaks = seq(0,800000,100000),limits = c(0,800000),labels = label_comma())+
  scale_x_continuous(breaks = seq(1915,2008,5),limits = c(1915,2008))
```




```
# Check the Unique number of movies
n_distinct(edx$movieId)
```

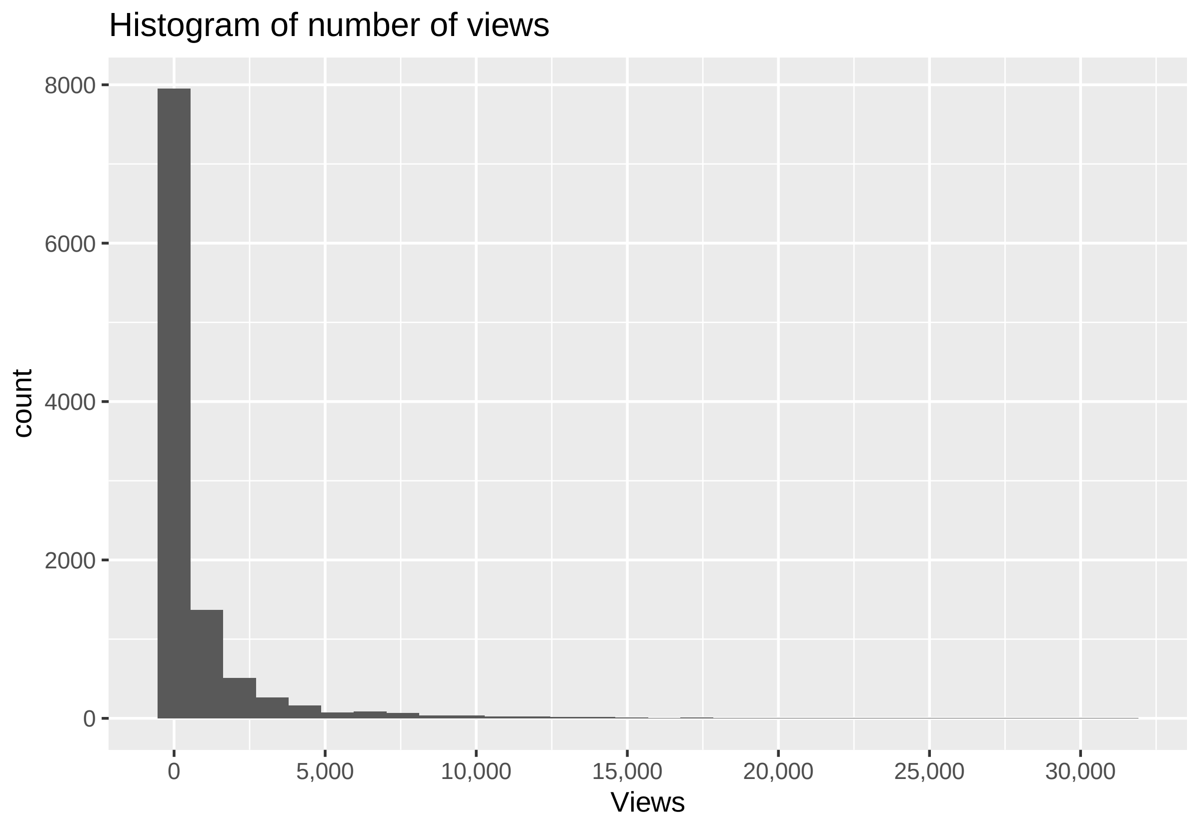
```
## [1] 10677
```

```
# Check the number of views and average rating
edx %>% group_by(movieId,title) %>%
  summarise(Review_times = n(),
            Avg_rating = mean(rating)) %>%
  arrange(desc(Review_times))
```

```
## # A tibble: 10,677 x 4
## # Groups:   movieId [10,677]
##    movieId title                                          Review_times Avg_rating
##      <int> <chr>                                                 <int>      <dbl>
##  1     296 "Pulp Fiction "                                       31362       4.15
```

```
## 2      356 "Forrest Gump "                                     31079        4.01
## 3      593 "Silence of the Lambs, The "                        30382        4.20
## 4      480 "Jurassic Park "                                    29360        3.66
## 5      318 "Shawshank Redemption, The "                        28015        4.46
## 6      110 "Braveheart "                                       26212        4.08
## 7      457 "Fugitive, The "                                    25998        4.01
## 8      589 "Terminator 2: Judgment Day "                       25984        3.93
## 9      260 "Star Wars: Episode IV - A New Hope (a.k.a. ~       25672        4.22
## 10     150 "Apollo 13 "                                        24284        3.89
## # i 10,667 more rows
```

```
# Check the number of viewers by movie
# The following checks confirm that no one film is seen more than once by the same person
edx %>% group_by(movieId,title) %>%
  summarise(Views = n()) %>%
  arrange(desc(Views)) %>%
  ggplot(aes(Views))+
  geom_histogram()+
  scale_x_continuous(breaks = seq(0,32000,5000),labels = label_comma())+
  labs(title = "Histogram of number of views")
```




```
edx %>% group_by(userId) %>%
  summarise(n = n()) %>%
  arrange(desc(n))
```

```
## # A tibble: 69,878 x 2
##    userId     n
##     <int> <int>
##  1  59269  6616
##  2  67385  6360
##  3  14463  4648
##  4  68259  4036
##  5  27468  4023
##  6  19635  3771
##  7   3817  3733
##  8  63134  3371
##  9  58357  3361
## 10  27584  3142
## # i 69,868 more rows
```

```r
edx %>% group_by(userId,movieId) %>%
  summarise(n = n()) %>%
  arrange(desc(n))
```

```
## # A tibble: 9,000,055 x 3
## # Groups:   userId [69,878]
##     userId movieId     n
##      <int>   <int> <int>
##  1       1     122     1
##  2       1     185     1
##  3       1     292     1
##  4       1     316     1
##  5       1     329     1
##  6       1     355     1
##  7       1     356     1
##  8       1     362     1
##  9       1     364     1
## 10       1     370     1
## # i 9,000,045 more rows
```

```r
edx_views_summarise <- edx %>% group_by(movieId,title) %>%
  summarise(Views = n(),
            Avg_rating = mean(rating)) %>%
  arrange(desc(Views)) %>%
  ungroup() %>%
  mutate(Total_Views = sum(Views),
         Proportion = Views / Total_Views)

summary(edx_views_summarise$Views)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##     1.0    30.0   122.0   842.9   565.0 31362.0
```

```r
sd(edx_views_summarise$Views)
```

```
## [1] 2238.481
```

```r
edx_views_summarise
```

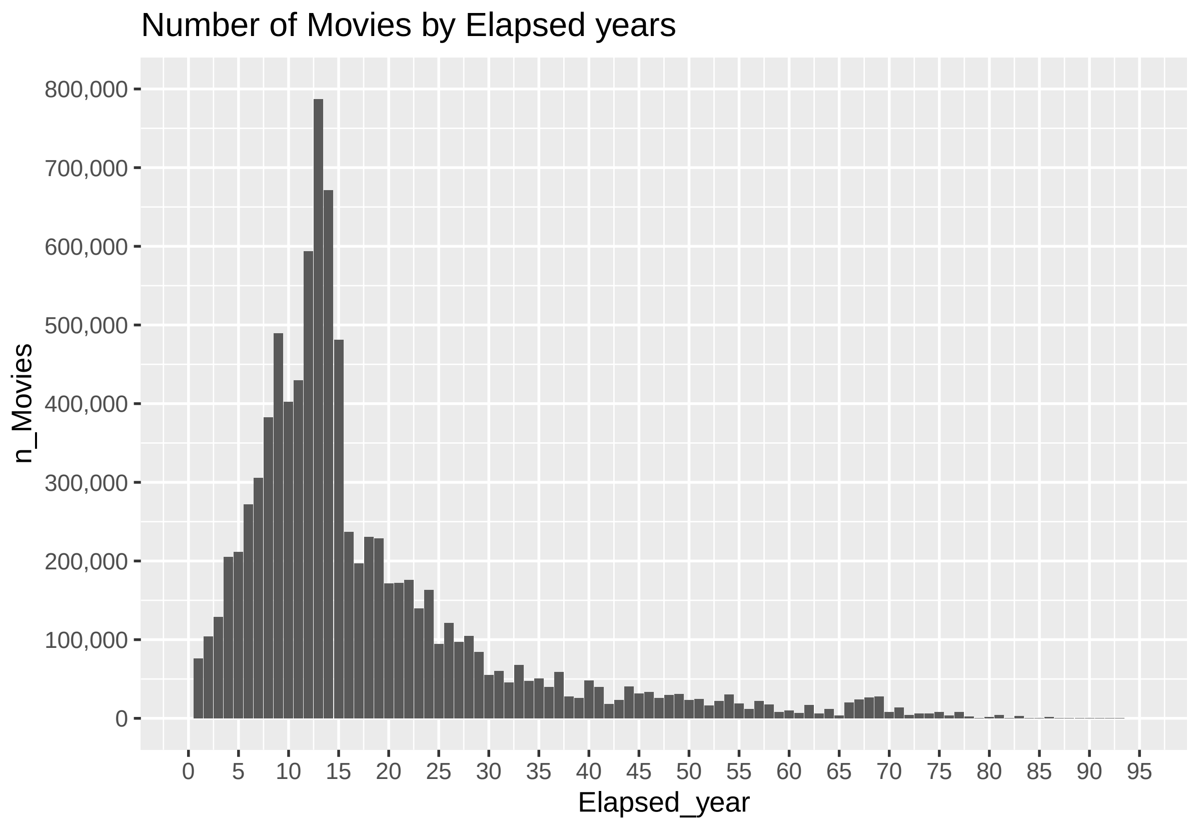
```
## # A tibble: 10,677 x 6
```

```
##    movieId title                             Views Avg_rating Total_Views Proportion
##      <int> <chr>                             <int>      <dbl>       <int>      <dbl>
##  1     296 "Pulp Fiction "                   31362       4.15     9000055    0.00348
##  2     356 "Forrest Gump "                   31079       4.01     9000055    0.00345
##  3     593 "Silence of the Lambs, The "      30382       4.20     9000055    0.00338
##  4     480 "Jurassic Park "                  29360       3.66     9000055    0.00326
##  5     318 "Shawshank Redemption, The "      28015       4.46     9000055    0.00311
##  6     110 "Braveheart "                     26212       4.08     9000055    0.00291
##  7     457 "Fugitive, The "                  25998       4.01     9000055    0.00289
##  8     589 "Terminator 2: Judgment Day "     25984       3.93     9000055    0.00289
##  9     260 "Star Wars: Episode IV - A N~     25672       4.22     9000055    0.00285
## 10     150 "Apollo 13 "                      24284       3.89     9000055    0.00270
## # i 10,667 more rows
```

```
# Add a column for elapsed years when the latest release year is 2008 and that is the base year.
edx <- edx %>%
  mutate(Elapsed_year = 2008 - Movie_year)

summary(edx$Elapsed_year)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    0.00   10.00   14.00   17.78   21.00   93.00
```

```
edx %>%
  mutate(Elapsed_year = 2008 - Movie_year) %>%
  group_by(Elapsed_year) %>%
  summarise(n_Movies = n()) %>%
  ggplot(aes(Elapsed_year,n_Movies))+
  geom_bar(stat = "identity", position = "dodge")+
  labs(title = "Number of Movies by Elapsed years")+
  scale_y_continuous(breaks = seq(0,800000,100000),limits = c(0,800000),labels = label_comma())+
  scale_x_continuous(breaks = seq(0,95,5),limits = c(0,95))
```




```r
# Check the number of movies and ratings by genre
edx %>%
  group_by(genres) %>%
  summarise(Movies_n = n(),
            Avg_Rating = mean(rating)) %>%
  arrange(desc(Movies_n))
```

```
## # A tibble: 797 x 3
##    genres                         Movies_n Avg_Rating
##    <chr>                             <int>      <dbl>
##  1 Drama                            733296       3.71
##  2 Comedy                           700889       3.24
##  3 Comedy|Romance                   365468       3.41
##  4 Comedy|Drama                     323637       3.60
##  5 Comedy|Drama|Romance             261425       3.65
##  6 Drama|Romance                    259355       3.61
##  7 Action|Adventure|Sci-Fi          219938       3.51
##  8 Action|Adventure|Thriller        149091       3.43
##  9 Drama|Thriller                   145373       3.45
```

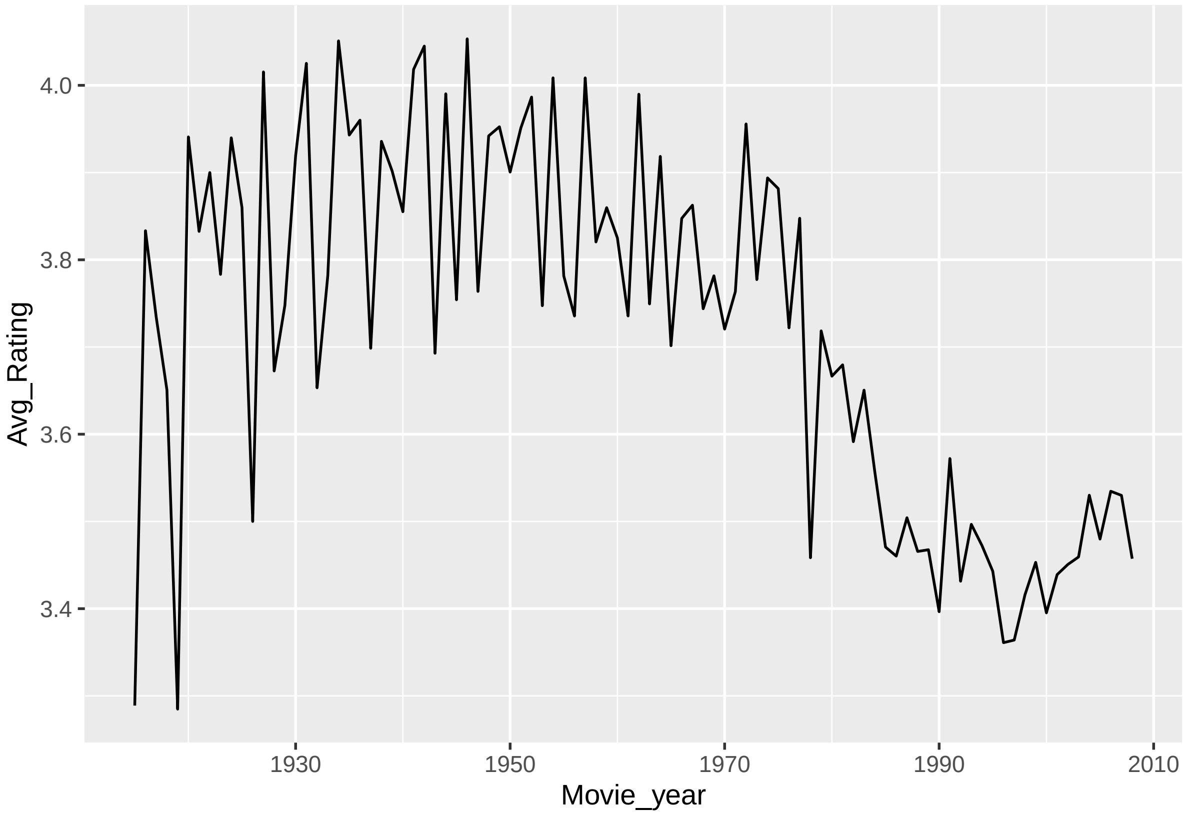
```
## 10 Crime|Drama                    137387        3.95
## # i 787 more rows
```

```r
# Unique genre check
edx %>%
  separate_rows(genres, sep = "\\|") %>%
  group_by(genres) %>%
  summarize(count = n(),
            Avg_rating = mean(rating)) %>%
  arrange(desc(Avg_rating))
```

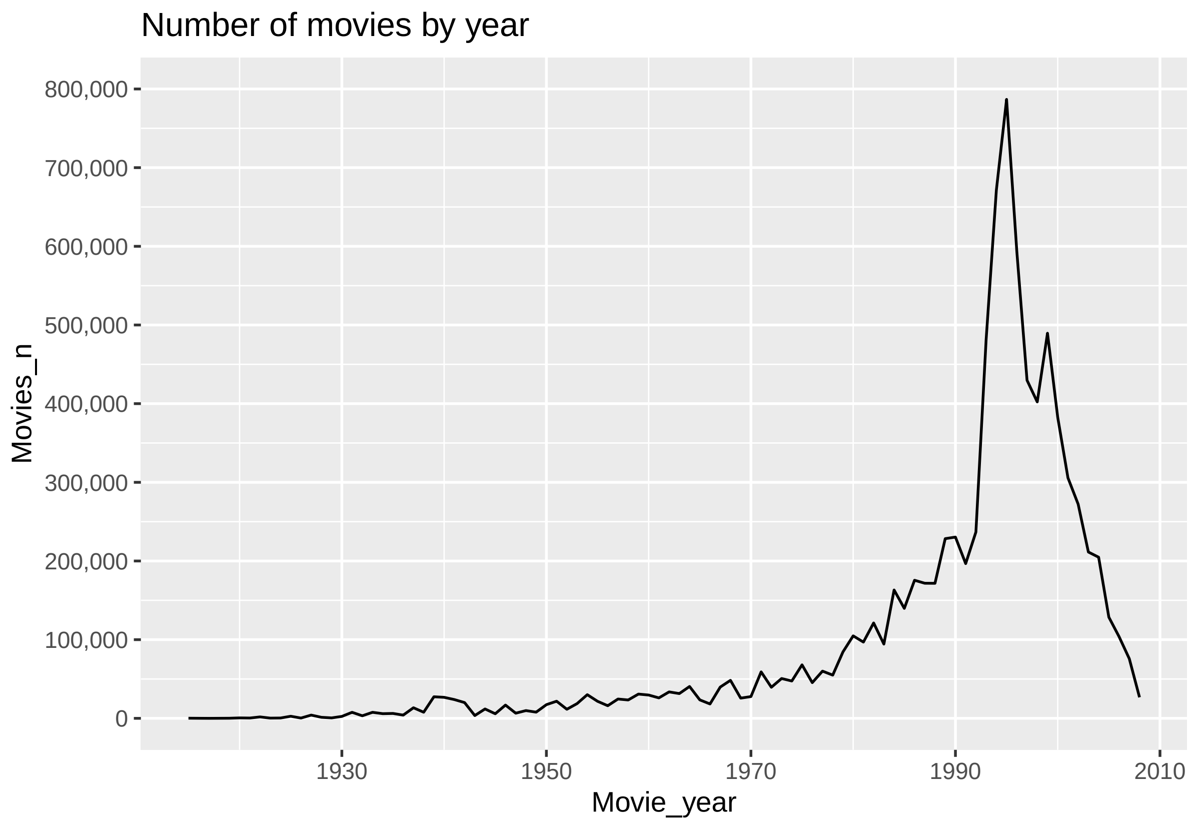
```
## # A tibble: 20 x 3
##    genres               count Avg_rating
##    <chr>                <int>      <dbl>
##  1 Film-Noir           118541       4.01
##  2 Documentary          93066       3.78
##  3 War                 511147       3.78
##  4 IMAX                  8181       3.77
##  5 Mystery             568332       3.68
##  6 Drama              3910127       3.67
##  7 Crime              1327715       3.67
##  8 (no genres listed)       7       3.64
##  9 Animation           467168       3.60
## 10 Musical             433080       3.56
## 11 Western             189394       3.56
## 12 Romance            1712100       3.55
## 13 Thriller           2325899       3.51
## 14 Fantasy             925637       3.50
## 15 Adventure          1908892       3.49
## 16 Comedy             3540930       3.44
## 17 Action             2560545       3.42
## 18 Children            737994       3.42
## 19 Sci-Fi             1341183       3.40
## 20 Horror              691485       3.27
```

```r
# Number of movies and ratings by year.
edx %>% group_by(Movie_year) %>%
  summarise(Movies_n =n(),
            Avg_Rating = mean(rating)) %>%
  ggplot(aes(Movie_year,Avg_Rating))+
```

```
  geom_line()
```




```
edx %>% group_by(Movie_year) %>%
  summarise(Movies_n =n(),
            Avg_Rating = mean(rating)) %>%
  ggplot(aes(Movie_year,Movies_n))+
  geom_line()+
  scale_y_continuous(breaks = seq(0,800000,100000),limits = c(0,800000),labels = label_comma())+
  labs(title = "Number of movies by year")
```



Build several models to increase accuracy & Accuracy evaluation

```
###################################
# Now that I have a general understanding of the data set, I will move on to the modeling phase.
# First, I will separate the dataset from edx into train set and test set.
test_index <- createDataPartition(y = edx$rating, times = 1, p = 0.2, list = FALSE)
train_set <- edx[-test_index,]
temp <- edx[test_index,]

# Ensure that the userId and movieId from the test set are also included in the training set.
test_set <- temp %>%
  semi_join(train_set, by = "movieId") %>%
  semi_join(train_set, by = "userId")

# Next, the rows removed from the test set are added to the training set.
removed <- anti_join(temp, test_set)
```

```r
train_set <- rbind(train_set, removed)

# Delete temporary files to keep the environment tidy
rm(test_index, temp, removed)


##################################
# Creating RMSE Functions
RMSE <- function(true_ratings, predicted_ratings){
  sqrt(mean((true_ratings - predicted_ratings)^2))
}

# Doing model development in the spirit of trial and error.

#1  Simple Average Model
mu <- mean(train_set$rating)
mu
```

```
## [1] 3.51257
```

```r
Model_1_RMSE <- RMSE(test_set$rating,mu)
Model_1_RMSE
```

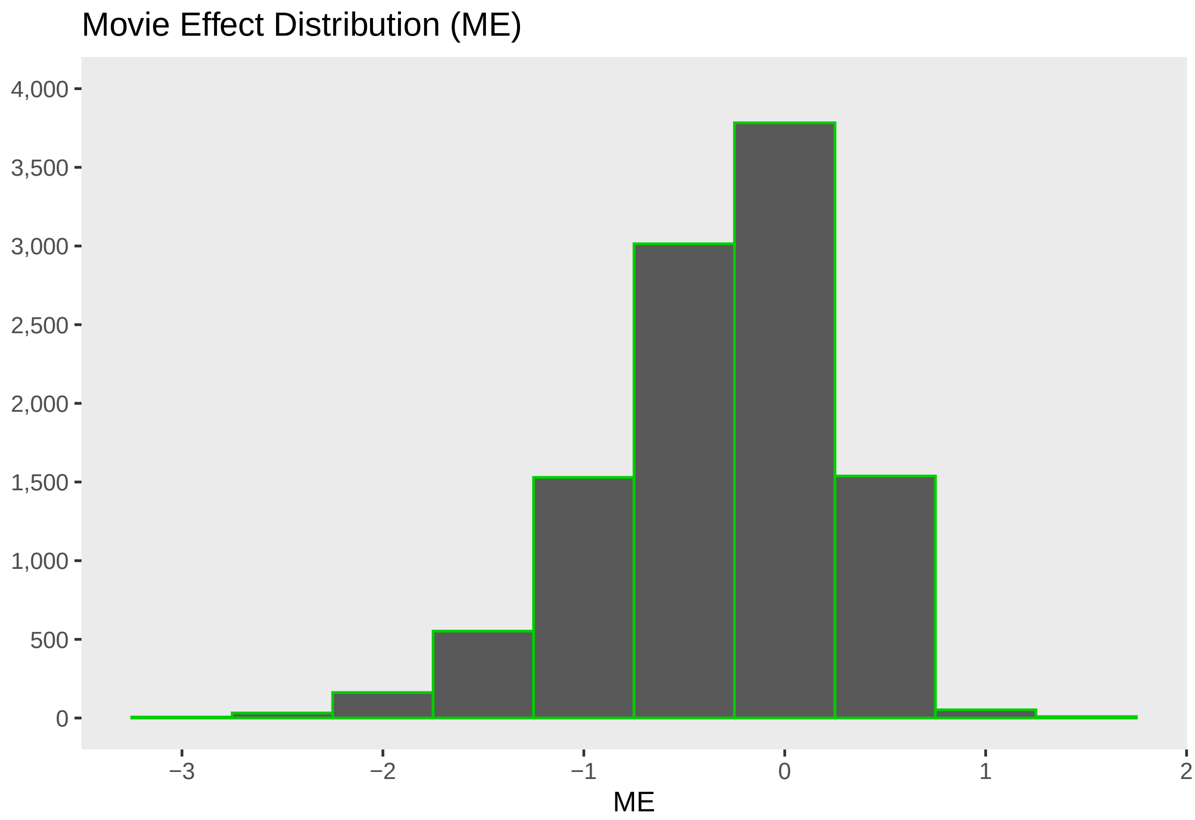
```
## [1] 1.060704
```

```r
#2  Building a model that takes into account User effects and Movie effects

# Movie Effects
ME <- train_set %>%
  group_by(movieId) %>%
  summarise(ME = mean(rating - mu))

ME
```

```
## # A tibble: 10,677 x 2
##    movieId      ME
##      <int>   <dbl>
##  1       1  0.415
##  2       2 -0.299
##  3       3 -0.363
```

```
##   4         4 -0.624
##   5         5 -0.449
##   6         6  0.304
##   7         7 -0.159
##   8         8 -0.367
##   9         9 -0.526
## 10        10 -0.0898
## # i 10,667 more rows
```

```
# Distribution Visualization
ME %>% ggplot(aes(x = ME)) +
  geom_histogram(bins=10, col = I("green3")) +
  ggtitle("Movie Effect Distribution (ME) ") +
  theme(panel.grid = element_blank(),axis.title.y = element_blank())+
  scale_y_continuous(breaks = seq(0,4000,500),limits = c(0,4000),labels = label_comma())
```

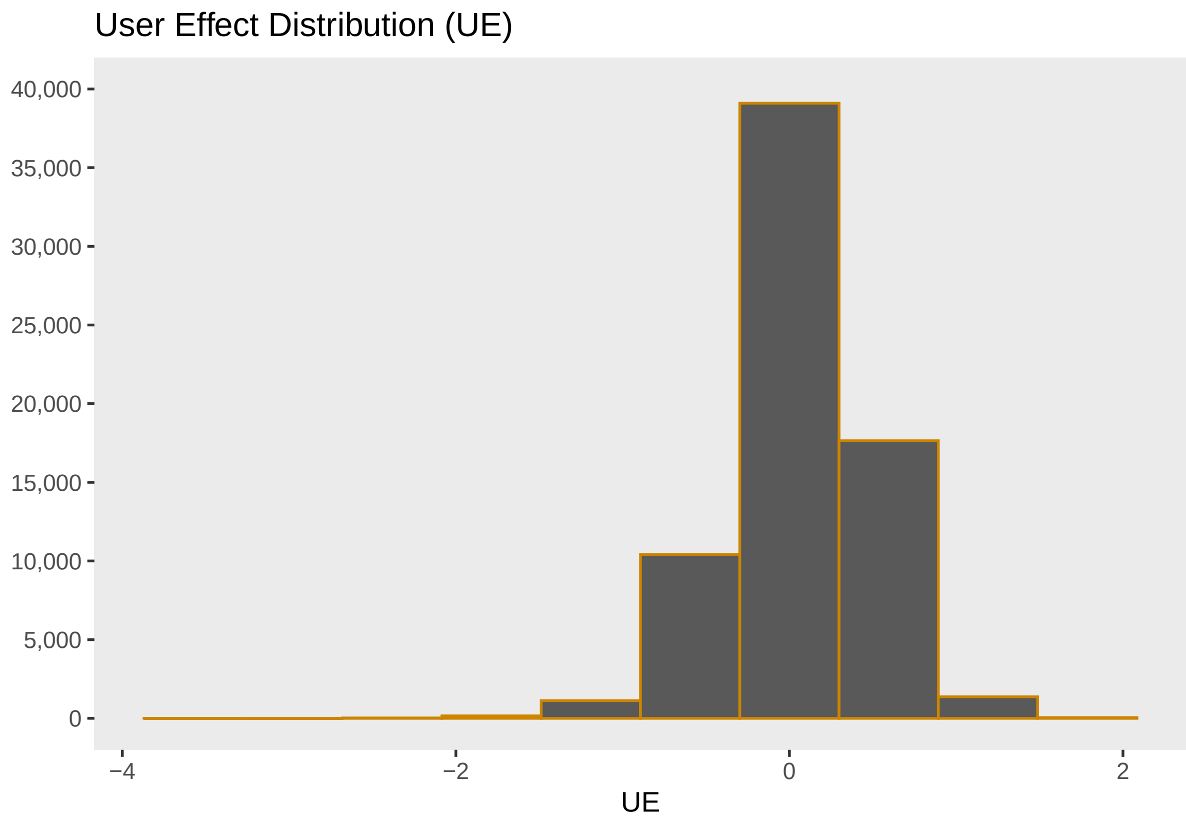



```
# User Effects
UE <- train_set %>%
  left_join(ME,by = "movieId") %>%
```

```r
  group_by(userId) %>%
  summarise(UE = mean(rating - mu - ME))

UE
```

```
## # A tibble: 69,878 x 2
##    userId     UE
##     <int>  <dbl>
##  1      1  1.67
##  2      2 -0.202
##  3      3  0.372
##  4      4  0.765
##  5      5  0.0517
##  6      6  0.338
##  7      7  0.0470
##  8      8  0.189
##  9      9  0.249
## 10     10  0.113
## # i 69,868 more rows
```

```r
# Distribution Visualization
UE %>% ggplot(aes(x = UE)) +
  geom_histogram(bins=10, col = I("orange3")) +
  ggtitle("User Effect Distribution (UE) ") +
  theme(panel.grid = element_blank(),axis.title.y = element_blank())+
  scale_y_continuous(breaks = seq(0,40000,5000),limits = c(0,40000),labels = label_comma())
```




```r
# Confirmation of RMSE in this model
predicted_ratings <- test_set %>%
  left_join(ME, by="movieId") %>%
  left_join(UE, by="userId") %>%
  mutate(prediction = mu + ME + UE) %>%
  .$prediction

summary(predicted_ratings)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  -1.346   3.135   3.566   3.513   3.946   6.153
```

```r
Model_2_RMSE <- RMSE(test_set$rating,predicted_ratings)
Model_2_RMSE
```

```
## [1] 0.8661625
```

```r
#3  Further regularized model by taking into account User effects and Movie effects
# The term "regularization" is derived from the regular matrix of linear algebra.
```

```r
# The purpose of regularization in machine learning is to avoid over-fitting.
# Simpler explanation is that it has the effect of simplifying complex models.
# Lambda is also the most versatile method that can be used to prevent over-learning in any analysis
# Lambda: regularization parameter This adjusts the influence of the regularization term

# Create a sequence of values for lambda ranging from 0 to 10 with 0.25 increments
lambda <- seq(0, 10, 0.25)

# After building the regularization model, the ratings are predicted and the RMSE at each lambda val
RMSES <- sapply(lambda, function(l){
  ME <- train_set %>%
    group_by(movieId) %>%
    summarise(ME = sum(rating - mu)/(n()+l))
  UE <- train_set %>%
    left_join(ME, by="movieId") %>%
    group_by(userId) %>%
    summarise(UE = sum(rating - ME - mu)/(n()+l))
  predicted_ratings <- test_set %>%
    left_join(ME, by="movieId") %>%
    left_join(UE, by="userId") %>%
    mutate(pred = mu + ME + UE) %>%
    pull(pred)
  return(RMSE(predicted_ratings, test_set$rating))
})

# Assign optimal regularization parameters, i.e., lambda
lambda <- lambda[which.min(RMSES)]
lambda
```

```
## [1] 4.75
```

```r
# Apply and validate the regularized model against the validation data set
ME <- edx %>%
  group_by(movieId) %>%
  summarise(ME = sum(rating - mu)/(n()+lambda))

UE <- edx %>%
  left_join(ME, by="movieId") %>%
  group_by(userId) %>%
```

```r
  summarise(UE = sum(rating - ME - mu)/(n()+lambda))

# Ratings prediction for "final_holdout_test" indicated in the course
predicted_ratings <- final_holdout_test %>%
  left_join(ME, by="movieId") %>%
  left_join(UE, by="userId") %>%
  mutate(pred = mu + ME + UE) %>%
  pull(pred)

# Calculate RMSE and evaluate accuracy
rmse_valid_result <- RMSE(final_holdout_test$rating, predicted_ratings)
rmse_valid_result
```

```
## [1] 0.8648201
```

Comments

I managed to achieve my RMSE score through trial and error. I would have liked to try the matrix factorization approach, but due to the looming deadline, I avoided it this time. I would like to try that approach someday.

References

1.https://atmarkit.itmedia.co.jp/ait/articles/2108/27/news013.html

2.https://qiita.com/yo_fuji/items/56a22c9829d40ce7a3ff

3.https://note.com/ryuichiro/n/n3ef1fc1026f6

4.https://data-viz-lab.com/overfitting

5.https://qiita.com/c60evaporator/items/784f0640004be4eefc51