HarvardX PH125.9x Data Science: Capstone

Lê Công Bình

2021-12-07

1. Introduction

This report is part of the capstone project of HarvardX's Data Science Professional Certificate program. The most of content bases on the knowledge of module 6.2 - Recommendation Systems in HarvardX PH125.8x - Data Science: Machine Learning course. It also presents my skill in "Data Science: Wrangling" and "Data Science: Visualization" courses.

Because of large dataset I can not use almost of algorithm (lm, lgm, knn, or tree base) to forecast. I build model by estimating important factor of dataset.

2. EDA and Modeling

I first set up the working environment by installing essential packages. Almost of package already have in assessment. In addition I used lubridate package for date processing

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

```
## Warning: package 'tidyverse' was built under R version 4.1.2

## Warning: package 'forcats' was built under R version 4.1.2
```

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

```
## Warning: package 'caret' was built under R version 4.1.2
```

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

```
## Warning: package 'data.table' was built under R version 4.1.2
```

```
library(tidyverse)
library(caret)
library(data.table)
library(lubridate)
```

2.1. Data Wrangling

Information of the MovieLens 10M dataset can be found: https://grouplens.org/datasets/movielens/10m/

The dataset can be downloaded here: http://files.grouplens.org/datasets/movielens/ml-10m.zip

I use the following code to download the dataset and combine the information of the ratings and the movies into movielens:

```
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")

# if using R 3.6 or earlier:
#movies <- as.data.frame(movies) %>% mutate(movieId = as.numeric(levels(movieId))[movieId],
#                                            title = as.character(title),
#                                            genres = as.character(genres))
# if using R 4.0 or later:
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")

# Validation set will be 10% of MovieLens data
set.seed(1, sample.kind="Rounding") # if using R 3.5 or earlier, use `set.seed(1)`
```

```
## Warning in set.seed(1, sample.kind = "Rounding"): non-uniform 'Rounding' sampler
## used
```

```
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)
```

2.2 Exploratory Data Analysis

Data size and structure:

- There are 9000055 observes with 6 features for modeling (edx dataset) each of row is a rate by a user for a movie at specific time
- Exclude title column all other columns will be use to predict

```
edx %>% as_tibble()
```

```
## # A tibble: 9,000,055 x 6
##    userId movieId rating timestamp title                   genres
##     <int>   <dbl>  <dbl>     <int> <chr>                   <chr>
##  1      1     122      5 838985046 Boomerang (1992)        Comedy|Romance
##  2      1     185      5 838983525 Net, The (1995)         Action|Crime|Thriller
##  3      1     292      5 838983421 Outbreak (1995)         Action|Drama|Sci-Fi|T~
##  4      1     316      5 838983392 Stargate (1994)         Action|Adventure|Sci-~
##  5      1     329      5 838983392 Star Trek: Generation~ Action|Adventure|Dram~
##  6      1     355      5 838984474 Flintstones, The (199~ Children|Comedy|Fanta~
##  7      1     356      5 838983653 Forrest Gump (1994)     Comedy|Drama|Romance|~
##  8      1     362      5 838984885 Jungle Book, The (199~ Adventure|Children|Ro~
##  9      1     364      5 838983707 Lion King, The (1994)  Adventure|Animation|C~
## 10      1     370      5 838984596 Naked Gun 33 1/3: The~ Action|Comedy
## # ... with 9,000,045 more rows
```

- We have 68878 users who rated for 10677 movies

```
edx %>%
  summarize(n_users = n_distinct(userId),
            n_movies = n_distinct(movieId))
```

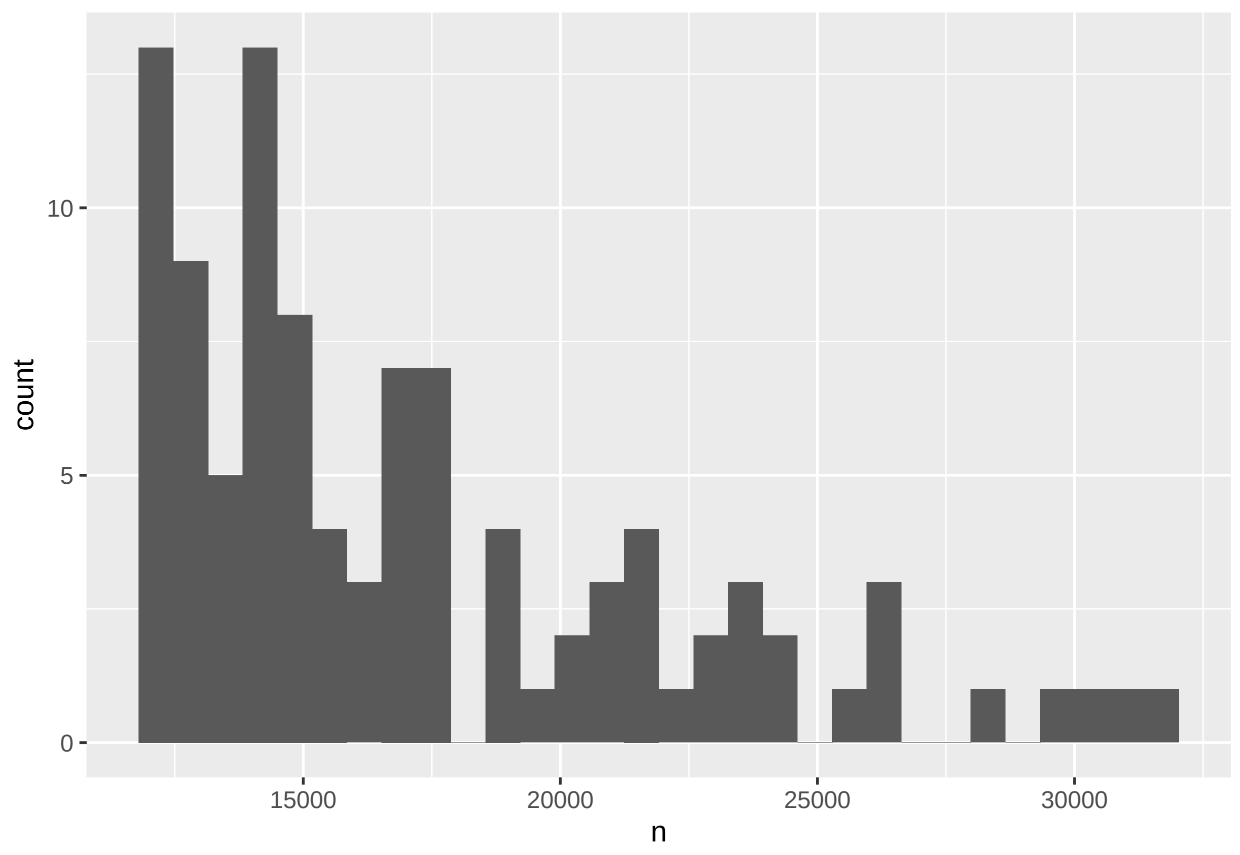
```
##   n_users n_movies
## 1   69878    10677
```

- There are some movie get many rate from user but other don't

```
edx %>%
  group_by(movieId) %>% summarize(n=n()) %>%
  arrange(desc(n)) %>% top_n(100) %>%
  ggplot(aes(x=n)) +
  geom_histogram()
```

```
## Selecting by n
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```

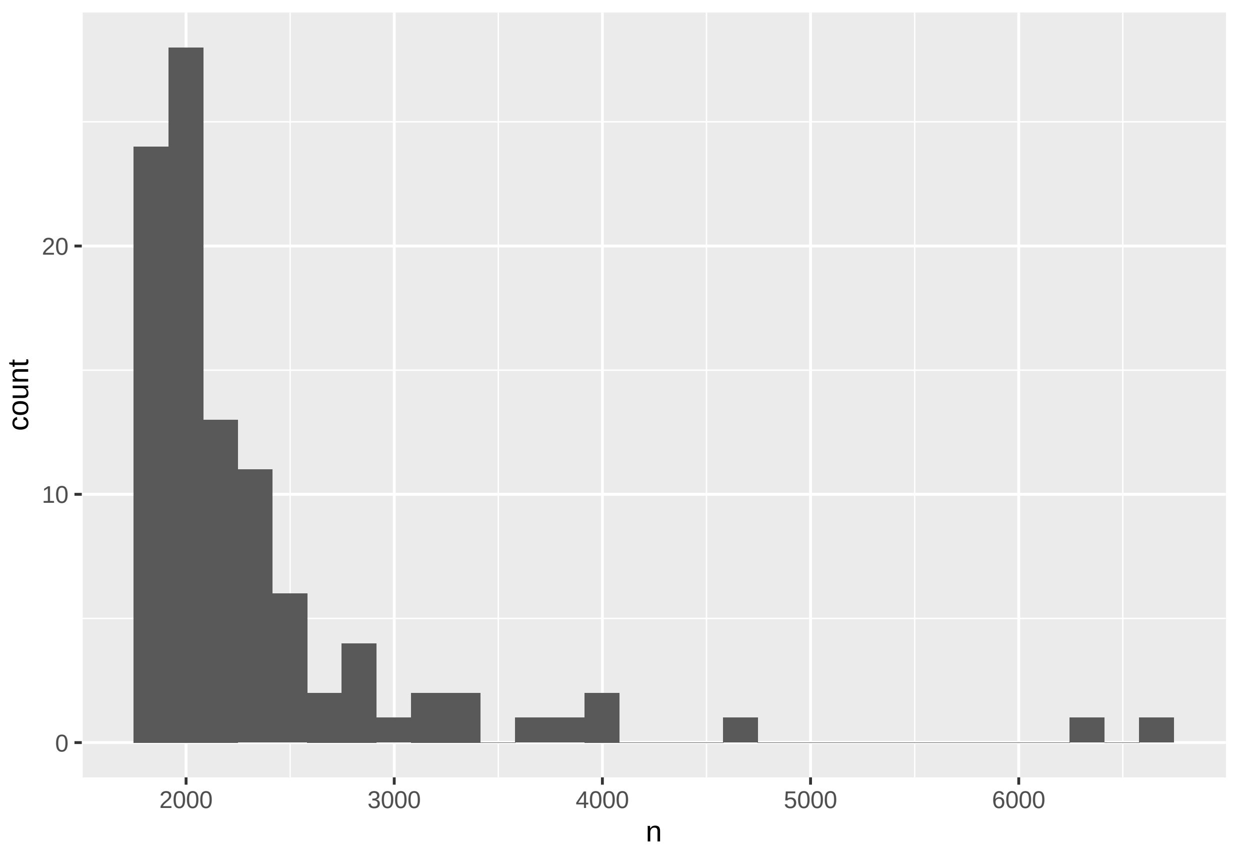


- Some users are more active than others at rating movies

```
edx %>%
  group_by(userId) %>% summarize(n=n()) %>% arrange(desc(n)) %>%
  top_n(100) %>%
  ggplot(aes(x=n)) +
  geom_histogram()
```

```
## Selecting by n
```

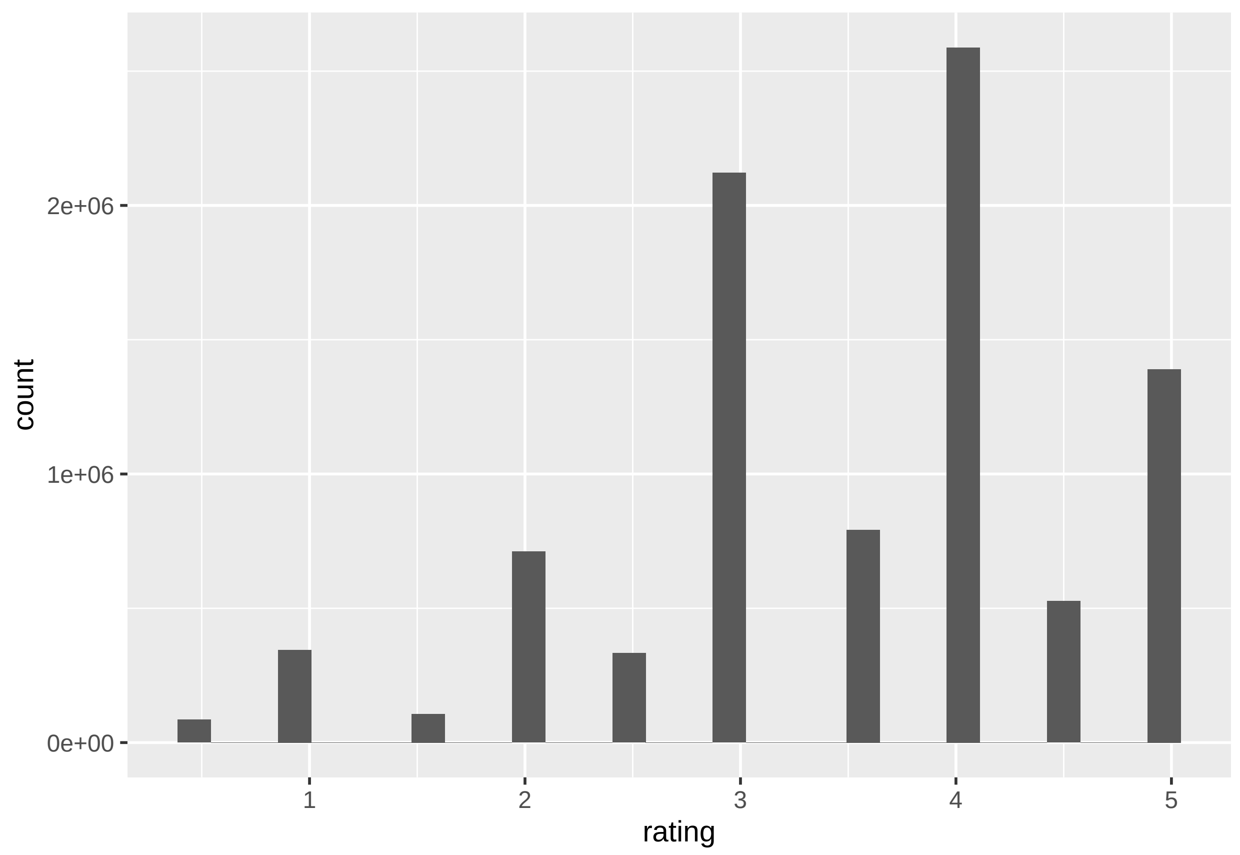
```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



- The most of rating is 3 or 4 star. We can see a few lower rating than higher rating

```
edx %>%
    ggplot(aes(x=rating)) +
    geom_histogram()
```

```
## ‘stat_bin()‘ using ‘bins = 30‘. Pick better value with ‘binwidth‘.
```

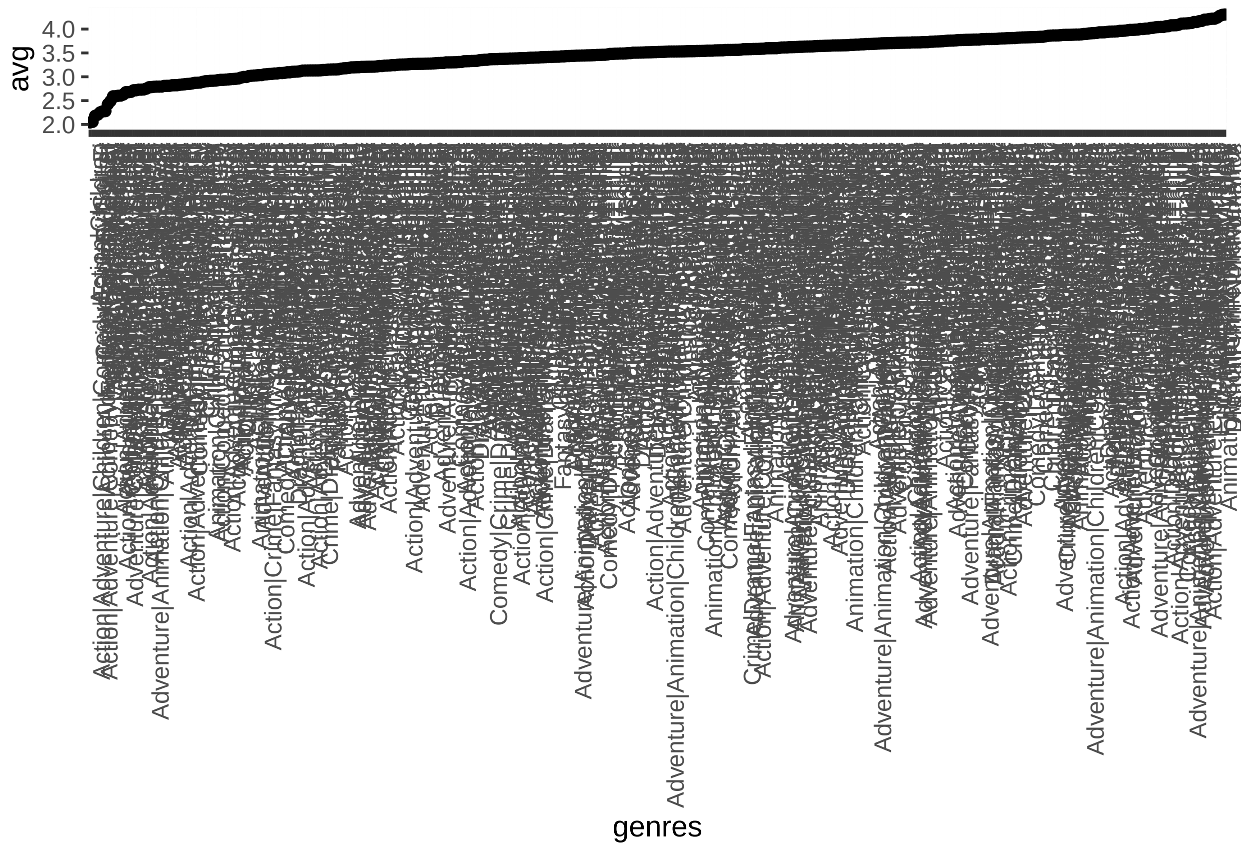


- Timestamp data: There is some evidence of a time effect on average rating.

```
edx <- mutate(edx, date = as_datetime(timestamp),week = round_date(date,"week"))
edx <- mutate(edx, date = as_datetime(timestamp),week = round_date(date,"week"))
```

- Strong evidence of a genre effect.

```
# genres_avg <- edx %>% separate_rows(genres,sep = "\\|") %>% group_by(genres) %>% summarize(avg_rate=m
edx %>% group_by(genres) %>%
  summarize(n = n(), avg = mean(rating), se = sd(rating)/sqrt(n())) %>%
    filter(n >= 1000) %>%
    mutate(genres = reorder(genres, avg)) %>%
    ggplot(aes(x = genres, y = avg, ymin = avg - 2*se, ymax = avg + 2*se)) +
    geom_point() +
    geom_errorbar() +
    theme(axis.text.x = element_text(angle = 90, hjust = 1))
```



- Overall Quality: There isn't any NA value in dataset.

```
c(sum(is.na(edx$UserId)),sum(is.na(edx$movieId)),sum(is.na(edx$rating)))
```

```
## [1] 0 0 0
```

2.3 Loss Function

- Loss function based on the residual mean squared error (RMSE) on a test set

```
RMSE <- function(true_ratings, predicted_ratings){
    sqrt(mean((true_ratings - predicted_ratings)^2))
  }
```

2.4. Modeling

Base on EDA, the modeling will estimate by train rating, user, movie, genre and timestamp

```
# Naive by average
mu <- mean(edx$rating)
naive_rmse <- RMSE(edx$rating, mu)
rmse_results <- tibble(method = "Just the average", RMSE = naive_rmse)
```

```
# Add movie effect
movie_avgs <- edx %>% group_by(movieId) %>% summarize(b_i = mean(rating - mu))
predicted_ratings <- mu + validation %>% left_join(movie_avgs, by='movieId') %>% .$b_i
model_1_rmse <- RMSE(predicted_ratings, validation$rating)
rmse_results <- bind_rows(rmse_results, tibble(method="Movie Effect Model", RMSE = model_1_rmse ))

# Add user effect
user_avgs <- edx %>% left_join(movie_avgs, by='movieId') %>%
    group_by(userId) %>%
    summarize(b_u = mean(rating - mu - b_i))
predicted_ratings <- validation %>%
    left_join(movie_avgs, by='movieId') %>%
    left_join(user_avgs, by='userId') %>%
    mutate(pred = mu + b_i + b_u) %>% .$pred
model_2_rmse <- RMSE(predicted_ratings, validation$rating)
rmse_results <- bind_rows(rmse_results, tibble(method="Movie + User Effects Model", RMSE = model_2_rmse

# Add genre effect
genre_avgs <- edx %>%
    left_join(movie_avgs, by='movieId') %>%
    left_join(user_avgs, by='userId') %>%
    group_by(genres) %>%
    summarize(b_g = mean(rating - mu - b_i - b_u))
predicted_ratings <- validation %>%
    left_join(movie_avgs, by='movieId') %>%
    left_join(user_avgs, by='userId') %>%
    left_join(genre_avgs, by='genres') %>%
    mutate(pred = mu + b_i + b_u + b_g) %>% .$pred
model_3_rmse <- RMSE(predicted_ratings, validation$rating)
rmse_results <- bind_rows(rmse_results, tibble(method="Movie + User + Genre Effects Model", RMSE = model

# Add time stamp effect
week_avgs <- edx %>%
    left_join(movie_avgs, by='movieId') %>%
    left_join(user_avgs, by='userId') %>%
    left_join(genre_avgs, by='genres') %>%
    group_by(week) %>%
    summarize(b_w = mean(rating - mu - b_i - b_u - b_g))
predicted_ratings <- validation %>% mutate(date = as_datetime(timestamp),week = round_date(date,"week"))
    left_join(movie_avgs, by='movieId') %>%
    left_join(user_avgs, by='userId') %>%
    left_join(genre_avgs, by='genres') %>%
    left_join(week_avgs, by='week') %>%
    mutate(pred = mu + b_i + b_u + b_g + b_w) %>% .$pred
model_4_rmse <- RMSE(predicted_ratings, validation$rating)
rmse_results <- bind_rows(rmse_results, tibble(method="Movie + User + Genre + Week Effects Model", RMSE

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


rmse_target <- 0.86490
rmse_results %>% mutate(RMSE_target = ifelse(RMSE < rmse_target, TRUE, FALSE ))
```

```
## # A tibble: 5 x 3
##   method                                  RMSE RMSE_target
##   <chr>                                  <dbl> <lgl>
## 1 Just the average                       1.06  FALSE
## 2 Movie Effect Model                     0.944 FALSE
## 3 Movie + User Effects Model             0.865 FALSE
## 4 Movie + User + Genre Effects Model     0.865 FALSE
## 5 Movie + User + Genre + Week Effects Model 0.865 TRUE
```

2.5. Model Tuning

```
set.seed(1, sample.kind="Rounding")
```

Create train and test using edx dataset only

```
## Warning in set.seed(1, sample.kind = "Rounding"): non-uniform 'Rounding' sampler
## used
```

```
test_index <- createDataPartition(y = edx$rating, times = 1, p = 0.1, list = FALSE)
train <- edx[-test_index,]
test <- edx[test_index,]
```

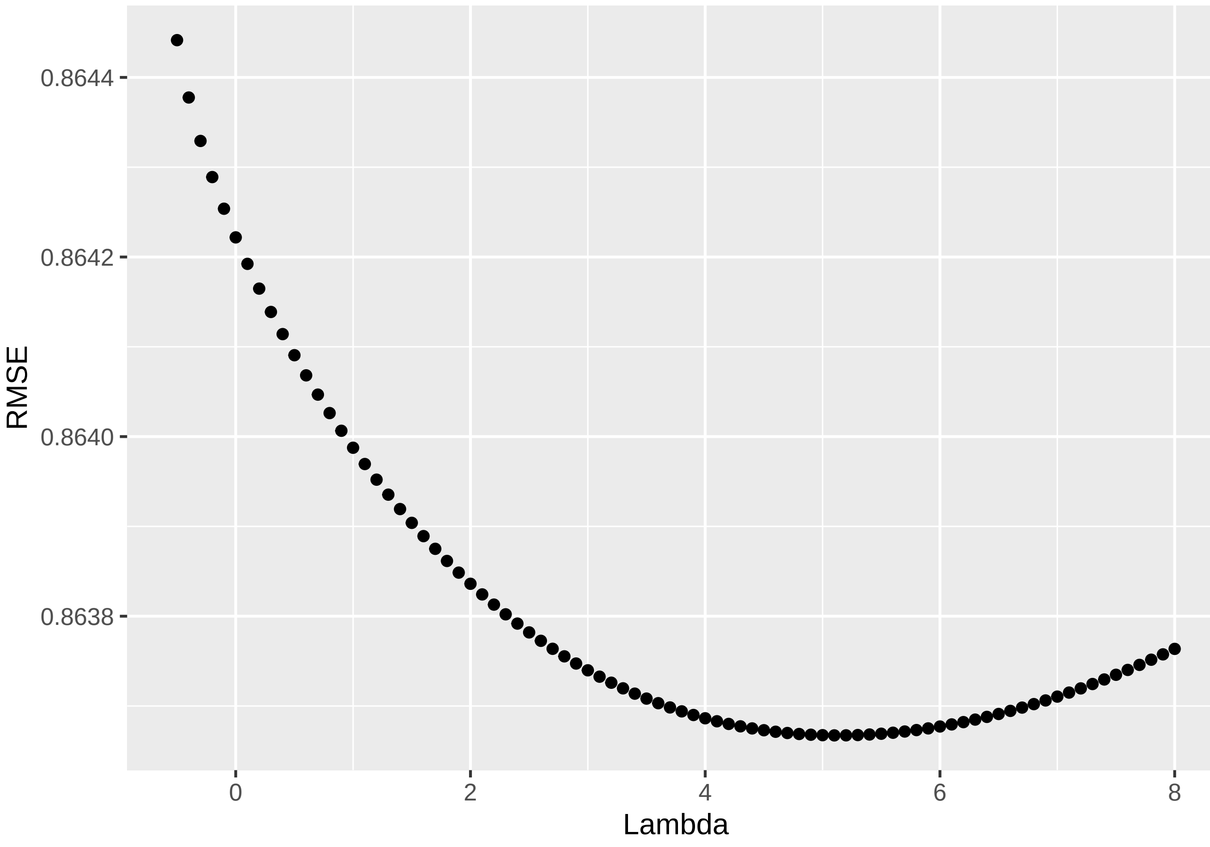
```
reglr_fit <- function(lambda, trainset, testset){
    mu <- mean(trainset$rating)
    b_i <- trainset %>% group_by(movieId) %>% summarize(b_i = sum(rating - mu)/(n()+lambda))
    b_u <- trainset %>% left_join(b_i, by="movieId") %>%
        group_by(userId) %>%
        summarize(b_u = sum(rating - b_i - mu)/(n()+lambda))
    b_g <- trainset %>% left_join(b_i, by="movieId") %>% left_join(b_u, by='userId') %>%
        group_by(genres) %>%
        summarize(b_g = sum(rating - b_i - b_u - mu)/(n()+lambda))
    b_w <- trainset %>% left_join(b_i, by="movieId") %>% left_join(b_u, by='userId') %>% left_join(b_g,
        group_by(week) %>%
        summarize(b_w = sum(rating - b_g - b_i - b_u - mu)/(n()+lambda))

    predicted_ratings <- testset %>%
        left_join(b_i, by = "movieId") %>%
        left_join(b_u, by = "userId") %>%
        left_join(b_g, by = "genres") %>%
        left_join(b_w, by = "week") %>%
        filter(!is.na(b_i), !is.na(b_u), !is.na(b_g), !is.na(b_w)) %>%
        mutate(pred = mu + b_i + b_u + b_g + b_w) %>%
        select(pred, rating)
    return(RMSE(predicted_ratings$pred, predicted_ratings$rating))
}

lambdas <- seq(-0.5, 8, 0.1)
```

```
rmses <- sapply(lambdas, reglr_fit, trainset = train, testset = test)

tibble(Lambda = lambdas, RMSE = rmses) %>%
  ggplot(aes(x = Lambda, y = RMSE)) +
    geom_point()
```



Regularization

From chart above I choosed lambda = 5 for evaluation

```
lambda <- 5

mu <- mean(edx$rating)
b_i <- edx %>% group_by(movieId) %>% summarize(b_i = sum(rating - mu)/(n()+lambda))
b_u <- edx %>% left_join(b_i, by="movieId") %>%
    group_by(userId) %>%
    summarize(b_u = sum(rating - b_i - mu)/(n()+lambda))
b_g <- edx %>% left_join(b_i, by="movieId") %>% left_join(b_u, by='userId') %>%
    group_by(genres) %>%
    summarize(b_g = sum(rating - b_i - b_u - mu)/(n()+lambda))
b_w <- edx %>% left_join(b_i, by="movieId") %>% left_join(b_u, by='userId') %>% left_join(b_g, by='genr
    group_by(week) %>%
    summarize(b_w = sum(rating - b_g - b_i - b_u - mu)/(n()+lambda))

predicted_ratings <- validation %>% mutate(date = as_datetime(timestamp),week = round_date(date,"week")
    left_join(b_i, by = "movieId") %>%
    left_join(b_u, by = "userId") %>%
    left_join(b_g, by = "genres") %>%
    left_join(b_w, by = "week") %>%
    #filter(!is.na(b_i), !is.na(b_u), !is.na(b_g), !is.na(b_w)) %>%
    mutate(pred = mu + b_i + b_u + b_g + b_w) %>% .$pred
model_5_rmse <- RMSE(predicted_ratings, validation$rating)

rmse_results <- bind_rows(rmse_results, tibble(method="Movie + User + Genre + Week + Regularization Effe
rmse_target <- 0.86490
rmse_results %>% mutate(RMSE_target_passing = ifelse(RMSE < rmse_target, TRUE, FALSE ))
```

```
## # A tibble: 6 x 3
##   method                                                RMSE RMSE_target_passi~
##   <chr>                                                <dbl> <lgl>
## 1 Just the average                                     1.06  FALSE
## 2 Movie Effect Model                                   0.944 FALSE
## 3 Movie + User Effects Model                           0.865 FALSE
## 4 Movie + User + Genre Effects Model                   0.865 FALSE
## 5 Movie + User + Genre + Week Effects Model            0.865 TRUE
## 6 Movie + User + Genre + Week + Regularization Effects~ 0.864 TRUE
```

It sounds better and passing target of project.

Linear model Can not done due to perfomance

```
#fit <- lm(rating ~ as.factor(movieId) + as.factor(userId), data = edx)
#y_hat <- predict(fit, newdata = test)
#RMSE(test$rating, y_hat)
```

I tried with glm but it don't have better score: RMSE for glm is 1.06

```
#fit <- train(rating ~ movieId + userId, method = "glm", data = train)
#glm_preds <- predict(fit, test)
#RMSE(test$rating,glm_preds)

#glm_vals <- predict(fit, validation)
#RMSE(validation$rating,glm_vals)
```

3. Result

- The best of model is use all feature: Movie, User, Genre, Week (round up date) and Regularization with lambda = 5: RMSE = 0.8643062.
- It's a bit better than project target: 0.86490

```
rmse_target <- 0.86490
rmse_results %>% mutate(RMSE_target_passing = ifelse(RMSE < rmse_target, TRUE, FALSE ))
```

```
## # A tibble: 6 x 3
##   method                                                RMSE RMSE_target_passi~
##   <chr>                                                <dbl> <lgl>
## 1 Just the average                                     1.06  FALSE
## 2 Movie Effect Model                                   0.944 FALSE
## 3 Movie + User Effects Model                          0.865 FALSE
## 4 Movie + User + Genre Effects Model                  0.865 FALSE
## 5 Movie + User + Genre + Week Effects Model           0.865 TRUE
## 6 Movie + User + Genre + Week + Regularization Effects~ 0.864 TRUE
```

4. Conclusion

Some machine learning algorithms are computationally expensive. The required amount of memory far exceeded the available in a commodity laptop, even with increased virtual memory. The model works only for existing users, movies and rating values, so the algorithm must run every time a new user or movie is included, or when the rating changes.

In the future I will try with Matrix factorization or use open source package recosystem to evaluation

5. References

1. Rafael A. Irizarry (2021), Introduction to Data Science: https://rafalab.github.io/dsbook/
2. Model Fitting and Recommendation Systems Overview: https://learning.edx.org/course/course-v1:HarvardX+PH125.8x+2T2021/block-v1:HarvardX+PH125.8x+2T2021+type@sequential+block@568d02a4412440489621921b87e7536f/block-v1:HarvardX+PH125.8x+2T2021+type@vertical+block@d7b4b0783dd443d1bd14504fe2333c24