Movielens capstone project

Daisuke Ohnuki

4/26/2021

Contents

1 Introduction

Recommendation system are among most important applications of machine learning deployed by digital companies including Amazon and Netflix today. They use the systems to understand their customers' tendencies to target them with their products in more effective and efficient way. For example, Netflix awarded a one million prize for data scientists who could successfully achieved the challenge for improving their recommendation algorithm by 10 per cent.

The MovieLens datasets have provided a popular environment for experiencing with machie learning since their launching in 1997.

The goal of this capstone project is to develop a recommendation system using the MovieLens datasets with ten million movie ratings, for achieving the RMSE, root means square error, of less than 0.86490.

To facilitate this project, first we split the datasets into a training set, as "edx set", and final hold-out test set, as "validation set". Second, we set out the exploratory data analysis with visualization techniques. Then we build up the modelings to find an appropriate one, which we would apply to the validation set. We conclude with our outcome for the final model, with limitations of this project and possibilities for future works.

2 Exploratory Data Analysis

First of all, download the Movielens dataset and create Edx set and validation set from the Movielens dataset.

```
#Create Edx set and validation set from Movielens data set.
options(timeout=300)
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")
# Validation set will be 10% of MovieLens data
set.seed(1, sample.kind = "Rounding")
```

```
## 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)
```

Second, looking through the head data and the dimension of the edx data set. The edx data set consists of 9,000,055 rows and 6 colums, with 10,677 unique movies and 69,878 unique users.

```
##    userId movieId rating timestamp                         title
## 1:      1     122      5 838985046              Boomerang (1992)
## 2:      1     185      5 838983525               Net, The (1995)
## 3:      1     292      5 838983421               Outbreak (1995)
## 4:      1     316      5 838983392               Stargate (1994)
## 5:      1     329      5 838983392 Star Trek: Generations (1994)
## 6:      1     355      5 838984474       Flintstones, The (1994)
##                            genres
## 1:                Comedy|Romance
## 2:          Action|Crime|Thriller
## 3:  Action|Drama|Sci-Fi|Thriller
## 4:        Action|Adventure|Sci-Fi
## 5: Action|Adventure|Drama|Sci-Fi
## 6:        Children|Comedy|Fantasy
```

```
## [1] 9000055       6
```

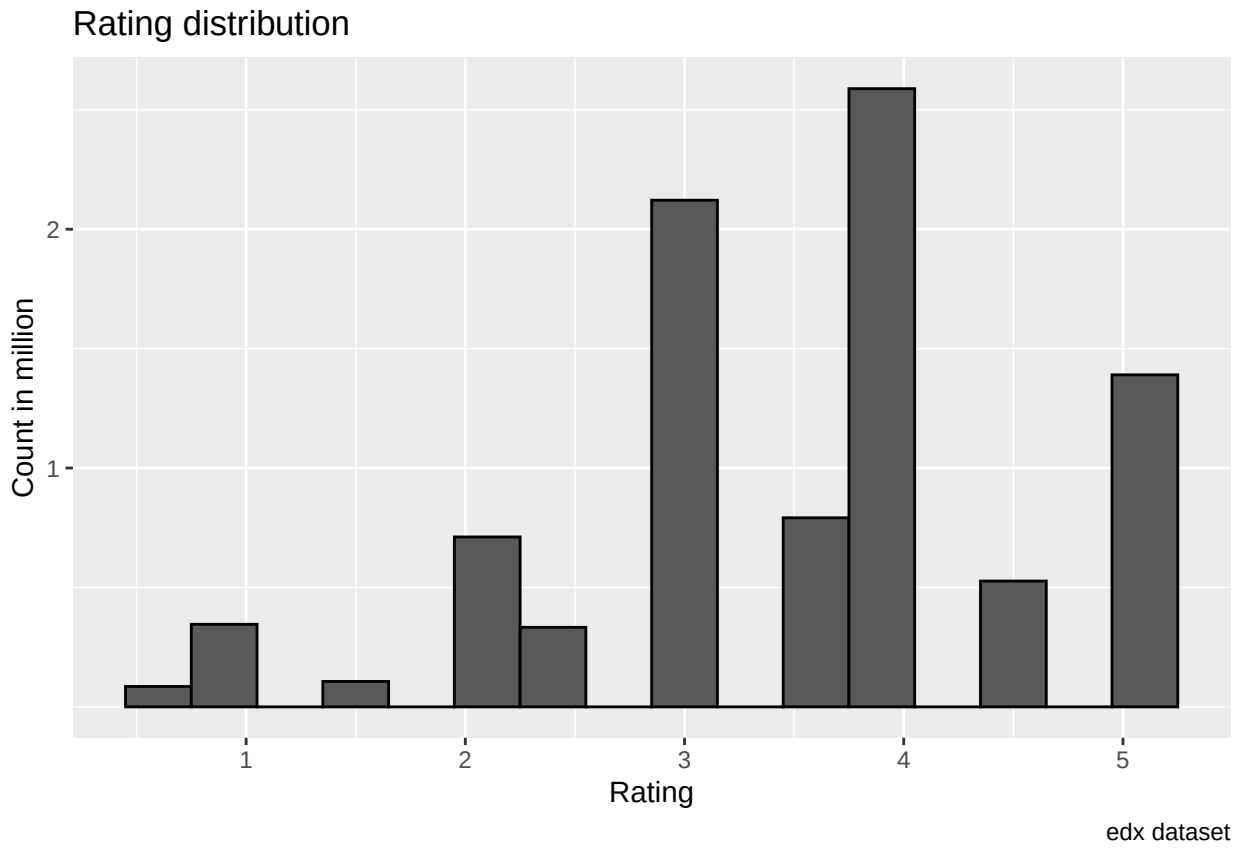
Also, check the dimention of the train set.

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

2.1 Ratings

The average of rating is 3.51. The median is 4.

```
# Plot distribution of ratings in the edx dataset
edx %>% ggplot(aes(rating)) +
  geom_histogram(binwidth = 0.3, color = I("black")) +
  scale_y_continuous(breaks = c(1000000, 2000000), labels = c("1", "2")) +
  labs(x = "Rating", y = "Count in million", caption = "edx dataset") +
  ggtitle("Rating distribution")
```



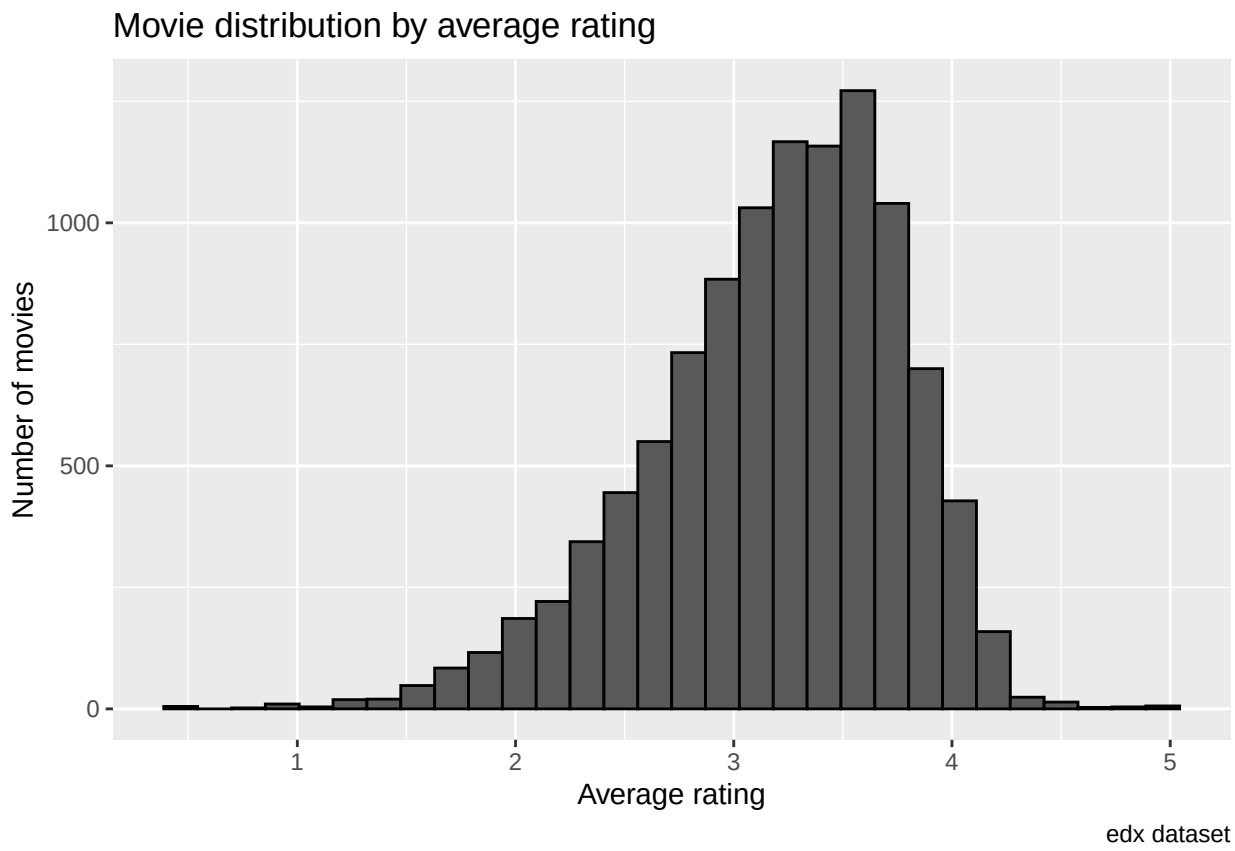

```
## [1] 3.512465
```

```
## [1] 4
```

2.2 Movies

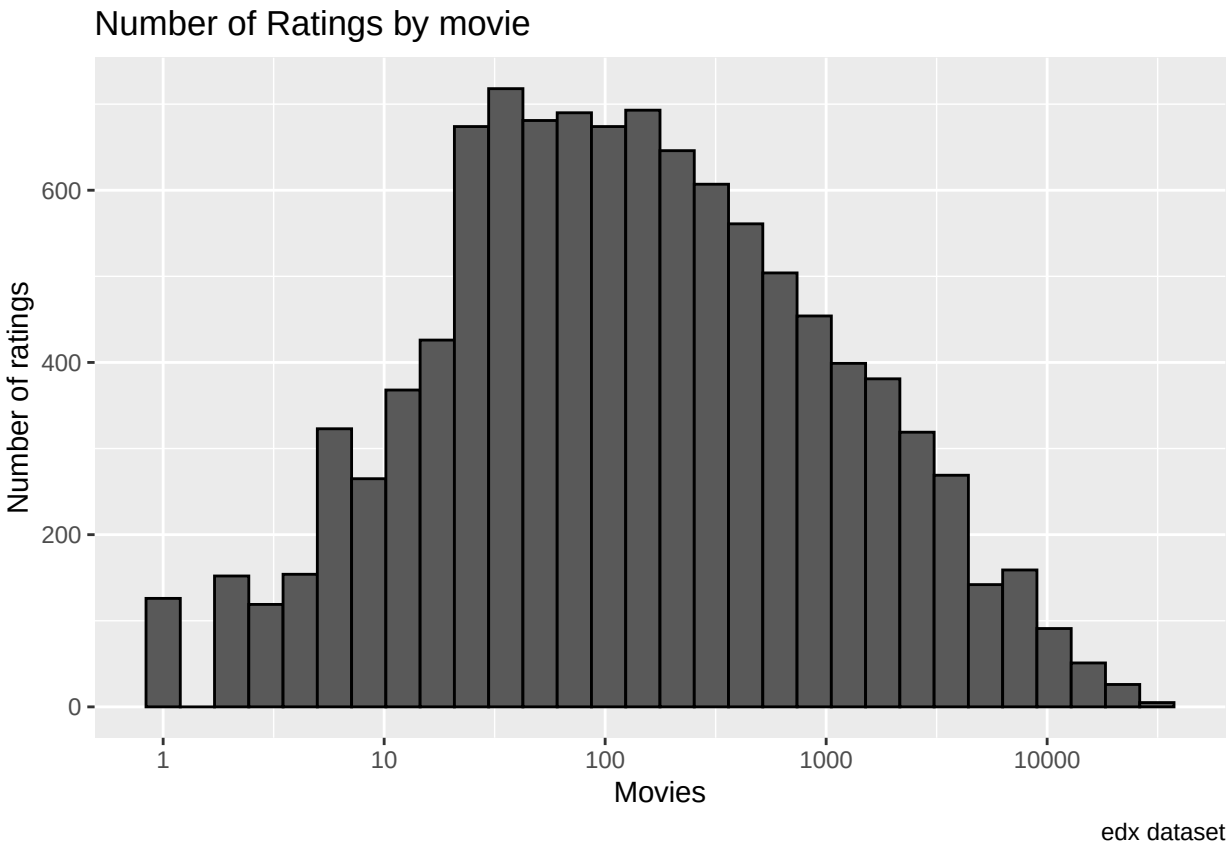
As the following figure indicates, the average rating by movie concentrates around the total average.

```
# Plot average rating by movie in the edx dataset
edx %>% group_by(movieId) %>%
  summarise(ave_rating = sum(rating)/n()) %>%
  ggplot(aes(ave_rating)) +
  geom_histogram(bins=30, color = I("black")) +
  labs(x = "Average rating", y = "Number of movies", caption = "edx dataset") +
  ggtitle("Movie distribution by average rating")
```



Plot the number of rating by movie.

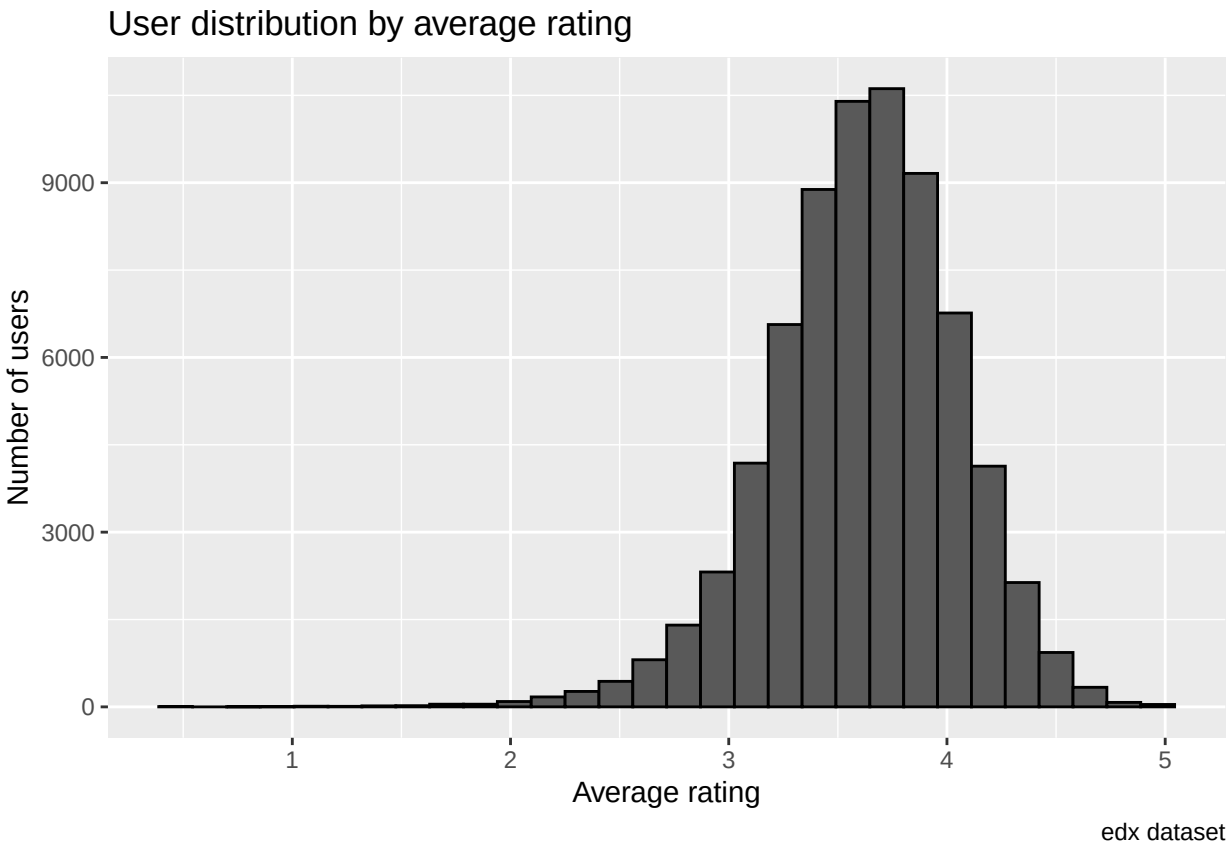
```
edx %>%
  count(movieId) %>%
  ggplot(aes(n)) +
  geom_histogram( bins=30, color = I("black"))+
  scale_x_log10() +
  labs(x = "Movies", y = "Number of ratings", caption = "edx dataset") +
  ggtitle("Number of Ratings by movie")
```



2.3 Users

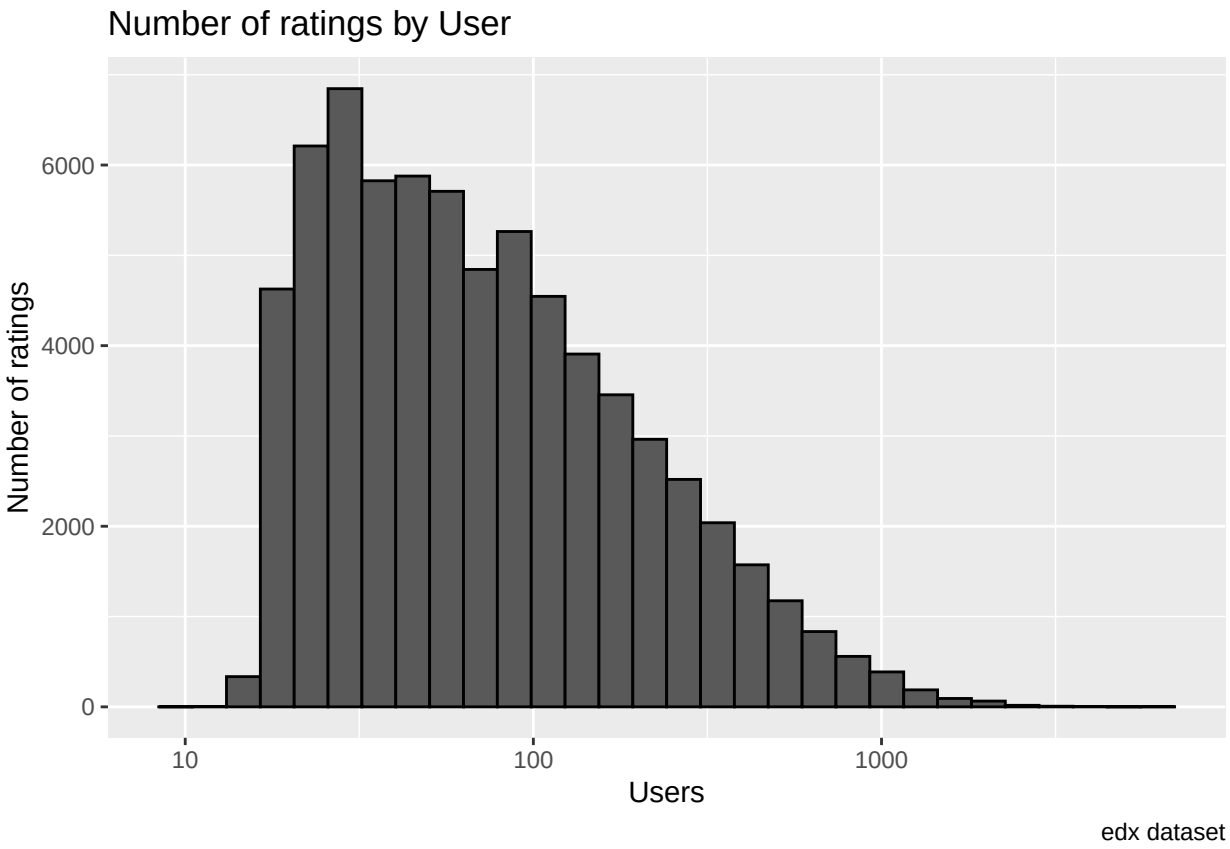
We can see users' distribution.

```
# Plot average rating by user in the edx dataset
edx %>% group_by(userId) %>%
  summarise(ave_rating = sum(rating)/n()) %>%
  ggplot(aes(ave_rating)) +
  geom_histogram(bins=30, color = I("black")) +
  labs(x = "Average rating", y = "Number of users", caption = "edx dataset") +
  ggtitle("User distribution by average rating")
```



Plot number of ratings by user in the edx dataset.

```
# Plot number of ratings by user in the edx dataset
edx %>%
  count(userId) %>%
  ggplot(aes(n)) +
  geom_histogram( bins=30, color = I("black")) +
  scale_x_log10() +
  labs(x = "Users", y = "Number of ratings", caption = "edx dataset") +
  ggtitle("Number of ratings by User")
```



2.4 Genre effect

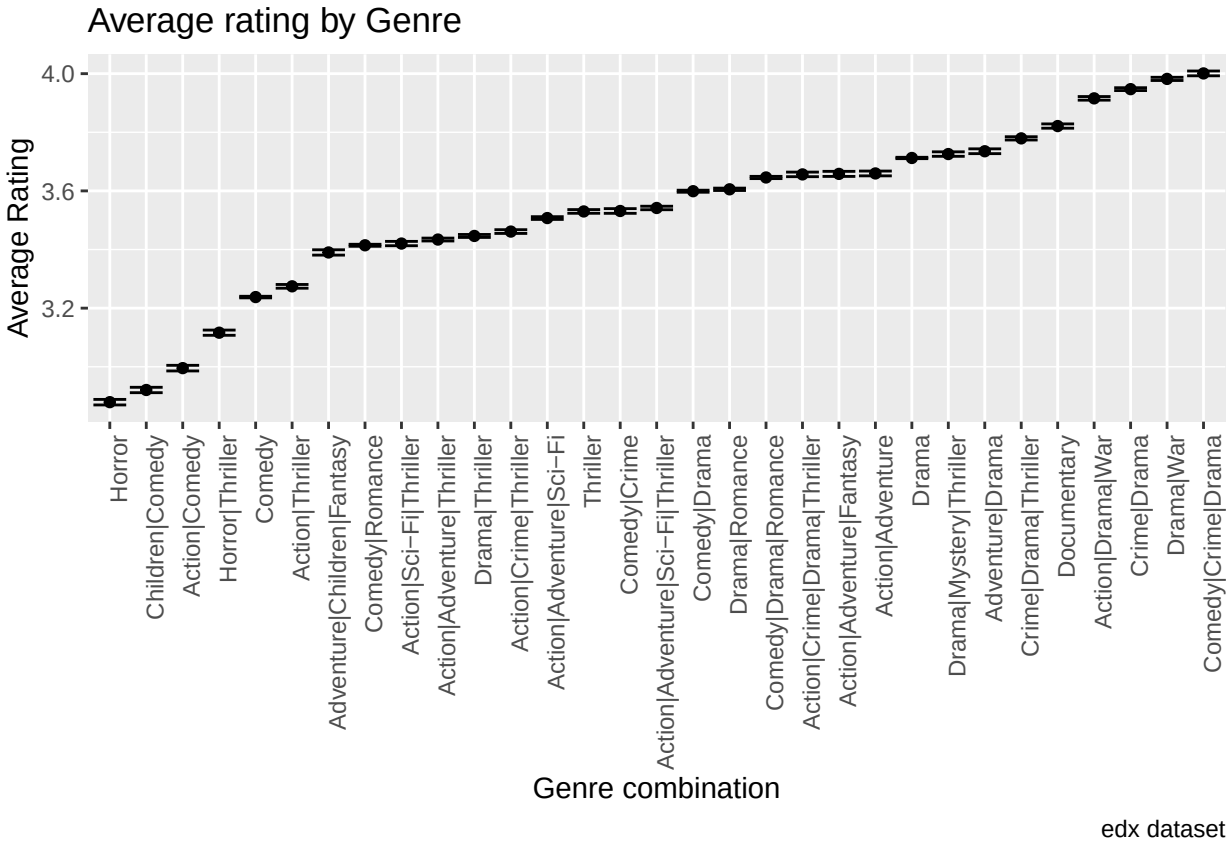
We can also see the head of genres.

```
## # A tibble: 6 x 2
##   genres                     n
##   <chr>                  <int>
## 1 Drama                 733296
## 2 Comedy                700889
## 3 Comedy|Romance        365468
## 4 Comedy|Drama          323637
## 5 Comedy|Drama|Romance  261425
## 6 Drama|Romance         259355
```

As the figure below shows, the average rating of genre combination varies. Thus, we suspect genres effect ratings.

```
# Plot average rating by genre for genre combinations with at least 50,000 ratings
edx %>% group_by(genres) %>%
  summarize(n = n(), avg = mean(rating), se = sd(rating)/sqrt(n())) %>%
  filter(n >= 50000) %>%
  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)) +
labs(x = "Genre combination", y = "Average Rating", caption = "edx dataset")+
ggtitle("Average rating by Genre")
```



3 Modeling and results

As we observed the effects of variables in the dataset, we focuse on Movies, UserID and genre as variables for rating and modeling with them.

Root mean Square Error, or RMSE is used to measure the differences between predicted values(y hat) as predicted rating of movie(i) by user(u) and observed values(y). If this number is larger than 1, it means our typical error is larger than 1 rating star. Which is not good. Which can write in code as below.

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

As the validation dataset was reserved for the final test, the edx dataset is necessary for both to train and test the algorithm in development. It would refrain the risk of over-training in cross-validation.

Check the the average rating of the train set.

```
## [1] 3.512456
```

Our first model is just the average model RMSE.

$$Y_{u,i} = \mu + \epsilon_{u,i}$$

We obtain the following RMSE:

```
# Our first model formula is Yu,i = mu + Eu,i
# If we predict all unknown ratings with mu, we obtain the following RMSE:
options(pillar.sigfig = 5)
naive_RMSE <- RMSE(test_set$rating, mu)
naive_RMSE
```

```
## [1] 1.060054
```

As above, the result of this model is the average of rating equal 3.5125. The first model, naive model, is 1.06005.

3.1 Movie Effect

Our second model: modeling movie effect formula will be adding movie bias as follow:

$$Y_{u,i} = \mu + b_i + \epsilon_{u,i}$$

```
# Our second model: modeling movie effect formula will be adding movie bias as follow: Yu,i = mu + b_i
mu <- mean(train_set$rating)
movie_avgs <- train_set %>%
  group_by(movieId) %>%
  summarize(b_i = mean(rating - mu))
```

```
options(pillar.sigfig = 5)
predicted_ratings <- mu + test_set %>%
  left_join(movie_avgs, by='movieId') %>%
  pull(b_i)
model2 <- RMSE(predicted_ratings, test_set$rating)
model2
```

```
## [1] 0.9429615
```

The Movie effect model RMSE is 0.94296.

3.2 Movie + User Effect

Our third one is movie plus user effect model.

$$Y_{u,i} = \mu + b_i + b_u + \epsilon_{u,i}$$

```
user_avgs <- train_set %>%
  left_join(movie_avgs, by='movieId') %>%
  group_by(userId) %>%
  summarise(b_u = mean(rating - mu - b_i))
```

```
## [1] 0.8646843
```

3.3 Movie + User + Genre Effect

Our fourth model is movie, user and genre effected model.

$$Y_{u,i} = \mu + b_i + b_u + b_g + \epsilon_{u,i}$$

```
# Estimate genre effect (b_g)
options(pillar.sigfig = 5)
genre_avgs <- train_set %>%
  left_join(movie_avgs, by = "movieId") %>%
  left_join(user_avgs, by = "userId") %>%
  group_by(genres) %>%
  summarise(b_g = mean(rating - mu - b_i - b_u))
# Predict ratings adjusting for movie, user and genre effects
predicted_ratings <- test_set %>%
  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) %>%
  pull(pred)
# Calculate RMSE based on genre effects model
model4 <- RMSE(predicted_ratings, test_set$rating)
model4
```

```
## [1] 0.8643241
```

3.4 Regularized Movie + User + Genre Effect Model

For our fifth model, as the final model, we are going to add regularization to improve our predicted RMSE. We can use regularization for the estimate movie + user effects as well. We are minimizing:

$$\frac{1}{N}\sum_{u,i}\left(y_{u,i} - \mu - b_i - b_u - b_g\right)^2 + \lambda\left(\sum_i b_i^2 + \sum_u b_u^2 + \sum_g b_g^2\right)$$

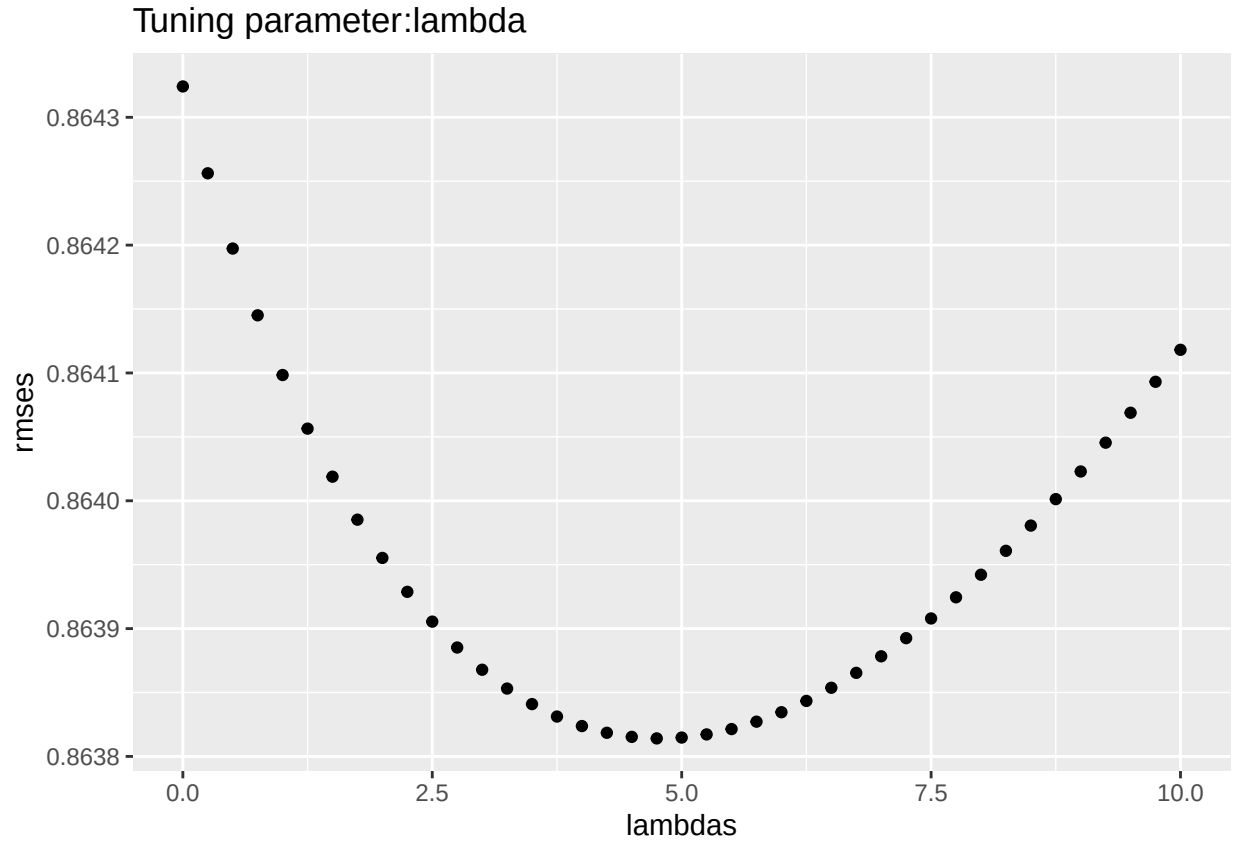
Our final model: Regularized Movie Effect + User Effect + Genre Effect Model

$$Y_{u,i} = \mu + b_i + b_u + b_g + \epsilon_{u,i}$$

Choosing the best penalty terms. Lambda is a tuning parameter. We can use cross-validation to choose it to get our optimized lambda. Here is the plot of lambda by increment of 0.25 from 0 to 10, and optimized lambda value.

```
set.seed(1)
lambdas <- seq(0, 10, 0.25)
rmses <- sapply(lambdas, function(l){

  mu <- mean(train_set$rating)

  b_i <- train_set %>%
    group_by(movieId) %>%
    summarise(b_i = sum(rating - mu)/(n()+l))

  b_u <- train_set %>%
```

```
    left_join(b_i, by="movieId") %>%
    group_by(userId) %>%
    summarise(b_u = sum(rating - b_i - mu)/(n()+l))

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

  predicted_ratings <- test_set %>%
    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, test_set$rating))
})
qplot(lambdas, rmses)+ggtitle("Tuning parameter:lambda")
```




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

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

3.5 Regularized Movie + User + Genre Effect model to the Validation set.

Apply the regularized final model with the lambda of 4.75 to the validation set.

```
set.seed(1)
options(pillar.sigfig = 5)
lambda <- lambda
mu <- mean(edx$rating)
b_i <- edx %>%
  group_by(movieId) %>%
  summarise(b_i = sum(rating - mu)/(n()+lambda))
b_u <- edx %>%
  left_join(b_i, by="movieId") %>%
  group_by(userId) %>%
  summarise(b_u = sum(rating - b_i - mu)/(n()+lambda))
b_g <- train_set %>%
  left_join(b_i, by="movieId") %>%
  left_join(b_u, by="userId") %>%
  group_by(genres) %>%
  summarise(b_g = sum(rating - b_i - b_u - mu)/(n()+lambda))
predicted_ratings <- validation %>%
  left_join(b_i, by = "movieId") %>%
  left_join(b_u, by = "userId") %>%
  left_join(b_g, by = "genres") %>%
  mutate(pred = mu + b_i + b_u + b_g) %>%
  pull(pred)
finalmodel_validation_rmse <- RMSE(predicted_ratings, validation$rating)
finalmodel_validation_rmse
```

```
## [1] 0.8644561
```

Overall results are as follows:

```
options(pillar.sigfig = 5)
rmse_results <- data.frame(method = "RMSE Goal", RMSE = 0.86490)
finalmodel_validation_rmse <- bind_rows(rmse_results,
                          data.frame(method = "Regularized Movie   User + Genre model on Validation set"
                                     RMSE = finalmodel_validation_rmse))
finalmodel_validation_rmse
```

```
##                                                    method      RMSE
## 1                                              RMSE Goal 0.8649000
## 2 Regularized Movie   User + Genre model on Validation set 0.8644561
```

4 Conclusion

We achieved RMSE less than 0.86490 with our final model, the regularized Movie, User and Genre effected model in the validation set. The limitation of this project is derived from that we did not calculate the inter-variable contribution for the modeling. For future work, for example, we should find out inter variation correlation on how each variable contribute and effect.

5 Reference

Irizarry A. Rafael (2018) Introduction to Data Science: Data Analysis and Prediction Algorithms with R.