MovieLens Project

Rahma Ali

01/02/2019

This project is submitted in partial fulfillment of the requirements for obtaining HarvardX Professional Certificate of Data Science, offered via EdX.

1. Introduction and Project Motivation

This project aims at creating a movie recommendation system for users. It uses data on users, their ratings on different movies and movie genres to suggest the highest rated movie.The data for this project is available online through: https://grouplens.org/datasets/movielens/10m.

At first, the data is downloaded from the web.

```
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(levels(movieId))[movieId],
                                           title = as.character(title),
                                           genres = as.character(genres))

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

Create training and validation sets

Training (edx) and validation datasets are created. Validation set is 10% of the entire dataset.

```
# 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)
```

```
## Joining, by = c("userId", "movieId", "rating", "timestamp", "title", "genres")
```

```
edx <- rbind(edx, removed)
rm(dl, ratings, movies, test_index, temp, movielens, removed)
```

Explore the training data

Now, we take an overall look on the edx data.

```
str(edx)
```

```
## 'data.frame':    9000055 obs. of  6 variables:
##  $ userId   : int  1 1 1 1 1 1 1 1 1 1 ...
##  $ movieId  : num  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 838984885 8
##  $ title    : chr  "Boomerang (1992)" "Net, The (1995)" "Outbreak (1995)" "Stargate (1994)" ...
##  $ genres   : chr  "Comedy|Romance" "Action|Crime|Thriller" "Action|Drama|Sci-Fi|Thriller" "Action|A
```

The data contains 9,000,055 records and 6 variables, including 1 response variable (rating) and 5 features.

```
n_distinct(edx$userId)
```

```
## [1] 69878
```

```
n_distinct(edx$movieId)
```

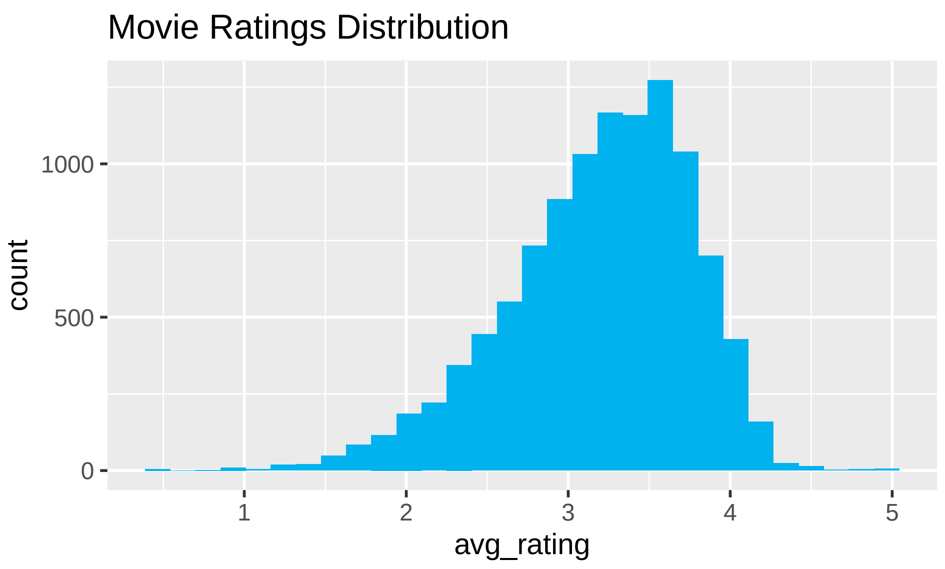
```
## [1] 10677
```

The data includes ratings on 10,677 movies from 69,878 unique users.

The distribution of the movie ratings is bell shaped and skewed to the left a little, suggesting presence of small number of very low rated movies. The average movie rating is 3.5124652.

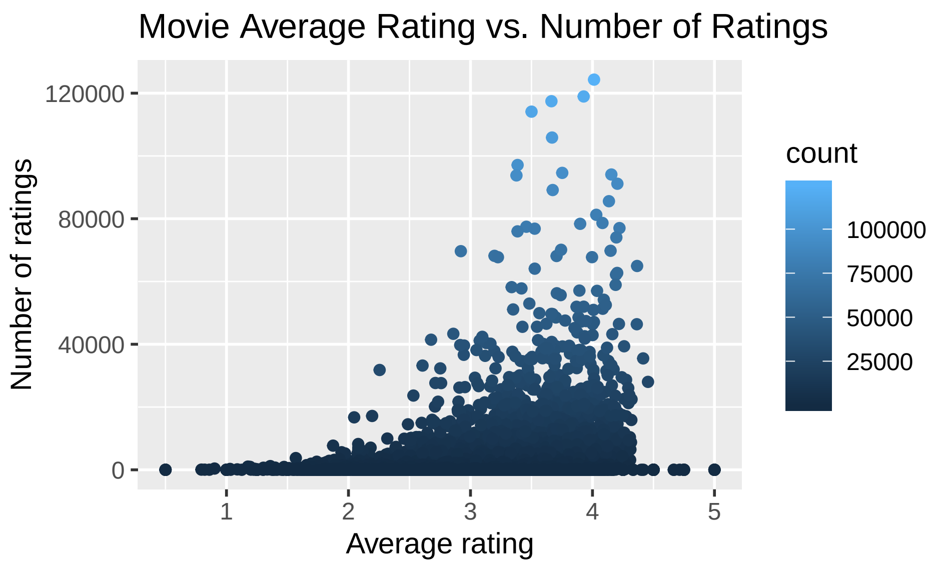
```
edx %>% group_by(movieId) %>%
  summarize(count=n(), avg_rating=mean(rating)) %>%
  ggplot(aes(avg_rating)) +
  geom_histogram(fill="deepskyblue2") +
  ggtitle("Movie Ratings Distribution")
```

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



Exploring rating variability by movie

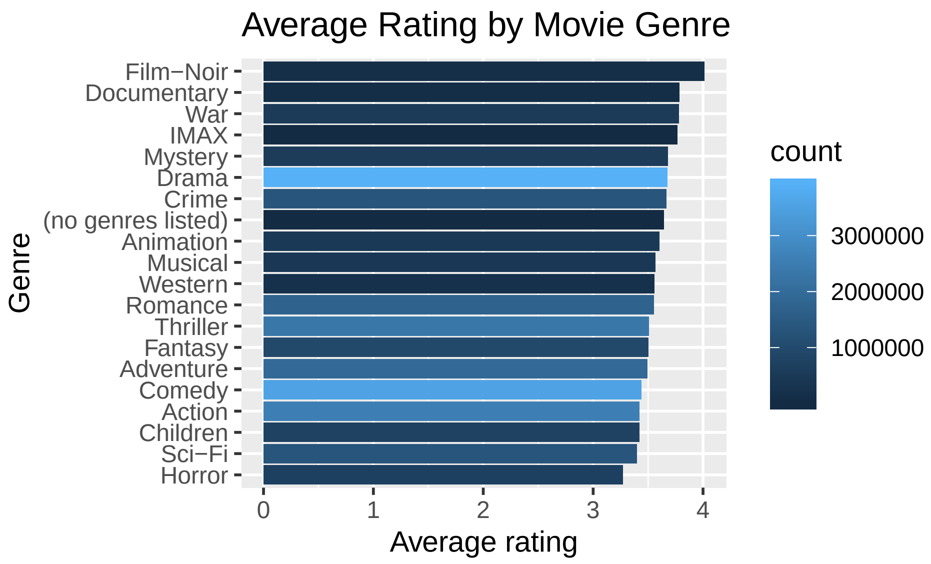
```r
# Exploring rating variability by number of ratings
edx %>% separate_rows(genres, sep = "\\|") %>%
  group_by(movieId) %>%
  summarize(count=n(), avg_rating=mean(rating), genre=first(genres)) %>%
  ggplot(aes(avg_rating, count, col=count)) +
  geom_point() +
  ggtitle("Movie Average Rating vs. Number of Ratings ") +
  labs(y="Number of ratings", x="Average rating")
```



Movie ratings seem to be associated with their rating frequency.

Exploring rating variability by movie genre:

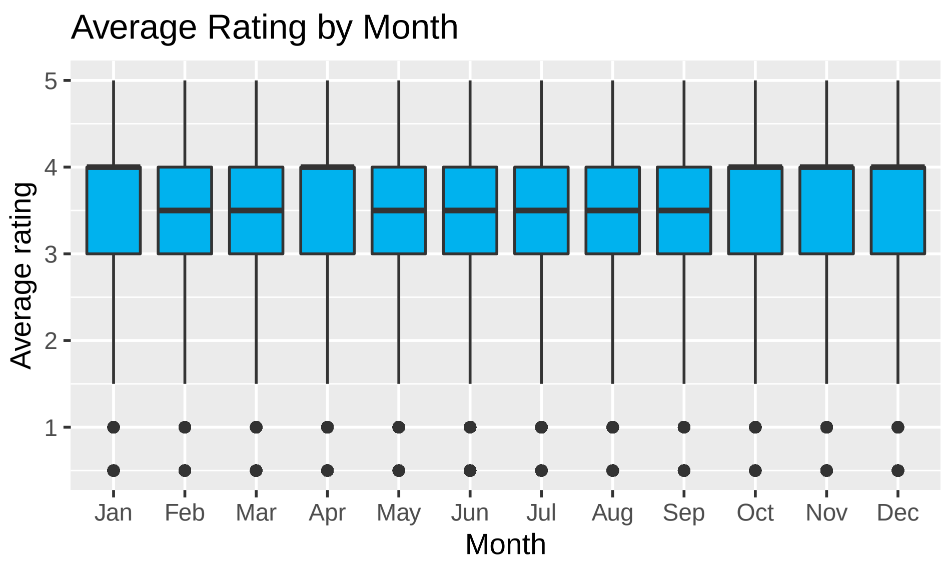
```
edx %>% separate_rows(genres, sep = "\\|") %>%
  group_by(genres) %>%
  summarize(count = n(), avg_ratings=mean(rating)) %>%
  ggplot(aes(reorder(genres, avg_ratings), avg_ratings, fill=count)) +
  geom_bar(stat="identity") +
  coord_flip() +
  scale_color_hue(direction = -1, h.start=90) +
  ggtitle("Average Rating by Movie Genre") +
  labs(x="Genre", y="Average rating")
```



The genre 'Film-Noir' received the highest average rating among all genres. However, 'Drama' movies were rated the most.

Exploring seasonality of movie ratings:

```
edx %>%
  mutate(month=factor(month.abb[month(as.Date(as.POSIXlt(timestamp, origin="1970-01-01")))])) %>%
  ungroup() %>%
  arrange(month) %>%
  ggplot(aes(month,rating)) +
  geom_boxplot(fill="deepskyblue2") +
  labs(x="Month", y="Average rating") +
  ggtitle("Average Rating by Month") +
  scale_x_discrete(limits = month.abb)
```



The distrubution of the rating does not seem to vary drastically across the months of the year. We see relatively higher ratings in Jan, Apr, Oct, Nov and Dec but, in general, it does not seem that there is a descernable trend either monthly or seasonaly affecting movie ratings.

2. Method and Analysis

A model is considered that takes into account movie and user effects, since these two aspect seem to have the highest impact on rating variability as seen from the exploratory analysis above.

$rating_{i,u} = \mu + \beta_i + \beta_u + \epsilon_{i,u}$

where β_i is the movie effect, β_u is the user effect and ϵ is the model error term.

Train the model on the training data:

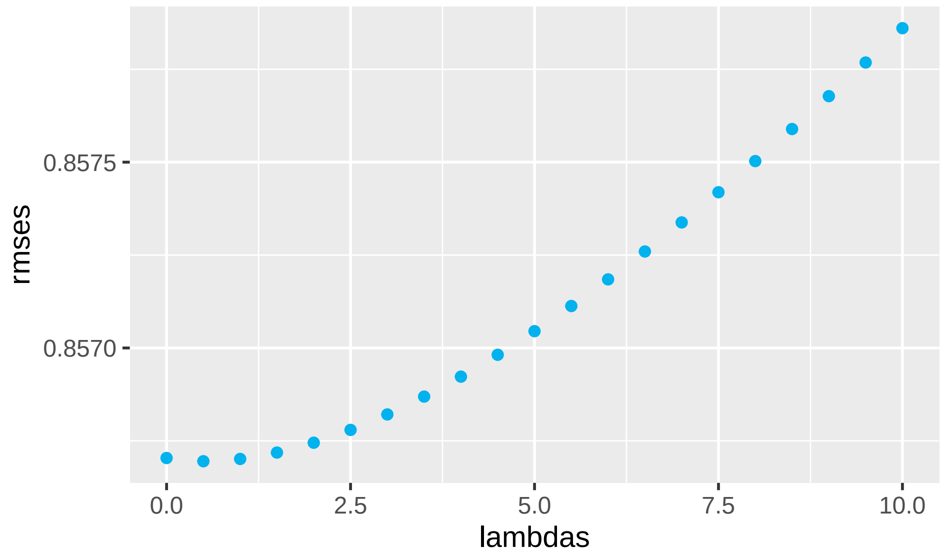
```r
lambdas <- seq(0, 10, 0.5)
mu_hat <- mean(edx$rating)

rmses <- sapply(lambdas, function(l){
  movie_avgs <- edx %>%
    group_by(movieId) %>%
    summarize(b_i = sum(rating - mu_hat)/(n()+l), n_i = n())

  user_avgs <- edx %>%
    left_join(movie_avgs, by='movieId') %>%
    group_by(userId) %>%
    summarize(b_u=sum(rating-mu_hat-b_i)/(n()+l), n_i = n())

  predicted_ratings <- edx %>%
    left_join(movie_avgs, by='movieId') %>%
    left_join(user_avgs, by='userId') %>%
    mutate(pred = mu_hat + b_i +b_u) %>%
    pull(pred)
  return(RMSE(predicted_ratings, edx$rating))
})

qplot(lambdas, rmses, colour=I("deepskyblue2"))
```



RMSE is minimized at 0.5

Apply the model on the validation set

```
# Apply the model on the validation set
mu <- mean(validation$rating)
lambda <- lambdas[which.min(rmses)]

# Movie effect
b_i <- validation %>%
  group_by(movieId) %>%
  summarize(b_i = sum(rating - mu)/(n() + lambda))

# User effect
b_u <- validation %>%
  left_join(b_i, by='movieId') %>%
  group_by(userId) %>%
  summarize(b_u = sum(rating - b_i - mu)/(n() + lambda))

# Generate predictions
predicted_ratings <- validation %>%
  left_join(b_i, by = "movieId") %>%
  left_join(b_u, by = "userId") %>%
  mutate(pred = mu + b_i +  b_u)

# Calculate RMSE for the final model
RMSE(predicted_ratings$pred, validation$rating)
```

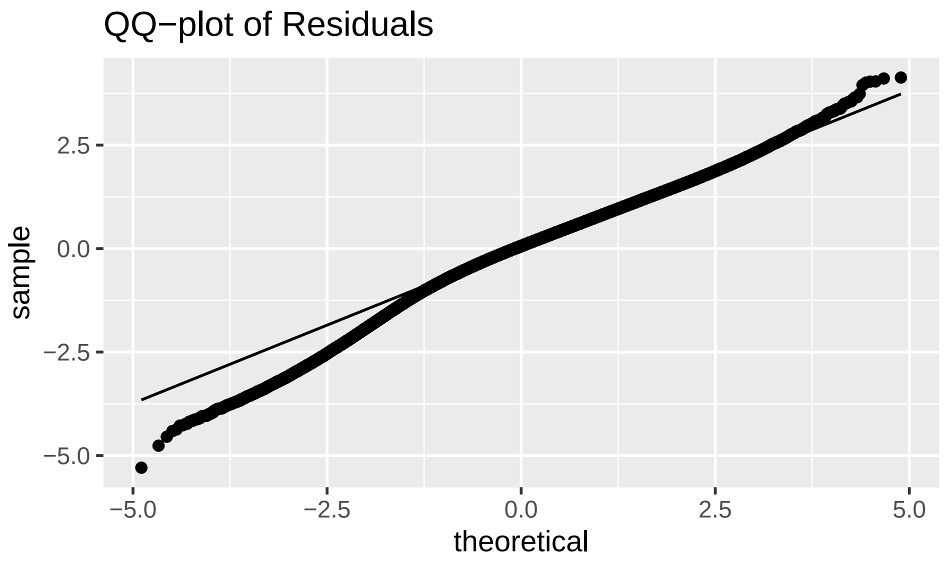
```
## [1] 0.8258487
```

The RMSE of the model is 0.8258487

Model residuals

We take a look at the model residuals as a model diagnostics check.

```
# QQ plot of residuals
predicted_ratings %>%
  mutate(residuals=rating-pred) %>%
  ggplot(aes(sample=residuals)) +
  stat_qq(fill="deepskyblue2") +
  stat_qq_line() +
  ggtitle("QQ-plot of Residuals")
```

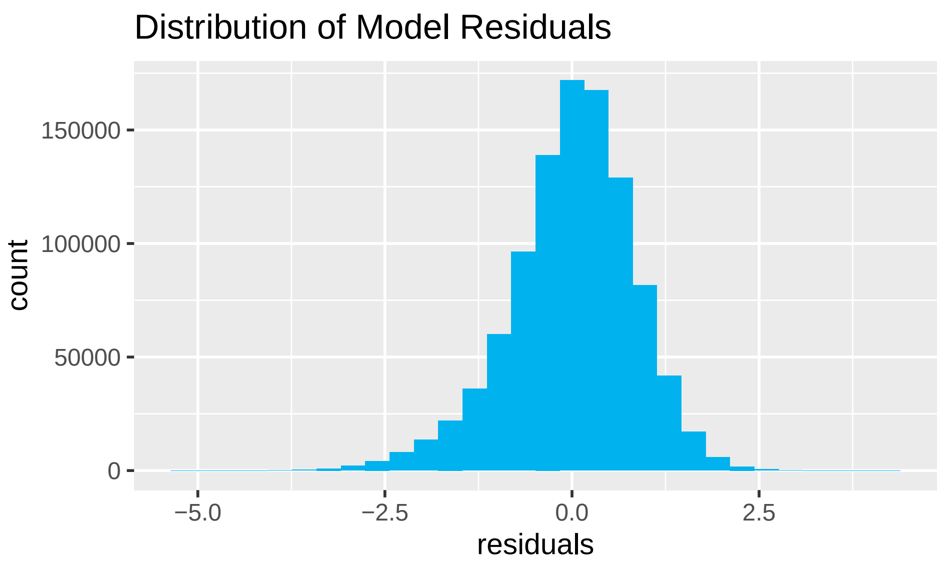


When observing the qq-plot of the residuals, there seems to be some deviation from the normal distribution quantiles. This suggests that the model residuals are not quiet normally distributed.

The histogram of the residuals suggest the same thing. Although the distribution is symmetric and bell-shaped, the spread is different than that of a normal distribution.

```
# Distribution of residuals
predicted_ratings %>%
  mutate(residuals=rating-pred) %>%
  ggplot(aes(residuals)) +
  geom_histogram(fill="deepskyblue2") +
  ggtitle("Distribution of Model Residuals")
```

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



3. Results and Conclusions

The final model includes user and movie effects. Additional attempts were made to include genre effect but the RMSE of the final model did not improve. As monthly variation of movie ratings was explored earlier, another attempt was made to include a month effect term by generating a month variable from the `timestamp` variable. This attempt also did not result in improved RMSE in the final model. Both attempts are not included in this report.