Movie Recommendation Model - edx

Suliman Alkhalifa

12 April 2019

Project Overview

For this project, the objective is to build a recommendation system for users to watch movies. I am happy to be working in such a project as I like watching movies and have always been wondering how the system knows what I like and don't like. Now, it is an opportunity to build a similar system and learn from my peers. After searching for best practices on how to build an accurate model/system for recommending movies, I decided for this project to use three different models which I learned from previous courses and then compare the results to see which one is the best.

Used Libraries

I will use the following libraries for my project:

```r
library(tidyverse)
```

```
## -- Attaching packages ---------------------------------------------------------- tidyverse 1.2.1 --

## v ggplot2 3.1.1       v purrr   0.3.2
## v tibble  2.1.1       v dplyr   0.8.0.1
## v tidyr   0.8.3       v stringr 1.4.0
## v readr   1.3.1       v forcats 0.4.0

## -- Conflicts ------------------------------------------------------------- tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```r
library(caret)
```

```
## Loading required package: lattice

##
## Attaching package: 'caret'

## The following object is masked from 'package:purrr':
##
##     lift
```

```r
library(ggplot2)
```

Data Gathering

The dataset used was from Movielens as suggested in the course and I used the same codes provided to download the data.

```r
dl <- tempfile()
download.file("http://files.grouplens.org/datasets/movielens/ml-10m.zip", dl)
```

Data Pre-processing

I will use the ratings and movies files from the list of files downloaded.

```r
ratings <- read.table(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)
```

I will add column names to movies data and change their classes as follow:

```r
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))
```

Next is to join the ratings and movies datasets into movielens dataset:

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

Data Exploration

Let us take a look at some of the dataset features and insights

First is to view how many rows and columns:

```r
nrow(movielens)
```

```
## [1] 10000054
```

```r
ncol(movielens)
```

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

Second is to look at summary of the movies and ratings:

```r
summary(movielens)
```

```
##      userId         movieId          rating        timestamp
##  Min.   :    1   Min.   :    1   Min.   :0.500   Min.   :7.897e+08
##  1st Qu.:18123   1st Qu.:  648   1st Qu.:3.000   1st Qu.:9.468e+08
##  Median :35741   Median : 1834   Median :4.000   Median :1.035e+09
##  Mean   :35870   Mean   : 4120   Mean   :3.512   Mean   :1.033e+09
##  3rd Qu.:53608   3rd Qu.: 3624   3rd Qu.:4.000   3rd Qu.:1.127e+09
##  Max.   :71567   Max.   :65133   Max.   :5.000   Max.   :1.231e+09
##     title              genres
##  Length:10000054    Length:10000054
##  Class :character   Class :character
##  Mode  :character   Mode  :character
##
##
##
```

Next is to see how many unique movies, ratings and users we have:

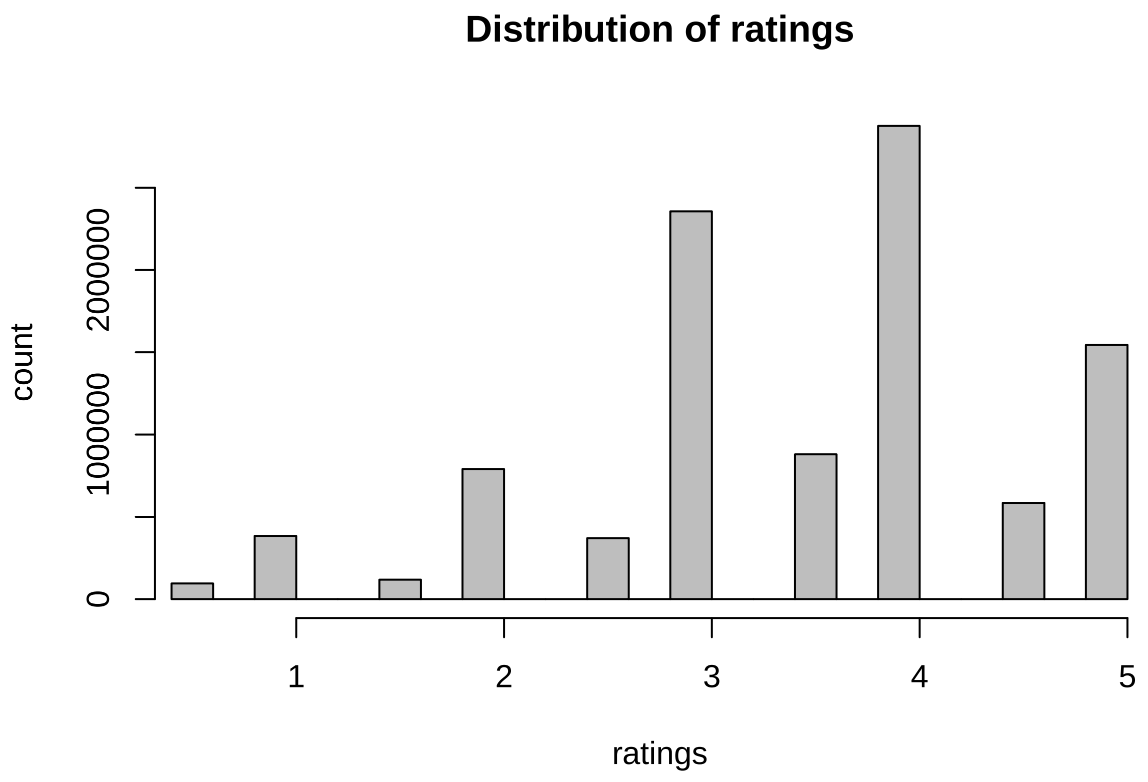
```r
movielens %>%
  summarize(n_users = n_distinct(userId),
            n_movies = n_distinct(movieId),
            n_ratings = n_distinct(rating))
```

```
##   n_users n_movies n_ratings
## 1   69878    10677        10
```

The number of users exceeds the number of movies which indicates that users may have rated more than one movie at different times.

Next is to see the distribution of ratings:

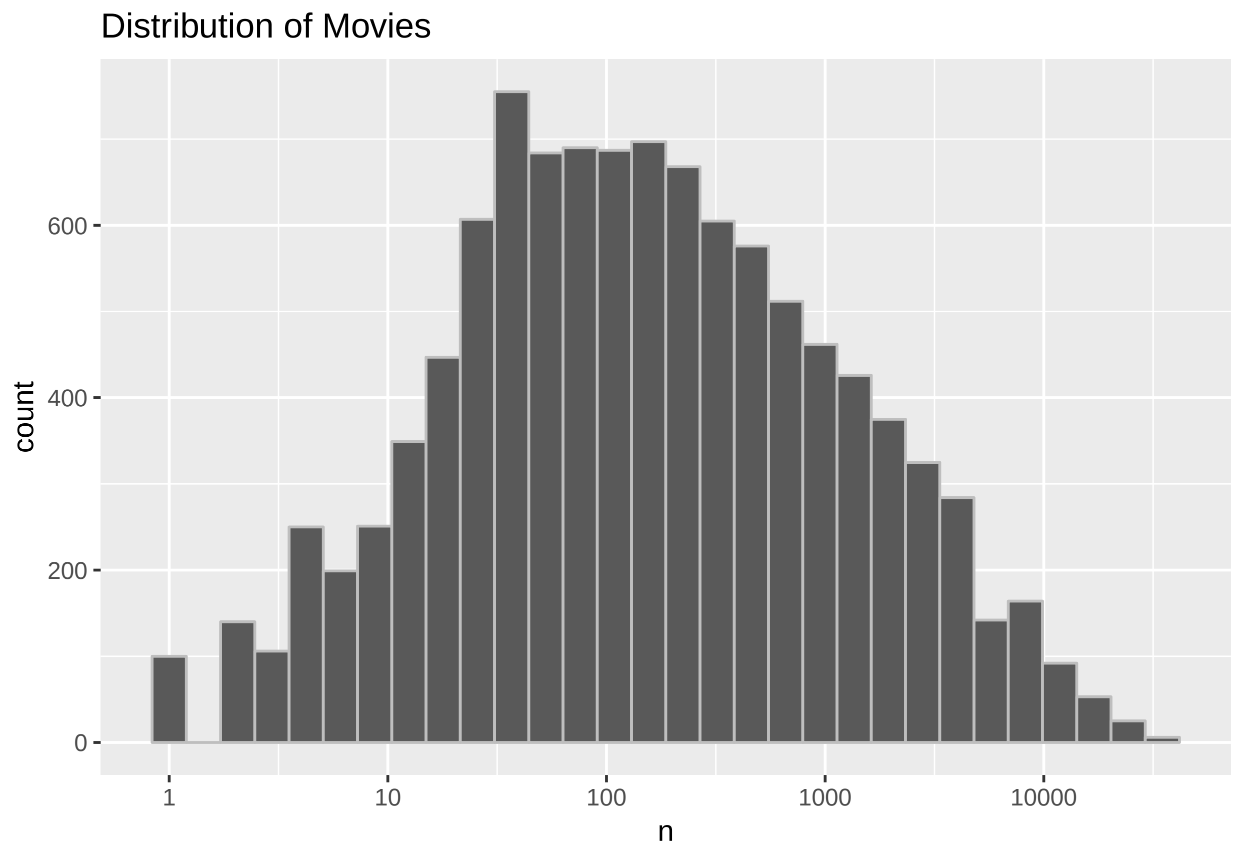
```r
hist(movielens$rating, col = "grey", main = "Distribution of ratings",
     xlab = "ratings", ylab = "count", xlim = range(0.5,5))
```



We can see that majority of ratings start from 3 and above and the most common rating is 4. Also, half star ratings are less common than full star ratings.

In addition, we can look at the distribution of movies:

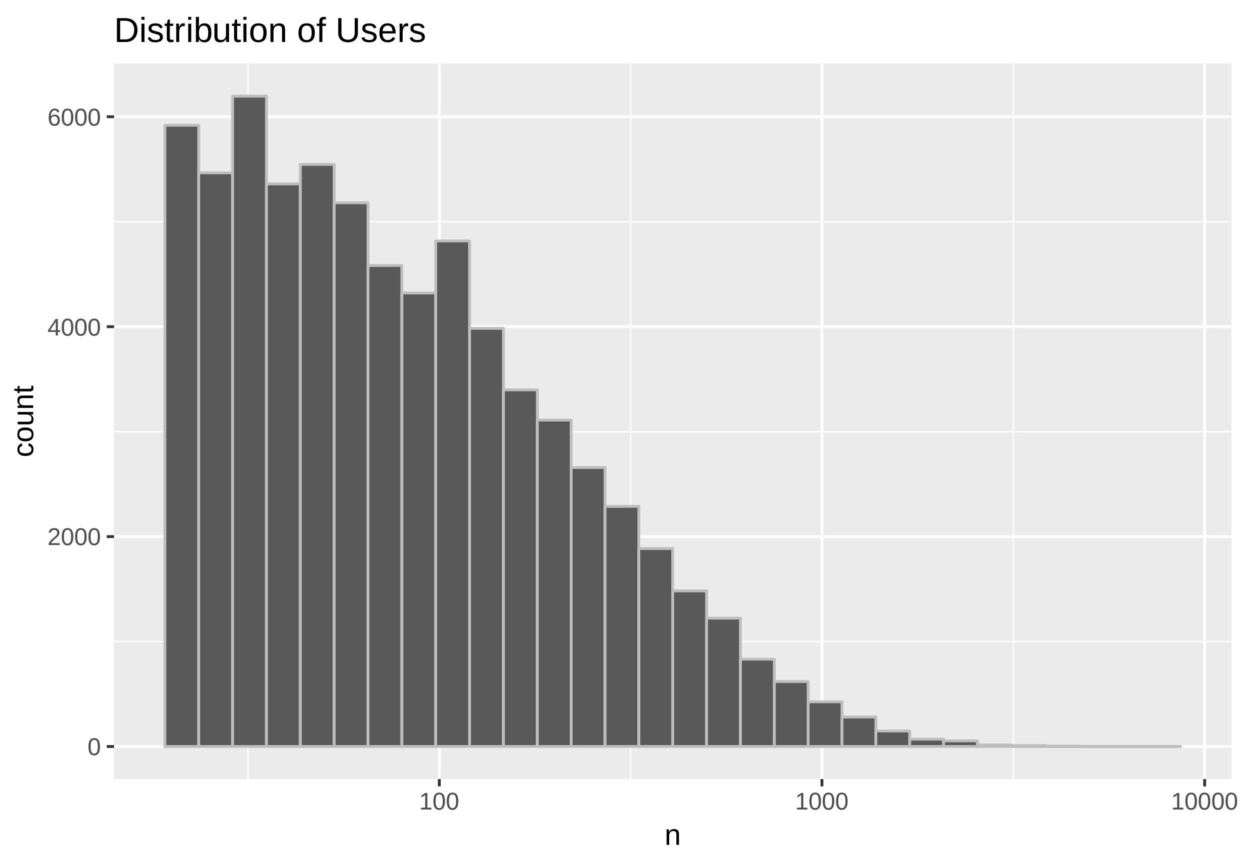
```r
movielens %>% count(movieId) %>% ggplot(aes(n)) +
  geom_histogram(bins = 30, color = "grey") + scale_x_log10() +
  ggtitle("Distribution of Movies")
```



This shows that some movies are rated many times as they are popular and many users have rated them.

We will also look at the distribution of users:

```
movielens %>% count(userId) %>% ggplot(aes(n)) +
  geom_histogram(bins = 30, color = "grey") + scale_x_log10() +
  ggtitle("Distribution of Users")
```



This indicates that some users are more active than others.

The next code is to show a list of top movies rated:

```
movielens %>% group_by(movieId, title) %>%
  summarize(count = n()) %>%
  arrange(desc(count))
```

```
## # A tibble: 10,677 x 3
## # Groups:   movieId [10,677]
##    movieId title                                                        count
##      <dbl> <chr>                                                        <int>
##  1     296 Pulp Fiction (1994)                                          34864
##  2     356 Forrest Gump (1994)                                          34457
##  3     593 Silence of the Lambs, The (1991)                             33668
##  4     480 Jurassic Park (1993)                                         32631
##  5     318 Shawshank Redemption, The (1994)                             31126
##  6     110 Braveheart (1995)                                            29154
##  7     457 Fugitive, The (1993)                                         28951
##  8     589 Terminator 2: Judgment Day (1991)                            28948
##  9     260 Star Wars: Episode IV - A New Hope (a.k.a. Star Wars) (19~ 28566
## 10     150 Apollo 13 (1995)                                             27035
## # ... with 10,667 more rows
```

Data Preparation

First I will create the training and testing set. I will use 10% of the dataset for the testing and 90% for the training set.

```r
set.seed(1)
test_index <- createDataPartition(y = movielens$rating, times = 1, p = 0.1, list = FALSE)
train_set <- movielens[-test_index,]
temp <- movielens[test_index,]
```

Make sure the userId and movieId in the testing set are also in the training set.

```r
test_set <- temp %>%
  semi_join(train_set, by = "movieId") %>%
  semi_join(train_set, by = "userId")
```

Add rows removed from testing set back to training set

```r
removed <- anti_join(temp, test_set)
```

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

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

Building The Recommendation Model

I will use different models to compare their RMSE and decide which one is the best.

Just the average model

The first model is the simplest or using just the average of all ratings.

I will calculate the average of all ratings in the training set.

```r
mu <- mean(train_set$rating)
mu
```

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

Next is to calculate the first RMSE. To do that I will use all ratings in the test_set as predictions.

```r
rmse_1 <- RMSE(test_set$rating, mu)
```

I will use a table to compare the RMSE results from different methods used.

```r
rmse_table <- data_frame(method = "Just the average", RMSE = rmse_1)
```

```
## Warning: `data_frame()` is deprecated, use `tibble()`.
## This warning is displayed once per session.
```

```r
rmse_table
```

```
## # A tibble: 1 x 2
##   method            RMSE
##   <chr>            <dbl>
## 1 Just the average  1.06
```

Movie effect model

Since different movies are rated differently, we need to calculate b_i (bias) which is the average ranking of each movie.

```r
movie_avgs <- train_set %>% group_by(movieId) %>% summarize(b_i = mean(rating - mu))
```

Now I will calculate the second RMSE to see if the predictions improve.

```r
predicted_ratings <- mu + test_set %>% left_join(movie_avgs, by= 'movieId') %>% .$b_i
rmse_2 <- RMSE(predicted_ratings, test_set$rating)
```

I will add the second RMSE to the table to compare it with the first one.

```r
rmse_table <- bind_rows(rmse_table, data_frame(method = "Movie effect model", RMSE = rmse_2))
rmse_table
```

```
## # A tibble: 2 x 2
##   method              RMSE
##   <chr>              <dbl>
## 1 Just the average   1.06
## 2 Movie effect model 0.944
```

The RMSE of the second model is a little bit less than the first one but we still need to find a better model.

Movie and User Effects Model

Similar to what we did with the movies, I will calculate the average rating for each user and add it to the previous model.

```r
user_avgs <- test_set %>% left_join(movie_avgs, by='movieId') %>% group_by(userId) %>% summarize(b_u = m
```

Now I will calculate the third RMSE to see the results.

```r
predicted_ratings <- test_set %>% left_join(movie_avgs, by='movieId') %>% left_join(user_avgs, by='userI
rmse_3 <- RMSE(predicted_ratings, test_set$rating)
```

I will add the third RMSE to the table.

```r
rmse_table <- bind_rows(rmse_table, data_frame(method = "Movie + User Effects Model", RMSE = rmse_3))
rmse_table
```

```
## # A tibble: 3 x 2
##   method                      RMSE
##   <chr>                      <dbl>
## 1 Just the average           1.06
## 2 Movie effect model         0.944
## 3 Movie + User Effects Model 0.829
```

Evaluation of models

Looking at the results of RMSE in the table we can see that using the Movie and User Effect Model can generate more accurate results. The final step is to fine-tune the predicted ratings so that the minimum and maximum ratings are equal to those in the training set.

```r
predicted_ratings[which(predicted_ratings<1)] <- min(train_set$rating)
predicted_ratings[which(predicted_ratings>5)] <- max(train_set$rating)
```

Conclusion

Building a recommendation system for movies requires the consideration of different variabilities such as users, movies, genres, etc. For this project, I decided to use the movie and user effect model as it takes into account the differences between movies and users. This has resulted in an RMSE of around 0.82. However, there are other models that can be used to further improve the accuracy of our recommendation system and we can always test these models with larger set of data for different products.