Movie Lens EDX Project

Introduction

A recommendation system is a filtering system that predicts the rating or preference of an user. Recommendation system uses rating that user have given to make recommendation. Companies will use ratings the customers have given to predict their recommendations on other items. Streaming services like Netflix use recommendation systems that was inspired by winners of a challenge Netflix put in 2006. The challenge was to improve the recommendation algorithm by 10%.

Data set

The data was from GroupLens Research and can be found at http://movielens.org

Loading the data set

The code at the beggining of the project was given by the EDX course to split the data into training and test set and have 10% validation.

```
library(tidyverse)
```

```
## -- Attaching packages --------------------------------------- tidyverse 1.3.1 --
```

```
## v ggplot2 3.3.3     v purrr   0.3.4
## v tibble  3.1.2     v dplyr   1.0.6
## v tidyr   1.1.3     v stringr 1.4.0
## v readr   1.4.0     v forcats 0.5.1
```

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

```
library(caret)
```

```
## Loading required package: lattice
```

```
##
## Attaching package: 'caret'
```

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

```
library(data.table)
```

```
##
## Attaching package: 'data.table'
```

```
## The following objects are masked from 'package:dplyr':
##
##     between, first, last
```

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

```
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")
```

```
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 = 0.2, list = FALSE)
edx <- movielens[-test_index,]
temp <- movielens[test_index,]
```

```
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)
```

Summary of the Data Set

Here we take a look at the first 6 rows of the data set and the structure of the data set as well. The output for str(edx) we can see there are 6 variables which are userId, movieId,rating, timestamp, title, and genres. We can also see the Min, 1st Qu, Median, 3rd Qu, Max for userId, movieId, rating, and timestamp. For tile and genres we see the Length, Class and Mode.

```
head(edx)
```

```
##    userId movieId rating timestamp                    title
## 1:      1     231      5 838983392    Dumb & Dumber (1994)
## 2:      1     316      5 838983392         Stargate (1994)
## 3:      1     355      5 838984474 Flintstones, The (1994)
## 4:      1     356      5 838983653     Forrest Gump (1994)
## 5:      1     364      5 838983707   Lion King, The (1994)
## 6:      1     377      5 838983834            Speed (1994)
##                           genres
## 1:                        Comedy
## 2:         Action|Adventure|Sci-Fi
## 3:         Children|Comedy|Fantasy
## 4:        Comedy|Drama|Romance|War
## 5: Adventure|Animation|Children|Drama|Musical
## 6:         Action|Romance|Thriller
```

```
str(edx)
```

```
## Classes 'data.table' and 'data.frame':   8000071 obs. of  6 variables:
##  $ userId   : int  1 1 1 1 1 1 1 1 1 1 ...
##  $ movieId  : num  231 316 355 356 364 377 420 466 586 588 ...
##  $ rating   : num  5 5 5 5 5 5 5 5 5 5 ...
##  $ timestamp: int  838983392 838983392 838984474 838983653 838983707 838983834 838983834 838984679 838984068 8
38983339 ...
##  $ title    : chr  "Dumb & Dumber (1994)" "Stargate (1994)" "Flintstones, The (1994)" "Forrest Gump (1994)"
...
##  $ genres   : chr  "Comedy" "Action|Adventure|Sci-Fi" "Children|Comedy|Fantasy" "Comedy|Drama|Romance|War" ...
##  - attr(*, ".internal.selfref")=<externalptr>
```

```
summary(edx)
```

```
##      userId         movieId          rating        timestamp
##  Min.   :    1   Min.   :    1   Min.   :0.500   Min.   :7.897e+08
##  1st Qu.:18127   1st Qu.:  648   1st Qu.:3.000   1st Qu.:9.468e+08
##  Median :35757   Median : 1834   Median :4.000   Median :1.035e+09
##  Mean   :35875   Mean   : 4123   Mean   :3.513   Mean   :1.033e+09
##  3rd Qu.:53617   3rd Qu.: 3626   3rd Qu.:4.000   3rd Qu.:1.127e+09
##  Max.   :71567   Max.   :65133   Max.   :5.000   Max.   :1.231e+09
##     title              genres
##  Length:8000071     Length:8000071
##  Class :character   Class :character
##  Mode  :character   Mode  :character
##
##
##
```

Total count per rating

The code will output the total number of each rating. From this we see rating of 4.0 was given out the most.

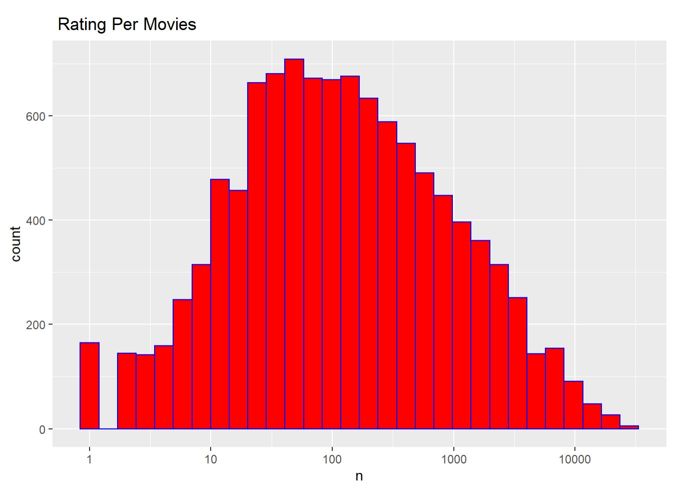
```
#counts the number of rating of each given rating
edx %>% group_by(rating) %>% summarize(count=n())%>% top_n(5)
```

```
## Selecting by count
```

```
## # A tibble: 5 x 2
##   rating   count
##    <dbl>   <int>
## 1    2    632317
## 2    3   1885510
## 3    3.5  703197
## 4    4   2301303
## 5    5   1235743
```

```
#number of rating per movies
edx %>% count(movieId) %>% ggplot(aes(n))+ geom_histogram(color= "blue",fill="red") + scale_x_log10() + ggtitle("
Rating Per Movies")
```

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

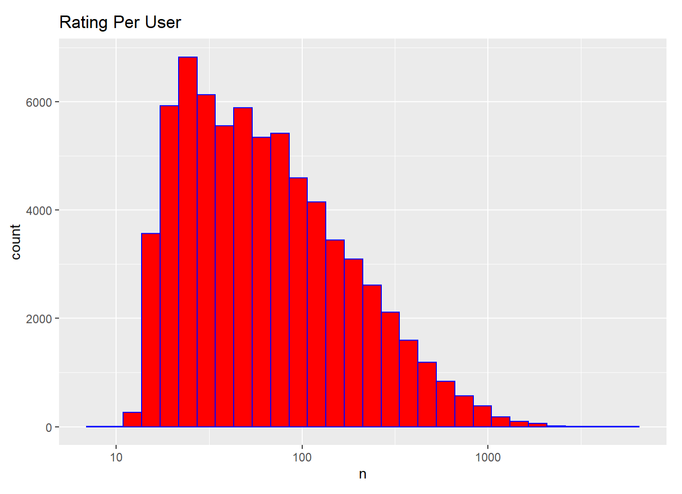


Rating Per User

The code will output a histogram of ratings per movie to see distriubution of given rating per movie.

```
edx %>% count(userId) %>%  ggplot(aes(n)) + geom_histogram(color= "blue",fill="red") + scale_x_log10() + ggtitle(
"Rating Per User")
```

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



This table will show the number of ratings given per genre from descending order

```
edx %>% separate_rows(genres, sep = "\\|") %>%
  group_by(genres) %>%
  summarize(count = n()) %>%
  arrange(desc(count))
```

```
## # A tibble: 20 x 2
##    genres                count
##    <chr>                 <int>
##  1 Drama               3475173
##  2 Comedy              3147018
##  3 Action              2276050
##  4 Thriller            2067880
##  5 Adventure           1697300
##  6 Romance             1521656
##  7 Sci-Fi              1192070
##  8 Crime               1180234
##  9 Fantasy              822592
## 10 Children             656041
## 11 Horror               614284
## 12 Mystery              505354
## 13 War                  454126
## 14 Animation            415322
## 15 Musical              384674
## 16 Western              168099
## 17 Film-Noir            105529
## 18 Documentary           82882
## 19 IMAX                   7287
## 20 (no genres listed)        7
```

Here we set the data set to 20% test set and 80% training set. The code was give by the edx course

```
set.seed(1)
test_index <- createDataPartition(y=edx$rating,times = 1,p=0.2,list=FALSE)
train <- edx[-test_index,]
test <- edx[test_index,]
```

Building the Model with RMSE

The winner for the Netflix challenge was decided by residual mean squared error (RMSE). The RMSE will measure the accuracy. There will be three model regarding the RMSE

```
RMSE <- function(true_ratings, predicted_ratings){
  sqrt(mean((true_ratings - predicted_ratings)^2, na.rm = TRUE))
}
```

First Model RMSE

First we get the average of the ratings in the train set to avoid any bias as all users are considered in this model. Once that has been outputted the mean is then put together with the rating from the test in RMSE syntax.The RMSE in the First model is 1.05

```
model_1 <- mean(train$rating)
model_1
```

```
## [1] 3.512505
```

```
RMSE1 <- RMSE(test$rating,model_1)
RMSE1
```

```
## [1] 1.059682
```

2nd RMSE model

For the 2nd model we took the mean of rating from train set. From there we grouped the train set by movieId and summarize the mean difference of rating and the mean of rating from train set.

```
model_2 <- mean(train$rating)
avgsmovie <- train %>% group_by(movieId) %>% summarize(avgrating_model2= mean(rating-model_2))
```

From there we took the value from rating_predicted and rating from the rest and put them in RMSE syntax, took get the RMSE for model 2. Which was 0.94.

```
rating_predicted <- model_2 + test %>% left_join(avgsmovie, by ='movieId') %>% pull(avgrating_model2)
RMSE2_Model_2 <- RMSE(rating_predicted,test$rating)
RMSE2_Model_2
```

```
## [1] 0.9435628
```

3rd Model RMSE

Here follows a simllar proccess to 2nd model expect a left_join function was added by movieId. When the predicted_ratings and the rating from the test set were put in the RMSE fucntion. The output 0.86

```
user_avgs <- train %>%
  left_join(avgsmovie, by='movieId') %>%
  group_by(userId) %>%
  summarize(b_u = mean(rating - RMSE2_Model_2 - avgrating_model2))
```

```
predicted_ratings <- test %>%
  left_join(avgsmovie, by='movieId') %>%
  left_join(user_avgs, by='userId') %>%
  mutate(pred =  RMSE2_Model_2 + avgrating_model2 + b_u) %>%
  pull(pred)
```

```
model_3_rmse <- RMSE(predicted_ratings, test$rating)
model_3_rmse
```

```
## [1] 0.8662753
```

Validation set for RMSE

```
valid_pred_rating <- validation %>%
  left_join(avgsmovie, by = "movieId" ) %>%
  left_join(user_avgs , by = "userId") %>%
  mutate(pred = RMSE2_Model_2 + avgrating_model2 + b_u ) %>%
  pull(pred)

model_3_valid <- RMSE(validation$rating, valid_pred_rating)
model_3_valid
```

```
## [1] 0.8674108
```

Table of all model’s RMSE and validation

```
RMSE_table<-data.frame(RMSE1,RMSE2_Model_2,model_3_rmse,model_3_valid)
RMSE_table
```

```
##      RMSE1 RMSE2_Model_2 model_3_rmse model_3_valid
## 1 1.059682     0.9435628    0.8662753     0.8674108
```