MovieLens

Brahmjot Kaur

2023-02-11

OVERVIEW

This project is a part of MovieLens Project of the Harvardx: PH125.9x Data Science: Capstone course. The MovieLens 10M dataset is a dataset that contains ratings and movie information provided by users on the MovieLens website. The dataset includes 10 million ratings and 100,000 tag applications applied to 10,677 movies by 69878 users. The ratings are on a scale from 0.5 to 5. The key steps that were performed to achieve this goal include:

1. Loading and cleaning of the data from the ratings and movies files in the dataset by reading in the data from the files, splitting the data, and setting appropriate column names. The data is then transformed so that the userId and movieId variables are integers and the rating variable is numeric.

2. Joining the ratings and movies data to create a single data frame, movielens by using a left_join function, which combines the two data frames on the movieId variable. A data frame is created that includes all the information from both the ratings and movies data frames.

3. Splitting the MovieLens data frame into two sets, edx and final_holdout_test, where final_holdout_test will be used to evaluate the performance of the recommendation algorithm and edx data set will be used for training the algorithm. The data is split using createDataPartition function with a seed of 1, and 10% of the data is selected as final_holdout_test set.

4. Creating an algorithm to predict the ratings for movies in the final_holdout_test set using the edx set and calculating the Root Mean Squared Error (RMSE) to evaluate the performance of the algorithm.

AIM

The goal of this project is to use MovieLens dataset to train an algorithm for rating prediction. The recommendation algorithm will use user's past movie ratings and predict what movies they would rate highly in the future. This can be useful for making personalized movie recommendations to users on a movie streaming platform or in a movie rental store. RMSE stands for Root Mean Squared Error. It measures how well a prediction model is performing by calculating the difference between the actual values and the predicted values. The RMSE is calculated by taking the square root of the mean of the squared differences between the actual and predicted values. The value of RMSE is inversely proportional to the performance of the model that is smaller the RMSE value, better the performance of the model, as it indicates that the model is making predictions that are very close to the actual values.

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

The algorithm with best results in terms of RMSE value will be used for prediction of movie ratings.

DATASET

The MovieLens 10M dataset is downloaded using the following links:

- [MovieLens 10M dataset] https://grouplens.org/datasets/movielens/10m/
- [MovieLens 10M dataset - zip file] http://files.grouplens.org/datasets/movielens/ml-10m.zip

```
############################################################
# Create edx and final_holdout_test sets
############################################################

if(!require(tidyverse)) install.packages("tidyverse", repos = "http://cran.us.r-project.org")
if(!require(caret)) install.packages("caret", repos = "http://cran.us.r-project.org")
if(!require(ggplot2)) install.packages("ggplot2", repos = "http://cran.us.r-project.org")
if(!require(reshape2)) install.packages("reshape2", repos = "http://cran.us.r-project.org")
if(!require(tinytex)) install.packages("reshape2", repos = "http://cran.us.r-project.org")

library(tidyverse)
library(ggplot2)
library(caret)
library(reshape2)
library(tinytex)

options(timeout = 120)

dl <- "ml-10M100K.zip"
if(!file.exists(dl))
  download.file("https://files.grouplens.org/datasets/movielens/ml-10m.zip", dl)

ratings_file <- "ml-10M100K/ratings.dat"
if(!file.exists(ratings_file))
  unzip(dl, ratings_file)

movies_file <- "ml-10M100K/movies.dat"
if(!file.exists(movies_file))
  unzip(dl, movies_file)

ratings <- as.data.frame(str_split(read_lines(ratings_file), fixed("::"), simplify = TRUE),
                         stringsAsFactors = FALSE)
colnames(ratings) <- c("userId", "movieId", "rating", "timestamp")
ratings <- ratings %>%
  mutate(userId = as.integer(userId),
         movieId = as.integer(movieId),
         rating = as.numeric(rating),
         timestamp = as.integer(timestamp))

movies <- as.data.frame(str_split(read_lines(movies_file), fixed("::"), simplify = TRUE),
                        stringsAsFactors = FALSE)
colnames(movies) <- c("movieId", "title", "genres")
movies <- movies %>%
  mutate(movieId = as.integer(movieId))

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

The dataset for the movies and ratings is unzipped and downloaded. These files are read into the following datasets using function 'str_split' with separator '::'. Column names for ratings and movie datasets are provided. These datasets are joined using the 'left_join' to create the movieLens data frame.

```
# Final hold-out test set will be 10% of MovieLens data
set.seed(1, sample.kind="Rounding") # if using R 3.6 or later
# set.seed(1) # if using R 3.5 or earlier
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 final hold-out test set are also in edx set
final_holdout_test <- temp %>%
  semi_join(edx, by = "movieId") %>%
  semi_join(edx, by = "userId")

# Add rows removed from final hold-out test set back into edx set
removed <- anti_join(temp, final_holdout_test)
edx <- rbind(edx, removed)

rm(dl, ratings, movies, test_index, temp, movielens, removed)
```

Dataset 'edx' is created from the 'movieLens' data by using a sample of 90% of the movieLens dataset. This is done by using function createDataPartition (). The other 10% of dataset 'MovieLens' is used to create the dataset 'temp'. 'final_holdout_test' dataset is created by altering the 'temp' to only include the users and movies that are also present in the 'edx' dataset using 'semi_join ()' on "movieId" and "UserId".

METHODS

DATA ANALYSIS

The dataset "edx" has six variables 'userId', 'movieId', 'rating', 'timestamp', 'title', and 'genres'.

```
##   userId movieId rating timestamp                         title
## 1      1     122      5 838985046              Boomerang (1992)
## 2      1     185      5 838983525               Net, The (1995)
## 4      1     292      5 838983421               Outbreak (1995)
## 5      1     316      5 838983392               Stargate (1994)
## 6      1     329      5 838983392 Star Trek: Generations (1994)
## 7      1     355      5 838984474       Flintstones, The (1994)
##                         genres
## 1                Comedy|Romance
## 2          Action|Crime|Thriller
## 4  Action|Drama|Sci-Fi|Thriller
## 5       Action|Adventure|Sci-Fi
## 6 Action|Adventure|Drama|Sci-Fi
## 7       Children|Comedy|Fantasy
```

```
##      userId         movieId          rating        timestamp
##  Min.   :    1   Min.   :    1   Min.   :0.500   Min.   :7.897e+08
##  1st Qu.:18124   1st Qu.:  648   1st Qu.:3.000   1st Qu.:9.468e+08
##  Median :35738   Median : 1834   Median :4.000   Median :1.035e+09
```

```
##  Mean   :35870   Mean   : 4122   Mean   :3.512   Mean   :1.033e+09
##  3rd Qu.:53607   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:9000055     Length:9000055
##  Class :character   Class :character
##  Mode  :character   Mode  :character
##
##
##
```

Initiating the data analysis for 'edx' by calculating the mean, median and standard deviation of the ratings in 'edx'.

```
mean(edx$rating)
```

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

```
median(edx$rating)
```

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

```
sd(edx$rating)
```

```
## [1] 1.060331
```

The 'edx' dataset has 9000055 entries which consists ratings of 10677 movies given by 69878 users where one user can rate multiple movies and one movie can be rated by multiple users.

```
nrow(edx)
```

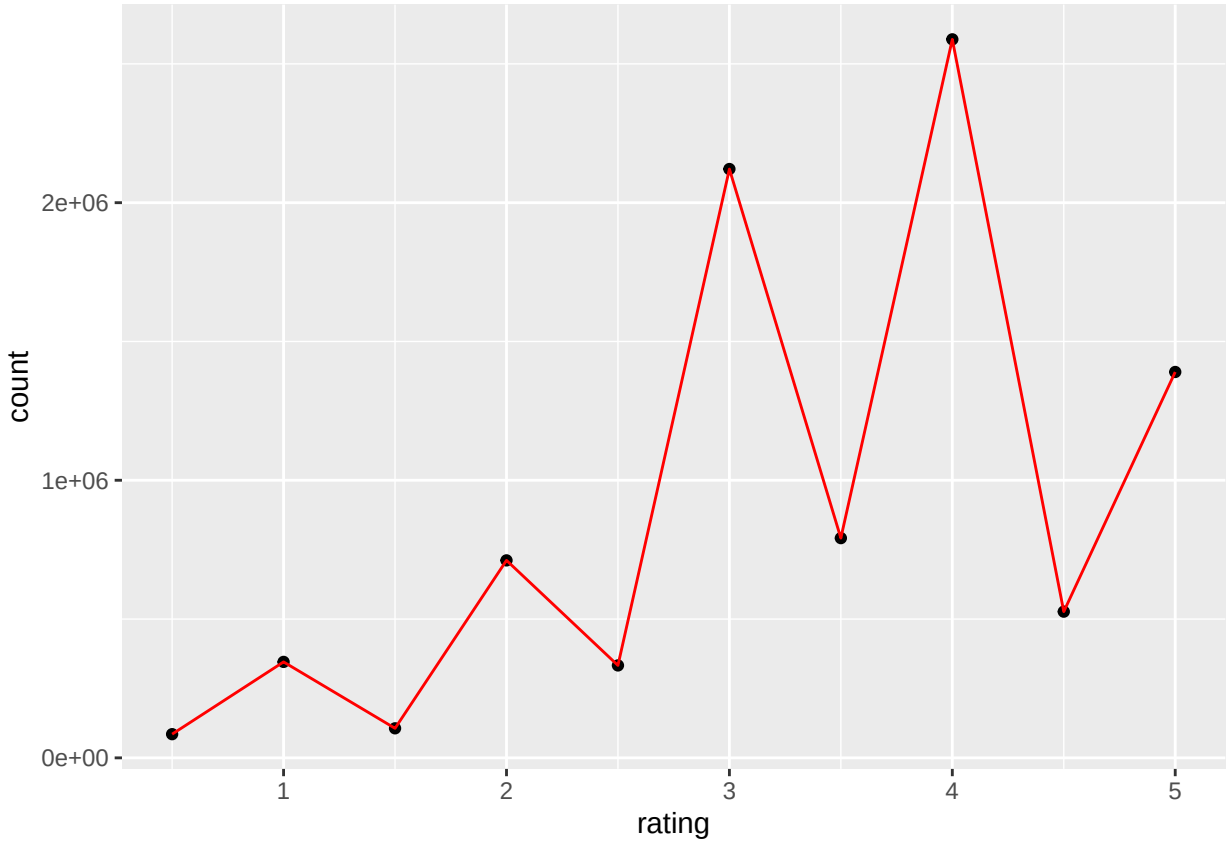
```
## [1] 9000055
```

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

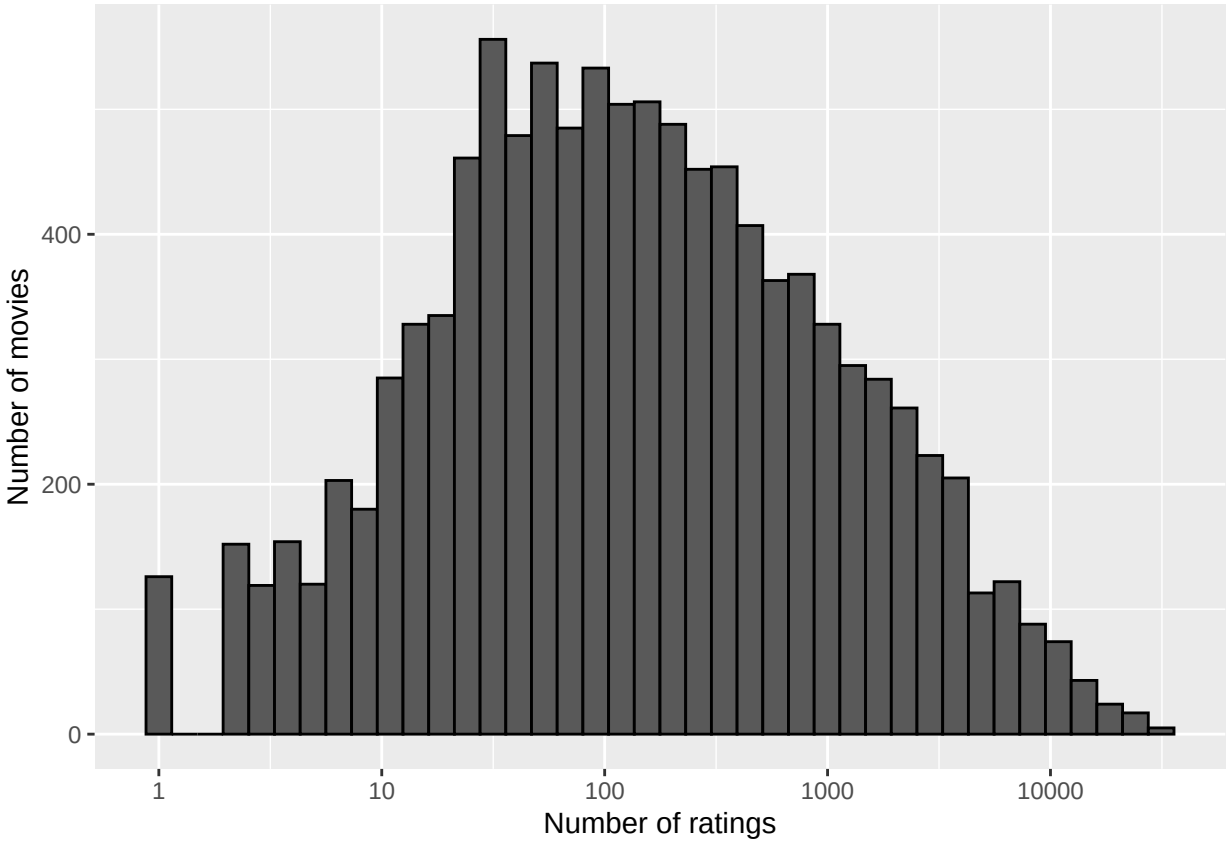
```
## [1] 69878
```

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

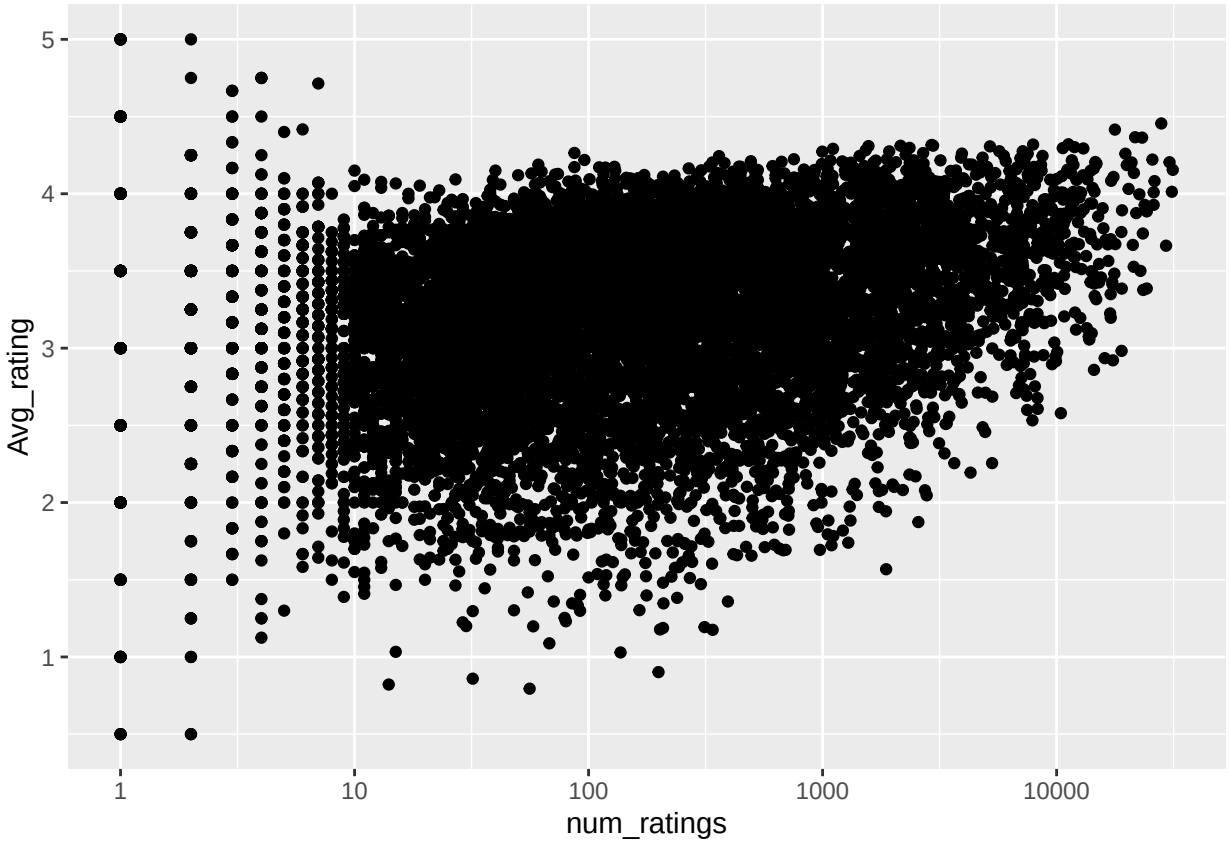
```
## [1] 10677
```



As this data depicts , users prefer rating higher than lower. We notice that 4 is the most common rating. It is followed by 3 and 5. It can be noticed as well that whole star ratings(eg 3,4,5) are much common than the half ratings(eg 0.5,1.5).



The above bar graph shows number of ratings vs the number of movies. There are some movies rated much often than others. There are around 126 movies which are rated only once and only 3 movies with over 30000 ratings. This is important for our model as very less rating numbers may result in untrustworthy estimate for predictions. A movie on average receives around 935 ratings.
This graph clearly shows the inclined distribution of movies for number of ratings over 100.



Is Average ratings affected by the number of ratings? This graph clearly shows that the average rating is directly proportional to the number of ratings.

Analysis on the basis of genre

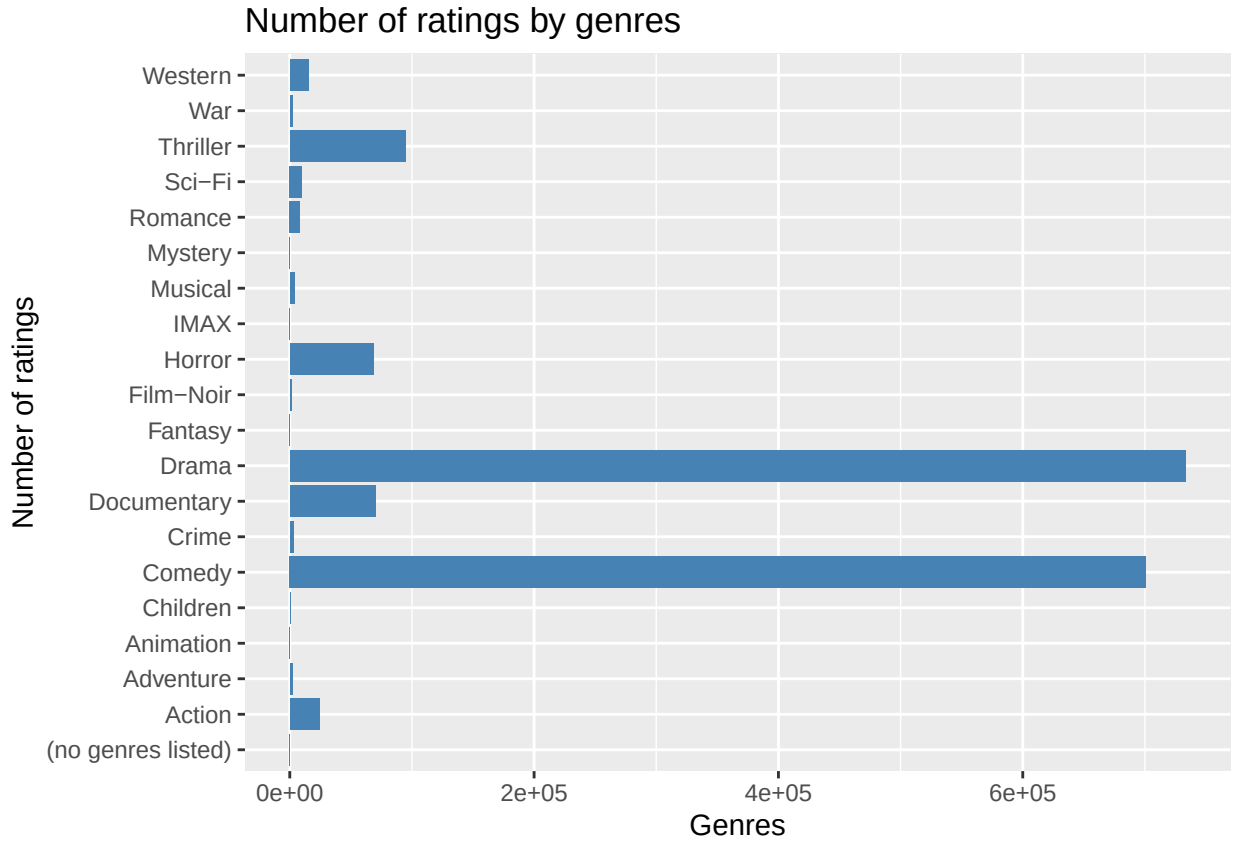
There are around 797 genres in the edx dataset.

The 10 genres rated the lowest are listed below

```
## # A tibble: 797 x 3
##    genres                                        Avg_rating num_ratings
##    <chr>                                              <dbl>       <int>
##  1 Documentary|Horror                                  1.45         619
##  2 Action|Animation|Comedy|Horror                      1.5            2
##  3 Action|Horror|Mystery|Thriller                      1.61         327
##  4 Comedy|Film-Noir|Thriller                           1.64          21
##  5 Action|Drama|Horror|Sci-Fi                          1.75           4
##  6 Adventure|Drama|Horror|Sci-Fi|Thriller              1.75         217
##  7 Action|Adventure|Drama|Fantasy|Sci-Fi               1.90          57
##  8 Action|Children|Comedy                              1.91         518
##  9 Action|Adventure|Children                           1.92         824
## 10 Adventure|Animation|Children|Fantasy|Sci-Fi         1.92         691
## # ... with 787 more rows
```

The 10 genres with highest ratings are listed as below

```
## # A tibble: 797 x 3
##    genres                                  Avg_rating num_ratings
##    <chr>                                        <dbl>       <int>
##  1 Animation|IMAX|Sci-Fi                         4.71           7
##  2 Drama|Film-Noir|Romance                       4.30        2989
##  3 Action|Crime|Drama|IMAX                       4.30        2353
##  4 Animation|Children|Comedy|Crime               4.28        7167
##  5 Film-Noir|Mystery                             4.24        5988
##  6 Crime|Film-Noir|Mystery                       4.22        4029
##  7 Film-Noir|Romance|Thriller                    4.22        2453
##  8 Crime|Film-Noir|Thriller                      4.21        4844
##  9 Crime|Mystery|Thriller                        4.20       26892
## 10 Action|Adventure|Comedy|Fantasy|Romance       4.20       14809
## # ... with 787 more rows
```



10 highest rated genres

```
## # A tibble: 20 x 3
##    genres             Avg_rating num_ratings
##    <chr>                   <dbl>       <int>
##  1 Film-Noir                3.83        1575
##  2 Documentary              3.82       70041
##  3 Drama                    3.71      733296
##  4 War                      3.67        2300
##  5 (no genres listed)       3.64           7
##  6 Western                  3.54       15300
```

```
##  7 Thriller                3.53       94662
##  8 Fantasy                 3.51          86
##  9 Musical                 3.45        3851
## 10 Romance                 3.27        8410
## # ... with 10 more rows
```

MODELLING

Model 1- Average movie rating

The first basic model predicts rating for all movies as the mean of the ratings in the dataset MovieLens. While we all know this is not the optimal approach, we get the baseline RMSE value to which we can compare the future models.

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

'*mu*' represents the mean of the ratings in the edx dataset.

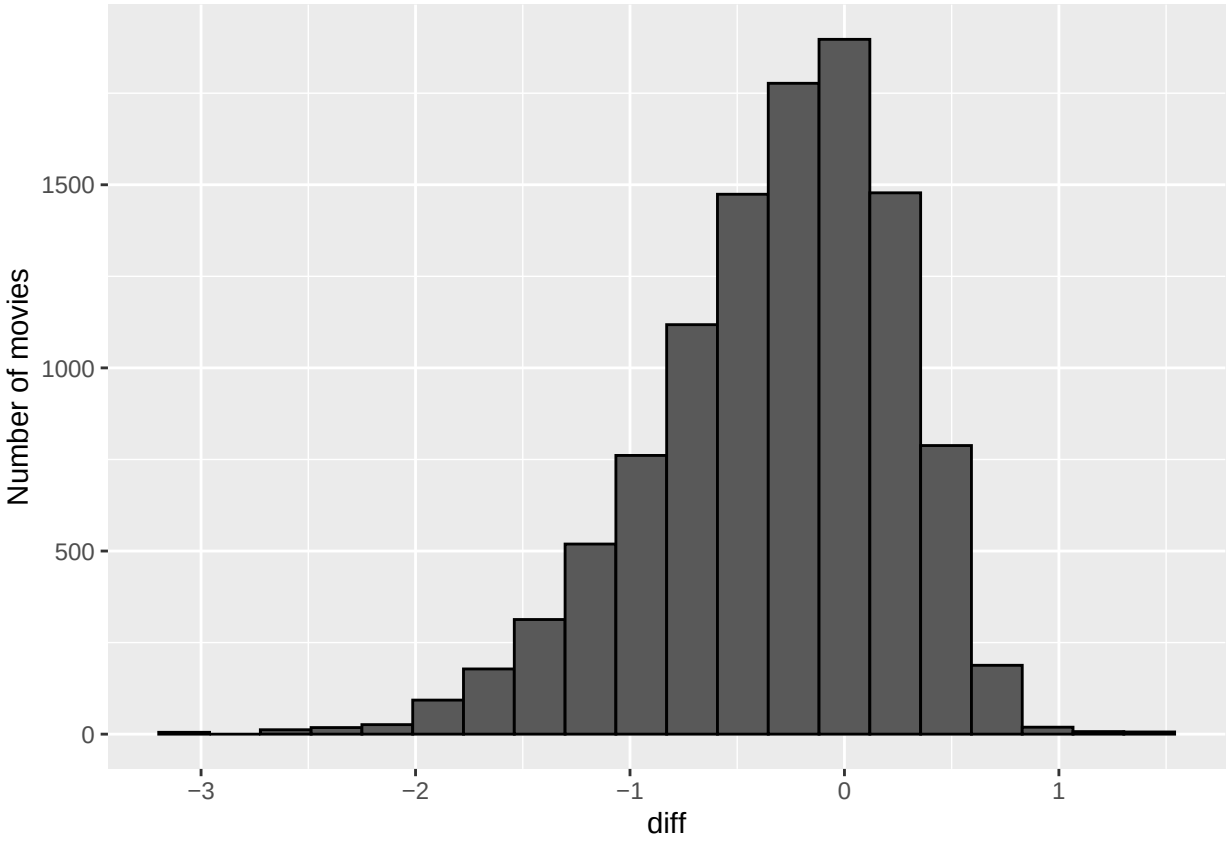
```
## Warning: `data_frame()` was deprecated in tibble 1.1.0.
## i Please use `tibble()` instead.
```

Method	RMSE
Mean Movie rating	1.061202

In order to find other better approach , collaborative filtering can be used in order to evaluate the model.

Model 2- Movie Effect Model

Some movies are rated higher than the others. Hence, providing every movie with the same rating as in Model 1 would be unfair. This model is built by taking account the the average ratings of each individual movie. '*diff*' in the plot below is the deviation of the mean rating of each movie from the total mean calculated earlier as '*mu*'.

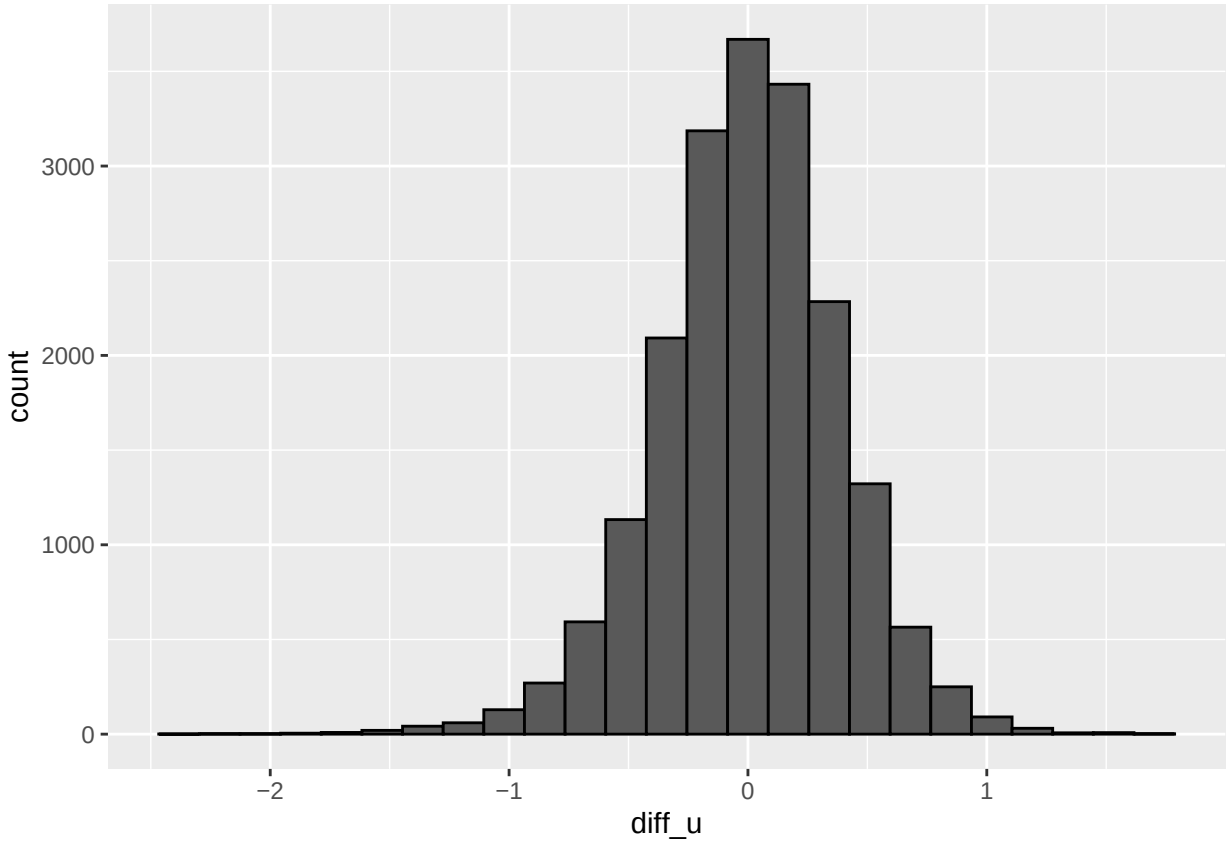


We predict the movie ratings on the basis of the fact that if a movie has an average rating lesser than the average of all movies. The predicted rating would be '*diff*' less than '*mu*' and vice versa.

Method	RMSE
Mean Movie rating	1.0612018
Movie effect	0.9439087

Model 3- Movie and User Effect

Knowing that some users are more critical than the others. Hence, it is recommmend to include the effect of an individual user in the prediction of ratings. User Effect is taken into account by calculating the average rating for the users rating more than 128 movies. 128 is used as this the approximate average number of ratings provided by a user.



The table below represents the mean difference in the mean ratings provided by each user to the total rating. '*diff_u*' is calculated by subtracting the total average rating '*mu*' to the average rating provided by each user.'*diff*' is further subtracted to add the movie effect in the predicted data.

```
## # A tibble: 69,878 x 2
##   userId  diff_u
##    <int>   <dbl>
## 1      1  1.68
## 2      2 -0.236
## 3      3  0.264
## 4      4  0.652
## 5      5  0.0853
## # ... with 69,873 more rows
```

mean(training set+movie bias+user bias)= mu+diff+diff_u

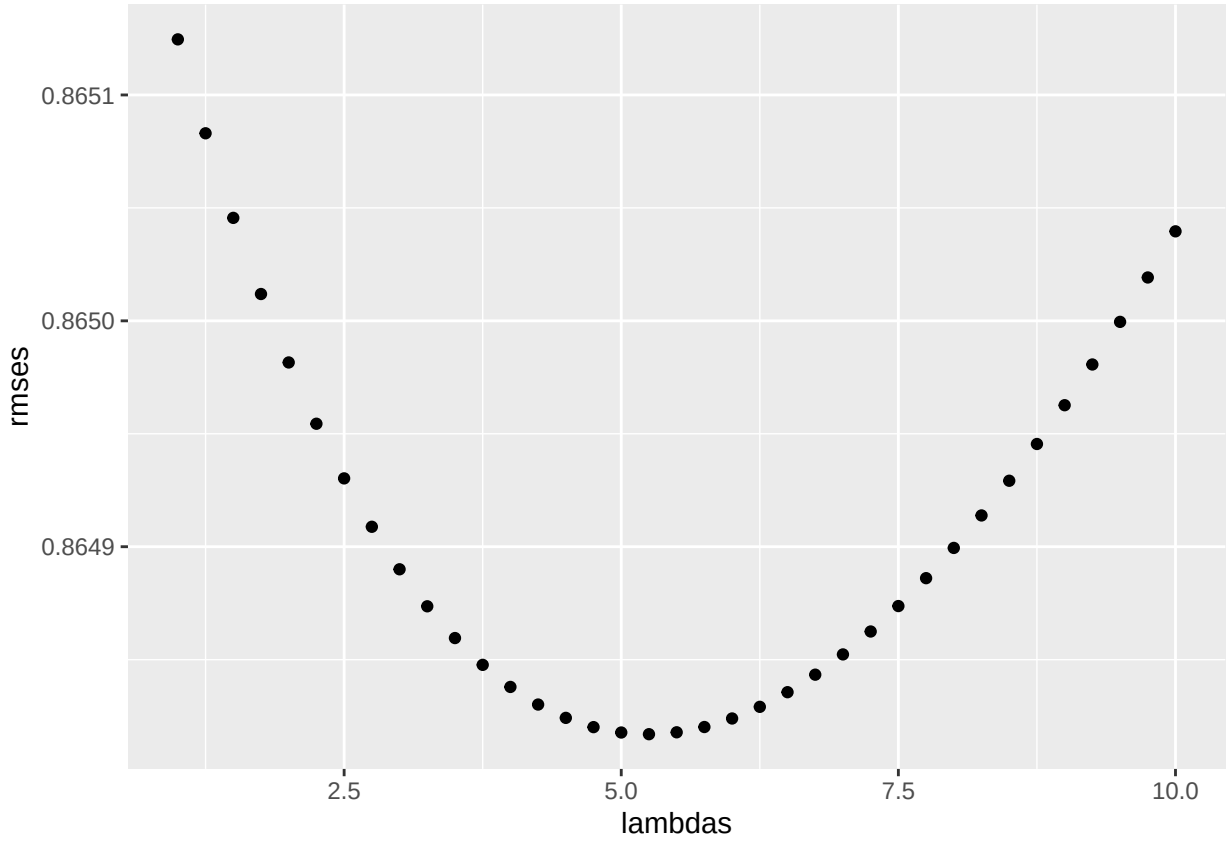
Method	RMSE
Mean Movie rating	1.0612018
Movie effect	0.9439087
Movie and User effect	0.8653488

Method 4- Movie and User Effect Regularized

To improve the previous model, regularization is implemented. Some of the movies were rated less and some users rated only a few movies. This influences the prediction. Hence, we find the value of lambda that will

minimize the RMSE. This will shrink bias values '*diff*' and ' *diff_u*' in case of fewer ratings.

```
## Warning: `qplot()` was deprecated in ggplot2 3.4.0.
```

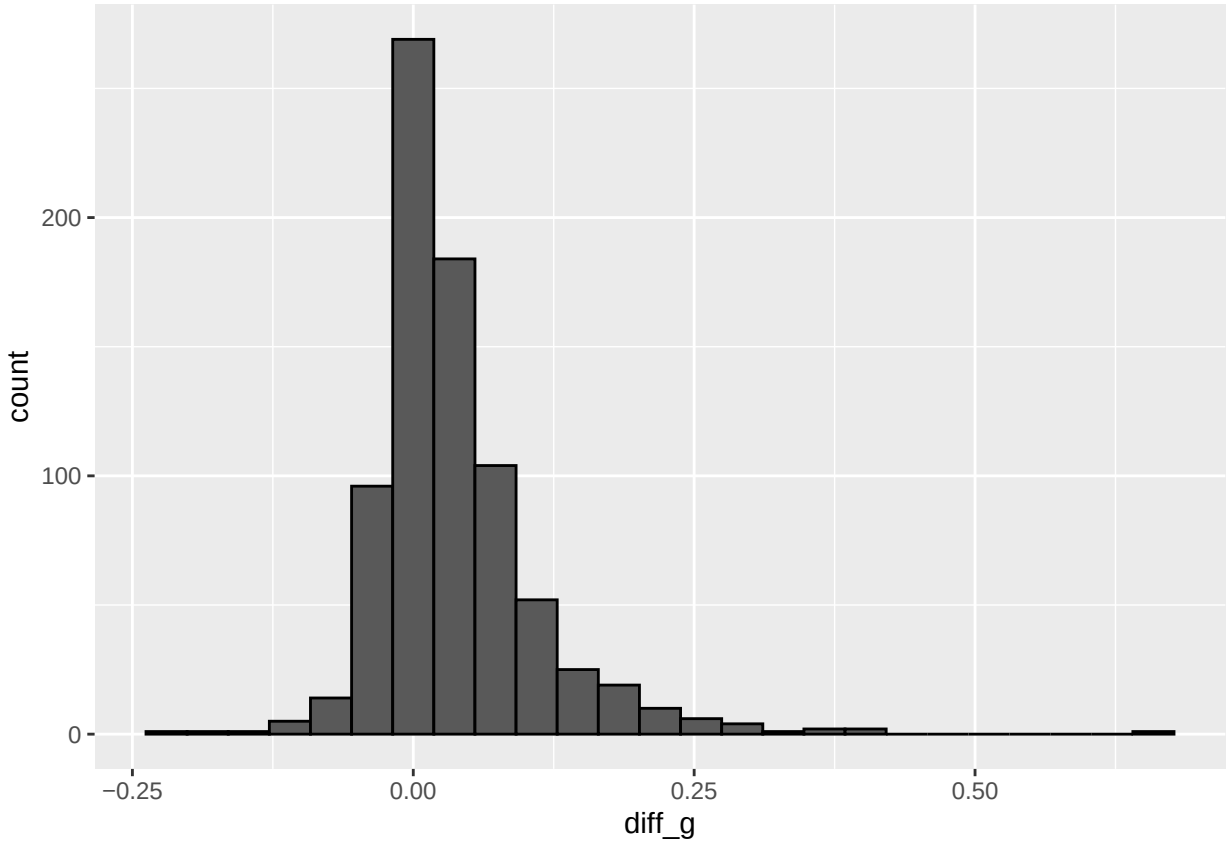


The above plot indicates the RMSE values for each lambda. Hence, simply by finding the lowest RMSE value and the corresponding lambda(tuning factor)

Method	RMSE
Mean Movie rating	1.0612018
Movie effect	0.9439087
Movie and User effect	0.8653488
Movie and User effect(optimal lambda)	0.8648170

Method 5- Movie, User and Genre Effects

The genre variable may help in prediction of rating as some genres have higher ratings than others. There are around 797 genres. Calculating the bias for genre can affect the prediction.



The table below shows the genres along with difference in the ratings and the mean of particular movie , user and genre i.e. '*diff_g*'.

```
## # A tibble: 797 x 2
##   genres                                              diff_g
##   <chr>                                                <dbl>
## 1 (no genres listed)                                  0.232
## 2 Action                                             -0.0345
## 3 Action|Adventure                                   -0.0131
## 4 Action|Adventure|Animation|Children|Comedy          0.0109
## 5 Action|Adventure|Animation|Children|Comedy|Fantasy -0.0199
## # ... with 792 more rows
```

Method	RMSE
Mean Movie rating	1.0612018
Movie effect	0.9439087
Movie and User effect	0.8653488
Movie and User effect(optimal lambda)	0.8648170
Movie,User and genre effect	0.8649469

RESULTS

Below is the table consisting all the RMSE values with their respective methods:

Method	RMSE
Mean Movie rating	1.0612018
Movie effect	0.9439087
Movie and User effect	0.8653488
Movie and User effect(optimal lambda)	0.8648170
Movie,User and genre effect	0.8649469

The capstone project required the value of RMSE less than 0.86490.

CONCLUSIONS

This project is based on the implementation of various machine learning and data visualization tools for analyzing the dataset. The visualizations are created on the basis of various attributes in the dataset which could affect the prediction of the ratings. Visualization of the dataset was the most interesting past of this project as it could help identifying various patterns between the various attributes which weren't evident. Libraries such as *recommmenderlab* and *Matrix* are installed for matrix factorization and collaborative filtering. Considering collaborative filtering of the data of the basis of *MovieId, UserId, and genre* have affected the RMSE value. Other machine models can also be used in order to predict the ratings in a more optimal way.