HarvardX:PH125.9x Data Science - Capstone: MovieLens Recommendation System

Richard Jonyo

20 February 2022

Contents

1. Executive Summary 3
2. Methods and Exploratory Analysis 3
    2.1 Required Libraries 3
    2.2 The MovieLens Dataset 4
    2.3 Splitting the Edx Dataset 5
    2.4 Exploratory Analysis 6
    2.4.1 Movie ratings 7
    2.4.2 Users rating 9
    2.4.3 Year of release 11
    2.4.4 Genre 12
3. Results 13
    3.1 The Prediction Approach 13
    3.2 First Model (Naive Model) 14
    3.3 Second Model (Movie Bias) 14
    3.4 Third Model (Movie & User Biases) 15
    3.5 Fourth Model (Regularized Movie & User Biases) 17
4. Conclusion 19

1. Executive Summary

The goal of this project was to come up with a movie recommendation system using the MovieLens datasets which consists of 10 million movie ratings. A movie recommendation system is an information filtering system that attempts to predict the rating or preference a user would give to a movie. Recommendation systems are an improvement over the traditional classification models as they can take many classes of input and provide similarity ranking based on algorithms hence providing the user with more accurate results.

This project is part of the HarvardX:PH125.9x Data Science: Capstone course and we use a smaller subset of the MovieLens dataset which is 10M that contains 10,677 movies by 69,878 users. The dataset has genres, and the movies are rated from 0.5 to 5 with increments of 0.5. A movie can be categorized under a number of genres.

The dataset is split into two sets: a training set (edx) and a final hold-out test set (validation) using code provided by the course. The objective was for the final algorithm to predict ratings with a root mean square error (RMSE) of less than 0.86490 versus the actual ratings included in the validation set.

To develop the recommendation model, a 4-step approach was employed where the initial model assumed the recommendations agreed with the naive mean of all movie ratings. The second model added the movie bias to the initial model since some movies seem to be more popular than others. The third model added users bias which improved the RMSE slightly. The final model included the regularized movie and user biases to improve the RMSE.

Exploratory analysis was conducted on the data using R and R Studio, a language and a software environment for statistical computing. R Markdown, a simple formatting syntax for authoring HTML, PDF, and MS Word documents and a component of R Studio was used to compile the report.

2. Methods and Exploratory Analysis

This section helps us to understand the structure of the Movilens dataset for us to gain insights that will aid in a better prediction of movie ratings. It explains the process and techniques used, including data cleaning, exploration, visualization, insights gained, and the modeling approach used.

2.1 Required Libraries

The project utilized and loaded several CRAN libraries to assist with the analysis. The libraries were automatically downloaded and installed during code execution. These included: tidyverse, caret, data.table, lubridate, and dplyr libraries.

```
# Note: this process could take a couple of minutes
if(!require(tidyverse)) install.packages("tidyverse", repos = "http://cran.us
.r-project.org")
if(!require(caret)) install.packages("caret", repos = "http://cran.us.r-proje
```

```
ct.org")
if(!require(data.table)) install.packages("data.table", repos = "http://cran.
us.r-project.org")

library(tidyverse)
library(caret)
library(data.table)
library(lubridate)
library(dplyr)
```

2.2 The MovieLens Dataset

The Movielens Dataset (25M) contains 25 million ratings and one million tag applications applied to 62,000 movies by 162,000 users. The dataset has no demographic information. Each user is represented by an id and no other information about the users is provided. For this project we use the Movielens Dataset (10M) which is a subset of the full dataset. Edx is the name provided to the subset of the MovieLens dataset that consists of 9,000,055 observations and 6 columns. Each observation is a rating provided by a user for a movie. The edx dataset contains ratings provided by a total of 69,878 unique users for a total of 10,677 unique movies. Below is the code that was provided by the course so that this project can build on it.

```
# MovieLens 10M dataset:
# https://grouplens.org/datasets/movielens/10m/
# http://files.grouplens.org/datasets/movielens/ml-10m.zip

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

ratings <- fread(text = gsub("::", "\t", readLines(unzip(dl, "ml-10M100K/rati
ngs.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")

# We are using R 4.0.4:
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")

# The validation set will be 10% of MovieLens data
set.seed(1, sample.kind="Rounding") # sing R version 4.0.4`
test_index <- createDataPartition(y = movielens$rating, times = 1, p = 0.1, l
ist = FALSE)
```

```
edx <- movielens[-test_index,]
temp <- movielens[test_index,]

# Make sure userId and movieId in validation set are also in edx set
test_dataset  <- 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, test_dataset)
edx <- rbind(edx, removed)
#dim(test_dataset)#test set has 1,000,003 records
#rm(test_index, temp, removed)
rm(dl, ratings, movies, test_index, temp, movielens, removed)
```

2.3 Splitting the Edx Dataset

The edx dataset is divided into two sets: training and test datasets. The test dataset is used to evaluate model when building the model. The training set has 8,100,048 observations and 6 columns. The validation set which represents 10% of the 10M Movielens dataset has the same features, but with a total of 899,993 occurrences and it is used to evaluate the final model. The model with the lowest RMSE will be used to make the final predictions on the validation set.

```
#We divide into two sets: training and test sets
#The test set will be split into two
test_index  <- createDataPartition(y = edx$rating, times = 1, p = 0.1, list =
FALSE)
training_dataset <- edx[-test_index,]
temp <- edx[test_index,]
dim(training_dataset) #training set has 8,100,048 records

## [1] 8100048       6
```

We ensure that the userId and movieId are in both training and test sets.

```
test_dataset <- temp %>%
  semi_join(training_dataset, by = "movieId") %>%
  semi_join(training_dataset, by = "userId")

# We add the rows removed from or test dataset into training set
removed_rows <- anti_join(temp, test_dataset)
training_dataset <- rbind(training_dataset, removed_rows)
dim(test_dataset)#test set has 1,000,003 records

## [1] 899988      6

rm(test_index, temp, removed_rows)
```

2.4 Exploratory Analysis

We need to understand the edx dataset before starting to develop the models hence an exploratory analysis is significant.

Below is a quick summary (top five records), and the data types of the dataset:

```
head(edx, 5) #Preview edx
```

```
##    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)
##                           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
```

```
knitr::kable(summary(edx))
```

userId	movieId	rating	timestamp	title	genres
Min. : 1	Min. : 1	Min. :0.500	Min. :7.897e+08	Length:9000055	Length:9000055
1st Qu.:18124	1st Qu.: 648	1st Qu.:3.000	1st Qu.:9.468e+08	Class :character	Class :character
Median :35738	Median : 1834	Median :4.000	Median :1.035e+09	Mode :character	Mode :character
Mean :35870	Mean : 4122	Mean :3.512	Mean :1.033e+09	NA	NA
3rd Qu.:53607	3rd Qu.: 3626	3rd Qu.:4.000	3rd Qu.:1.127e+09	NA	NA
Max. :71567	Max. :65133	Max. :5.000	Max. :1.231e+09	NA	NA

The columns in the edx dataset include userId (Unique ID for the user), movieId (Unique ID for the movie), rating (A rating between 0 and 5 for the movie), timestamp (Date and time the rating was given), title (Movie title), and genres.

```
## Rows: 9,000,055
## Columns: 6
## $ userId    <int> 1, 1, 1, 1, 1, 1, 1, 1, 1, ...
## $ movieId   <dbl> 122, 185, 292, 316, 329, 35...
## $ rating    <dbl> 5, 5, 5, 5, 5, 5, 5, 5, 5, ...
## $ timestamp <int> 838985046, 838983525, 83898...
```

```
## $ title     <chr> "Boomerang (1992)", "Net, T...
## $ genres    <chr> "Comedy|Romance", "Action|C...
```

There are no missing values.

```
anyNA(edx) #check missing values - results to FALSE

## [1] FALSE
```

We have 69,878 users and 10,677 movies in the edx set.

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

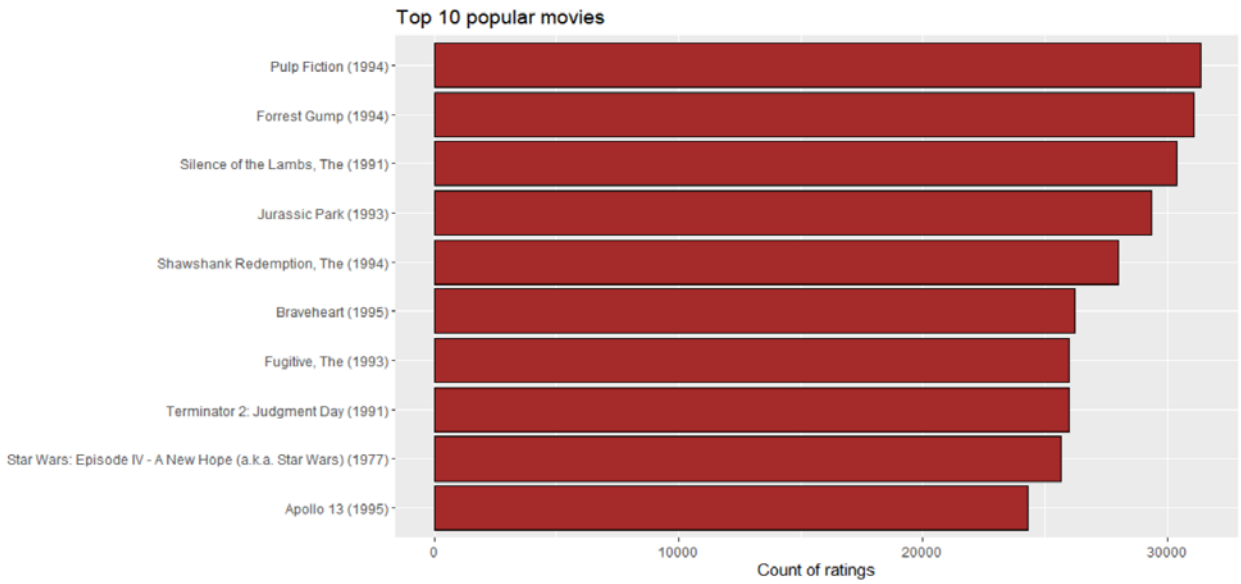
2.4.1 Movie ratings

Pulp Fiction (1994), Forrest Gump (1994) and Silence of the Lambs (1991) are the highest rated in that order with over 30,000 ratings. The average rating in the edx dataset was 3.51. The minimum rating for the movies was 0.5 and the maximum rating was 5.

```
##  [1] 5.0 3.0 2.0 4.0 4.5 3.5 1.0 1.5 2.5 0.5
```

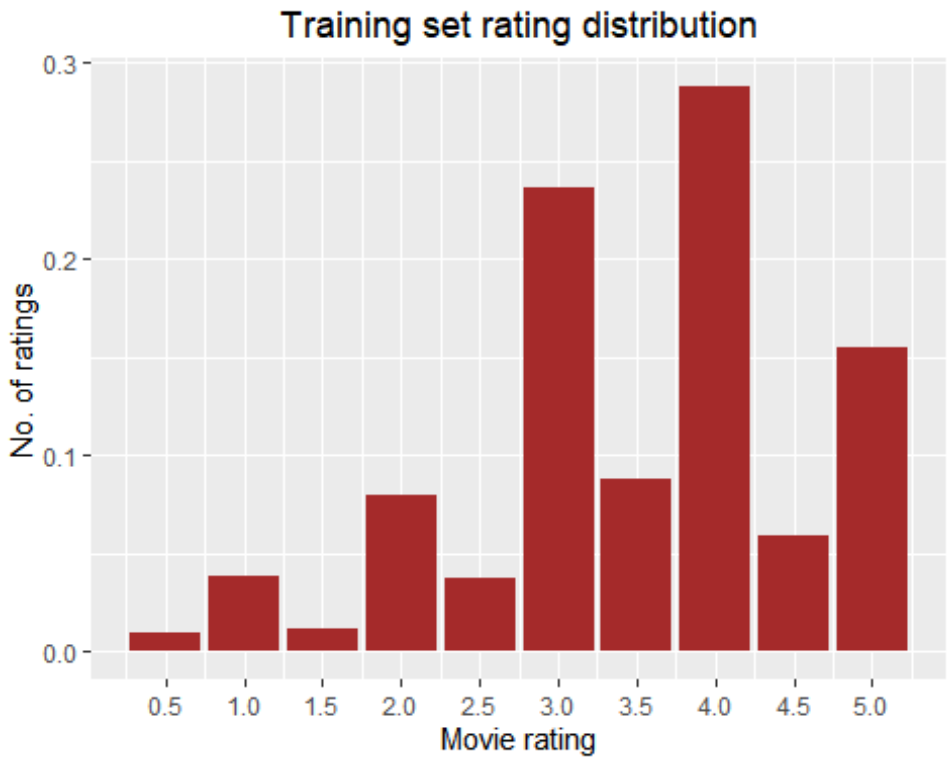
The figure below shows the top 10 most popular movies.

```
#Plotting top 10 most popular movies
edx %>%
  group_by(title) %>%
  summarize(count = n()) %>%
  arrange(-count) %>%
  top_n(10, count) %>%
  ggplot(aes(count, reorder(title, count))) +
  geom_bar(color = "black", fill = "brown", stat = "identity") +
  ggtitle("Top 10 popular movies")+
  xlab("Count of ratings") +
  ylab(NULL)
```



The figure below shows the distribution of movie ratings. The average rating in the edx dataset was 3.512465. The minimum rating for a movie was 0.5 and the maximum rating was 5.0. Some movies ratings fall below the average rating, and some are above. We can see that there is a bias for some movies which makes their ratings deviate from the mean rating.

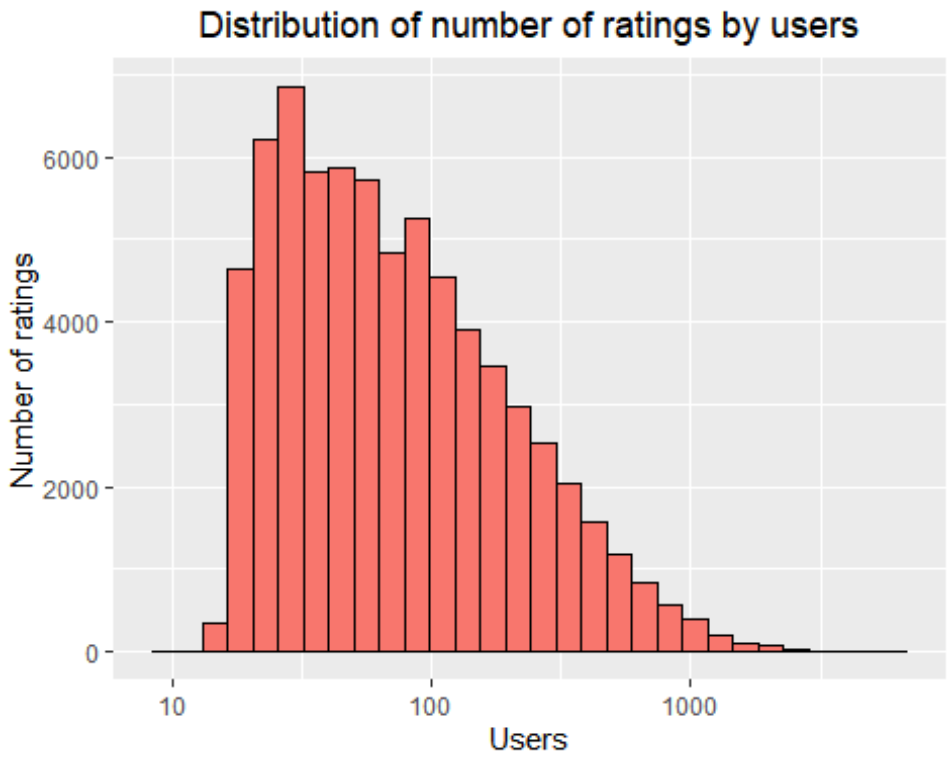
```
mean = mean(edx$rating)#mean

# We visualize the training set rating distribution
edx %>%
ggplot(aes(rating, y = ..prop..)) +
  geom_bar(fill = "brown") +
  ggtitle("Training set rating distribution")+
  theme(plot.title = element_text(hjust = 0.5))+
  scale_x_continuous(breaks = c(0.5, 1, 1.5, 2, 2.5, 3, 3.5, 4, 4.5, 5))+
  labs(x = "Movie rating", y = "No. of ratings")
```



2.4.2 Users rating

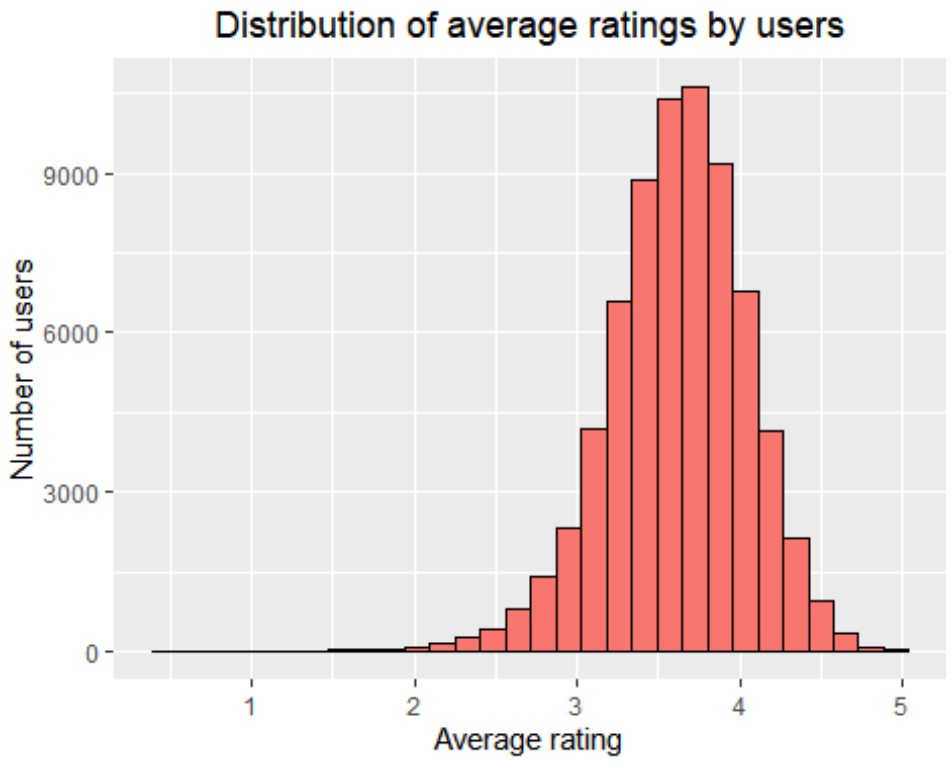
The plot below shows the distribution of the number of ratings by users. We notice that users do not have a uniform average rating. Users may have a bias towards particular movies, and some may have a tendency to rate a movie high or less.

```
#We plot the no. of ratings by users
edx %>%
  count(userId) %>%
  ggplot(aes(n, fill = "brown")) +
  geom_histogram( bins=30, color="black", show.legend = FALSE) +
  scale_x_log10() +
  labs(x = "Users", y = "Number of ratings")+
  ggtitle("Distribution of number of ratings by users")+
  theme(plot.title = element_text(hjust = 0.5))
```



We plot mean rating by users and it is evident that the distribution of the average ratings is right-skewed, meaning that many users tend to provide good ratings.

```
#We plot mean rating by users
edx %>% group_by(userId) %>%
  summarise(mean_rating = sum(rating)/n()) %>%
  ggplot(aes(mean_rating, fill = "brown")) +
  geom_histogram( bins=30, color="black", show.legend = FALSE) +
  labs(x = "Average rating", y = "Number of users")+
  ggtitle("Distribution of average ratings by users")+
  theme(plot.title = element_text(hjust = 0.5))
```

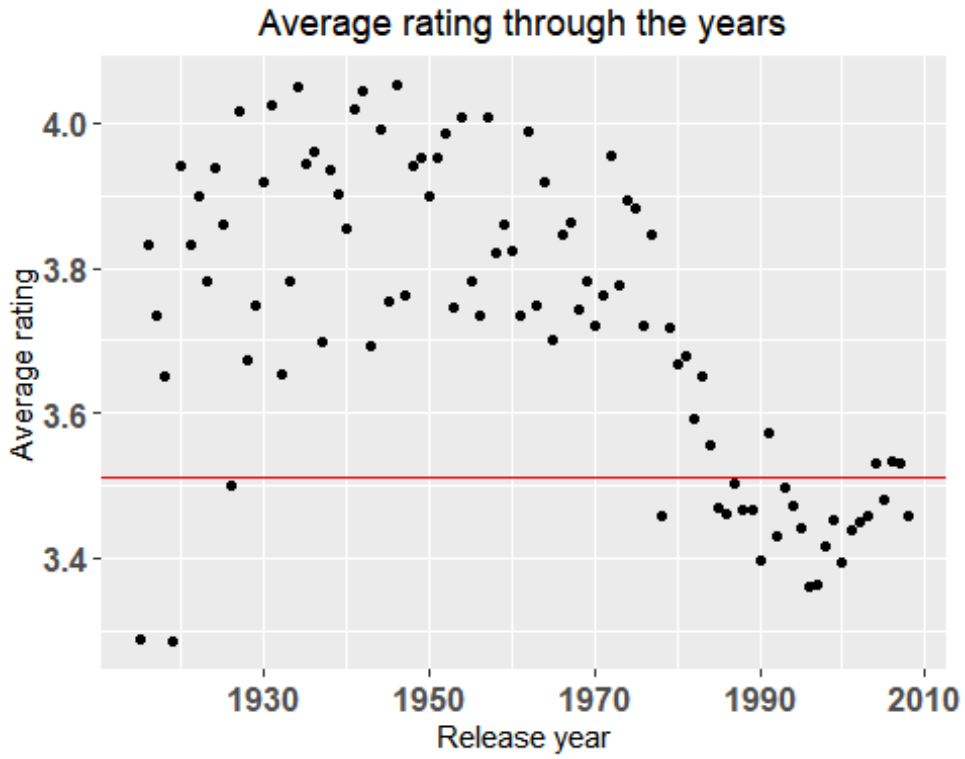


2.4.3 Year of release

We need to extract the year of release in the edx dataset into a separate column that will be of a more usable format.

The figure below shows the movie ratings over the years. We can see that prior to 1990s movies were highly rated than between the years 1930 and 1970. After 1970, the movie ratings decreased gradually.

```
#Plot for average rating through the years
edx %>% group_by(release_year) %>%
  summarise(n =n(), avg = mean(rating)) %>%
  ggplot(aes(release_year, avg))+
  geom_point()+geom_hline(yintercept = mean, color = "red")+labs(x="Release y
ear",y="Average rating")+theme(axis.text = element_text(size=12,face = "bold"
))+
  ggtitle('Average rating through the years')+
  theme(plot.title = element_text(hjust = 0.5))
```



2.4.4 Genre

The top 3 movie genres which were highly reviewed were Drama (733,296), Comedy (700,889), and Comedy|Romance (365,468)

```
#Top 3 movie genres which were highly reviewed
top_genres <- edx %>% group_by(genres) %>%
  summarize(count = n()) %>% arrange(desc(count)) %>% head(3)
top_genres %>% knitr::kable()
```

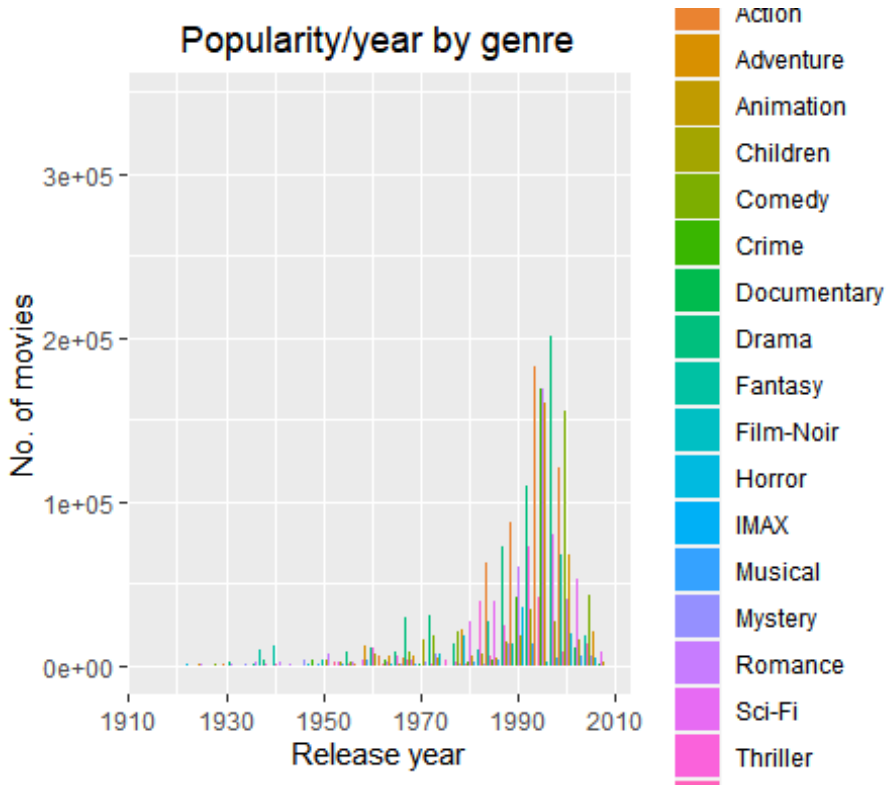
genres	count
Drama	733296
Comedy	700889
Comedy|Romance	365468

The figure below shows a summary of the genres by year.

```
#We summarize the popular genres by year
genres_by_year <- edx %>%
  separate_rows(genres, sep = "\\|") %>%
  select(movieId, release_year, genres) %>%
  group_by(release_year, genres) %>%
  summarise(count = n()) %>% arrange(desc(release_year))
```

Different periods show certain genres being more popular during those periods. It is for this reason that we will not include genre into the prediction model. Majority of the ratings were done between the years 1990 and 2010.

```
ggplot(genres_by_year, aes(x = release_year, y = count)) +
  geom_col(aes(fill = genres), position = 'dodge') +
  ylab('No. of movies') +
  xlab('Release year') +
  ggtitle('Popularity/year by genre')+
  theme(plot.title = element_text(hjust = 0.5))
```



3. Results

This section presents the prediction approach that was employed, modeling results and discusses the model performance.

3.1 The Prediction Approach

The accuracy was evaluated using the RMSE which measures the difference between predicted and observed values. The goal is to reduce the error below 0.8649. The RMSE function was defined as follows:

```
#function to compute RMSE
rmse <- function(true_ratings, predicted_ratings){
  sqrt(mean((true_ratings - predicted_ratings)^2))
}
```

3.2 First Model (Naive Model)

For the first model we make a prediction using the mean of the movie ratings. Also known as the naive model which assumes the movies will have the same rating regardless of genre or user.

```
mean_movie_rating <- mean(training_dataset$rating) #mean of the ratings
mean_movie_rating

## [1] 3.512509
```

The mean obtained was 3.512509.

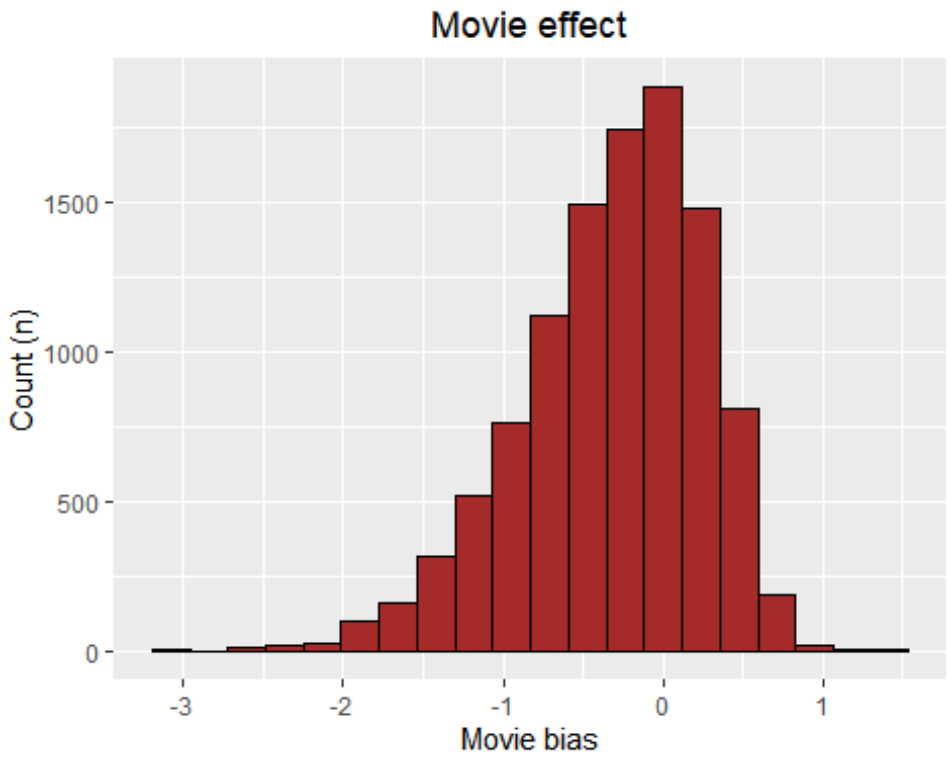
```
first_rmse <- rmse(test_dataset$rating, mean_movie_rating)

#Save RMSE result in a dataframe
results = data_frame(Method = "Model 1: Using the mean (Naive model)", RMSE =
first_rmse)
results %>% knitr::kable()
```

Method	RMSE
Model 1: Using the mean (Naive model)	1.061135

We obtained the first RMSE = 1.061135. The error is greater than 1 hence lacks accuracy. The Second Model will attempt to improve the error. We can confirm that some movies are more popular than others hence creating a bias.

3.3 Second Model (Movie Bias)

```
#We include bias_1 to represent the mean rating of the movies
movie_bias <- training_dataset %>%
  group_by(movieId) %>%
  summarize(bias_1 = mean(rating - mean_movie_rating))

#We plot movie bias
movie_bias %>% ggplot(aes(bias_1)) +
  geom_histogram(color = "black", fill = "brown", bins = 20) +
  xlab("Movie bias") +
  ylab("Count (n)")+
  ggtitle("Movie effect")+
  theme(plot.title = element_text(hjust = 0.5))
```




```
#Testing RMSE by adding the movie bias to our second model
#There is a slight improvement on our second model
predictions <- mean_movie_rating + test_dataset %>%
  left_join(movie_bias, by = "movieId") %>%
  pull(bias_1)
second_rmse <- rmse(predictions, test_dataset$rating)
results <- bind_rows(results,
                           data_frame(Method="Model 2: Mean + movie bias",
                                      RMSE = second_rmse))
results %>% knitr::kable()
```

Method	RMSE
Model 1: Using the mean (Naive model)	1.0611350
Model 2: Mean + movie bias	0.9441568

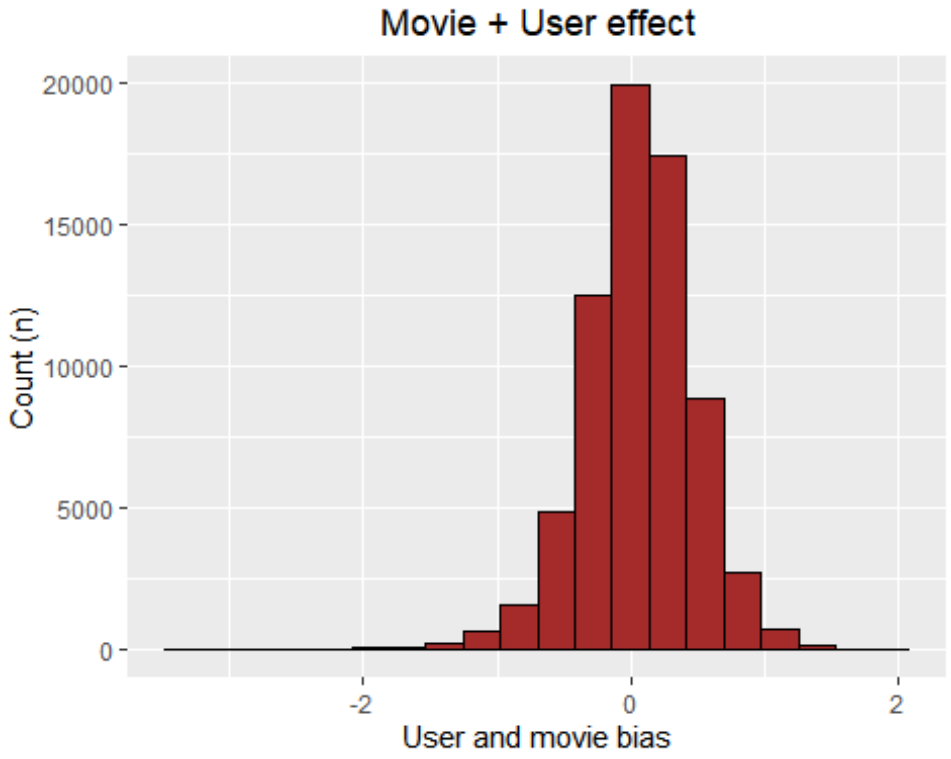
There seems to be is a slight improvement on the second model where there RMSE = 0.9441568 but it still lacks the accuracy we need. The Third Model will attempt to improve the RMSE.

3.4 Third Model (Movie & User Biases)

As seen previously users can add bias by rating some movies very highly or very low. These extreme ratings adds this bias to the model.

```
users_bias <- training_dataset %>%
  left_join(movie_bias, by = "movieId") %>%
```

```
  group_by(userId) %>%
  summarize(bias_2 = mean(rating - mean_movie_rating - bias_1))

#We plot movie + User bias
users_bias %>% ggplot(aes(bias_2)) +
  geom_histogram(color = "black", fill = "brown", bins = 20) +
  xlab("User and movie bias") +
  ylab("Count (n)")+
  ggtitle("Movie + User effect")+
  theme(plot.title = element_text(hjust = 0.5))
```




```
predictions <- test_dataset %>%
  left_join(movie_bias, by = "movieId") %>%
  left_join(users_bias, by = "userId") %>%
  mutate(new_pred = mean_movie_rating + bias_1 + bias_2) %>%
  pull(new_pred)
  third_rmse <- RMSE(predictions, test_dataset$rating)
  results <- bind_rows(results,
                       data_frame(Method="Model 3: Mean + movie + user bias",
                                  RMSE = third_rmse))
  results %>% knitr::kable()
```

Method	RMSE
Model 1: Using the mean (Naive model)	1.0611350
Model 2: Mean + movie bias	0.9441568

Model 3: Mean + movie + user bias	0.8659736

Having included movie and user biases the error is slightly lower where there RMSE = 0.8659736.

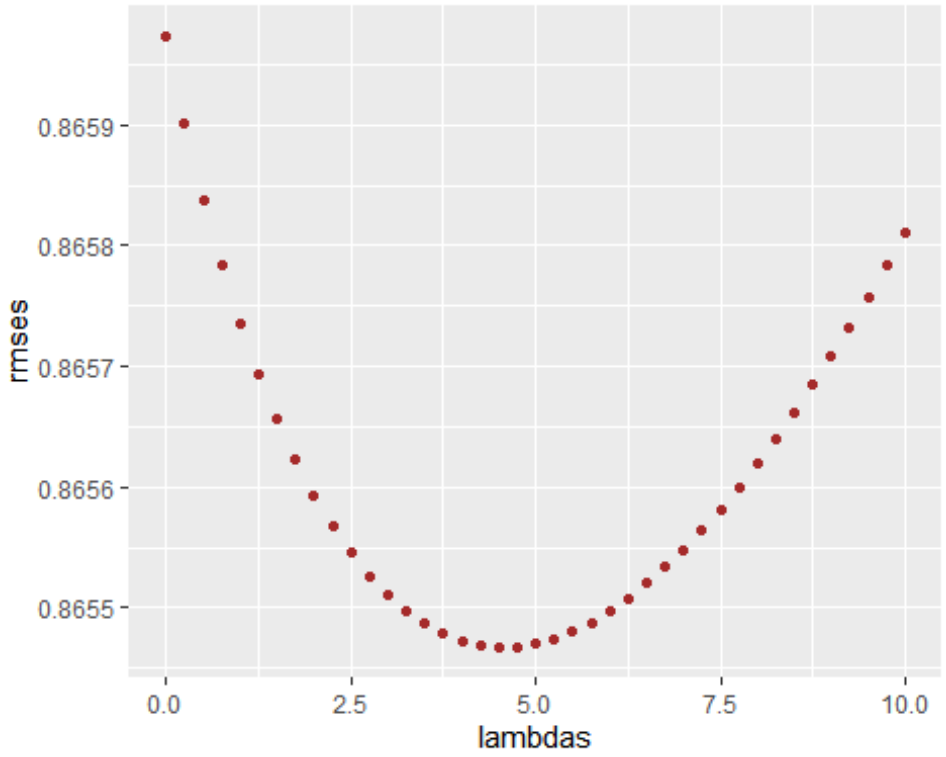
3.5 Fourth Model (Regularized Movie & User Biases)

Some movies are rated by very few users, this can increase RMSE. Regularization allows for reduced errors caused by movies with few ratings which can influence the prediction and eventually influence the error.

```
  lambdas <- seq(from=0, to=10, by=0.25)
  rmses <- sapply(lambdas, function(x){

     #Adjust mean by movie effect and penalize low number on ratings
    movie_bias <- training_dataset %>%
      group_by(movieId) %>%
      summarize(movie_bias = sum(rating - mean_movie_rating)/(n()+x))

    #Adjust mean by user + movie effect and penalize low number of ratings
    user_bias <- training_dataset %>%
      left_join(movie_bias, by = "movieId") %>%
      group_by(userId) %>%
      summarize(user_bias = sum(rating - movie_bias - mean_movie_rating)/(n()
+x))
    predictions <- test_dataset %>%
      left_join(movie_bias, by = "movieId") %>%
      left_join(user_bias, by = "userId") %>%
      mutate(new_pred = mean_movie_rating + movie_bias + user_bias) %>%
      pull(new_pred)
    return(rmse(predictions, test_dataset$rating))
  })
```

We plot RMSE against lambdas to obtain optimal lambda.

```
  qplot(lambdas, rmses, color = I("brown"))
```




```
  #We calculate the regularized accuracy with the best lambda
  lamd <- lambdas[which.min(rmses)]
  lamd #best lamda
```

```
## [1] 4.5
```

According to the plot below lambda at 4.5 produced the lowest RMSE.

```
  movie_bias <- edx %>%
    group_by(movieId) %>%
    summarize(movie_bias = sum(rating - mean_movie_rating)/(n()+lamd))

  user_bias <- edx %>%
    left_join(movie_bias, by="movieId") %>%
    group_by(userId) %>%
    summarize(user_bias = sum(rating - movie_bias - mean_movie_rating)/(n()+l
amd))

  predictions <- test_dataset %>%
    left_join(movie_bias, by = "movieId") %>%
    left_join(user_bias, by = "userId") %>%
    mutate(new_pred = mean_movie_rating + movie_bias + user_bias) %>%
    pull(new_pred)
  fourth_rmse <- rmse(predictions, test_dataset$rating)
  results <- bind_rows(results,
                       data_frame(Method="Model 4: Mean + movie + user bias +
Regularisation",
```

```
                                    RMSE = fourth_rmse))
 results %>% knitr::kable()
```

Method	RMSE
Model 1: Using the mean (Naive model)	1.0611350
Model 2: Mean + movie bias	0.9441568
Model 3: Mean + movie + user bias	0.8659736
Model 4: Mean + movie + user bias + Regularisation	0.8574155

The final RMSE obtained for the final model is 0.8574155 which is less than the target RMSE of 0.86490 hence providing a better accuracy.

4. Conclusion

We employed a 4-step approach to come up with a movie recommendation model. The final model (fourth model) is the preferred movie recommendation model since its RMSE is 0.8574155 which is below the target of 0.8649. The limitations of this project included having movies that were rated by a small number of users which were not credible enough. Similarly, users who rated only a small number of movies should not be taken into account to remove the biases. Preferably we should only consider users who have given at least 50 ratings. It is also suggested that other data such as age, user behavior, gender, etc. could further enhance the final model.