HarvardX: PH125.9x Data Science
MovieLens Rating Prediction Project

Roberto A. Aponte Rivera

December 10, 2024

Contents

1 Overview

This project completes the objective established by the MovieLens Project of the HervardX: PH125.9x Data Science: Capstone course.

Initial data wrangling and subset creation was handled by the code provided on the course. An exploratory data analysis was carried out in order further curate the data and establish a development strategy for the machine learning (ML) algorithm that could predict movie ratings. Results obtained through the application of the ML algorithms were defined and showcased. Concluding remarks are included at the end.

2 Introduction

User ratings are exploited by recommendation systems in order to make specific suggestions that might appeal to the user. Many retail companies that permit customers to rate their products and services, such as Amazon, Netflix, Spotify, and Airbnb, are able to collect massive amounts of data that can be used to predict what rating a particular user will assign to a given product/service. Products and services with high predicted ratings for a given user are later recommended to them base on past activity or likelihood to purchase.

In part, one of the reasons for Netflix's success among other streaming platforms is its strong recommendation system. To explore how such recommendation systems work, we will use the 10M version of MovieLens dataset (collected by GroupLens Research), to generate a robust movie recommendation system.

2.1 RMSE

To evaluate model performance, we will use the Root Mean Square Error (RMSE).

The RMSE measures the average difference between a statistical model's predicted values and the actual values. Mathematically, it is the standard deviation of the residuals which represent the distance between the regression line and the data points. RMSE quantifies how dispersed these residuals are, revealing how tightly the observed data clusters around the predicted values.

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

2.2 Dataset

The MovieLens dataset is downloaded from: https://grouplens.org/datasets/movielens/

2.2.1 Loading Data

```
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(kableExtra))
    install.packages("kableExtra", repos = "http://cran.us.r-project.org")

library(tidyverse)
library(caret)
library(kableExtra)

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)
```

2.2.2 Initial Data Wrangling

Code provided by the course

```
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")
```

2.2.3 Creating Data Subsets

The MovieLens dataset is divided in two subsets:

- "edx" subset: used for training and testing the model
- "final_holdout_test" subset: used for the final evaluation of the model

```
# 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)
```

3 Exploratory Data Analysis

Through EDA, we will identify the most relevant features that could be used to predict movie ratings. Further, it will help us define the most relevant strategy the predictive model should follow.

According to the GroupLens website, hosting the data sets we will use for this project, the MovieLens 10M Dataset, released in January 2009 *"contains 10,000,054 ratings and 95,580 tags applied to 10,681 movies by 71,567 users of the online movie recommender service MovieLens."*

3.1 DataSet Structure

Check the structure of the dataset to understand the variables and their types.

```
# Quick look at data frame
head(edx, 5) %>% knitr::kable(booktabs = TRUE)  %>%
  kable_styling(latex_options = c("striped", "scale_down"))
```

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

EDX Structure

```
str(edx)
```

```
## 'data.frame':    9000055 obs. of  6 variables:
##  $ userId   : int  1 1 1 1 1 1 1 1 1 1 ...
##  $ movieId  : int  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 83
##  $ title    : chr  "Boomerang (1992)" "Net, The (1995)" "Outbreak (1995)" "Stargate (1994)" ...
##  $ genres   : chr  "Comedy|Romance" "Action|Crime|Thriller" "Action|Drama|Sci-Fi|Thriller" "Action|Ac
```

There are 6 Variables in this dataset:

1. userId: **Integer**. Movielens users that were selected at random for inclusion. Their ids have been anonymised.
2. movieId: **Integer**. MovieID is the real MovieLens id.
3. rating: **Numeric**. Rating of one movie by one user. Ratings are made on a 5-star scale, with half-star increments.
4. timestamp: **Integer**. When the rating was made, expressed in seconds since midnight Coordinated Universal Time (UTC) of January 1, 1970.

5. title: **Character**. Movies titles + year of its release.
6. genres: **Character**. Genres are a pipe-separated list, and are selected from the following: Action, Adventure, Animation, Childrens, Comedy, Crime, Documentary, Drama, Fantasy, Film-Noir, Horror, Musical, Mystery, Romance, Sci-Fi, Thriller, War, Western, "no genre listed".

3.2 Summary Statistics

```
# Summarized data frame with mean rating and timestamps
summary(edx) %>% knitr::kable(booktabs = TRUE, caption = "Summary Statistics") %>%
  kable_styling(latex_options = c("striped", "scale_down"))
```

Table 2: Summary Statistics

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 dataset is clean with no missing/NA values.

We will now check the number of unique values for the "users", "movies" and "genre" variables.

```
edx %>%
  summarize(n_users = n_distinct(userId),
            n_movies = n_distinct(movieId),
            n_genres = n_distinct(genres)) %>%
            knitr::kable(caption = "Unique Values") %>%
            kable_styling(position = "center",
            latex_options = c("scale_down", "scale_down"))
```

Table 3: Unique Values

n_users	n_movies	n_genres
69878	10677	797

The data contains 9,000,055 ratings applied to 10,677 different movies of 797 different unique or combination of genre from 69,878 single users between 1995 and 2009.

To better perform our analysis and build our models , we will further wrangle the datasets as following:

- Convert the "timestamp" column to "datetime" format;
- Extract the "Release Year" observation from the "Title" column.
- Use the new "Release Year" column to add a "MovieAge" column (using year 2023 to calculate the age).

```
# Convert timestamp to datetime
edx <- edx %>%
  mutate(datetime = as.POSIXct(timestamp, origin = "1970-01-01", tz = "UTC")) %>%
  select(-timestamp) %>%
```

```
  relocate(datetime, .after = rating) %>%
  tibble()

final_holdout_test <- final_holdout_test %>%
  mutate(datetime = as.POSIXct(timestamp, origin = "1970-01-01", tz = "UTC")) %>%
  select(-timestamp)  %>%
  relocate(datetime, .after = rating) %>%
  tibble()

# Extract release year from title
edx <- edx %>%
  mutate(release_year = as.integer(str_extract(title, "(?<=\\()\\d{4}(?=\\))"))) %>%
  relocate(release_year, .after = title)


final_holdout_test <- final_holdout_test %>%
  mutate(release_year = as.integer(str_extract(title, "(?<=\\()\\d{4}(?=\\))"))) %>%
  relocate(release_year, .after = title)

# Calculate movie age
edx <- edx %>%
  mutate(movie_age = 2024 - release_year) %>%
  relocate(movie_age, .after = release_year)

final_holdout_test <- final_holdout_test %>%
  mutate(movie_age = 2024 - release_year) %>%
  relocate(movie_age, .after = release_year)
```

3.3 Users

3.3.1 Ratings per User

We will explore the number of ratings per user to understand the distribution of ratings.

```
ratings_per_user <- edx %>%
                    group_by(userId) %>%
                    summarize(nb_ratings = n()) %>%
                    ungroup()

summary(ratings_per_user) %>%
knitr::kable(booktabs = TRUE, caption = "Ratings per User Summary Statistics") %>%
kable_styling(position = "center", latex_options = c("striped", "scale_down"))
```

Table 4: Ratings per User Summary Statistics

userId	nb_ratings
Min. : 1	Min. : 10.0
1st Qu.:17943	1st Qu.: 32.0
Median :35799	Median : 62.0
Mean :35782	Mean : 128.8
3rd Qu.:53620	3rd Qu.: 141.0
Max. :71567	Max. :6616.0

A number of users rated less than 20 movies, while the majority of users rated between 20 and 100 movies.

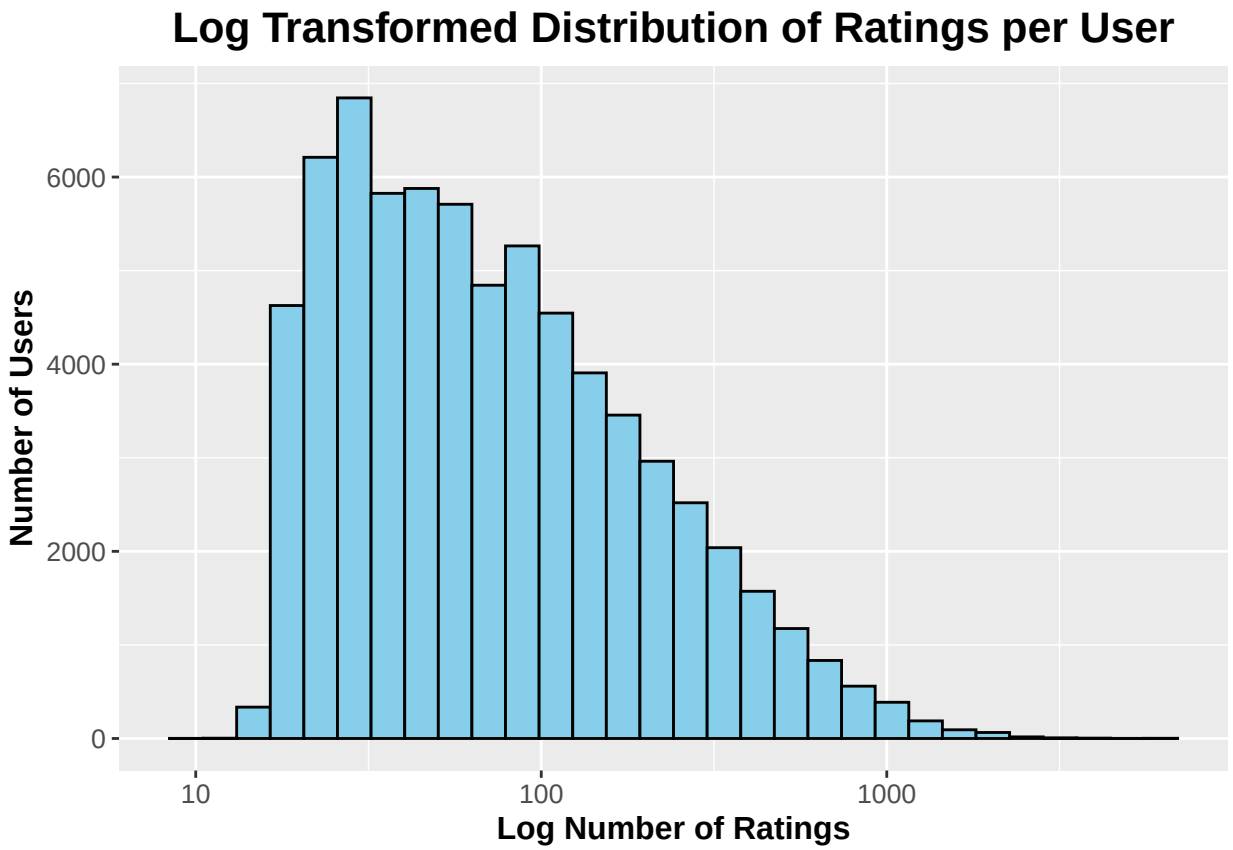
```
length(which(ratings_per_user$nb_ratings <= 20))
```

```
## [1] 4966
```

Since 4966 users rated less than 20 movies, we should consider as a potential bias affecting our models.

Visualizing the distribution of ratings per user. (We will use a log transformation to better visualize the distribution).

```
ratings_per_user %>%
  ggplot(aes(x = nb_ratings)) +
  geom_histogram(bins = 30, fill = "skyblue", color = "black") +
  scale_x_log10() +
  labs(title = "Log Transformed Distribution of Ratings per User",
       x = "Log Number of Ratings",
       y = "Number of Users") +
  theme(plot.title = element_text(size = 16 , face = "bold", hjust = 0.5)) +
  theme(axis.text=element_text(size=10),
        axis.title=element_text(size=12,face="bold"))
```



Right skewed log transformed distribution with median at 62 and mean at 129. It appears that the majority of users rated a number of movies that sits between 25 and 100 which seems somehow low: to be taken in account as a "bias" when developing our prediction models with potentially adding a "user penalty term".

3.4 Movies

3.4.1 Number of Movies vs Number of Ratings

We will explore the number of ratings per movie to understand the distribution of ratings.

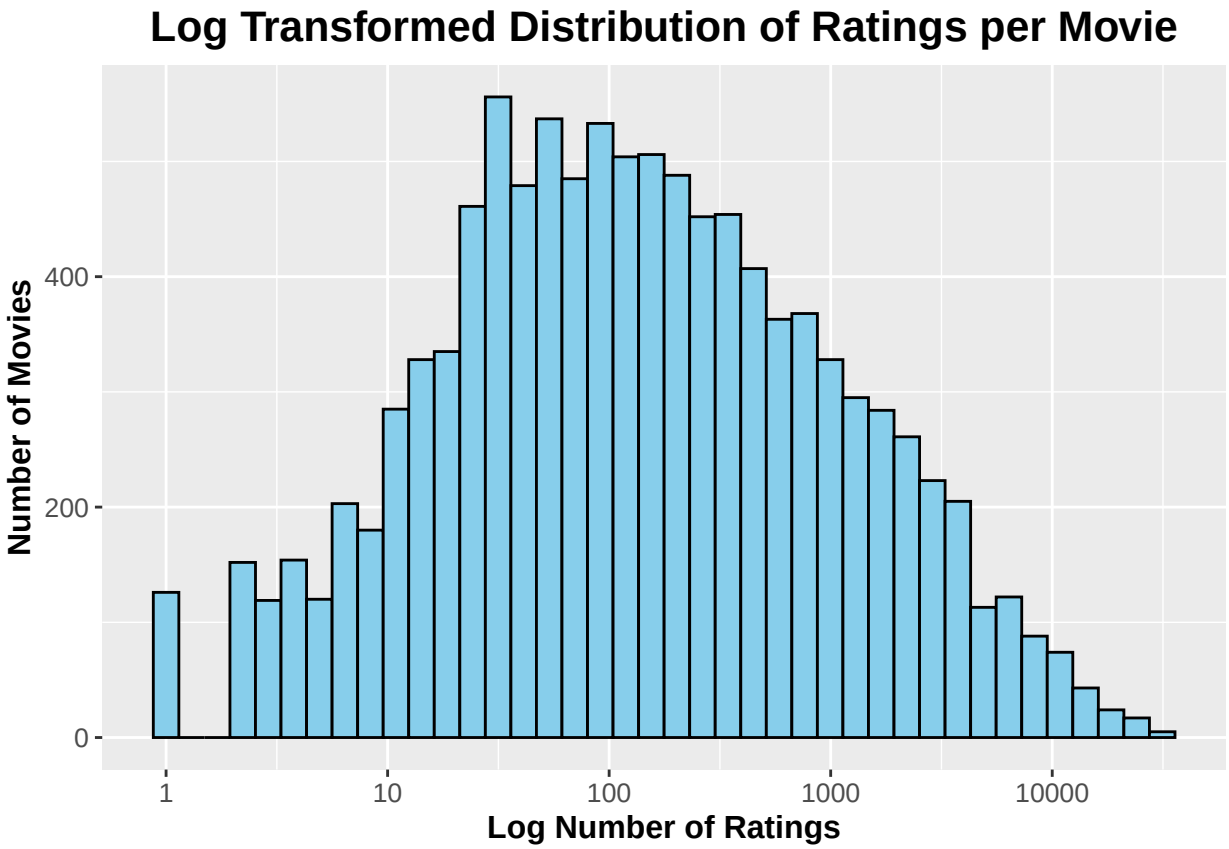
```
ratings_per_movie <- edx %>%
                     group_by(movieId) %>%
                     summarize(nb_ratings = n()) %>%
                     ungroup()

summary(ratings_per_movie) %>%
knitr::kable(booktabs = TRUE, caption = "Ratigns per Movie Summary Statistics") %>%
kable_styling(position = "center", latex_options = c("striped", "scale_down"))
```

Table 5: Ratigns per Movie Summary Statistics

movieId	nb_ratings
Min. : 1	Min. : 1.0
1st Qu.: 2754	1st Qu.: 30.0
Median : 5434	Median : 122.0
Mean :13105	Mean : 842.9
3rd Qu.: 8710	3rd Qu.: 565.0
Max. :65133	Max. :31362.0

Visualizing the distribution of ratings per movie. (We will use a log transformation to better visualize the distribution).

```
ratings_per_movie %>%
  ggplot(aes(x = nb_ratings)) +
  geom_histogram(bins = 40, fill = "skyblue", color = "black") +
  scale_x_log10() +
  labs(title = "Log Transformed Distribution of Ratings per Movie",
       x = "Log Number of Ratings",
       y = "Number of Movies") +
  theme(plot.title = element_text(size = 16 , face = "bold", hjust = 0.5)) +
  theme(axis.text=element_text(size=10),
        axis.title=element_text(size=12,face="bold"))
```



The distribution is not far from being symmetric which tends to show that popular movies are rated more frequently than less popular ones. The fact that there are a number of films with fewer ratings implies potential biases that may affect our recommendation modelling.

3.5 Ratings

3.5.1 Number of Monthly Ratings

We will explore the number of ratings per month to understand the distribution of ratings.

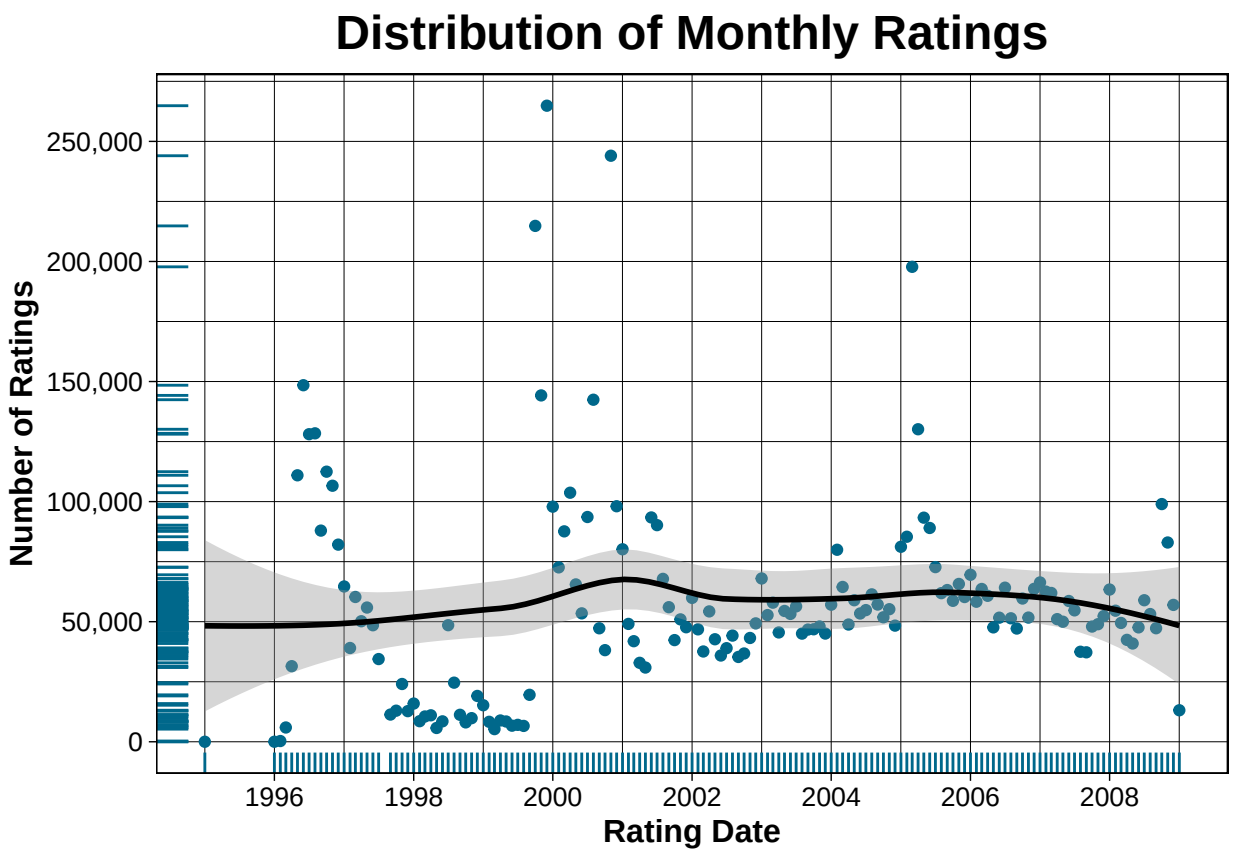
```
ratings_per_month <- edx %>%
                     mutate(datetime = as.Date(datetime),
                      datetime = make_date(year(datetime), month(datetime))) %>%
                     group_by(datetime) %>%
                     summarize(nb_ratings = n()) %>%
                     ungroup()

summary(ratings_per_month) %>%
knitr::kable(booktabs = TRUE, caption = "Monthly Ratings Summary Statistics") %>%
kable_styling(position = "center", latex_options = c("striped", "scale_down"))
```

Table 6: Monthly Ratings Summary Statistics

	datetime	nb_ratings
	Min. :1995-01-01	Min. : 2
	1st Qu.:1999-04-01	1st Qu.: 38148
	Median :2002-07-01	Median : 51841
	Mean :2002-06-25	Mean : 57325
	3rd Qu.:2005-10-01	3rd Qu.: 64464
	Max. :2009-01-01	Max. :264856

Visualizing the distribution of ratings per month.

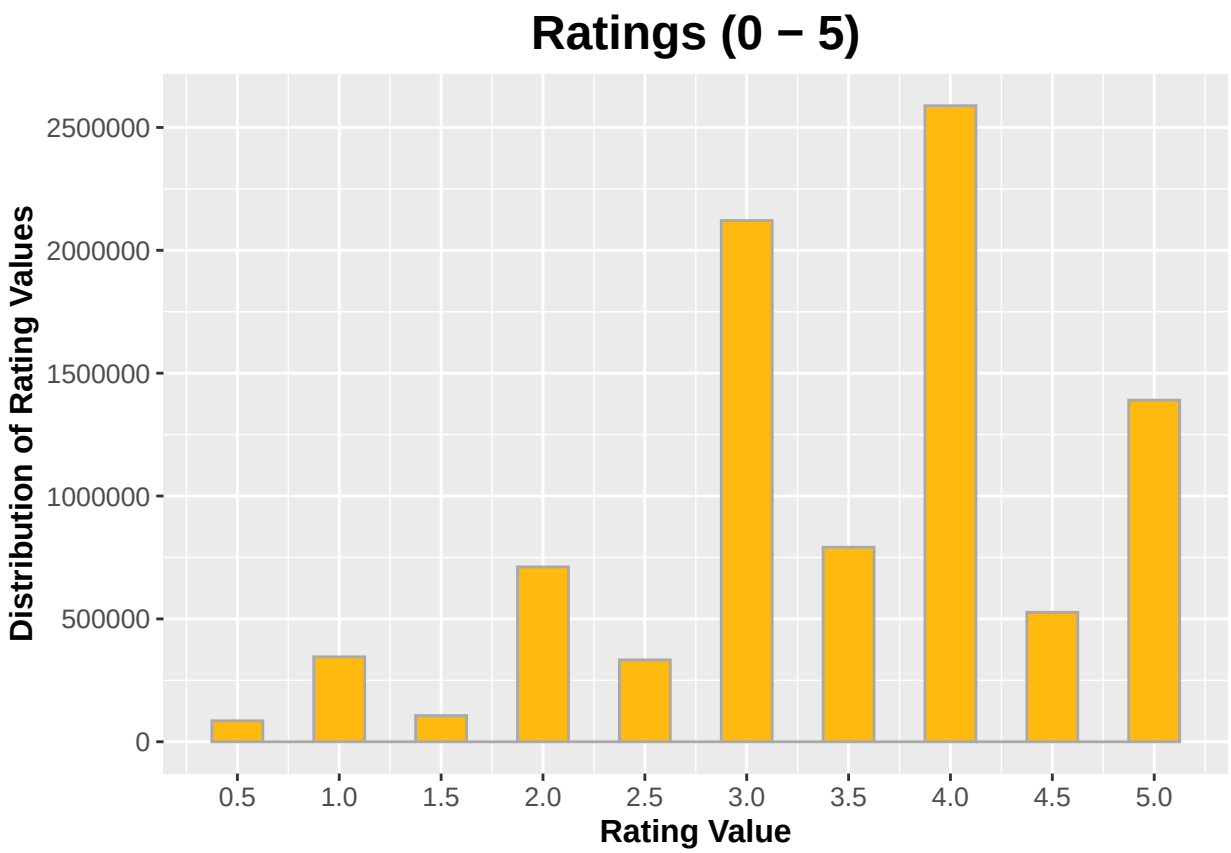
```
ratings_per_month %>%
  ggplot(aes(x = datetime, y = nb_ratings)) +
  geom_point(color = "deepskyblue4", size = 1.5) +
  scale_x_date(date_breaks = "2 years", date_labels = "%Y") +
  scale_y_continuous(breaks = seq(0, 300000, 50000), labels = scales::comma) +
  labs(title = "Distribution of Monthly Ratings",
         x = "Rating Date",
         y = "Number of Ratings") +
  geom_rug(color = "deepskyblue4") +
  geom_smooth(color = "black") +
  theme_linedraw() +
  theme(plot.title = element_text(size = 18 , face = "bold", hjust = 0.5)) +
  theme(axis.text=element_text(size=10),
        axis.title=element_text(size=12,face="bold"))
```



3.5.2 Distribution of Ratings

We will explore the distribution of ratings.

```
edx %>%
  ggplot(aes(rating)) +
  geom_histogram(binwidth = 0.25, fill = "darkgoldenrod1", color = "darkgray") +
  scale_x_continuous(breaks = seq(0.5, 5, 0.5)) +
  ylab("Distribution of Rating Values") +
  xlab("Rating Value") +
  scale_y_continuous(breaks = c(seq(0, 3000000, 500000))) +
  ggtitle("Ratings (0 - 5)") +
  theme(plot.title = element_text(size = 18 , face = "bold", hjust = 0.5)) +
  theme(axis.text=element_text(size=10),
        axis.title=element_text(size=12,face="bold"))
```



There are 10 different ratings users can award a movie: 0.5, 1, 1.5, 2, 2.5, 3, 3.5, 4, 4.5 and 5. A rating of "4" is the most popular rating followed by "3", "5", "3.5", etc. while "0.5" was awarded the least frequently.

3.5.3 Average Rating per User

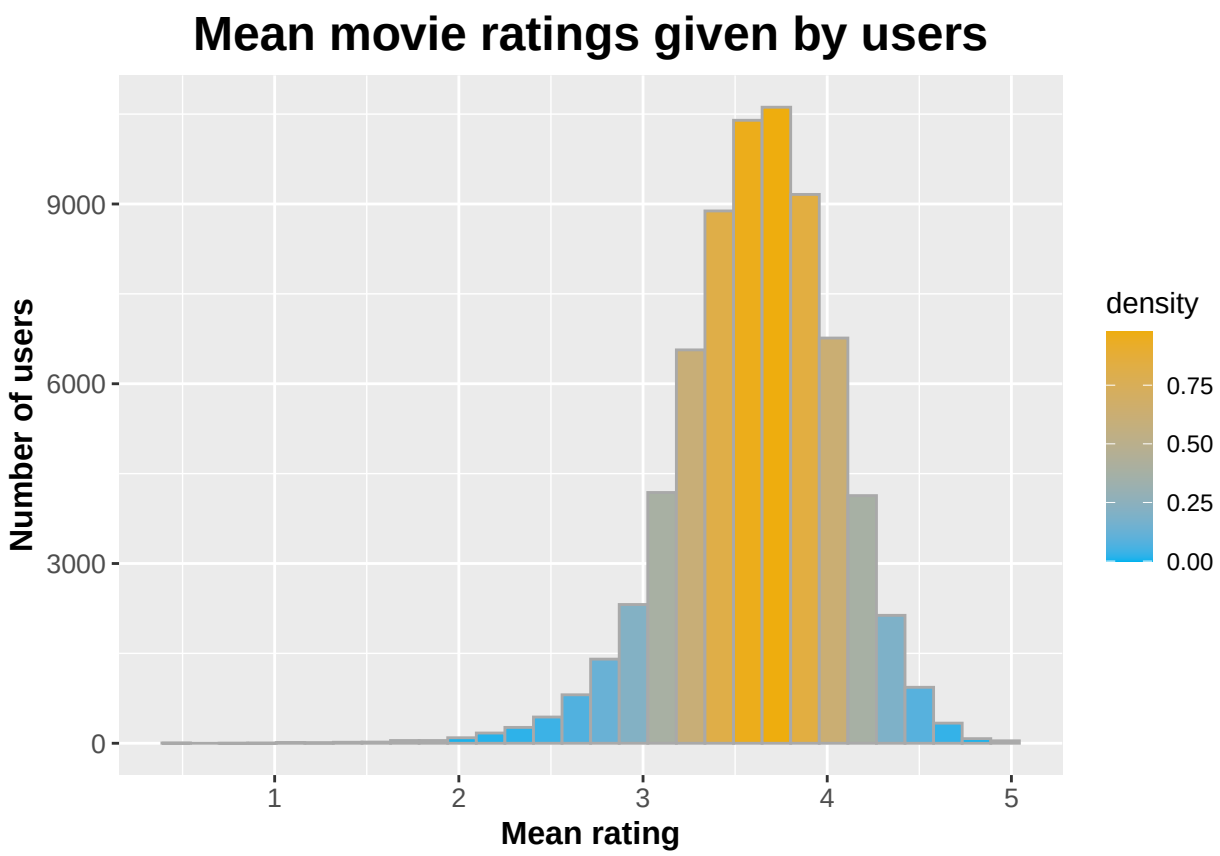
We will explore the average rating per user.

```
avg_rating_per_user <- edx %>%
                       group_by(userId) %>%
                       summarize(avg_rating = mean(rating)) %>%
                       ungroup()

summary(avg_rating_per_user) %>%
knitr::kable(booktabs = TRUE, caption = "Summary Statistics") %>%
  kable_styling(position = "center", latex_options = c("striped", "scale_down"))
```

Table 7: Summary Statistics

userId	avg_rating
Min. : 1	Min. :0.500
1st Qu.:17943	1st Qu.:3.357
Median :35799	Median :3.635
Mean :35782	Mean :3.614
3rd Qu.:53620	3rd Qu.:3.903
Max. :71567	Max. :5.000

```
avg_rating_per_user %>%
  ggplot(aes(x = avg_rating, fill = after_stat(density))) +
  geom_histogram(bins = 30, color = "darkgray") +
  scale_fill_gradient(low="deepskyblue2", high="darkgoldenrod2") +
  xlab("Mean rating") +
  ylab("Number of users") +
  ggtitle("Mean movie ratings given by users") +
  theme(plot.title = element_text(size = 18 , face = "bold", hjust = 0.5)) +
  theme(axis.text=element_text(size=10),
        axis.title=element_text(size=12,face="bold"))
```



The distribution is symmetric and, as seen previously, centred around 3.5 which shows a tendency for users to favour rating movies they appreciated rather than the one they disliked.

3.6 Movie Age

3.6.1 Mean Rating vs. Movie Age

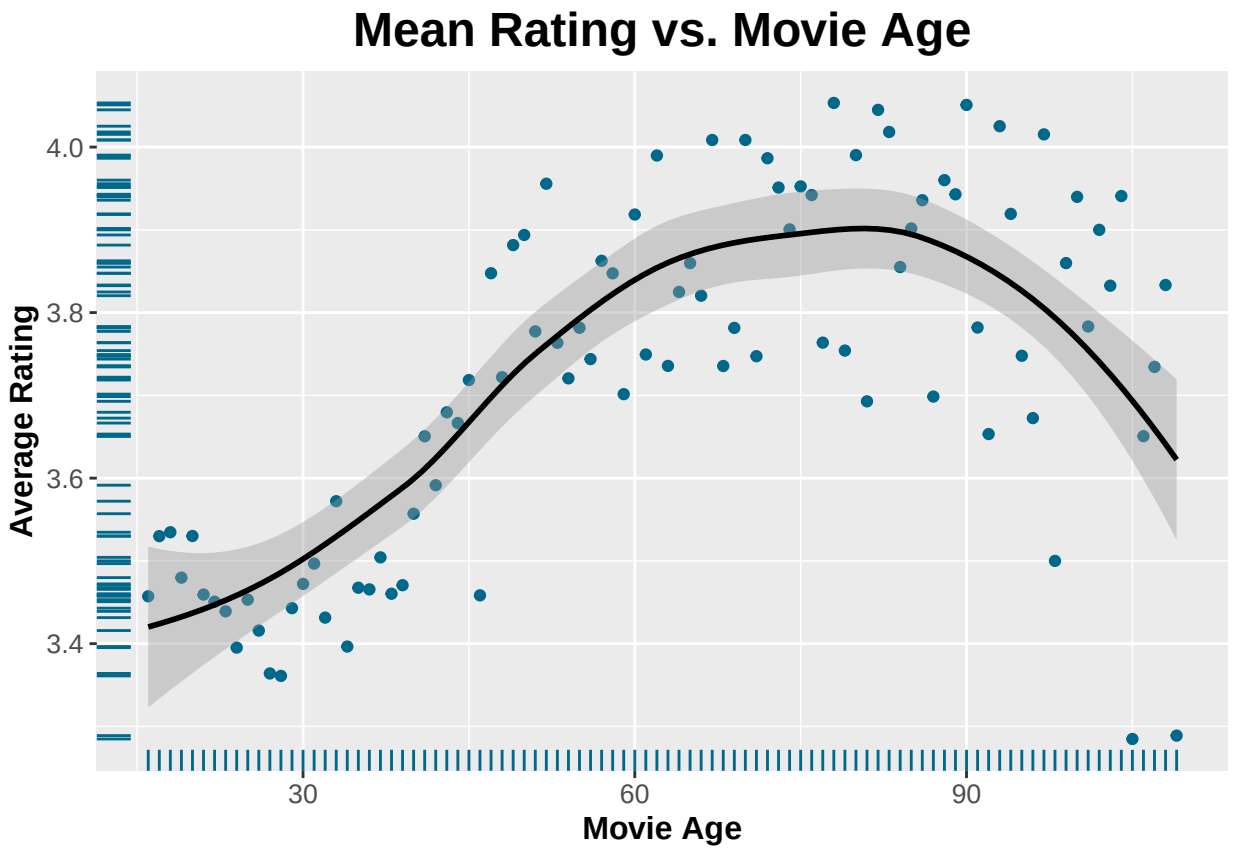
We will explore whether the age of the movie affects the overall rating given by users.

```
ratings_avg_movie_age <- edx %>%
                         group_by(movie_age) %>%
                         summarize(avg_rating = mean(rating)) %>%
                         ungroup()

summary(ratings_avg_movie_age) %>%
knitr::kable(booktabs = TRUE, caption = "Summary Statistics") %>%
kable_styling(position = "center", latex_options = c("striped", "scale_down"))
```

Table 8: Summary Statistics

movie_age	avg_rating
Min. : 16.00	Min. :3.285
1st Qu.: 39.25	1st Qu.:3.511
Median : 62.50	Median :3.748
Mean : 62.50	Mean :3.721
3rd Qu.: 85.75	3rd Qu.:3.900
Max. :109.00	Max. :4.053

```
ratings_avg_movie_age %>%
ggplot(aes(x = movie_age, y = avg_rating)) +
  geom_point(color = "deepskyblue4", size = 1.5) +
  geom_smooth(color = "black") +
  geom_rug(color = "deepskyblue4") +
  labs(title = "Mean Rating vs. Movie Age",
       x = "Movie Age",
       y = "Average Rating") +
  theme(plot.title = element_text(size = 18 , face = "bold", hjust = 0.5)) +
  theme(axis.text=element_text(size=10),
        axis.title=element_text(size=12,face="bold"))
```



The correlation between the two features seems quite clear with users tending to award higher ratings to older movies rather than to newer releases. While probably interesting, this effect / bias will be taken in account if we don't reach our RMSE targeted value through simpler models.

4 Machine Learning Algorithms

4.1 Benchmark model

As a benchmark, we will naively predict all ratings as the average rating of the trainning set. The formula is defined as follows: Y_{ui} = predicted rating, μ = average rating, and ϵ_{ui} = independent errors centered at 0.

$$Y_{u,i} = \mu + \epsilon_{u,i}$$

```
# Compute the dataset's mean rating
mu <- mean(edx$rating)

# Test results based on simple prediction
naive_rmse <- RMSE(edx$rating, mu)

# Check results
# Save prediction in data frame
rmse_results <- tibble(method = "Average movie rating model", RMSE = naive_rmse)
rmse_results %>% knitr::kable()
```

method	RMSE
Average movie rating model	1.060331

The RMSE for this model is **1.0603**, which is unsurprisingly high.This will nevertheless be used as a benchmark for comparison with the different models we will now design.

4.2 "Movie Bias" Model

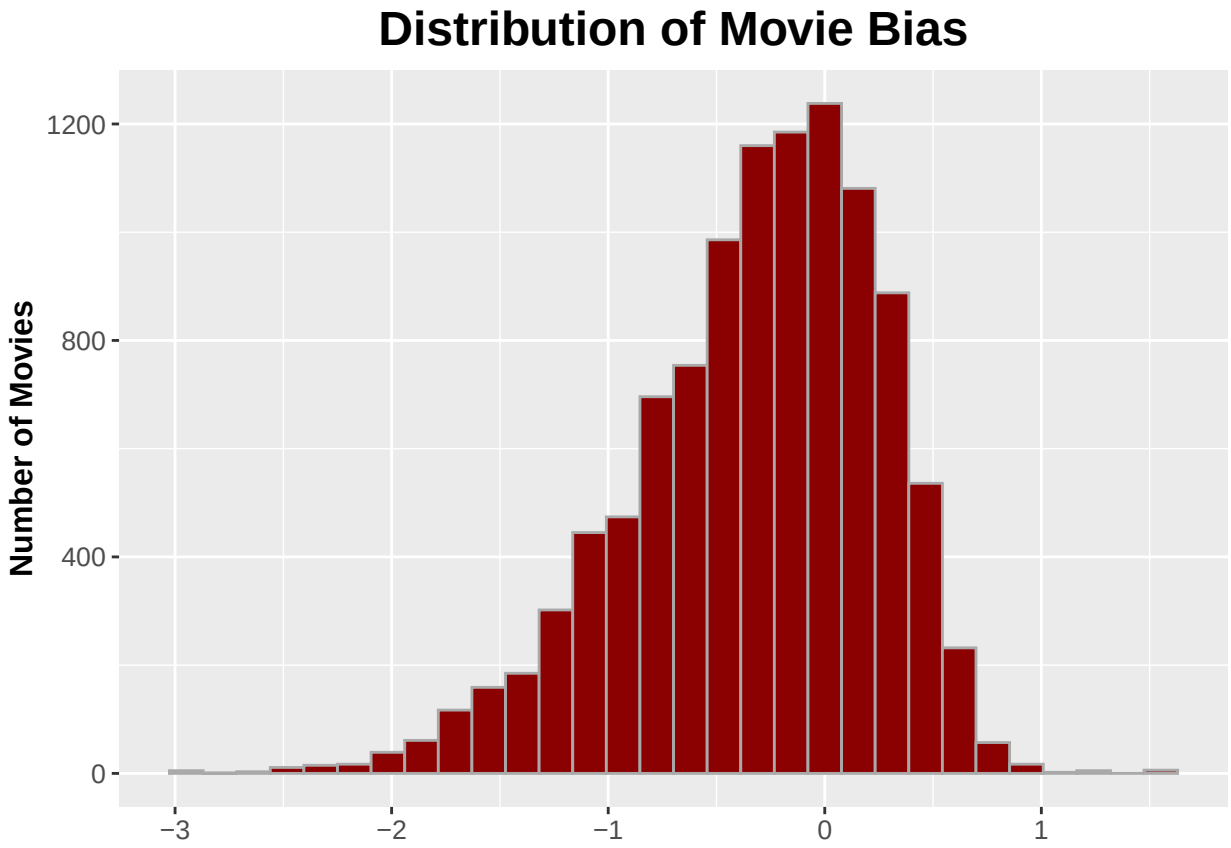
To improve our model we will take in account the idea that some movies are subjectively rated higher than others. Higher ratings are mostly linked to popular movies among users and the opposite is true for unpopular movies. We compute the estimated deviation of each movie's mean rating from the total mean of all movies μ. The resulting variable is called b_m with "b" for *bias* and "m" for *movie*. The formula is defined as:

$$Y_{u,i} = \mu + b_m + \epsilon_{u,i}$$

With $Y_{u,i}$ = predicted rating, μ = average rating, b_m = "movie bias" variable and ϵ_{ui} = independent errors centered at 0.

Let's visualize the b_m distribution:

```
movie_bias <- edx %>%
  group_by(movieId) %>%
  summarize(b_i = mean(rating - mu))
movie_bias %>%
ggplot(aes(b_i)) +
geom_histogram(bins=30, color = "darkgray", fill = "darkred") +
ylab("Number of Movies")  +
ggtitle("Distribution of Movie Bias") +
theme(plot.title = element_text(size = 18 , face = "bold", hjust = 0.5)) +
  theme(axis.text=element_text(size=10),
        axis.title=element_text(size=12,face="bold"),
        axis.title.x = element_blank())
```



We will now check the prediction against the edx set to determine the related RMSE:

```
# Test and save rmse results
predicted_ratings <- mu + edx %>%
  left_join(movie_bias, by='movieId') %>%
  pull(b_i)
model_2_rmse <- RMSE(edx$rating, predicted_ratings)
rmse_results <- bind_rows(rmse_results,
                          tibble(method="Movie Bias model",
                                 RMSE = model_2_rmse ))
# Check results
rmse_results %>% knitr::kable()
```

method	RMSE
Average movie rating model	1.0603313
Movie Bias model	0.9423475

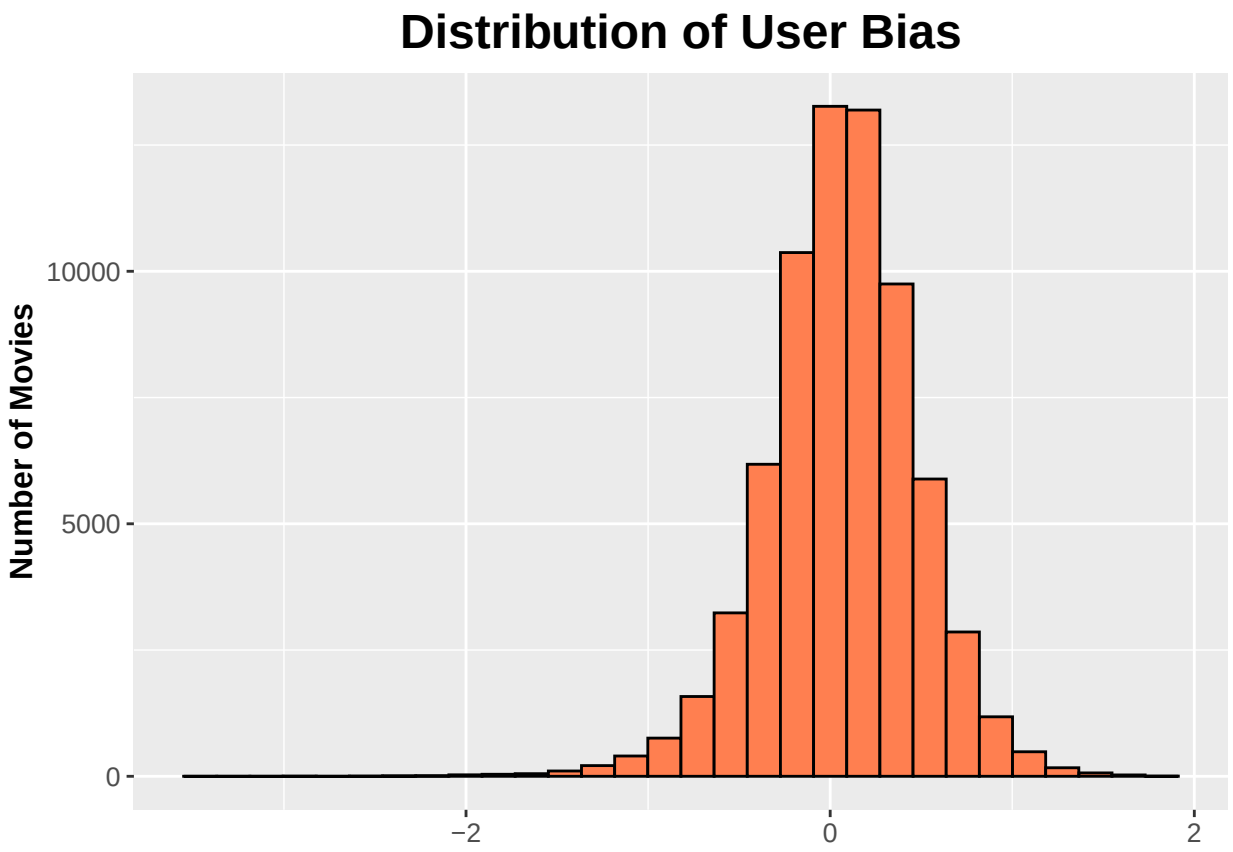
4.3 "Movie & User Biases" Model

This model introduces "User Effect / Bias" that reflects the fact that individual users tend to rate films according to their own standards (which vary widely in distribution). The formula can be defined as follows with $Y_{u,i}$ = predicted rating, μ = average rating, b_m = "movie bias" variable, b_u = "user bias" variable and $\epsilon_{u,i}$ = independent errors centered at 0.

$$Y_{u,i} = \mu + b_m + b_u + \epsilon_{u,i}$$

Let's visualize the b_u distribution:

```
user_bias <- edx %>%
  left_join(movie_bias, by='movieId') %>%
  group_by(userId) %>%
  summarize(b_u = mean(rating - mu - b_i))

user_bias %>%
ggplot(aes(b_u)) +
geom_histogram(bins = 30, color = "black", fill = "coral") +
ylab("Number of Movies") +
ggtitle("Distribution of User Bias") +
theme(plot.title = element_text(size = 18 , face = "bold", hjust = 0.5)) +
  theme(axis.text=element_text(size=10),
        axis.title=element_text(size=12,face="bold"),
        axis.title.x = element_blank())
```



We will now check the prediction against the edx set to determine the related RMSE:

```
# Test and save rmse results
predicted_ratings <- edx %>%
left_join(movie_bias, by='movieId') %>%
left_join(user_bias, by='userId') %>%
mutate(pred = mu + b_i + b_u) %>%
pull(pred)

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

```
rmse_results <- bind_rows(rmse_results, tibble(method="Movie and User Bias model",
                          RMSE = model_3_rmse))

# Check result
rmse_results %>% knitr::kable()
```

method	RMSE
Average movie rating model	1.0603313
Movie Bias model	0.9423475
Movie and User Bias model	0.8567039

Accounting for both movie and user biases, the RMSE is reduced to **0.8567**. This model is more accurate than the previous ones, but we will continue to explore other models to further improve our predictions.

4.4 Regularised "Movie & User" Biases Model

We will now introduce the concept of regularisation: the idea is to add a tuning parameter λ to further reduce the RMSE. The idea is penalise outliers from the *Movie Bias* and *User Bias* sets which shall optimise the recommendation system.

First we define λ:

```
lambdasReg <- seq(0, 10, 0.25)

# Function to test lambdas
RMSEreg <- sapply(lambdasReg, function(l){

  mu <- mean(edx$rating)

  b_m <- edx %>%
    group_by(movieId) %>%
    summarize(bm = sum(rating - mu)/(n() + l))

  b_u <- edx %>%
    left_join(b_m, by = 'movieId') %>%
    group_by(userId) %>%
    summarize(bu = sum(rating - bm - mu)/(n() + l))

  predicted_ratings <- edx %>%
    left_join(b_m, by = "movieId") %>%
    left_join(b_u, by = "userId") %>%
    mutate(pred = mu + bm + bu) %>%
    pull(pred)

return(RMSE(edx$rating, predicted_ratings))
})
```

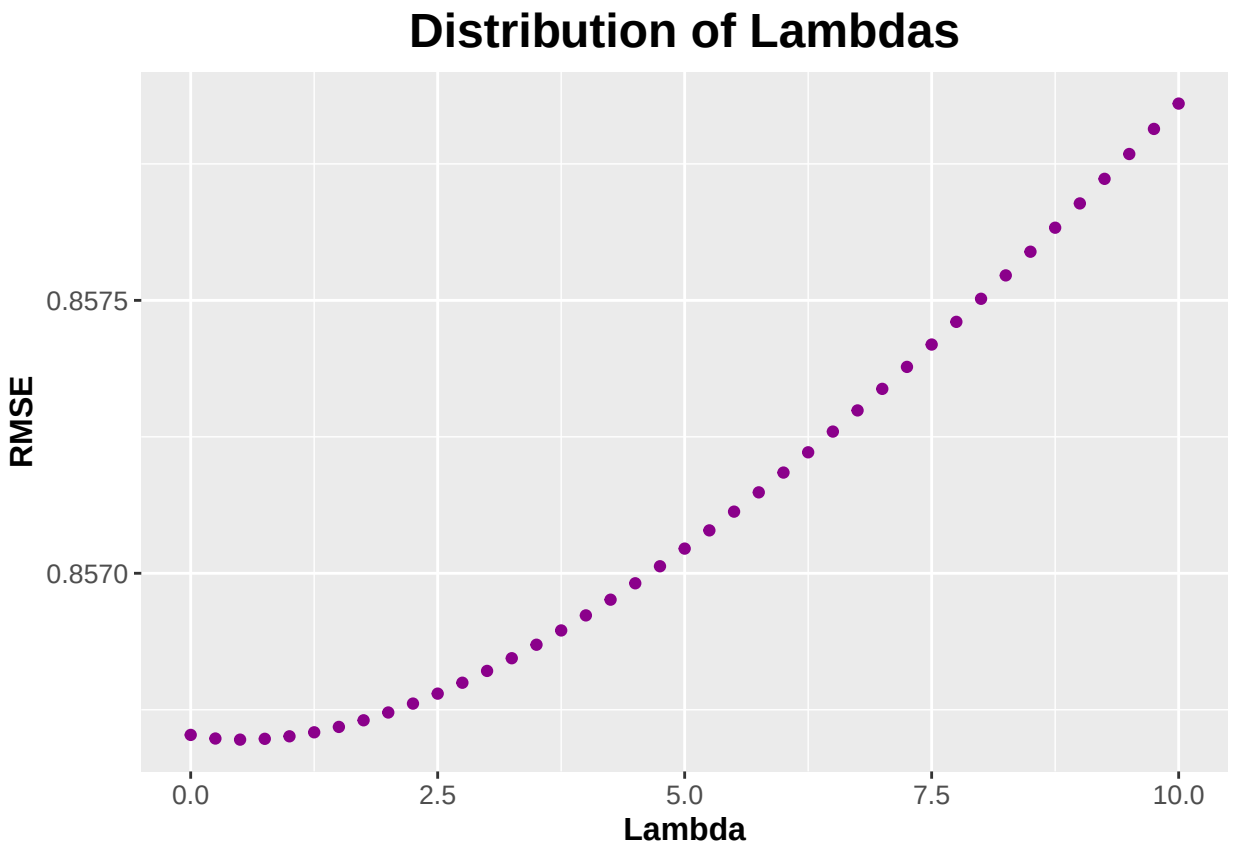
Now we determine which λ will be best at reducing the RMSE.

```
lambda <- lambdasReg[which.min(RMSEreg)]

lambda
```

```
## [1] 0.5
```

Let's also visualize the distribution of λ.

```
ggplot(mapping = aes(x = lambdasReg, y = RMSEreg)) +
  geom_point(color = "darkmagenta", size = 1.5) +
  labs(title = "Distribution of Lambdas",
         x = "Lambda",
         y = "RMSE") +
         theme(plot.title = element_text(size = 18 , face = "bold", hjust = 0.5)) +
         theme(axis.text=element_text(size=10),
        axis.title=element_text(size=12,face="bold"))
```



We will use 0.5 for the tuning parameter λ to further reduce our RMSE of the following "Regularised Movie & User Biases" model.

```
b_i_reg <- edx %>%
  group_by(movieId) %>%
  summarize(b_i = sum(rating - mu)/(n() + lambda))

b_u_reg <- edx %>%
  left_join(b_i_reg, by = "movieId") %>%
  group_by(userId) %>%
  summarize(b_u = sum(rating - b_i - mu)/(n() + lambda))

pred_reg <- edx %>%
  left_join(b_i_reg, by = "movieId") %>%
  left_join(b_u_reg, by = "userId") %>%
  mutate(pred = mu + b_i + b_u) %>%
  pull(pred)
```

```
RMSE_4 <- RMSE(edx$rating, pred_reg)

result4_table <- tibble(Model = "Regularised Movie & User Biases", RMSE = RMSE_4)

result4_table %>% knitr::kable()
```

Model	RMSE
Regularised Movie & User Biases	0.8566952

This model shows a marginal improvement of from the previous one with a RMSE at **0.85669**.

Let's summarize the RMSEs of the different models we developed:

```
results_table <- tibble(Model = c("Benchmark", "Movie Bias",
    "Movie & User Biases", "Regularised Movie & User Biases"),
      RMSE = c(naive_rmse, model_2_rmse, model_3_rmse, RMSE_4))

results_table %>% knitr::kable()
```

Model	RMSE
Benchmark	1.0603313
Movie Bias	0.9423475
Movie & User Biases	0.8567039
Regularised Movie & User Biases	0.8566952

5 Results

We will now apply the "Regularised Movie and User Biases" Model (the one with lowest RMSE yet) to the "final_holdout_test" dataset. Since this dataset comprises only 10 of the edx dataset observations, it will very likely reduce the RMSE.

```
b_if <- final_holdout_test %>%
  group_by(movieId) %>%
  summarize(b_i = sum(rating - mu)/(n() + lambda))

b_uf <- final_holdout_test %>%
  left_join(b_if, by = "movieId") %>%
  group_by(userId) %>%
  summarize(b_u = sum(rating - b_i - mu)/(n() + lambda))

pred_regf <- final_holdout_test %>%
  left_join(b_if, by = "movieId") %>%
  left_join(b_uf, by = "userId") %>%
  mutate(pred = mu + b_i + b_u) %>%
  pull(pred)

RMSE_finalHT <- RMSE(final_holdout_test$rating, pred_regf)

resultfinal2_table <- tibble(Model = "Regularised Movie & User Biases",
                             RMSE = RMSE_finalHT)

resultfinal2_table %>% knitr::kable()
```

Model	RMSE
Regularised Movie & User Biases	0.8258487

By producing a model with an RMSE of **0.8258**, the have achieved our exercise objective.

5.1 Conclusion

By using EDA, we were able to determine key aspects of the data that allowed us to produce a succesful model to predict movie rating in the MovieLens Dataset. We could have further refined the model by taking into account other aspects such as "Movie Age", "Genre", "Gender" of user or even omitted the 4966 users that rated less than 20 movies. However, in the world of machine learning, more data is better than none. Furthermore, more complex models could be built to achieve a significantly lower RMSE, such as a collaborative filtering model (*neighborhood method*) or a latent factor model. Regardless, we were able to achive the objective of this assignment solely refining a very basic model.

6 Appendix - Enviroment

```
print("Operating System:")
```

```
## [1] "Operating System:"
```

```
version
```

```
##                _
## platform       x86_64-w64-mingw32
## arch           x86_64
## os             mingw32
## crt            ucrt
## system         x86_64, mingw32
## status
## major          4
## minor          4.2
## year           2024
## month          10
## day            31
## svn rev        87279
## language       R
## version.string R version 4.4.2 (2024-10-31 ucrt)
## nickname       Pile of Leaves
```