Capstone project, MovieLens Rating Prediction Project Report

Eyad Alkronz

April 24th, 2021

Table of Contents

Overview 1
- Project Introduction 2
- The Data 2

Methods and Analysis 4
- Data Analysis 4
- Modeling 8
  - The Average Movie Rating Naive Model 8
  - The Movie Effect Model 9
- The Movie and User Effect model 11
- Regularized Movie & User Effect Model 13

Results 15

Discussion 15

Conclusion 15

Appendix 16
- Environment 16

Overview

This project is part of the HarvardX course PH125.9x Data Science: Capstone project. With the increasing popularity of streaming services, online news sites, social media, and the internet overall, recommendation systems become part of every service, website, or blog. The goal of these systems is to offer the user more content that will appeal to him and keep him on the website consuming more content. One application of machine learning is to predict and make recommendations to the user. The user will potentially like these recommendations, navigate to the relevant content, and consume more of it. Then the given dataset will be prepared and setup. An exploratory data analysis is carried out in order to develop a machine learning algorithm that could predict movie ratings until a final model. Results will be explained. Finally the report ends with some concluding remarks.

Project Introduction

The project's goal is to use this dataset to predict the score a user will give a particular movie. There are various biases to consider when facing this problem. These can be social, geographical, cultural, psychological, and more. Each one of these can change the likings of every user. I will train four different machine learning algorithms using the dataset. Starting with the most simple prediction algorithm, which is just the mean of the ratings, then considering different effects and combining them with other biases. The algorithms will use the Root Mean Squared Error (RMSE) to evaluate performance. It is a way to measure the difference between the value observed to the value predicted, and the goal is to get it as low as possible. The RMSE formula:

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

N is the sample size. The term inside the paranthesis is the difference, or distance, by rating between the predicted rating to the observed rating, squared. The target RMSE for this project is lower than 0.86490.

The Data

The data used for the project is the 10M version of MovieLens dataset, collected by GroupLens research lab at the University of Minnesota. The data contains movies, users, ratings, genres, and times. The following code is included in the HarvardX capstone project course. This code splits the data to a training set and a validation set. The training set, edx in the code, is used to train the algorithms. The validation set, validation in the code, is used to test the algorithms on new data ("the real world").

```
##########################################################
# Create edx set, validation set (final hold-out test set)
##########################################################

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

# 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/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")

# if using R 3.6 or earlier
movies <- as.data.frame(movies) %>% mutate(movieId =
as.numeric(levels(movieId))[movieId],
                                           title = as.character(title),
                                           genres = as.character(genres))
# if using R 4.0 or later
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")

# Validation set will be 10% of MovieLens data
set.seed(1, sample.kind="Rounding")
# if using R 3.5 or earlier, use `set.seed(1)` instead
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 validation set are also in edx set
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)
edx <- rbind(edx, removed)

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

Methods and Analysis

Data Analysis

Looking at the first few rows of the "edx", we can see the features which are "userId", "movieId", "rating", "timestamp", "title", and "genres". Each row represents a single rating a unique user gave a particular movie.

```
##    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)
## 6:      1     355      5 838984474       Flintstones, The (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
## 6:       Children|Comedy|Fantasy
```

A summary of the data can confirm that there are no missing values.

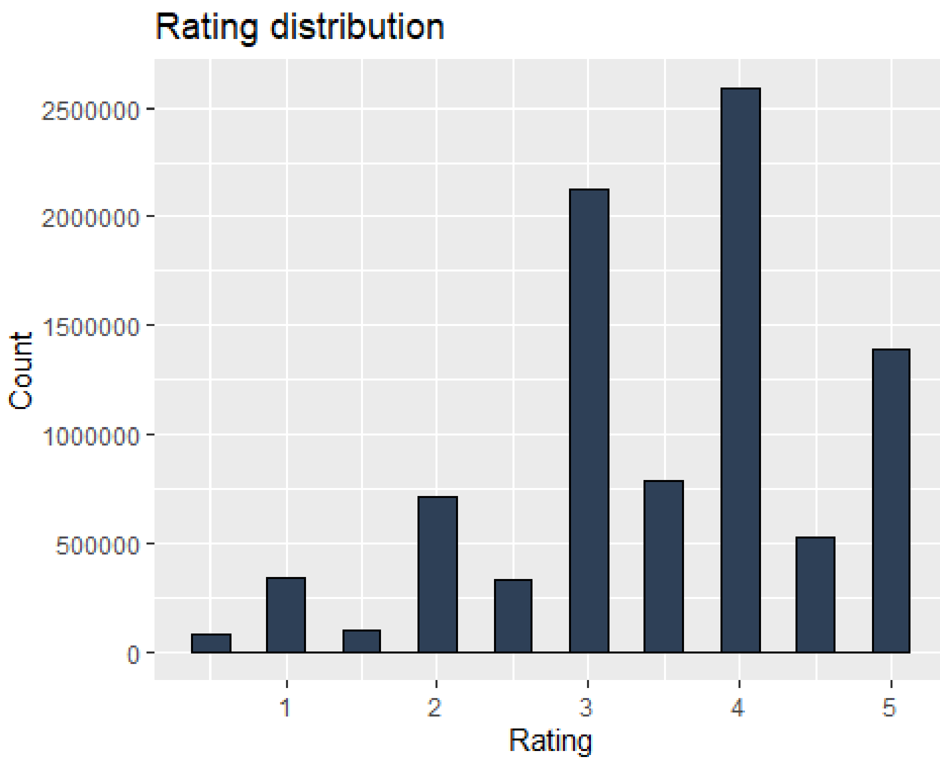
```
##      userId         movieId          rating        timestamp
##  Min.   :    1   Min.   :    1   Min.   :0.50   Min.   :7.90e+08
##  1st Qu.:18124   1st Qu.:  648   1st Qu.:3.00   1st Qu.:9.47e+08
##  Median :35738   Median : 1834   Median :4.00   Median :1.04e+09
##  Mean   :35870   Mean   : 4122   Mean   :3.51   Mean   :1.03e+09
##  3rd Qu.:53607   3rd Qu.: 3626   3rd Qu.:4.00   3rd Qu.:1.13e+09
##  Max.   :71567   Max.   :65133   Max.   :5.00   Max.   :1.23e+09
##     title              genres
##  Length:9000055     Length:9000055
##  Class :character   Class :character
##  Mode  :character   Mode  :character
##
##
##
```

The total number of unique users is about 70,000. The number of unique movies is about 10,700.

```
##   num_users num_movies
## 1     69878      10677
```

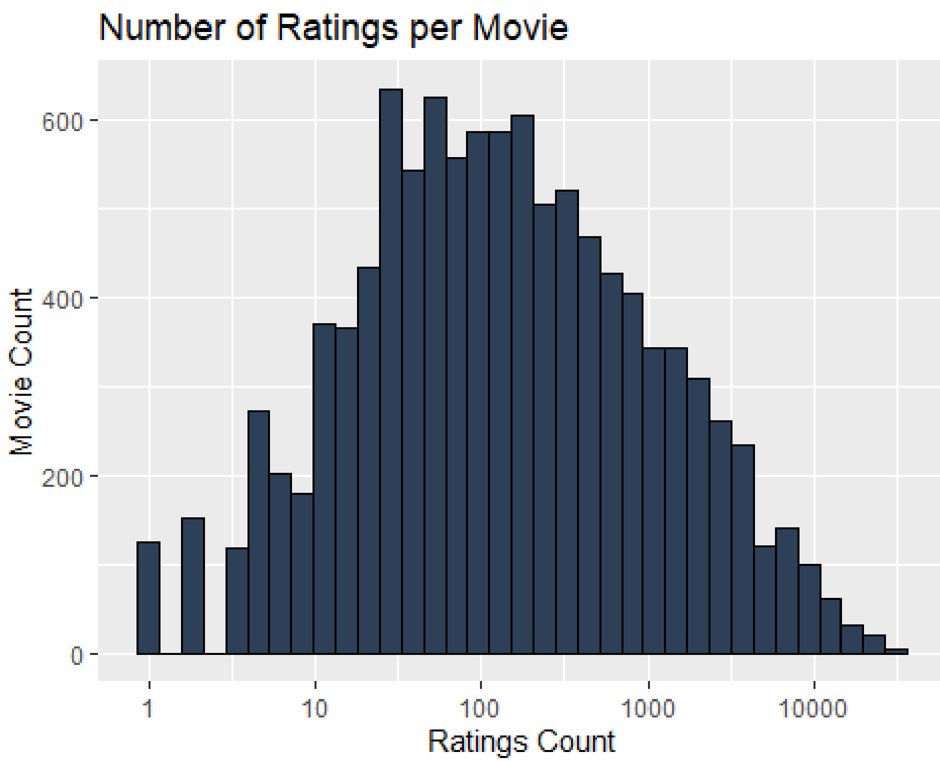
Ratings are on a scale of 0.5 to 5 with increments of 0.5. Users tend to rate movies relatively high. The most common rating is a rating of 4, followed by a rating of 3. 0.5 is the least common rating. We can deduce from the chart that users tend to use whole numbers to rate movies.

```r
# Ratings distribution
edx %>%
  ggplot(aes(rating)) +
  geom_histogram(binwidth = 0.25, color = "black", fill = "#2e4057") +
  xlab("Rating") +
  scale_y_continuous(breaks = c(seq(0, 3000000, 500000))) +
  ylab("Count") +
  ggtitle("Rating distribution")
```



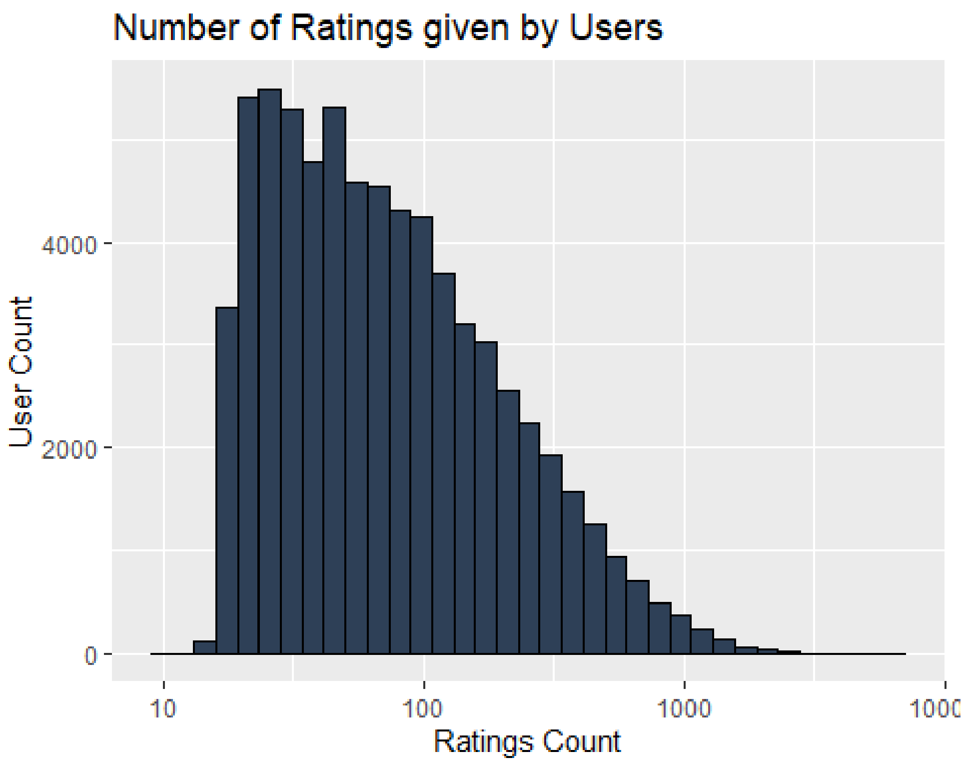
Users watch some movies more than others, for example, blockbuster films. Therefore they are being rated a lot more and usually higher than others. On the other hand, users rate some movies very few times, and some movies only once. These facts can result in not to be trusted estimations, especially for movies that are rated a few times. This term controls the function used for prediction, which is very fluctuating, such that the coefficients don't take extreme values. Nonetheless, predict for movies rated only once will be challenging.

```r
# Number of ratings per Movie
edx %>%
  count(movieId) %>%
  ggplot(aes(n)) +
  geom_histogram(bins = 35, color = "black", fill = "#2e4057") +
  xlab("Ratings Count") +
  scale_x_log10() +
  ylab("Movie Count") +
  ggtitle("Number of Ratings per Movie")
```



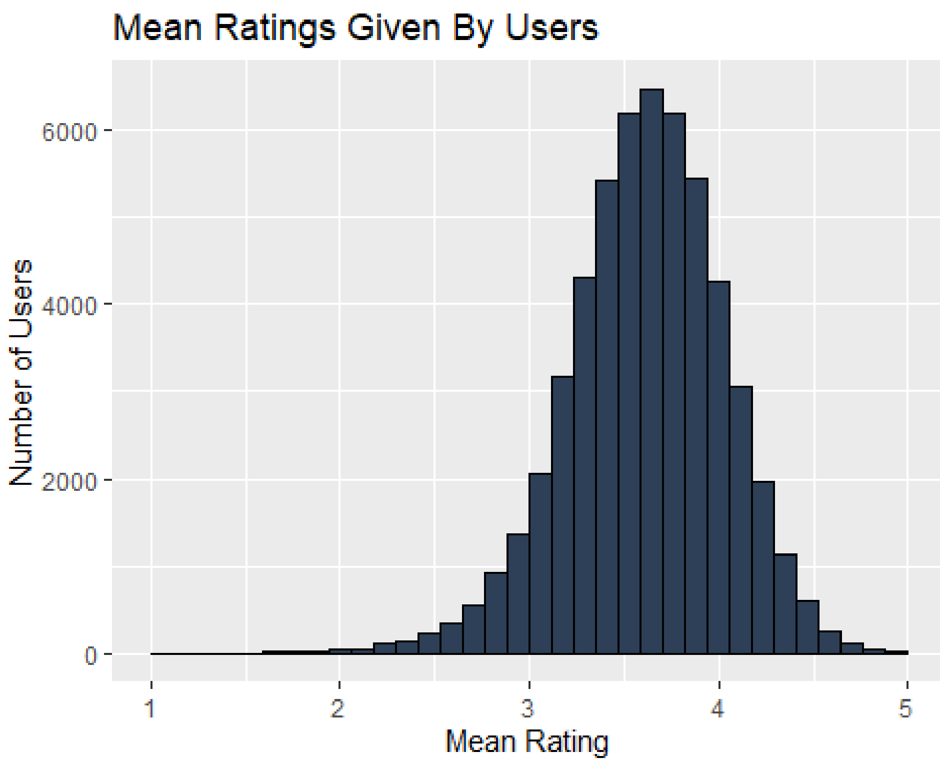
Some users rate more movies than others, as seen in the first chart. The majority of users rate between 30 and 100 films. From the second chart, we can deduce that some users rate movies higher than others and some lower than others, possibly because some people are more positive and others negative. Another reason for that is that some users tend to rate movies they liked, thus rating on average higher, and some tend to rate movies they don't like, thus rating on average lower. We will later introduce a user penalty term since some users don't rate a lot of movies.

```
# Number of ratings given by users
edx %>%
  count(userId) %>%
  ggplot(aes(n)) +
  geom_histogram(bins = 35, color = "black", fill = "#2e4057") +
  xlab("Ratings Count") +
  scale_x_log10() +
  ylab("User Count") +
  ggtitle("Number of Ratings given by Users")
```




```
# Mean ratings given by user
edx %>%
  group_by(userId) %>%
  filter(n() >= 30) %>%
  summarize(mean_rating = mean(rating)) %>%
  ggplot(aes(mean_rating)) +
  geom_histogram(bins = 35,  color = "black", fill = "#2e4057") +
  xlab("Mean Rating") +
  ylab("Number of Users") +
  ggtitle("Mean Ratings Given By Users")

## `summarise()` ungrouping output (override with `.groups` argument)
```



Modeling

The Average Movie Rating Naive Model

This model guesses the mean rating over all movies for every prediction. We start by computing the mean μ rating which is expected to be between 3 and 4. Than we predict μ for every movie. This model predicts the same rating for all movide with all differences explaind by some random variation.

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

With

$$\epsilon_{u,i}$$

some small error.

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

## [1] 3.5125

# Testing results
mu_rmse <- RMSE(validation$rating, mu)
mu_rmse
```

```
## [1] 1.0612
```

We now initialize the results table to store all our RMSEs and insert the first RMSE value.

```
# Initializing a RMSE table to save the results and saving first model's data
rmse_results <- tibble(method = "Average movie rating model", RMSE = mu_rmse)
rmse_results %>% knitr::kable()
```

method	RMSE
Average movie rating model	1.0612

This model will be used as the base model to improve. We will use insights from the data analysis section.

The Movie Effect Model

As noted earlier, some movies are generaly rated higher than others. These higher rated movies are mostly linked to more popular movies. We now introduce a movie penalty term

$$b_i$$

, which is the estimated deviation of each movies' mean rating from the total mean rating of all movies. We add

$$b_i$$

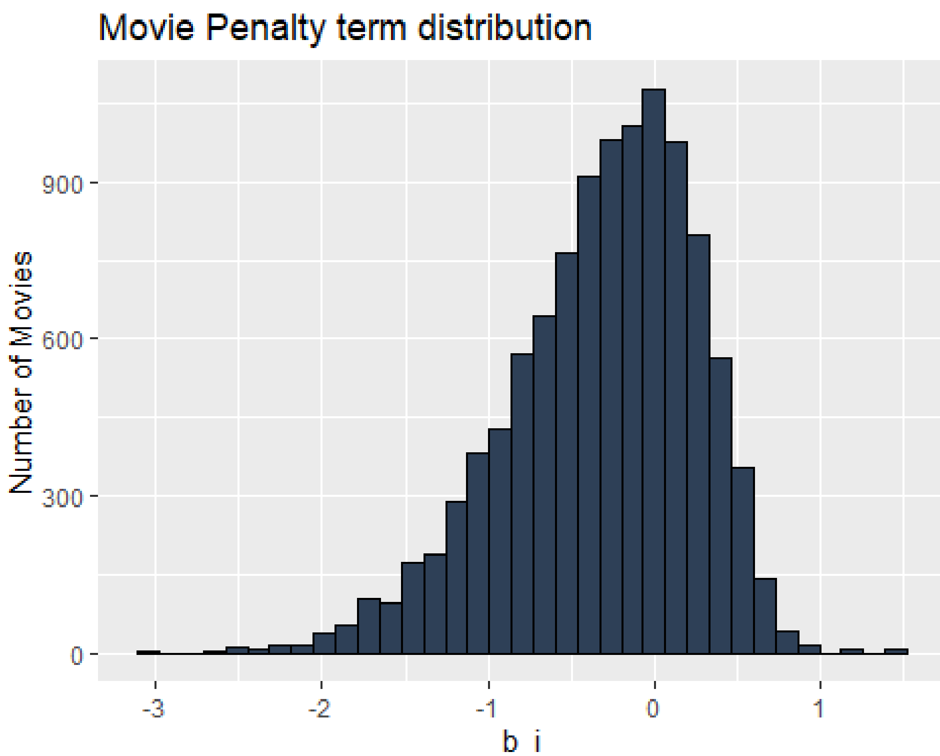
to the previous model. :

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

```
# Movie effect penalty term b_i
# Subtract the mean from the rating
# Plot the penalty term distribution
movie_penalties <- edx %>%
  group_by(movieId) %>%
  summarize(b_i = mean(rating - mu))

## `summarise()` ungrouping output (override with `.groups` argument)

movie_penalties %>%
  ggplot(aes(b_i)) +
  geom_histogram(bins = 35, color = "black", fill = "#2e4057") +
  ylab("Number of Movies") +
  ggtitle("Movie Penalty term distribution")
```



The histogram is left skewed, implying more movies have negative effects.

```
movie_effect_predictions <- validation %>%
  left_join(movie_penalties, by = "movieId") %>%
  mutate(prediction = mu + b_i)
movie_effect_rmse <- RMSE(validation$rating,
movie_effect_predictions$prediction)
movie_effect_rmse

## [1] 0.94391
```

Considering the movie penalty term improves our RMSE result.

```
# Saving reslts to table
rmse_results <- rmse_results %>%
  add_row(method = "Movie Effect Model", RMSE = movie_effect_rmse)
rmse_results %>% knitr::kable()
```

method	RMSE
Average movie rating model	1.06120
Movie Effect Model	0.94391

There is an improvement of about 11% in the RMSE using the movie effect model compared to the naive model.

The Movie and User Effect model

Some users are positive. Therefore they tend to rate movies higher or rate only movies they like, which causes them to rate movies high all the time. Some users are pessimistic. Therefore they tend to evaluate films low or rate only movies they don't like, which causes them to rate movies low all the time. We now introduce a user penalty term

$$b_u$$

, which is the estimated deviation of each users' mean rating from the total mean rating of all users. We add

$$b_u$$

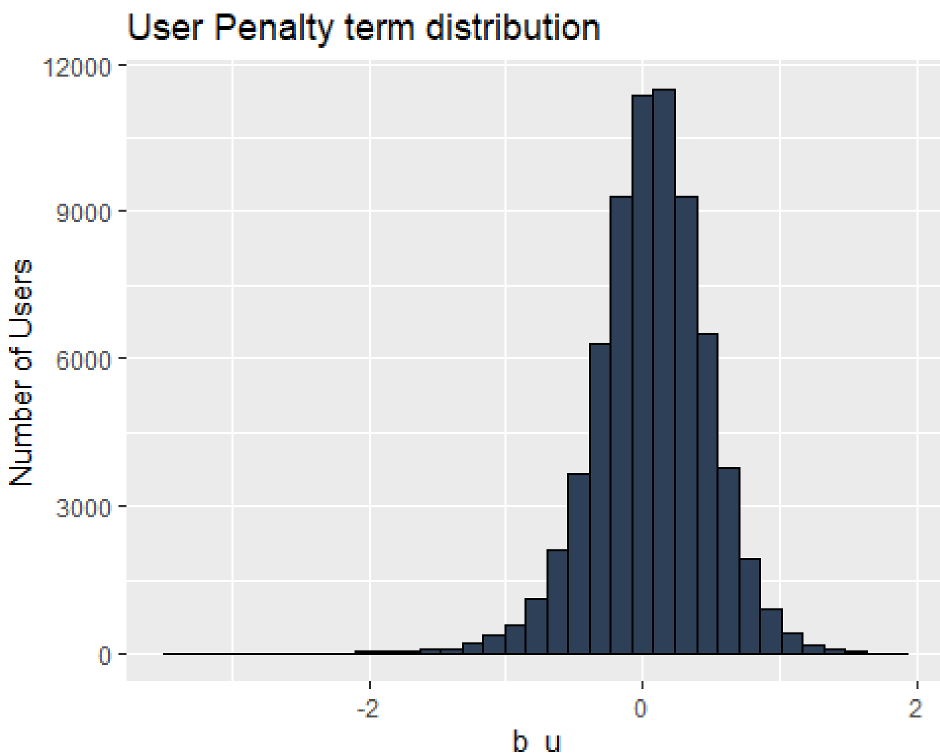
to the previous model:

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

```
# User effect penalty term b_u
# Subtract the movie penalty term and mean from the rating
# Plot the penalty term distribution
user_penalties <- edx %>%
  left_join(movie_penalties, by = "movieId") %>%
  group_by(userId) %>%
  summarize(b_u = mean(rating - mu - b_i))

## `summarise()` ungrouping output (override with `.groups` argument)

user_penalties %>%
  ggplot(aes(b_u)) +
  geom_histogram(bins = 35, color = "black", fill = "#2e4057") +
  ylab("Number of Users") +
  ggtitle("User Penalty term distribution")
```



We can see the histogram implies the user penalty term distributes normally.

```
# A model using the movie effect penalty term  b_i and the user effect
penalty term b_u
user_effect_predictions <- validation %>%
  left_join(movie_penalties, by = "movieId") %>%
  left_join(user_penalties, by = "userId") %>%
  mutate(prediction = mu + b_i + b_u)
user_effect_rmse <- RMSE(validation$rating,
user_effect_predictions$prediction)
user_effect_rmse

## [1] 0.86535
```

Considering the user effect improves our model.

```
# Saving results to table
rmse_results <- rmse_results %>%
  add_row(method = "Movie & User Effect Model", RMSE = user_effect_rmse)
rmse_results %>% knitr::kable()
```

method	RMSE
Average movie rating model	1.06120
Movie Effect Model	0.94391
Movie & User Effect Model	0.86535

There is an improvement of about 8% in the RSME using the user effect model compared to the movie effect model.

Regularized Movie & User Effect Model

A movie that was rated only once with a five-star rating has an average of five stars. Therefore it will be one of the best movies. It is most likely an obscure movie since it was rated only once. When very few users rate a movie, the uncertainty level rises. To handle these situations, we introduce the concept of regularization to our model. It allows the model to penalize extreme predictions, low or high, that derive from small sample size and reduces the possibility of overfitting. Lambda is a tuning parameter of regularization; it shrinks

$$b_i$$

and

$$b_u$$

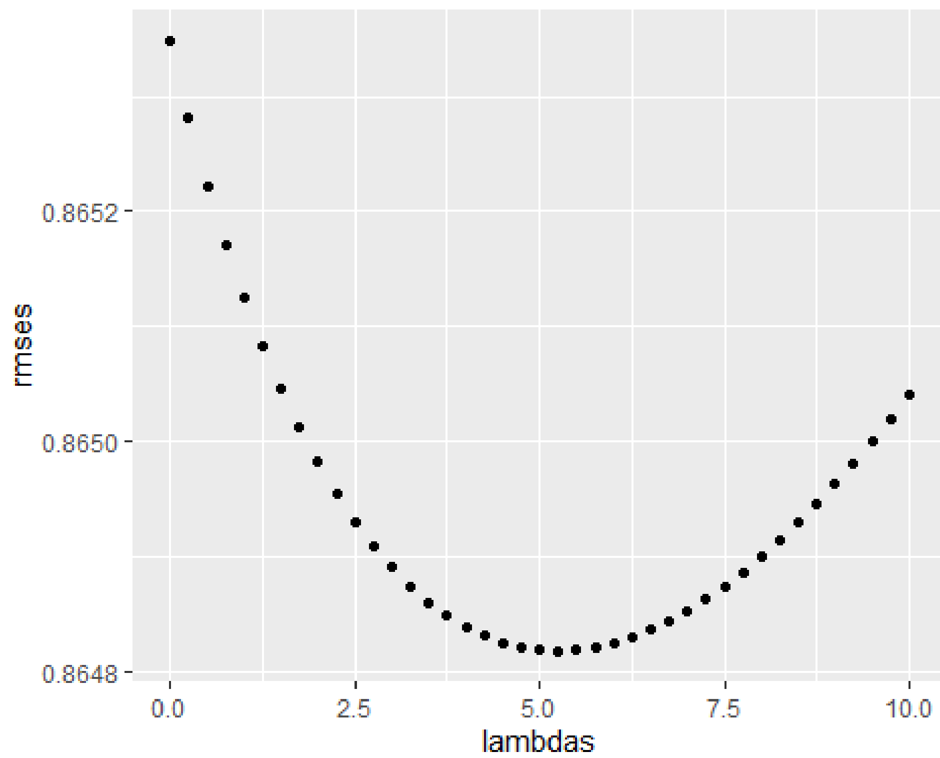
in cases of small sample size. We want to find the correct value of lambda, the one that will minimize the RMSE.

```
# Using cross-validation to tune the tuning parameter lambda
lambdas <- seq(0, 10, 0.25)

# For each lambda, find movie penalty term b_i and
# user effect penalty term b_u, followed by rating prediction & testing
rmses <- sapply(lambdas, function(lambda){
  mu <- mean(edx$rating)
  b_i <- edx %>%
    group_by(movieId) %>%
    summarize(b_i = sum(rating - mu) / (n() + lambda))
  b_u <- edx %>%
    left_join(b_i, by = "movieId") %>%
    group_by(userId) %>%
    summarize(b_u = sum(rating - b_i - mu) / (n() + lambda))
  predictions <- validation %>%
    left_join(b_i, by = "movieId") %>%
    left_join(b_u, by = "userId") %>%
    mutate(prediction = mu + b_i + b_u) %>%
    .$prediction

  RMSE(validation$rating, predictions)
})

# Plot the lambdas against the RMSEs to visualize what is the best lambda
tibble(lambda = lambdas, rmse = rmses) %>%
  ggplot(aes(lambdas, rmses)) +
  geom_point()
```



According to the chart the lambda that will acheive our goal is:

```
lambda <- lambdas[which.min(rmses)]
lambda

## [1] 5.25
```

The corresponding RMSE:

```
min(rmses)

## [1] 0.86482

# Saving results to table
rmse_results <- rmse_results %>%
  add_row(method = "Regularized Movie & User Effect Model", RMSE =
min(rmses))
rmse_results %>% knitr::kable()
```

method	RMSE
Average movie rating model	1.06120
Movie Effect Model	0.94391
Movie & User Effect Model	0.86535
Regularized Movie & User Effect Model	0.86482

There is very little improvement compared to the user effect model.

Results

method	RMSE
Average movie rating model	1.06120
Movie Effect Model	0.94391
Movie & User Effect Model	0.86535
Regularized Movie & User Effect Model	0.86482

The best result was using the regularized movie & user effect model with an RMSE of 0.86482.

Discussion

The final model is the one that uses regularized movie and user effects, using the following model:

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

This model handles very well the different biases derived from famous movies with big marketing crews behind them or those that are not very well known and biases derived from different kinds of users, the optimistic and pessimistic. It also handles very well the fact that some movies are rated only once. Therefore they can be at the top or bottom with high levels of uncertainty and the fact that some users rated only one movie also making their mean rating hard to predict with a high degree of uncertainty.

Conclusion

We have built a machine learning algorithm to recommend streaming service's users new movies. When first introducing the movie bias and then the users' bias, we had significant improvements both of the times. The introduction of the movie's bias improved our predictions by 11% RMSE = 0.94391, and the introduction of the users' bias further improved our predictions by another 8% RMSE = 0.86535. The regularization method, although it showed an improvement, had minimal impact on the final RMSE but still was the best performing model with an RMSE of 0.86482.

Appendix

Environment

Operating System:

```
##                _
## platform       x86_64-w64-mingw32
## arch           x86_64
## os             mingw32
## system         x86_64, mingw32
## status
## major          3
## minor          6.2
## year           2019
## month          12
## day            12
## svn rev        77560
## language       R
## version.string R version 3.6.2 (2019-12-12)
## nickname       Dark and Stormy Night
```