HarvardX Capstone Project PH125.9: MovieLens

Iheanyichukwu Achareke

2/25/2020

Contents

1 Overview

This project is related to the capstone course of the HarvardX data science professional certificate PH125.

The objective of this project is to build a movie recommendation system based on the movielens 10M dataset with an approach that is motivated by the Netflix challenge. This report documents data acquisition, wrangling, exploratory analysis and model developement.

1.1 Introduction

Predicting user choice has been one of the leading uses of machine learning in the online retailing sector. These predictions are used to make recommendations to customers for what to buy or movies to watch or songs to listen to. In 2006, Netflix offered a challenge to the data science community. The challenge was to improve thier in-house recommendation system by 10% and win a $1M prize.

The aim of this project is to develop machine learning algorithm that predicts the ratings a user will give a movie. This can be turned into a recommendation system by recommending the movies with the highest predicted ratings.

The algorithms developed in this paper are based on the techniques used in the winning algorithm of the netflix challenge.

1.2 Problem Definition

This project is based on the MovieLens 10M dataset. This dataset contains 10 million ratings of 10,677 movies by 69,878 people. Clearly, the dataset is sparse as it take about 746 million records if all users rated all movies. The dataset also lacks any information that could provide context about the users which could be important predictors in building an algorithm. eg race, age, sex, location, nationality, language etc.

We will develop an alogorithm by exploring the different biases that can be indentified in an exploratory analysis of the data and the accuracy of the model will be determined using RMSE. The goal is to develop and algorithm with an RMSE < 0.86490.

1.3 Data and Environment Preparation

Our project enviroment is Microsoft R open R 3.5.3 with the following packages loaded from cran:

- tidyverse
- caret
- data.frame
- lubridate

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

This project is based on the movielens 10M dataset. The movielens 10M dataset is a subset of the movielens dataset with 10 million movie ratings and is available at https://grouplens.org/datasets/movielens/10m/

```
# 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")
movies <- as.data.frame(movies) %>% mutate(movieId = as.numeric(levels(movieId))[movieId],
                                           title = as.character(title),
                                           genres = as.character(genres))

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

In order to test our prediction model without overfitting, the movielens dataset will be split into a training set *edx* and a test set *validation*.

```
# Validation set will be 10% of MovieLens data
set.seed(1)
# 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)
```

To ensure the developement of the model is completely independent of the validation set, the edx set will be further split into training set *edx_train* and test set *edx_test*. The model will be developed based on the edx_train set, tested on the edx_test and the final RMSE will be calculated on the validation set.

```
set.seed(1)

#Test set will be 10% of the edx dataset
edx_test_index <- createDataPartition(y = edx$rating, times = 1, p = 0.1, list = FALSE)
edx_train <- edx[-edx_test_index,]
temp <- edx[edx_test_index,]

# Make sure userId and movieId in edx_test set are also in edx_train set
edx_test <- temp %>%
  semi_join(edx_train, by = "movieId") %>%
  semi_join(edx_train, by = "userId")

# Add rows removed from edx_test set back into edx_train
removed <- anti_join(temp, edx_test)
edx_train <- rbind(edx_train, removed)

rm(removed, temp, edx_test_index)
```

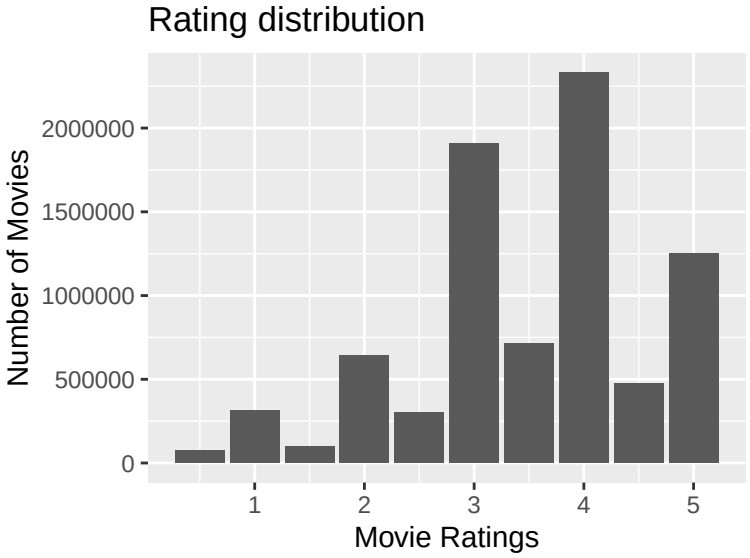
2 Methods and Analysis

2.1 Data Exploration

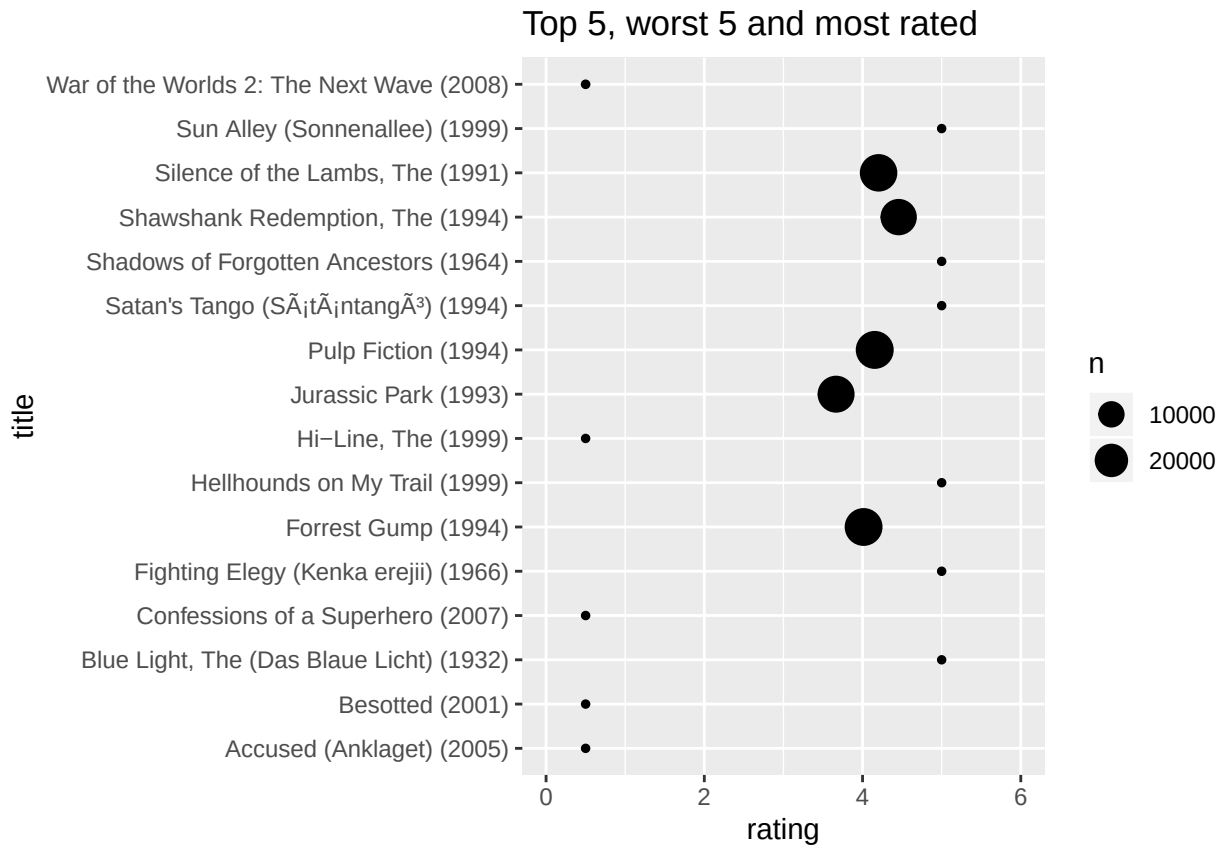
To gain a preliminary understanding of the structure of the dataset, we display the first few rows of the edx_train dataset

```
##   userId movieId rating timestamp                          title
## 1      1     122      5 838985046               Boomerang (1992)
## 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)
## 8      1     356      5 838983653           Forrest Gump (1994)
##                         genres
## 1                Comedy|Romance
## 4  Action|Drama|Sci-Fi|Thriller
## 5       Action|Adventure|Sci-Fi
## 6 Action|Adventure|Drama|Sci-Fi
## 7       Children|Comedy|Fantasy
## 8      Comedy|Drama|Romance|War
```

From a review of the distibution of user ratings, it can be seen the users have a tendency to rate movies higher rather than lower. The most common rating is 4 and the leas common is 0.5. Half ratings are generally less common than full ratings

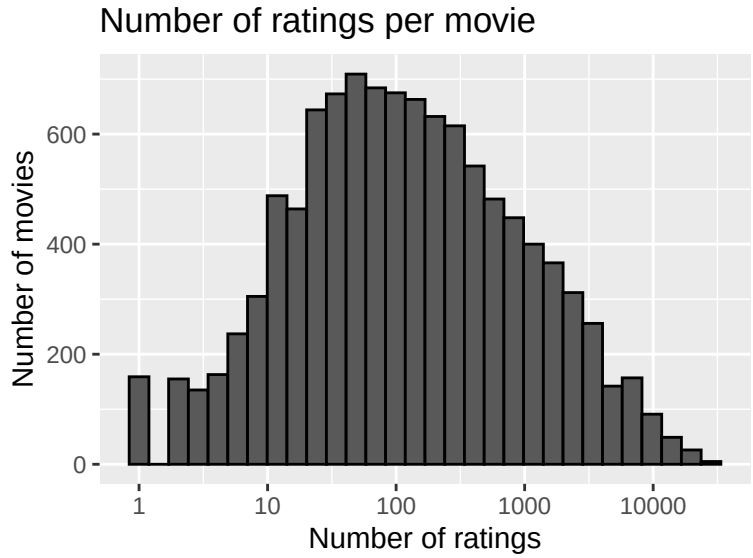


Ratings are also seen to vary from movie to movie which suggests that there is a movie bias in the ratings.



The movies that appear in the top 5 and worst 5 by user rating are mostly obscure.

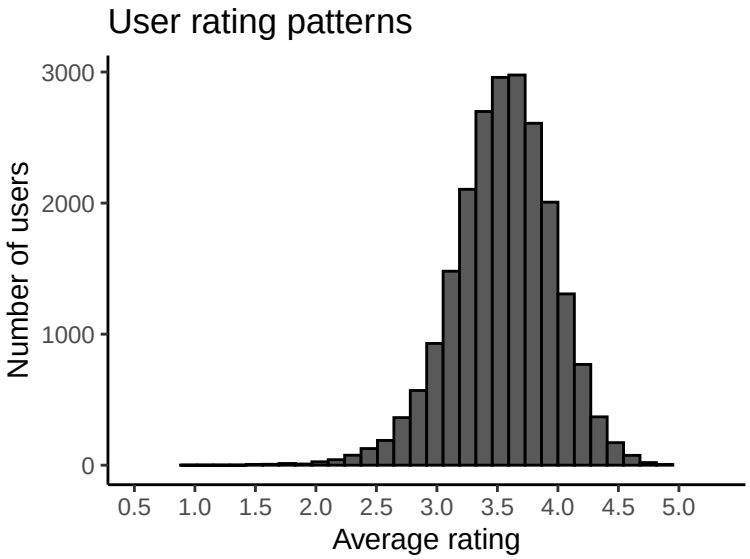
The data suggests that alot of these extreme values are movies which were rated by a few users and hence cannot be relied on. this posses a problem for modelling as these noisy values can distort the model. To better understand the severity of this problem, we visualize the distribution the number of user ratings per movie



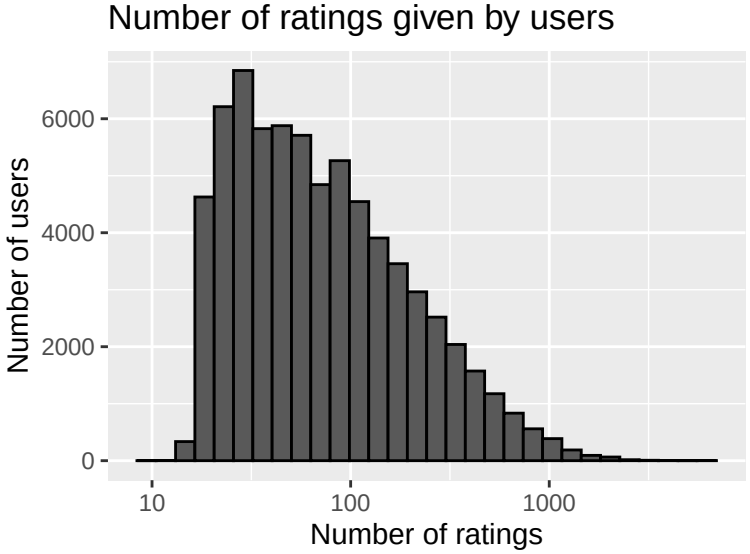
We can observe that some movies have been rated much more often that others. some movies have been

rated only a few time and in some cases only once. Regularization and a penalty term will have to be applied to our model to mitigate the effect of these noisy estimates.

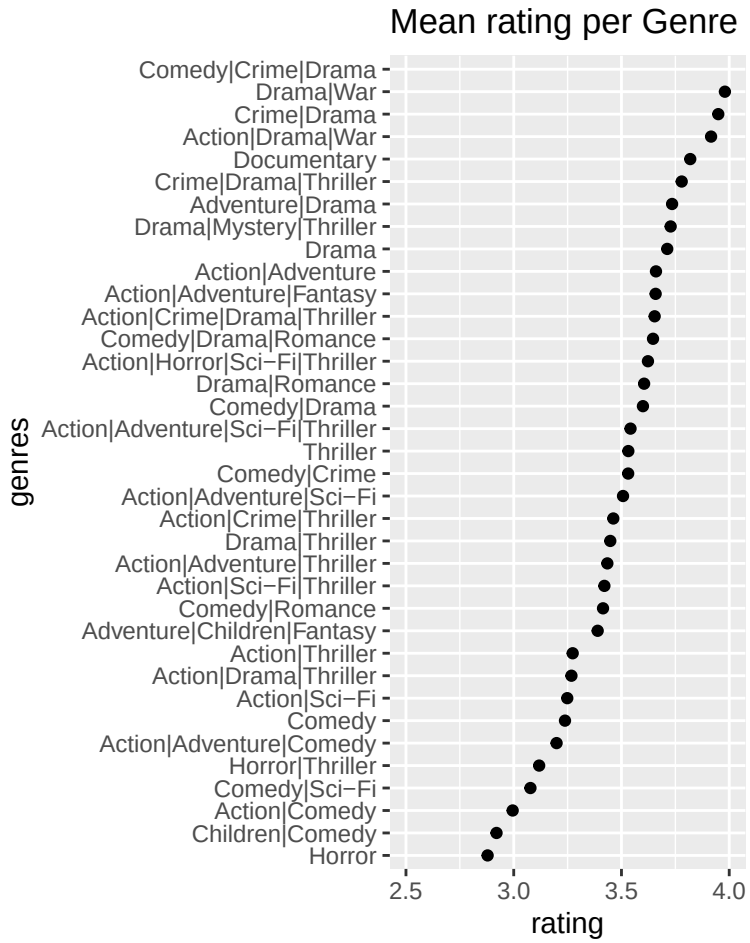
We further probe the data to see if there is a pattern in the rating of the movies based on individual differences amongst users. Unfortunately, the data does not contain any background data on users such as ethnicity, location, education, income bracket etc.



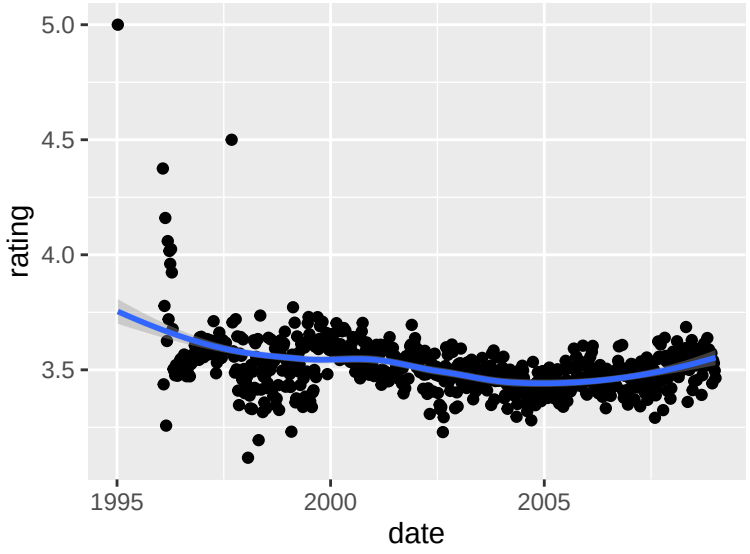
We observe that the average rating varies per user with some users being very critical of movies in general and some users other users enjoying movies accross board.



The number of ratings given by eact user varies widely from 20 to over 1000. Regularization and a penalty term will be applied to this bias.



There is a clear variation in the mean rating across genres. there is a bias with people rating comedies much higer than horror movies.



There appears to be some effect of time on ratings with ratings declining from 1995 to about 2005 and then rises again

2.2 Modelling Approach

Our model shall be developed by iteratively incorporating the biases already identified during the exploratory analysis of the data. The accuracy of the model will be evaluated using Root Mean Square Error (RMSE). The RMSE is the standard deviation of the residuals(prediction errors). RMSE is defines as follows:

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

Where: N = *Number of user/movie combinations*

The interpretation of the RMSE is similar to standard deviation. a lower result is better. a result higher than 1 means our typical erroris larger than one star which is not good. The r function to compute RMSE is

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

2.2.1 Average movie rating model

To minimise RMSE, we start with a simple model that predicts the same rating for all movies. By its definition, the value that minimises the RMSE for a set of numbers is its mean. This model assumes the same rating for all movies and all differences are explained by random variation:

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

with $\epsilon_{u,i}$ the independent error and μ the true rating of the movies. The average is given by

```
mu <- mean(edx_train$rating)
mu
```

```
## [1] 3.512456
```

Using this model, we predict all ratings in the edx_train set as μ and calculate the RMSE on the edx_test set.

```
avg_rmse = RMSE(edx_test$rating, mu)
avg_rmse
```

```
## [1] 1.060054
```

To keep a log of our RMSE results as we go along, we create a table to store our RMSE values.

```
rmse_results <- data_frame(method = "Simple Average model", RMSE = avg_rmse)
rmse_results %>% knitr::kable()
```

method	RMSE
Simple Average model	1.060054

An RMSE greater than 1 implies our prediction is off by 1 star on the average which isnt very good. to improve this, we incorporate the biases that we discovered during the data exploration and analysis

2.2.2 Movie effect model

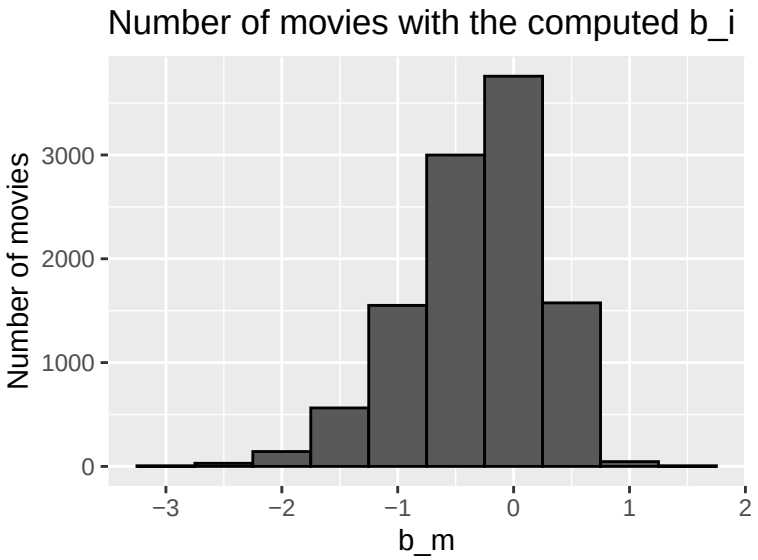
From real world experiences which is coroborated by our exploratory analysis of the data, the ratings differ from movie to movie. Higer ratings are usually linked to popular movies, block busters and critically acclaimed movies. To incorporate this into out model, we compute the estimated deviation of each movies mean from

the overall mean μ. This deviation per movie "m" is called a bias and is added to the model as b_m, that represents average ranking for movie i:

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

A histogram of the bias b_i is shown below:

```
movie_avgs <- edx_train %>%
  group_by(movieId) %>%
  summarize(b_m = mean(rating - mu))
movie_avgs %>% qplot(b_m, geom ="histogram", bins = 10, data = ., color = I("black"),
ylab = "Number of movies", main = "Number of movies with the computed b_i")
```



The plot is skewed to the left, this bias will adjust the ratings of alot of movies to a lower value. We predict using this model on the test set and calculate the RMSE:

```
predicted_ratings <- mu +  edx_test %>%
  left_join(movie_avgs, by='movieId') %>%
  pull(b_m)
model_1_rmse <- RMSE(predicted_ratings, edx_test$rating)
rmse_results <- bind_rows(rmse_results,
                          data_frame(method="Movie effect model",
                                     RMSE = model_1_rmse ))
rmse_results %>% knitr::kable()
```

method	RMSE
Simple Average model	1.0600537
Movie effect model	0.9429615

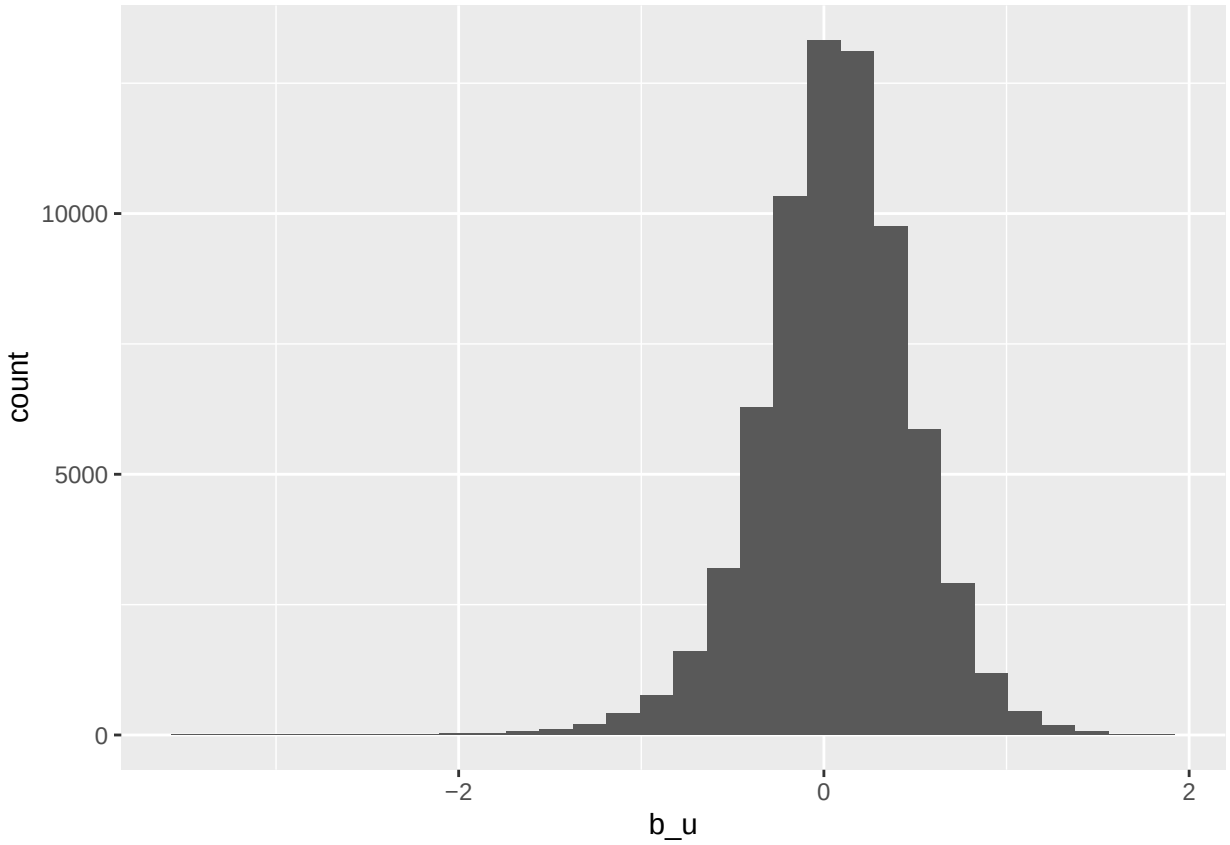
The RMSE is seen to improve with this prediction where we have added the bias b_m per movie to the mean μ.

To further improve the accuract of this model, we incorporat the effect of the difference in user tastes and temprament.

2.2.3 User and movie effect model

Some users love all movies on the average whereas some users are very critical of movies and will only give a good rating to the best of the best. We can incorpoate this varaibility per user by determning the average rating a user has given.

```
user_avg <- edx_train %>% left_join(movie_avgs, by='movieId') %>% group_by(userId) %>%
  summarize(b_u = mean(rating - mu - b_m))

user_avg %>% ggplot(aes(b_u)) + geom_histogram()
```



There is a marked variability from user to user. The data supports the theory that some users on the average rate higher than others. we can incorporate this bias into the model as follows:

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

where b_u is a user-specific effect. Critical users will have a negative b_u while "happy" users will have positive b_u.

We calculate b_u as follow:

```
user_avg <- edx_train %>% left_join(movie_avgs, by='movieId') %>% group_by(userId) %>%
  summarize(b_u = mean(rating - mu - b_m))
```

Using this new model, we make new predictions on the edx_test dataset and recalculate the RMSE:

```
predict_b_u <- edx_test %>% left_join(movie_avgs, by='movieId') %>%
  left_join(user_avg, by='userId') %>%
  mutate(pred = mu + b_m + b_u) %>% .$pred
```

```
rmse_b_u <- RMSE(predict_b_u, edx_test$rating)

rmse_results <- bind_rows(rmse_results, data_frame(method = "User Effect", RMSE = rmse_b_u))
rmse_results %>% knitr::kable()
```

method	RMSE
Simple Average model	1.0600537
Movie effect model	0.9429615
User Effect	0.8646843

2.2.4 Regularized Movie and user effect model

The average rating per movie is affected by movies that have received an extreme rating (very high or very low) by one or a few users. Such ratings cannot be relied upon and unfortunately, RMSE is sensitive to such extreme values. The same principle applies to users who have rated a few movies. To compensate for these effects, we introduce a penalty term Lambda to our model as follows

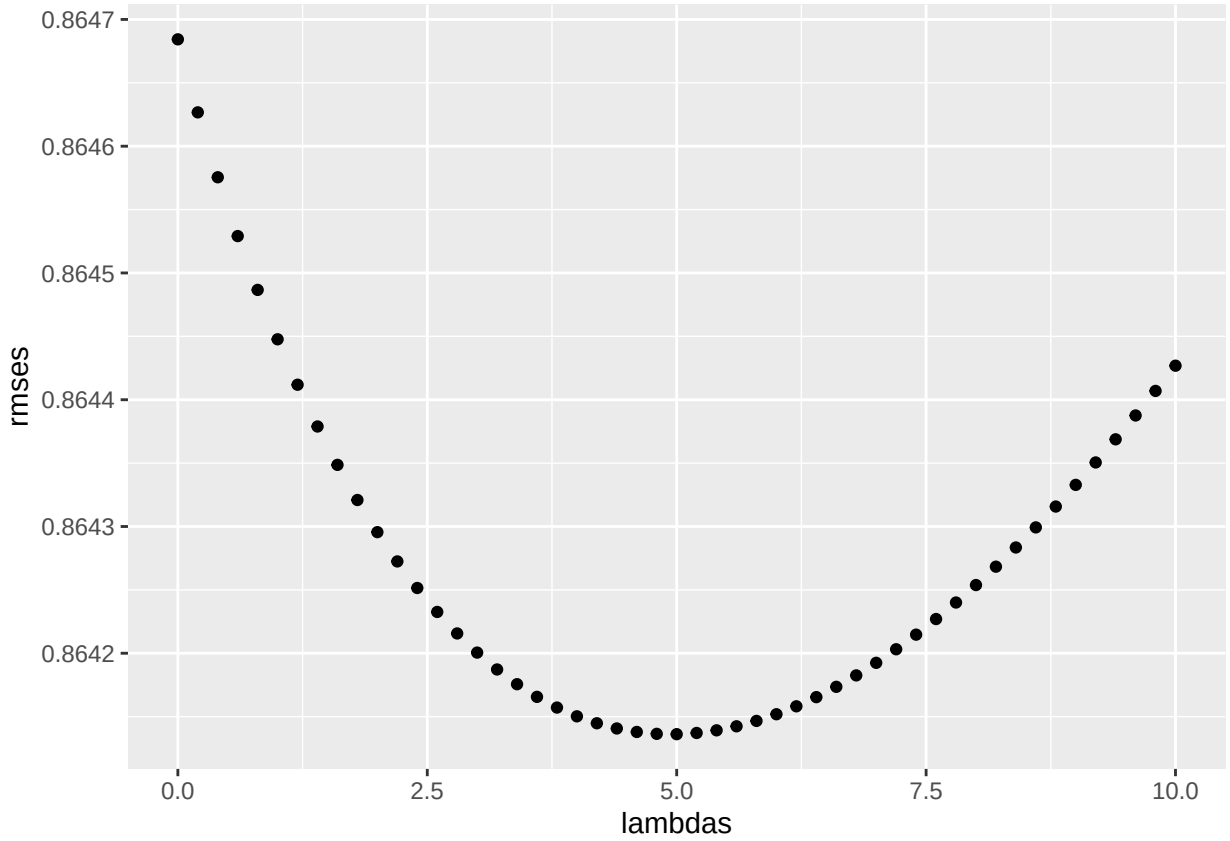
$$Y_{u,i} = \mu + \frac{1}{\lambda + n_m} b_m + \frac{1}{\lambda + n_u} b_u + \epsilon_{u,i}$$

We shall determine the value of λ the minimises the RMSE:

```
lambdas <- seq(0,10,0.2)

rmses <- sapply(lambdas, function(l){

  mu <- mean(edx_train$rating)

  b_i <- edx_train %>%
    group_by(movieId) %>%
    summarize(b_i = sum(rating - mu)/(n()+l))

  b_u <- edx_train %>%
    left_join(b_i, by="movieId") %>%
    group_by(userId) %>%
    summarize(b_u = sum(rating - b_i - mu)/(n()+l))

  predicted_ratings <-
    edx_test %>%
    left_join(b_i, by = "movieId") %>%
    left_join(b_u, by = "userId") %>%
    mutate(pred = mu + b_i + b_u) %>%
    pull(pred)

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

The results of the different values of lambda tested are shown below

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



And the resulting RMSE from $\lambda =$ `lambda` is `min(rmses)`

```
rmse_results <- bind_rows(rmse_results,
                          data_frame(method="Regularized movie and user effect model",
                                     RMSE = min(rmses)))
rmse_results %>% knitr::kable()
```

method	RMSE
Simple Average model	1.0600537
Movie effect model	0.9429615
User Effect	0.8646843
Regularized movie and user effect model	0.8641362

2.2.5 Regularized Movie and user bias plus Genre bias

Form our exploratory analysis, it was observed that some of the variability in movie ratings were explained by the genres. We incorporate this bias into our model as follows:

$$Y_{u,i} = \mu + \frac{1}{\lambda + n_m} b_m + \frac{1}{\lambda + n_u} b_u + b_g + \epsilon_{u,i}$$

the bias b_g is estimated as follows

```
l <- lambda

  b_i <- edx_train %>%
```

```
    group_by(movieId) %>%
    summarize(b_i = sum(rating - mu)/(n()+l))

  b_u <- edx_train %>%
    left_join(b_i, by="movieId") %>%
    group_by(userId) %>%
    summarize(b_u = sum(rating - b_i - mu)/(n()+l))

  b_g <- edx_train %>%
    left_join(b_i, by="movieId") %>%
    left_join(b_u, by="userId") %>%
    group_by(genres) %>%
    summarize(b_g = mean(rating - b_i - b_u - mu))
```

We can now predict using this new model and calulate the RMSE as folllows

```
 predicted_ratings <-
    edx_test %>%
    left_join(b_i, by = "movieId") %>%
    left_join(b_u, by = "userId") %>%
    left_join(b_g, by = "genres") %>%
    mutate(pred = mu + b_i + b_u + b_g) %>%
    pull(pred)


rmse_b_g <- RMSE(predicted_ratings, edx_test$rating)
rmse_results<- bind_rows(rmse_results,
                         data_frame(method="Regularized movie, user and Genre effect model",
                                    RMSE = rmse_b_g))
rmse_results %>% knitr::kable()
```

method	RMSE
Simple Average model	1.0600537
Movie effect model	0.9429615
User Effect	0.8646843
Regularized movie and user effect model	0.8641362
Regularized movie, user and Genre effect model	0.8638134

3 Results

Our data was split into 3. A training set, a test set and a validation set. to obtain the final result, we shall use our model to predict and calculate RMSE on the validation set.

```
predicted_ratings <-
  validation %>%
  left_join(b_i, by = "movieId") %>%
  left_join(b_u, by = "userId") %>%
  left_join(b_g, by = "genres") %>%
  mutate(pred = mu + b_i + b_u + b_g) %>%
  pull(pred)

rmse_v_g <- RMSE(predicted_ratings, validation$rating)
rmse_results<- bind_rows(rmse_results,
                         data_frame(method="Regularized movie, user and Genre effect model on Validatio
                                    RMSE = rmse_v_g))
rmse_results %>% knitr::kable()
```

method	RMSE
Simple Average model	1.0600537
Movie effect model	0.9429615
User Effect	0.8646843
Regularized movie and user effect model	0.8641362
Regularized movie, user and Genre effect model	0.8638134
Regularized movie, user and Genre effect model on Validation Set	0.8648541

4 Conclusion

The Regularized movie, user and Genre effect model performed best with an RMSE of 0.8648541.

Further improvements can be made to the model by enriching the dataset to include predictors such as age, sex, region, nationality and language. Additionally, exploring other machine learning models such as neural networks, item based collaborative filtering, decision trees and ensembles could also improve the model further.