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

Data Science: Capstone Project for Harvardx Professional Data Science Certificate
(Movielens Project: PH125.9x)

Delpagodage Ama Nayanahari Jayaweera

01 December 2024

Contents

1 Chapter 1

1.1 Overview

This project is related to the MovieLens Project of the HarvardX: PH125.9x Data Science: Capstone course. The present report begins by providing a general overview of the project and describing its objectives. The given dataset is then prepared and set up for analysis. An exploratory data analysis is conducted to develop a machine learning algorithm capable of predicting movie ratings. The process involves refining the model until a final version is achieved. The results are explained in detail. Finally, the report concludes with some insightful remarks.

1.2 Introduction

Recommendation systems utilize user ratings to provide personalized suggestions based on their preferences. Companies like Amazon, which offer diverse products and allow users to rate them, accumulate extensive datasets. These datasets enable the prediction of user ratings for specific items, and items with predicted high ratings are recommended to users, enhancing their overall experience.

This approach is not limited to products and extends to various domains, such as movies, as demonstrated in project.Recommendation systems represent a widely employed model in machine learning algorithms. Netflix, for instance, is recognized for its success, largely attributed to a robust recommender system. The significance of such algorithms in product recommendation systems is underscored by events like the Netflix Prize—an open competition seeking the best collaborative filtering algorithm to predict user ratings for films solely based on previous ratings, without additional information about users or films. Project focuses on the development of a movie recommendation system using the 10M version of the MovieLens dataset, curated by GroupLens Research.

1.3 Executive Summary

The objective of this project is to train a machine learning algorithm capable of predicting user ratings (ranging from 0.5 to 5 stars). The algorithm will utilize a provided subset (edx dataset supplied by the staff) to predict movie ratings in a given validation set.

Algorithm performance will be assessed using the Root Mean Square Error (RMSE), a widely used metric that measures the differences between model-predicted values and observed values. RMSE serves as an accuracy metric, with lower values indicating better performance. It is particularly sensitive to outliers, as larger errors have a more significant impact on the RMSE.

Four models will be developed and compared based on their resulting RMSE to evaluate their quality. The benchmark for this algorithm is an RMSE expected to be lower than 0.86490. The RMSE computation function, for vectors of ratings and their corresponding predictors is as follows:

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

Finally, the most effective model will be utilized for forecasting movie ratings.

1.4 Dataset

The MovieLens dataset is automatically downloaded

1. [MovieLens 10M dataset] https://grouplens.org/datasets/movielens/10m/

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

```r
##############################################################
# Create edx set, validation set, and submission file
##############################################################
# Note: this process could take a couple of minutes for loading required package: tidyverse
# and package caret
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")

data_dir <- "D:/Edex/Data_Science_Capstone_MovieLens/ml-10M100K"
ratings_file <- file.path(data_dir, "ratings.dat")
movies_file <- file.path(data_dir, "movies.dat")

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

if (!file.exists(ratings_file) || !file.exists(movies_file)) {
  stop("Failed to extract the required data files. Check file paths.")
}

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")
```

Next, proceed to import the essential libraries that shall constitute the backbone of analytical framework within this kernel.

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

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

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

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

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

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

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

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

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


library(tidyverse)
library(scales)
library(arules)
library(gridExtra)
library(purrr)
library(readr)
library(tidyr)
library(dplyr)
library(ggplot2)
```

To enhance the accuracy of predicting movie ratings for users who haven't seen the movie yet, the MovieLens dataset will be divided into two subsets: "edx," serving as the training subset to train the algorithm, and "validation," a subset for testing movie ratings.

```r
# 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") %>%
```

```r
  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)
```

The algorithm development will exclusively take place on the "edx" subset, with the "validation" subset reserved for testing the final algorithm.

2 Chapter 2

2.1 Methods and Analysis

2.1.1 Data Analysis

To acquaint with the dataset, examine the initial rows of the "edx" subset as shown below. This subset comprises six variables: UserID MovieID Rating Timestamp Title Genres

Each row corresponds to a singular rating provided by a user for a specific movie.

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

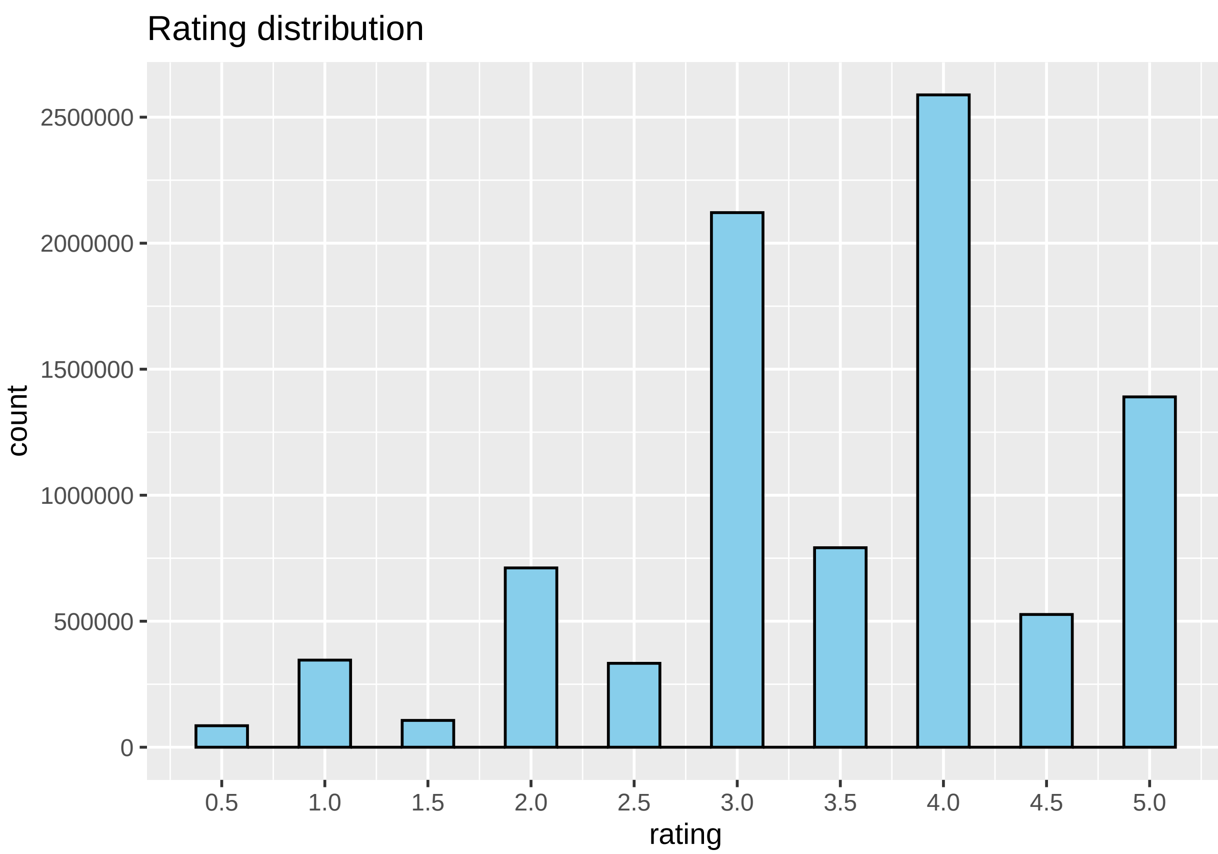
A summary of the subset confirms that there are no missing values.

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

The total of unique movies and users in the edx subset is about 70.000 unique users and about 10.700 different movies:

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

Users have a preference to rate movies rather higher than lower as shown by the distribution of ratings below. 4 is the most common rating, followed by 3 and 5.0.5 is the least common rating. In general, half rating are less common than whole star ratings.

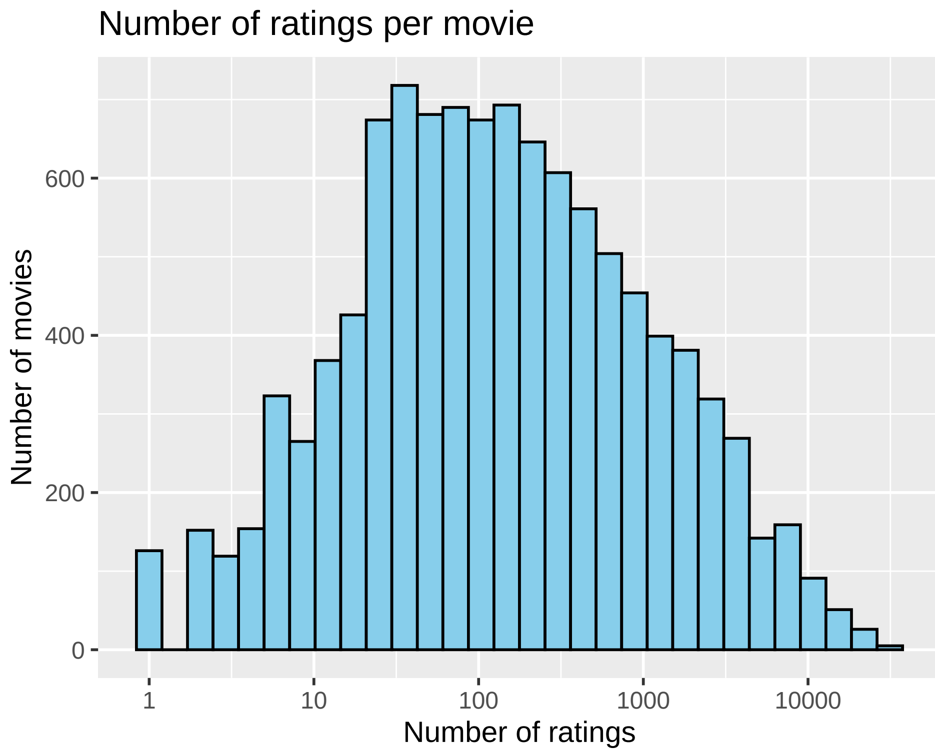


It is observe that some movies have been rated much often that other, while some have very few ratings and sometimes only one rating.This will be important for model as very low rating numbers might results in untrustworthy estimate for predictions.In fact 125 movies have been rated only once.

Thus regularisation and a penalty term will be applied to the models in this project. Regularizations are techniques used to reduce the error by fitting a function appropriately on the given training set and avoid overfitting (the production of an analysis that corresponds too closely or exactly to a particular set of data, and may therefore fail to fit additional data or predict future observations reliably).

Regularization is a technique used for tuning the function by adding an additional penalty term in the error function. The additional term controls the excessively fluctuating function such that the coefficients don't take extreme values.

```
edx %>%
count(movieId) %>%
ggplot(aes(n)) +
geom_histogram(bins = 30, color = "black", fill = "skyblue") +
scale_x_log10() +
xlab("Number of ratings") +
  ylab("Number of movies") +
ggtitle("Number of ratings per movie")
```



As 20 movies that were rated only once appear to be obscure, predictions of future ratings for them will be difficult.

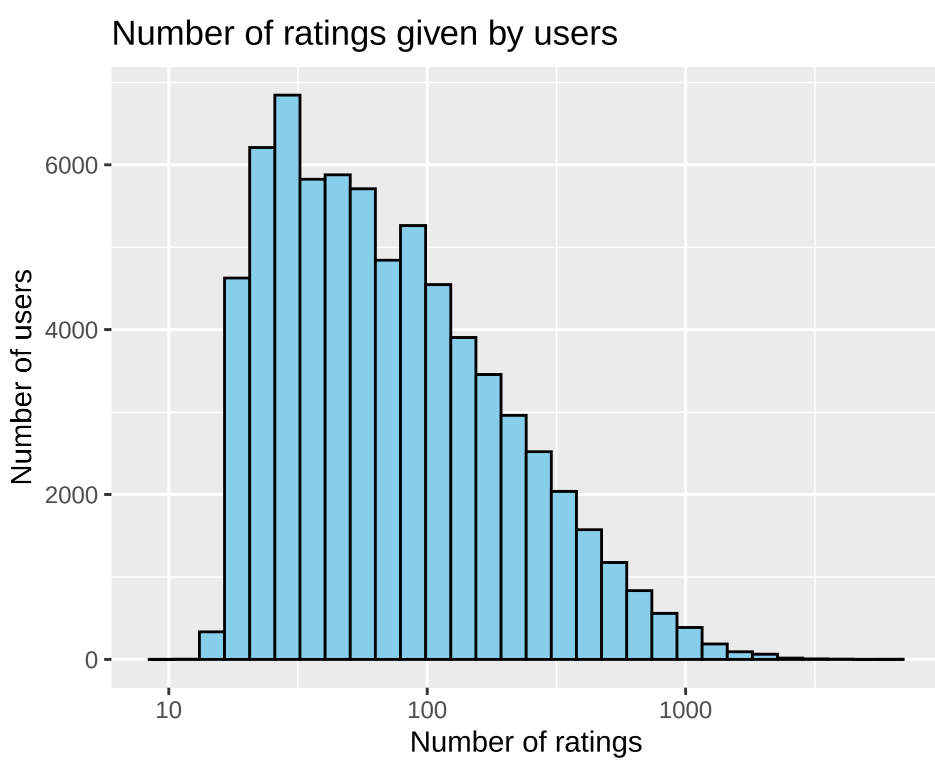
```
library(dplyr)
library(knitr)

edx %>%
  group_by(movieId) %>%
  summarize(count = n(), .groups = 'drop') %>%
  filter(count == 1) %>%
  left_join(edx, by = "movieId") %>%
  group_by(title) %>%
  reframe(rating = rating, n_rating = count) %>%
  slice(1:20) %>%
  kable()
```

title	rating	n_rating
1, 2, 3, Sun (Un, deuz, trois, soleil) (1993)	2.0	1
100 Feet (2008)	2.0	1
4 (2005)	2.5	1
Accused (Anklaget) (2005)	0.5	1
Ace of Hearts (2008)	2.0	1
Ace of Hearts, The (1921)	3.5	1
Adios, Sabata (Indio Black, sai che ti dico: Sei un gran figlio di…) (1971)	1.5	1
Africa addio (1966)	3.0	1
Aleksandra (2007)	3.0	1
Bad Blood (Mauvais sang) (1986)	4.5	1
Battle of Russia, The (Why We Fight, 5) (1943)	3.5	1

title	rating	n_rating
Bellissima (1951)	4.0	1
Big Fella (1937)	3.0	1
Black Tights (1-2-3-4 ou Les Collants noirs) (1960)	3.0	1
Blind Shaft (Mang jing) (2003)	2.5	1
Blue Light, The (Das Blaue Licht) (1932)	5.0	1
Borderline (1950)	3.0	1
Brothers of the Head (2005)	2.5	1
Chapayev (1934)	1.5	1
Cold Sweat (De la part des copains) (1970)	2.5	1

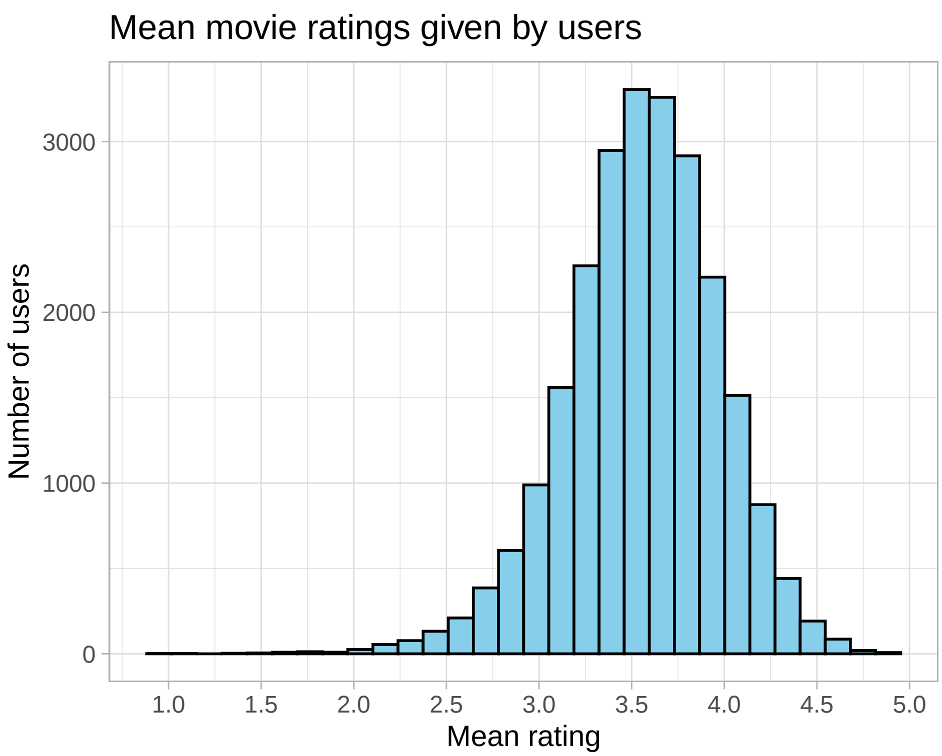
It is observe that the majority of users have rated between 30 and 100 movies. So, a user penalty term need to be included later in models.

```
edx %>%
count(userId) %>%
ggplot(aes(n)) +
geom_histogram(bins = 30, color = "black", fill = "skyblue") +
scale_x_log10() +
xlab("Number of ratings") +
ylab("Number of users") +
ggtitle("Number of ratings given by users")
```



Furthermore, users differ vastly in how critical they are with their ratings. Some users tend to give much lower star ratings and some users tend to give higher star ratings than average. The visualization below includes only users that have rated at least 100 movies.

```r
library(ggplot2)
library(dplyr)

edx %>%
  group_by(userId) %>%
  filter(n() >= 100) %>%
  summarize(b_u = mean(rating)) %>%
  ggplot(aes(b_u)) +
  geom_histogram(bins = 30, color = "black", fill = "skyblue") +
  xlab("Mean rating") +
  ylab("Number of users") +
  ggtitle("Mean movie ratings given by users") +
  scale_x_continuous(breaks = seq(0.5, 5, 0.5)) +
  theme_light()
```



2.2 Modelling Approach

Previously anticipated, that compute the RMSE, defined as follows:

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

Letting NN represent the count of user/movie combinations and summing across all such combinations, the RMSE serves as metric for assessing model accuracy. The RMSE to a standard deviation, signifying the usual error in predictions of movie ratings.

If the outcome exceeds 1, it indicates that typical error surpasses one star, which is deemed undesirable. The formula for computing the RMSE for vectors of ratings and their corresponding predictions is expressed as follows:

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

The lower the better, as said previously.

2.2.1 Average movie rating model

The first basic model predicts the same rating for all movies and compute the dataset's mean rating. The expected rating of the underlying data set is between 3 and 4.

By building the simplest possible recommender system by predicting the same rating for all movies regardless of user who give it. A model based approach assumes the same rating for all movie with all differences explained by random variation :

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

with $\epsilon_{u,i}$ independent error sample from the same distribution centered at 0 and μ the "true" rating for all movies.

This very simple model makes the assumption that all differences in movie ratings are explained by random variation alone. The estimate that minimize the RMSE is the least square estimate of $Y_{u,i}$, in this case, is the average of all ratings: The expected rating of the underlying data set is between 3 and 4.

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

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

If predict all unknown ratings with μ or mu, it obtain the first naive RMSE:

```
naive_rmse <- RMSE(final_holdout_test$rating, mu)
naive_rmse
```

```
## [1] 1.061202
```

Here, it represent results table with the first RMSE:

```
rmse_results <- data_frame(method = "Average movie rating model", RMSE = naive_rmse)
```

```
## Warning: `data_frame()` was deprecated in tibble 1.1.0.
## i Please use `tibble()` instead.
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_lifecycle_warnings()` to see where this warning was
## generated.
```

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

method	RMSE
Average movie rating model	1.061202

This give us baseline RMSE to compare with next modelling approaches. In order to do better than simply predicting the average rating, incorporate some of insights gained during the exploratory data analysis.

2.2.2 Movie effect model

To improve above model focus on the fact that, from experience, know that some movies are just generally rated higher than others. Higher ratings are mostly linked to popular movies among users and the opposite is true for unpopular movies.

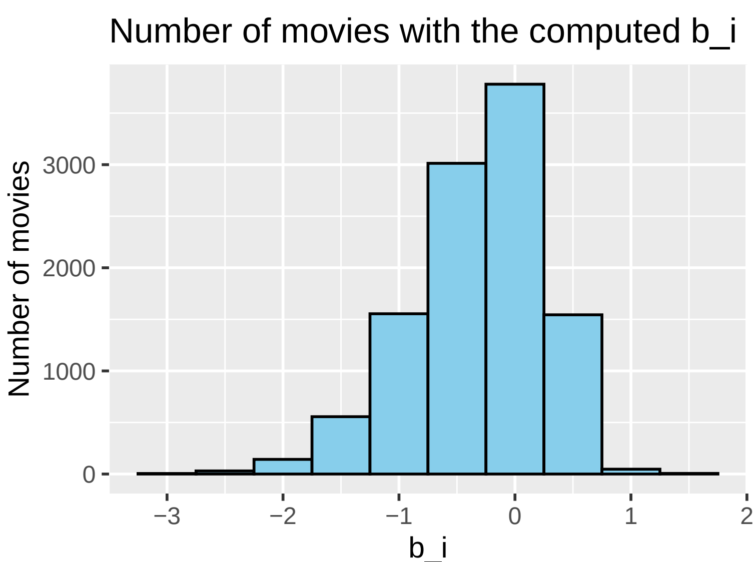
The estimated deviation of each movies' mean rating from the total mean of all movies μ. The resulting variable is called "b" (as bias) for each movie "i" b_i, that represents average ranking for movie i:

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

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

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

```
## Warning: ‘qplot()‘ was deprecated in ggplot2 3.4.0.
## This warning is displayed once every 8 hours.
## Call ‘lifecycle::last_lifecycle_warnings()‘ to see where this warning was
## generated.
```



This is called the penalty term movie effect. Once prediction improve, using this model.

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

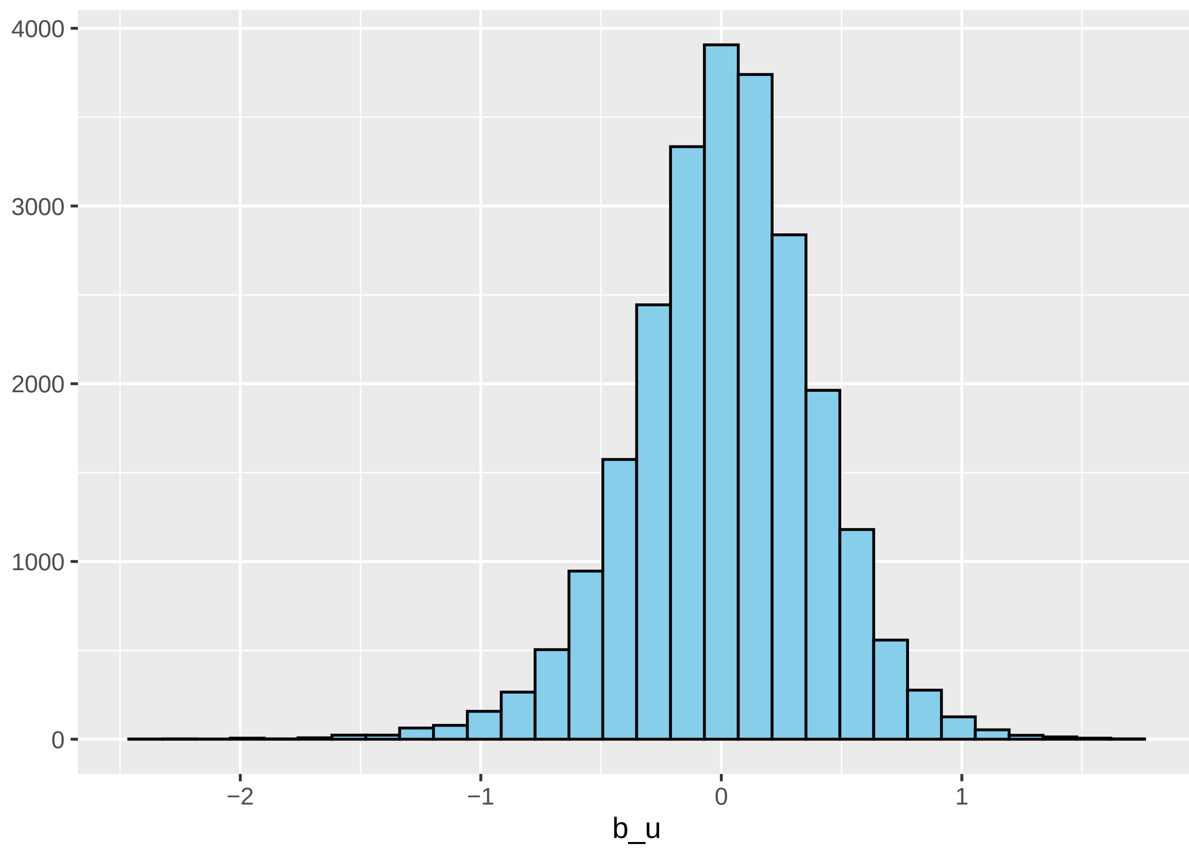
method	RMSE
Average movie rating model	1.0612018
Movie effect model	0.9439087

Predicted movie rating based on the fact that movies are rated differently by adding the computed b_i to μ. If an individual movie is on average rated worse that the average rating of all movies μ , predict that it will rated lower that μ by b_i, the difference of the individual movie average from the total average. Then see an improvement but this model does not consider the individual user rating effect.

2.2.3 Movie and user effect model

Compute the average rating for user μ, for those that have rated over 100 movies, said penalty term user effect. In fact users affect the ratings positively or negatively.

```r
user_avgs<- edx %>%
  left_join(movie_avgs, by='movieId') %>%
  group_by(userId) %>%
  filter(n() >= 100) %>%
  summarize(b_u = mean(rating - mu - b_i))
user_avgs%>% qplot(b_u, geom ="histogram", bins = 30, data = ., color = I("black"), fill = I("skyblue"))
```



There is substantial variability across users as well:

some users are very cranky and other love every movie. This implies that further improvement to model my be:

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

where b_u is a user-specific effect. If a cranky user (negative b_u rates a great movie (positive b_i), the effects counter each other and it may be able to correctly predict that this user gave this great movie a 3 rather than a 5.

Compute an approximation by computing μ and b_i, and estimating b_u, as the average of $Y_{u,i} - \mu - b_i$.

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

Construct predictors and see RMSE improves:

```
predicted_ratings <- final_holdout_test%>%
  left_join(movie_avgs, by='movieId') %>%
  left_join(user_avgs, by='userId') %>%
  mutate(pred = mu + b_i + b_u) %>%
  pull(pred)

model_2_rmse <- RMSE(predicted_ratings, final_holdout_test$rating)
rmse_results <- bind_rows(rmse_results,
```

```
                          data_frame(method="Movie and user effect model",
                                     RMSE = model_2_rmse))
rmse_results %>% knitr::kable()
```

method	RMSE
Average movie rating model	1.0612018
Movie effect model	0.9439087
Movie and user effect model	0.8653488

2.2.4 Rating predictions further reduced the RMSE

But made stil mistakes on first model (using only movies). The supposes "best" and "worst"movie were rated by few users, in most cases just one user. These movies were mostly obscure ones. This is because with a few users, we have more uncertainty. Therefore larger estimates of b_i, negative or positive, are more likely.

2.2.5 Large errors can increase RMSE

Until now, computed standard error and constructed confidence intervals to account for different levels of uncertainty.However, when making predictions, need one number, one prediction, not an interval.

For this, introduce the concept of regularization, that permits to penalize large estimates that come from small sample sizes. The general idea is to add a penalty for large values of b_i to the sum of squares equation that minimize.

So having many large b_i, make it harder to minimize. Regularization is a method used to reduce the effect of overfitting.

2.2.6 Regularized movie and user effect model

So estimates of b_i and b_u are caused by movies with very few ratings and in some users that only rated a very small number of movies. Hence this can strongly influence the prediction.

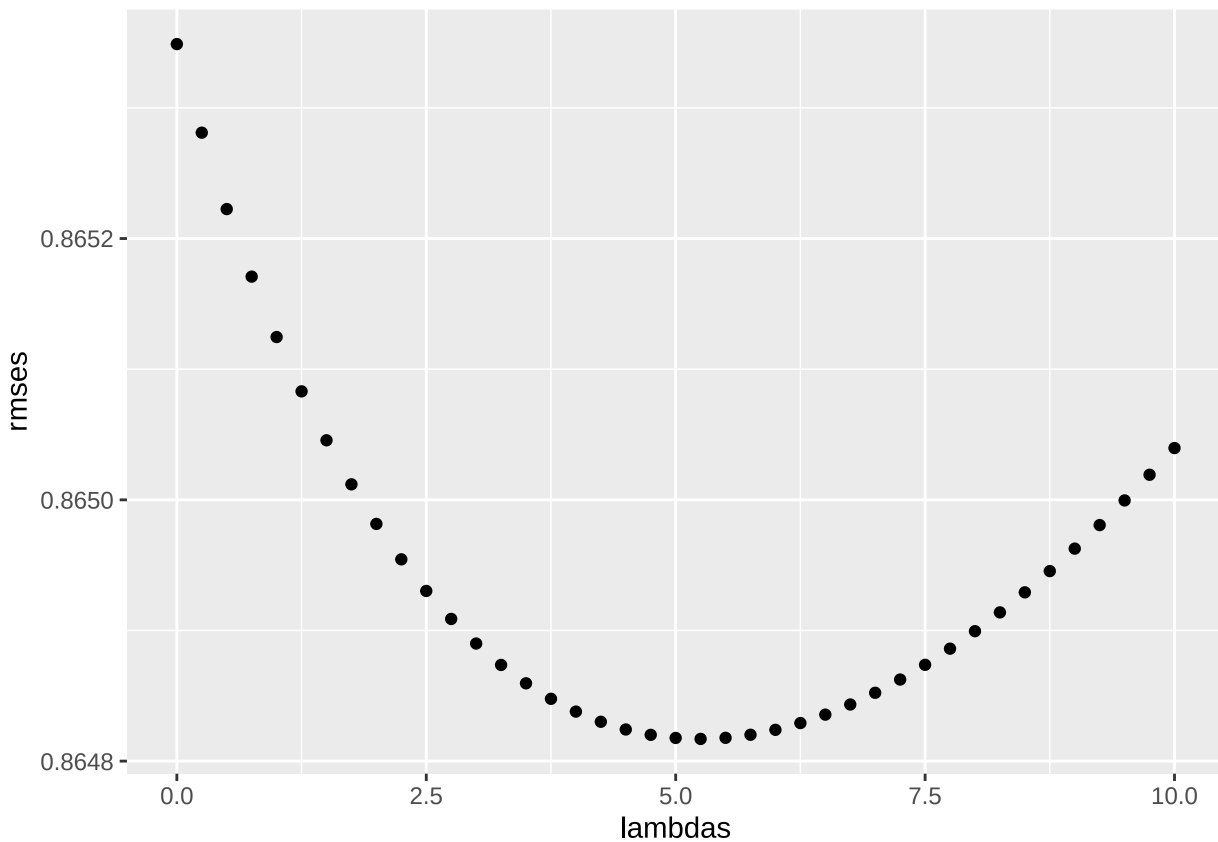
The use of the regularization permits to penalize these aspects.Find the value of lambda (that is a tuning parameter) that will minimize the RMSE. This shrinks the b_i and b_u in case of small number of ratings.

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

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

  mu <- mean(edx$rating)

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

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

```
  predicted_ratings <-
    final_holdout_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, final_holdout_test$rating))
})
```

Plot RMSE vs lambdas to select the optimal lambda

```
qplot(lambdas, rmses)
```



For the full model, the optimal lambda is:

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

```
## [1] 5.25
```

For the full model, the optimal lambda is: 5.25

The new results will be:

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

method	RMSE
Average movie rating model	1.0612018
Movie effect model	0.9439087
Movie and user effect model	0.8653488
Regularized movie and user effect model	0.8648170

3 Chapter 3

3.1 Results

The RMSE values of all the represented models are the following:

method	RMSE
Average movie rating model	1.0612018
Movie effect model	0.9439087
Movie and user effect model	0.8653488
Regularized movie and user effect model	0.8648170

Therefore found the lowest value of RMSE that is 0.8648170.

4 Chapter 4

4.1 Discussion

The final model for project is the following:

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

In this model,$Y_{u,i}$ represents the predicted rating for user$u$ on item i. The components of the model are explained as follows:

1. μ is the overall average rating across all users and items.
2. b_i is the deviation in the rating for item ii from the overall average, capturing its individual effect.
3. b_u is the deviation in the rating behavior of user uu from the overall average, accounting for user-specific preferences.
4. $\epsilon_{u,i}$ denotes the error term, representing any random or unaccounted factors affecting the rating.

This model is effective under the assumption that the average user tends not to rate a particularly good or popular movie with an excessively large positive b_i value, indicating a strong liking for a specific movie.The model accommodates the nuances of user preferences and item characteristics, striving to provide accurate predictions for user-item ratings based on deviations from the overall average.

5 Chapter 4

5.1 Conclusion

This analysis affirm to have built a machine learning algorithm to predict movie ratings with MovieLens dataset.The regularized model including the effect of user is characterized by the lower RMSE value and is hence the optimal model to use for the present project.

The optimal model characterised by the lowest RMSE value (0.8649857) lower than the initial evaluation criteria (0.86490) given by the goal of the present project.Improvements in the RMSE could be achieved by adding other effect (genre, year, age,..).Other different machine learning models could also improve the results further, but hardware limitations, as the RAM, are a constraint.

In conclusion, this project aimed to develop a machine learning algorithm for predicting movie ratings using the MovieLens dataset.The final model, a regularized movie and user effect model, achieved the lowest Root Mean Square Error (RMSE) of 0.8649857, surpassing the initial evaluation criteria of 0.86490.This model considers both movie and user effects, providing a more accurate prediction of movie ratings.

5.2 Limitations

Despite the success of the model, there are limitations to consider. The dataset's performance was influenced by movies with few ratings, leading to larger estimates that could impact the RMSE. Additionally, hardware limitations, such as RAM constraints, restricted the exploration of more complex machine learning models.

5.3 Future Work

Future work could involve further enhancing the model by incorporating additional features such as genre, year, and age. Exploring different machine learning models beyond the scope of this project may also yield improvements. Addressing hardware limitations to enable the exploration of more sophisticated algorithms and larger datasets could contribute to refining the predictive accuracy of movie ratings. Additionally, evaluating the model's performance on diverse datasets or extending the analysis to different domains could provide valuable insights.Continuous refinement and optimization are crucial for enhancing the robustness and applicability of the movie rating prediction algorithm.

6 Reference

[1] https://grouplens.org/datasets/movielens/10m/

[2] https://files.grouplens.org/datasets/movielens/ml-10m.zip

[3] https://www.statisticshowto.com/probability-and-statistics/regression-analysis/rmse-root-mean-square-error/