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

Adrian Zinovei

June 4, 2019

Contents

1 Overview **2**

1.1 Introduction 2

2 Methods and Analysis **5**

2.1 Data Analysis 5

2.2 Modelling Approach 9

2.2.1 Average movie rating model 10

2.2.2 II. Movie effect model 11

2.2.3 III. Movie and user effect model 12

2.2.4 IV. Regularized movie and user effect model 14

3 Results **15**

4 Conclusion **16**

5 The end **16**

1 Overview

This project is related to the MovieLens Project of the HervardX: PH125.9x Data Science: Capstone course. The present report starts with a general idea of the project and by representing its objective.

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.

1.1 Introduction

Introduction

Recommendation systems are one of the most famous machine learning models. They are extensively used by many companies (eg: Netflix, Amazon, Facebook, etc.) to improve and enhance user experiences and increase revenues by recommending the most relevant products to their customers.

This Harvard Data Science Capstone is the final assignment for HarvardX - Data Science Professional Certificate from Harvard University.

This project is motivated by the Netflix challenge that was organized in October, 2006. Netflix offered one million dollars reward to anyone that could improve their recommendation systems by 10%.

Dataset->>

For this assignment, we will go through all the steps to create a movie recommendation system using the MovieLens dataset, collected by GroupLens Research as the Netflix Datasets are private.

We will be using the 10M version of the MovieLens dataset to make the computation a little easier.

Goal->>

The objective of this report is to predict, in the most accurate and comprehensive way, the user movie

ratings by implementing, testing and validating different machine learning models.

The outcome is to provide a minimal typical error or RMSE (Root Mean Square Error) on the validation dataset with RMSE lower or equal to 0.87750.

The RMSE is defined as:

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

We define $y_{u,i}$ as the rating for movie i by user u and denote our prediction with $\hat{y}_{u,i}$ with N being the number of user/movie combinations and the sum occurring over all these combinations.

Key Steps

To achieve the project objectives, we will follow a comprehensive machine learning workflow:

1. Run the R code provided by Edx staff on the Capstone project. The script executes the following steps:

• Download the MovieLens 10M dataset

• Split the MovieLens dataset into training (Edx) and test (validation) datasets.

2. Explore the Edx dataset to discover the data and the available features. We will use some exploratory techniques such as data description, preparation, exploration and visualization.

3. Train different predictive models (4 in our case) and algorithms in order to find a recommendation model with the best possible outcome (RMSE) that meets our goals.

4. Results and Conclusions.

In the end all results and documents should be posted in My GitHub repository.

Data Preparation

The generated datasets are divided into two subsets:

• a training subset to train our algorithm, called Edx. This subset represents 90% of the generated dataset.

• a validation subset to predict the movie ratings, called validation. This subset represents 10% of the generated dataset.

Create Test and Validation Sets according to the requirements from the Capstone chapter. Here we have the result of the script.

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

Algorithm development is to be carried out on the “edx” subset only, as “validation” subset will be used to test the final algorithm.

2 Methods and Analysis

2.1 Data Analysis

To get familiar with the dataset, we find the first rows of “edx” subset as below. The subset contains the six variables “userID”, “movieID”, “rating”, “timestamp”, “title”, and “genres”. Each row represents a single rating of a user for a single movie.

```
userId movieId rating timestamp                         title                ge
nres
1      1     122       5 838985046              Boomerang (1992)              Comedy|R
omance
2      1     185       5 838983525               Net, The (1995)       Action|Crime|Th
riller
4      1     292       5 838983421               Outbreak (1995)  Action|Drama|Sci-Fi|Th
riller
5      1     316       5 838983392               Stargate (1994)        Action|Adventure|
Sci-Fi
6      1     329       5 838983392 Star Trek: Generations (1994) Action|Adventure|Drama|
Sci-Fi
7      1     355       5 838984474       Flintstones, The (1994)        Children|Comedy|F
antasy
```

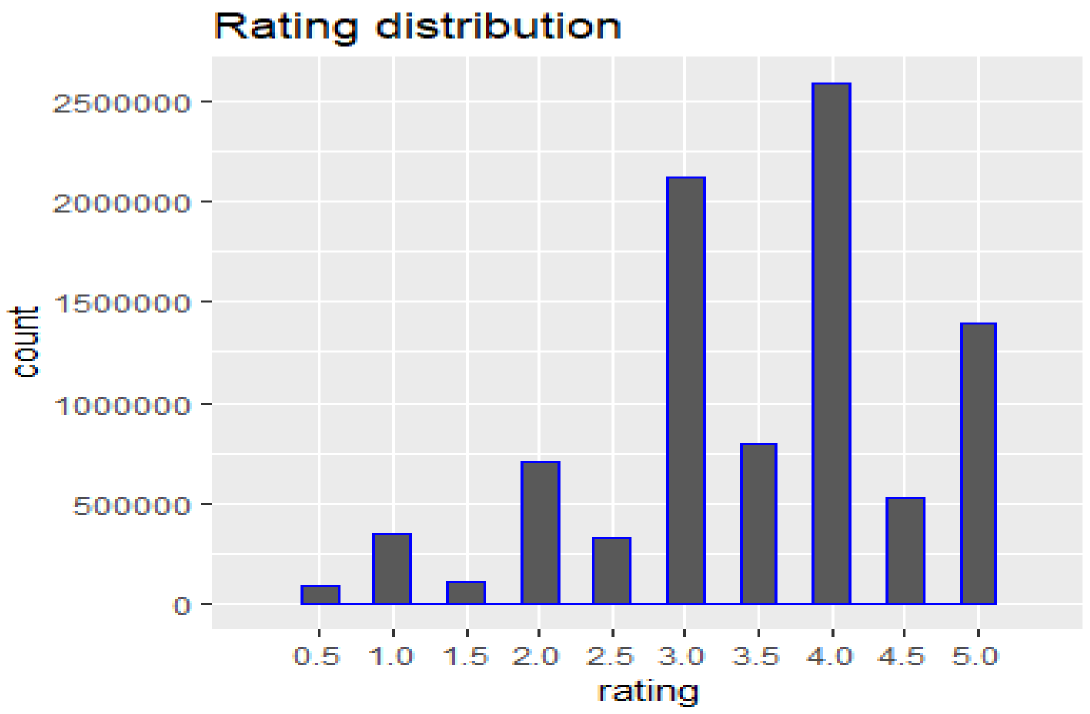
Summary for edx dataset:

```
     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

 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
```

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

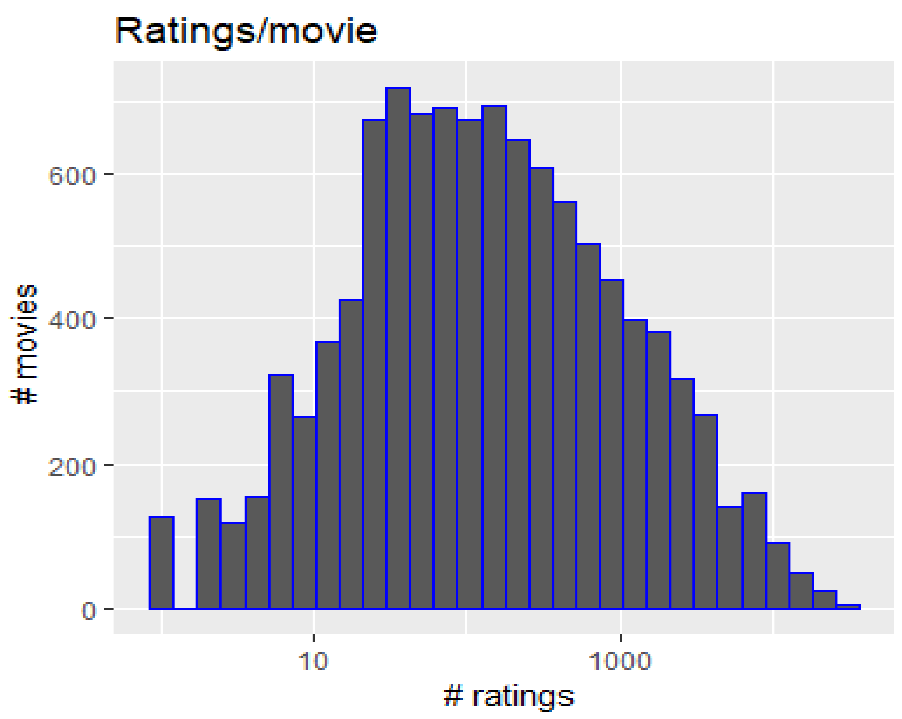
```
  nr_users nr_movies
1    69878     10677
```

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.



We can 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 our model as very low rating numbers might results in untrustworthy estimate for our predictions.

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.



As 15 movies that were rated only once appear to be obscure, predictions of future ratings for them will be difficult.

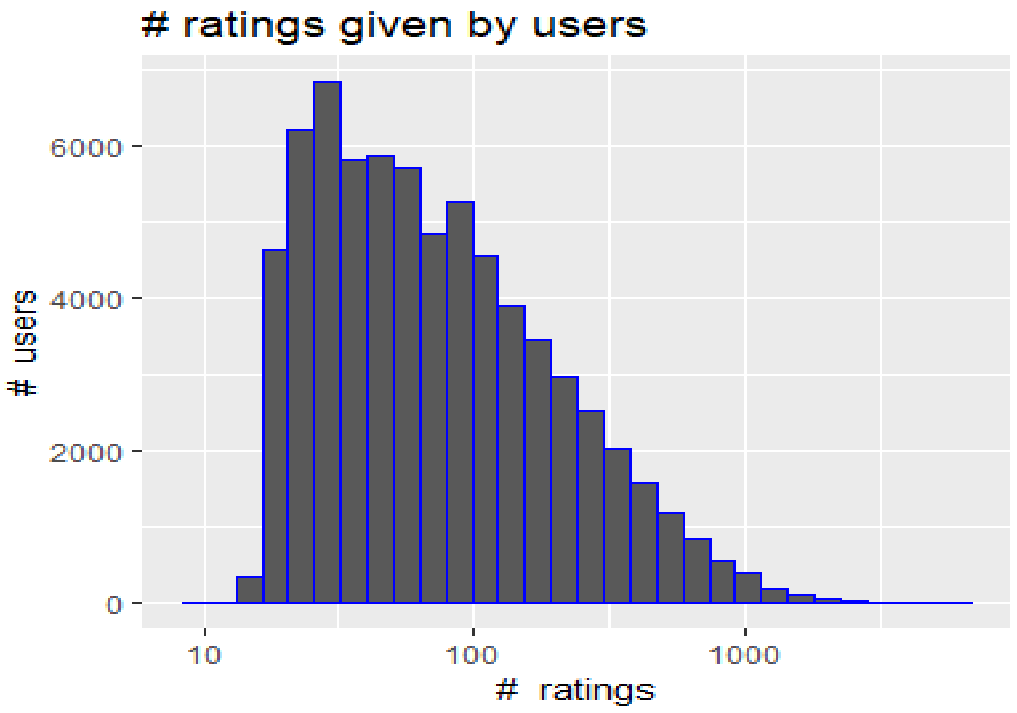
# Movie only rated once

edx %>% group_by(movieId) %>%
  summarize(count = n()) %>%
  filter(count == 1) %>%
  left_join(edx, by = "movieId") %>%
  group_by(title) %>%
  summarize(rating = rating, n_rating = count) %>%
  slice(1:15) %>%
  knitr::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|
|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|
```

# Nr of ratings given by users (plot)

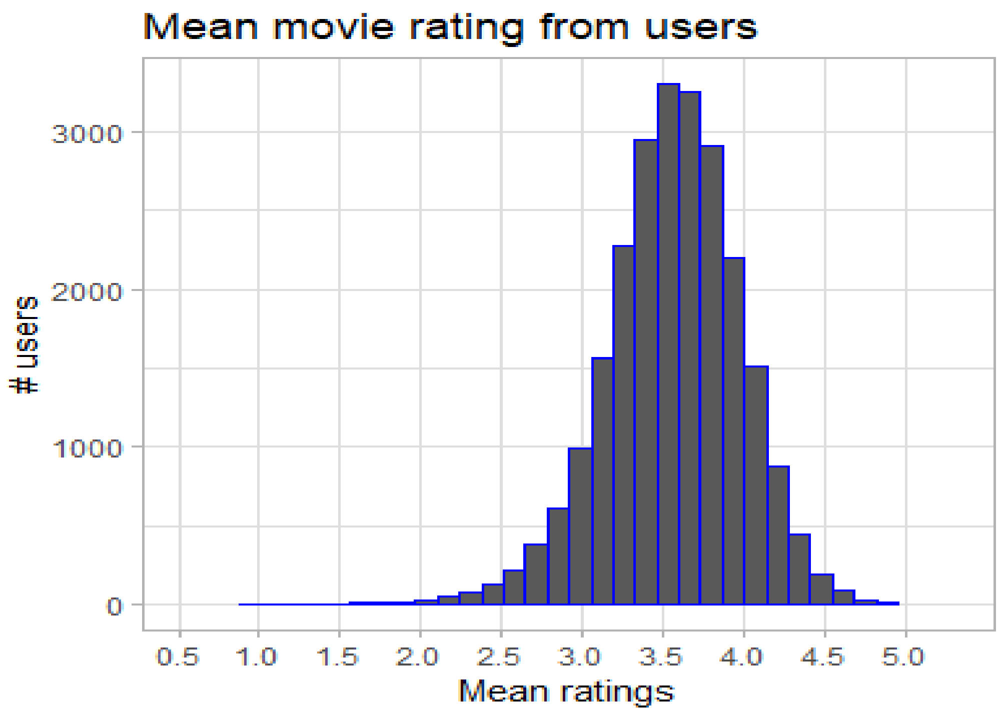
```
edx %>%  count(userId) %>%
  ggplot(aes(n)) +
  geom_histogram(bins = 30, color = "blue") +
  scale_x_log10() +
  xlab("#  ratings") +
  ylab("#  users") +
  ggtitle("# ratings given by users")
```



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.

# Mean movie rating from users (plot)

```
edx %>%  group_by(userId) %>%
  filter(n() >= 100) %>%
  summarize(b_u = mean(rating)) %>%
  ggplot(aes(b_u)) +
  geom_histogram(bins = 30, color = "blue") +
  xlab("Mean ratings") +
  ylab("# users") +
  ggtitle("Mean movie rating from users ") +
  scale_x_discrete(limits = c(seq(0.5,5,0.5))) +
  theme_light()
```



2.2 Modelling Approach

We write now the loss-function, previously anticipated, that compute the RMSE, defined as follows:

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

with N being the number of user/movie combinations and the sum occurring over all these combinations. The RMSE is our measure of model accuracy. We can interpret the RMSE similarly to a standard deviation: it is the typical error we make when predicting a movie rating. If its result is larger than 1, it means that our typical error is larger than one star, which is not a good result. The written function to compute the RMSE for vectors of ratings and their corresponding predictions is:

```
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, so we compute the dataset's mean rating. The expected rating of the underlying data set is between 3 and 4. We start 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:

Yu,i = μ + u,i

with $_{u,i}$ independent error sample from the same distribution centered at 0 and $_{\mu}$ the "true" rating for all movies. This very simple model assumes that all differences in movie ratings are explained by random variation alone. We know that 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.

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

mean_of_ratings
```

```
[1] 3.512465
```

If we predict all unknown ratings with $_{\mu}$ or mu, we obtain the first naive RMSE:

```
test_naive <- RMSE(validation$rating, mean_of_ratings)
```

```
[1] 1.061202
```

Here, we represent results table with the first RMSE:

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

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

```
|method                     |     RMSE|
|:--------------------------|--------:|
|Average movie rating model | 1.061202|
```

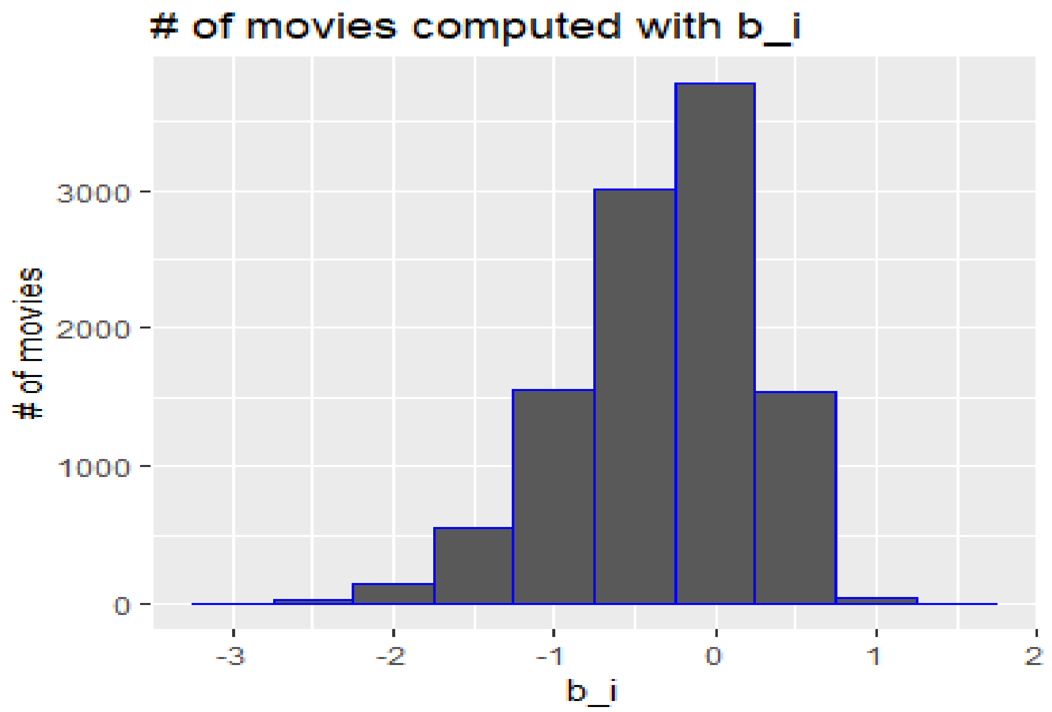
This give us our baseline RMSE to compare with next modelling approaches.

In order to do better than simply predicting the average rating, we incorporate some of insights we gained during the exploratory data analysis.

2.2.2 II. Movie effect model

To improve above model, we focus on the fact that, from experience, we 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. We compute 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:
Yu,i = μ + bi + u,i

```
movie_averages <- edx %>%
  group_by(movieId) %>%
  summarize(b_i = mean(rating - mean_of_ratings))
movie_averages %>% qplot(b_i, geom ="histogram", bins = 10, data = ., color = I("blue"),
                         ylab = "# of movies", main = "# of movies computed with b_i")
```



This is called the penalty term movie effect.

Our prediction improves once we predict using this model.
So, we have 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, we predict that it will rated lower that μ by b_i, the difference of the individual movie average from the total average. We can see an improvement, but this model does not consider the individual user rating effect.

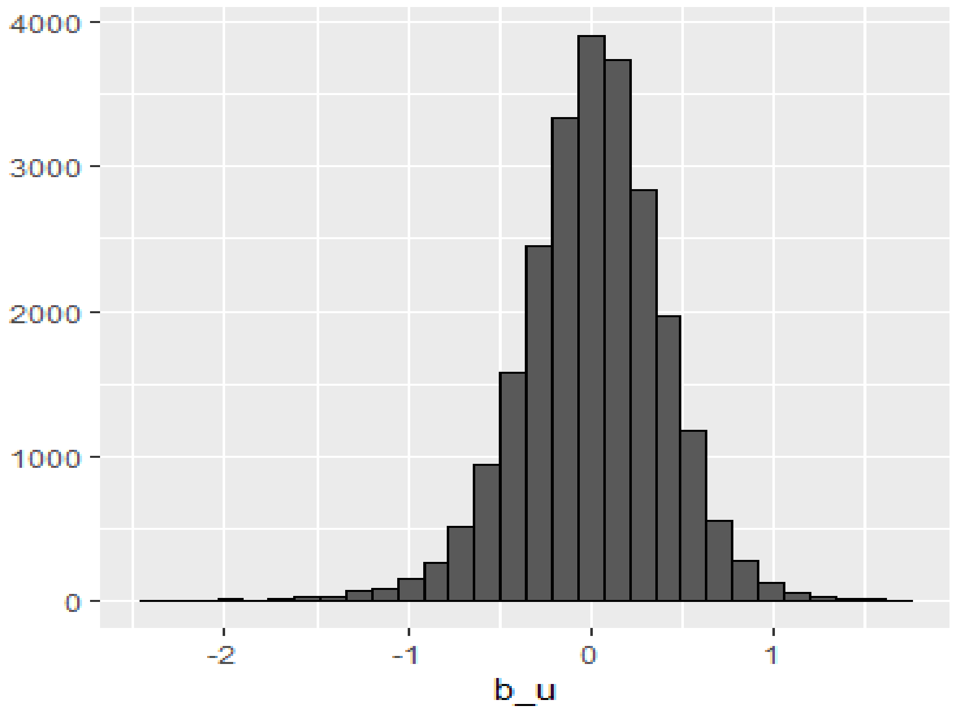
```
# Test and save RMSE results
forecasted_ratings <- mean_of_ratings+  validation %>%
  left_join(movie_averages, by='movieId') %>%
  pull(b_i)
1st_model_rmse <- RMSE(forecasted_ratings, validation$rating)
rmse_results <- bind_rows(rmse_results,
                          data_frame(method="Movie effect model",
                                     RMSE = 1st_model_rmse ))
# Check results
rmse_results %>% knitr::kable()
```

```
|method                     |     RMSE|
|:--------------------------|--------:|
|Average movie rating model | 1.061202|
```

2.2.3 III. Movie and user effect model

We 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.

```
averages_per_user<- edx %>%
  left_join(movie_averages, by='movieId') %>%
  group_by(userId) %>%
  filter(n() >= 100) %>%
  summarize(b_u = mean(rating - mean_of_ratings- b_i))
   averages_per_user%>% qplot(b_u, geom ="histogram", bins = 30, data = ., color = I("black"))
```



There is substantial variability across users as well: some users are very cranky and other love every movie.
This implies that further improvement to our model my be:

Yu,i = μ + bi + bu + 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 we may be able to correctly predict that this user gave this great movie a 3 rather than a 5.

We compute an approximation by computing μ and b_i, and estimating b_u, as the average of

Yu,i −μ−bi

```
# Test and save rmse results
forecasted_ratings <- validation%>%
  left_join(movie_averages, by='movieId') %>%
  left_join(averages_per_user, by='userId') %>%
  mutate(pred = mean_of_ratings+ b_i + b_u) %>%
  pull(pred)

2nd_model_rmse  <- RMSE(forecasted_ratings, validation$rating)
rmse_results <- bind_rows(rmse_results,
                          data_frame(method="Movie and user effect model",
                                     RMSE = 2nd_model_rmse ))

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

```
|method                     |     RMSE|
|:--------------------------|--------:|
|Average movie rating model | 1.061202|
```

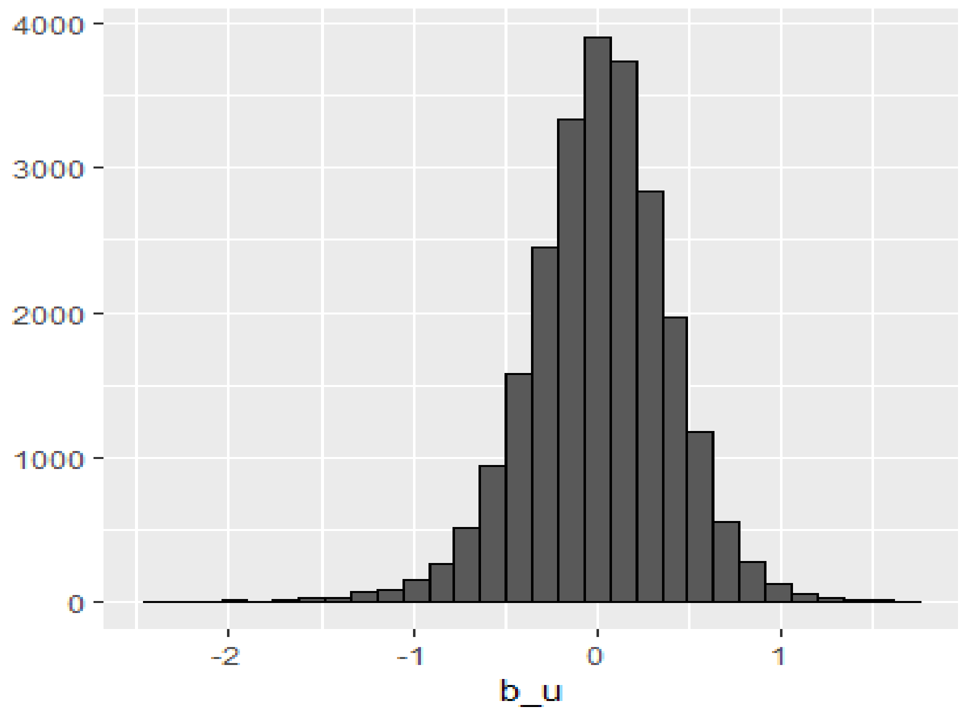
The supposes “best“ and “worst“ movie were rated by few users, in most cases just one user. These movies were mostly obscure ones. This is because with a few users, we have more uncertainty. Therefore, larger estimates of b_i, negative or positive, are more likely. Large errors can increase our RMSE.

We computed standard error and constructed confidence intervals to account for different levels of uncertainty. However, when making predictions, we need one number, one prediction, not an interval. For this we 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 we minimize. So, having many large b_i, make it harder to minimize. Regularization is a method used to reduce the effect of overfitting.

2.2.4 IV. Regularized movie and user effect model

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. We should 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.
We plot RMSE vs lambdas to select the optimal lambda:

```
# lambda is a tuning parameter
# Use cross-validation to choose it.
lambdas <- seq(0, 10, 0.25)


# For each lambda,find b_i & b_u, followed by rating prediction & testing
# note:the below code could take some time
all_rmses <- sapply(lambdas, function(l){

  mean_of_ratings<- mean(edx$rating)

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

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

  forecasted_ratings <-
    validation %>%
    left_join(b_i, by = "movieId") %>%
    left_join(b_u, by = "userId") %>%
    mutate(pred = mean_of_ratings+ b_i + b_u) %>%
    pull(pred)

  return(RMSE(forecasted_ratings, validation$rating))
})
# Plot all_rmses vs lambdas to select the optimal lambda
qplot(lambdas, all_rmses)
# The optimal lambda for all rmses
lambda <- lambdas[which.min(all_rmses)]
lambda
```



For the full model, the optimal lambda is:

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

> [1] 5.25

For the full model, the optimal lambda is: 5.25 The

new results will be:

```
rmse_results <- bind_rows(rmse_results,
                    data_frame(method="Regularized movie and user effect model",
                          RMSE = min(all_rmses)))
```

3 Results

The RMSE values of all the represented models are the following:
We therefore found the lowest value of RMSE that is 0.864817.

```
|Average movie rating model                 | 1.061202|
|Regularized movie and user effect model | 0.864817|
```

4 Conclusion

In the end 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 (0.864817) lower than the initial evaluation criteria (0.8775) given by the goal of the present project.

5 The end

```
> print("Operating System Settings Adrian Zinovei’s Notebook:")
[1] "Operating System Settings Adrian Zinovei’s Notebook:"
> version
               _
platform       x86_64-w64-mingw32
arch           x86_64
os             mingw32
system         x86_64, mingw32
status
major          3
minor          5.1
year           2018
month          07
day            02
svn rev        74947
language       R
version.string R version 3.5.1 (2018-07-02)
nickname       Feather Spray
```