HarvardX Data Science Capstone: MovieLens

Mootaz Abdel-Dayem

03 January 2021

Overview

This capstone project is 1 out of 2 for the final course in the Harvard University Professional Certificate in Data Science. The goal of the MovieLens capstone project is to utlilize the skills gained in the program to build a prediction and recommendation model with a target RMSE < 0.86490. To accomplish this goal, the dataset was loaded and divided into two sets: a Test dataset was created, and an Exploratory dataset was also created for exploratory data analysis. In addition to the original libraries provided by edx, more libraries were loaded to be used with data visualizations and other operations to produce this predictive model. Once the data was explored and examined, four models were developed and tested with final 4th model producing RMSE $= 0.8648170$ which is lower than the target RMSE < 0.86490.

Introduction

MovieLens is a large dataset available on the web about movies and their ratings. It was created in 1997 by GroupLens Research, a research lab at the University of Minnesota. The goal of the research was to collect research data provided by users and turn this this large dataset to a personalized recommendation system. Many of Data Scientists around the world use MovieLens dataset to learn data science and machine learning. Specifically to use it to learn recommendations based on machine learning practices. The MovieLens dataset includes $10M^+$ movie ratings, including more than 60,000 users and 10,000 movies.

This MovieLens capstone project consists of the following files: 1) MovieLens Capstone.Rproj: Project file for RStudio. 2) MovieLens R Script.R: An R script to build, test and validate movie ratings predictions models based on the MovieLens dataset. 3) MovieLens Report.Rmd: An R Markdown report. 4) MovieLens Report.pdf: Detailed description of MovieLens Project and Script. This is a PDF version of the R Markdown report.

The R Script for this project consists of three parts: I. LOADING DATA: Data gets loaded and prepared using data wrangling. II. EXPLORATORY DATA ANALYSIS: Understanding and examing the dataset III. MODELS: 4 Models are developed to compare their RMSES

Finally, we conclude which model provides the lowest RMSE.

I. LOADING DATA

The dataset is loaded using the code provided by edx.The datset is divided into edx set and 10% validation set. The edx set will be split into training and test set,and validation set will be used for final evaluation.

Now we save the edx and validation files:

```
save(edx, file="edx.RData")
save(validation, file = "validation.RData")
```

Before exploring the data, we need to install more packages and load their libraries for visualizations and analysis

```r
if(!require(ggplot2)) install.packages("ggplot2", repos = "http://cran.us.r-project.org")
if(!require(lubridate)) install.packages("lubridate", repos = "http://cran.us.r-project.org")
if(!require(dplyr)) install.packages("dplyr", repos = "http://cran.us.r-project.org")
if(!require(knitr)) install.packages("knitr", repos = "http://cran.us.r-project.org")
if(!require(markdown)) install.packages("markdown", repos = "http://cran.us.r-project.org")


  library(ggplot2)
  library(lubridate)
  library(dplyr)
  library(knitr)
  library(markdown)
```

II. EXPLORATORY DATA ANALYSIS

After the data gets loaded, we start exploring and examing the data by looking at the data structure and data types. Based on running the below code, we can see that there are 6 variables: userId, movieID, rating, timestamp, title, genres.

```
##   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
```

By exploring the data, we also see that the year needs to be separated from the title and also the Genres need to be separated. This is necessary if we decide to use the year and genres in our prediction models.

Now we let's collect some statistics about the data by summarizing it:

```
##      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
##
##
##
```

Based on the summarized data, we see that the minimum rating is 0.5 and max is 5 and the mean for the rating is 3.512 and the median value is 4.0.

Now let's find how how many unique users and unique movies in the edx dataset:

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

Let's check both the edx and the validation files:

```
## Classes 'data.table' and 'data.frame':   9000055 obs. of  6 variables:
##  $ userId   : int  1 1 1 1 1 1 1 1 1 1 ...
##  $ movieId  : num  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|A
##  - attr(*, ".internal.selfref")=<externalptr>
```

```
## Classes 'data.table' and 'data.frame':   999999 obs. of  6 variables:
##  $ userId   : int  1 1 1 2 2 2 3 3 4 4 ...
##  $ movieId  : num  231 480 586 151 858 ...
##  $ rating   : num  5 5 5 3 2 3 3.5 4.5 5 3 ...
##  $ timestamp: int  838983392 838983653 838984068 868246450 868245645 868245920 1136075494 1133571200
##  $ title    : chr  "Dumb & Dumber (1994)" "Jurassic Park (1993)" "Home Alone (1990)" "Rob Roy (1995)
##  $ genres   : chr  "Comedy" "Action|Adventure|Sci-Fi|Thriller" "Children|Comedy" "Action|Drama|Roman
##  - attr(*, ".internal.selfref")=<externalptr>
```

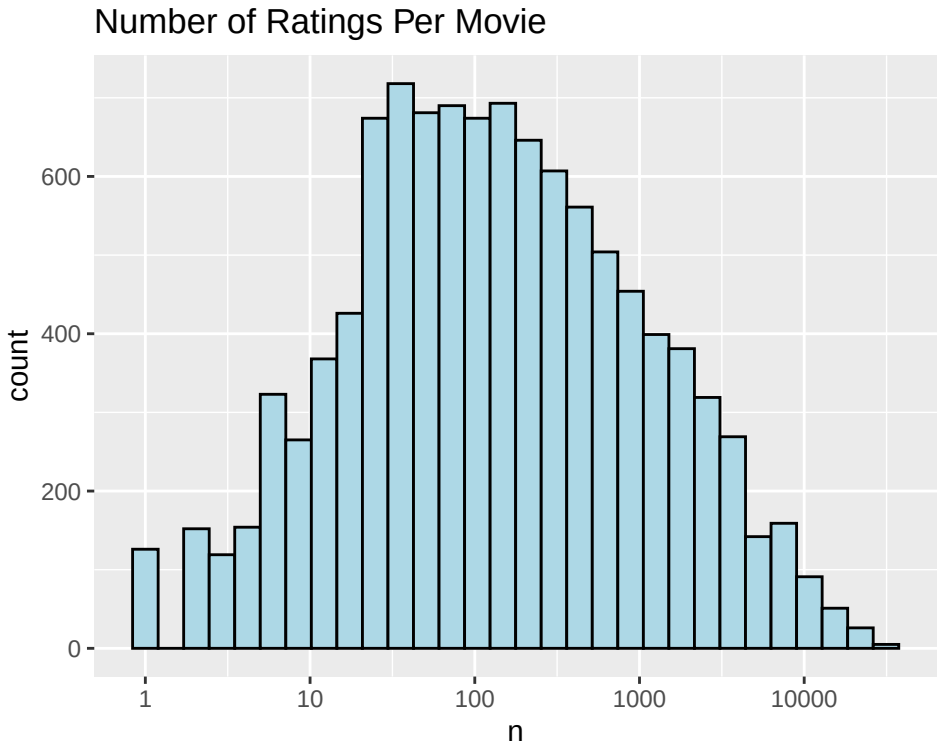
Let's explore the data to find out more about the most ratings:

```
## # A tibble: 5 x 2
##   rating   count
##    <dbl>   <int>
## 1    4   2588430
## 2    3   2121240
## 3    5   1390114
## 4    3.5  791624
## 5    2    711422
```

Now we can conclude that the most ratings belong to the 4 Stars Rating followed by the 3 and then 5 stars ratings.

To see the number of ratings for each movie in the dataset, we will plot the data in the following histogram:

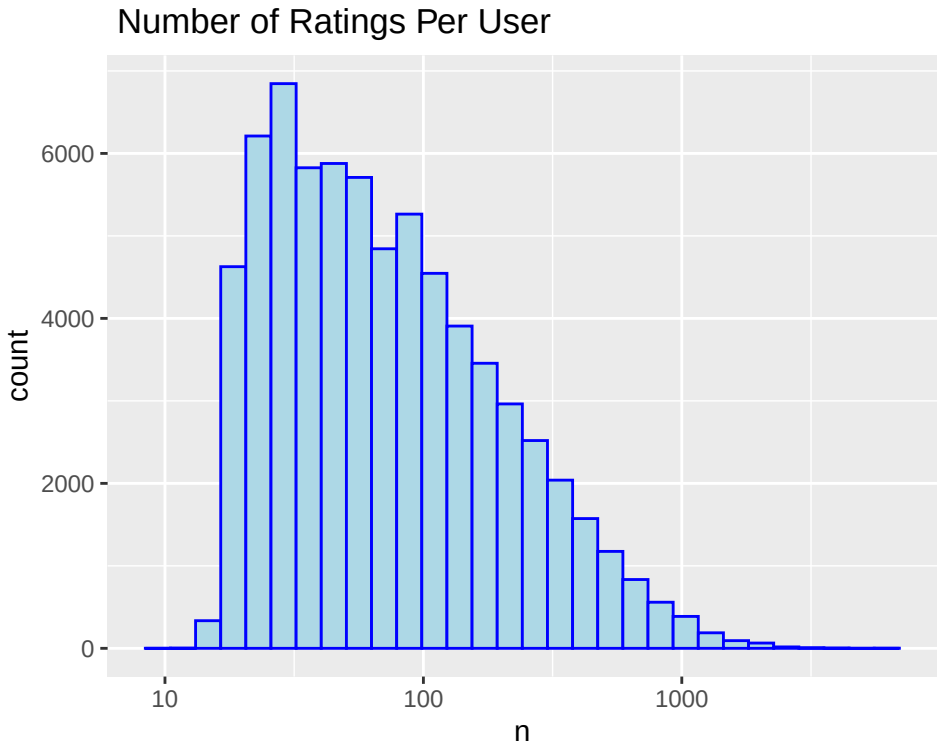
```
edx %>% count(movieId) %>% ggplot(aes(n))+
     geom_histogram(color = "black" , fill= "light blue")+
     scale_x_log10()+
     ggtitle("Number of Ratings Per Movie")+
     theme_gray()
```



Based on this plot, it seems that some movies get more ratings possibly because of popularity.

Now let's examine the number of ratings for each user:

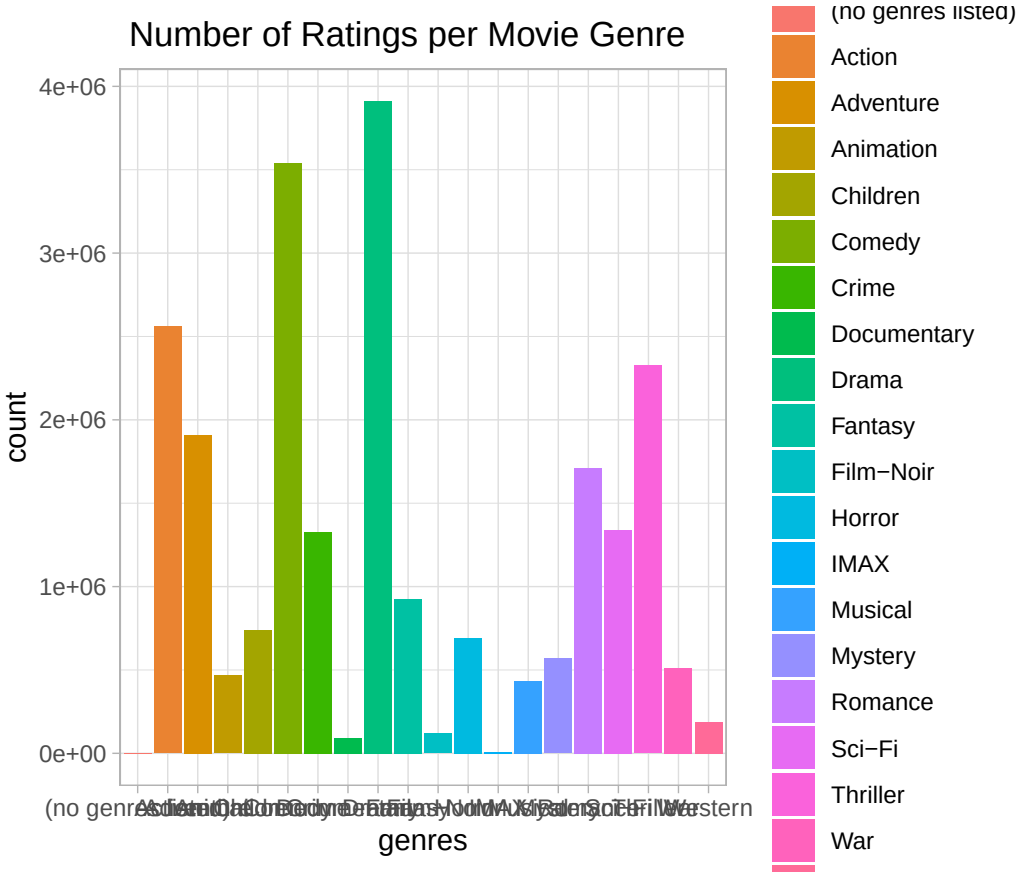
```
edx %>% count(userId) %>% ggplot(aes(n))+
    geom_histogram(color = "blue" , fill= "light blue")+
    ggtitle(" Number of Ratings Per User")+
    scale_x_log10()+
    theme_gray()
```



It seems that some users are more active than others.

Now let's visualize the number of ratings per Movie Genre:

```
edx %>% separate_rows(genres, sep = "\\|") %>%
    group_by(genres) %>%
    summarize(count = n()) %>%
    arrange(desc(count)) %>% ggplot(aes(genres,count)) +
    geom_bar(aes(fill =genres),stat = "identity")+
    labs(title = " Number of Ratings per Movie Genre")+
    theme(axis.text.x  = element_text(angle= 90, vjust = 50 ))+
    theme_light()
```



At this point, we need to partition the edx dataset before building the model into 20% for test set and 80% for the training set:

Also before building the models, we need to remove Users and Movies from the training set using the semi_join function:

```
  test_set <- test_set %>%
    semi_join(train_set, by = "movieId") %>%
    semi_join(train_set, by = "userId")
```

Finally, before going to build our first model, we need to create our RMSE Calculation Function. This RMSE function will compute the RMSE for vectors of ratings and their corresponding predictors:

```
  RMSE <- function(true_ratings, predicted_ratings){
    sqrt(mean((true_ratings - predicted_ratings)^2, na.rm = TRUE))
  }
```

III. MODELS

Model 1: Average Movie Rating Model

We will start with the simplest model. The first model will predict the same rating for all movies regardless of users. The model doesn't account for any bias because again, this is the simplest model for a start.

Simply computing the Mean rating for the dataset:

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

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

Now let's test the results based on a simple prediction:

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

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

At this point, we need to create a table to hold the RSME results for each model for comparaison:

```
   rmse_results <- data_frame(method = "Average Movie Rating Model", RMSE = naive_rmse)
   rmse_results %>% knitr::kable()
```

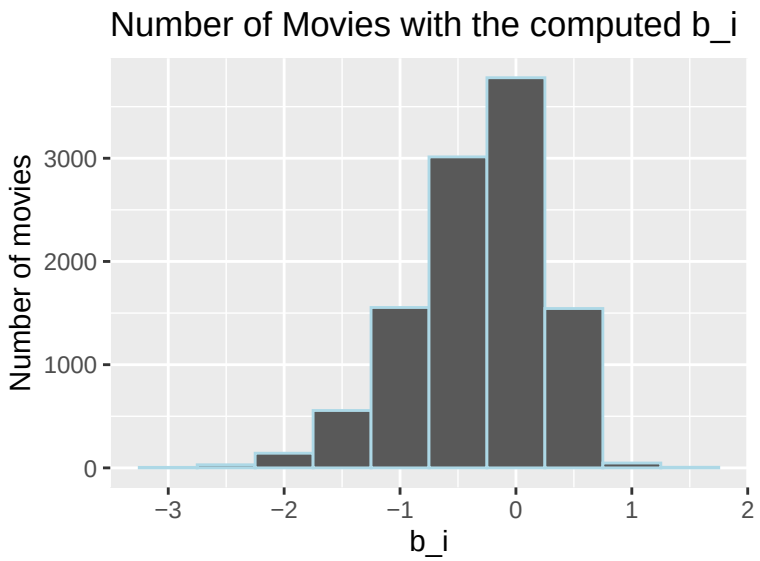
method	RMSE
Average Movie Rating Model	1.061202

Model 2: Movie Effect Model

Based on our earlier exploratory analysis, some movies were rated more than others. Let's modify model 1 by adding $b\ i$ to represent the average ranking for movie i .

Also because of the movie bias effect that we ignored in our previous model, let's now deal with it and use the least squared to do our estimation:

```
# This is a simple model taking into consideration the movie effect b_i
# Subtract  (Rating -  Mean) for each rating per Movie
# Then Plot the Number of Movies and including the Movie Effect (b_i)
   movie_avgs <- edx %>%
     group_by(movieId) %>%
     summarize(b_i = mean(rating - mu))
   movie_avgs %>% qplot(b_i, geom ="histogram", bins = 10, data = ., color = I("light blue"),
                        ylab = "Number of movies", main = "Number of Movies with the computed b_i")
```



We need to check and save the RMSE for this model:

```
predicted_ratings <- mu +  validation %>%
     left_join(movie_avgs, by='movieId') %>%
     pull(b_i)
   model_1_rmse <- RMSE(predicted_ratings, validation$rating)
```

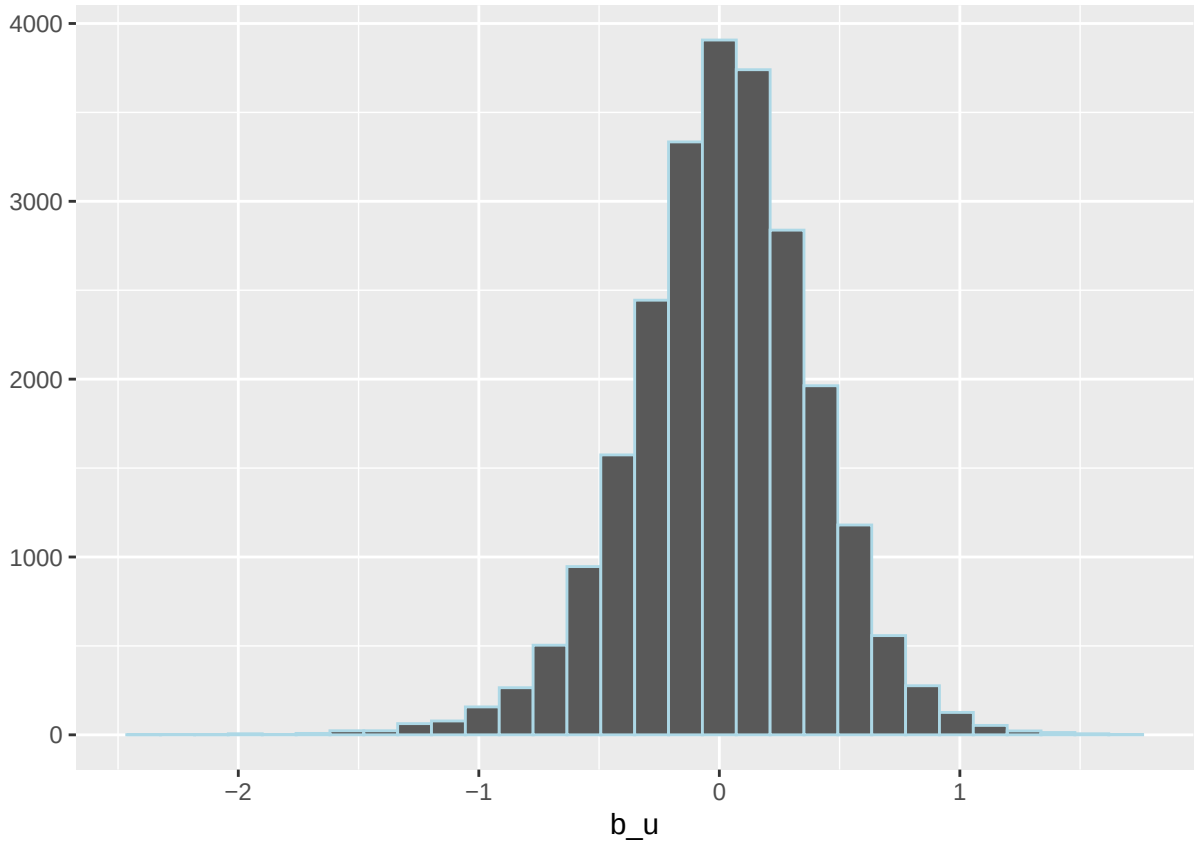
```
  rmse_results <- bind_rows(rmse_results,
                            data_frame(method="Movie Effect Model",
                                       RMSE = model_1_rmse ))


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

method	RMSE
Average Movie Rating Model	1.0612018
Movie Effect Model	0.9439087

Model 3: Movie and user effect model

In this model we will compare the user u with those users who have rated 100+ movies:

```
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("light blue"))
```



Plot shows variability across users ratings which suggests that this model needs some more improvement:

```
user_avgs <- edx %>%
    left_join(movie_avgs, by='movieId') %>%
```

```
    group_by(userId) %>%
    summarize(b_u = mean(rating - mu - b_i))
```

Checking and saving the RMSE for this model:

```
predicted_ratings <- validation%>%
    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, validation$rating)
  rmse_results <- bind_rows(rmse_results,
                            data_frame(method="Movie & User Effect Model",
                                       RMSE = model_2_rmse))

  # Check RMSE Result
  rmse_results %>% knitr::kable()
```

method	RMSE
Average Movie Rating Model	1.0612018
Movie Effect Model	0.9439087
Movie & User Effect Model	0.8653488

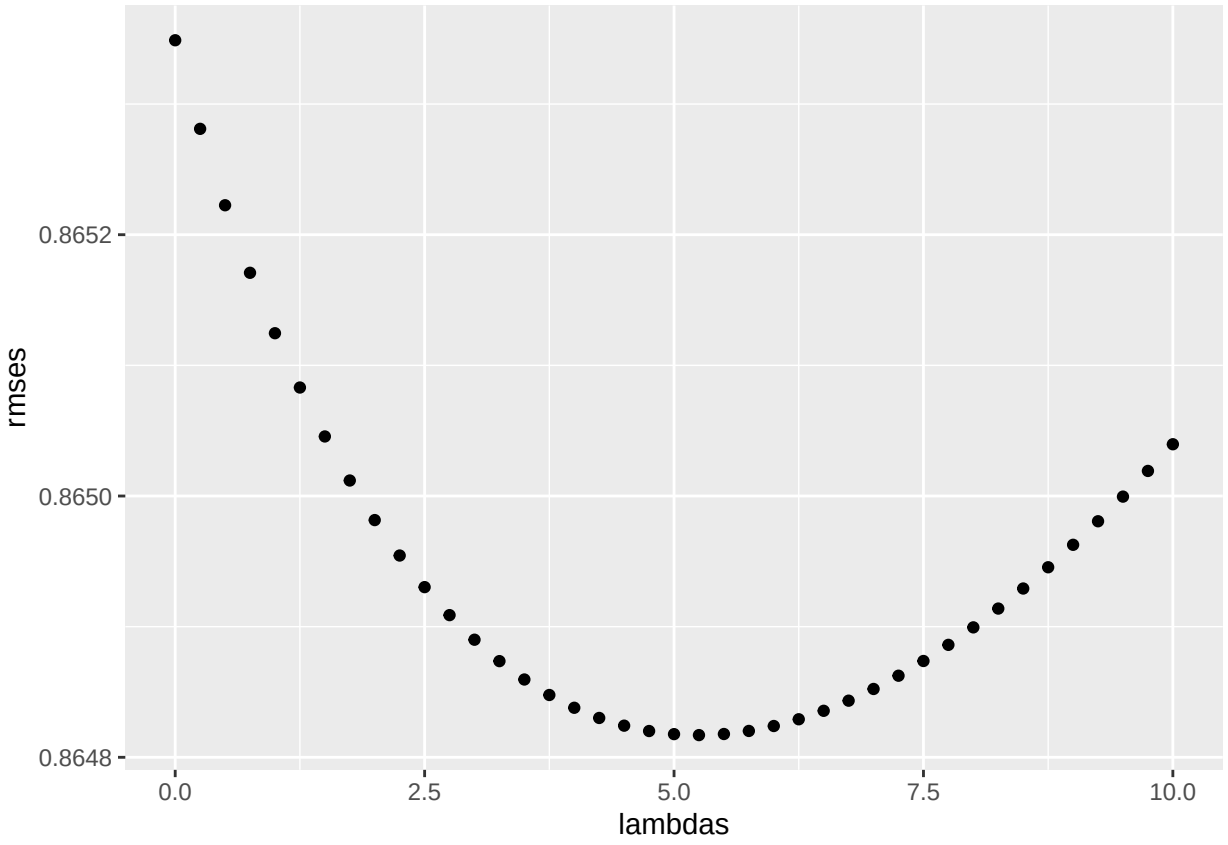
###Model 4: Regularized movie and user effect model

lambda is a tuning parameter, we will 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
  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 <-
      validation %>%
      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, validation$rating))
  })
```

Now let's plot the data to select the best lambda:

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



The optimal lambda is:

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

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

Now let's test and validate the RMSE Results:

```
rmse_results <- bind_rows(rmse_results,
                          data_frame(method="Regularized Movies & Users Effect Model",
                                     RMSE = min(rmses)))
```

Let's look at the RMSE table to compare the RMSE results for each model and fins the lowest RMSE:

method	RMSE
Average Movie Rating Model	1.0612018
Movie Effect Model	0.9439087
Movie & User Effect Model	0.8653488
Regularized Movies & Users Effect Model	0.8648170

Based on the RMSE table, each model kept lowering the RMSE but the last model (#4) is the best one because it lowered the RMSE below the target RMSE < 0.86490

Best Performing Model

Best Performing Model: "Regularized Movies & Users Effect Model"

Model RMSE = 0.8648170

RMSE Model result is less than target RMSE < 0.86490

Conclusion

This report covers the entire process of building a simple predictive recommendation system. It all started with downloading the MovieLens dataset, cleaning and preparing the data for analysis. Also the data was divided into a training and a testing sets. Next, an exploratory data analysis was performed to understand the data and to collect some statistics about it. After exploring the data, four models were created with the goal to get a lower RMSE less than our target RMSE goal < 0.86490.

Four models resulted in decreasing RMSE but a variation on the fourth model introducing a Regularization Parameter returned a minimized RMSE result.

The RMSE table shows the improvement of the model using different scenarios. We started with the simplest model using the mean value. However, we were missing the rating by one star. The next models included the Movie effect and Movie and user effect providing an improvement in lowered RMSE. Finally, the Regularization Model was used to penalize the data. The final RMSE is 0.8648170 with a significant improvement compared with the other previous models.