HarvardX Data Science Capstone Project _ MovieLens

Muditha Hapudeniya

01/10/2021

---

1. Introduction

This project report was done as a part of the capstone project to complete the EDX data science course. This project involves developing a movie recommender system using a publicly available MovieLens data set. MovieLens data set is a popular dataset for building a recommender system. This data set is available in different sizes. The full set has 27,000,000 ratings and 1,100,000 tag applications applied to 58,000 movies by 280,000 users. We use the 10M version for this project, which has 10 million ratings and 100,000 tag applications applied to 10,000 movies by 72,000 users.

According to Wikipedia, A recommender system, or a recommendation system, is a subclass of information filtering system that seeks to predict the "rating" or "preference" a user would give to an item. This function is very popular in some industries like movies, books and merchandise. Netflix and Amazon are two examples of companies that use recommender systems to recommend items.

There are three main approaches to build a recommender system: Content-based filtering, Collaborative filtering, and Hybrid recommender system.

- Content-based filtering - Content-based filtering methods are based on a description of the item and a profile of the user's preferences. These profiles are usually collected at the time of adding the item to the systems or at the time of user registration.
- Collaborative filtering - Collaborative filtering is based on the assumption that people who liked similar items in the past will like them equally in the future. The system tries to predict the user rating for a particular item by the history of the rating given by many users for the specific item. It generates the recommendations using this neighborhood. Collaborative filtering methods can also be memory-based and model-based.
- Hybrid recommender system - This method employs the features of both content-based and collaborative filtering to generate a more accurate prediction.

In this project, I use the collaborative filtering method.

The rest of the document describes the data acquisition, data description, data cleaning, data exploration, model building, and model evaluation.

2. Methodology

For this project, the course has provided the R script for downloading the data set and generating the training set and validation set.

2.1. Data Preparation

2.1.1. Installing Packpage In the given script, the required libraries are loaded. Here we use the tidyverse, caret and data.table library.

```
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")

library(tidyverse)
library(caret)
library(data.table)
```

2.1.2. Downloading the data set and Creating Train Set and Validation Set Once the libraries are loaded, it is possible to download the MovieLens data set. After downloading the dataset as a zip file, extract the movies data file and ratings data files to read into movies and ratings data frames. These two files are then joined to generate the full data set. Then combined dataset was splited in to a test set (90% of data) and a validation set (10% of data). Since the data set contains only a very few hyperparamenters, 10% of the total dataset was defined as a validation dataset. Further, checks were done to make sure that all the users and movies in the validation set were also present in the test set.

```
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 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)`
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)
```

2.2 Exploring the Training Dataset

2.2.1 Structure and basic information Before developing a movie recommendation system, it is important to examine the general properties or the structure of the data. The original data set contained almost 10 million of data. Test data set contains 90% of it.

```
nrow<-dim(edx)[1]
ncol<-dim(edx)[2]
nmovie<-n_distinct(edx$movieId)
nuser<-n_distinct(edx$userId)

tibble(Summary = c("Number of Row in Dataset",
                   "Number of Column in Dataset",
                   "Number of Movies",
                   "Number of Users"),
       Count = c(nrow,ncol,nmovie,nuser))%>% knitr::kable(caption = "Summary")
```

Table 1: Summary

Summary	Count
Number of Row in Dataset	9000055
Number of Column in Dataset	6
Number of Movies	10677
Number of Users	69878

As shown in table 1, The dataset contains 9,000,055 rows and 6 columns, involving 10,677 movies and 69,878 users. Each row represents a rating given by one user to one movie.

The data set has the following column headings.

```
colnames(edx)
```

```
## [1] "userId"    "movieId"   "rating"    "timestamp" "title"     "genres"
```

Missing values in the data set

Missing values can have an adverse effect on the model predictions. When checked for missing values,the data set did not contain any missing values.

```
sum(is.na(edx))
```

```
## [1] 0
```

2.2.2. Top 10 user rated the movies Table 2 shows the top to users who have rated the movies and the number of movies they have rated.

```
edx %>% group_by(userId) %>% summarise(Number_of_ratings = n()) %>%
  arrange(desc(Number_of_ratings)) %>%
  head(10) %>% knitr::kable(caption = "Top 10 users who had given most ratings")
```

Table 2: Top 10 users who had given most ratings

userId	Number_of_ratings
59269	6616
67385	6360
14463	4648

userId	Number_of_ratings
68259	4036
27468	4023
19635	3771
3817	3733
63134	3371
58357	3361
27584	3142

2.2.3. Top rated movies Table 3 shows the top to movies rated by users and the number of users who have rated.

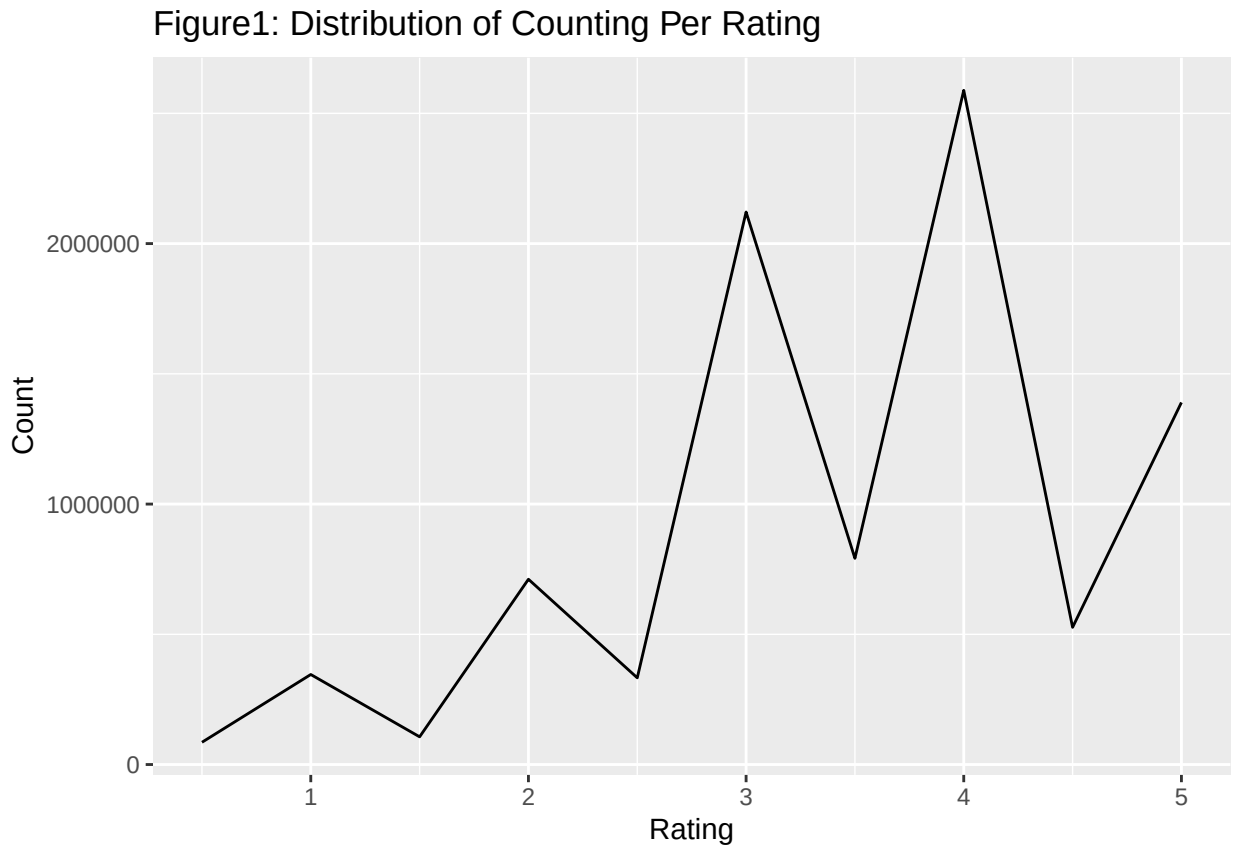
```
edx %>% group_by(movieId) %>% summarise(Number_of_ratings = n()) %>%
  arrange(desc(Number_of_ratings)) %>%
  head(10)%>% knitr::kable(caption = "Top 10 movies which had most ratings")
```

Table 3: Top 10 movies which had most ratings

movieId	Number_of_ratings
296	31362
356	31079
593	30382
480	29360
318	28015
110	26212
457	25998
589	25984
260	25672
150	24284

2.2.4. Distribution of rating The ratings are in the range from 0 to 5 with 0.5 increments. The figure 1 shows the number of ratings in each rating category. We can see that there are more full numbers than half numbers. Uses have generally given a rating between 3-4.

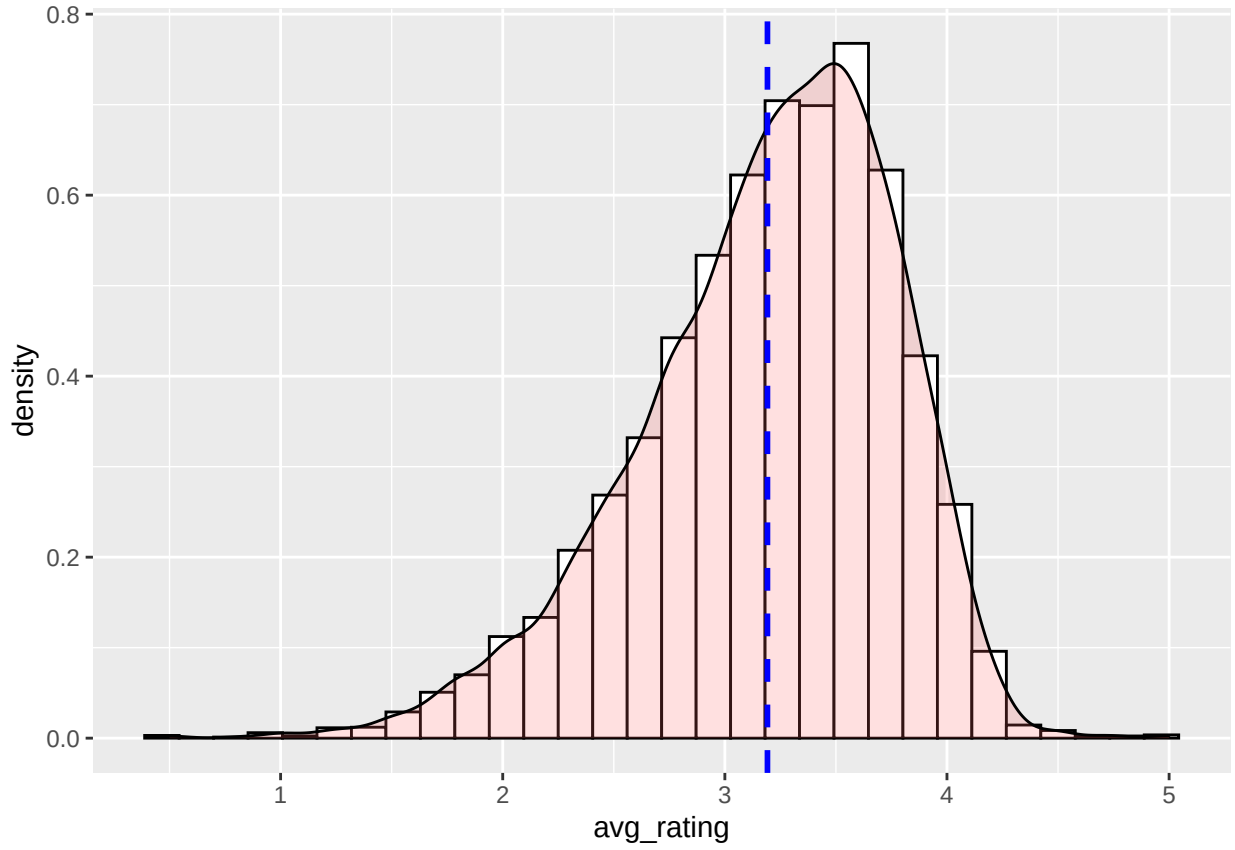
```
edx %>%
    group_by(rating) %>%
    summarize(count = n()) %>%
    ggplot(aes(x = rating, y = count)) +
    geom_line() +
  ggtitle("Figure1: Distribution of Counting Per Rating") +
  xlab("Rating") +
  ylab("Count") +
  scale_y_continuous(labels = function(x) format(x, scientific = FALSE))
```



2.2.5. Distribution of average rating per movie Figure 2 shows the distribution of the average rating for movies. We can see that there are very few movies with very low rating. The mean rating falls between 3 and 3.5. The distribution is slightly skewed to the right.

```
edx %>% group_by(movieId) %>% summarise(avg_rating = mean(rating)) %>%
  ggplot(aes(avg_rating)) +
  geom_histogram(aes(y=..density..), colour="black", fill="white")  +
  geom_density(alpha=.2, fill="#FF6666") +
  geom_vline(aes(xintercept=mean(avg_rating)),
             color="blue", linetype="dashed", size=1)
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



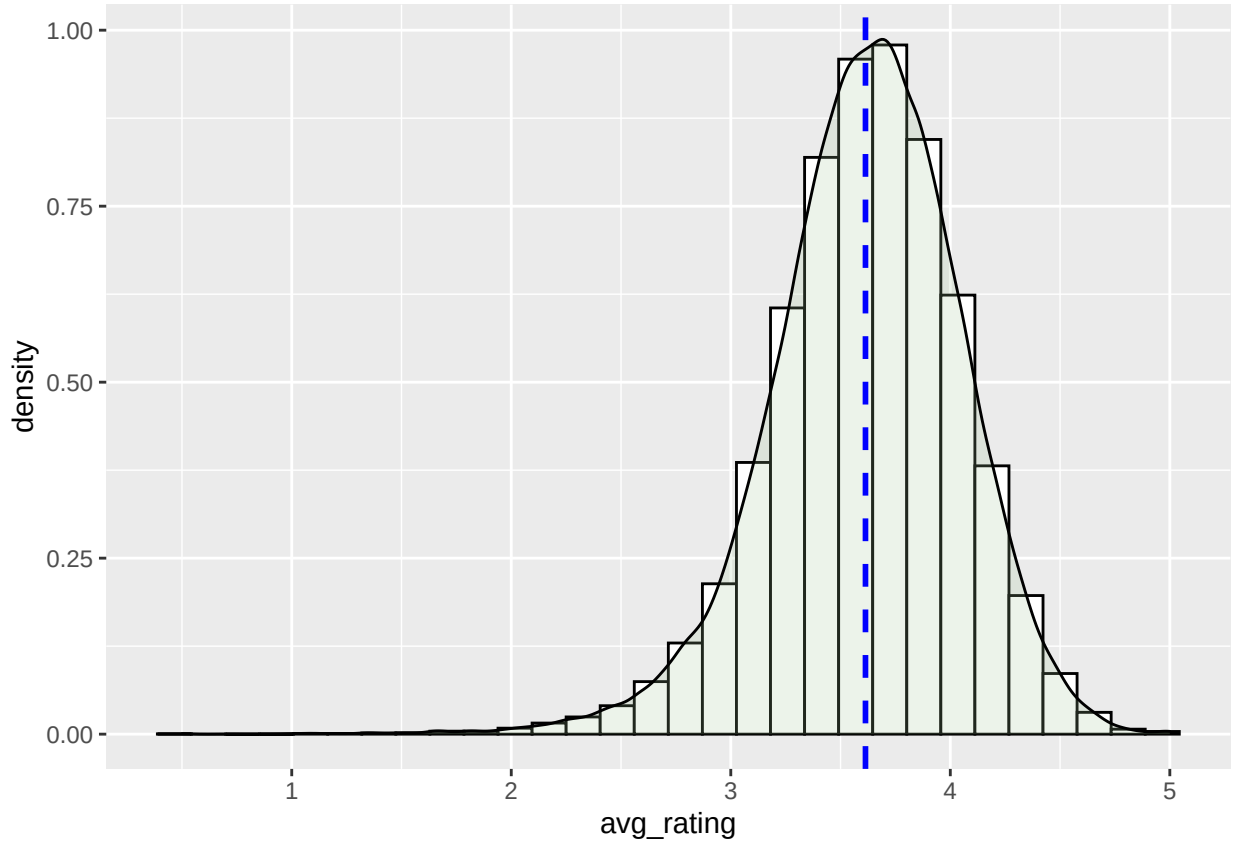

```
  ggtitle("Figure 1: Distribution of average ratings for movies")
```

```
## $title
## [1] "Figure 1: Distribution of average ratings for movies"
##
## attr(,"class")
## [1] "labels"
```

2.2.6. Distribution of average rating for users Figure 2 shows the distribution of the average rating for users. There are very few users who has given very low rating for movies. The distribution is almost normal.

```
edx %>% group_by(userId) %>% summarise(avg_rating = mean(rating)) %>%
  ggplot(aes(avg_rating)) +
  geom_histogram(aes(y=..density..), colour="black", fill="white")  +
  geom_density(alpha=.2, fill="#a4c496") +
  geom_vline(aes(xintercept=mean(avg_rating)),
             color="blue", linetype="dashed", size=1)
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```




```
  ggtitle("Figure 2: Distribution of average ratings for users")
```

```
## $title
## [1] "Figure 2: Distribution of average ratings for users"
##
## attr(,"class")
## [1] "labels"
```

2.3. Designing the Movie Recommendation System

2.3.1. Root Mean Squared Error In order to identify the best prediction model, we need to build several models and picks the best one that can predict more accurately. For this reason, we need to have a model evaluation method. Root-mean-square error (RMSE) is frequently used to measure the differences between values predicted by a model and the true values observed. The RMSE is a very common method for evaluating a model in recommender systems.

RMSE is the square root of the average of the squared difference between actual observation and prediction. In other words, it measures the average magnitude of the error. RMSE is always non-negative, and a value of zero would indicate a perfect fit to the data; but it is almost never achieved. In general, a lower RMSE is better than a higher one, so our aim is to find the model which has the lowest RMSE. As the RMSE is the square root of the average of squared error, larger errors have a disproportionately large effect on RMSE. It is important to consider this when building models.

```
RMSE <- function(true_rating, predicted_rating){
  sqrt(mean((true_rating - predicted_rating)^2))
}
```

2.3.2. Model by Considering average only The simplest way to predict rating is by assigning the global mean value. Here we assume that all the users give the same value to all the movies. Let's calculate the RSME by using the global mean.

```
# calculate the mean value for all the movie ratings
mu_hat <- mean(edx$rating)
mu_hat
```

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

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

# save this value for future comparisons
rmse_results <- data.frame("Predictive Method" = "Using the Global Average",
                           RMSE = naive_rmse)
rmse_results %>%
  knitr::kable(caption = "RMSE for models ")
```

Table 4: RMSE for models

Predictive.Method	RMSE
Using the Global Average	1.061202

This is a very crude way of predicting the rating as we have not considered the movie and user contribution here.

2.3.3. Modeling Movie Effects From figure 1, we have seen that not all movies are rated equally by users. The mean rating of the movie can indicate the general user preference. It is important to adjust the predicted rating considering the mean rating of a movie.

Let's calculate the mean bias rating for movies.

```
movie_avgs <- edx %>%
  group_by(movieId) %>%
  summarize(b_m = mean(rating - mu_hat))
```

Now add the movie bias to the mu_hat to generate the new rating value for the movies.

```
predicted_rating <- mu_hat + validation %>%
  left_join(movie_avgs, by='movieId') %>%
  .$b_m

model_1_rmse <- RMSE(predicted_rating, validation$rating)

# save RMSE for future reference

rmse_results <- bind_rows(rmse_results,
                          data.frame("Predictive Method"="Adding Movie Effect to the Model",
                                     RMSE = model_1_rmse ))
rmse_results %>% knitr::kable(caption="RMSE for models" )
```

Table 5: RMSE for models

Predictive.Method	RMSE
Using the Global Average	1.0612018

Predictive.Method	RMSE
Adding Movie Effect to the Model	0.9439087

According to the table 5, we can see that the RMSE came down, indicating a better prediction than using the global mean.

2.3.4. Modeling by adding the User Effects From figure 2, we have seen that not all users have rated movies equally. The mean rating given by users for each of the movies can indicate the general user preference. It is important to adjust the predicted rating considering the mean rating given by the users.

Let's calculate the mean bias rating for users and add them to the movie bias.

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

```
predicted_rating <- validation %>%
  left_join(movie_avgs, by='movieId') %>%
  left_join(user_avgs, by='userId') %>%
  mutate(pred = mu_hat + b_m + b_u) %>%
  .$pred

model_2_rmse <- RMSE(predicted_rating, validation$rating)

# save the result
rmse_results <- bind_rows(rmse_results,
                          data.frame("Predictive Method"="Add Movie + User bias to the Model",
                                     RMSE = model_2_rmse ))

rmse_results %>% knitr::kable(caption="RMSE for models")
```

Table 6: RMSE for models

Predictive.Method	RMSE
Using the Global Average	1.0612018
Adding Movie Effect to the Model	0.9439087
Add Movie + User bias to the Model	0.8653488

By adding both movie bias and user bias, the RMSE further came down.

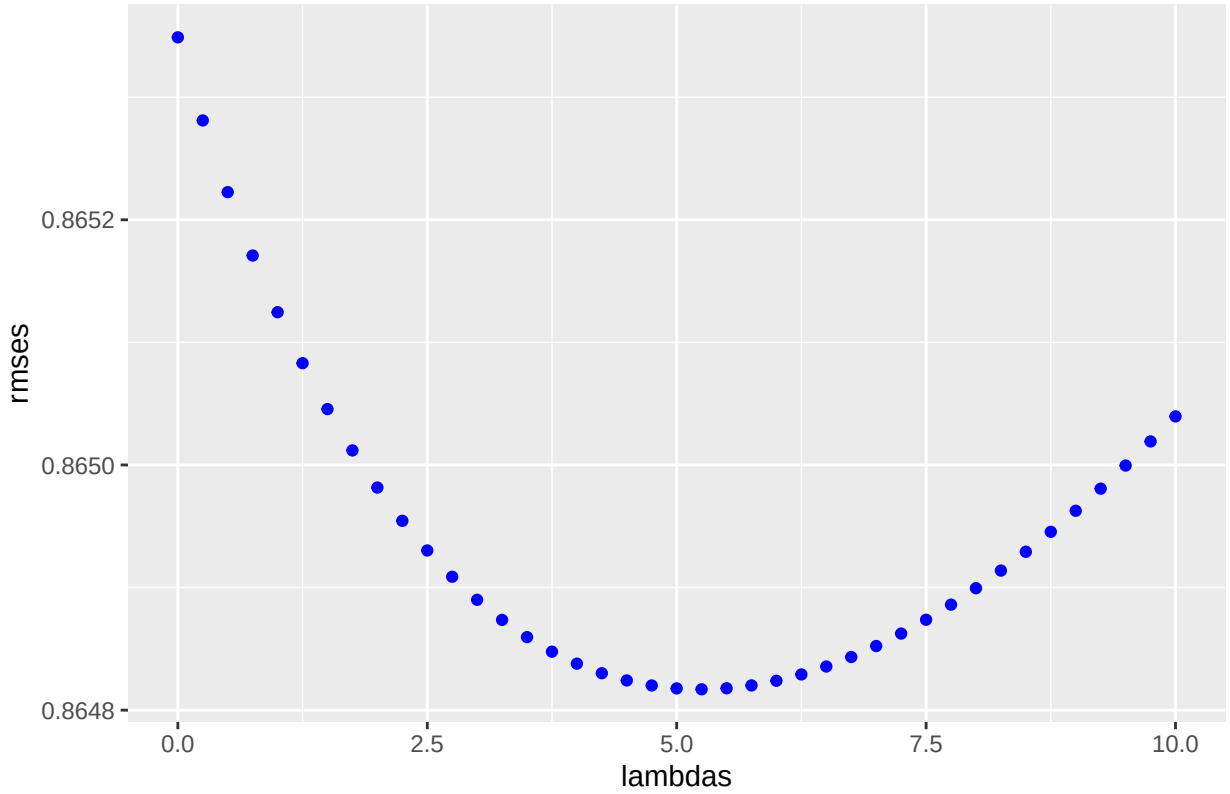
2.3.5. Regularization for movie and User Effect in the model There can be few users giving few extreme ratings for movies and few movies getting few extreme ratings affecting the mean rating value significantly. Since the RSME is sensitive to the outliers, it is important to tune the model to reduce the effects of outliers.

Regularization is a technique used to reduce the error in prediction by adding a penalty into the model to improve the function fitting on the training set and avoid over-fitting. It penalizes large estimate generated by using the small dataset and control the total variability of the effect. When the movie effect or the user effect is large, the penalty will also be increased.

We need to define a penalty tuning parameter, which is called lambda for this purpose. I defined the lambda parameter from 0 to 10 with 0.25 increment and ran the model with each parameter value to identify the minimal RMSE and the optimal lambda value.

```r
# define lambda
lambdas <- seq(0, 10, 0.25)

# calculate RMSE for each lambda
rmses <- sapply(lambdas, function(l){
# calculate the global mean
  mu_hat <- mean(edx$rating)

  # calculate the movie bias
  b_m <- edx %>%
    group_by(movieId) %>%
    summarize(b_m = sum(rating - mu_hat)/(n()+l))

  # calculate the user bias
  b_u <- edx %>%
    left_join(b_m, by="movieId") %>%
    group_by(userId) %>%
    summarize(b_u = sum(rating - b_m - mu_hat)/(n()+l))

  # predict the rating by adding the movie and user bias
  predicted_rating <-
    validation %>%
    left_join(b_m, by = "movieId") %>%
    left_join(b_u, by = "userId") %>%
    mutate(pred = mu_hat + b_m + b_u) %>%
    .$pred

    return(RMSE(predicted_rating, validation$rating))
})

# Generate the plot to visualize the RMSE for each lambda value
df <- data.frame(lambdas,rmses)
ggplot(df,aes(lambdas,rmses)) +
  geom_point(color="blue") +
  ggtitle("Figure 3: RMSE by lambda")
```

Figure 3: RMSE by lambda



The following code gives the lambda value for the minimum RMSE value.

```
lambdas[which.min(rmses)]
```

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

The lambda value for the minimum RMSE is 5.25. Now I can calculate the final RMSE using the lambda value.

```
lambda <- 5.25

movie_reg_avgs <- edx %>%
  group_by(movieId) %>%
  summarize(b_m2 = sum(rating - mu_hat)/(n()+lambda), n_i = n())

user_reg_avgs <- edx %>%
  left_join(movie_reg_avgs, by="movieId") %>%
  group_by(userId) %>%
  summarize(b_u2 = sum(rating - mu_hat-b_m2)/(n()+lambda), n_i = n())

predicted_rating <- validation %>%
  left_join(movie_reg_avgs, by = "movieId") %>%
  left_join(user_reg_avgs, by='userId') %>%
  mutate(pred = mu_hat + b_m2 + b_u2) %>%
  .$pred

model_3_rmse <- RMSE(predicted_rating, validation$rating)
```

```
rmse_results <- bind_rows(rmse_results,
data.frame("Predictive Method"="Add Regularized Movie +
                                    Regularized User Effects to the Model",
                                    RMSE = model_3_rmse ))

rmse_results %>% knitr::kable(caption="RMSE for models")
```

Table 7: RMSE for models

Predictive.Method	RMSE
Using the Global Average	1.0612018
Adding Movie Effect to the Model	0.9439087
Add Movie + User bias to the Model	0.8653488
Add Regularized Movie + Regularized User Effects to the Model	0.8648170

3. Results

Initially, I got a high RSME value from using the global mean for the prediction. This value finally came down to **0.8648** by reducing the movie bias, user bias and regularization.

```
data_frame("Final Result" ="Regularized Movie +
           Regularized User Effects Model",RMSE = model_3_rmse ) %>%
  knitr::kable(caption="Final RMSE value achived")
```

```
## Warning: `data_frame()` was deprecated in tibble 1.1.0.
## Please use `tibble()` instead.
```

Table 8: Final RMSE value achived

Final Result	RMSE
Regularized Movie + Regularized User Effects Model	0.864817

4. Conclusion

In this project, I learned how to build a movie recommendation system using collaborative filtering method. The global average was not a good estimate as there are biases in rating movies by users. These factors have to take in to consideration when modeling. Since the RSME is sensitive to the outliers in the data set, regularization methods have to be used for tuning the model. Considering all the facts, I could reduce the RMSE from 1.0612 to 0.8648. It indicated that the recommendation system performs well.

5. Limitations

No Past Data for New Movie/User

Since the above predictive model use the past rating records to predict the user's preference to recommend a movie for each user. This model will not accurete ratings
for new user and new movies as no past data is available.

6. Future work

Cross-Validation

In this project, the dataset was only split into training and validation datasets once to build the movie recommendation model. It is possible to split the data set repeatedly in to training set and test set. Then each set can be used to get the best predicted value. This process is called cross-validation. We could try cross validation to produce more stable prediction with lower RSME.

Including additional variable

Here we have considered only the user and the movie to predict the rating. We have not considered other available variable such as movie Genres. We may also use other factors such as age, gender etc. of the user to do more accurate predictions if those were available.