PH125.9x:Movielens Capstone

Daniel Njoroge

6/16/2021

Loading the Dataset

```
############################################################
# Create edx set, validation set (final hold-out test set)
############################################################

# Note: this process could take a couple of minutes

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)

#Adding more libraries
library(ggplot2)
library(lubridate)
library(knitr)
library(kableExtra)

# 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(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")
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)
```

Executive Summary.

The main objective of this project is to build a movie recommendation system that predicts how a user is likely to rate a movie. The analysis section explains the process and tecgniques used to handle the datasets to create predcitve models.

Introduction

The movielens project is aimed at creating recommendation system using the edx data set in the training of the algorithms and movie ratings prediction.

Analysis Method

The main methods of this paper targets to develop recommendation based on users ratings and to evaluate the system using simple baseline techniques such as Linear regression model with regularized movie and user effects using Lambda for validation.

unique userId and movieId

```
#unique users and movies in the edx dataset
edx %>%
  summarize(users = n_distinct(userId),
            movies = n_distinct(movieId),
            avg_rating = round(mean(edx$rating),2),
            uniq_genres = n_distinct(edx$genres))
```

```
##   users movies avg_rating uniq_genres
## 1 69878  10677       3.51         797
```

```
#unique users and movies in the validation dataset
validation %>%
  summarize(users = n_distinct(userId),
            movies = n_distinct(movieId),
            avg_rating = round(mean(validation$rating),2),
            uniq_genres = n_distinct(validation$genres))
```

```
##   users movies avg_rating uniq_genres
## 1 68534   9809       3.51         773
```

From the summaries we can see that the number of unique values for the userId is 69878 and for the movieId 10677 in the edx dataset. This means that there are less movies provided for ratings than users that rated them. We can thus infer that some movies get rated more than others, and some users are more active than others at rating movies.

```
#Summary of the edx and validation data frames using kable
#Function 'kable() allows users to construct complex tables and customize styles using a readable syntax
summary(edx) %>%  kable(caption = "Top rows of edx data frame") %>%
  kable_styling(font_size = 10, position = "center",
                latex_options = c("scale_down", "HOLD_position"))
```

Table 1: Top rows of edx data frame

	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	NA	NA
	3rd Qu.:53607	3rd Qu.: 3626	3rd Qu.:4.000	3rd Qu.:1.127e+09	NA	NA
	Max. :71567	Max. :65133	Max. :5.000	Max. :1.231e+09	NA	NA

```
summary(edx) %>%  kable(caption = "Top rows of validation data frame") %>%
  kable_styling(font_size = 10, position = "center",
                latex_options = c("scale_down", "HOLD_position"))
```

Table 2: Top rows of validation data frame

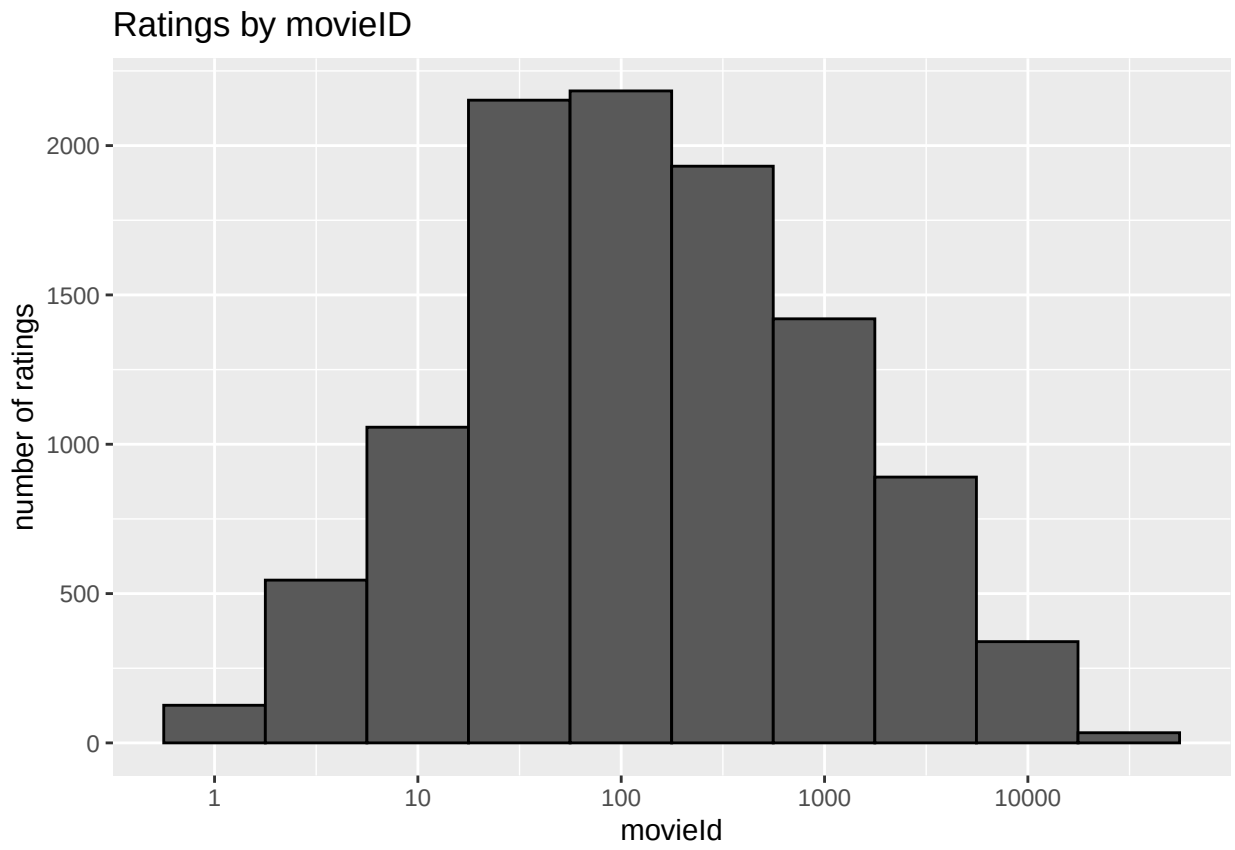
	userId	movieId	rating	timestamp	title	genres
	Min. : 1	Min. : 1	Min. :0.500	Min. :7.897e+08	Length:9000055	Length:9000055
	1st Qu.:18124	1st Qu.: 648	1st Qu.:3.000	1st Qu.:9.468e+08	Class :character	Class :character
	Median :35738	Median : 1834	Median :4.000	Median :1.035e+09	Mode :character	Mode :character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

The edx and validation datasets have six simiar variables (columns), namely ##userID##, ##movieID, ##rating##, ##timestamp##, ##title##, and ##genres##. The edx dataset has 9000055 observations while the validation dataset has 999999 observations. Now we can see that the dataset is in a tidy forma and good for further analysis.

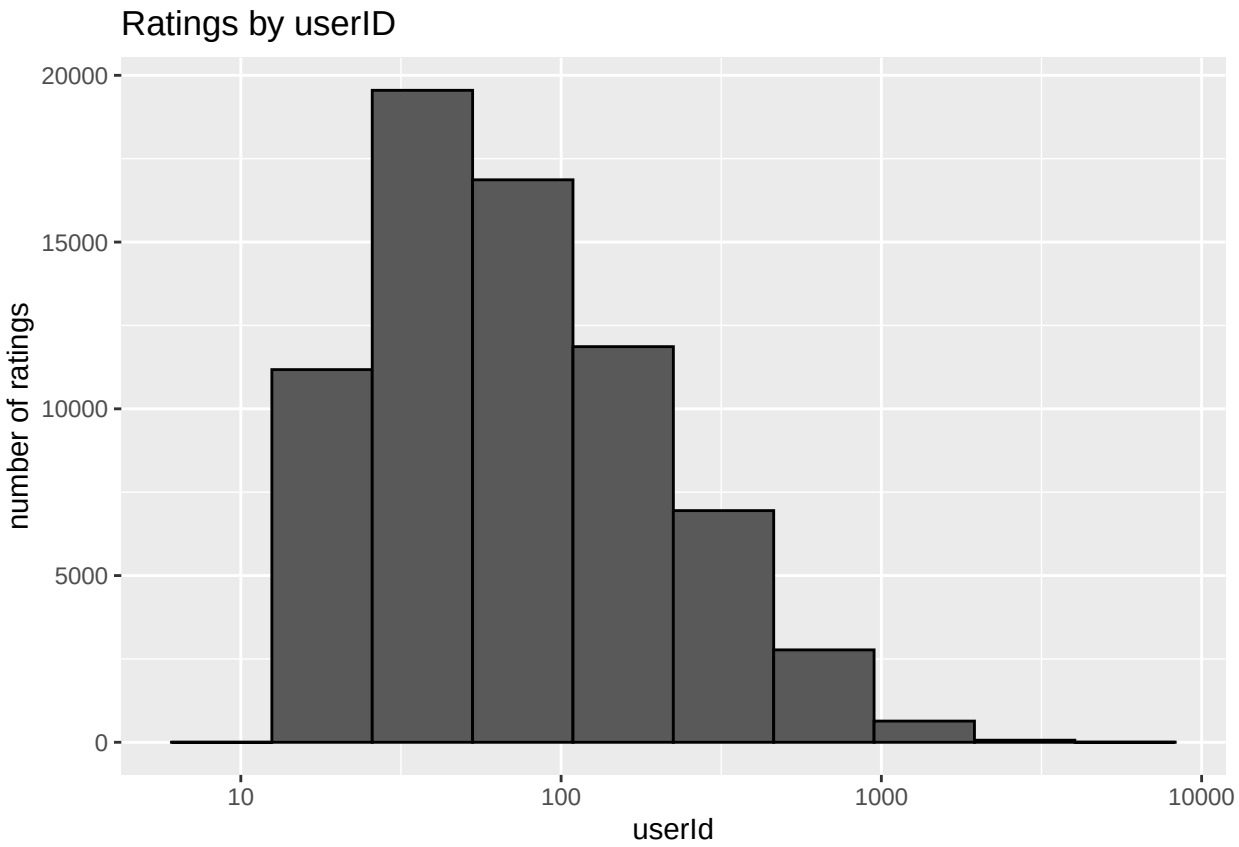
Visualization

Ratings by UserID and MovieID

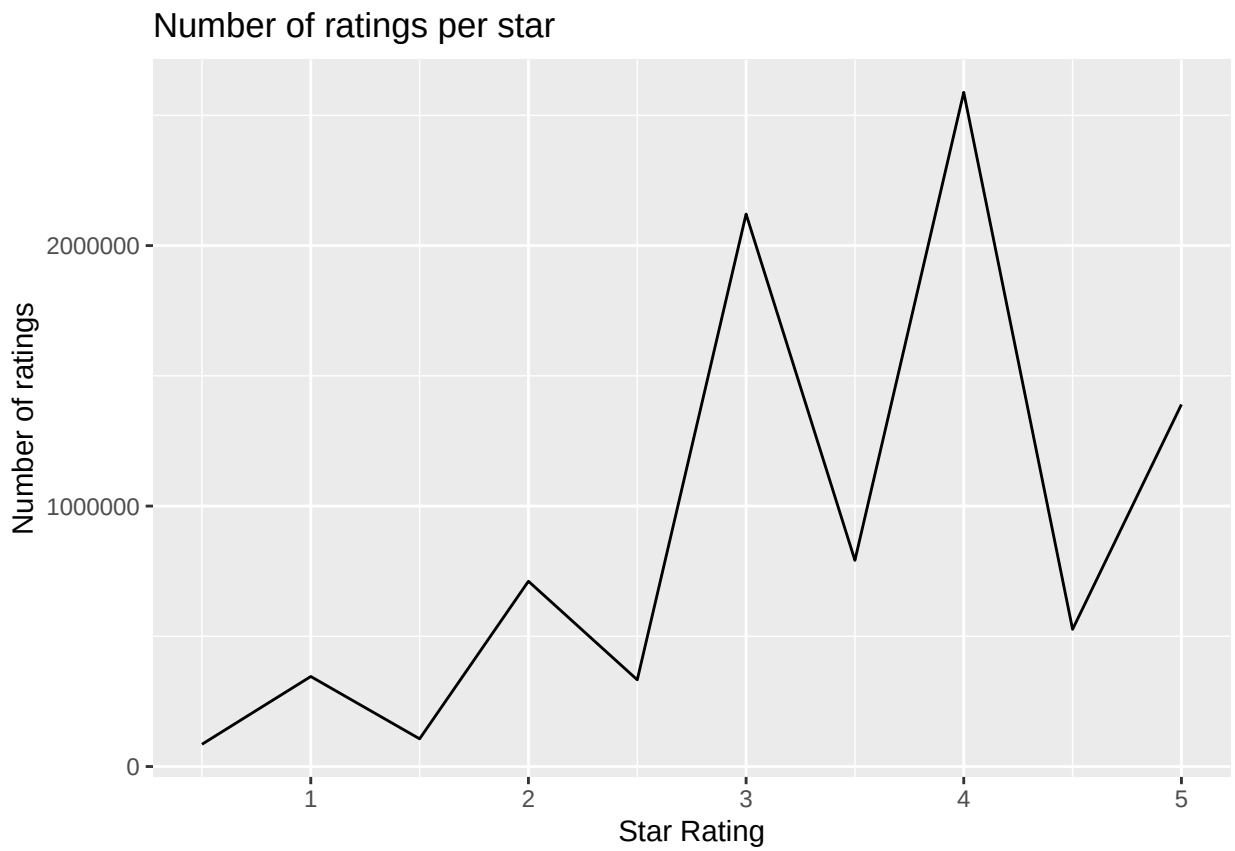
```
# Distribution of ratings by movieId
edx %>%
  count(movieId) %>%
  ggplot(aes(n)) +
  geom_histogram( bins=10, color = "black") +
  scale_x_log10() +
  ggtitle("Ratings by movieID") +
  labs(x="movieId" ,y="number of ratings")
```




```
# Distribution of ratings by userID
edx %>%
  count(userId) %>%
  ggplot(aes(n)) +
  geom_histogram( bins=10, color = "black") +
  scale_x_log10() +
  ggtitle("Ratings by userID") +
  labs(x="userId" , y="number of ratings")
```




```
#Star ratings
options(scipen = 100) # To avoid scientific notation
edx %>%
  group_by(rating) %>%
  summarize(count = n()) %>%
  ggplot(aes(x = rating, y = count)) +
  geom_line()+
  labs(x="Star Rating", y="Number of ratings") +
  ggtitle("Number of ratings per star")
```



The results above shows that 4-star and 3-star ratings are the most common followed by 5-star ratings.

```
# movie ratings by genres
memory.limit(size=56000) #expanding memory allocated to R
```

```
## [1] 56000
```

```
edx %>% separate_rows(genres, sep = "\\|") %>%
  group_by(genres) %>%
  summarize(count = n()) %>%
  arrange(desc(count))
```

```
## # A tibble: 20 x 2
##    genres               count
##    <chr>                <int>
##  1 Drama              3910127
##  2 Comedy             3540930
##  3 Action             2560545
##  4 Thriller           2325899
##  5 Adventure          1908892
##  6 Romance            1712100
##  7 Sci-Fi             1341183
##  8 Crime              1327715
##  9 Fantasy             925637
## 10 Children            737994
```

```
## 11 Horror              691485
## 12 Mystery             568332
## 13 War                 511147
## 14 Animation           467168
## 15 Musical             433080
## 16 Western             189394
## 17 Film-Noir           118541
## 18 Documentary          93066
## 19 IMAX                  8181
## 20 (no genres listed)       7
```

Data Transformation

```
#transforming userID and movieId as factors for analysis
edx$userId <- as.factor(edx$userId)
edx$movieId <- as.factor(edx$movieId)
edx$genres <- as.factor(edx$genres)
edx$timestamp <- as.POSIXct(edx$timestamp, origin = "1970-01-01") # Convert timestamp to POSIXct.

edx <- edx %>%
  mutate(title = str_trim(title)) %>%
  extract(title, c("title_tmp", "year"), # extracting the release year of the movie from the title colu
          regex = "^(.*) \\(([0-9 \\-]*)\\)$",
          remove = F) %>%
  mutate(year = if_else(str_length(year) > 4, #createing year column.
                        as.integer(str_split(year, "-",
                                             simplify = T)[1]),
                        as.integer(year))) %>%
  mutate(title = if_else(is.na(title_tmp), title, title_tmp)) %>%
  select(-title_tmp)  %>%
  mutate(genres = if_else(genres == "(no genres listed)",
                          `is.na<-`(genres), genres))

edx <- edx %>% mutate(year_rate = year(timestamp)) # creating a column for the year the movie was rated

# It extracts the year that the rate was given by the user.
edx <- edx %>% select(-title, -timestamp) # removing the title and the timsestamp columns (optional)
edx$genres <- as.factor(edx$genres)

head(edx)
```

```
##    userId movieId rating year                       genres year_rate
## 1:      1     122      5 1992               Comedy|Romance      1996
## 2:      1     185      5 1995         Action|Crime|Thriller      1996
## 3:      1     292      5 1995  Action|Drama|Sci-Fi|Thriller      1996
## 4:      1     316      5 1994       Action|Adventure|Sci-Fi      1996
## 5:      1     329      5 1994 Action|Adventure|Drama|Sci-Fi      1996
## 6:      1     355      5 1994       Children|Comedy|Fantasy      1996
```

```
summary(edx$rating) #edx rating summary
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
```

```
##   0.500   3.000   4.000   3.512   4.000   5.000
```

```
#genres rating using the mean
edx %>% group_by(genres) %>% summarize(avg_rating = mean(rating)) %>% arrange(desc(avg_rating))
```

```
## # A tibble: 797 x 2
##    genres                                    avg_rating
##    <fct>                                          <dbl>
##  1 Animation|IMAX|Sci-Fi                           4.71
##  2 Drama|Film-Noir|Romance                         4.30
##  3 Action|Crime|Drama|IMAX                         4.30
##  4 Animation|Children|Comedy|Crime                 4.28
##  5 Film-Noir|Mystery                               4.24
##  6 Crime|Film-Noir|Mystery                         4.22
##  7 Film-Noir|Romance|Thriller                      4.22
##  8 Crime|Film-Noir|Thriller                        4.21
##  9 Crime|Mystery|Thriller                          4.20
## 10 Action|Adventure|Comedy|Fantasy|Romance         4.20
## # ... with 787 more rows
```

The Model

```
edx <- edx %>% select(userId, movieId, rating) #Three parameter of interest
# Create the index
test_index <- createDataPartition(edx$rating, times = 1, p = .2, list = F)
#Creating the train and test sets
train <- edx[-test_index, ] # The train set
test <- edx[test_index, ] # The test set
test <- test %>% # join movieId and usersId since they appear in both sets
  semi_join(train, by = "movieId") %>%
  semi_join(train, by = "userId")
```

The Baseline Model

This model calculates the mean rating and is the baseline model for this purpose. Improvements on the RMSEs will be relative to this model.

```
#generating the model using the mean only
mu_0 <- mean(train$rating) # Mean rating on train set
RMSE_0 <- RMSE(test$rating, mu_0) # RMSE in the test set.
RMSE_0
```

```
## [1] 1.060704
```

The test dataset

We are going to generate the next prediction model using the movie effect and the user effect.

```
# obtaining prediction using the mean from movie and user effect
mu <- mean(train$rating)

m_avgs <- train %>%
  group_by(movieId) %>%
  summarize(mi = mean(rating - mu)) #movie effect

u_avgs <- test %>%
  left_join(m_avgs, by = "movieId") %>%
  group_by(userId) %>%
  summarize(ui = mean(rating - mu -mi)) #user effect

predicted_ratings <- test %>%
  left_join(m_avgs, by = "movieId") %>%
  left_join(u_avgs, by = "userId") %>%
  mutate(pred = mu +mi +ui) %>% .$pred

RMSE_1 <- RMSE(predicted_ratings, test$rating)
RMSE_1
```

```
## [1] 0.8435874
```

There is an improvement from the baseline model.

The validation dataset

We are now going to make predictions using the validation dataset. The method will be similar to the one applied on the test dataset.

```
#prediction using the validation dataset
validation <- validation %>% select(userId, movieId, rating) #we are interested in userId, movieId, & r
#treating userId & movieId as factors
validation$userId <- as.factor(validation$userId)
validation$movieId <- as.factor(validation$movieId)
```

The complete.cases function is used to identify complete rows of a data frame. i.e. rows without NAs.

```
validation <- validation[complete.cases(validation), ]
```

```
#The prediction
mu <- mean(train$rating)

m_avgs <- train %>%
  group_by(movieId) %>%
  summarize(mi = mean(rating - mu))

u_avgs <- test %>%
  left_join(m_avgs, by = "movieId") %>%
  group_by(userId) %>%
  summarize(ui = mean(rating - mu -mi))
```

```
predicted_ratings <- test %>%
  left_join(m_avgs, by = "movieId") %>%
  left_join(u_avgs, by = "userId") %>%
  mutate(pred = mu +mi +ui) %>% .$pred


predicted_val <- validation %>%
  left_join(m_avgs, by = "movieId") %>%
  left_join(u_avgs, by = "userId") %>%
  mutate(pred = mu +mi +ui) %>% .$pred

RMSE_VAL <- RMSE(predicted_val, validation$rating, na.rm = T)
RMSE_VAL
```

```
## [1] 0.8821153
```

The improvement on this model is not that good. We are going to perform regularization using movie effect and user effect as the predictors.

Regularization

With regularization, we evaluate different values for lambda, which a tuning parameter and picking the optimum value wich minimises the RMSE.

The Test Data

```
#regularizing test data
lambda_values <- seq(0, 10, .25)
t_RMSE_reg <- sapply(lambda_values, function(l){

  mu <- mean(train$rating)

 mi <- train %>%
    group_by(movieId) %>%
    summarize(mi = sum(rating - mu)/(n()+l)) #movie effect

 ui <- train %>%
    left_join(mi, by="movieId") %>%
    group_by(userId) %>%
    summarize(ui = sum(rating -mi - mu)/(n()+l)) #user effect

  predicted_ratings <- test %>%
    left_join(mi, by = "movieId") %>%
    left_join(ui, by = "userId") %>%
    mutate(pred = mu +mi +ui) %>% .$pred

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

```
t_lambda <- lambda_values[which.min(t_RMSE_reg)]
t_lambda # Lambda minimizing the RMSE
```

```
## [1] 4.75
```

The Validation Data

```
#regularization on validation dataset
val_RMSE_reg <- sapply(lambda_values, function(l){

  mu <- mean(train$rating)

 mi <- train %>%
    group_by(movieId) %>%
    summarize(mi = sum(rating - mu)/(n()+l)) #movie effect

 ui <- train %>%
    left_join(mi, by="movieId") %>%
    group_by(userId) %>%
    summarize(ui = sum(rating -mi - mu)/(n()+l)) #user effect

  predicted_val_reg <- validation %>%
    left_join(mi, by = "movieId") %>%
    left_join(ui, by = "userId") %>%
    mutate(pred = mu +mi +ui) %>% .$pred

  return(RMSE(predicted_val_reg, validation$rating, na.rm = T))
})
```

```
val_lambda <- lambda_values[which.min(val_RMSE_reg)]
val_lambda #  Lambda minimizing the RMSE
```

```
## [1] 5
```

```
min_rmse <- min(val_RMSE_reg) # Best RMSE
min_rmse
```

```
## [1] 0.8656225
```

Results

The regularization gets down the RMSE's value to 0.8656225. The model evaluation performance through the Root Mean Square Error (RMSE) shows that the model with regularized effects on movie and users is teh appropriate systems to predict ratings on the validation set.

Conclusion

The regularized approach gives us the lowest RMSE and thus the optimal model to consider for this kind of a project. Future improvements to this models can be achieved by adding predictors such as users gender and profession. These two largely affect how the users watch and rate movies. The biggest challenges working on this project is the size of the dataset. Some approaches took very long to execute and had to go for ones that are less time consuming and less on memory demand.