MovieLens Project

By: Mateusz Faltyn

Introduction

The MovieLens dataset was originally collected by GroupLens Research, a research team located in the Department of Computer Science and Engineering at the University of Minnesota. The dataset was collected in order to obtain research data on personal movie recommendations.

There are 9000055 rows, and 6 columns in the MovieLens dataset. In the edx dataset, there are 10677 movies, with a total of 69878 users.

The goal of the project was to compare four supervised machine learning models - one that uses just the mean, one that uses the movie effect, one that uses the movie effect (regularized), and one that uses both the movie and user effect (regularized) - to see which model has the lowest root mean square error (RSME).

Here are the key steps that were performed in order of completion: package installation and library loading, downloading the data, creating the dataset, creating the validation set, Dataset exploration and visualization, creating the predictive model, mean method, movie effect method, movie effect – regularized, movie and user effects – regularized.

The movie and user effect (regularized) model proved to have the lowest RSME (0.8646166).

Methods

Package Installation and Library Loading The packages that are necessary for this analysis are the following: tidyverse, caret, gridExtra, lubridate, knitr, and e1071.

```
##### Package Installation

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

```
## Loading required package: tidyverse
## -- Attaching packages ------------------------------- tidyverse 1.2.1 --
```

```
## v ggplot2 3.2.1     v purrr   0.3.2
## v tibble  2.1.3     v dplyr   0.8.3
## v tidyr   0.8.3     v stringr 1.4.0
## v readr   1.3.1     v forcats 0.4.0
```

```
## -- Conflicts ---------------------------------- tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```r
if(!require(caret)) install.packages("caret", repos = "http://cra
n.r-project.org")
```

```
## Loading required package: caret
```

```
## Loading required package: lattice
```

```
## 
## Attaching package: 'caret'
```

```
## The following object is masked from 'package:purrr':
## 
##     lift
```

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

```
## Loading required package: gridExtra
```

```
##
## Attaching package: 'gridExtra'
```

```
## The following object is masked from 'package:dplyr':
##
##     combine
```

```
if(!require(lubridate)) install.packages("lubridate", repos = "http://cra
n.r-project.org")
```

```
## Loading required package: lubridate
```

```
##
## Attaching package: 'lubridate'
```

```
## The following object is masked from 'package:base':
##
##     date
```

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

```
## Loading required package: knitr
```

```
if(!require(e1071)) install.packages("e1071", repos = "http://cra
n.r-project.org")
```

```
## Loading required package: e1071
```

```
##### Loading Libraries

library(tidyverse)
library(caret)
library(gridExtra)
library(lubridate)
library(knitr)
library(e1071)
```

Downloading the Data Below you will find the link to downloading the dataset.

```
##### Downloading the MovieLens Data

dl <- tempfile()
download.file("http://files.grouplens.org/datasets/movielens/ml-10m.zip", dl)

ratings <- read.table(text = gsub("::", "\t", readLines(unzip(dl, "ml-10M100K/ratings.dat"))),
     col.names = c("userId", "movieId", "rating", "timestamp"))
```

Creating the Dataset

```
##### Creating the Dataset

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(levels(movieId))[movieId], title = as.character(t
itle), genres = as.character(genres))

movielens <- left_join(ratings, movies, by = "movieId")
```

Ceating the Validation Set

```
##### Creating the Validation Set: 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 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)
```

```
## Joining, by = c("userId", "movieId", "rating", "timestamp", "title", "genres")
```

```
edx <- rbind(edx, removed)
```

Dataset Exploration and Visualization

```
##### Dataset Exploration: Inspecting the Dimensions of EDX

head(edx)
```

```
##   userId movieId rating timestamp                         title
## 1      1     122      5 838985046              Boomerang (1992)
## 2      1     185      5 838983525               Net, The (1995)
## 3      1     231      5 838983392          Dumb & Dumber (1994)
## 4      1     292      5 838983421               Outbreak (1995)
## 5      1     316      5 838983392               Stargate (1994)
## 6      1     329      5 838983392 Star Trek: Generations (1994)
##                          genres
## 1                Comedy|Romance
## 2         Action|Crime|Thriller
## 3                        Comedy
## 4  Action|Drama|Sci-Fi|Thriller
## 5       Action|Adventure|Sci-Fi
## 6 Action|Adventure|Drama|Sci-Fi
```

```
cat("The edx dataset has", nrow(edx), "rows and", ncol(edx), "columns.\n")
```

```
## The edx dataset has 9000061 rows and 6 columns.
```

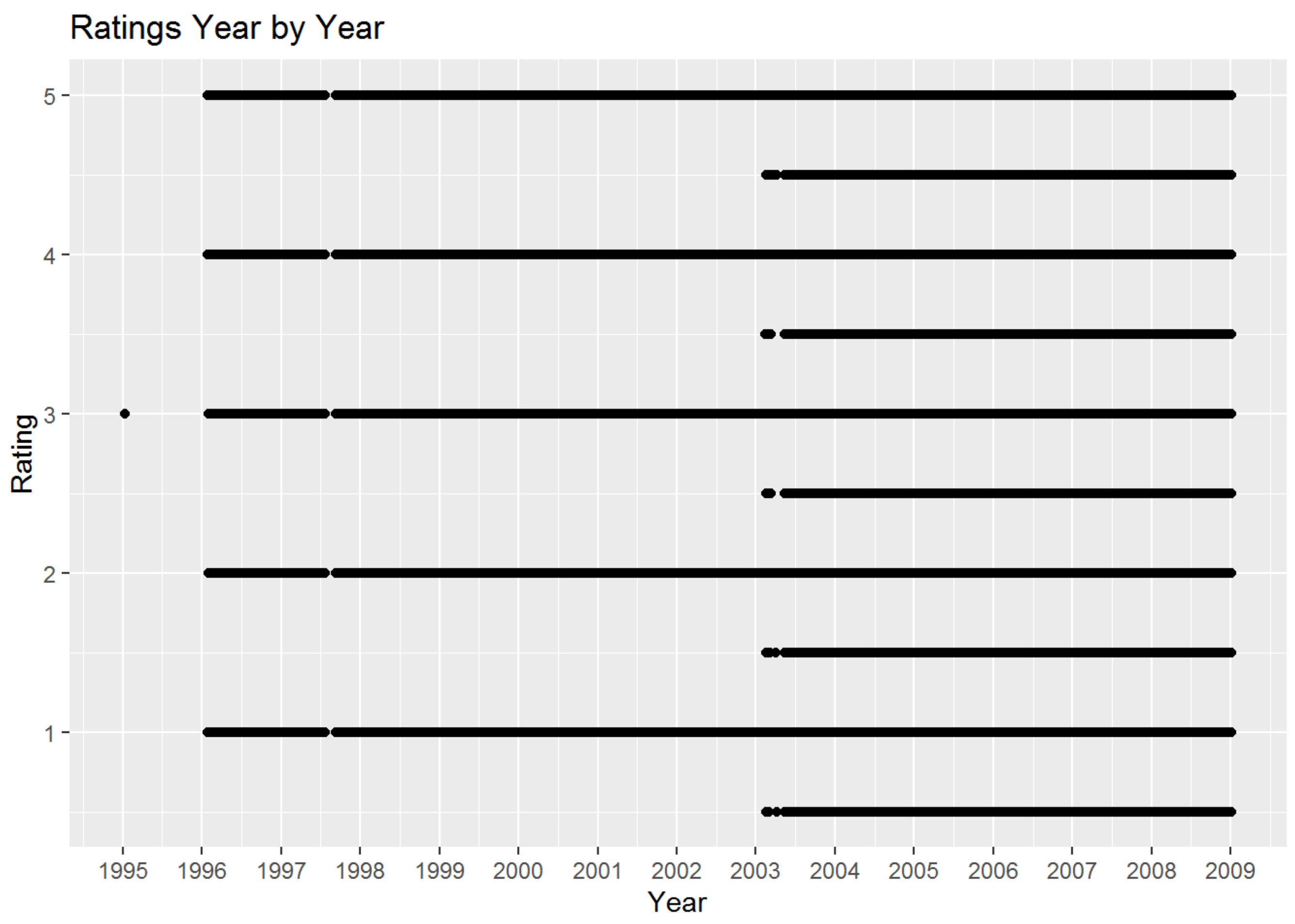
```
cat("There are", n_distinct(edx$userId), "different users and", n_distinct(edx$movieId), "different movies in the
 edx dataset.")
```

```
## There are 69878 different users and 10677 different movies in the edx dataset.
```

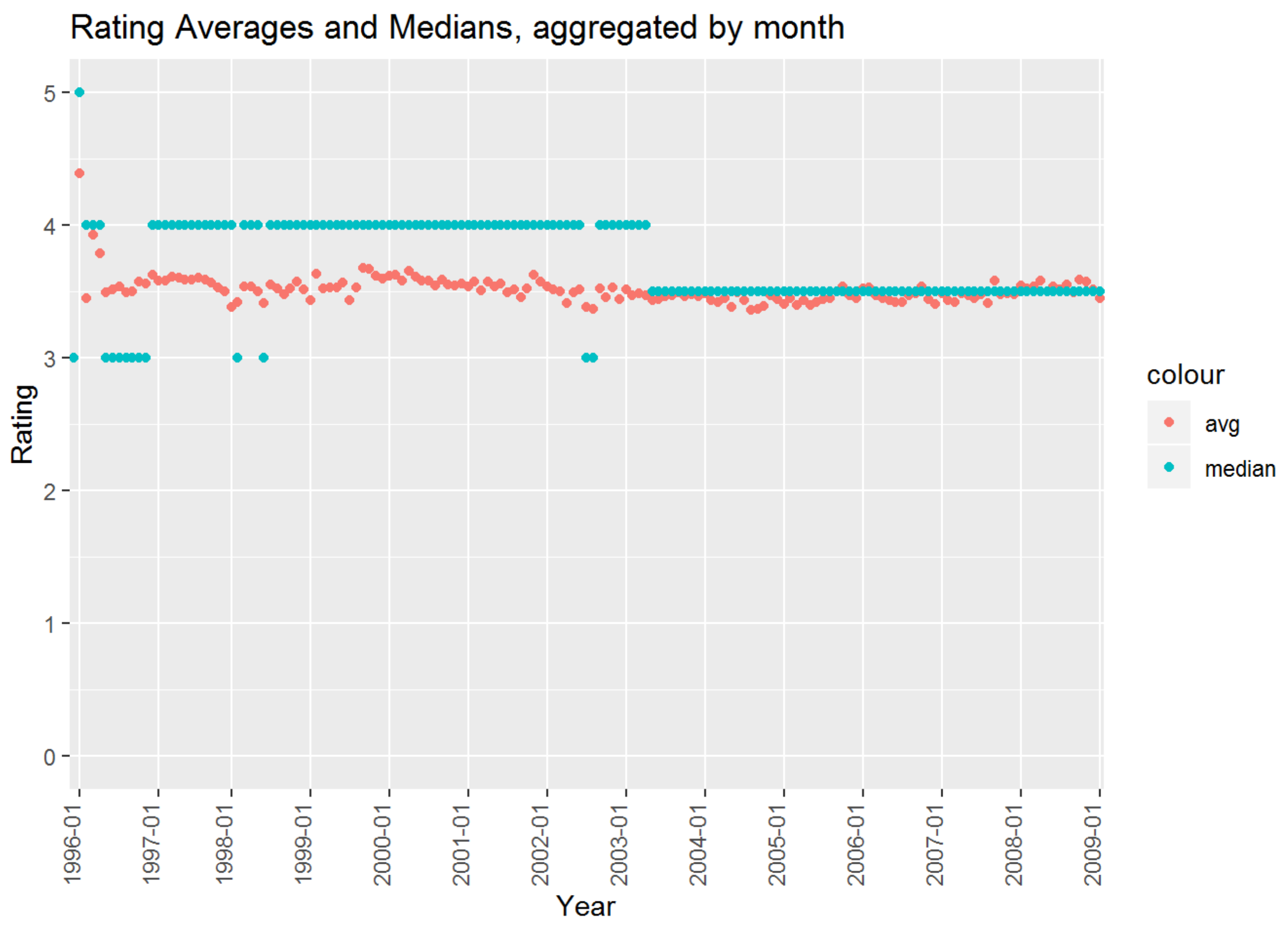
```
# Check if edx has missing values

any(is.na(edx))
```

```
## [1] FALSE
```

```
# What are the ratings year by year?

edx_year_rating <- edx %>%
  transform (date = as.Date(as.POSIXlt(timestamp, origin = "1970-01-01", format = "%Y-%m-%d"), format = "%Y-%m-%
d")) %>%
  mutate (year_month = format(as.Date(date), "%Y-%m"))

ggplot(edx_year_rating) +
  geom_point(aes(x = date, y = rating)) +
  scale_x_date(date_labels = "%Y", date_breaks  = "1 year") +
  labs(title = "Ratings Year by Year", x = "Year", y = "Rating")
```



```r
# What are the rating averages and medians year by year?

edx_yearmonth_rating <- edx_year_rating %>%
  group_by(year_month) %>%
  summarize(avg = mean(rating), median = median(rating))

ggplot(edx_yearmonth_rating) +
  geom_point(aes(x = year_month, y = avg, colour = "avg")) +
  geom_point(aes(x = year_month, y = median, colour = "median")) +
  ylim(0, 5) +
  scale_x_discrete(breaks = c("1996-01", "1997-01", "1997-01", "1998-01", "1999-01",
                              "2000-01", "2001-01", "2002-01", "2003-01", "2004-01",
                              "2005-01", "2006-01", "2007-01", "2008-01", "2009-01")) +
  theme(axis.text.x = element_text(angle = 90, hjust = 1, vjust = 0)) +
  labs(title = "Rating Averages and Medians, aggregated by month ", x = "Year", y = "Rating")
```

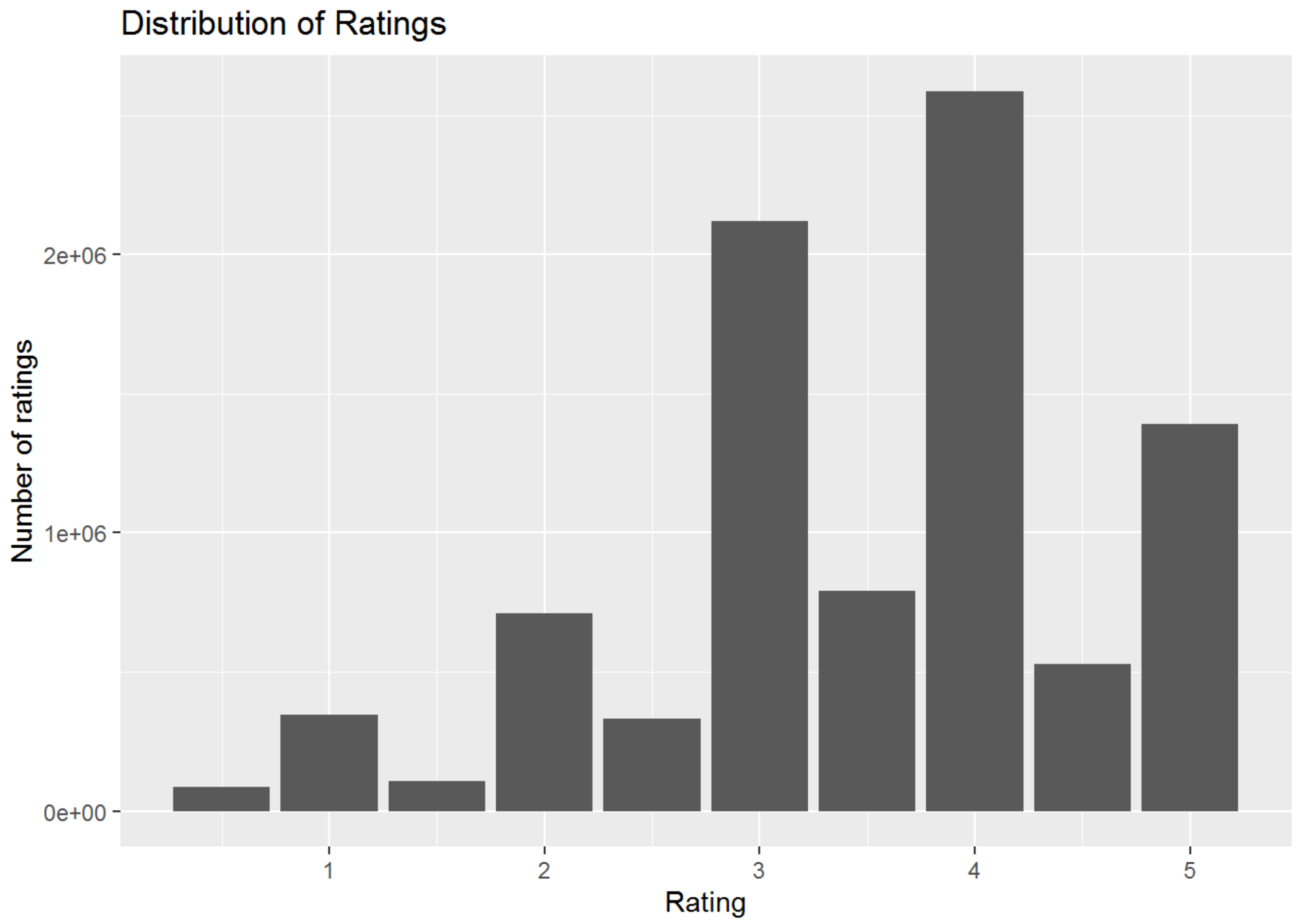


```
##### Dataset Exploration: Ratings
```

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

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
##   0.500   3.000   4.000   3.512   4.000   5.000
```

```
# Distribution of ratings
ggplot(data = edx, aes(x = rating)) +
  geom_bar() +
  labs(title = "Distribution of Ratings", x = "Rating", y = "Number of ratings")
```



```
##### Dataset Exploration: Movies

edx_movies <- edx %>%
  group_by(movieId) %>%
  summarize(count = n()) %>%
  arrange(desc(count))

summary(edx_movies$count)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
##     1.0    30.0   122.0   842.9   563.0 31336.0 
```

```
##### Dataset Exploration: Users

edx_users <- edx %>%
  group_by(userId) %>%
  summarize(count = n()) %>%
  arrange(desc(count))

summary(edx_users$count)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
##    13.0    32.0    62.0   128.8   140.0  6637.0 
```

Creating the Predictive Model

```
##### Creating the Predictive Model

set.seed(699)
test_index <- createDataPartition(y = edx$rating, times = 1, p = 0.2, list = FALSE)
train_set <- edx[-test_index,]
test_set <- edx[test_index,]
# Use semi_join() to ensure that all users and movies in the test set are also in the training set
test_set <- test_set %>% semi_join(train_set, by = "movieId") %>% semi_join(train_set, by = "userId")
```

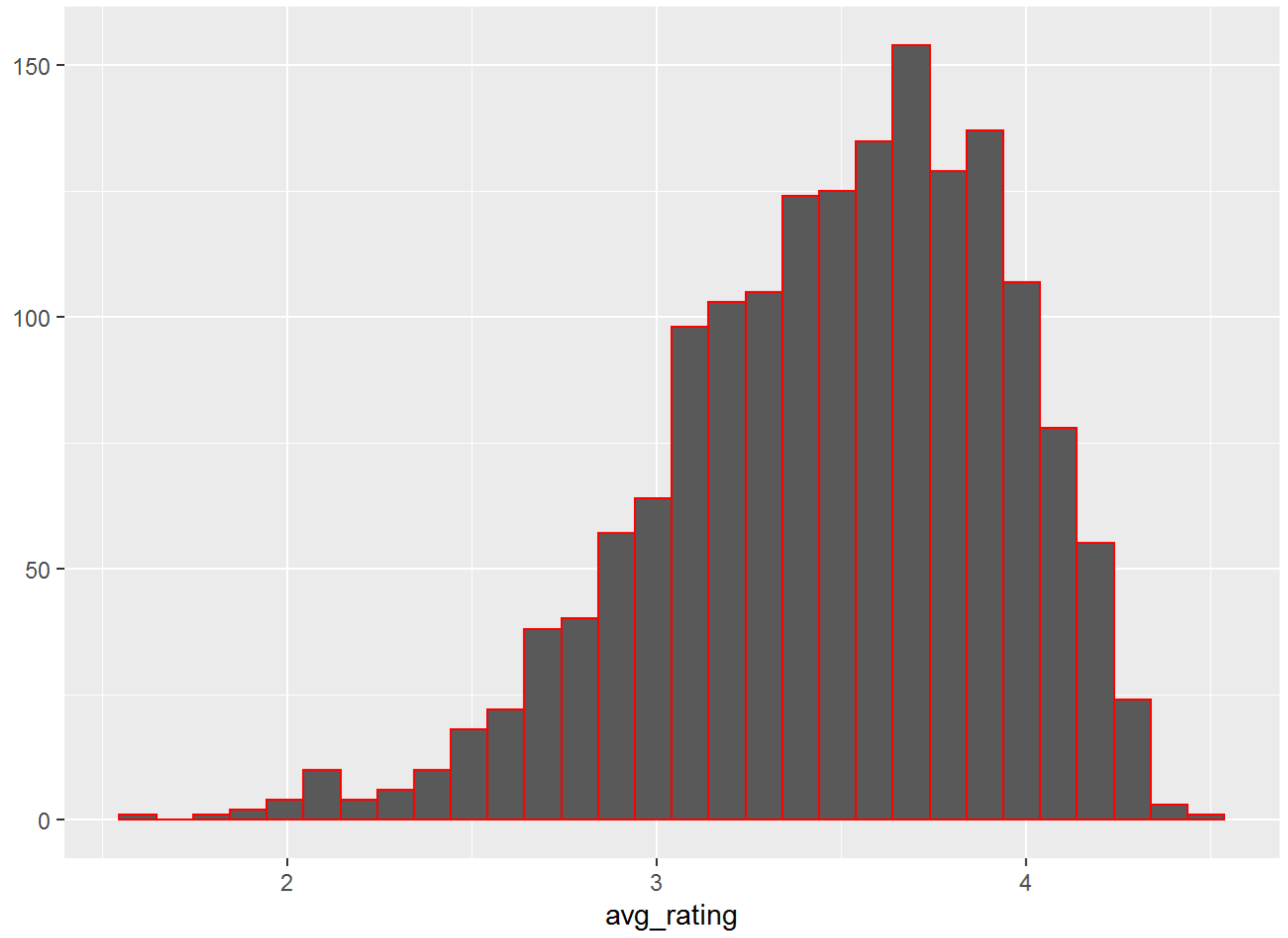
Mean Method

```
##### Mean Method

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

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

cat("The average rating of all movies across all users is:", mu)
```

```
## The average rating of all movies across all users is: 3.512316
```

```
train_set %>% group_by(movieId) %>% filter(n()>=1000) %>% summarize(avg_rating = mean(rating)) %>% qplot(avg_rati
ng, geom = "histogram", color = I("red"), bins=30, data = .)
```



```
predictions <- rep(mu, nrow(test_set))
naive_rmse <- RMSE(test_set$rating, predictions)
cat("The RMSE with mean method is:", naive_rmse)
```

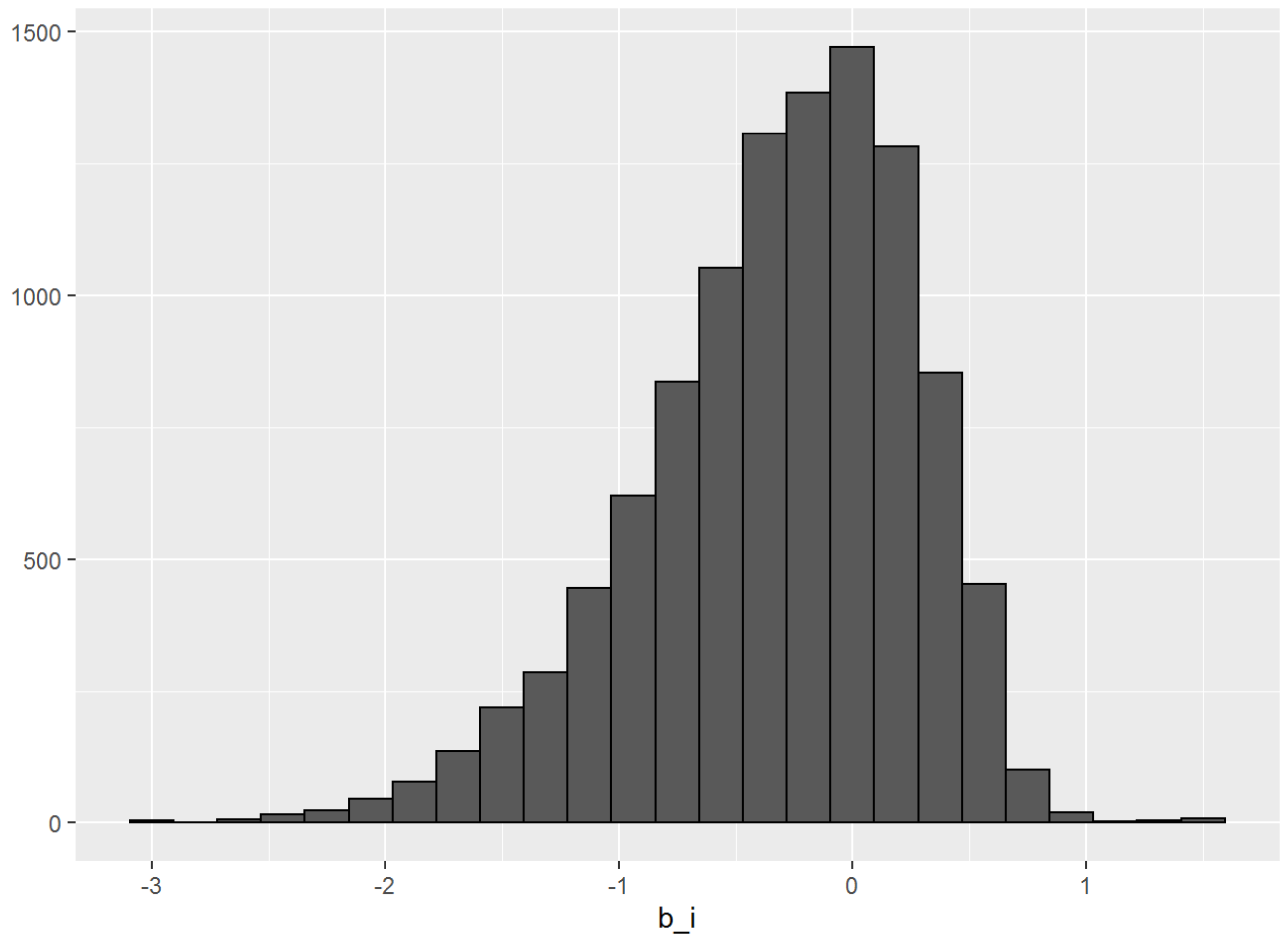
```
## The RMSE with mean method is: 1.059289
```

Movie Effect

```
##### Movie Effect Method

rmse_results <- data_frame(method = "Mean", RMSE = naive_rmse)
```

```
## Warning: `data_frame()` is deprecated, use `tibble()`.
## This warning is displayed once per session.
```

```
movie_means <- train_set %>% group_by(movieId) %>% summarize(b_i = mean(rating - mu))

qplot(b_i, geom = "histogram", color = I("black"), bins=25, data = movie_means)
```



```
joined <- test_set %>% left_join(movie_means, by='movieId')

predicted_ratings <- mu + joined$b_i

model1_rmse <- RMSE(predicted_ratings, test_set$rating)

rmse_results <- bind_rows(rmse_results, data_frame(method = "Movie effect model", RMSE = model1_rmse ))

rmse_results %>% kable
```

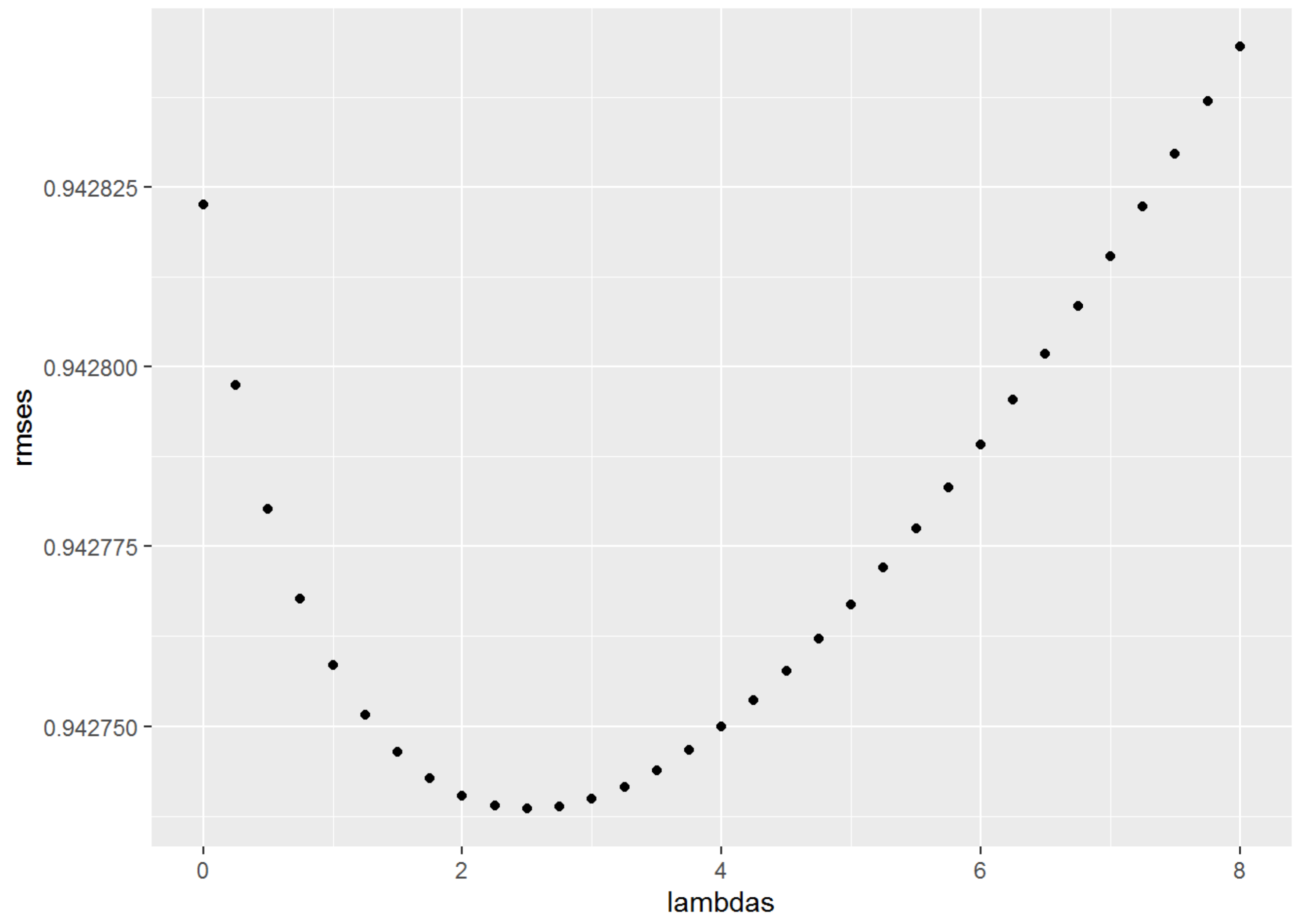
method	RMSE
Mean	1.0592889
Movie effect model	0.9428225

Movie Effect – Regularized

```
##### Movie Effect - Regularized

lambdas <- seq(0, 8, 0.25)

tmp <- train_set %>%
  group_by(movieId) %>%
  summarize(sum = sum(rating - mu), n_i = n())

rmses <- sapply(lambdas, function(l){
  joined <- test_set %>%
    left_join(tmp, by='movieId') %>%
    mutate(b_i = sum/(n_i+l))
  predicted_ratings <- mu + joined$b_i
  return(RMSE(predicted_ratings, test_set$rating))
})

cat("The best lambda (which minimizes the RMSE) for the movie effec
t is:", lambdas[which.min(rmses)])
```

```
## The best lambda (which minimizes the RMSE) for the movie effec
## t is: 2.5
```

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



```
lambda <- 2.75
movie_reg_means <- train_set %>% group_by(movieId) %>% summarize(b_i = sum(rating - mu)/(n()+lambda), n_i = n())

joined <- test_set %>% left_join(movie_reg_means, by='movieId') %>% replace_na(list(b_i=0))

predicted_ratings <- mu + joined$b_i

model1_reg_rmse <- RMSE(predicted_ratings, test_set$rating)
```

```
rmse_results <- bind_rows(rmse_results, data_frame(method = "Movie Effect - Regularized", RMSE = model1_reg_rmse
 ))

rmse_results %>% kable
```

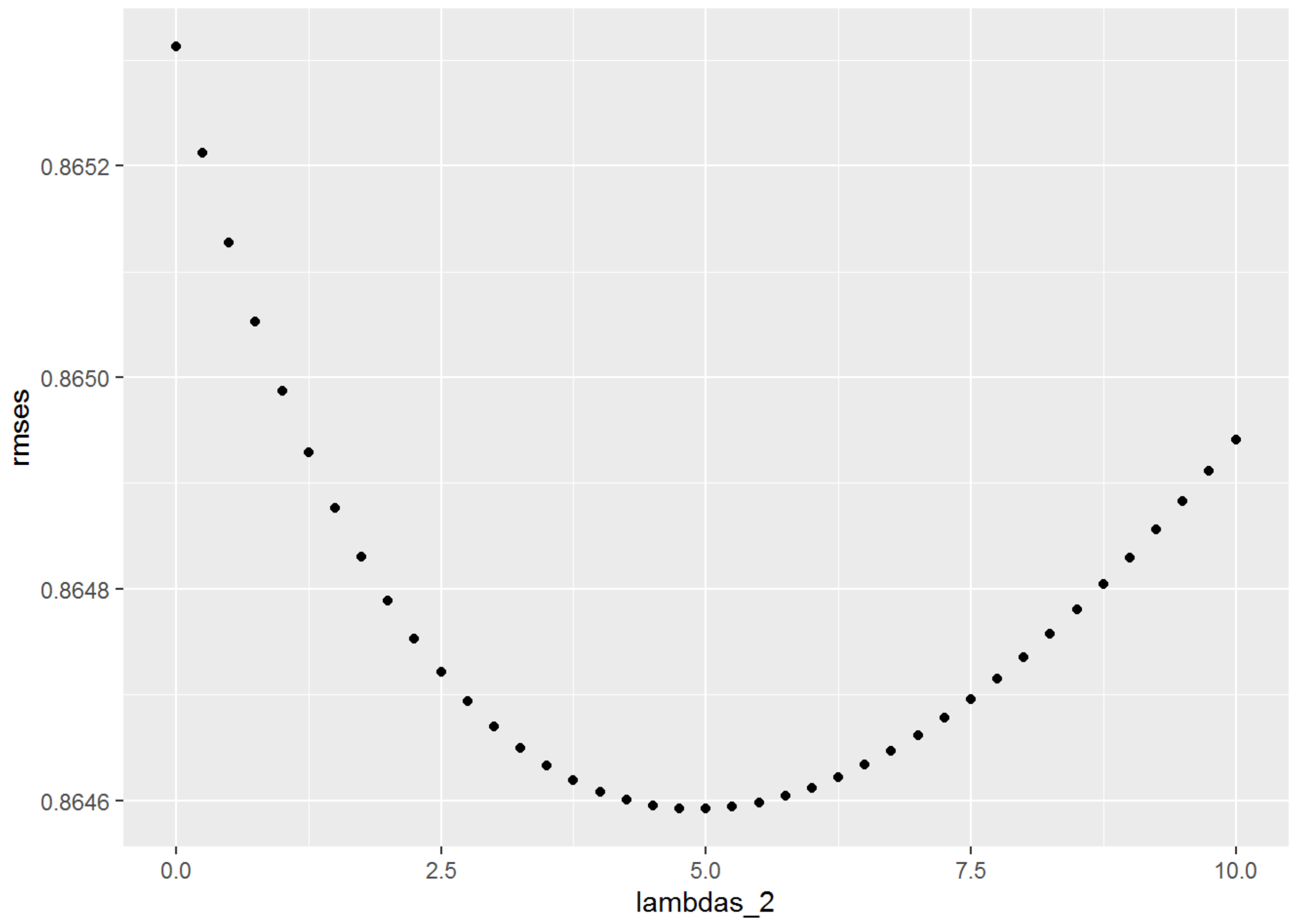
method	RMSE
Mean	1.0592889
Movie effect model	0.9428225
Movie Effect - Regularized	0.9427388

Movie and User Effects – Regularized

```
##### Movie and User Effects - Regularized

lambdas_2 <- seq(0, 10, 0.25)

rmses <- sapply(lambdas_2, function(l){mu <- mean(train_set$rating)

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

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

predicted_ratings <- test_set %>% left_join(b_i, by = "movieId") %>% left_join(b_u, by = "userId") %>% mutate(pre
d = mu + b_i + b_u) %>% .$pred
return(RMSE(predicted_ratings, test_set$rating))
})

cat("The best lambda_2 (which minimizes the RMSE) for the user and
movie effects is", lambdas_2[which.min(rmses)])
```

```
## The best lambda_2 (which minimizes the RMSE) for the user and
## movie effects is 5
```

```
qplot(lambdas_2, rmses)
```



```
lambda_2 <- 5
user_reg_means <- train_set %>% left_join(movie_reg_means) %>% mutate(resids = rating - mu - b_i) %>% group_by(us
erId) %>% summarize(b_u = sum(resids)/(n()+lambda_2))
```

```
## Joining, by = "movieId"
```

```
joined <- test_set %>% left_join(movie_reg_means, by='movieId') %>% left_join(user_reg_means, by='userId') %>% re
place_na(list(b_i=0, b_u=0))

predicted_ratings <- mu + joined$b_i + joined$b_u

model2_reg_rmse <- RMSE(predicted_ratings, test_set$rating)

rmse_results <- bind_rows(rmse_results, data_frame(method = "Regularized movie and user effects model, lambda2 =
 5", RMSE = model2_reg_rmse ))
```

Results

Model Comparison Out of the four models, the movie and user effect (regularized) model had the lowest RSME (0.8646166).

```
rmse_results %>% kable
```

method	RMSE
Mean	1.0592889
Movie effect model	0.9428225
Movie Effect - Regularized	0.9427388
Regularized movie and user effects model, lambda2 = 5	0.8646166

Conclusion

The goal of the project was to compare four supervised machine learning models - one that uses just the mean, one that uses the movie effect, one that uses the movie effect (regularized), and one that uses both the movie and user effect (regularized). The movie and user effect (regularized) model proved to have the lowest RSME (0.8646166).