MovieLens

Irina Shevchuk

2024-11-12

HarvardX PH125.9x Data Science: Capstone

MovieLens Project

```
##########################################################
# Create edx and final_holdout_test sets
##########################################################

# 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")

library(tidyverse)
library(caret)

# MovieLens 10M dataset:
# https://grouplens.org/datasets/movielens/10m/
# http://files.grouplens.org/datasets/movielens/ml-10m.zip

options(timeout = 120)

dl <- "ml-10M100K.zip"
if(!file.exists(dl))
  download.file("https://files.grouplens.org/datasets/movielens/ml-10m.zip", dl)

ratings_file <- "ml-10M100K/ratings.dat"
if(!file.exists(ratings_file))
  unzip(dl, ratings_file)

movies_file <- "ml-10M100K/movies.dat"
if(!file.exists(movies_file))
  unzip(dl, movies_file)

ratings <- as.data.frame(str_split(read_lines(ratings_file), fixed("::"), simplify = TRUE),
                         stringsAsFactors = FALSE)
colnames(ratings) <- c("userId", "movieId", "rating", "timestamp")
ratings <- ratings %>%
  mutate(userId = as.integer(userId),
         movieId = as.integer(movieId),
```

```
         rating = as.numeric(rating),
         timestamp = as.integer(timestamp))

movies <- as.data.frame(str_split(read_lines(movies_file), fixed("::"), simplify = TRUE),
                        stringsAsFactors = FALSE)
colnames(movies) <- c("movieId", "title", "genres")
movies <- movies %>%
  mutate(movieId = as.integer(movieId))

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

# Final hold-out test set will be 10% of MovieLens data
set.seed(1, sample.kind="Rounding") # if using R 3.6 or later
# set.seed(1) # if using R 3.5 or earlier
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 final hold-out test set are also in edx set
final_holdout_test <- temp %>%
  semi_join(edx, by = "movieId") %>%
  semi_join(edx, by = "userId")

# Add rows removed from final hold-out test set back into edx set
removed <- anti_join(temp, final_holdout_test)
edx <- rbind(edx, removed)

rm(dl, ratings, movies, test_index, temp, movielens, removed)
```

Data set research

Counting rows and columns are there in the edx dataset

```
dim(edx)
```

```
## [1] 9000055       6
```

Display the first n rows present in the input data frame

```
head(edx)
```

```
##   userId movieId rating timestamp                         title
## 1      1     122      5 838985046              Boomerang (1992)
## 2      1     185      5 838983525               Net, The (1995)
## 4      1     292      5 838983421               Outbreak (1995)
## 5      1     316      5 838983392               Stargate (1994)
## 6      1     329      5 838983392 Star Trek: Generations (1994)
## 7      1     355      5 838984474       Flintstones, The (1994)
```

```
##                          genres
## 1                Comedy|Romance
## 2         Action|Crime|Thriller
## 4  Action|Drama|Sci-Fi|Thriller
## 5       Action|Adventure|Sci-Fi
## 6 Action|Adventure|Drama|Sci-Fi
## 7       Children|Comedy|Fantasy
```

Displays structures

```
str(edx)
```

```
## 'data.frame':    9000055 obs. of  6 variables:
##  $ userId   : int  1 1 1 1 1 1 1 1 1 1 ...
##  $ movieId  : int  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
```

Display the minimum, maximum, mean, median, and 1st and 3rd quartiles for a numerical vector

```
summary(edx)
```

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

Display unique users & movie

```
edx %>% summarize(n_users = n_distinct(userId), n_movies = n_distinct(movieId))
```

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

Loading the library

```
library(ggplot2)
```

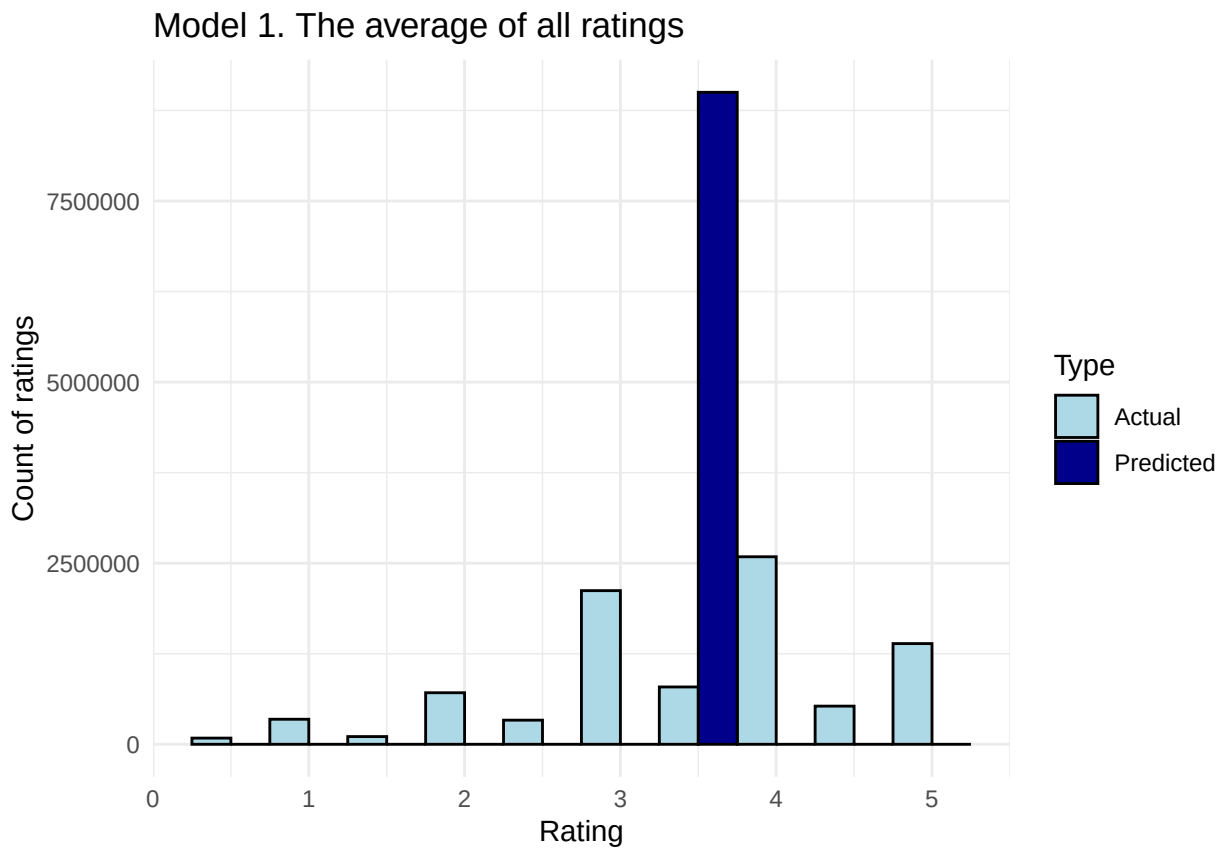
Model 1.

The basic model is to calculate the average of all ratings and use it as a prediction for each movie.

```
## The average value of all ratings
mu <- mean(edx$rating)

## Predictions based on the average
predictions_mu <- rep(mu, nrow(edx))

## Function for calculating RMSE
rmse <- function(actual, predicted) {sqrt(mean((actual - predicted)^2))}

rmse_base <- rmse(edx$rating, predictions_mu)
rmse_base
```

```
## [1] 1.060331
```

```
## Creating a dataframe for real and predicted estimates of the Model 1
data_plot_base <- data.frame(
  Rating = c(edx$rating, predictions_mu),
  Type = c(rep("Actual", length(edx$rating)), rep("Predicted", length(predictions_mu)))
)

## Plotting Model 1
ggplot(data_plot_base, aes(x = Rating, fill = Type)) +
  geom_histogram(binwidth = 0.5, position = "dodge", color = "black") +
  scale_fill_manual(values = c("Actual" = "lightblue", "Predicted" = "darkblue")) +
  labs(title = "Model 1. The average of all ratings",
       x = "Rating",
       y = "Count of ratings") +
  theme_minimal()
```



Model 2.

Calculate the average rating for each movie based on the edx training set.

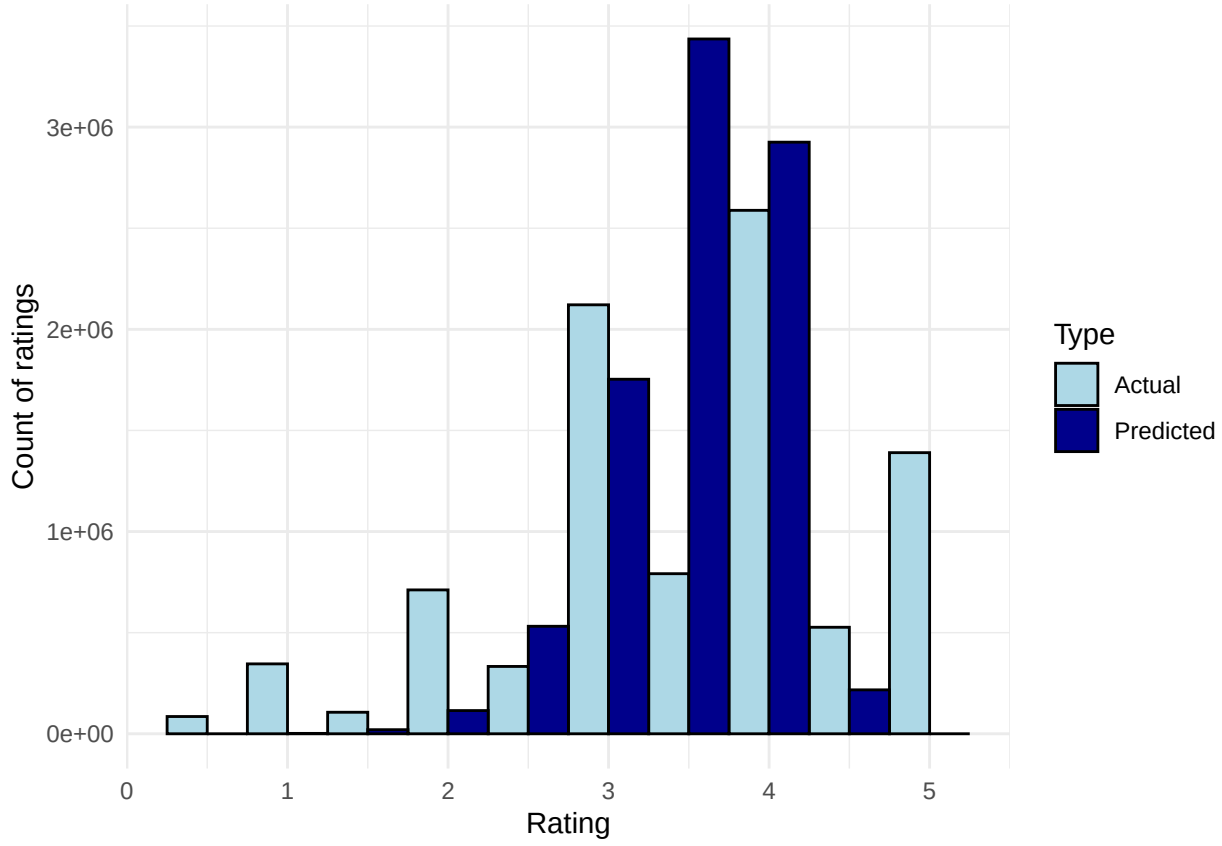
Use this average rating as a prediction for each movie.

```
## Calculate the average rating for each movie
movie_avgs <- edx %>% group_by(movieId) %>% summarise(avg_movie_rating = mean(rating))

## Adding the average ratings to the test set
test_set_movie <- edx %>% left_join(movie_avgs, by = "movieId")

## Use average movie ratings to predict
predictions_movie <- test_set_movie$avg_movie_rating

rmse_movie <- rmse(test_set_movie$rating, predictions_movie)
rmse_movie
```

```
## [1] 0.9423475
```

```r
## Creating a dataframe for real and predicted estimates of the Model 2
data_plot_movie <- data.frame(
  Rating = c(test_set_movie$rating, predictions_movie),
  Type = c(rep("Actual", length(test_set_movie$rating)), rep("Predicted", length(predictions_movie)))
)

## Plotting Model 2
ggplot(data_plot_movie, aes(x = Rating, fill = Type)) +
  geom_histogram(binwidth = 0.5, position = "dodge", color = "black") +
  scale_fill_manual(values = c("Actual" = "lightblue", "Predicted" = "darkblue")) +
  labs(title = cat("Model 2. The average of rating for each movie"),
       x = "Rating",
       y = "Count of ratings") +
  theme_minimal()
```

```
## Model 2. The average of rating for each movie
```



Model 3.

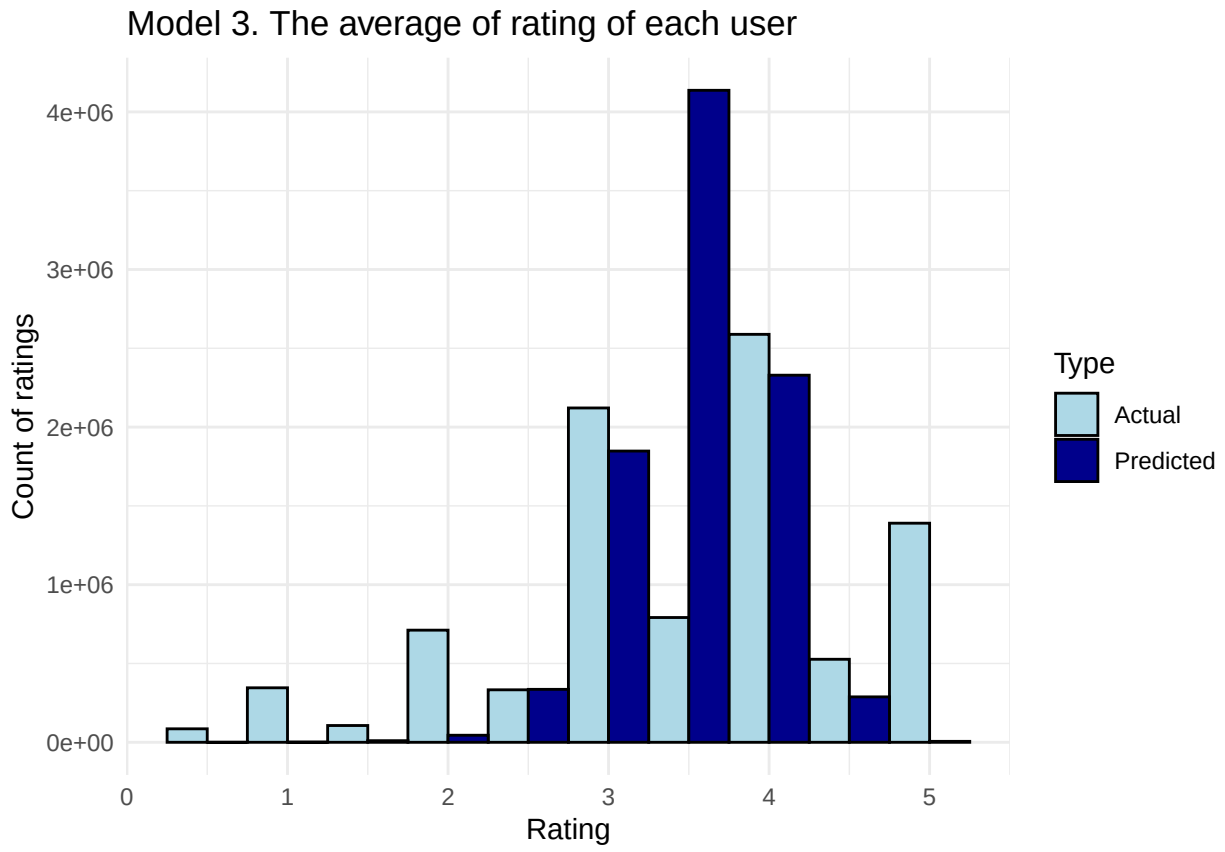
Calculate the average rating of each user based on the edx training set.

Use this average rating as a prediction.

```
## Calculate the average rating for each user
user_avgs <- edx %>% group_by(userId) %>% summarise(avg_user_rating = mean(rating))

## Adding average user ratings to the test set
test_set_user <- edx  %>% left_join(user_avgs, by = "userId")

## Use average user ratings to predict
predictions_user <- test_set_user$avg_user_rating

rmse_user <- rmse(test_set_user$rating, predictions_user)
rmse_user
```

```
## [1] 0.9700086
```

```
## Creating a dataframe for real and predicted estimates of the Model 3
data_plot_user <- data.frame(
  Rating = c(test_set_user$rating, predictions_user),
  Type = c(rep("Actual", length(test_set_user$rating)), rep("Predicted", length(predictions_user)))
)

## Plotting Model 3
ggplot(data_plot_user, aes(x = Rating, fill = Type)) +
  geom_histogram(binwidth = 0.5, position = "dodge", color = "black") +
  scale_fill_manual(values = c("Actual" = "lightblue", "Predicted" = "darkblue")) +
  labs(title = "Model 3. The average of rating of each user",
       x = "Rating",
       y = "Count of ratings") +
  theme_minimal()
```



Model 4.

To improve the model, calculate the average ratings for each movie and for each user.

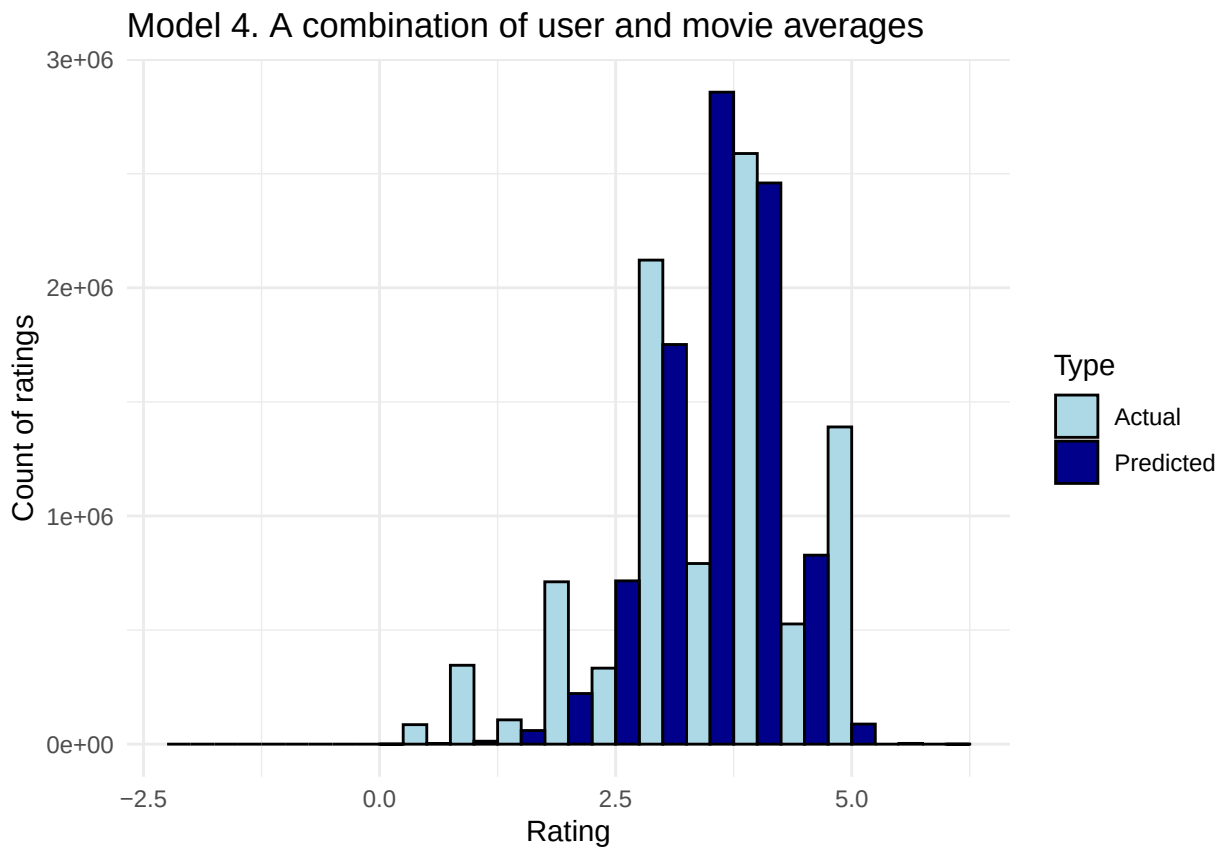
Add them up with the total average estimate (calculated in the base model) for prediction.

```
## Calculate the overall average score (from the base model)
mu <- mean(edx$rating)

## Calculate the deviation for each film from the total average
movie_avgs_b_i <- edx %>% group_by(movieId) %>% summarise(b_i = mean(rating - mu))

## Calculate the deviation for each user from the total average
user_avgs_b_u <- edx %>% left_join(movie_avgs_b_i, by = "movieId") %>% group_by(userId) %>% summarise(b

## Adding deviations to the test set
test_set_combined <- edx %>% left_join(movie_avgs_b_i, by = "movieId") %>% left_join(user_avgs_b_u, by =
```

```r
## Predict the rating as the sum of the total average, the rejection of the movie and the rejection of
predictions_combined <- mu + test_set_combined$b_i + test_set_combined$b_u
rmse_combined <- rmse(test_set_combined$rating, predictions_combined)
rmse_combined
```

```
## [1] 0.8567039
```

```r
## Creating a dataframe for real and predicted estimates of the Model 4
data_plot_combined <- data.frame(
  Rating = c(test_set_combined$rating, predictions_combined),
  Type = c(rep("Actual", length(test_set_combined$rating)), rep("Predicted", length(predictions_combine
)

## Plotting Model 4
ggplot(data_plot_combined, aes(x = Rating, fill = Type)) +
  geom_histogram(binwidth = 0.5, position = "dodge", color = "black") +
  scale_fill_manual(values = c("Actual" = "lightblue", "Predicted" = "darkblue")) +
  labs(title = "Model 4. A combination of user and movie averages",
       x = "Rating",
       y = "Count of ratings") +
  theme_minimal()
```



Model 5.

To avoid skewing due to rare movies or users who rated few movies.

Regularization helps mitigate the impact of such data by adding a smoothing parameter.

The lambda regularization parameter has been introduced to smooth out the average values.

Cross-validation was used to select the best lambda value.

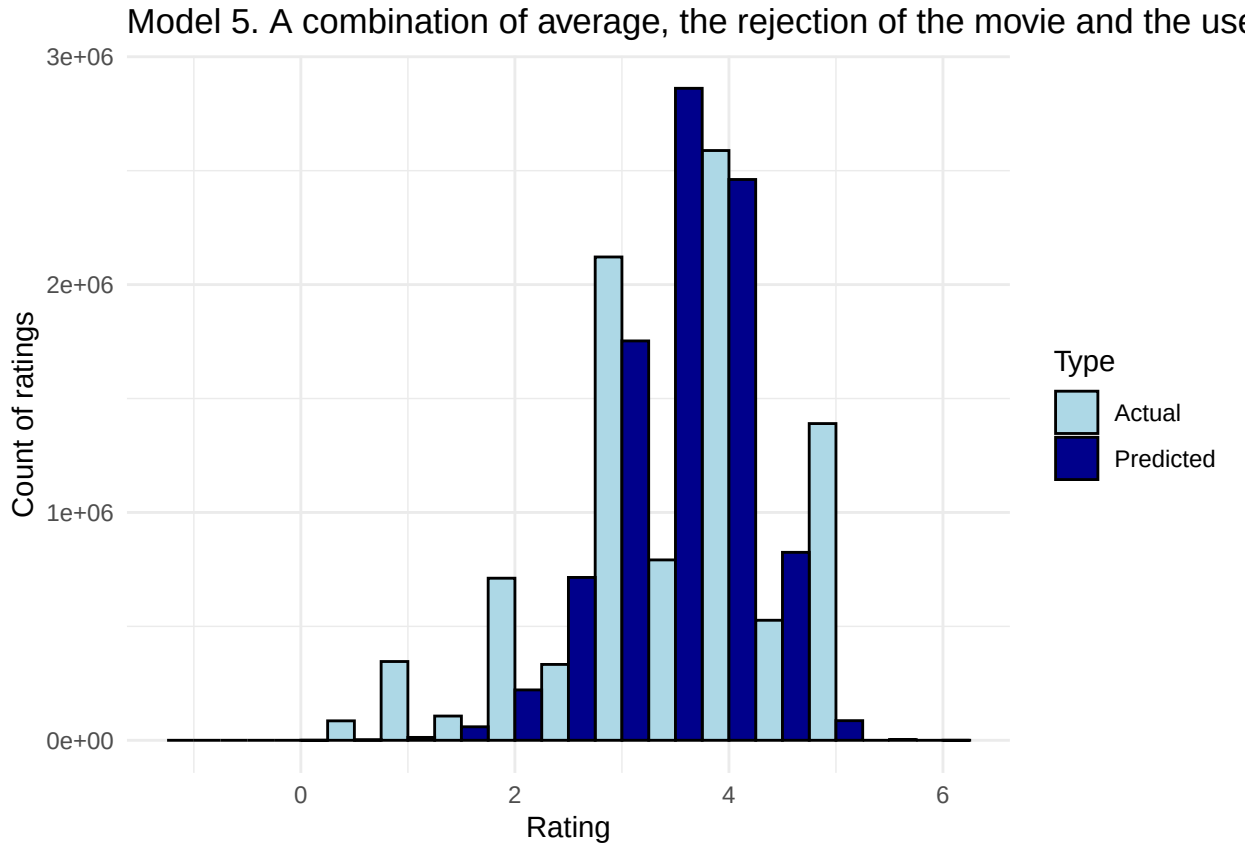
```
## Selecting the lambda parameter
lambdas <- seq(0, 10, 0.25)

## Function for calculating RMSE with regularization
rmses <- sapply(lambdas, function(lambda) {
  ### Smoothed averages for movies
  b_i_smoothed <- edx %>%
    group_by(movieId) %>%
    summarise(b_i_smoothed = sum(rating - mu) / (n() + lambda))

  ### Smoothed averages for movies
  b_u_smoothed <- edx %>%
    left_join(b_i_smoothed, by = "movieId") %>%
    group_by(userId) %>%
    summarise(b_u_smoothed = sum(rating - mu - b_i_smoothed) / (n() + lambda))

  ### Forecast with regularization
  test_set_smoothed <- edx %>%
    left_join(b_i_smoothed, by = "movieId") %>%
    left_join(b_u_smoothed, by = "userId")

  predictions_smoothed <- mu + test_set_smoothed$b_i_smoothed + test_set_smoothed$b_u_smoothed

  return(rmse(test_set_smoothed$rating, predictions_smoothed))
})

## The optimal lambda parameter
optimal_lambda <- lambdas[which.min(rmses)]
optimal_lambda
```

```
## [1] 0.5
```

```
## Lambda regularization parameter
## Replace it with the optimal value found earlier
lambda <- optimal_lambda
```

```r
## Calculate the overall average score (from the base model)
mu <- mean(edx$rating)

## Smoothed averages for films with regularization
movie_avgs_b_i_smoothed <- edx %>%
  group_by(movieId) %>%
  summarise(b_i_smoothed = sum(rating - mu) / (n() + lambda))

## Smoothed averages for users with regularization
user_avgs_b_u_smoothed <- edx %>%
  left_join(movie_avgs_b_i_smoothed, by = "movieId") %>%
  group_by(userId) %>%
  summarise(b_u_smoothed = sum(rating - mu - b_i_smoothed) / (n() + lambda))

## Adding deviations to the test set
test_set_smoothed <- edx %>%
  left_join(movie_avgs_b_i_smoothed, by = "movieId") %>%
  left_join(user_avgs_b_u_smoothed, by = "userId")

## Predict the rating as the sum of the total average, the rejection of the movie and of the user
predictions_smoothed <- mu + test_set_smoothed$b_i_smoothed + test_set_smoothed$b_u_smoothed

## Calculating the RMSE
rmse_smoothed <- rmse(test_set_smoothed$rating, predictions_smoothed)
```

```r
## Creating a dataframe for real and predicted estimates of the Model 5
data_plot_smoothed <- data.frame(
  Rating = c(test_set_smoothed$rating, predictions_smoothed),
  Type = c(rep("Actual", length(test_set_smoothed$rating)), rep("Predicted", length(predictions_smoothe
)

## Plotting Model 5
ggplot(data_plot_smoothed, aes(x = Rating, fill = Type)) +
  geom_histogram(binwidth = 0.5, position = "dodge", color = "black") +
  scale_fill_manual(values = c("Actual" = "lightblue", "Predicted" = "darkblue")) +
  labs(title = "Model 5. A combination of average, the rejection of the movie and the user",
       x = "Rating",
       y = "Count of ratings") +
  theme_minimal()
```



Assessment of the quality of predictions of the project.

The model with the lowest RMSE on the test set was selected and applied to the final sample final_holdout_test to assess the quality of predictions for the project completion.

```
final_holdout_test <- final_holdout_test %>%
  left_join(movie_avgs_b_i_smoothed, by = "movieId") %>%
  left_join(user_avgs_b_u_smoothed, by = "userId")

## Predictions on `final_holdout_test` using a model with a combination of user and movie averages
final_predictions <- mu + final_holdout_test$b_i_smoothed + final_holdout_test$b_u_smoothed

## The final calculation of RMSE on `final_holdout_test`
rmse_final <- rmse(final_holdout_test$rating, final_predictions)
rmse_final
```

```
## [1] 0.8652226
```