MovieLens Project: edX Harvard Data Science Capstone Project

By: Xaiver Lim

November 5, 2020

Contents

Introduction

The purpose of this project is to create a model that will predict how a user will rate a specific movie, similar to a movie recommendation system. My model will make predictions based on user ratings of other movies and the average rating of the specific movie. First, I will run an exploratory data analysis to get a general overview of the data and explore possible predictor variables. Then, I will include the most important features into numerous models and train them using training data. Finally, I will deploy the model with the smallest error to the test set and evaluate the results.

Data

For this project, I will use the 10M version of the MovieLens dataset which I will collect from https://grouplens.org/datasets/movielens/10m/.

The dataset presents information about 10 million movie ratings including user id, movie id, user rating of the movie (between 0.5 to 5 stars), timestamp of the rating (seconds since midnight Coordinated Universal Time of January 1, 1970), title of the movie, and movie genre(s): Action, Adventure, Animation, Children's, Comedy, Crime, Documentary, Drama, Fantasy, Film-Noir, Horror, IMAX, Musical, Mystery, Romance, Sci-Fi, Thriller, War, and/or Western.

After downloading and unzipping the data file, I will extract its contents and place the data in a dataframe. Next, I will split the data into two sets: a training set and a validation set. The training set will contain 90% of the data (9 million ratings) which the model will learn from. The validation set will contain the remaining 10% of data (1 million ratings) which will be used to evaluate the performance of my model.

```
############################################################
# 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")
```

```
## Loading required package: tidyverse
```

```
## Warning: package 'tidyverse' was built under R version 3.6.2
```

```
## -- Attaching packages ---------------------------------------- tidyverse 1.3.0 --
```

```
## v ggplot2 3.2.1     v purrr   0.3.3
## v tibble  2.1.3     v dplyr   0.8.3
## v tidyr   1.0.0     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://cran.us.r-project.org")
```

```
## Loading required package: caret

## Warning: package 'caret' was built under R version 3.6.3

## Loading required package: lattice

##
## Attaching package: 'caret'

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

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

```
## Loading required package: data.table

## Warning: package 'data.table' was built under R version 3.6.2

##
## Attaching package: 'data.table'

## The following objects are masked from 'package:dplyr':
##
##     between, first, last

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

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

# 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")

# if using R 3.6 or earlier:
```

```
movies <- as.data.frame(movies) %>% mutate(movieId = as.numeric(levels(movieId))[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)`
```

```
## Warning in set.seed(1, sample.kind = "Rounding"): non-uniform 'Rounding' sampler
## used
```

```
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)

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

Training Data

```
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
```

Methods & Analysis

Prior to the exploratory data analysis, I will check if the dataset contains any missing values.

```
# Check for missing values
any(is.na(edx))
```

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

There are no missing values in this dataset. Now, I will explore the data to extract insights.

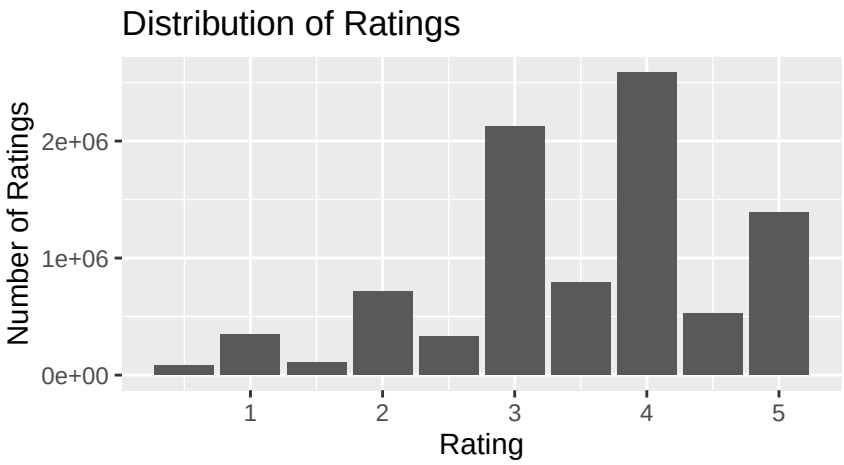
Exploratory Data Analysis (EDA)

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

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

The average rating across the 9 million ratings in the training set is 3.512 stars.

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



As you can see, the ratings appear to be left-skewed since there are few ratings between 0 to 2 stars and many ratings between 3 to 5 stars. It is important to note that users only had the option to select whole number or half ratings. Thus, there were only ten options users could select from (0.5, 1, 1.5, 2, 2.5, 3, 3.5, 4, 4.5, 5). In general, half star ratings appear to be less common than whole star ratings.

Now, I will perform a deeper analysis of ratings by movie. Specifically, I will compare the average rating of movies and determine whether the number of ratings a movie receives impacts its average rating.

Ratings by Movie

```
length(unique(edx$movieId))
```

```
## [1] 10677
```

There are 10,677 movies in the training dataset.

- Movies with the Most Ratings

```
# Movies with the most ratings, accompanied by their average rating
edx %>% group_by(title) %>%
  summarize(avg_movie_rating = mean(rating), num_ratings = n()) %>%
  arrange(desc(num_ratings))-> movie_ratings

head(movie_ratings)
```

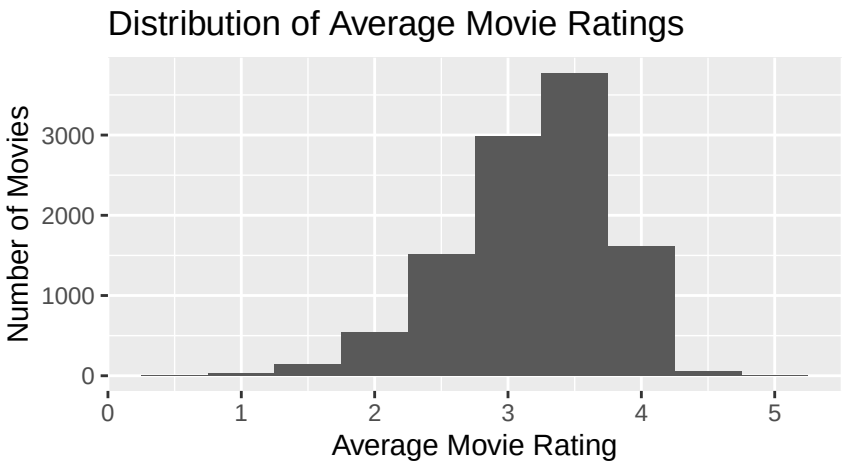
```
## # A tibble: 6 x 3
##   title                              avg_movie_rating num_ratings
##   <chr>                                         <dbl>       <int>
## 1 Pulp Fiction (1994)                            4.15       31362
## 2 Forrest Gump (1994)                            4.01       31079
## 3 Silence of the Lambs, The (1991)               4.20       30382
## 4 Jurassic Park (1993)                           3.66       29360
## 5 Shawshank Redemption, The (1994)               4.46       28015
## 6 Braveheart (1995)                              4.08       26212
```

Pulp Fiction, *Forrest Gump*, *The Silence of the Lambs*, *Jurassic Park*, *Shawshank Redemption*, and *Braveheart* have the most ratings (about 30,000 each) with an average rating ranging between 3.66 and 4.46.

- Most Common Average Movie Rating

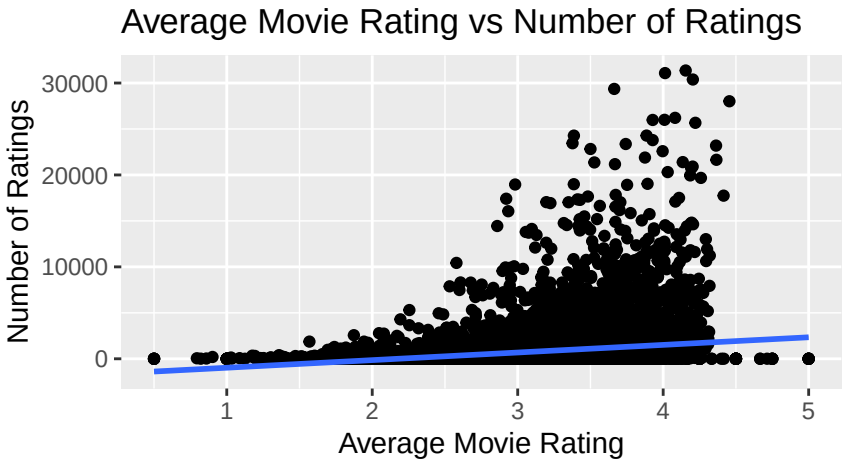
Based on the histogram below, most movies appear to have an average rating between 2.5 and 4. In addition, there are only be a few movies with an average rating of 0.5 stars (worst possible rating) and 5 stars (perfect rating).

```
ggplot(movie_ratings, aes(x=avg_movie_rating)) + geom_histogram(bins=10) +
    labs(title = "Distribution of Average Movie Ratings",
         x = "Average Movie Rating",
         y = "Number of Movies")
```



- Do movies with many ratings tend to be rated higher than movies with few ratings?

```
ggplot(movie_ratings,aes(x = avg_movie_rating, y = num_ratings)) + geom_point() + geom_smooth(method =
    labs(title = "Average Movie Rating vs Number of Ratings",
         x = "Average Movie Rating",
         y = "Number of Ratings")
```




```
cor(movie_ratings$avg_movie_rating, movie_ratings$num_ratings)
```

```
## [1] 0.2114024
```

In general, the more a movie is rated by users, the greater its average rating. However, this relationship is relatively weak.

Now, I will perform a deeper analysis of ratings by user. Similar to my analysis of ratings by movie, I will compare the average rating of users and determine whether the number of ratings they have given in total impact their average rating.

Ratings by User

```
length(unique(edx$userId))
```

```
## [1] 69878
```

There are 69,878 users in the training dataset.

- Users Who Rated the Most Movies

```
# Users who rated the most movies, accompanied by their average rating
edx %>% group_by(userId) %>%
  summarize(avg_user_rating = mean(rating), num_ratings = n()) %>%
  arrange(desc(num_ratings)) -> user_ratings

head(user_ratings)
```

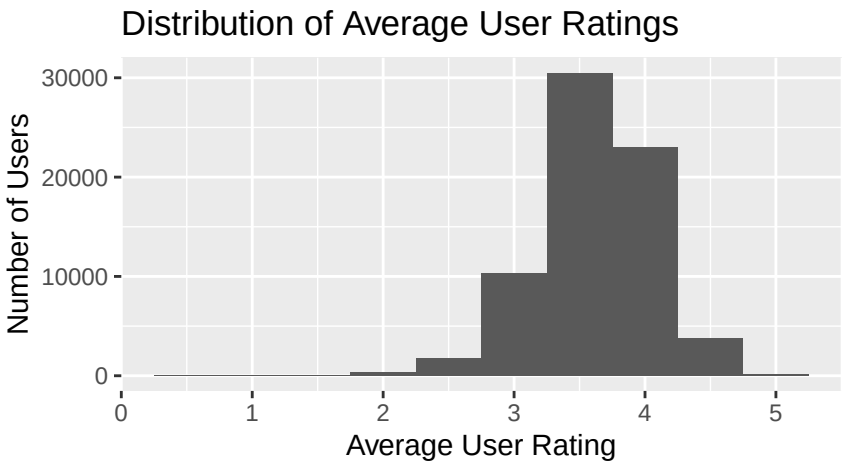
```
## # A tibble: 6 x 3
##   userId avg_user_rating num_ratings
##    <int>           <dbl>       <int>
## 1  59269            3.26        6616
## 2  67385            3.20        6360
## 3  14463            2.40        4648
## 4  68259            3.58        4036
## 5  27468            3.83        4023
## 6  19635            3.50        3771
```

The user that rated the most movies rated a total of 6616 movies with an average rating of 3.26.

- Most Common Average Rating Given by Users

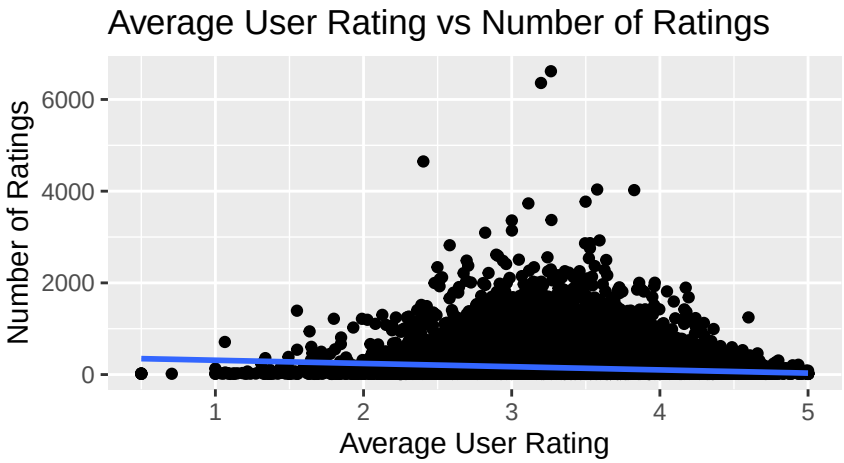
Based on the histogram below, most users give an average rating between 3 and 4.5. In addition, only a few users have a very high or very low average rating (i.e. 0 to 2 stars; 5 stars).

```
ggplot(user_ratings, aes(x=avg_user_rating)) + geom_histogram(bins=10) +
    labs(title = "Distribution of Average User Ratings",
         x = "Average User Rating",
         y = "Number of Users")
```



- Do users who rate many movies tend to rate higher than users who rate few movies?

```
ggplot(user_ratings,aes(x = avg_user_rating, y = num_ratings)) + geom_point() + geom_smooth(method = "lm
    labs(title = "Average User Rating vs Number of Ratings",
         x = "Average User Rating",
         y = "Number of Ratings")
```




```
cor(user_ratings$avg_user_rating, user_ratings$num_ratings)
```

```
## [1] -0.1550551
```

In general, the more a user rates movies, the lower their average rating tends to be. However, this relationship is relatively weak.

Key Insights

Based on my EDA, I would expect most ratings to be between 3 and 4 stars. In addition, both movie and user averages appear to have an impact on the actual rating, so I will include these features in model development. However, the number of ratings a movie receives and the number of movies a user rates has a small impact on the actual rating. Thus, I will not include either component in model development.

Model Development

Models

I will test three different regression models to predict each rating in the training set (edx). Then, I will select the best model and apply it to the test set (validation).

Model 1: Predicted Rating = Global Average Rating + Movie Effect

Model 2: Predicted Rating = Global Average Rating + User Effect

Model 3: Predicted Rating = Global Average Rating + Movie Effect + User Effect

The global average rating is the average rating across all entries in the dataset. The movie effect is the difference between the average rating for the specific movie and the global average rating. Similarly, the user effect is the difference between the average rating for the specific user and the global average rating.

Model Example

For example, suppose I am trying to predict User X's rating of *Forest Gump* and the overall average rating (across all movies/users) is 3 stars, the average rating of Forest Gump is 4 stars, and the average rating User X gives is 2.5 stars. Thus the global average rating is 3, the movie effect is +1 (4-3), and the user effect is -0.5 (2.5-3).

Model 1: Predicted Rating = 3 + 1 = 4

Model 2: Predicted Rating = 3 - 0.5 = 2.5

Model 3: Predicted Rating = 3 + 1 - 0.5 = 3.5

To evaluate the three models, I will use RMSE (Root Mean Square Error). Ultimately, I will select the model with the lowest RMSE.

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

The global average rating is 3.512465.

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

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

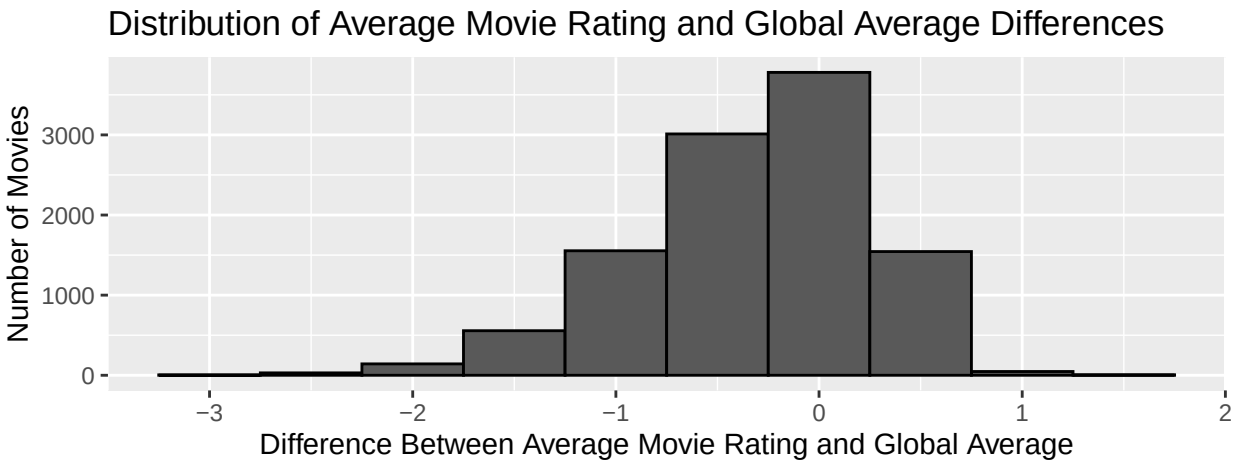
Now I will calculate the movie effect for each movie.

Model 1: Movie Effect

```
# Calculate difference between each movie's average rating and the overall average rating
movie_ratings %>% mutate(movie_avg_diff = avg_movie_rating - overall_avg_rating) -> movie_ratings
head(movie_ratings)
```

```
## # A tibble: 6 x 4
##   title                              avg_movie_rating num_ratings movie_avg_diff
##   <chr>                                         <dbl>       <int>          <dbl>
## 1 Pulp Fiction (1994)                            4.15       31362          0.642
## 2 Forrest Gump (1994)                            4.01       31079          0.500
## 3 Silence of the Lambs, The (1991)               4.20       30382          0.692
## 4 Jurassic Park (1993)                           3.66       29360          0.151
## 5 Shawshank Redemption, The (1994)               4.46       28015          0.943
## 6 Braveheart (1995)                              4.08       26212          0.569
```

```
qplot(movie_avg_diff, data = movie_ratings, bins = 10, color = I("black"))+
    labs(title = "Distribution of Average Movie Rating and Global Average Differences",
         x = "Difference Between Average Movie Rating and Global Average",
         y = "Number of Movies")
```



As you can see, a lot of movies have an average rating close to the global average (difference of 0) and few movies have an average rating that is far from the global average (difference of +/- 2). In addition, the average rating for the specific movie is more likely to have a negative impact on each rating since there are more negative differences.

```
model_1_predictions <- overall_avg_rating + edx %>%
  left_join(movie_ratings, by='title') %>%
  pull(movie_avg_diff)

RMSE(model_1_predictions, edx$rating)
```

```
## [1] 0.9423477
```

The RMSE taking into account only the movie effect is 0.9423.

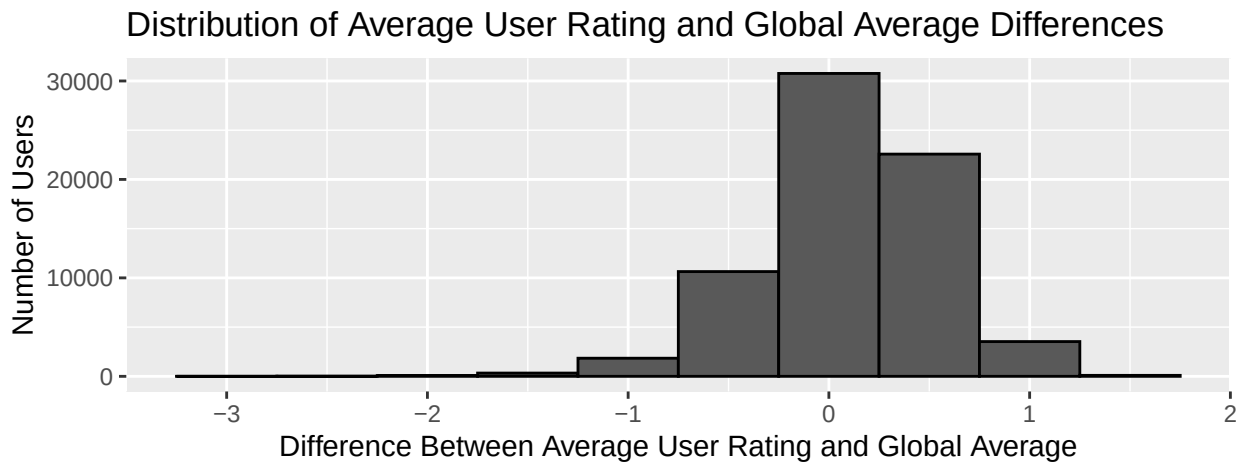
Now, let's analyze the user effect.

Model 2: User Effect

```
# Calculate difference between each user's average rating and the overall average rating
user_ratings %>% mutate(user_avg_diff = avg_user_rating - overall_avg_rating) -> user_ratings
head(user_ratings)
```

```
## # A tibble: 6 x 4
##   userId avg_user_rating num_ratings user_avg_diff
##    <int>           <dbl>       <int>         <dbl>
## 1  59269            3.26        6616       -0.248
## 2  67385            3.20        6360       -0.315
## 3  14463            2.40        4648       -1.11
## 4  68259            3.58        4036        0.0645
## 5  27468            3.83        4023        0.314
## 6  19635            3.50        3771       -0.0137
```

```
qplot(user_avg_diff, data = user_ratings, bins = 10, color = I("black")) +
    labs(title = "Distribution of Average User Rating and Global Average Differences",
         x = "Difference Between Average User Rating and Global Average",
         y = "Number of Users")
```



As you can see, a lot of users have an average rating close to the global average (difference of 0) and few users have an average rating that is far from the global average (difference of +/- 2). In addition, the average rating for the specific user is more likely to have a positive impact on each rating since there are more positive differences.

```
model_2_predictions <- overall_avg_rating + edx %>%
  left_join(user_ratings, by='userId') %>%
  pull(user_avg_diff)

RMSE(model_2_predictions, edx$rating)
```

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

The RMSE taking into account only the user effect is 0.9700, which is higher (worse) than model 1.

Finally, I will take into account both movie and user effect.

Model 3: Movie & User Effect

```
model_3_predictions <- edx %>%
  left_join(movie_ratings, by='title') %>%
  left_join(user_ratings, by='userId') %>%
  mutate(pred = overall_avg_rating + movie_avg_diff + user_avg_diff) %>%
  pull(pred)

RMSE(model_3_predictions, edx$rating)
```

```
## [1] 0.8767534
```

The RMSE taking into account both the movie and user effect is 0.8768.

Results

Model Evaluation

Model	RMSE
Model 1: Movie Effect	0.9423
Model 2: User Effect	0.9700
Model 3: Movie & User Effect	0.8768

Model 3 (Movie & User Effect) has the lowest RMSE. Thus, I will deploy this model to the validation dataset.

Model Deployment

After deploying model 3, which incorporates movie and user effects, the resulting RMSE is 0.8850

```
validation_predictions <- validation %>%
  left_join(movie_ratings, by='title') %>%
  left_join(user_ratings, by='userId') %>%
  mutate(pred = overall_avg_rating + movie_avg_diff + user_avg_diff) %>%
  pull(pred)

RMSE(validation_predictions, validation$rating)
```

```
## [1] 0.8850398
```

Conclusion

In conclusion, taking into account user preferences and a movie's average rating does a better job of predicting ratings than simply taking into account only user effects or only movie effects.

My model does not take into account changes in user preferences which is a limitation of my model. However, a component can be implemented in future iterations to capture this feature. For example, I could create a rolling average metric that would determine a user's average rating of their 10 most recent ratings. This may better represent user preferences because it takes into account possible changes in user tendencies. However, it may also create a lot of "noise". For example, a user may decide to watch a lot of highly rated movies back-to-back which would skew predictions.

Ideally, I would also like my model to include a component that would capture similarity scores between users. For example, if user A rates movies similar to user B, and user B rated *Jurassic Park* 3 stars, user A should be likely to rate *Jurassic Park* close to 3 stars. Similarly, I would like my model to integrate an element that would capture similarity scores between movies, based on genres. For example, if Drama and War movies tend to be rated higher than other movies, the model should take it into account. However, considering the dataset contains about 10 million ratings, it is challenging to implement these components without significantly slowing down the time it takes for the model to make predictions.