Movielens Movie Rating Prediction - Capstone Project

Christopher Pickering

2019 06 15

Introduction

MovieLens is a data science research site at https://movielens.org/ (https://movielens.org/) initiated by the GroupLens research group of University of Minnesota. Users can rate movies and based on various machine learning approaches they will receive recommendations of future movies to watch. The goal is that these recommendations would be customized and that the user would then get the chance to watch more movies that they will actually enjoy.

The commercial interest in these types of predictions follows from Netflix and other streaming services. When you finished watching a show, Netflix gives you a list of other shows to watch. When you first join there is no data to base these suggestions on, so these are probably not very useful recommendations. However, the more you watch the more personalized user data is available. With time, recommendations become much better and one can devote all your time to watching shows that you will like. This would not have been possible without data science and the development of machine learning regression models to predict the rating outcome. This challenge was put forth to the data science community and this HarvardX Data Science Capstone project illustrates how this process can be used.

Aim

The aim of this project was to use machine learning to train a model to predict the outcome **movie rating**. This variable is continuous so a regression approach was used with error reported as Root Mean Square Error (RMSE).

Methods

The dataset for this project was generated using code provided for the 2019 edition of the HarvardX Data Science Capstone course. The GitHub repo https://github.com/nordicbychris/MovielensMLEdxProject20190608.git (https://github.com/nordicbychris/MovielensMLEdxProject20190608.git) has a separate R script for this generation if this is desired. However, all of the necessary code is available in a single R script file called **MovielensModelDev.R**.

Pre-processing, wrangling and partitioning

The following code was supplied. It downloads the data, joins the movies and ratings and creates a partition into **edx** which is the *training* set and **validation** which is the *testing* set (10% of the total size).

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

# Merge the ratings with the title and genre information, since these are in separate
files

ratings <- read.table(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(levels(movieId))[movie
Id],
                                           title = as.character(title),
                                           genres = as.character(genres))

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

# edx is the training set
# validation is the test set and consists of 10% of MovieLens dataset

set.seed(1) # if using R 3.6.0: set.seed(1, sample.kind = "Rounding")
test_index <- createDataPartition(y = movielens$rating, times = 1, p = 0.1, list = FAL
SE)
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)
```

Data visualization

Data were explored first using various base characteristics.

```
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
```

```
dim(edx)
```

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

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

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

The dataset was clean in that it did not have any missing (NA) or unrealistic values. There were 10677 movies in the database and 69878 users. Each row contained a different individual movie rating, giving 9000055 ratings in total in the training set.

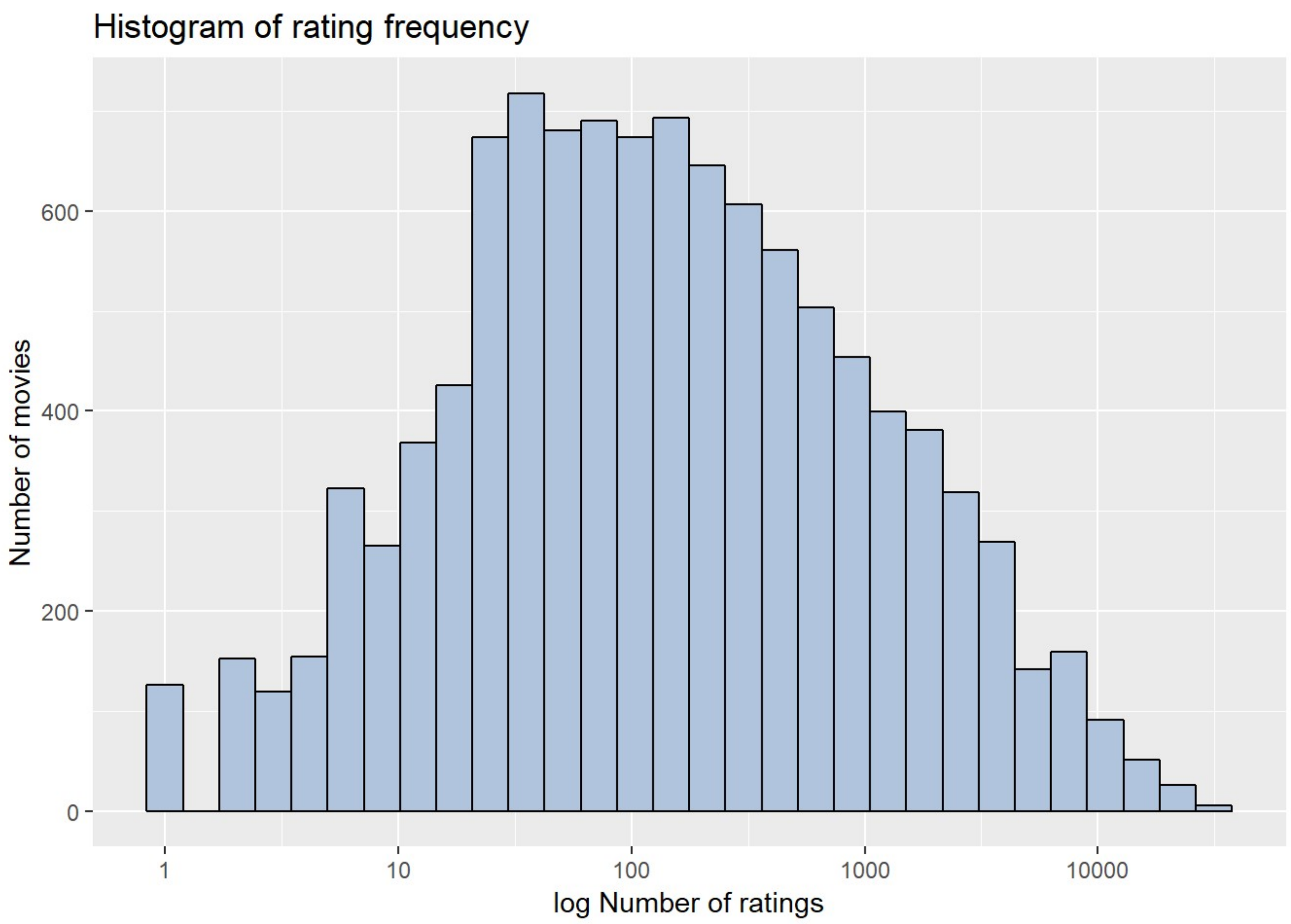
```
edx %>%
  group_by(title) %>%
  summarise(rating = mean(rating)) %>%
  top_n(10) %>%
  arrange(desc(rating))
```

```
## Selecting by rating
## # A tibble: 10 x 2
##    title                                                              rating
##    <chr>                                                               <dbl>
##  1 Blue Light, The (Das Blaue Licht) (1932)                             5   
##  2 Fighting Elegy (Kenka erejii) (1966)                                 5   
##  3 Hellhounds on My Trail (1999)                                        5   
##  4 Satan's Tango (SÃ¡tÃ¡ntangÃ³) (1994)                                  5   
##  5 Shadows of Forgotten Ancestors (1964)                                5   
##  6 Sun Alley (Sonnenallee) (1999)                                       5   
##  7 Constantine's Sword (2007)                                           4.75
##  8 Human Condition II, The (Ningen no joken II) (1959)                  4.75
##  9 Human Condition III, The (Ningen no joken III) (1961)                4.75
## 10 Who's Singin' Over There? (a.k.a. Who Sings Over There) (Ko to t~   4.75
```

For interest, the above code gives the Top 10 movies in terms of average rating in this dataset. There were 6 movies with an average rating of 5 out of 5.

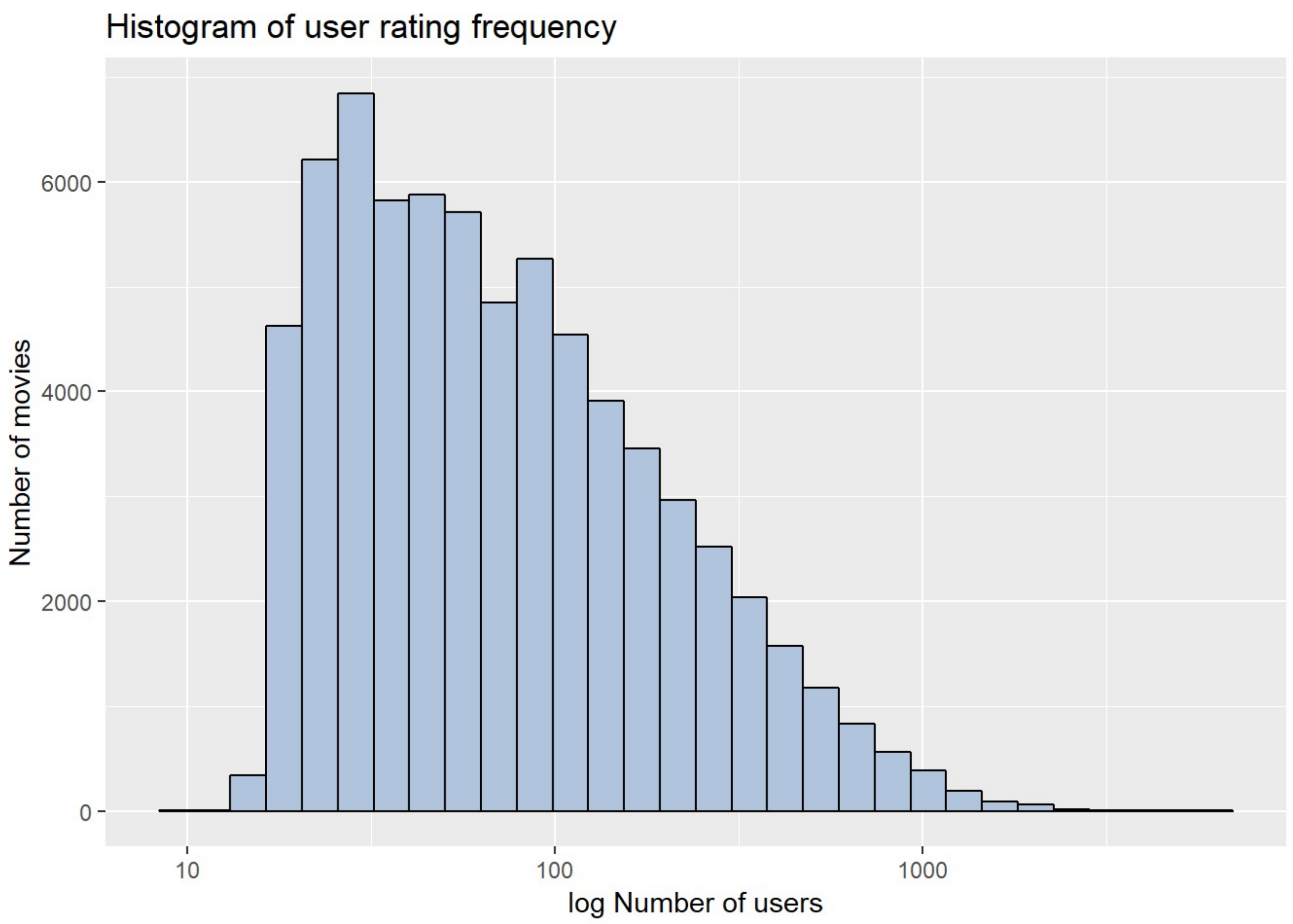
To look for possible predictors the data for movie, user and genre were visualized as follows.

```
edx %>%
  count(movieId) %>%
  ggplot(aes(n)) +
  geom_histogram(bins = 30, color = "black", fill = "lightsteelblue") +
  scale_x_log10() +
  ylab("Number of movies") +
  xlab("log Number of ratings") +
  ggtitle("Histogram of rating frequency")
```



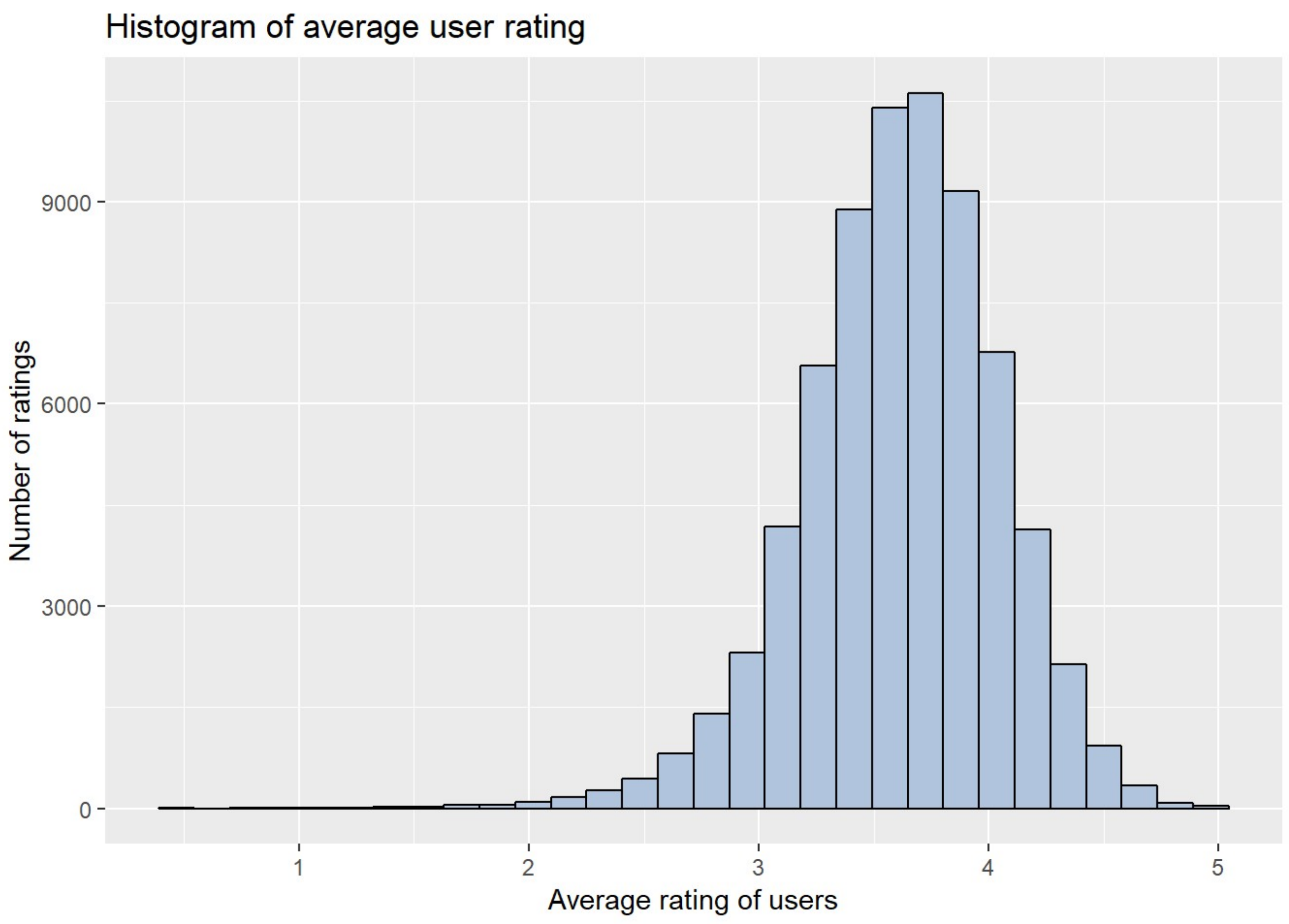
Some movies were rated a lot more frequently and one can assume that the more ratings given the more likely this rating will approach its true value (expected value).

```
edx %>%
  count(userId) %>%
  ggplot(aes(n)) +
  geom_histogram(bins = 30, color = "black", fill = "lightsteelblue") +
  scale_x_log10() +
  ylab("Number of movies") +
  xlab("log Number of users") +
  ggtitle("Histogram of user rating frequency")
```



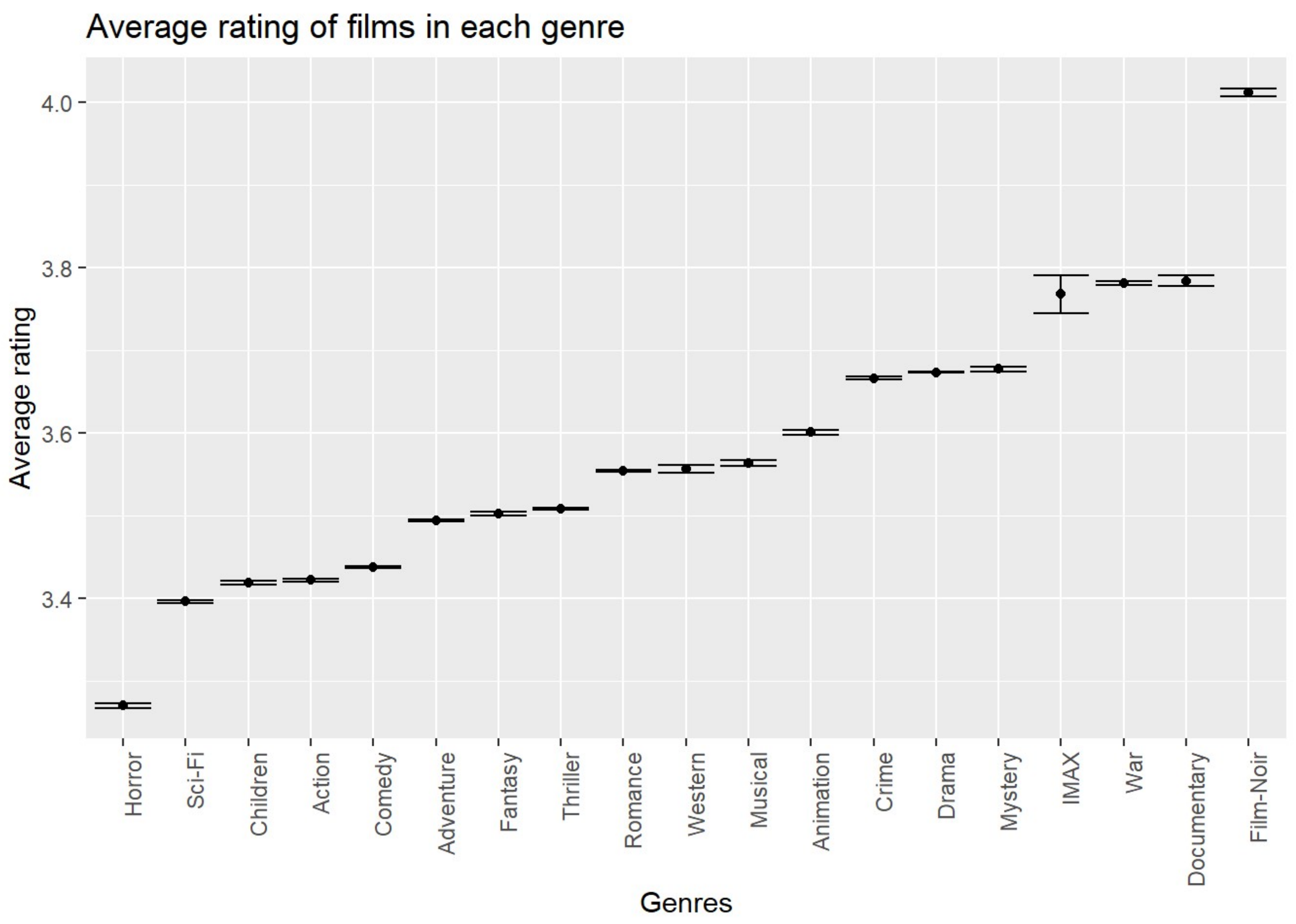
A similar observation could be seen for users in that some rated more movies than others. This can be further explored as follows.

```
edx %>%
  group_by(userId) %>%
  summarize(b_u = mean(rating)) %>%
  filter(n()>=100) %>%
  ggplot(aes(b_u)) +
  geom_histogram(bins = 30, color = "black", fill = "lightsteelblue") +
  ylab("Number of ratings") +
  xlab("Average rating of users") +
  ggtitle("Histogram of average user rating")
```



It can also be seen that some users tend to give higher ratings than others. Because of this, using the mean of the user's individual score may be a good reflection of how they will rate future movies. Following this logic, this could be useful in predicting the movie rating outcome and ultimately recommending a new film to watch.

```
edx %>%
  separate_rows(genres, sep = "\\|") %>%
  group_by(genres) %>%
  summarize(n = n(), avg = mean(rating), se = sd(rating)/sqrt(n())) %>%
  filter(n >= 1000) %>%
  mutate(genres = reorder(genres, avg)) %>%
  ggplot(aes(x = genres, y = avg, ymin = avg - 2*se, ymax = avg + 2*se)) +
  geom_point() +
  geom_errorbar() +
  xlab("Genres") +
  ylab("Average rating") +
  ggtitle("Average rating of films in each genre") +
  theme(axis.text.x = element_text(angle = 90, hjust = 1))
```



It was also noted that some genres received a higher rating than others. For example, films in the *Film-Noir* category were significantly higher rated than those in category *Horror*.

Modeling and Results

To predict the outcome of **movie rating** regression models were built based on the predictors *existing movie ratings*, *user ratings* and *genre*. To compare the results a function to calculate RMSE was necessary.

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

After each step the RMSE was stored in a data frame (rmse_results) to keep track of the progress.

Model 1: Average

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

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

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

```
## [1] 1.061202
```

```
# Report model RMSE results in a table
rmse_results <- data_frame(method = "Taking the average", RMSE = naive_rmse)
```

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

```
rmse_results
```

```
## # A tibble: 1 x 2
##   method              RMSE
##   <chr>              <dbl>
## 1 Taking the average  1.06
```

The first model assumed that the rating of an unknown movie would be the mean value for all movie ratings in the dataset. **Y_hat = mu_hat** This is logical in that an ‘okay’ movie that was neither good nor bad would be given a rating of 3 out of 5, the value in the middle for an okay movie. The RMSE indicated that this was a good starting point for prediction.

Model 2: Movie Effect

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

predicted_ratings <- mu_hat + validation %>%
  left_join(movie_avgs, by='movieId') %>%
  .$b_i

model_2_rmse <- RMSE(predicted_ratings, validation$rating)
rmse_results <- bind_rows(rmse_results,
                          data_frame(method="Movie Rating Effect Model",
                                     RMSE = model_2_rmse ))
rmse_results %>% knitr::kable()
```

method	RMSE
Taking the average	1.0612018
Movie Rating Effect Model	0.9439087

The second model was based on the idea that movies with more ratings may reach a overall score that approaches the true value. If we use this, a better prediction may be possible. So now the model was as follows: **Y_hat = mu_hat + b_i** where *b_i* is rating of the ith movie in the dataset. This improved the RMSE score of the prediction.

Model 3: Movie + User Effect

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

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

model_3_rmse <- RMSE(predicted_ratings, validation$rating)
rmse_results <- bind_rows(rmse_results,
                          data_frame(method="Movie + User Effects Model",
                                     RMSE = model_3_rmse ))
rmse_results %>% knitr::kable()
```

method	RMSE
Taking the average	1.0612018
Movie Rating Effect Model	0.9439087
Movie + User Effects Model	0.8653488

The third model also takes into account that some users rate more and give higher ratings than others. To test the effect of user on predicting the outcome the following was used: **Y_hat = mu_hat + b_i + b_u** where *b_u* is rating of the ith movie given by that particular user. This reduced the RMSE score even further.

Model 4: Genre Effect

```
genre_avgs <- edx %>%
  group_by(userId) %>%
  summarize(b_g = mean(rating - mu_hat))

predicted_ratings <- mu_hat + validation %>%
  left_join(genre_avgs, by='userId') %>%
  .$b_g

model_4_rmse <- RMSE(predicted_ratings, validation$rating)
rmse_results <- bind_rows(rmse_results,
                          data_frame(method="Genre Effect Model",
                                     RMSE = model_4_rmse ))
```

Taking a step back, what if the prediction was simply based on the genre of the film? So this would assume that, for example, *Dramas* are rated higher than *Comedies* simply because they are in the *Drama* genre. This was tested as follows: **Y_hat = mu_hat + b_g** where *b_g* is the genre of the ith movie. This, however, produced the second highest RMSE. This is not surprising given that people naturally have preferences for different types of films. This Genre Effect Model essentially ignores the role of individual preference.

Final comparison of RMSE

The final list of RMSEs sorted by increasing value can be seen here.

```
rmse_results %>%
  arrange(RMSE) %>%
  knitr::kable()
```

method	RMSE
Movie + User Effects Model	0.8653488
Movie Rating Effect Model	0.9439087
Genre Effect Model	0.9783360
Taking the average	1.0612018

The best model (lowest RMSE) was a prediction based on the combination of movie and user effects, with RMSE of 0.865. Using the genre in Model 4 was actually worse at predicting compared to using the average movie rating on its own. Perhaps these genre data are better fit using a non-linear model, or should be combined together with the Movie + User models. However, modeling the fit of these data using the **train** function was beyond the available computer processing capacity. Examples of the code for these fits are given as comments in the code but running those is not advisable.

Conclusion

With this approach it as possible to predict the outcome *Movie Rating* using machine learning approaches and data from previous ratings. A final RMSE of 0.865 was achieved by using a model that took movie rating and user rating into account. This illustrated that the user's behaviour is important if the goal is to ultimately recommend a film that the user will like to watch later. But this can also be assisted by reviews of others put together (the movie effect) which would ultimately assess the true rating.