MovieLens Project

Syed Imran

June 14, 2019

Executive Summary

Recommendation systems use ratings that *users* have given *items* to make specific recommendations. In this project, we use a smaller subset of a dataset provided by Grouplens research lab for movie ratings for over 27,000 movies left by 138,000 users. *Movies* are *rated* by giving 0 - 5 stars by *users*. The *genres* of the movies is also provided in the dataset. The dataset was partitioned into a training set **edx** and a testing set **validation**. All the models were developed using the training set and evalated based on the **RMSE** of the errors in prediction on the **validation** set. For the recommendation model, we used a 3-step approach. First we asssumed that recommendations followed a **naive mean** of all movie ratings, then added **movie effect** and **users effect** to improve the model **RMSE**. A summary of the results is shown in the table below.

Method	RMSE
Naive average model	1.0606506
Movie Effect Model	0.9437046
Movie + User Effects Model	0.8655329

Based on the results the Movie + User Effects Model is chosen as our recommendation model as it has the **RMSE** below the target.

Project Overview

MovieLens

This project is a requirement for the HarvardX: PH125.9x Data Science: Capstone course offered on Edx.org. For this project, we will be creating a movie recommendation system using the MovieLens dataset. The version of movielens is a smaller subset of a much larger dataset with millions of ratings. The entire latest MovieLens dataset is available here https://grouplens.org/datasets/movielens/latest/.

Problem Statement

In this project, we will be creating a recommendation system for movies using the tools learned during the courses in this series. To facilitate computation a smaller 10M version of the MovieLens dataset will be used.

Methods and Analysis

Loading the required libraries

The analysis was conducted on R version 3.6.0 (2019-04-26). First, we load the **tidyverse** and **caret** libraries required for the analysis. The code will check if the required libraries are installed on the users system. If the required libraries are installed, they will be loaded. If the required libraries are not installed, they will be downloaded from the **CRAN** archive, installed and loaded.

```
# Check if the required libraries are installed.
# Load these libraries if they are installed.
# If the libraries are not installed, download and
# install the libraries from the CRAN archive.

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")
```

Movielens Dataset

From the Movielens Dataset description we learn that this data set contains 10000054 ratings and 95580 tags applied to 10681 movies by 71567 users of the online movie recommender service MovieLens. Users were selected at random for inclusion. All users selected had rated at least 20 movies. Unlike previous MovieLens data sets, no demographic information is included. Each user is represented by an id, and no other information is provided.

Downloading Dataset

The following instructions were provided to download and create the test and training sets: You will use the following code to generate your datasets. Develop your algorithm using the edx set. For a final test of your algorithm, predict movie ratings in the validation set as if they were unknown. RMSE will be used to evaluate how close your predictions are to the true values in the validation set.

The required subset of the Movielens data set is downloaded from http://files.grouplens.org/datasets/movielens/ml-10m.zip. The downloaded zip file contains three different datasets *movies*, *ratings* and *tags*. The *ratings* and *movies* datasets are extracted and meaningful names are assigned to the column fields. It is observed that *movieId* field is common to both datasets. This field can therefore be used as a key to join both datasets. The joined dataframe is named *movielens*.

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

# Download raw data set
#if(!file.exists(dl))
{
dl <- tempfile()
download.file("http://files.grouplens.org/datasets/movielens/ml-10m.zip", dl)
}
# Extract the ratings file from the dataset and give user friendly names to columns.

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

# Extract the movies file from the dataset and give user friendly names to columns.

movies <- str_split_fixed(readLines(unzip(dl, "ml-10M100K/movies.dat")), "\\::", 3)
colnames(movies) <- c("movieId", "title", "genres")

#convert to dataframe
```

```
movies <- as.data.frame(movies) %>% mutate(movieId = as.numeric(levels(movieId))[movieId],
                                           title = as.character(title),
                                           genres = as.character(genres))

# Join the two files into a single file using the "movieId" field as key.

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

Creating a validation and training set

The *movielens* dataframe created in the above step is partitioned into two dataframes. The first dataframe contains 90% of the data and will be used as the training set and named as *edx*. The second dataframe is the test set that will contain 10% of the data and named as *validation*.

```
set.seed(1)
#set.seed(1, sample.kind = "Rounding")
# if using R 3.6.0: 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,]

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)
```

Exploratory Data Analysis

The *edx* dataframe is used for the training set. A summary of the dataframe is given below:

```
glimpse(edx, width = 50)
```

```
## Observations: 9,000,061
## Variables: 6
## $ userId    <int> 1, 1, 1, 1, 1, 1, 1, 1, 1, ...
## $ movieId   <dbl> 122, 185, 231, 292, 316, 32...
## $ rating    <dbl> 5, 5, 5, 5, 5, 5, 5, 5, 5, ...
## $ timestamp <int> 838985046, 838983525, 83898...
## $ title     <chr> "Boomerang (1992)", "Net, T...
## $ genres    <chr> "Comedy|Romance", "Action|C...
```

It can be observed that the *edx* dataframe has 9000061 observations of 6 variables. Also we notice that the variables *userId* and *timestamp* are of type int (discreet); *movieId* and *rating* are of type dbl (continuous);

and *titles* and *genres* are of type character (string or text). Additionally, the *genres* variable is a composite character variable with many genres that are saperated by a |. This indicates that one movie may be categoried under a number of genres (for instance, RomComs or Sci-fi thriller dramas).

There are 69878 users in the *edx* dataframe. The number of movies that were rated are 10677. The *rating* is between 0.5 and 5.

Is there an effect of movie genres on ratings?

```
# List of all genres of movies in the dataset.

genres_list <- c("Action", "Adventure", "Animation","Children","Comedy",
                 "Crime","Drama", "Fantasy", "Film-Noir", "Horror", "Musical",
                 "Mystery","Romance", "Sci-Fi", "Thriller", "War", "Western")

#instantiate and initialize empty objects which will be used in the for loop.

n_genre <- NULL; mean_rating <- NULL; sd_rating <- NULL

# For loop to loop over the genres in the genre list to see basic stats of
# different genres

for(genre in genres_list) {
  index <- str_detect(edx$genres, pattern = genre)
  n_genre[genre] <- sum(index)
  mean_rating[genre] <- mean(edx$rating[index])
  sd_rating[genre] <- sd(edx$rating[index])
}

# Compile the outcome from the for loop

genre_summary <- tibble(genres_list, n_genre, mean_rating, sd_rating)
```

The first variable of interest is the genre of the movie. It could be possible that some genres are highly preferable to most users and thus highly rated. There are 17 different movie genres in the dataset. These are:

Action, Adventure, Animation, Children, Comedy, Crime, Drama, Fantasy, Film-Noir, Horror, Musical, Mystery, Romance, Sci-Fi, Thriller, War, Western

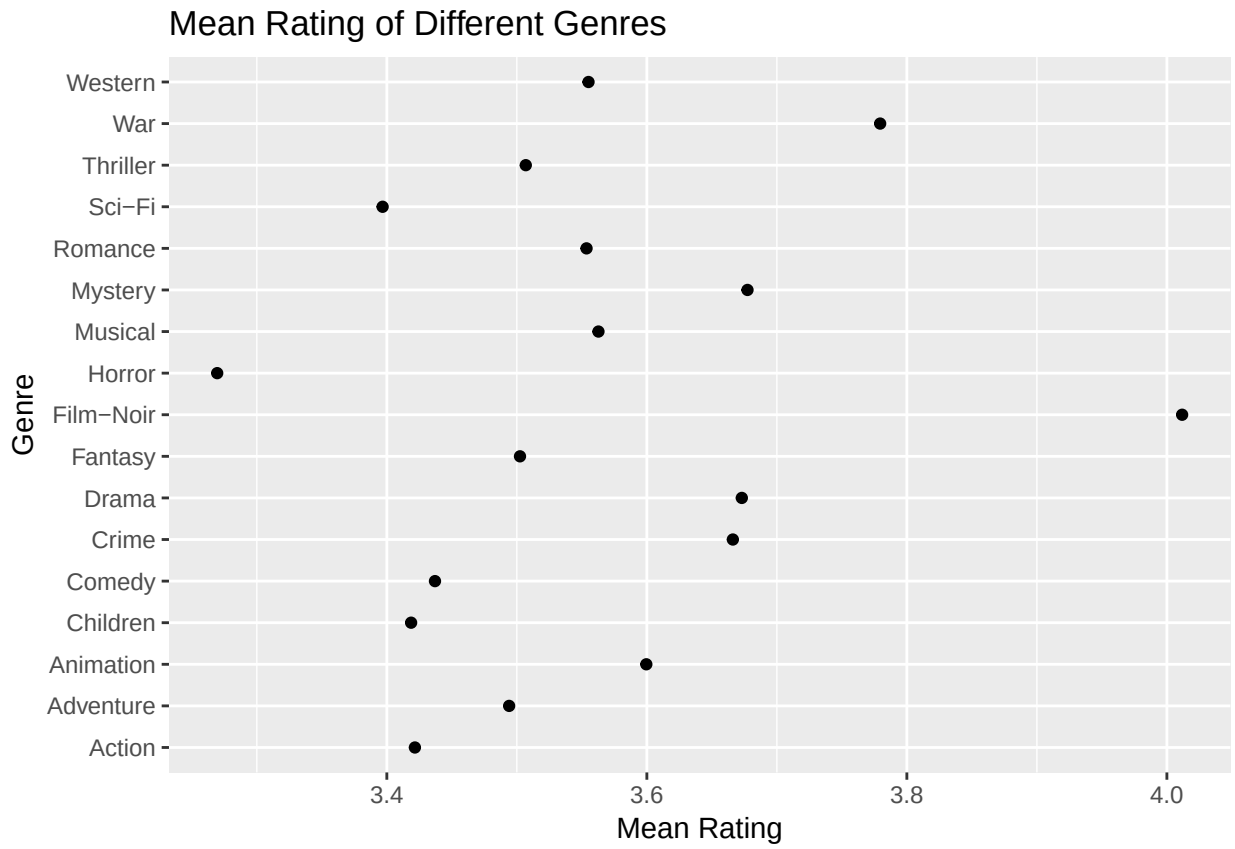
Before we begin our analysis, we need to evaluate whether the dataset is balanced (in terms of the numbers of ratings across the genres and their relative ratings). From the table below, we notice that there are a significant number of ratings for each genre. The average rating is also similar (between 3 and 4) and a standard deviation of around 1.

Table 2: Summary of Movie Genre Ratings

genres_list	n_genre	mean_rating	sd_rating
Action	2560649	3.421589	1.0664580
Adventure	1908692	3.494076	1.0529224
Animation	467220	3.599588	1.0198550
Children	737851	3.418673	1.0924888
Comedy	3541284	3.437040	1.0746937

genres_list	n_genre	mean_rating	sd_rating
Crime	1326917	3.666151	1.0114958
Drama	3909401	3.673047	0.9955311
Fantasy	925624	3.502419	1.0651920
Film-Noir	118394	4.011732	0.8870370
Horror	691407	3.269523	1.1502138
Musical	432960	3.562761	1.0573655
Mystery	567865	3.677412	0.9997424
Romance	1712232	3.553594	1.0302940
Sci-Fi	1341750	3.396756	1.0927568
Thriller	2325349	3.506879	1.0312087
War	511330	3.779457	1.0130096
Western	189234	3.555122	1.0236112

```
genre_summary %>% ggplot(aes(y=genres_list, x=mean_rating)) + geom_point() +
  labs(title ="Mean Rating of Different Genres", x = "Mean Rating", y = "Genre")
```



The maximum mean rating of 4.011732 was given to the genre Film-Noir. The minimum mean rating of 3.2695229 was given to the genre Horror. This result is encouraging as it indicates that the ratings are not highly influenced by individual genres, since there is not a lot of difference in the mean ratings for the highest rated and the lowest rated genres.

Predictive Recommendation Models

Naive Recommendation

The naive model is the simplest possible model that assumes that the movie will have the same rating regardless of genre or user. The estimate that minimizes the RMSE in this case is the average of all the movies μ with $\epsilon_{u,i}$ independent errors.

$Y_{u,i} = \mu + \epsilon_{u,i}$

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

#What would be the RMSE if we took the rating equal to 3.
predictions <- rep(3, nrow(validation))
Mid_value_RMSE <- RMSE(validation$rating, predictions)

rmse_results <- tibble(method="Naive average model", RMSE=naive_rmse)
```

In this case, the mean rating $\hat{\mu} = 3.512464$ is set as the expected rating for all movies. The RMSE for the naive model is 1.0606506.

Modeling Movie Effects

We can augment the previous naive model by adding another parameter b_i to represent average ranging for a movie i.

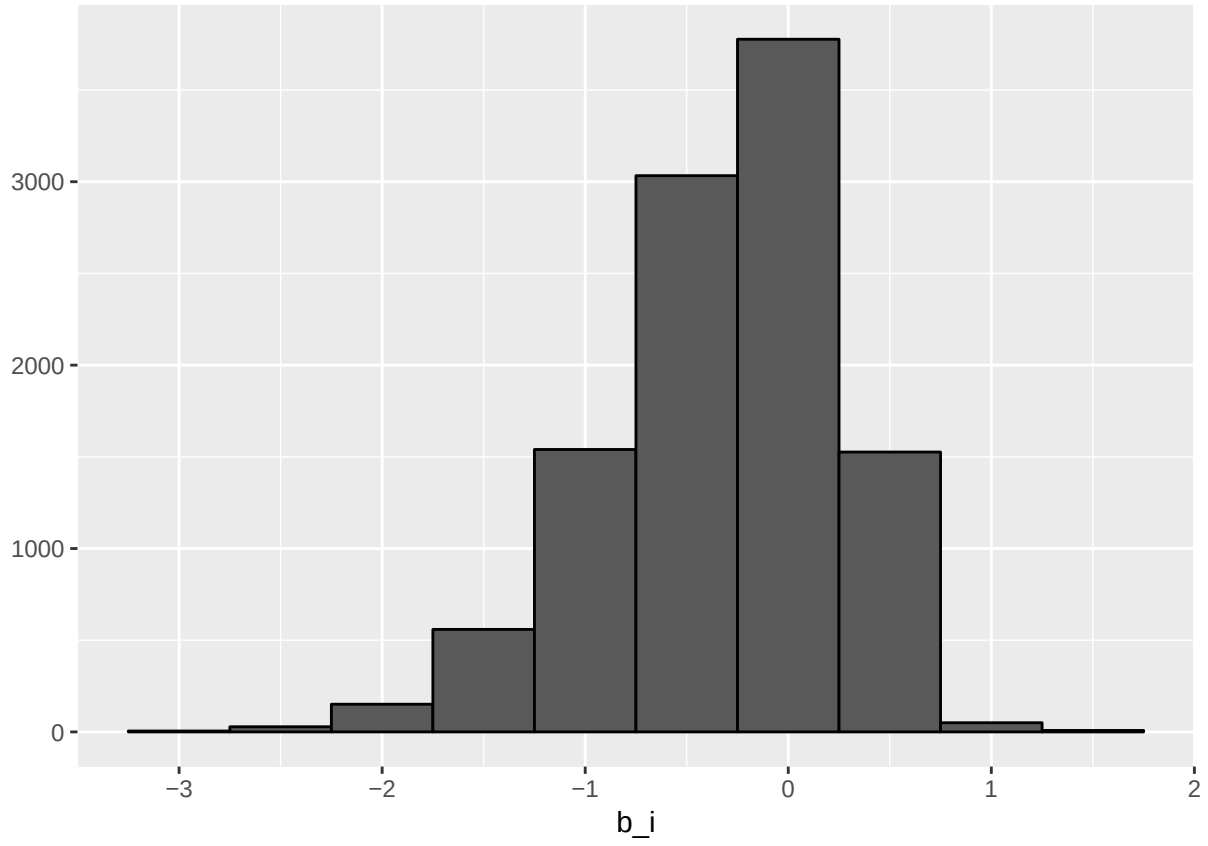
$Y_{u,i} = \mu + b_i + \epsilon_{u,i}$

Since there are 10677 movies in **edx**, we will get 10677 unique b_i's. In this case, we have to evaluate the least square estimate of b_i. This is the average of $Y_{u,i} - \hat{\mu}$ for each movie i.

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

These estimates vary substantially, indicating that the inclusion of the movie effect can improve the recommendation model.

```
movie_avgs %>% qplot(b_i, geom = "histogram", bins = 10, data = ., color = I("black"))
```



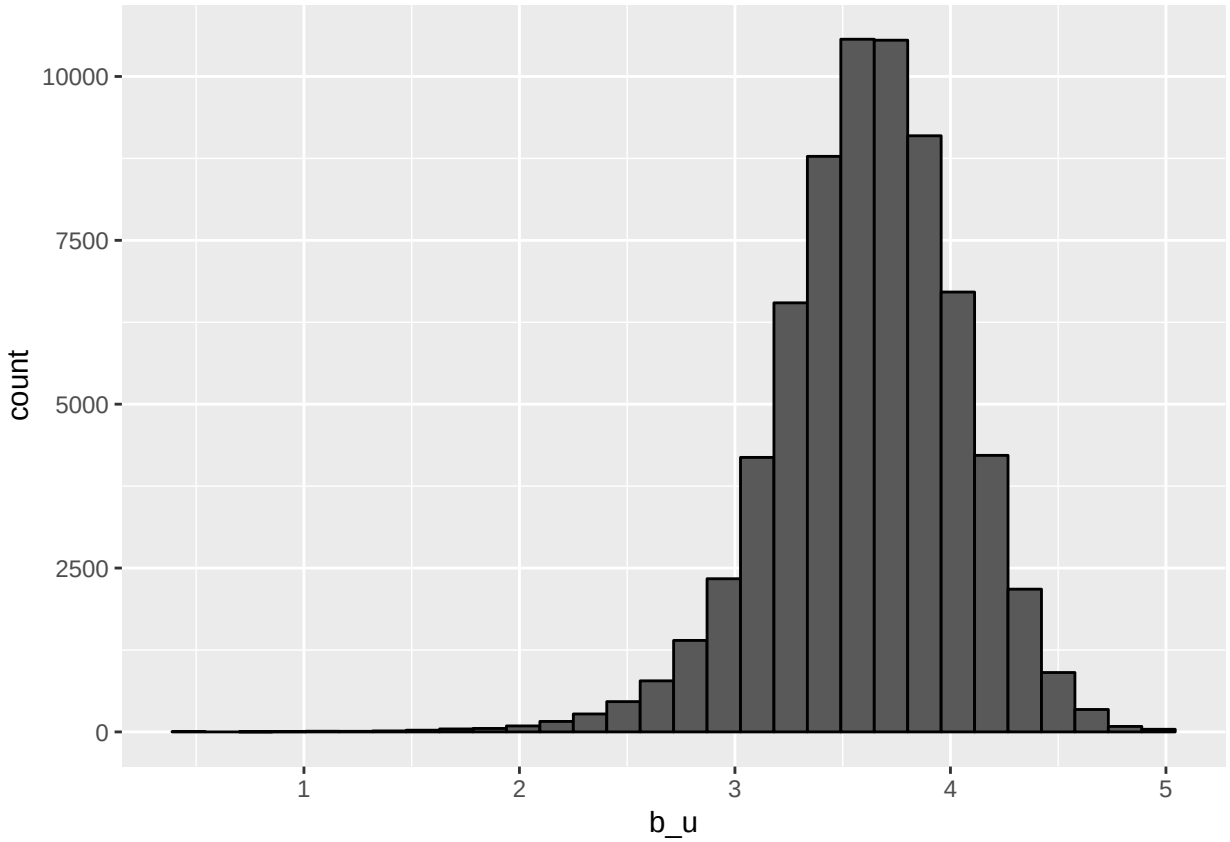

```
predicted_ratings <- mu_hat + validation %>%
  left_join(movie_avgs, by='movieId') %>%
  pull(b_i)

model_1_rmse <- RMSE(predicted_ratings, validation$rating)
rmse_results <- bind_rows(rmse_results, tibble(method ="Movie Effect Model",
                                               RMSE=model_1_rmse))
```

By incorporating the movie effects b_i, we get a RMSE of 0.9437046. This is an improvement over the naive model. ‘r knitr:: kable(rmse_results, title=“Summary of RMSE from Models”)

Modeling User Effects

We now include the user u effects to evaluate whether we can see an improvement in our model.

```
edx %>%
  group_by(userId) %>%
  summarize(b_u = mean(rating)) %>%
  filter(n()>=100) %>%
  ggplot(aes(b_u)) +
  geom_histogram(bins = 30, color ="black")
```



This indicates that there is a substantial variability across users as well: which implies that a further improvement in our model can be obtained by incorporating user effect b_u.

$Y_{u,i} = \mu + b_i + b_u + \epsilon_{u,i}$

We will compute an approximation by computing $\hat{\mu}$ and $\hat{b}_i$ and estimating $\hat{b}_u$ as average of $y_{u,i} - \hat{\mu} - \hat{b}_i$.

```
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) %>%
  pull(pred)

model_2_rmse <- RMSE(predicted_ratings, validation$rating)
rmse_results <- bind_rows(rmse_results, tibble(method = "Movie + User Effects Model",
                                               RMSE = model_2_rmse))
```

By incorporating the user effects b_u, we get a RMSE of 0.8655329. This is an improvement over the naive model as well as the Movie effect model.

Conclusions and Recommendations

For the recommendation model, we used a 3-step approach. First we asssumed that recommendations followed a **naive mean** of all movie ratings, then added **movie effect** and **users effect** to improve the model **RMSE**. A summary of the results is shown in the table below.

method	RMSE
Naive average model	1.0606506
Movie Effect Model	0.9437046
Movie + User Effects Model	0.8655329

Based on the results the Movie + User Effects Model is chosen as our recommendation model as it has the **RMSE** below the target.