HarvardX_MovieLens_Capstone_Project

Roan Indra

5/23/2020

1 1. Introduction

The MovieLens data set by the GroupLens Research Group contains 10,000,054 ratings and 95,580 tags applied to 10,681 movies by 71,567 users of the online movie recommender service MovieLens.

Users were selected at random for inclusion. All users selected had rated at least 20 movies. Each user is represented by an id, and no other information is provided.

The goal of the project is to build a machine learning model that can predict the rating of a movie. To that end, we are going to: - Explore and study the data; - Determine the dependent variables that have significant influence over the rating; and - Propose the machine learning algorithm that has the least RSME

##Reference: MovieLens 10M dataset - About the data set - The data set

#2. Method / Analysis

1.1 a. Data Pre-Processing: Create edx set, validation set

The steps performed here are exactly the same the steps recommended for the Quiz section. We will first ensure that the following packages are installed: tidyverse, caret, and data.table.

```
# 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
```

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

```
## v ggplot2 3.3.0     v purrr   0.3.3
## v tibble  3.0.0     v dplyr   0.8.5
## v tidyr   1.0.2     v stringr 1.4.0
## v readr   1.3.1     v forcats 0.5.0
```

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

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

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

```
## Loading required package: lattice

##
## Attaching package: 'caret'

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

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

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

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

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

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

Next, we have to download the data set.

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

Once we have downloaded the data set, we have to extract the rating data.

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

Afterwards, we have to tidy up the data.

```
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(title),
                                            genres = as.character(genres))
movielens <- left_join(ratings, movies, by = "movieId")
```

The validation set will be 10% of the MovieLens data. We have to set seed to ensure consistent results.

```
set.seed(1, sample.kind="Rounding")
```

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

```r
# if using R 3.5 or earlier, use `set.seed(1)` instead
test_index <- createDataPartition(y = movielens$rating, times = 1, p = 0.1, list = FALSE)
 edx <- movielens[-test_index,]
 temp <- movielens[test_index,]
```

We have to make sure that we don't include users and movies in the test set that do not appear in the training set. Hence, we remove these entries using the semi_join function.

```r
validation <- temp %>%
semi_join(edx, by = "movieId") %>%
semi_join(edx, by = "userId")
```

To ensure that we don't lose any information, we add rows excluded from the validation set to the previous data set, which is edx.

```r
# Add rows removed from validation set back into edx set
removed <- anti_join(temp, validation)
```

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

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

1.2 b. Data Exploration and Visualisation

We are now ready to explore edx, which will be the focus in building our machine learning model. Let's start by taking a peek of the data and examine the structure.

```r
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
```

```r
str(edx)
```

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

Before we can proceed with our data analysis, it is clear that we have to convert userId and movieId into factors, as well as convert the timestamp into date and time.

```
edx$userId <- as.factor(edx$userId)
edx$movieId <- as.factor(edx$movieId)
edx$timestamp <- as.POSIXlt(edx$timestamp,  origin="1970-01-01")
```

Let's check if those three variables have been converted into the desired classes.

```
summary(edx)
```

```
##      userId           movieId           rating
##  59269  :   6616   296    :  31362   Min.   :0.500
##  67385  :   6360   356    :  31079   1st Qu.:3.000
##  14463  :   4648   593    :  30382   Median :4.000
##  68259  :   4036   480    :  29360   Mean   :3.512
##  27468  :   4023   318    :  28015   3rd Qu.:4.000
##  19635  :   3771   110    :  26212   Max.   :5.000
##  (Other):8970601   (Other):8823645
##    timestamp                      title              genres
##  Min.   :1995-01-09 19:46:49   Length:9000055     Length:9000055
##  1st Qu.:2000-01-02 07:11:23   Class :character   Class :character
##  Median :2002-10-25 05:11:58   Mode  :character   Mode  :character
##  Mean   :2002-09-21 21:45:07
##  3rd Qu.:2005-09-15 10:21:21
##  Max.   :2009-01-05 13:02:16
##
```

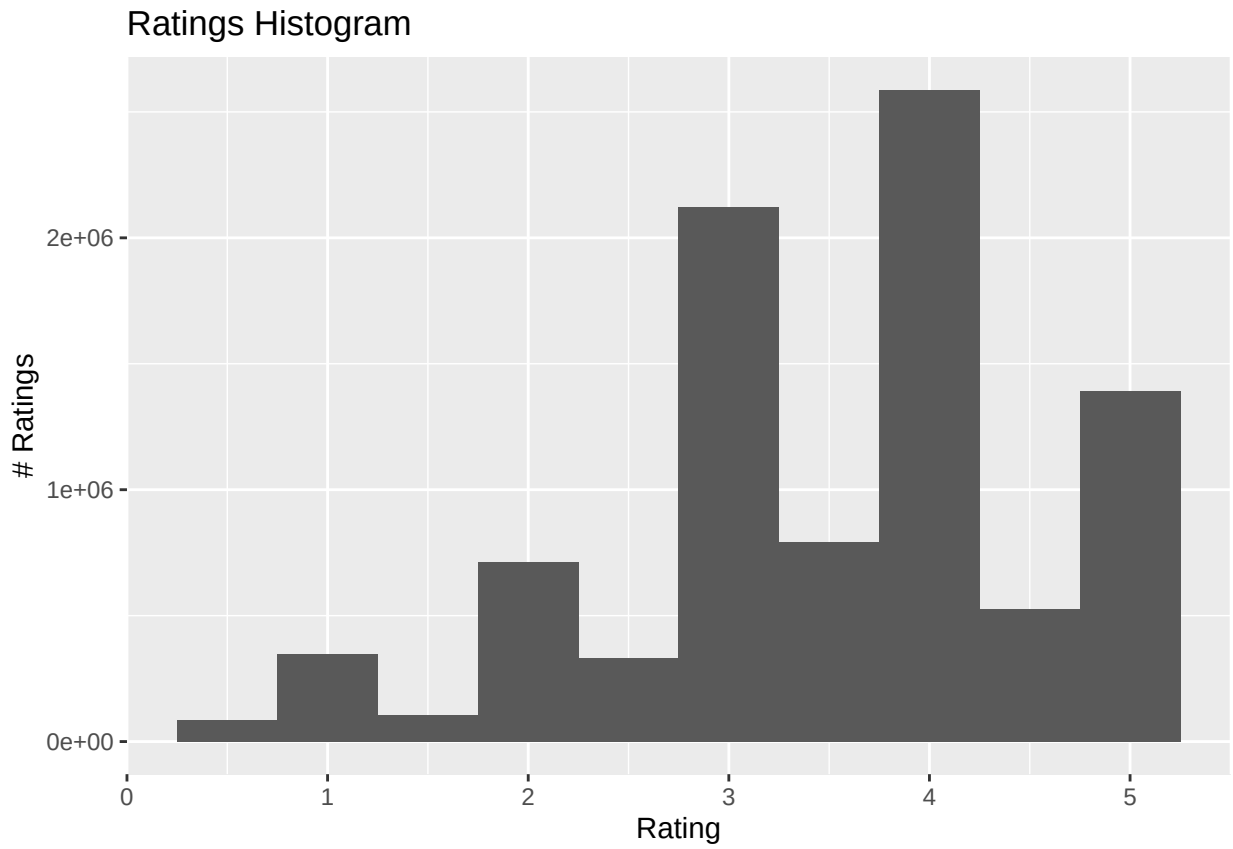
The data is now ready to be used for building a machine learning model. Let's create training and test sets.

```
#Creating an additional partition of training and test sets from the edx dataset so as to not to use th
test_index2 <- createDataPartition(y = edx$rating, p=0.2, list = FALSE)
train_set <- edx[-test_index2,]
test_set <- edx[test_index2,]

#removing unnecessary variables
rm(movies, removed, temp)

#To make sure that we don't include users and movies in the test set that do not appear in the training
test_set <- test_set %>%
semi_join(train_set, by = "movieId") %>%
semi_join(train_set, by = "userId")
```

This time round, there is no need to add rows removed back to the edx dataset as we are not going to make further partition.

It's time to study the distribution of the data so that we could determine the machine learning algorithm that would be best for training and testing the data. Let's now visualise the data, particularly the rating distribution in edx.

```r
# explore ratings distribution
edx %>% ggplot() +
  geom_histogram(mapping = aes(x = rating), binwidth = 0.5) +
  xlab("Rating") + ylab("# Ratings") + ggtitle("Ratings Histogram")
```



We can see that the distribution is approxomately a normal distribution. So **the linear regression model** can be used as the machine learning model with reasonable performance.

#3. Results Before we can build our model, we need some way to evaluate our model performance. This can be done via RMSE, which is part of this capstone project deliverables.

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

RMSE is benchmarked against the mean. So let's create the simplest model by assuming the predicted rating is equal to the mean.

```r
mu_hat <- mean(train_set$rating)
mu_hat
```

```
## [1] 3.512574
```

Let's evaluate the performance of this "mean-based" model using the RMSE.

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

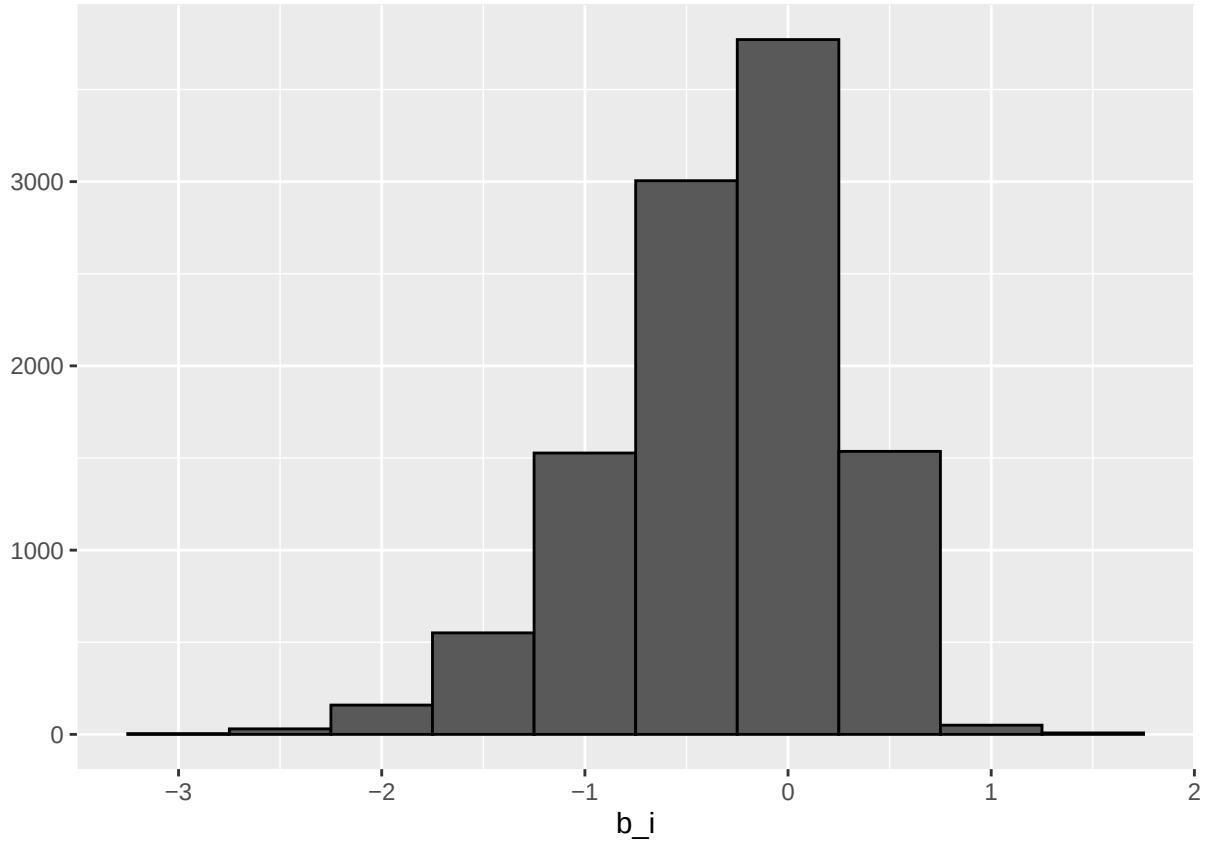
```
## [1] 1.060704
```

We get 1.06. We can definitely do better by using the linear regression. From our data pre-processing steps, we can surmise that **movieId** and **userId** seems to be most influential dependent variables. Let's quickly confirm that.

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

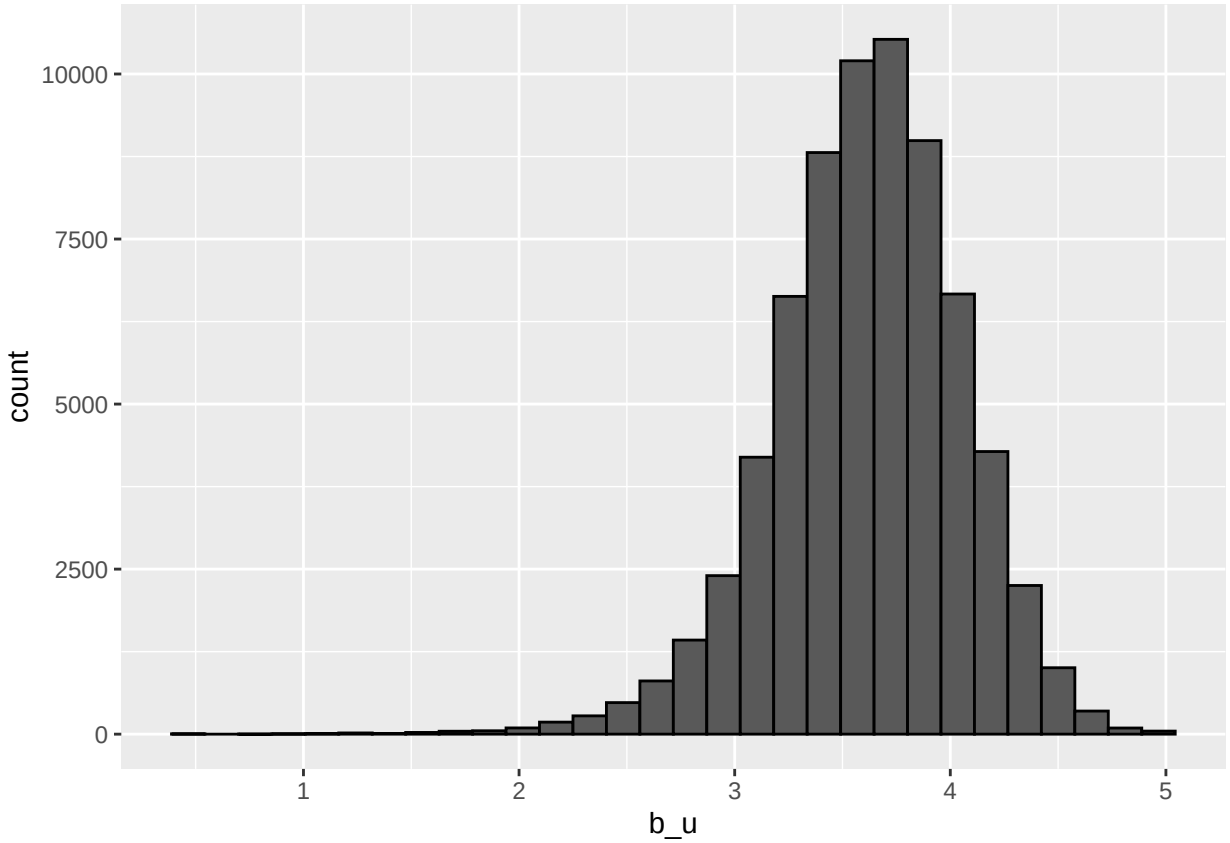
Let's see if these estimates vary.

```
qplot(b_i, data = movie_avgs, bins = 10, color = I("black"))
```



Yes, we have confirmed that the estimates for movieId vary. Let's quickly confirm the same for userId.

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



It's clear that the estimates for userId vary too.

So our linear regression model would be: lm(rating ~ movieId + userId). However, the dataset is a large one, we are not able to run the regression model (R will crash!). Instead, we will examine the model using approximation. Here is the performance of the proposed model.

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

```
predicted_ratings <- test_set %>%
left_join(movie_avgs, by='movieId') %>%
left_join(user_avgs, by='userId') %>%
mutate(pred = mu + b_i + b_u) %>%
pull(pred)
RMSE(predicted_ratings, test_set$rating)
```

```
## [1] 0.8661625
```

Our linear regression model beats the "mean-based" model!

It's time for use to compute the final RMSE, using the validation dataset as benchmark. Let's use the similar approach as before, i.e. start by calculting the RMSE for a "mean-based" model.

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

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

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

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

We get 1.06, which is the same as the one that we got using the smaller data set. Let's see if we get the same results if we benchmark against the validation data set.

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

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

```
predicted_ratings <- validation %>%
left_join(movie_avgs, by='movieId') %>%
left_join(user_avgs, by='userId') %>%
mutate(pred = mu + b_i + b_u) %>%
pull(pred)
RMSE(predicted_ratings, validation$rating)
```

```
## [1] 0.825177
```

0.825177 is our final RMSE using lm(rating ~ movieId + userId) model, which outperforms the ealier RMSE of 0.8665681.

#4. Conclusion We have shown that we can predict movie rating based on movieId and userId, and that it is possible to build such a model using linear regression. We have also shown that a linear regression model can beat prediction using the mean.

There is scope to further explore better machine learning algorithms and tuning such algorithms for improved performance. Alas, this report is limited in that sense, as R is unable to compute a machine learning model using such a large dataset (close to 10M observations).

Future work could use Hadoop or equivalent to explore such a large dataset and determine the machine learning model that can best predict the movie rating. There could be other dependant variables that are statistically significant that can be added to the model to improve its predictive accuracy.