Capstone Project I

Berkeley Loveless

5/20/2019

HarvardX - PH125.9x: Data Science: Capstone

Summary

This project is defined as creating a recommendation system for movies. The data set used is the movielens data set. The requirement is to predict movie ratings on a verification data set and test with Residual Means Square Error (RMSE). To achieve the maximum points for accuracy, the RMSE must be less than .87750. My personal goal is to have an RMSE of one percentage point better than the requirement (.86750). If I achieve that I will stop as I am new to R programming and I will need as much time as possible on the second Capstone project.

Intent

I intend to follow the same process for the recommendation system as was demonstrated in the Machine Learning class: HarvardX - PH125.8x: Data Science: Machine Learning. This involves:

- Setting up a base case where the movies were all given the average rating.
- Calculating the Residual Mean Square Error (RMSE) of the base case and then each case going forward.
- Using biases starting with movie bias and then user bias, and other bias and possibly clustering as necessary to refine predictions that will achieve the goal. Upon achieving the goal, I will stop for the above reasons.

Code to extract and transform the data used for this project was provided in the exercise instructions.

R version 3.6.0 was used for this project.

The data include a list of movies, individual users who rate movies, and the ratings given for each movie by these users. Other data include a time stamp and a movie genre. Detailed descriptions of the files and structures used is contained in the body of the project.

Conclusion

The model began with $Y_{u,i} = \mu + \epsilon_{u,i}$ predicting the average movie rating for all movies and added movie bias b_i and then user bias b_u to end up with a final model of $Y_{u,i} = \mu + b_i + b_u + \epsilon_{u,i}$ which was sufficient to meet the goal with an RMSE of .8651799.

The following code and text were included in the excercise instructions:

Create edx set, validation set, and submission file

Note: this process could take a couple of minutes

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

```
## Loading required package: tidyverse
## Registered S3 methods overwritten by 'ggplot2':
##   method         from
##   [.quosures     rlang
##   c.quosures     rlang
##   print.quosures rlang
```

```
## -- Attaching packages --------------------------------------------------- tidyverse 1.2.1 --
```

```
## v ggplot2 3.1.1     v purrr   0.3.2
## v tibble  2.1.1     v dplyr   0.8.1
## v tidyr   0.8.3     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
```

```
## Loading required package: lattice
```

```
##
## Attaching package: 'caret'
```

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

MovieLens 10M dataset:

https://grouplens.org/datasets/movielens/10m/

http://files.grouplens.org/datasets/movielens/ml-10m.zip

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

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))[movieId],
                                           title = as.character(title),
                                           genres = as.character(genres))

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

Validation set will be 10% of MovieLens data

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

```
## 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)
```

# This concludes the code given in the excercise instructions

---

Visually checking out the data

The data sets given for the project are subsets of the Movielens data set. The training file is named *edx* and a subset of 10% of edx is the test file named *validation*. I will examine the training set, knowing that the test set is a true subset of the training set.

```
glimpse(edx)
```

```
## Observations: 9,000,055
## Variables: 6
## $ userId    <int> 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1...
## $ movieId   <dbl> 122, 185, 292, 316, 329, 355, 356, 362, 364, 370, 37...
## $ rating    <dbl> 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5...
## $ timestamp <int> 838985046, 838983525, 838983421, 838983392, 83898339...
## $ title     <chr> "Boomerang (1992)", "Net, The (1995)", "Outbreak (19...
## $ genres    <chr> "Comedy|Romance", "Action|Crime|Thriller", "Action|D...
```

The file includes nine million observations with six variables. Variables include *userId*, an **integer** that will be used to identify each user, *movieId*, a **double** that will be used to identify each movie, *rating*, a **double**, that will be used to identify the rating each movie is given by each user, *timestamp*, a **double** that could be used to tune a model, *title*, a **character** type that is the title of the movie which could be used in clustering analysis and *genres* that is type **character** that will be used to classify the movie types. Since *genres* will be used for classification, I will change they type to **factor** in both data sets.

Turn *genres* into factors

Turning *genres* into **factors** allows for treating the genres and genre combinations as discrete element for classification.

```r
edx <- mutate(edx, genres = as.factor(genres))
validation <- mutate(validation, genres = as.factor(genres))
```

Testing to make sure *genres* are set as factors

```r
class(edx$genres)
```

```
## [1] "factor"
```

```r
class(validation$genres)
```

```
## [1] "factor"
```

Building the Recommendation System

The basic idea is to follow the same process for the recommendation system as was demonstrated in the Machine Learning class. Basically this involved:

- Setting up a base case where the movies were all given the average rating
- Subtracting biases
- Alliteratively check predictions and record the RMSE

Residual Mean Squared Error Function

Using the formula $RMSE = \sqrt{\frac{1}{N} \sum_{u,i}(\hat{y}_{u,i} - y_{u,i})^2}$

The function is built this way:

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

Setting up the baseline

The base case predicts the mean rating regardless of user or movie.

Getting the mean rating and storing it as *mu*

```r
mu <- mean(edx$rating)
mu
```

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

Buildng the basline predicting the average rating overall, regardless of movie or user.

The prediction model begins as $Y_{u,i} = \mu + \epsilon_{u,i}$

```r
predictions <- rep(mu, nrow(edx))
```

Calculating the R mean squared error of the prediction

```r
base_rmse <- RMSE(edx$rating, predictions)
```

Creating a table to track progress and adding the RMSE to the running total in that table.

```r
rmse_results <- tibble(method = "Just the average", RMSE = base_rmse)
```

Printing out the running total table

```r
rmse_results %>% knitr::kable()
```

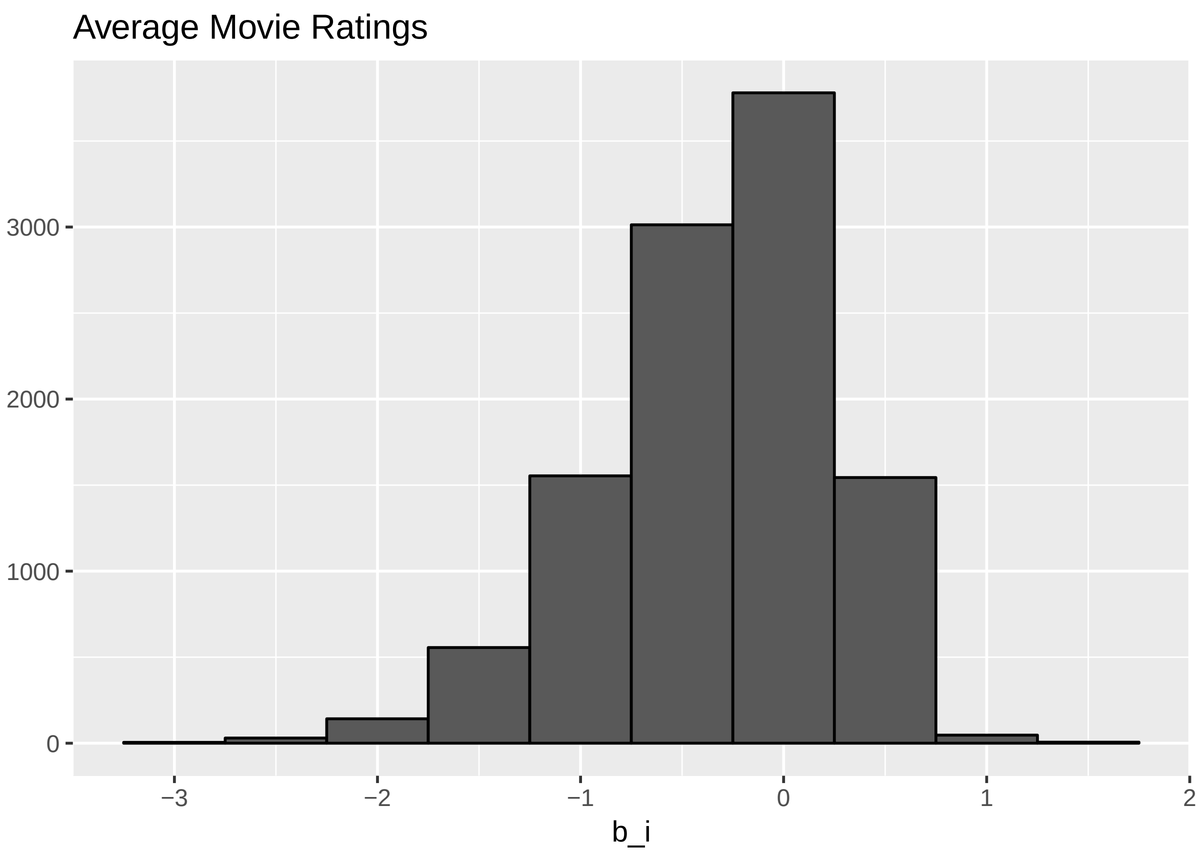
method	RMSE
Just the average	1.060331

Getting average rating delta from the mean, per movie

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

Checking out the movie ratings by histogram

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



The histogram shows there are biases by movie so using the movie bias $Y_{u,i} = \mu + b_i + \epsilon_{u,i}$ should improve the prediction.

Adding the μ to the movie delta, b_i and predicting rating by movieId

```
predicted_ratings <- mu + edx %>%
  left_join(movie_avgs, by='movieId') %>%
  .$b_i
```

Calculating the RMSE of the prediction

```
model_1_rmse <- RMSE(predicted_ratings, edx$rating)
```

Adding the RMSE to the running total

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

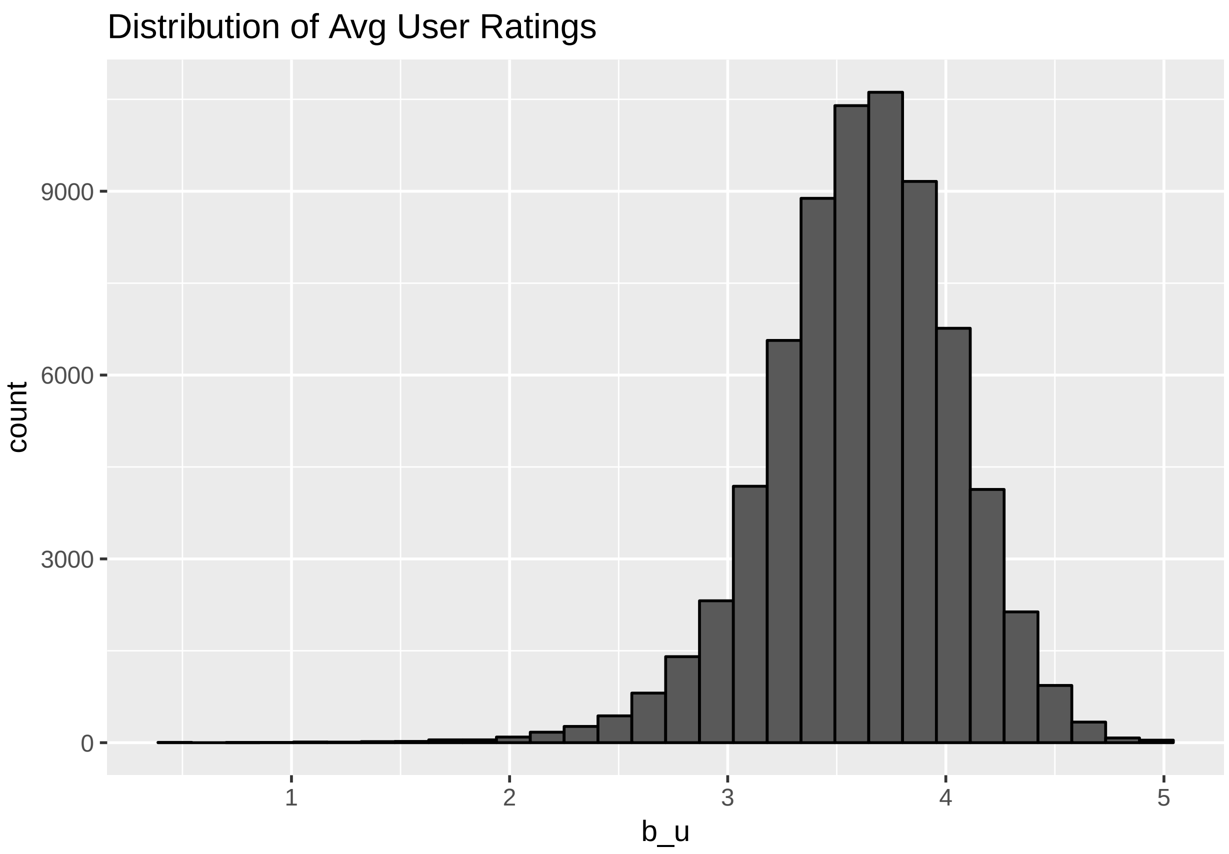
Printing out the running total table

```r
rmse_results %>% knitr::kable()
```

method	RMSE
Just the average	1.0603313
Movie Effect Model	0.9423475

Checking out the average rating by user

```r
edx %>%
  group_by(userId) %>%
  summarize(b_u = mean(rating)) %>%
  ggplot(aes(x=b_u)) +
  ggtitle("Distribution of Avg User Ratings") +
  geom_histogram(bins = 30, color = "black")
```



The histogram shows considerable variance in user ratings. Some have a negative bias, rating movies lower in general, and some have a positive bias, rating movies higher in general.

Getting the user bias by subracting the modified average rating (the average rating considering the movie title)

Instead of getting the user bias independently it uses the refinement of the movie title bias for a better result. in other words, building on the prior predictor rather than just identifying a separate one independently. Adding b__u should improve the model. $Y_{u,i} = \mu + b_i + b_u + \epsilon_{u,i}$

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

Joining the two biases together, calculating the RMSE, adding the results to the table and displaying the table.

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

model_2_rmse <- RMSE(predicted_ratings, edx$rating)
rmse_results <- bind_rows(rmse_results,
                          tibble(method="Movie + User Effects Model",
                                     RMSE = model_2_rmse ))
rmse_results %>% knitr::kable()
```

method	RMSE
Just the average	1.0603313
Movie Effect Model	0.9423475
Movie + User Effects Model	0.8567039

Finalizing the model

With an RMSE of .8567, I think the goal can be met with the final data set. As fun as this was and as curious as I am to experiment more, I have to give myself appropriate time to work on the "Choose Your Own!" project which will take considerable time with my lack of experience.

Checking final model against validation set

```
predicted_ratings_final <- validation %>%
  left_join(movie_avgs, by='movieId') %>%
  left_join(user_avgs, by='userId') %>%
  mutate(pred =(ifelse(mu + b_i + b_u <= 5, mu + b_i + b_u , 5))) %>%
  pull (pred)
```

Calculating the RMSE and adding the result to the table

```
model_3_rmse <- RMSE(validation$rating, predicted_ratings_final)
rmse_results <- bind_rows(rmse_results,
                          tibble(method="Final Model against validation",
                                     RMSE = model_3_rmse ))
```

Final Results

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

method	RMSE
Just the average	1.0603313
Movie Effect Model	0.9423475
Movie + User Effects Model	0.8567039
Final Model against validation	0.8651799

Conclusion

The final model meets the goals set in the Summary section by achieving a RMSE of .8652. The final model takes into consideration the bias of an individual rater - some rate movies higher than others, and the movie bias, where some movies are generally rated better than others. These are found by comparing ratings over the average rating in each case. As these to biases are considered together, they combine to counteract effect of users bias on movies to achieve a more likely rating. For example, combining the biases counteract a user with a negative bias rating a generally well rated movie to bring the prediction closer to being correct.