Goodreads Books

Ionut Costache

Introduction

The purpose of this analysis is to build a prediction model of average book ratings, starting from a data set which was downloaded in CSV format from https://www.kaggle.com/jealousleopard/goodreadsbooks

In preparing the analysis, the following steps were taken:

1. Initial review - reviewing the structure of the data set and understanding what data is available / missing
2. Separated the data into training / testing / validation sets
3. Defined the necessary transformations to the data to facilitate further analysis
4. Data exploration - with the help of the additional data transformations, summarize and visualize data and gather insights
5. Attempted various prediction models: linear regression, random forest as well as the prediction model explained in chapter "33.7 Recommendation systems" of the book (https://rafalab.github.io/dsbook/large-datasets.html#)
6. Calculated RMSE for all models and compare them
7. Summarized conclusions

Analysis

Let us begin by reviewing the structure of the dataset, after reading the CSV file.

```
##      bookID         title              authors          average_rating
##  Min.   :    1   Length:11127       Length:11127       Min.   :0.000
##  1st Qu.:10287   Class :character   Class :character   1st Qu.:3.770
##  Median :20287   Mode  :character   Mode  :character   Median :3.960
##  Mean   :21311                                         Mean   :3.934
##  3rd Qu.:32104                                         3rd Qu.:4.135
##  Max.   :45641                                         Max.   :5.000
##      isbn              isbn13          language_code        num_pages
##  Length:11127       Length:11127       Length:11127       Min.   :   0.0
##  Class :character   Class :character   Class :character   1st Qu.: 192.0
##  Mode  :character   Mode  :character   Mode  :character   Median : 299.0
##                                                           Mean   : 336.4
##                                                           3rd Qu.: 416.0
##                                                           Max.   :6576.0
##  ratings_count     text_reviews_count publication_date    publisher
##  Min.   :      0   Min.   :    0.0    Length:11127       Length:11127
##  1st Qu.:    104   1st Qu.:    9.0    Class :character   Class :character
##  Median :    745   Median :   46.0    Mode  :character   Mode  :character
##  Mean   :  17936   Mean   :  541.9
##  3rd Qu.:   4994   3rd Qu.:  237.5
##  Max.   :4597666   Max.   :94265.0
```

We can see that minimum value of average rating is 0 so let's see how many records we have with 0 rating

```
dataset %>% filter(average_rating == 0) %>% summarize(n = n())
```

```
## # A tibble: 1 x 1
##       n
##   <int>
## 1    26
```

There are only a few records, so let's remove them from the data set as they will not help in any way with the prediction.

```
dataset <- dataset %>% filter(average_rating != 0)
```

Now, let's split the data set into a training set, a testing set and a validation set. Based on multiple online sources, I found 70/15/15 to be ideal ratios. Here is the code:

```
set.seed(1983, sample.kind = "Rounding")
# We will first consider 85% for the training & testing partition set
# and 15% for the validation set
test_index_1 <- createDataPartition(y = dataset$average_rating, times = 1,
                                    p = 0.85, list = FALSE)
partition <- dataset[test_index_1,]
validation <- dataset[-test_index_1,]

# Now splitting partition into a training set and a testing set
test_index_2 <- createDataPartition(y = partition$average_rating, times = 1,
                                    p = 70/85, list = FALSE)
train_set <- partition[test_index_2,]
test_set <- partition[-test_index_2,]
```

Checking the number of rows in each of the sets, to make sure we obtained what we expected

```
nrow(train_set)
```

```
## [1] 7773
```

```
nrow(test_set)
```

```
## [1] 1664
```

```
nrow(validation)
```

```
## [1] 1664
```

Looks good. Let's see a few basic correlations within the data set

```
# checking correlation between number of pages and book rating
cor(train_set$average_rating, train_set$num_pages)
```

```
## [1] 0.1500027
```

```r
# checking correlation between number of ratings and book rating
cor(train_set$average_rating, train_set$ratings_count)
```

```
## [1] 0.03537656
```

```r
# checking correlation between number of reviews and book rating
cor(train_set$average_rating, train_set$text_reviews_count)
```

```
## [1] 0.03071217
```

We can see a light correlation between number of pages and average rating. The other two are very low.

Let's also calculate the text review rate, based on number of text reviews divided to number of ratings:

```r
# adding a new column to calculate review rate (% of people who posted a review)
# out of the people who rated it
temp <- train_set %>%
  mutate(review_rate = ifelse(ratings_count == 0, 0,
                              text_reviews_count / ratings_count))

# checking correlation between review rate and book rating
cor(train_set$average_rating, temp$review_rate)
```

```
## [1] -0.1386073
```

We can observe a light negative correlation between average rating and review rate.

Let's also calculate the length of the book's title and check correlation:

```r
# adding a new column to calculate length of the book's title
temp <- temp %>% mutate(title_length = nchar(title))

# checking correlation between length of the book's title and book rating
cor(train_set$average_rating, temp$title_length)
```

```
## [1] 0.1395912
```

A light correlation is also found here.

We've exhausted the numerical fields in the data set and we will avoid publication date which is unlikely to influence the average rating.

After adding review rate and title length to test_set, let's see what we obtain with these three variables and with a basic linear model.

```r
fit <- lm(average_rating ~ title_length + review_rate +
            num_pages, data = train_set)
y_hat_lm <- predict(fit, test_set)
sqrt(mean((y_hat_lm - test_set$average_rating)^2))
```

```
## [1] 0.2843622
```

Let's also try a RandomForest prediction with the same variables. We will add authors and language to the prediction model. Note that adding the publisher did not improve accuracy.

After attempting several values for ntree, we settled for a value of 180, after which the prediction doesn't seem to improve significantly

```r
# train model using randomForest
set.seed(1983, sample.kind = "Rounding")
train_rf <- randomForest(average_rating ~ title_length + review_rate +
                           num_pages + authors + language_group,
                         data = train_set, ntree = 180,
                         importance=TRUE)
# see importance of each variable
varImp(train_rf)
```

```
##                  Overall
## title_length   22.37904
## review_rate    21.81735
## num_pages      27.03140
## authors        14.49113
## language_group 23.12304
```

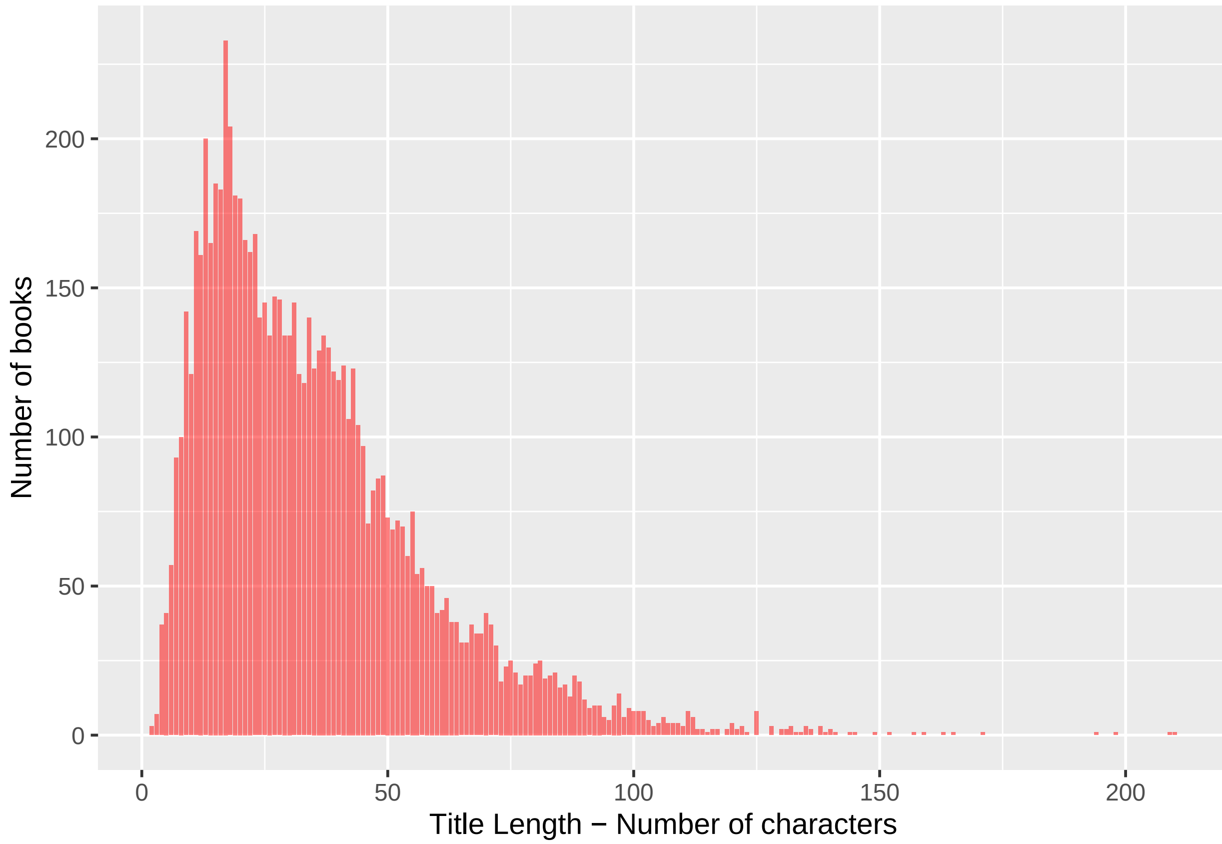
```r
# predict test set using RandomForest
y_hat_rf <- predict(train_rf, test_set)
# calculate RMSE
sqrt(mean((y_hat_rf - test_set$average_rating)^2))
```

```
## [1] 0.2771318
```

Now, let's try a prediction using the recommendation system approach.

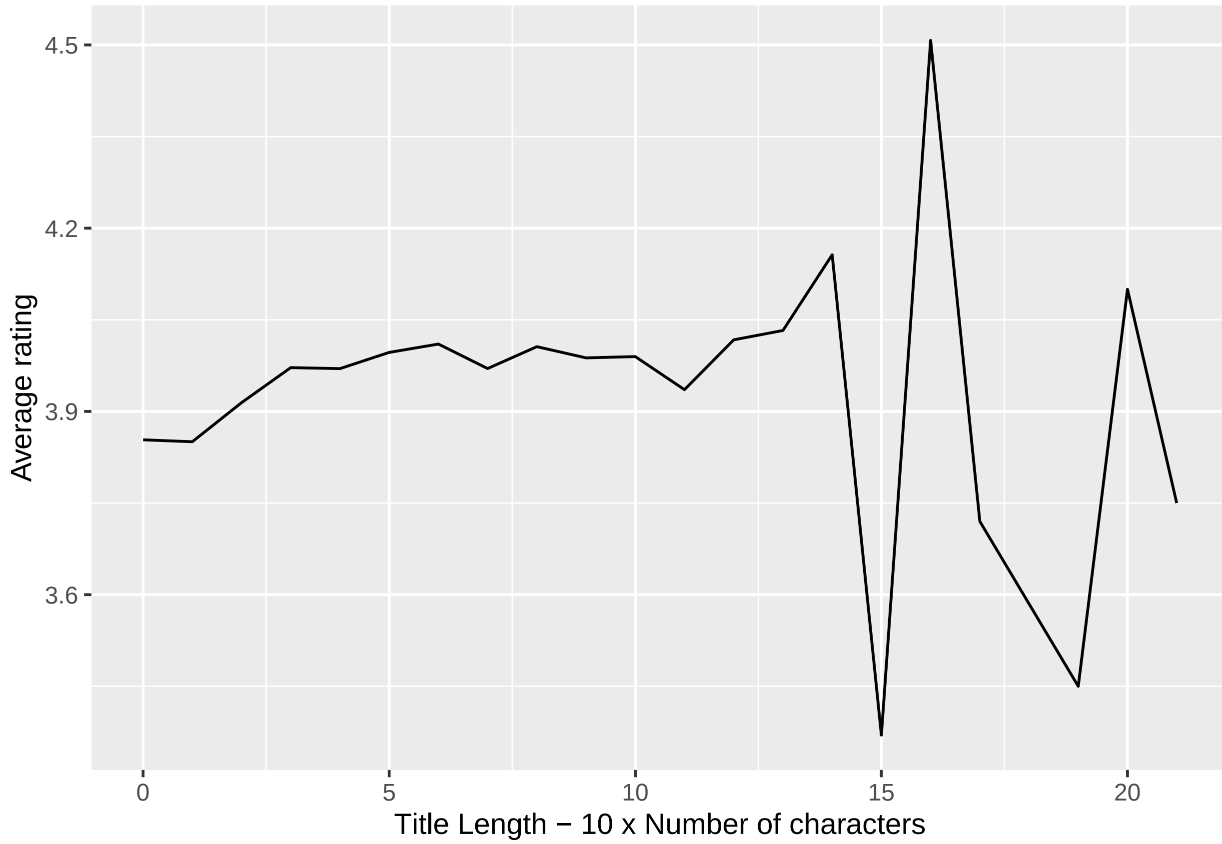
Let's observe a few things. First, let's look at how many books we have, based on the length of the title (number of characters).

```r
# number of books based on title length
train_set %>% group_by(title_length) %>%
  summarize(books = n()) %>%
  ggplot(aes(x = title_length, y = books)) +
  geom_col(fill = "red", alpha=0.5) +
  xlab("Title Length - Number of characters") +
  ylab("Number of books")
```



Since there are too many possible values to calculate stable effects, let's group these into increments of 10 characters.

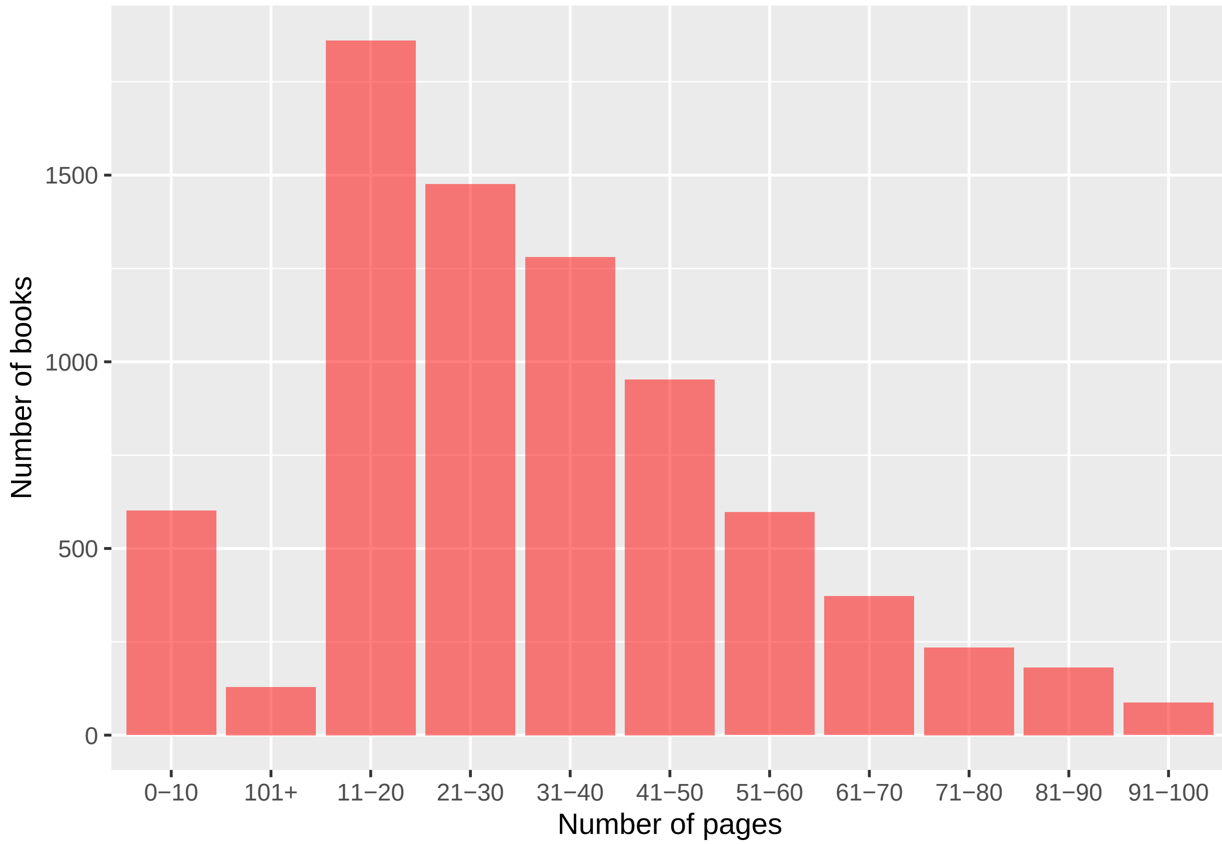
```
# average rating of books based on title length
train_set %>%
  mutate(title_length_range = round(title_length/10)) %>%
  group_by(title_length_range) %>%
  summarize(avg_rating = mean(average_rating)) %>%
  ggplot(aes(x = title_length_range, y = avg_rating)) +
  geom_line() +
  xlab("Title Length - 10 x Number of characters") +
  ylab("Average rating")
```



We can see that a trend exists but, after 100 pages (10 x 10 in the chart), the average rating is no longer stable, as a result of low book count. We will therefore group all the books with a title beyond 100 characters in a single category.

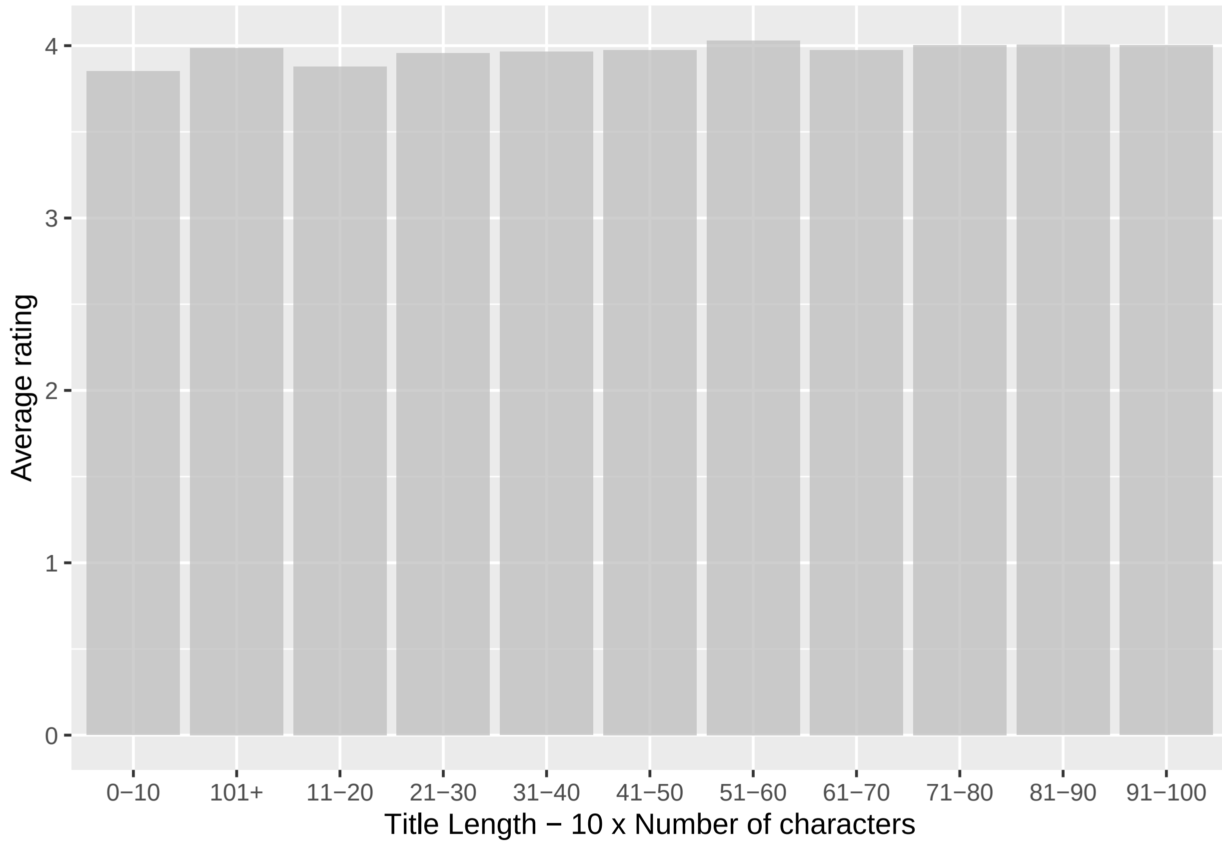
Here is how many books we have now in each group:

```
# number of books based on title length
train_set %>% group_by(title_length_group) %>%
  summarize(books = n()) %>%
  ggplot(aes(x = title_length_group, y = books)) +
  geom_col(fill = "red", alpha=0.5) +
  xlab("Number of pages") +
  ylab("Number of books")
```



and how the average rating looks like:

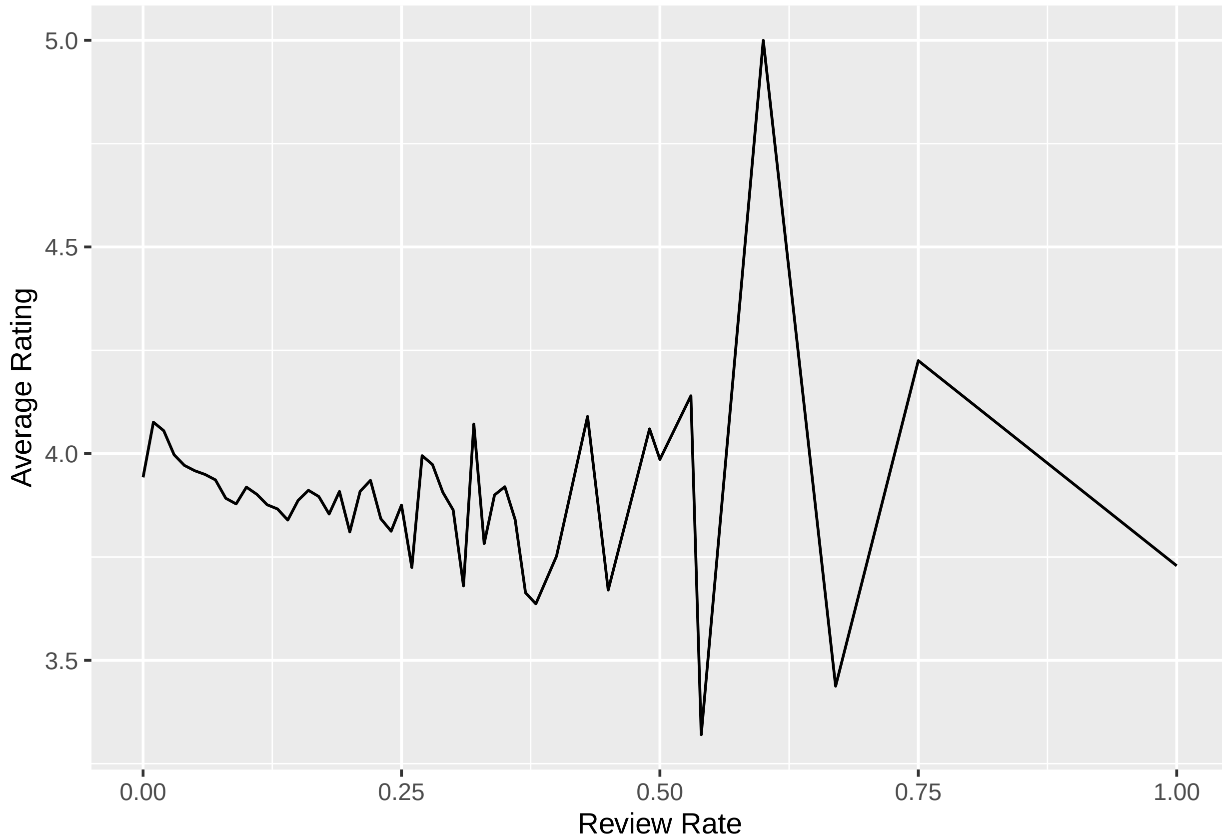
```
# number of books based on title length
train_set %>% group_by(title_length_group) %>%
  summarize(avg_rating = mean(average_rating)) %>%
  ggplot(aes(x = title_length_group, y = avg_rating)) +
  geom_col(fill = "gray", alpha=0.75) +
  xlab("Title Length - 10 x Number of characters") +
  ylab("Average rating")
```



We can see a slight trend, with longer titles getting close to or exceeding an average rating of 4.00.

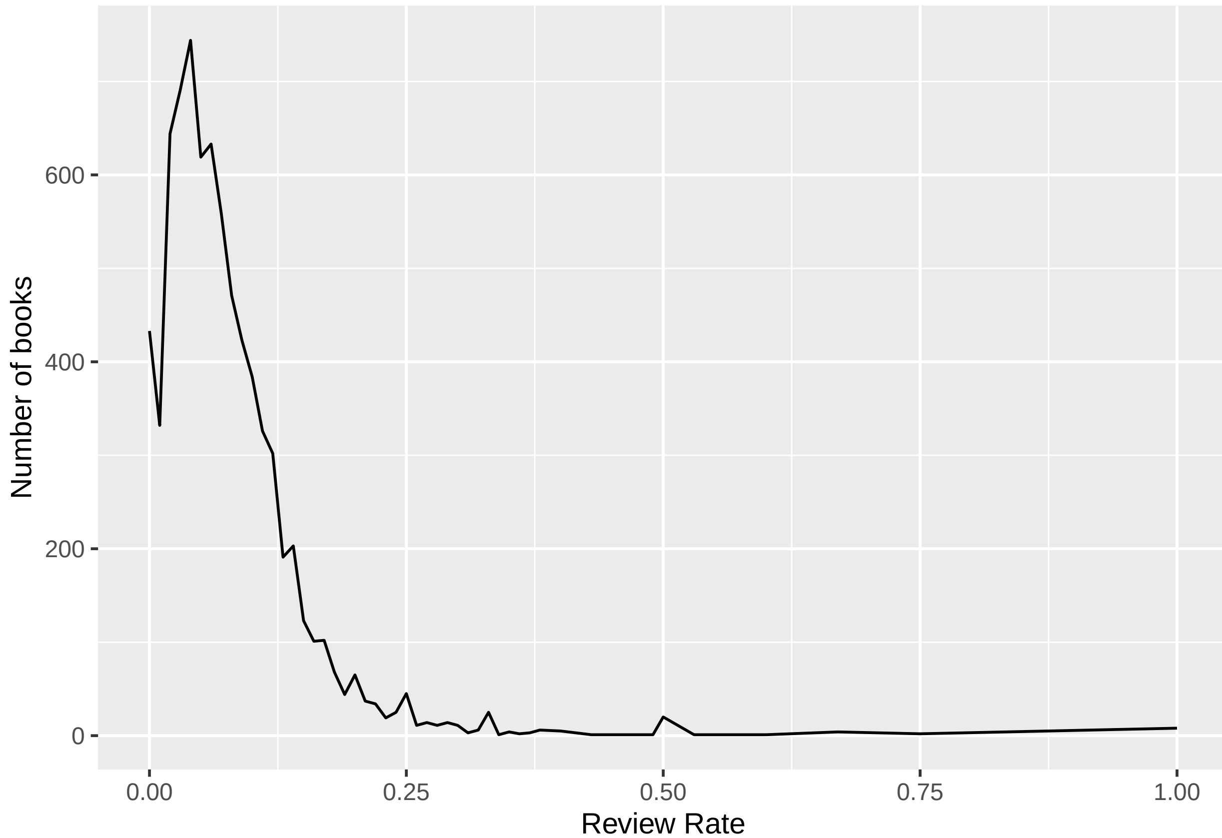
Now, let's do a similar thing for the review rate. Here is the average rating of books based on the review rate (0.25 means 25%).

```
# average rating of books based on review rate
train_set %>%
  mutate(review_rate_rounded = round(review_rate,2)) %>%
  group_by(review_rate_rounded) %>%
  summarize(avg_rating = mean(average_rating)) %>%
  ggplot(aes(x = review_rate_rounded, y = avg_rating)) +
  geom_line() +
  xlab("Review Rate") +
  ylab("Average Rating")
```



and number of books:

```
# number of books based on review rate
train_set %>%
  mutate(review_rate_rounded = round(review_rate,2)) %>%
  group_by(review_rate_rounded) %>%
  summarize(books = n()) %>%
  ggplot(aes(x = review_rate_rounded, y = books)) +
  geom_line() +
  xlab("Review Rate") +
  ylab("Number of books")
```



We can see that number of books decreases below 100 books after a 15% rate and the average rating trend becomes unstable after this rate. We will therefore group all books with a review rate of more than 15% in a single category.

Now, let's calculate effects of language, author, number of pages, title length, and review rate on the average rating.

Below is the code:

```
# store overall average of book ratings in the training set
overall_avg = mean(train_set$average_rating)

# calculate biases within the training set
train_set <- train_set %>%

  # calculate effect of language
  group_by(language_group) %>%
  mutate(b_language = mean(average_rating - overall_avg)) %>%
  ungroup() %>%

  # calculate effect of author, after deducting language
  group_by(authors) %>%
  mutate(b_authors = mean(average_rating - overall_avg - b_language)) %>%
  ungroup %>%

  # calculate effect of number of pages, after deducting language
  # and author effect
```

```
  group_by(num_pages_group) %>%
  mutate(b_num_pages = mean(average_rating - overall_avg - b_language -
                            b_authors)) %>%
  ungroup %>%

  # calculate effect of title length after deducting the other effects
  group_by(title_length_group) %>%
  mutate(b_title_length = mean(average_rating - overall_avg - b_language -
                                 b_num_pages - b_authors)) %>%
  ungroup() %>%

  # calculate effect of review rate, after deducting the other effects
  group_by(review_rate_group) %>%
  mutate(b_review_rate = mean(average_rating - overall_avg - b_language -
                                b_num_pages - b_authors - b_title_length)) %>%
  ungroup()
```

Now, let's extract these effects and join them with the testing set

```
# extracting language effects from the training set
language_averages <- train_set %>%
  group_by(language_group) %>%
  summarize(b_language = mean(b_language))

# extracting author effects from the training set
authors_averages <- train_set %>%
  group_by(authors) %>%
  summarize(b_authors = mean(b_authors))

# extracting title length effects from the training set
title_length_averages <- train_set %>%
  group_by(title_length_group) %>%
  summarize(b_title_length = mean(b_title_length))

# extracting number of pages effect from the training set
num_pages_averages <- train_set %>%
  group_by(num_pages_group) %>%
  summarize(b_num_pages = mean(b_num_pages))

# extracting review rate effects from the training set
review_rate_averages <- train_set %>%
  group_by(review_rate_group) %>%
  summarize(b_review_rate = mean(b_review_rate))

# add data transformations to test set
test_set <- format_df(test_set)

# calculate predicted ratings for the test set
predicted_ratings <- test_set %>%

  # integrate language effect as extracted from training set
  left_join(language_averages, by='language_group') %>%
  # applying a default value for categories not found
```

```r
  mutate(b_language_clean = ifelse(is.na(b_language), 0, b_language)) %>%

  # integrate author effect as extracted from training set
  left_join(authors_averages, by='authors') %>%
  mutate(b_authors_clean = ifelse(is.na(b_authors), 0, b_authors)) %>%

  # integrate number of pages effect as extracted from training set
  left_join(num_pages_averages, by='num_pages_group') %>%
  mutate(b_num_pages_clean = ifelse(is.na(b_num_pages), 0,
                                    b_num_pages)) %>%

  # integrate title length effect as extracted from training set
  left_join(title_length_averages, by='title_length_group') %>%
  mutate(b_title_length_clean = ifelse(is.na(b_title_length), 0,
                                    b_title_length)) %>%

  # integrate review rate effect as extracted from training set
  left_join(review_rate_averages, by='review_rate_group') %>%
  mutate(b_review_rate_clean = ifelse(is.na(b_review_rate), 0,
                                   b_review_rate)) %>%

  # calculate prediction
  mutate(pred = overall_avg +
           b_language_clean +
           b_authors_clean +
           b_num_pages_clean +
           b_title_length_clean +
           b_review_rate_clean) %>%
  # limit prediction to 0.5 to 5.0 range to avoid going outside the range
  mutate(pred_capped = ifelse(pred < 0.5, 0.5, ifelse(pred > 5.0, 5.0, pred)))
```

After we calculate the RMSE, we obtain:

```r
# calculate RMSE for test set
sqrt(mean((predicted_ratings$pred_capped - test_set$average_rating)^2))
```

```
## [1] 0.266859
```

Results

The results we have obtained are:

Linear Regression: 0.2843622

Random Forest: 0.2771318

Recommendation System approach: 0.266859

Applying the last one on the validation set, we get an RMSE of:

```
## [1] 0.2732935
```

Conclusion

We have observed that, for this specific data set, using a recommendation system approach still produces better results than a RandomForest approach (at least with the parameters attempted in this analysis). However, unlike a typical recommendation system data set, such as the one we had from the movie database, we actual don't have individual ratings but only book averages. This reduces the possibility of analysis since average ratings tend to be more stable.

While we have obtained a low RSME, note that it was not a major improvement over linear regression and is actually close to the standard deviation of the training data set (0.2968966) and testing data sets (0.2930259).

Still, this was an interesting exercise to attempt, especially since there weren't many things with strong correlation, which made me think about additional indicators. As an example, it was interesting to see that while the number of reviews and the number of ratings had very low correlation to average rating, when combining them as a rate, we actually found a bit of a correlation.