MovieLens project

Signe Arias

July 2023 – June 2024

Contents

1 Introduction

1.1 Aim of the project

The aim of this project is to develop a movie recommendation system using the MovieLens dataset provided to us. The system is to be created by utilising all the tools covered throughout the courses in the Data Science course. The 10M version of the MovieLens dataset will be used and code to generate the training and validation sets is provided (see below in the next section). The project starts off with a brief quiz in order to gives us a chance to familiarise ourselves with the data provided. Then a machine learning algorithm needs to be developed which can be used to predict movie ratings.

1.2 MovieLens dataset[1]

The MovieLens dataset was developed by the University of Minnesota. 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 randomly and each rated at least 20 movies. Moreover, the only information available on each user is their allocated id.

1.3 Data layout

To get us started, the initial code was provided to us by edx.

```
############################################################################
## Create edx set, validation set
############################################################################

# Note: this process could take a couple of minutes

# Package Installs
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")
```

```
## Warning: package 'caret' was built under R version 4.3.2
```

```
# Libraries
library(tidyverse)
library(caret)
library(hexbin)
```

```
## Warning: package 'hexbin' was built under R version 4.3.3
```

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

# Check for download
datafile <- "MovieLens.RData"
if(!file.exists("MovieLens.RData"))
{
  print("Download")
```

```
  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

  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,]

  # 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)
  edx <- rbind(edx, removed)
  rm(dl, ratings, movies, test_index, temp, movielens, removed)
  save(edx, validation, file = datafile)

} else {
  load(datafile)
}
```

To begin with, I wanted to refamiliarise myself with the data provided. I did so by firstly looking at the headings and then the summary of the data.

```
head(edx)
```

```
##   userId movieId rating timestamp title genres
## 1      1     122      5 838985046  <NA>   <NA>
## 2      1     185      5 838983525  <NA>   <NA>
## 4      1     292      5 838983421  <NA>   <NA>
## 5      1     316      5 838983392  <NA>   <NA>
## 6      1     329      5 838983392  <NA>   <NA>
## 7      1     355      5 838984474  <NA>   <NA>
```

The data seems fairly straightforward and contains just six columns where timestamp is based on seconds since midnight UTC January 1, 1970.

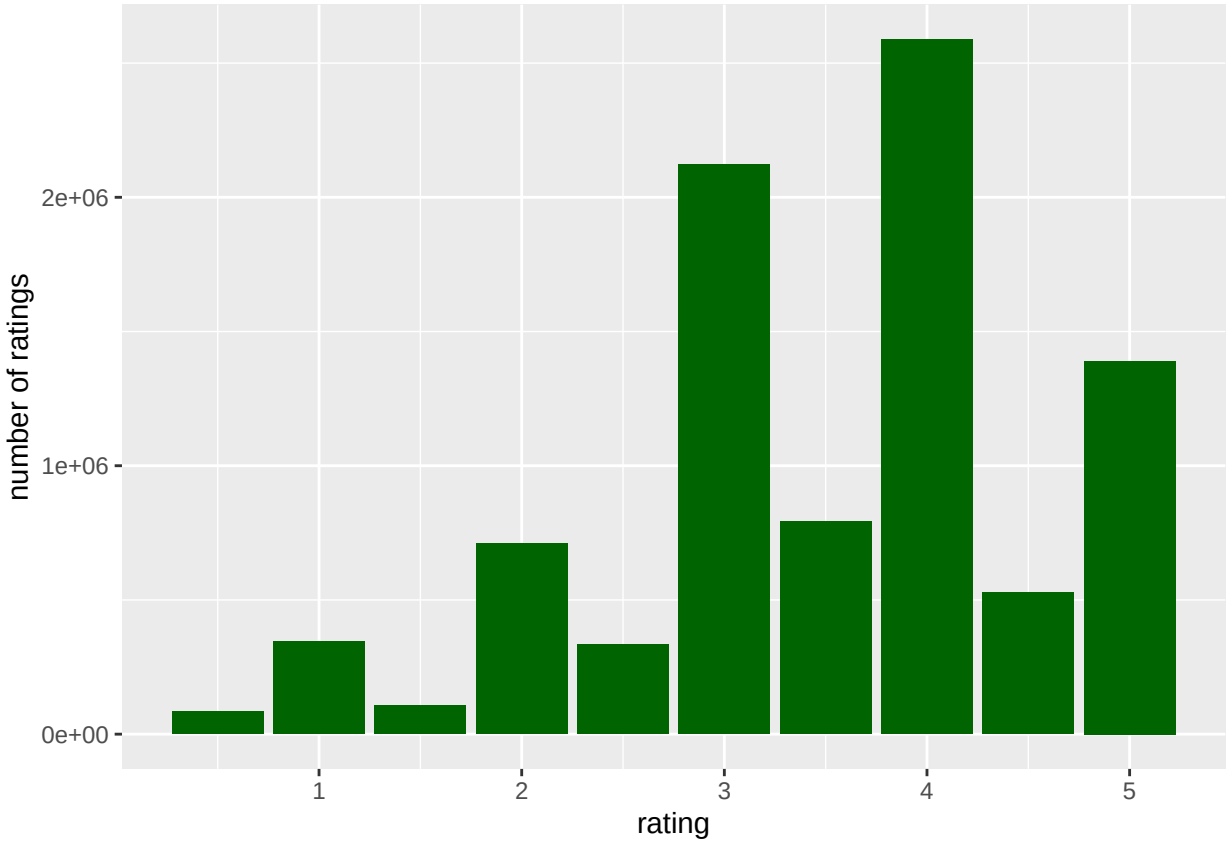
```
summary(edx)
```

```
##      userId         movieId          rating        timestamp
##  Min.   :    1   Min.   :    1   Min.   :0.500   Min.   :7.897e+08
##  1st Qu.:18124   1st Qu.:  648   1st Qu.:3.000   1st Qu.:9.468e+08
##  Median :35738   Median : 1834   Median :4.000   Median :1.035e+09
##  Mean   :35870   Mean   : 4122   Mean   :3.512   Mean   :1.033e+09
##  3rd Qu.:53607   3rd Qu.: 3626   3rd Qu.:4.000   3rd Qu.:1.127e+09
##  Max.   :71567   Max.   :65133   Max.   :5.000   Max.   :1.231e+09
##     title              genres
##  Length:9000055     Length:9000055
##  Class :character   Class :character
##  Mode  :character   Mode  :character
##
##
##
```

Looking at the summary above, we can obtain statistical data e.g. the mean value of ratings as well as the number of userIDs and titles.

I then wanted to see the frequency of the ratings:

```
edx %>%
  ggplot(aes(rating)) +
  geom_bar(fill = "darkgreen") +
  xlab("rating") +
  ylab("number of ratings")
```



The bar plot above tells us than the most used rating is 4 and that half ratings are overall less used than full ones.

2 Method

In this section several methods used to develop the movie recommendation system are discussed. We will start with the simplest one using just the average.

To evaluate and compare our data a loss function called residual mean squared error (RMSE) will be used. It is defined as follows:

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

Whereby N is the number of user – movie combinations, $Y_{u,i}$ is the rating for movie i by user u, and $\hat{y}_{u,i}$ is our prediction. RMSE can be interpreted the same way as a standard deviation and therefore the lower it is the better.

In my analysis, RMSE is computed as follows:

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

2.1 The initial model

The initial model used is the most basic system whereby the model assumes that all movies and users have the same rating, with any differences attributed to random variation. The model is defined as follows:

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

where μ stands for the true rating applicable to all movies and users, and $\epsilon_{u,i}$ denotes independent errors sampled from a distribution centred at zero.

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

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

And therefore, the first RMSE is

```
first_rmse <- RMSE(edx$rating, mu_hat)
first_rmse
```

```
## [1] 1.060331
```

We can now use this as a basis to improve on our model.

2.2 The movie effect

The initial model above can be improved upon by adding in an average movie rating. This needs to be done as we know that some movies can be generally rated highly. So we add a term called b (for bias) and we do so for every movie i.

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

Now we run the improved code and compute the new RMSE.

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

predicted_ratings <- mu_hat + edx %>%
  left_join(movie_eff, by='movieId') %>%
  pull(b_i)

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

```
## [1] 0.9423475
```

To make the comparison with the first model easier, let's put together a table:

```
comp_table <- data_frame(Method = "The initial model", RMSE = first_rmse)
```

```
## Warning: `data_frame()` was deprecated in tibble 1.1.0.
## i Please use `tibble()` instead.
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_lifecycle_warnings()` to see where this warning was
## generated.
```

```
comp_table <- bind_rows(comp_table,
                        data_frame(Method="The movie effect",
                                   RMSE = movie_eff_rmse))
comp_table %>% knitr::kable()
```

Method	RMSE
The initial model	1.0603313
The movie effect	0.9423475

As we can see the value of RMSE has improved significantly.

2.3 The user effect

Users can also affect the ratings greatly as both conscious and unconscious bias come into play, as well as the particular mood of the day. So let's account for that in the next model.

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

```
user_eff <- edx %>%
  left_join(movie_eff, by = "movieId") %>%
  group_by(userId) %>%
  summarise(b_u = mean(rating - mu_hat - b_i))

predicted_ratings <- edx %>%
  left_join(movie_eff, by = "movieId") %>%
  left_join(user_eff, by = "userId") %>%
  mutate(pred = mu_hat + b_i + b_u) %>%
  pull(pred)

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

```
## [1] 0.8567039
```

Comparison with the previous two methods in the table below shows further improvements.

```
comp_table <- bind_rows(comp_table,
                        data_frame(Method="The user effect",
                                   RMSE = user_eff_rmse))
comp_table %>% knitr::kable()
```

Method	RMSE
The initial model	1.0603313
The movie effect	0.9423475
The user effect	0.8567039

2.4 Regularisation

In this project so far, we have accounted for bias by users but we still need to account for the extreme ratings. As we know not all movies are rated by the same number of people which can significantly skew the data. To account for this something called regularisation can be used which takes extremes into account. This is given by the equation where λ is the penalty term used:

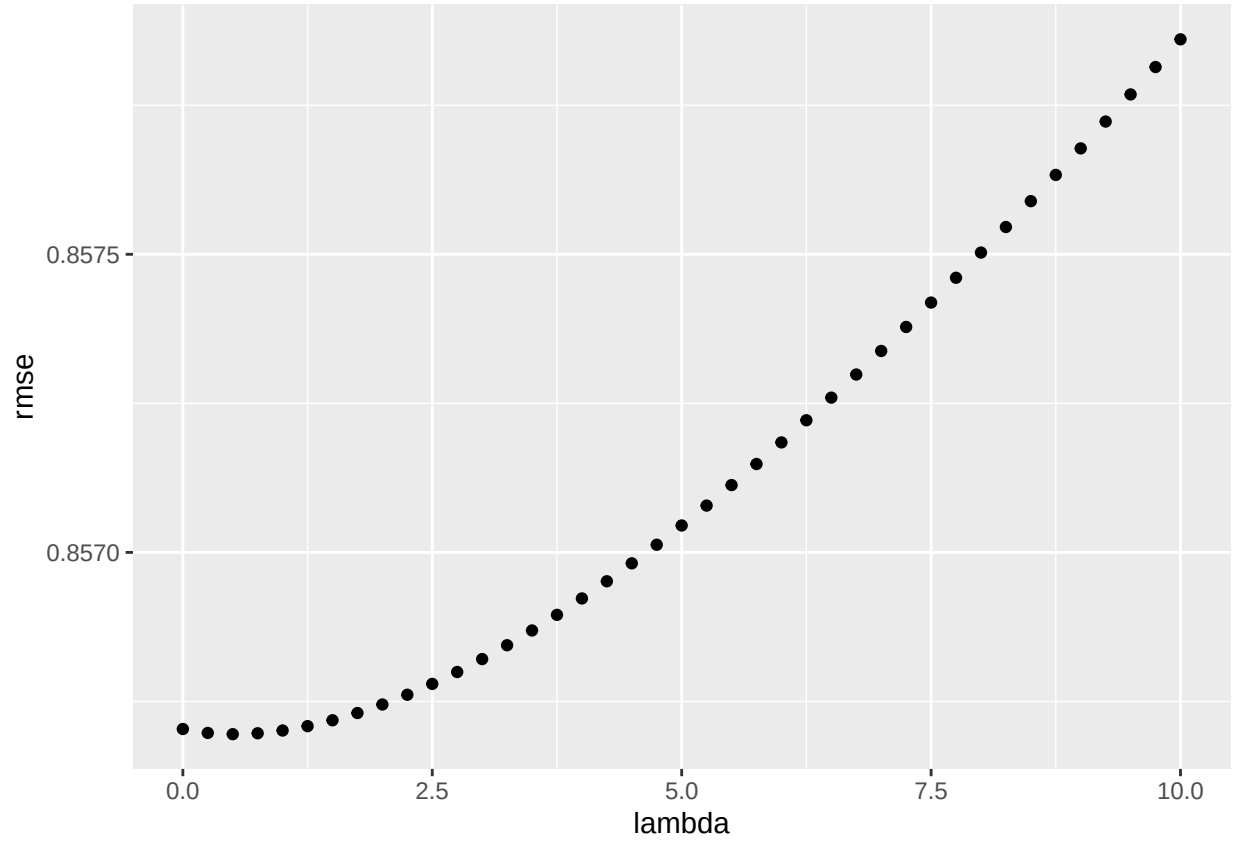
$$Y_{u,i} = \mu + b_{i,n,\lambda} + b_{u,n,\lambda} + \epsilon_{u,i}$$

```
lambda <- seq(0, 10, 0.25)
rmse <- sapply(lambda, function(x){
  b_i <- edx %>%
    group_by(movieId) %>%
    summarize(b_i = sum(rating - mu_hat)/(n()+x))
  b_u <- edx %>%
    left_join(b_i, by = "movieId") %>%
    group_by(userId) %>%
    summarize(b_u = sum(rating - b_i - mu_hat)/(n()+x))
  predicted_ratings <- edx %>%
    left_join(b_i, by = "movieId") %>%
    left_join(b_u, by = "userId") %>%
    mutate(pred = mu_hat + b_i + b_u) %>%
    pull(pred)
  return(RMSE(predicted_ratings, edx$rating))
})
```

Now we plot the values of RMSE versus λ to determine the lowest value for an optimum output.

```
qplot(lambda, rmse)
```

```
## Warning: `qplot()` was deprecated in ggplot2 3.4.0.
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_lifecycle_warnings()` to see where this warning was
## generated.
```



The optimum λ is

```
low_lambda <- lambda[which.min(rmse)]
low_lambda
```

```
## [1] 0.5
```

Therefore, the RMSE with $\lambda = 0.5$ is

```
rmse_regularisation <- min(rmse)
rmse_regularisation
```

```
## [1] 0.8566952
```

3 Results

The summary of the RMSE results are shown in the table below:

```
comp_table <- bind_rows(comp_table,
                        data_frame(Method="Regularisation",
                                   RMSE = rmse_regularisation))
comp_table %>% knitr::kable()
```

Method	RMSE
The initial model	1.0603313
The movie effect	0.9423475
The user effect	0.8567039
Regularisation	0.8566952

From these results we can conclude that the more bias is taken out of the equation, the better the results and that regularisation decreased the RMSE only by a small fraction.

4 Conclusion

To summarise, four different methods were applied to the MovieLens data provided to create a recommendation system. We know that the more bias was taken out of the system the better the results so future work could include more bias taken out. For example, the time of the day, the time of year or user age and so on.

5 References

1. F. Maxwell Harper and Joseph A. Konstan. 2015. The MovieLens Datasets: History and Context. ACM Transactions on Interactive Intelligent Systems (TiiS) 5, 4, Article 19 (December 2015), 19 pages. DOI=http://dx.doi.org/10.1145/2827872