edx_flora_movielens_recommandation_system_project

Flora NIYOKWIZERWA

11/18/2022

Introduction

the movie lens recommendation system project is part of HarvardX: PH125.9x Data Science: Capstone course.With this project I created movie recommendation system using movieLens dataset. I used 10M version of movieLens dataset. I downloaded the dataset using the guidelines given from the course. The goal of the project was to train machine learning algorithm and predict the movie rating in the validation set. the steps followed: we downloaded the 10M movieLens dataset,train algorithm, analyse data, data visualization and used RMSE.

we downloaded dataset from MovieLens as per instruction from the course

the purpose of this part is to understand the use of dataset and help me to work on our project and familiarize myself with libraries to be used

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

```
## Warning: package 'tidyverse' was built under R version 4.1.3
## Warning: package 'ggplot2' was built under R version 4.1.3
## Warning: package 'tibble' was built under R version 4.1.3
## Warning: package 'tidyr' was built under R version 4.1.3
## Warning: package 'readr' was built under R version 4.1.3
## Warning: package 'purrr' was built under R version 4.1.3
## Warning: package 'dplyr' was built under R version 4.1.3
## Warning: package 'stringr' was built under R version 4.1.3
## Warning: package 'forcats' was built under R version 4.1.3
```

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

```
## Warning: package 'caret' was built under R version 4.1.3
```

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

```
## Warning: package 'data.table' was built under R version 4.1.3
```

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

```
## Warning: package 'lubridate' was built under R version 4.1.3
```

```r
library(tidyverse)
library(caret)
library(data.table)
library(lubridate)
```

```r
# MovieLens 10M dataset:
# https://grouplens.org/datasets/movielens/10m/
# http://files.grouplens.org/datasets/movielens/ml-10m.zip
filename <- "comeon.R"

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

  ratings <- fread(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")


  # if using R 4.0 or later:
  movies <- as.data.frame(movies) %>% mutate(movieId = as.numeric(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 3.5 or earlier, use `set.seed(1)`
```

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

```r
  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)

  #glimpse(edx)
  #we want to check the exact year movie was rated
  #so we convert timestamp to year
  edx <- mutate(edx,year_rated= year(as_datetime(timestamp)))
  head(edx)
```

```
##    userId movieId rating timestamp                         title
## 1:      1     122      5 838985046              Boomerang (1992)
## 2:      1     185      5 838983525               Net, The (1995)
## 3:      1     292      5 838983421               Outbreak (1995)
```

```
## 4:      1     316      5 838983392              Stargate (1994)
## 5:      1     329      5 838983392 Star Trek: Generations (1994)
## 6:      1     355      5 838984474       Flintstones, The (1994)
##                           genres year_rated
## 1:                Comedy|Romance       1996
## 2:         Action|Crime|Thriller       1996
## 3:  Action|Drama|Sci-Fi|Thriller       1996
## 4:       Action|Adventure|Sci-Fi       1996
## 5: Action|Adventure|Drama|Sci-Fi       1996
## 6:       Children|Comedy|Fantasy       1996
```

```r
#data cleaning
 #we want to see the number of unique movies we have without repetition
 n_distinct(edx$movieId)
```

```
## [1] 10677
```

```r
 #we also need to see number of unique users using userId who rated our movies
 n_distinct(edx$userId)
```

```
## [1] 69878
```

```r
 #we start exploring our data set
 #firstly let's take a look on our data structure using str function
 str(edx)
```

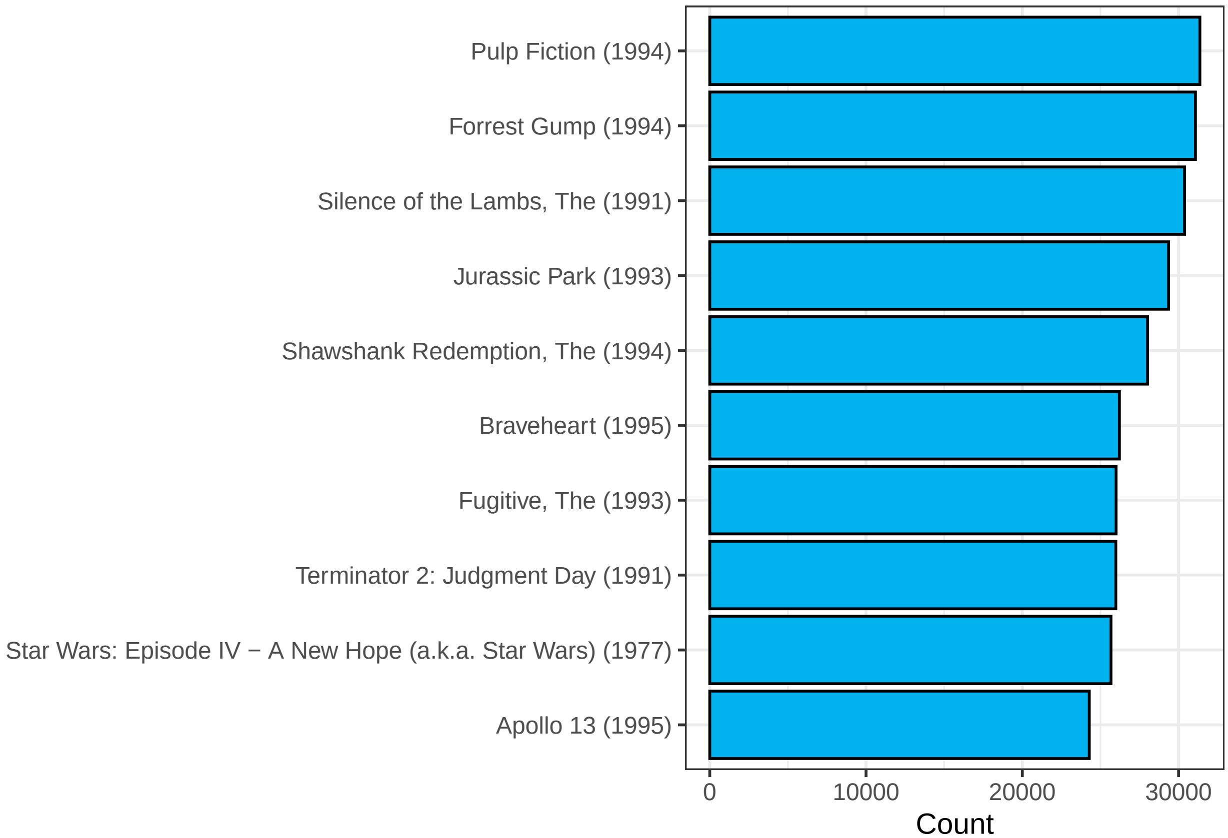
```
## Classes 'data.table' and 'data.frame':   9000055 obs. of  7 variables:
##  $ userId    : int  1 1 1 1 1 1 1 1 1 1 ...
##  $ movieId   : num  122 185 292 316 329 355 356 362 364 370 ...
##  $ rating    : num  5 5 5 5 5 5 5 5 5 5 ...
##  $ timestamp : int  838985046 838983525 838983421 838983392 838983392 838984474 838983653 838984885 8
##  $ title     : chr  "Boomerang (1992)" "Net, The (1995)" "Outbreak (1995)" "Stargate (1994)" ...
##  $ genres    : chr  "Comedy|Romance" "Action|Crime|Thriller" "Action|Drama|Sci-Fi|Thriller" "Action|A
##  $ year_rated: num  1996 1996 1996 1996 1996 ...
##  - attr(*, ".internal.selfref")=<externalptr>
```

```r
 #now let's see our movies summary which presents statistical summary of our data set
 summary(edx)
```

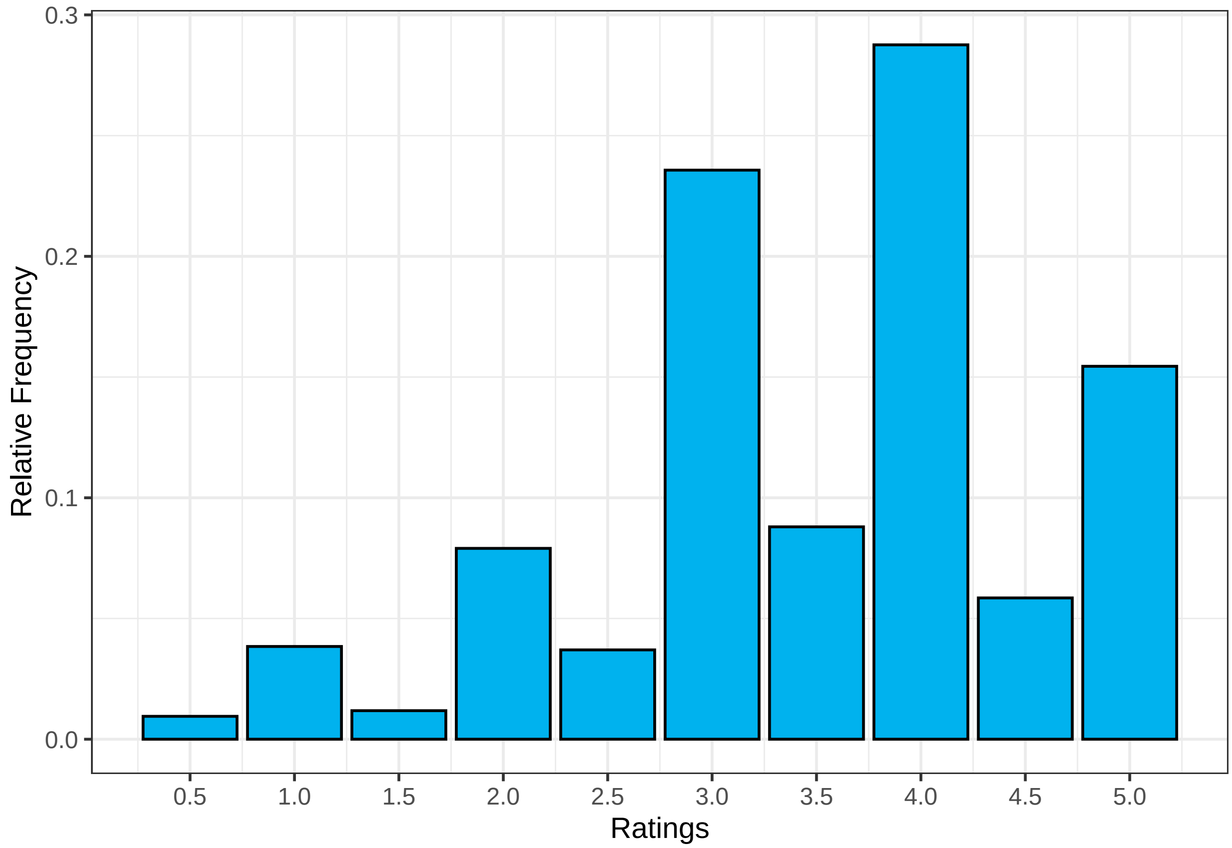
```
##      userId         movieId          rating        timestamp
##  Min.   :    1   Min.   :    1   Min.   :0.500   Min.   :7.897e+08
##  1st Qu.:18124   1st Qu.:  648   1st Qu.:3.000   1st Qu.:9.468e+08
##  Median :35738   Median : 1834   Median :4.000   Median :1.035e+09
##  Mean   :35870   Mean   : 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            year_rated
##  Length:9000055     Length:9000055     Min.   :1995
##  Class :character   Class :character   1st Qu.:2000
##  Mode  :character   Mode  :character   Median :2002
##                                        Mean   :2002
##                                        3rd Qu.:2005
##                                        Max.   :2009
```

```r
 #we need to see the most watched movies from our data set and plot the result on bar chart
 edx %>%
   group_by(title) %>%
   summarize(count = n()) %>%
```

```
  arrange(-count) %>%
  top_n(10, count) %>%
  ggplot(aes(count, reorder(title, count))) +
  geom_bar(color = "black", fill = "deepskyblue2", stat = "identity") +
  xlab("Count") +
  ylab(NULL) +
  theme_bw()
```




```
#the bar graph to show the highest rate given to any movie by any user
edx %>%
  ggplot(aes(rating, y = ..prop..)) +
  geom_bar(color = "black", fill = "deepskyblue2") +
  labs(x = "Ratings", y = "Relative Frequency") +
  scale_x_continuous(breaks = c(0.5, 1, 1.5, 2, 2.5, 3, 3.5, 4, 4.5, 5)) +
  theme_bw()
```



##Methods/analysis section Within this section,I'm going to discuss different models used to analyse my dataset.I'm going to use different model and the residual mean square error(RMSE) to measure the accuracy and quantify typical error I may make when predicting the movie rating. ## the first model used is Simple mean Rating. this method uses the mean of whole dataset to predict movie rating by assuming that any difference is due to random error.

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

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

```
simplermse<-RMSE(validation$rating,aver)
simplermse
```

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

```
## save the result from rmse using simple mean into data frame
simple_rmse = data_frame(method= "Simple Mean model",
                         RMSE=simplermse)
```

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

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

method	RMSE
Simple Mean model	1.061202

the above model produced RMSE=1.061202 which is above 0.90000 and this typical error is larger than accepted error in our project

the second model used is Movie Bias Effect model

after analysing with the simple mean rating model I found that error is large than accepted one, so I conducted further analysis to come up with movie rating prediction with small error The movie bias effect model it uses the mean from the dataset and taking consideration of movie bias b_i based on difference between movie mean rating and overall rating

```
#calculate overall dataset mean rating and store it in variable aver
  aver <- mean(edx$rating)
  aver
```

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

```
  #calculate movie rating mean
  movies_aver <- edx %>% group_by(movieId) %>% summarise(b_i= mean(rating-aver))
  p_rating <- aver + validation %>%
    left_join(movies_aver, by="movieId") %>%
    pull(b_i)
  rmse_modelmoviebias <- RMSE(p_rating, validation$rating)
  rmse_modelmoviebias
```

```
## [1] 0.9439087
```

```
  simple_rmse <- bind_rows(simple_rmse, tibble(Method = "movie bias model",
                                               RMSE=rmse_modelmoviebias))
  simple_rmse %>% knitr::kable()
```

method	RMSE	Method
Simple Mean model	1.0612018	NA
NA	0.9439087	movie bias model

with the above approach, it gave me lower error compared to simple mean rating model, the rmse got: 0.94390. But still the error is elevated so I had to continue my analysis with other approach.

the third model used was movies and users effect model

with this model, I assumed that user feelings may affect movie rating outcome, that's why i used this model taking into consideration user mean rating b_u

```
user_aver <- edx %>%
    left_join(movies_aver, by="movieId") %>%
    group_by(userId)%>%
    summarise(b_u = mean(rating - aver - b_i))
p_rating <- validation %>%
  left_join(movies_aver, by="movieId")%>%
  left_join(user_aver, by="userId") %>%
  mutate(pred = aver + b_i + b_u)%>%
  pull(pred)
```

```r
rmse_user_effect <- RMSE(p_rating,validation$rating)
rmse_user_effect
```

```
## [1] 0.8653488
```

```r
simple_rmse <- bind_rows(simple_rmse,
                         data_frame(method="Movie-User effects",
                                    RMSE=rmse_user_effect))
simple_rmse %>% knitr::kable()
```

method	RMSE	Method
Simple Mean model	1.0612018	NA
NA	0.9439087	movie bias model
Movie-User effects	0.8653488	NA

the above method produced low error of 0.8653488 compared to 0.9439087 produced by movie bias effect mode, I continued my analysis to the next model

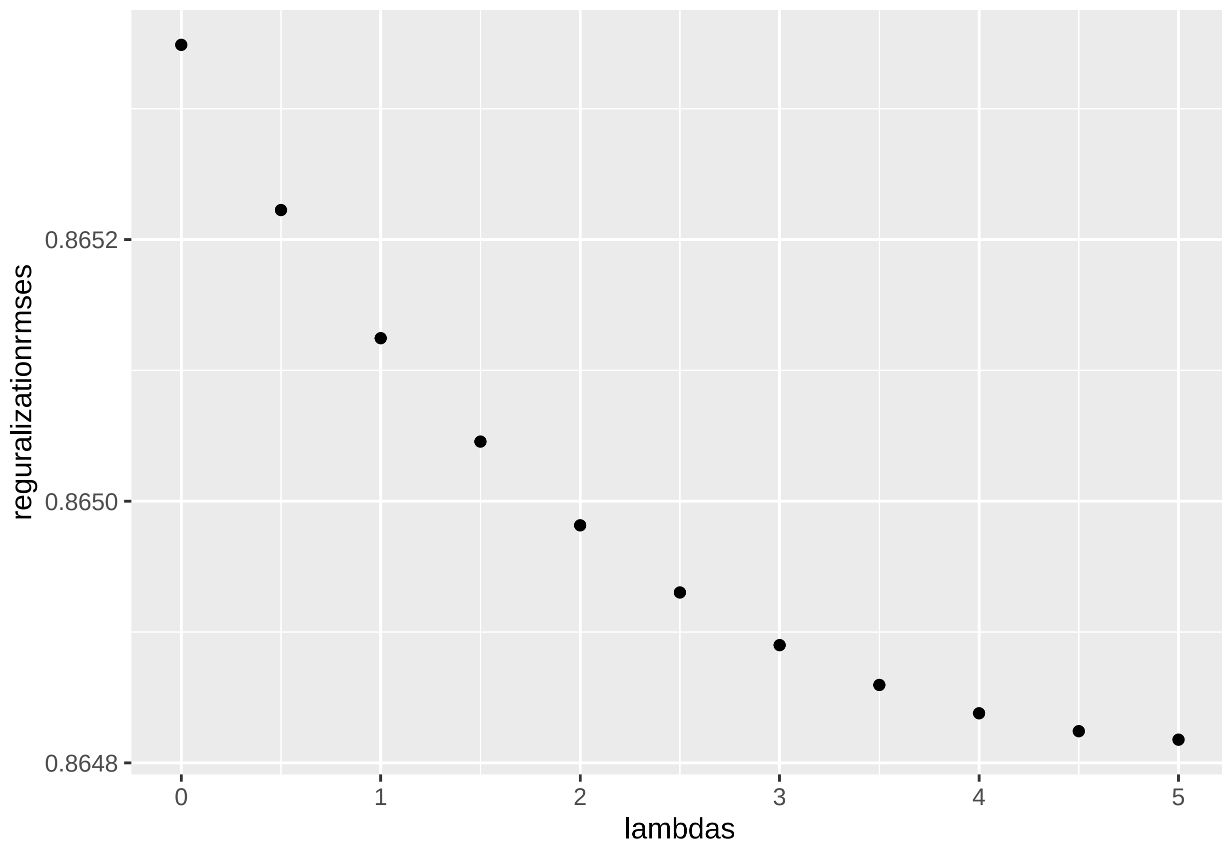
##Regularization model

Regularisation allows for reduced errors caused by movies with few ratings which can influence the prediction and skew the error metric. The method uses a tuning parmeter, λ, to minimise the RMSE. Modifying b_i and b_u for movies with limited ratings.

```r
lambdas <- seq(0,5,0.5)
reguralizationrmses <- sapply(lambdas, function(x){
  aver <- mean(edx$rating)

  b_i <- edx %>%
    group_by(movieId) %>%
    summarise(b_i = sum(rating - aver)/(n()+x))
  b_u <- edx %>%
    left_join(b_i, by="movieId")%>%
    group_by(userId)%>%
    summarise(b_u = sum(rating - b_i - aver)/(n()+x))
  p_rating <- validation %>%
    left_join(b_i, by ="movieId")%>%
    left_join(b_u, by ="userId") %>%
    mutate(pred = aver + b_i + b_u)%>%
    pull(pred)

  return(RMSE(p_rating, validation$rating))
})
rmes_regul <- min(reguralizationrmses)
rmes_regul
```

```
## [1] 0.8648177
```

```r
#we plot RMSE against lambdas

qplot(lambdas,reguralizationrmses)
```




```
lambda<- lambdas[which.min(reguralizationrmses)]
lambda
```

```
## [1] 5
```

```
simple_rmse<-bind_rows(simple_rmse,
                       data_frame(method="Regularization model",
                                  RMSE= rmes_regul))
simple_rmse %>% knitr::kable()
```

method	RMSE	Method
Simple Mean model	1.0612018	NA
NA	0.9439087	movie bias model
Movie-User effects	0.8653488	NA
Regularization model	0.8648177	NA

with this method I got rmse: 0.86418177 which is somehow the nearest accepted typical error.

##Results

the results of the models above can be found in our data frame table below the lowest rmse found is 0.8648177

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

method	RMSE	Method
Simple Mean model	1.0612018	NA

method	RMSE	Method
NA	0.9439087	movie bias model
Movie-User effects	0.8653488	NA
Regularization model	0.8648177	NA

##Conclusion

The project objective was to create movie recommendation system using movieLens data set version 10M to predict the movie ratings. I used different models to construct this project and got the lowest Residual mean square Error (RMSE). The RMSE got was 0.8648177.