AlexBoffa-MovieLens-Project-Harvard.R

YogaI5

2020-03-18

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

# Note: this process took my computer about 10 min create edx and validation sets.

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  2.1.3     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
```

```
## Warning: package 'ggplot2' was built under R version 3.6.3
```

```
## Warning: package 'dplyr' was built under R version 3.6.3
```

```
## Warning: package 'forcats' was built under R version 3.6.3
```

```
## -- 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
```

```
## Warning: package 'caret' was built under R version 3.6.3
```

```
## 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
```

```
## Warning: package 'data.table' was built under R version 3.6.3
```

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

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

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

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

dl <- tempfile()
download.file("http://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")
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")
```

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

```
# 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,]

# 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)

#############################################################################
#####   Starting MovieLens Recomendation System Project
```

```
##########################################################################

# Install all needed libraries if they are not present

if(!require(tidyverse)) install.packages("tidyverse")
if(!require(tidyr)) install.packages("tidyr")
if(!require(stringr)) install.packages("stringr")
if(!require(forcats)) install.packages("forcats")
if(!require(ggplot2)) install.packages("ggplot2")
if(!require(kableExtra)) install.packages("kableExtra")
```

```
## Loading required package: kableExtra
```

```
## Warning: package 'kableExtra' was built under R version 3.6.3
```

```
##
## Attaching package: 'kableExtra'
```

```
## The following object is masked from 'package:dplyr':
##
##     group_rows
```

```
# Loading all needed libraries

library(dslabs)
library(caret)
library(dplyr)
library(tidyverse)
library(kableExtra)
library(tidyr)
library(stringr)
library(forcats)
library(ggplot2)


##############################################################################
# This is the RMSE function that will give us the scores found with our models.
# Our goal is obtain a RMSE < 0.86490

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


#####################################
# We start computing our models here #
#####################################

############################
### Starting Naive model ###

# Calculating "just the average" of all movies

mu <- mean(edx$rating)

# Calculating the RMSE on the validation set
```

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

# Creating a results dataframe that will contains all RMSE results.
# Here we insert our first RMSE.

rmse_results <- data.frame(method = "Just the average", RMSE = naive_rmse)

###################################
### Starting Movie Effect Model ###

# Calculating the average by movie

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

# Computing the predicted ratings on validation dataset

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

# Computing Movie effect model

model_1_rmse <- RMSE(predicted_ratings, validation$rating)

# Adding the results to the rmse_results table

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

```
## Warning in bind_rows_(x, .id): Unequal factor levels: coercing to character
```

```
## Warning in bind_rows_(x, .id): binding character and factor vector, coercing
## into character vector

## Warning in bind_rows_(x, .id): binding character and factor vector, coercing
## into character vector
```

```
#########################################
### Starting Movie + User Effect Model ###

# Calculating the average by user

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

# Computing the predicted ratings on validation dataset

predicted_ratings <- validation %>%
   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, validation$rating)

# Adding the results to the results dataset

rmse_results <- bind_rows(rmse_results,
                          data.frame(method = "Movie + User Effects Model",
                                     RMSE = model_2_rmse ))
```

```
## Warning in bind_rows_(x, .id): binding character and factor vector, coercing
## into character vector
```

```
##############################################
# Starting Regularization of our models.
##############################################

########################################
# Regularizing Movie + User Effect Model
# Computing the predicted ratings on validation dataset using different values of lambda
# b_i is the Movie effect and b_u is User effect.
# Lambda is a tuning parameter.
# We are using cross-validation to choose the best lambda that minimize our RMSE.

lambdas <- seq(0, 10, 0.25)

# function rmses calculate predictions with several lambdas

rmses <- sapply(lambdas, function(l) {

   # Calculating the average by movie

   b_i <- edx %>%
      group_by(movieId) %>%
      summarize(b_i = sum(rating - mu) / (n() + l))

   # Calculating the average by user

   b_u <- edx %>%
      left_join(b_i, by="movieId") %>%
      group_by(userId) %>%
      summarize(b_u = sum(rating - b_i - mu) / (n() + l))

   # Computing the predicted ratings on validation dataset

   predicted_ratings <- validation %>%
      left_join(b_i, by = 'movieId') %>%
      left_join(b_u, by = "userId") %>%
      mutate(pred = mu + b_i + b_u) %>%
      .$pred

   # Predicting the RMSE on the validation set
```

```
   return(RMSE(predicted_ratings, validation$rating))
})

# Getting the best lambda value that minimize the RMSE on reg movie + user effects model

lambda <- lambdas[which.min(rmses)]

# We know that our best RMSE is given by: RMSE = min(rmses),
# but as a purpose of clarification,
# we compute again our estimate with best lambda found:

# Compute regularized estimates of b_i using best lambda

movie_avgs_reg <- edx %>%
   group_by(movieId) %>%
   summarize(b_i = sum(rating - mu) / (n() + lambda), n_i = n())

# Compute regularized estimates of b_u using best lambda

user_avgs_reg <- edx %>%
   left_join(movie_avgs_reg, by ='movieId') %>%
   group_by(userId) %>%
   summarize(b_u = sum(rating - mu - b_i) / (n() + lambda), n_u = n())

# Predict ratings

predicted_ratings <- validation %>%
   left_join(movie_avgs_reg, by = 'movieId') %>%
   left_join(user_avgs_reg, by = 'userId') %>%
   mutate(pred = mu + b_i + b_u) %>%
   .$pred

# Predicting the RMSE on the validation set

model_3_rmse <- RMSE(predicted_ratings, validation$rating)

# Adding the results to the rmse_results dataset

rmse_results <- bind_rows(rmse_results,
                          data.frame(method = "Regularized Movie + User Effects Model",
                                     RMSE = model_3_rmse ))
```

```
## Warning in bind_rows_(x, .id): binding character and factor vector, coercing
## into character vector
```

```
rmse_results
```

```
##                                     method      RMSE
## 1                          Just the average 1.0612018
## 2                        Movie Effect Model 0.9439087
## 3              Movie + User Effects Model 0.8653488
## 4 Regularized Movie + User Effects Model 0.8648170
```

```
#########################################################
#########################################################
# Remmember, our goal is obtain a RMSE < 0.86490
# And voila, running this project in R on my computer,
# and just as info, I have obtained these rmse_results,
# where the "Regularized Movie + User Effects Model"
# exceeded the goal:
#
#                                 method          RMSE
#
#1                       Just the average     1.0612018
#2                     Movie Effect Model     0.9439087
#3             Movie + User Effects Model     0.8653488
#4 Regularized Movie + User Effects Model     0.8648170
#########################################################
#########################################################

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