Capstone Project - MovieLens

Shweta Kothadiya

2/20/2021

Let's install some packages.

```
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 3.3.2     v purrr   0.3.4
## v tibble  3.0.4     v dplyr   1.0.2
## v tidyr   1.1.2     v stringr 1.4.0
## v readr   1.4.0     v forcats 0.5.0
```

```
## -- 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
```

```
## 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
```

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

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

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

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

Now download the data

```
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"))
```

Now let's build the data set

```
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(movieId),
                                           title = as.character(title),
                                           genres = as.character(genres))

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

Setting the seed

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

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

Now creating the partition. Validation set will be 10% of MovieLens data.

```
test_index <- createDataPartition(y = movielens$rating, times = 1, p = 0.1,
list = FALSE)
```

Saving training set as edx and test set as temp

```
edx <- movielens[-test_index,]
temp <- movielens[test_index,]
```

Now, we have to make sure the movieId and userId 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)
```

Creating RMSE function

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

Creating partition od edx set into train and test set

```
partition_edx <- createDataPartition(y = edx$rating, times = 1, p = 0.1, list
= FALSE)
edx_train_set <- edx[-partition_edx,]
edx_test_set <- edx[partition_edx,]
```

Now we have to make sure movieId & userId from edx_test_set is also in edx_train_set.

```
edx_test_set_1 <- edx_test_set %>%
     semi_join(edx_train_set, by = "movieId") %>%
     semi_join(edx_train_set, by = "userId")
```

Add rows removed from edx_test_set_1 back into edx_train_set

```
removed_1 <- anti_join(edx_test_set, edx_test_set_1)

## Joining, by = c("userId", "movieId", "rating", "timestamp", "title",
"genres")

edx_train_set <- rbind(edx_train_set, removed_1)
```

Take mean of ratings from edx_train_set and save it as mu_hat. Show output of mu_hat.

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

## [1] 3.512509
```

Calculate Naive RMSE on edx_test_set_1 and save it as naive_rmse. Show output of naive_rmse.

```
naive_rmse <- RMSE(edx_test_set_1$rating, mu_hat)
naive_rmse

## [1] 1.061135
```

We are getting naive_rmse as “1.061135”. Let’s save the naive_rmse in tibble format. Mention the method of calculation and save the results as rmse_results.

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

## # A tibble: 1 x 2
##   method            RMSE
##   <chr>            <dbl>
## 1 Just the average  1.06
```

Take the mean of rating under edx_train_set and save it as mu.

```
mu <- mean(edx_train_set$rating)
```

Calculate average of rating based on Movie effect only.

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

## `summarise()` ungrouping output (override with `.groups` argument)
```

Predict ratings on edx_test_set_1 based on movie_avgs model.

```
predicted_ratings <- mu + edx_test_set_1 %>% left_join(movie_avgs,
by='movieId') %>%  .$b_i
```

Calculate RMSE using with the predicted ratings vs the true ratings under edx_test_set_1.

```
model_1_rmse <- RMSE(predicted_ratings, edx_test_set_1$rating)
```

Let's save the results of rmse based on Movie effect model into the rmse_results tibble format.

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

## # A tibble: 2 x 2
##   method              RMSE
##   <chr>              <dbl>
## 1 Just the average   1.06
## 2 Movie Effect Model 0.944
```

The naive_rmse was 1.0611350 but movie effect model is giving us better rmse which is 0.9441568. Now, let's build a movie and user effect model to see if we can get better RMSE results.

```
user_avgs <- edx_test_set_1 %>% left_join(movie_avgs, by='movieId') %>%
group_by(userId) %>% summarize(b_u = mean(rating - mu - b_i))

## `summarise()` ungrouping output (override with `.groups` argument)
```

Predict ratings based on Movie and User effect model.

```
predicted_ratings <- edx_test_set_1 %>% left_join(movie_avgs, by='movieId')
%>% left_join(user_avgs, by='userId') %>% mutate(pred = mu + b_i + b_u) %>%
.$pred
```

Calculate RMSE using the predicted ratings from Movie plus user effect model vs. true ratings under edx_test_set_1. Save the results as model_2_rmse.

```
model_2_rmse <- RMSE(predicted_ratings, edx_test_set_1$rating)
```

Save the RMSE results from this Movie and User effect model into our rmse_results. See the rmse_results output.

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

```
## # A tibble: 3 x 2
##   method                      RMSE
##   <chr>                      <dbl>
## 1 Just the average           1.06
## 2 Movie Effect Model         0.944
## 3 Movie + User Effects Model 0.826
```

Movie and User effect combined model is giving better RMSE which is “0.8262583” than using only the Movie model. Now, let’s build Movie plus Genre effect model and we will see if that make’s any difference.

```
genre_avgs <- edx_test_set_1 %>% left_join(movie_avgs, by='movieId') %>%
group_by(genres) %>% summarize(b_g = mean(rating - mu - b_i))
```

```
## `summarise()` ungrouping output (override with `.groups` argument)
```

Predict ratings using this Movie plus Genre effect model.

```
predicted_ratings <- edx_test_set_1 %>% left_join(movie_avgs, by='movieId')
%>% left_join(genre_avgs, by='genres') %>% mutate(pred = mu + b_i + b_g) %>%
.$pred
```

Calculate RMSE using the predicted ratings from Movie plus Genre effect model vs. true ratings of edx_test_set_1. Save the results as model_3_rmse.

```
model_3_rmse <- RMSE(predicted_ratings, edx_test_set_1$rating)
```

Save the RMSE results based on Movie plus Genre model which is under model_3_rmse into rmse_results. Show the rmse_results output.

```
rmse_results <- bind_rows(rmse_results, tibble(method="Movie + Genre Effects
Model", RMSE = model_3_rmse ))
rmse_results
```

```
## # A tibble: 4 x 2
##   method                      RMSE
##   <chr>                      <dbl>
```

```
## 1 Just the average            1.06
## 2 Movie Effect Model          0.944
## 3 Movie + User Effects Model  0.826
## 4 Movie + Genre Effects Model 0.944
```

"Movie plus Genre Effects Model" is giving RMSE as "0.9436908" which is better than just the naive_rmse and "Movie Effect Model" but not better than the "Movie plus User Effects Model". Let's calculate Genre plus User model.

```
genre_avgs_with_user <- edx_test_set_1 %>% left_join(user_avgs, by='userId')
%>%group_by(genres)%>%summarize(b_g_u = mean(rating-mu-b_u))

## `summarise()` ungrouping output (override with `.groups` argument)
```

Predict ratings using this Genre plus User effect model.

```
predicted_ratings <- edx_test_set_1 %>% left_join(user_avgs, by='userId') %>%
left_join(genre_avgs_with_user, by='genres') %>% mutate(pred = mu + b_u +
b_g_u) %>% .$pred
```

Calculate RMSE using the predicted ratings from Genre plus User effect model vs. true ratings of edx_test_set_1. Save the results as model_4_rmse.

```
model_4_rmse <- RMSE(predicted_ratings, edx_test_set_1$rating)
```

Save the RMSE results based on Genre plus User effect model which is under model_4_rmse into rmse_results. Show the rmse_results output.

```
rmse_results <- bind_rows(rmse_results, tibble(method="User + Genre Effects
Model", RMSE = model_4_rmse))
rmse_results

## # A tibble: 5 x 2
##   method                       RMSE
##   <chr>                       <dbl>
## 1 Just the average            1.06
## 2 Movie Effect Model          0.944
## 3 Movie + User Effects Model  0.826
## 4 Movie + Genre Effects Model 0.944
## 5 User + Genre Effects Model  0.911
```

We can see that "User plus Genre Effects Model" is giving RMSE as "0.9112458", which is better than any RMSE that has been calculated EXCEPT "Movie plus User Effects Model" which has the lowest RMSE that is "0.8262583". Now, let's calculate another model which is "Movie plus User plus Genre Model".

```
genre_avgs_with_Movie_User <- edx_test_set_1 %>% left_join(movie_avgs,
by='movieId') %>% left_join(user_avgs, by='userId') %>% group_by(genres) %>%
summarize(b_g_m_u = mean(rating-mu-b_i-b_u))

## `summarise()` ungrouping output (override with `.groups` argument)
```

Predict ratings using this Movie plus User plus Genre Effects Model.

```
predicted_ratings <- edx_test_set_1 %>% left_join(movie_avgs, by='movieId')
%>% left_join(user_avgs, by='userId') %>%
left_join(genre_avgs_with_Movie_User, by='genres') %>% mutate(pred = mu + b_i
+ b_u + b_g_m_u) %>% .$pred
```

Calculate RMSE using the predicted ratings from Movie plus User plus Genre Effect Model vs. true ratings of edx_test_set_1. Save the results as model_5_rmse.

```
model_5_rmse <- RMSE(predicted_ratings, edx_test_set_1$rating)
```

Save the RMSE results based on Movie plus User plus Genre Effects Model which is under model_5_rmse into rmse_results. Show the rmse_results output.

```
rmse_results <- bind_rows(rmse_results, tibble(method="Movie + User + Genre
Effects Model", RMSE = model_5_rmse))
rmse_results
```

```
## # A tibble: 6 x 2
##   method                              RMSE
##   <chr>                              <dbl>
## 1 Just the average                   1.06
## 2 Movie Effect Model                 0.944
## 3 Movie + User Effects Model         0.826
## 4 Movie + Genre Effects Model        0.944
## 5 User + Genre Effects Model         0.911
## 6 Movie + User + Genre Effects Model 0.826
```

We can see that combined model which is “Movie plus User plus Genre Effects Model” is giving the RMSE as “0.8255252” which is lowest of all the models that we have calculated so far; however if we look at the Only 3 digits decimal after “0.”, then this “Movie plus User plus Genre Effects Model” would have RMSE as 0.826 which is the same RMSE that we have calculated for “Movie plus User Effects Model” which is 0.826. Based on the above calculations, I think we can use “Movie plus User Effects Model” on the Validation set now.

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

## `summarise()` ungrouping output (override with `.groups` argument)
```

Let’s predict the ratings on Validation set.

```
predicted_ratings_validation <- validation %>% left_join(movie_avgs,
by='movieId') %>% left_join(user_avgs_validation, by='userId') %>%
mutate(pred = mu + b_i + b_u) %>% .$pred
```

Let’s calculate RMSE using “predicted_ratings_validation” vs. true ratings under validation set.

```
validation_rmse <- RMSE(predicted_ratings_validation, validation$rating)
```

Return the “validation_rmse” to show the RMSE output on validation set.

```
validation_rmse
```

```
## [1] 0.8293454
```