Capstone Project - MovieLens

Shweta Kothadiya

2/20/2021

For this project, we will be creating a movie recommendation system using the MovieLens dataset. We will use the 10M version of the MovieLens dataset to make the computation a little easier.

To begin with, we will install packages that has been mentioned under EDX project overview including – “tidyverse”, “caret”, and “data.table”.

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

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

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

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

Now, after all the packages are installed correctly, we downloaded the file. `"http://files.grouplens.org/datasets/movielens/ml-10m.zip"` This was the location of file “mi-10m.zip” which was downloaded and saved as tempfile into dl variable. Ratings were saved separately under ratings.

```
dl <- tempfile()
download.file("http://files.grouplens.org/datasets/movielens/ml-10m.zip", dl)
ratings <- fread(text = gsub("::", "\t", readLines(unzip(dl, "ml-10M100K/rati
ngs.dat"))),
                 col.names = c("userId", "movieId", "rating", "timestamp"))
```

Then we built the movies data set. Column names were declared. Then the "movielens" was created by doing left join on "ratings" and "movies".

```
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")
```

The seed was set based on the version that I have which is "4.0.3".

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

Then the partition was created using "movielens" data. "Validation" set will be 10% of MovieLens data and rest all will go into "edx" set.

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

We saved 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")
```

We compared "temp" and "validation" and any rows that were not joined based on the "movieId" and "userId" were removed and saved it under "removed" variable. Then those removed rows were added back into "edx"set. Then rm() function was used to delete the specified objects from the memory.

```
removed <- anti_join(temp, validation)
edx <- rbind(edx, removed)
rm(dl, ratings, movies, test_index, temp, movielens, removed)
```

RMSE function was created to use it later on to calculate the predicted ratings on the models that we would create vs the true ratings in the test set.

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

Creating partition on edx set to split into train and test set. 10% would go into 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 that the “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")
```

We compared “edx_test_set” and “edx_test_set_1” and any rows that were not joined based on the “movieId” and “userId” were removed and saved it under “removed_1” variable. Then those removed rows were added back into “edx_train_set”set.

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

We took the mean of ratings from edx_train_set and saved it as mu_hat. Return the output of mu_hat.

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

We got the mean result from mu_hat as “3.512509”.

Calculation of “Naive RMSE” -

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
```

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
```

Now, let’s take the mean of rating under edx_train_set and save it as mu.

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

MOVIE Effects Model”

Calculate average rating on edx_train_set based on Movie effect only.

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

Predict ratings on edx_test_set_1 based on movie_avgs model.

```
predicted_ratings <- mu + edx_test_set_1 %>% left_join(movie_avgs, by='movieI
d') %>%  .$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", R
MSE = 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 results of rmse_results is as above. 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.

MOVIE + USER Effects Model"

Calculate average rating based on Movie and User Effects Model.

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

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 M
odel", 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
```

Above is the results from “rmse_results” showing the name of the model and RMSE for each. 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 makes any difference.

MOVIE + Genres Effects Model”

Calculate average rating based on Movie and Genre Effects Model.

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

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
```

Above is the results from “rmse_results” showing the name of the model and RMSE for each. “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.

USER + Genres Effects Model”

Calculate average rating based on User and Genre Effects 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))
```

Predict ratings using this User plus Genre 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 User plus Genre 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 User plus Genre 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 M
odel", 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
```

Above is the results from “rmse_results” showing the name of the model and RMSE for each. 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”.

MOVIE + USER + Genres Effects Model”

Calculate average rating based on Movie and User and Genre Effects 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))
```

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
```

Above is the results from “rmse_results” showing the name of the model and RMSE for each. 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.

VALIDATION set – Final testing

Since the “Movie plus User Effects Model” is giving the least RMSE out of all the models, I am using that model for validation set.

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

Let’s predict the ratings on Validation set using “user_avgs_validation”.

```
predicted_ratings_validation <- validation %>% left_join(movie_avgs, by='movi
eId') %>% 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
```

Finally our Validation Set is showing RMSE as “0.8293454”.