---

title: "HarvardX: PH125.9x Data Science MovieLens Rating Prediction Project" author: "Joaquin Emilio Jaime Coronel" date: "May 20 2021" output: html_document

Load the edx & validation data sets using the provided script

Create edx set, validation set, and submission file

Note: this process could take a couple of minutes

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") if(!require(dplyr)) install.packages("dplyr", repos = "http://cran.us.r-project.org") if(!require(data.table)) install.packages("data.table", repos = "http://cran.us.r-project.org") if(!require(lubridate)) install.packages("lubridate", repos = "http://cran.us.r-project.org") if(!require(knitr)) install.packages("knitr", repos = "http://cran.us.r-project.org") if(!require(ggplot2)) install.packages("ggplot2", repos = "http://cran.us.r-project.org") if(library(kableExtra))install.packages("kableExtra", repos = "http://cran.us.r-project.org")

library(dplyr) library(tidyverse) library(caret) library(data.table) library(lubridate) library(knitr) library(kableExtra)

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("::", ", 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(movieId), title = as.character(title), genres = as.character(genres))

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) 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)

Let´s see the content of the variables: userId, movieId, rating, timestamp, title()

edx %>% as_tibble()

Introduction

#One of the challenges of machine learning is to be used successfully in the information technology #giving the ability to make recommendations systems of items to show to entrepreneurs a good choice #to make its product to be sold. #This project has to do with the Capstone Project a section part of the HarvardX PH125.9 Data Science #to showing the ability to make recommendation Systems using machine learning, algorithms to predict #movie ratings in the validation set.
#We use an open source dataset from movieLens 10M version of the MovieLens due to The Netflix data #is not freely available. #The goal of this project is to develop a machine learning algorithm using the inputs in one subset to #predict movie ratings in the validation set. The machine learning algorithm has been used applying #some algorithm models to show the evaluation through RMSE, the less the better.

The RMSE code

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

Methods and Analysis

Data Analysis

How many movies were rated with zero in the edx dataset?

```
sum(edx$rating[] == 0)
```

How many movies were rated with three in the edx dataset?

```
sum(edx$rating[] == 3)
```

How many movies were rated with four in the edx dataset?

```
sum(edx$rating[] == 4)
```

How many movies were rated with five in the edx dataset?

```
sum(edx$rating[] == 5)
```

How many different movies are in the edx dataset?

```
summarize(edx, n_movies = n_distinct(movieId))
```

How many different users are in the edx dataset?

```
summarize(edx, n_users = n_distinct(userId))
```

Which movie has the greatest number of ratings?

```
edx %>% group_by(movieId, title) %>% summarize(count = n()) %>% arrange(desc(count))
```

What are the five most given ratings in order from most to least?

edx %>% group_by(rating) %>% summarize(count = n()) %>% arrange(desc(count))

This is the subset that contain the six variables “userID”, “movieID”, “rating”,“timestamp”, “title”,

and “genres”. Each row represent a single rating of a user for a single movie.

head(edx) %>% print.data.frame()

A subset summary confirming that there aren´t missing values.

summary(edx)

Total of rows in the edx dataset

Tot_rows<- length(edx*rating*) + *length*(*validation*rating)

The total in the edx subset of unique movies is 10.700 and unique users is 70.000.

edx %>% summarize(n_users = n_distinct(userId), n_movies = n_distinct(movieId))

Ratings distribution

vec_ratings <- as.vector(edx$rating) unique(vec_ratings)

vec_ratings <- vec_ratings[vec_ratings != 0] vec_ratings <- factor(vec_ratings)
qplot(vec_ratings) + ggtitle(“Ratings’ Distribution”)

The distribution of each user’s ratings for movies. This shows the users bias.

edx %>% count(userId) %>% ggplot(aes(n)) + geom_histogram(bins = 30, color =
“black”) + scale_x_log10() + ggtitle(“Users”)

Distribution of each user’s ratings

edx %>% count(movieId) %>% ggplot(aes(n)) + geom_histogram(bins = 30, color = “black”) + scale_x_log10() + ggtitle(“Movies”)

Baseline Predictors

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

I- Movie effect

Calculate the average of all ratings of the edx dataset

mu <- mean(edx$rating)

Calculate b_i on the training dataset

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

Predicted ratings

predicted_ratings_bi <- mu + validation %>% left_join(movie_m, by=“movieId”) %>% .$b_i

II- Movie and user effect

Calculate b_u using the training dataset

user_m <- edx %>% left_join(movie_m, by=“movieId”) %>% group_by(userId) %>% summarize(b_u = mean(rating - mu - b_i))

Predicted ratings

predicted_ratings_bu <- validation %>% left_join(movie_m, by=“movieId”) %>% left_join(user_m, by=“userId”) %>% mutate(pred = mu + b_i + b_u) %>% .$pred

III- Movie, user and time effect

Create a copy of validation set , valid, and create the date feature which is the timestamp converted to a datetime object and rounded by week.

valid <- validation valid <- valid %>% mutate(date = round_date(as_datetime(timestamp), unit = “week”))12

Calculate time effects (b_t) using the training set

temp_m <- edx %>% left_join(movie_m, by=“movieId”) %>% left_join(user_m, by=“userId”) %>% mutate(date = round_date(as_datetime(timestamp), unit = “week”)) %>% group_by(date) %>% summarize(b_t = mean(rating - mu - b_i - b_u))

Predicted ratings

predicted_ratings_bt <- valid %>% left_join(movie_m, by=“movieId”) %>% left_join(user_m, by=“userId”) %>% left_join(temp_m, by=“date”) %>% mutate(pred = mu + b_i + b_u + b_t) %>% .$pred

The root mean square error (RMSE) models for movies, users and time effects

Calculate the RMSE for movies

rmse_model_1 <- RMSE(validation$rating,predicted_ratings_bi) rmse_model_1

[1] 0.9439087

Calculate the RMSE for users

rmse_model_2 <- RMSE(validation$rating,predicted_ratings_bu) rmse_model_2

[1] 0.8653488

Calculate the RMSE for time effects

rmse_model_3 <- RMSE(valid$rating,predicted_ratings_bt) rmse_model_3

[1] 0.8652511

From the movie and user effects combined, our RMSE decreased by almost 10% with respect to the only

movie effect. The improvement on the time effect is not significant,(about a decrease of 0.011%). The

regularization would be performed using only the movie and user effects.

Remove valid before regularization

rm(valid)

G. Regularization

Remembering that lambda is a tuning parameter. We can use cross-validation to choose it

lambdas <- seq(0, 10, 0.25)

rmses <- sapply(lambdas, function(l){ mu_reg <- mean(edx$rating)b_{i_r}eg < -edx$pred return(RMSE(validation$rating,predicted_ratings_b_i_u)) })

qplot(lambdas, rmses)

The optimal lambda for the full model

For the full model, the optimal Î» is given as

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

[1] 5.25

rmse_model_4 <- min(rmses) rmse_model_4

[1] 0.864817

Summarizing all the rmse on validation set for Linear regression models

rmse_results <- data.frame(methods=c("movie effect","movie + user effects","movie + user + time effects", "Regularized Movie + User Effect Model"),rmse = c(rmse_model_1, rmse_model_2,rmse_model_3, rmse_model_4)) kable(rmse_results) %>% kable_styling(bootstrap_options = "striped" , full_width = F , position = "center") %>% kable_styling(bootstrap_options = "bordered", full_width = F , position ="center") %>% column_spec(1,bold = T) %>% column_spec(2,bold =T ,color = "white" , background ="#D7261E")

methods rmse movie effect 0.9439087 movie + user effects 0.8653488 movie + user + time effects 0.8652511 Regularized Movie + User Effect Model 0.8648170

The regularization gets down the RMSE's value to 0.8648170.

Interpretation of Results

The last model was the one that threw the lowest RMSE, it means that we can use it as the better result on movie recommendation system to apply in this project. Although the second and third model to the last one are not so far from it, they can´t be chosen to recommend.

Conclusion

We can conclude that we have done an algorithm that got a lower RMSE, each time we apply the different models, using the base line predictors as wit: User´s effect, movie effect, and time effect and lambda model.

The last calculus gave us the lower result using the algorithm presented in any case model, accompli- shing the goal of this project, I mean the lower the better in the RMSE number

Moreover, we can say that we could use other procedures using different algorithms but the compu- ting limitation do not let us do much more.