MovieLens Project

Agustín Villalobos

20/2/2022

SUMMARY

My project uses the edx dataset for the anlysis. Uses the “Guerrero” method (Guerrero’s (1993)), I obtain the lambda value that minimizes the coefficient of variation for the subseries of the ratings in the edx dataset.

Then, I apply a BoxCox distribution to correct biases in the error distribution, correct unequal variances and correct the nonlinearity in the relationship, generating a set of predictions for the ratings of the movies.

Finally, using the MSRE algorithm I obtain the resulting mean square error between the original ratings of the edx set and the ratings obtained in the BoxCox prediction.

```
############################################################
# Create edx set, validation set (final hold-out test set)
############################################################

# Note: this process could take a couple of minutes

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

```
## Loading required package: tidyverse
```

```
## ── Attaching packages ─────────────────────────────────────── tidyverse 1.3.1 ──
```

```
## ✓ ggplot2 3.3.5     ✓ purrr   0.3.4
## ✓ tibble  3.1.6     ✓ dplyr   1.0.8
## ✓ tidyr   1.2.0     ✓ stringr 1.4.0
## ✓ readr   2.1.2     ✓ forcats 0.5.1
```

```
## ── 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-projec
t.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)

# 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("/home/agustin/Escritorio/MovieLens/m
l-10M100K/ratings.dat")),
                 col.names = c("userId", "movieId", "rating", "timestamp"))

movies <- str_split_fixed(readLines("/home/agustin/Escritorio/MovieLens/ml-10M100K/movie
s.dat"), "\\::", 3)
colnames(movies) <- c("movieId", "title", "genres")


#ratings <- fread(text = gsub("::", "\t", readLines(unzip(dl, "ml-10M100K/ratings.da
t"))),
#                 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 3.6 or earlier:
movies <- as.data.frame(movies) %>% mutate(movieId = as.numeric(levels(movieId))[movieI
d],
                                           title = as.character(title),
                                           genres = as.character(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
```

```
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)
```

```
## Warning in rm(dl, ratings, movies, test_index, temp, movielens, removed): objeto
## 'dl' no encontrado
```

```
#USING A TEST DATA MODEL

#Necesary Libraries
library(forecast)
```

```
## Registered S3 method overwritten by 'quantmod':
##   method            from
##   as.zoo.data.frame zoo
```

```
library(Metrics)
```

```
##
## Attaching package: 'Metrics'
```

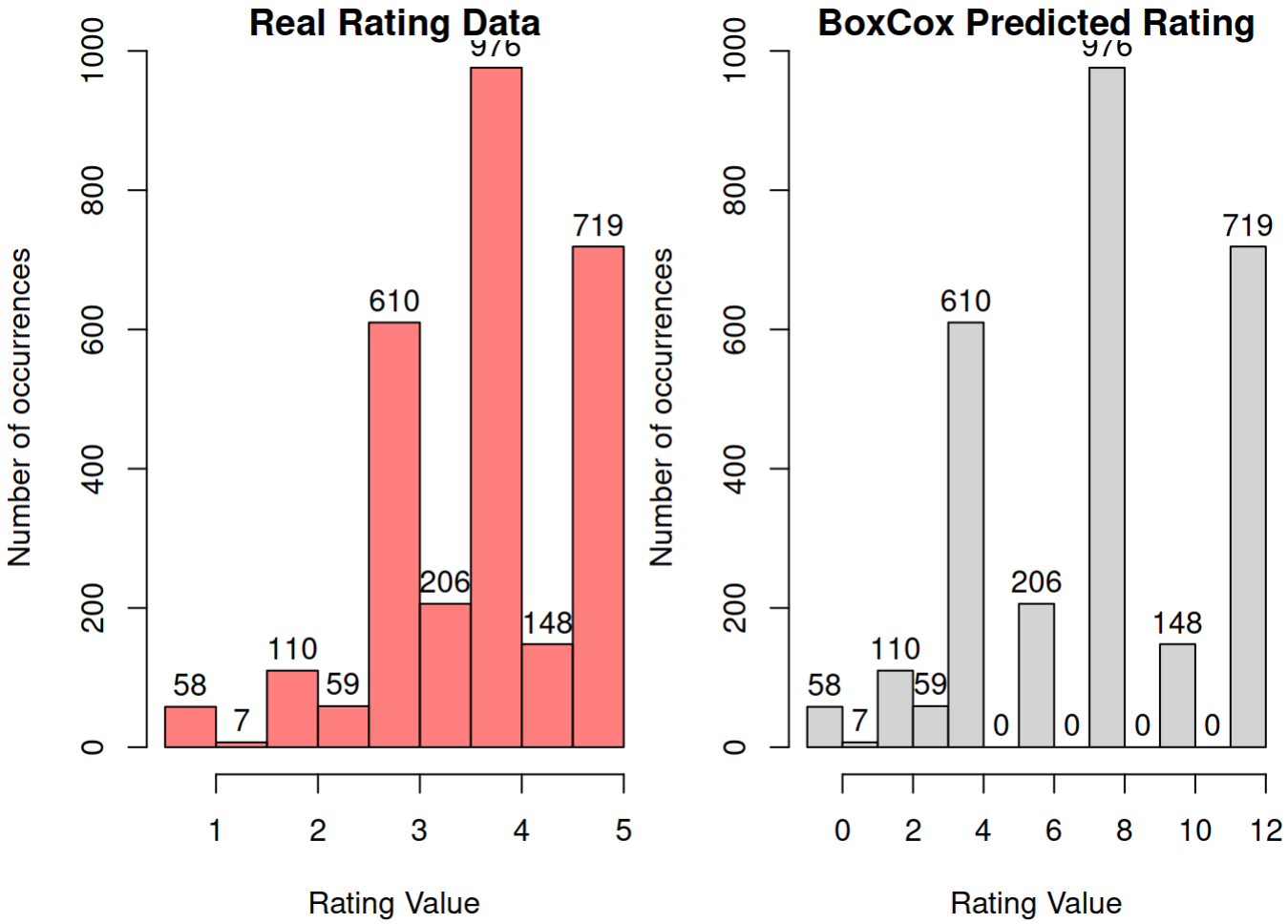
```
## The following object is masked from 'package:forecast':
##
##     accuracy
```

```
## The following objects are masked from 'package:caret':
##
##     precision, recall
```

```
#Create a training and test set
df_test <-filter(edx, movieId==122 | movieId==185)

#Obtain the best lambda with guerrero method (Guerrero's (1993))
lambda = BoxCox.lambda(df_test$rating, method="guerrero")

#Obtain then resulting mean square error, based en edx ratings and the prediction model
 generate by BoxCox
rmse <- sapply(lambda,function(l){
 predicted_ratings = BoxCox( df_test$rating, lambda)
 return(rmse(df_test$rating , predicted_ratings))
})

#Print the resultant rmse value
paste('The optimal RMSE of ',rmse,' is achieved with Lambda ',lambda)
```

```
## [1] "The optimal RMSE of  0.694496372096233  is achieved with Lambda  1.2017607680894
6"
```

```
#-------------------------------------------------------------------------------------
-----------------------------------
#USING THE VALIDATION SET

#Graphical representation of the actual model and the BoxCox representation

library(forecast)
library(Metrics)

#The Validation dataset needs a lot of process that I don't have on my personal compute
r, so in the deliverable I show an example analysis with a segment of it. I #invite my c
olleagues to use the entire data set if their possibilities allow it.
df_Resume <-filter(validation, movieId==122 | movieId==1)

#Obtain the best lambda with guerrero method (Guerrero's (1993))
lambda = BoxCox.lambda(df_Resume$rating, method="guerrero")
predicted_ratings = BoxCox( df_Resume$rating, lambda)

Ixos=rnorm(4000 , 120 , 30)
Primadur=rnorm(4000 , 200 , 30)

par(
  mfrow=c(1,2),
  mar=c(4,4,1,0)
)
hist(df_Resume$rating,col=rgb(1,0,0,0.5) , xlab="Rating Value" , ylab="Number of occurre
nces" , main="Real Rating Data",labels = TRUE)
hist(predicted_ratings, xlab="Rating Value" , ylab="Number of occurrences" , main="BoxCo
x Predicted Rating",labels = TRUE )
```




```
#USING THE VALIDATION SET

#Obtain then resulting mean square error, based en edx ratings and the prediction model
 generate by BoxCox
rmse <- sapply(lambda,function(l){
  predicted_ratings = BoxCox( df_Resume$rating, lambda)
  return(rmse(df_Resume$rating , predicted_ratings))
})

#Print the resultant rmse value
paste('The optimal RMSE of ',rmse,' is achieved with Lambda ',lambda)
```

```
## [1] "The optimal RMSE of  4.26768710301546  is achieved with Lambda  1.9999339590060
9"
```

CONCLUSION

With the use of some existing libraries, efficient data analysis can be performed. This has left me with the teaching that as a student I should not stop learning and researching about different tools and functionalities that can be exploited to perform data analytics.