Data Science Capstone Project - MovieLens

Justo Andrés Manrique Urbina

7 de junio de 2019

Introduction

This document narrates the creation process of a recommendation engine, whose purpose is to predict how will a user rate a specific movie. With this in mind, we use the MovieLens movies rating dataset, which is maintained by GroupLens Research Lab. From this dataset, we create our training and test set. Creating this recommendation engine is part of the capstone project for HarvardX Data Science Professional Certificate.

The process of creating a recommendation engine comprises exploratory data analysis and modelling effects that helps predicting ratings for a given rating. To assess the performance of our model, we will use RMSE metric.

In exploratory data analysis, we develop an understanding of the following:

- Understand the overall distribution of ratings, users and ratings per movie.
- Understand the possible effects each variable has on ratings.
- Evaluate if we need regularization for small samples penalization.

After exploratory data analysis we will iterate different models and improve RMSE, and compare each model.

Finally, we will conclude what's the best model given our RMSE and discuss further steps that can be taken to improve score.

Having said this, let's start working!

MovieLens analysis

To start analysis, we'll import data with the following code:

```
library(dslabs)
library(caret)
library(tidyverse)

rm(list=ls())

setwd("C:/Users/Justo.Manrique/Documents/Capstone-HarvardX")
edx <- readRDS("edx.rds")
validation <- readRDS("validation.rds")
```

Understanding our data

First, we get a glimpse of how the table is structured using the formula head. From this, we understand there is some cleaning to do: (1) "genres" field has different values that are separated by a pipe and (2) timestamp has to be converted to a readable format to take into account any time effect. We also understand the nature of our variables, which are:

- userId is a nominal variable which serves as a user identifier.
- movieId is a nominal variable which serves as a movie identifier.
- rating is a numeric variable which serves as a rating for a given movie and given user. This is what we'll predict.
- timestamp is a numeric variable which servers as a date identifier.

- title is a nominal variable (and a string) which serves as movie description.
- genres is a nominal variable which serves as genre identifier for each movie.

```
head(edx)
```

```
##   userId movieId rating timestamp                         title
## 1      1     122      5 838985046              Boomerang (1992)
## 2      1     185      5 838983525               Net, The (1995)
## 4      1     292      5 838983421               Outbreak (1995)
## 5      1     316      5 838983392               Stargate (1994)
## 6      1     329      5 838983392 Star Trek: Generations (1994)
## 7      1     355      5 838984474       Flintstones, The (1994)
##                           genres
## 1                Comedy|Romance
## 2          Action|Crime|Thriller
## 4   Action|Drama|Sci-Fi|Thriller
## 5        Action|Adventure|Sci-Fi
## 6 Action|Adventure|Drama|Sci-Fi
## 7        Children|Comedy|Fantasy
```

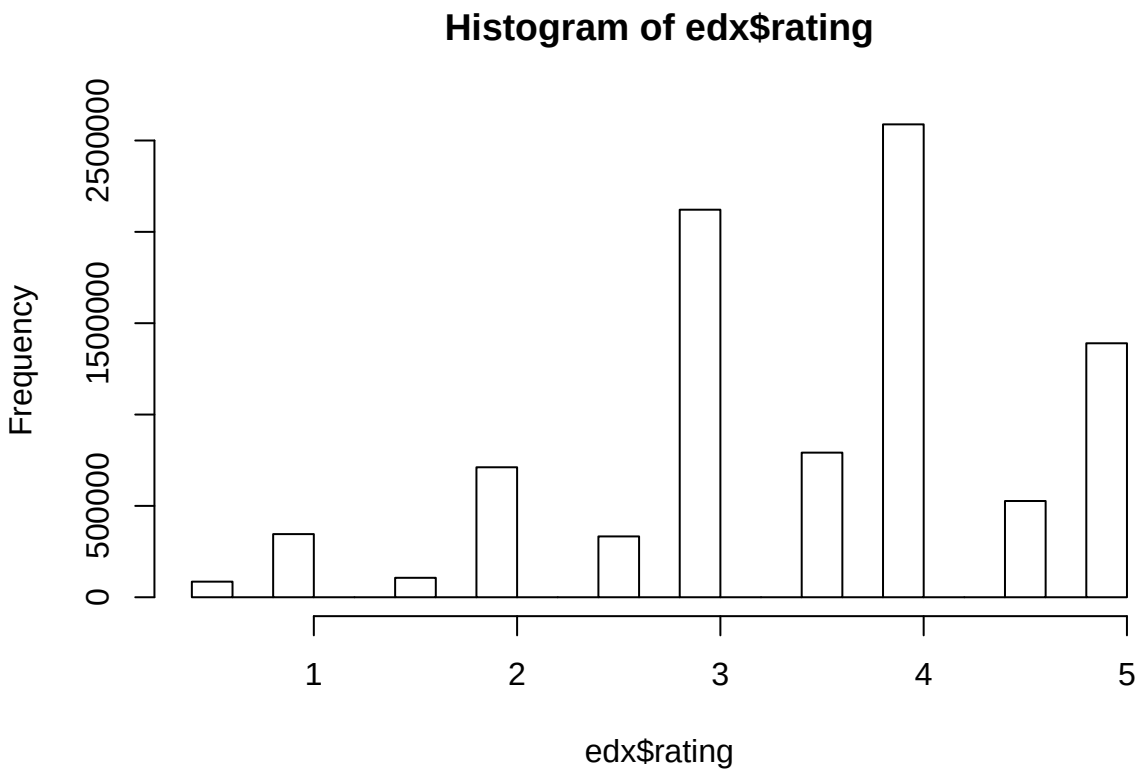
After this, we get descriptive statistics for our dataset. Given the nature of our variables, we will only focus in ratings summary statistics. From the 1st to 3rd quartile, we see that ratings are condensed in 3 to 4 values. We will look further into this.

```
summary(edx)
```

```
##      userId         movieId          rating        timestamp
##  Min.   :    1   Min.   :    1   Min.   :0.500   Min.   :7.897e+08
##  1st Qu.:18124   1st Qu.:  648   1st Qu.:3.000   1st Qu.:9.468e+08
##  Median :35738   Median : 1834   Median :4.000   Median :1.035e+09
##  Mean   :35870   Mean   : 4122   Mean   :3.512   Mean   :1.033e+09
##  3rd Qu.:53607   3rd Qu.: 3626   3rd Qu.:4.000   3rd Qu.:1.127e+09
##  Max.   :71567   Max.   :65133   Max.   :5.000   Max.   :1.231e+09
##     title              genres
##  Length:9000055     Length:9000055
##  Class :character   Class :character
##  Mode  :character   Mode  :character
##
##
##
```

By using an histogram, we can confirm that most of the ratings are in that interval.

```
hist(edx$rating)
```



Understanding users

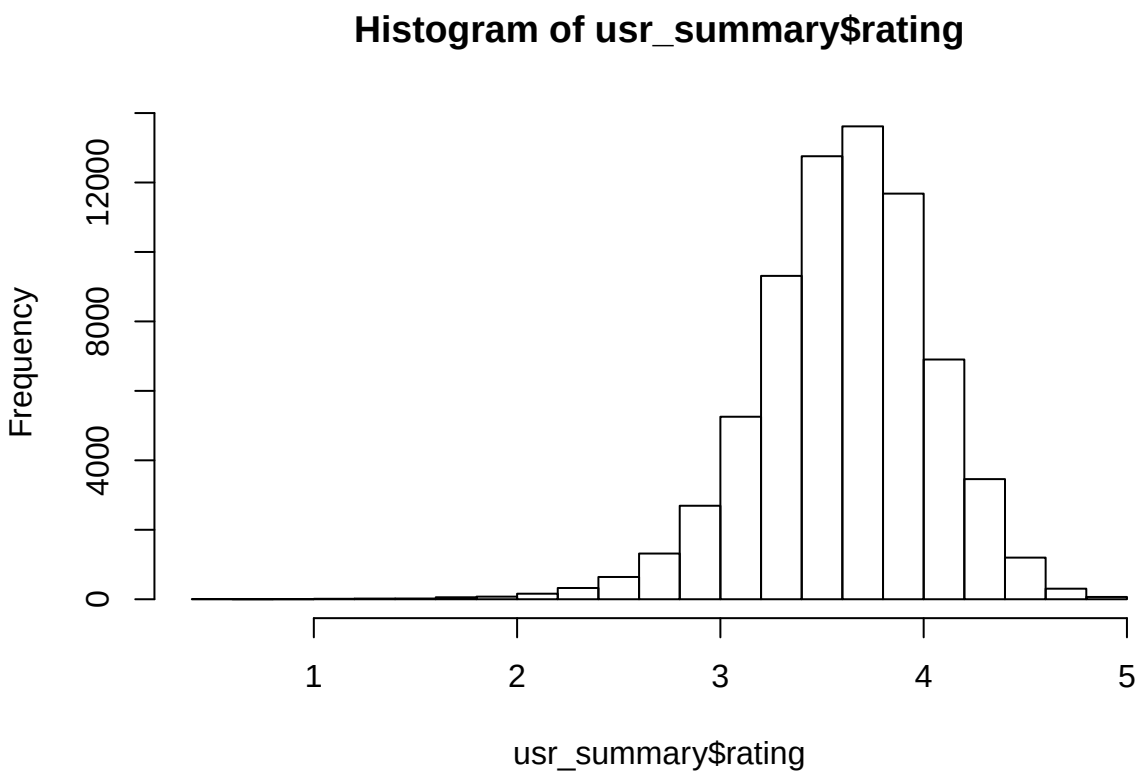
After this we can then understand how the users behave. For this, we create a summarization of the average rating and ratings count by userId. From this we gather:

- There are users that have as little as 10 ratings and as big as 6,116 ratings. As there are users that has little rating, there's a use case for regularization.
- The ratings overall distribution (which is centered at 3.5) also replicates at the user level.

```
usr_summary <- edx %>% group_by(userId) %>% summarise(rating = mean(rating),ratingscount=n())
summary(usr_summary)
```

```
##      userId          rating       ratingscount
##  Min.   :    1   Min.   :0.500   Min.   :  10.0
##  1st Qu.:17943   1st Qu.:3.357   1st Qu.:  32.0
##  Median :35799   Median :3.635   Median :  62.0
##  Mean   :35782   Mean   :3.614   Mean   : 128.8
##  3rd Qu.:53620   3rd Qu.:3.903   3rd Qu.: 141.0
##  Max.   :71567   Max.   :5.000   Max.   :6616.0
```

```
hist(usr_summary$rating)
```



Understanding movies

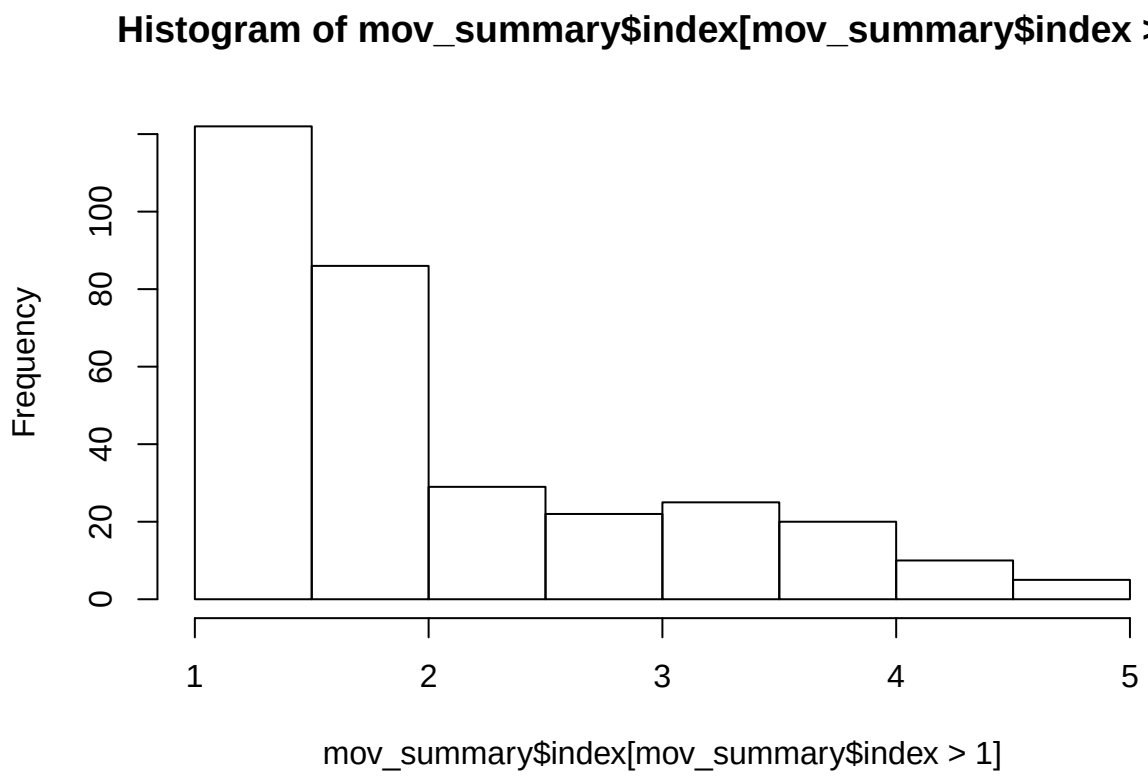
We'd like to understand ratings per movie. For this, we create a summarization of the average rating, median rating, max and min rating and an index, which is the average rating divided by ratings counts. This index is an indicator of ratings small sample sizes. If the index is greater than 1, it means that there is small proportion of average ratings by ratings count.

From this we gather:

- The ratings overal distribution also replicates at the movies level.
- There is approximately 300 movies that has an index greater than 1. Because of this, there's a use case for regularization.

```
mov_summary <- edx %>% group_by(movieId, title) %>% summarise(n_ratings = n(),avg_ratings = mean(rating

mov_summary <- mov_summary %>% arrange(avg_ratings,n_ratings)
hist(mov_summary$index[mov_summary$index>1])
```



This exploratory data analysis helps us understand that there's an user and model effect on rating, which will use for our model.

Defining our model

We'll predict the ratings taking into account first the movie effect. Based on the lectures, we create our RMSE function to assess the performance of our model and also a table object where we will store our results.

To model the movie effect, we create a variable that tell us how apart we are from the overall rating mean based on the movie. Then we add this to the overall rating mean for each rating and left join this with the validation set. With this, we model the full movie effect. We assess our model performance only using the movie effect and get a RMSE of approximately ~0.94.

See code below:

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

mean <- mean(edx$rating)
mov_effect <- edx %>% group_by(movieId) %>% summarise(b_i = mean(rating-mean))
mov_effect <- cbind(mov_effect,mean) %>% mutate(prd = b_i + mean) %>% select(-b_i,-mean)
validation <-  validation %>%  left_join(mov_effect,by='movieId')

mov_rmse <- RMSE(validation$rating,validation$prd)
rmse_table <- data.frame(method = 'Movie effect', RMSE=mov_rmse)
rmse_table
```

```
##         method      RMSE
## 1 Movie effect 0.9439087
```

Because our RMSE isn't adequate, we will add user effects to our current model. We do the same process here (we create a variable that tell us how apart we are from the overall rating mean based on the user). We then add this to the prediction created before and we left join this with the validation set. With this, we model both user and movie effect. We assess our model performance by using the RMSE function. Our model improves with a RMSE of approximately ~0.88.

```
usr_effect <- edx %>% group_by(userId) %>% summarise(u_i = mean(rating-mean))
validation <- validation %>% left_join(usr_effect,by='userId')
validation <- validation %>% mutate(prd_2 = prd + u_i) %>% select(-u_i)

movusr_rmse <- RMSE(validation$rating,validation$prd_2)
rmse_table <- bind_rows(rmse_table,data.frame(method = 'Movie and user effect', RMSE=movusr_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
```

```
rmse_table
```

```
##                   method      RMSE
## 1          Movie effect 0.9439087
## 2 Movie and user effect 0.8850398
```

Because we saw there are movies that have small sample sizes, and users too, we'll do regularization on the current model. This will give us accurate estimates because we'll penalized small sample ratings.

```
lambda <- seq(0,10,0.10)
rmses_u_i <- sapply(lambda,function(l){
  mean_f <- mean(edx$rating)
  b_i <- edx %>% group_by(movieId) %>% summarise(b_i=sum(rating-mean_f)/(n()+l))
  b_u <- edx %>% left_join(b_i, by='movieId') %>% group_by(userId) %>% summarise(b_u=sum(rating- b_i - r

  prd_r <- validation %>% left_join(b_i, by='movieId') %>% left_join(b_u,by='userId') %>% mutate(prd_r=r

  return(RMSE(prd_r$prd_r,prd_r$rating))
})

lambda[which.min(rmses_u_i)]
```

```
## [1] 5.2
```

```
reg_l <- lambda[which.min(rmses_u_i)]

mov_reg_effect = edx %>% group_by(movieId) %>% summarise(b_i = sum(rating-mean)/(n()+reg_l))
mov_usr_reg_effect = edx %>% left_join(mov_reg_effect, by='movieId') %>% group_by(userId) %>% summarise

validation <- validation %>% left_join(mov_reg_effect,by='movieId') %>% left_join(mov_usr_reg_effect,by=
validation <- validation %>% mutate(prd_reg = mean + b_i + b_u)
validation <- validation %>% select(-b_i,-b_u)

RMSE(validation$prd_reg,validation$rating)
```

```
## [1] 0.864817
```

```
mov_usr_reg_rmse <- RMSE(validation$rating,validation$prd_reg)
rmse_table <- bind_rows(rmse_table,data.frame(method = 'Movie effect and user effect, regularized', RMSE
```

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

```
rmse_table
```

```
##                                     method      RMSE
## 1                              Movie effect 0.9439087
## 2                    Movie and user effect 0.8850398
## 3 Movie effect and user effect, regularized 0.8648170
```

We see that our RMSE improve slightly, to approximately ~0.86, which is sufficient for our purpose.

Results

- Movie and user effects gives information on whether our prediction is accurate, since adding their effects improves RMSE.
- Since we found out regularization is needed, penalizing small samples will yield better predictions. This gave us a better RMSE when using regularized movie and user effects.

Conclusion

- We should take into account movies and users for this type of recommendation engines since it gives information on user preferences and movies quality. While this sounds rather obvious, it's best if we can use data to confirm this.
- Ratings seem rather consistent at each level (in the three levels - overall, movie and user wise -).

Next Steps

To further improve RMSE, we could take into account movies' genres, movie's release date. If possible, we can augment data in this dataset to yield better predictions. For example, we can get movie's author and starring actor, movie's cost and much more.