Applying Recommendation Systems approach on Movielens Data

First Capstone Project for Data Science Professional Certificate
Harvardx Professional Certificate Program

Felipe Vergara, M.Sc

Submission Date
September 17th, 2021

Summary

1 Introduction

Nowadays big companies are available to collect massive amount of customer data which can be useful to improve their products and services. Generally, one important approach is to make a simple and clear experience to users by filtering vital information, according to user's preferences, interest or behaviour to an item/product (F. O. Isinkaye, 2015). As a result, a customer finds easily an item related to his preference.

The present document is one of the items of the first **Capstone** project and part of the requirements to obtain the **HarvardX Data Science Certificate** (For further details, as the production of the training and test data, please look at Movie_Rating_FV). It presents the elaboration of a recommendation system using the Movielens data, obtained from this website. The aim is to predict how a user will rate a specific movie according to specific dataset characteristics such as average rating, average user rating, genre effect, etc.

There are different *Movielens* data versions, however, to make a simple analysis it was used the 10M version, released in January 2009. At difference with other versions, no demographic information is included, being the user represented by an ID.

In the next sections are presented the data exploration on different variables, then is applied the methodology applied by (Irizarry, 2021), extending the analysis including the genre effect variable. Afterwards, it is included the penalization approach, calculating λ for the variables user and movies through cross-validation. Finally, it is displayed the results and conclusions, fulfilling the requirement for this first Capstone project.

2 Analysis

2.1 Libraries and loading data

Diverse tools are necessary to run the project, thus the libraries here below where used through the project:

```
library(tidyverse)# organization and visualization data
library(caret)# machine learning procedure
library(kableExtra) #for table presentations
library(rlist)# more functions for list management
library(lubridate)#to manage time data
```

Likewise, the training and test data and other resources must be loaded at this stage, instead of being produced within this project, due to hardware processing.

2.2 Dataset description and sorting data

The dataset is compounded by ratings of users to movies from a score of 0 (very bad) to 5 (very good). It contains more than 10 million ratings of of 10681 movies done by 71567 users (F. M. Harper, 2015).

Table 1: Training data dimension

rows	columns
9,000,055	6

As the recommendation system is a machine learning application, it must be created a training data and testing data. **the former is for the algorithm creation, and the latter is for assessing our final algorithm**. Generally,

during the procedure is used mostly the training data which corresponds to the 90% of the dataset. The training dataset dimensions can be seen in Table 1.

As it can be seen, it is the 90 % of the raw data and consists of six columns. Below is presented an extract of the training data, being possible to identify the disposable variables in Table 2:

Table 2: Dataset example

userId	movieId	rating	timestamp	title	genres
1	122	5	838,985,046	Boomerang (1992)	Comedy|Romance
1	185	5	838,983,525	Net, The (1995)	Action|Crime|Thriller
1	292	5	838,983,421	Outbreak (1995)	Action|Drama|Sci-Fi|Thriller
1	316	5	838,983,392	Stargate (1994)	Action|Adventure|Sci-Fi
1	329	5	838,983,392	Star Trek: Generations (1994)	Action|Adventure|Drama|Sci-Fi

2.3 Exploration

2.3.1 Users & Movies

In this dataset, at least an user had rated 20 movies, however, due to the subdivision, this rating rate could have been decreased. Therefore, it was necessary to have a frame of how many users and movies are included in the training data (Table 3).

Table 3: User and movies numbers

users	movies
69,878	10,677

Additionally, let see how is the rating rate (Figure 1):

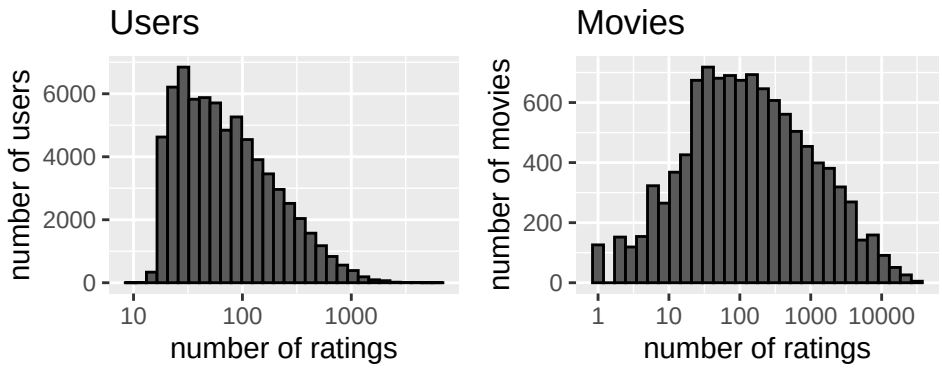

Figure 1: Count given by users (top) and by movies (bottom)

It can be seen that users are concentrated around 80 ratings per user and a movie was rated in average with 100 ratings. As a result, this indicate that there are a diverse user rating in the dataset, as well with the movies, which it could have an influence in how ratings were done. The next step so was to explore the rating average according to the mentioned above (Figure 2).

As it can be observed,there is a certain pattern of how users rate movies and how movies are rated. For example, both extremes (great and bad movies) are rated in much less quantity than regular movies. This is a similar pattern with users. Thus, both variables will be included in the algorithm (see Predicting algorithm section).

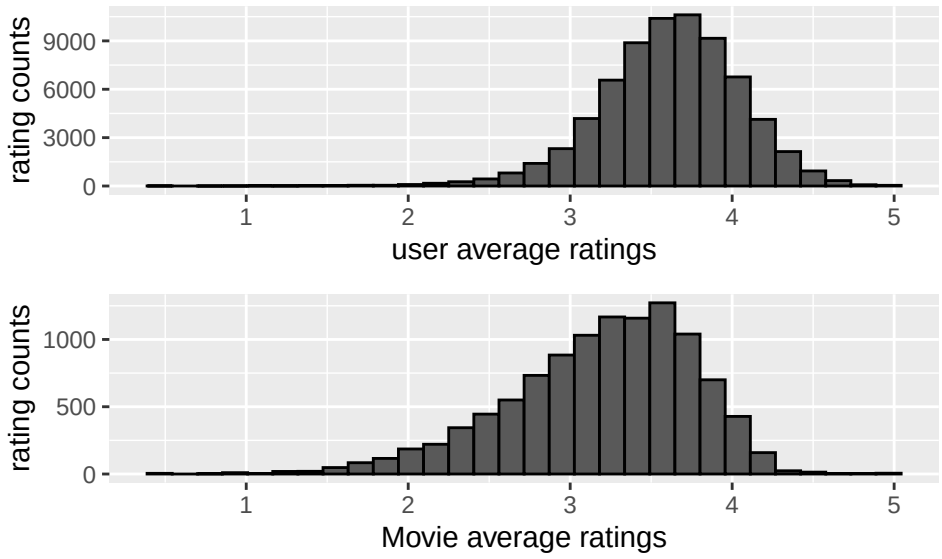

Figure 2: Average rating frequency given by users (top) and by movies (bottom)

2.3.2 Genre

Let evaluate if there is a genre effect over the data. First of all, it must be identified how many genres are in the dataset (Table 4).

```
##movies genre classification
#How many genres are in the dataset?
sep_genre<-list.append(str_split(edx$genres,'\\|'))
genres_edx<-unique(combine(sep_genre))
save(genres_edx,file="Capstone/rda_data/genres_edx.rda")
load(file="Capstone/rda_data/genres_edx.rda")
rm(sep_genre)
```

Table 4: Unique genre in edx dataset

Comedy	Thriller	Children	Musical	Horror
Romance	Drama	Fantasy	Western	Documentary
Action	Sci-Fi	War	Mystery	IMAX
Crime	Adventure	Animation	Film-Noir	(no genres listed)

20 genres can be discriminated, but as customer, it is known that the film industry predominates genres such as comedy, romance and action, while the rest is less present. So, how many ratings are for each of them? (Table 5).

Table 5: Rating per genre

genre	rating count	genre	rating count
Drama	3,910,127	Horror	691,485
Comedy	3,540,930	Mystery	568,332
Action	2,560,545	War	511,147
Thriller	2,325,899	Animation	467,168
Adventure	1,908,892	Musical	433,080
Romance	1,712,100	Western	189,394
Sci-Fi	1,341,183	Film-Noir	118,541
Crime	1,327,715	Documentary	93,066
Fantasy	925,637	IMAX	8,181
Children	737,994	(no genres listed)	7

As the assumption indicated, there are genres that predominates in the dataset. So, let see how is the rating among them. Is there a significant difference among them?

In this case, to set the script, it was adapted from (de Oliveira, 2021) algorithm:

```
#How is the rating for each genre? are there a difference among them?
#Identifying average rating
#table average, se (adapted from dfedeoli calculations)
genre_rating<-matrix(nrow=20, ncol=3)#matrix with ave and se values
j<-1 #fundamental for looping
for (i in genres_edx){
  grating<-edx[which(str_detect(edx$genres,i)),]%>%
    summarize(n = n(), avg = mean(rating), se = sd(rating)/sqrt(n()))#calculating value
  genre_rating[j,1]<-grating$avg#avg value for each genre
  genre_rating[j,2]<-grating$se# se value for each genre
  genre_rating[j,3]<-i# genre label
  j<-j+1
}
```

To make much plausible the figure, it was filtered out “no genres listed”,because it has a large standard error, due to the few movies associated with them (n=7), (Figure 3).

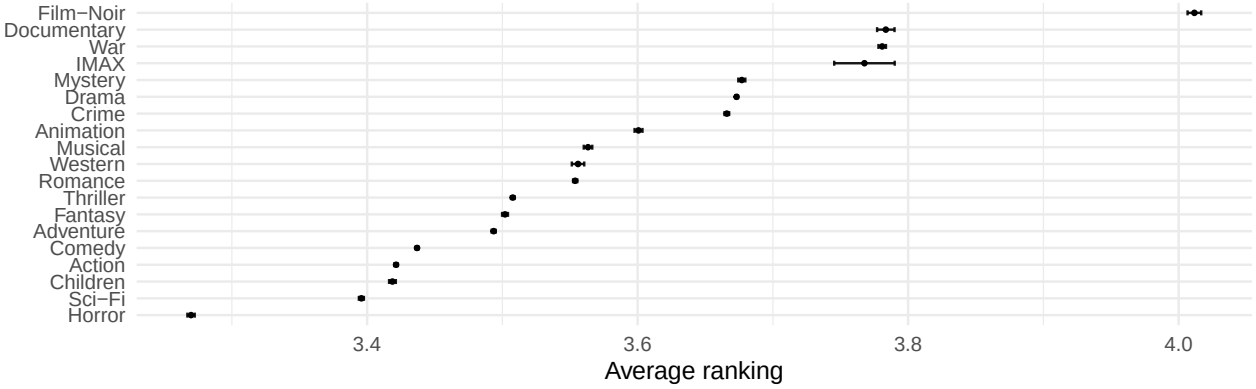

Figure 3: Average rating for each genre

As a result, it seems that there is a genre effect (a minimal one), so it will be included in the algorithm (see Predicting algorithm section).

2.3.3 Rating date

The last analysed variable was the rating date. As it could be seen in the Table 2, the rating date is "timestamp" column, which represents date as integer. So, this column data was transformed to "POSIXct".

```
#Rating date
rat_time<-edx%>%mutate(daterat=as_datetime(timestamp))%>%
  select(rating,daterat,genres)
rat_time$ym<-round_date(rat_time$daterat,unit="month")
rat_time$season<- quarter(rat_time$daterat)
save(rat_time,file="Capstone/rda_data/rat_time.rda")
```

Afterwards, to have an idea of how the data is concentrated, it was aggregated in months and displayed by a boxplot to identify anomalies.

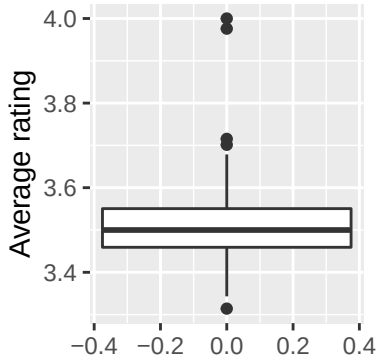

Figure 4: rating date distribution

The first conclusion from this figure is that the data is that ratings are concentrated below 3.8. Let see, how the rating date has been distributed from the beginning to the end date frame (Figure 5).

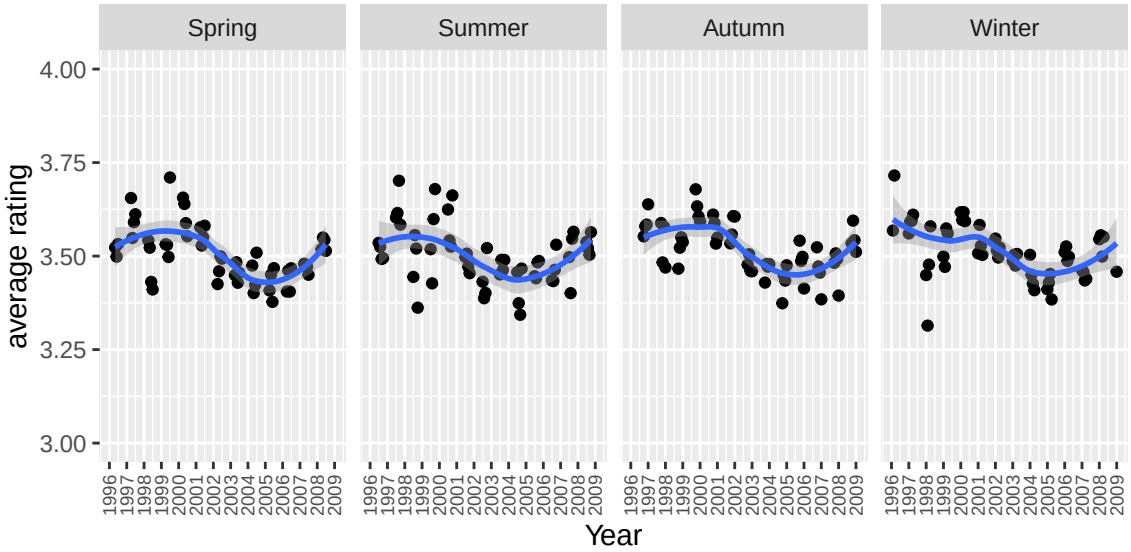

Figure 5: Rating through dates. Aggregated by months

In this figure, the data was organised through seasons, notwithstanding, a relevant point is that the dataset as it does not have demographic information and localisation, it is not known from which hemisphere the data

comes from. So, the season analysis does not provide realistic information, but the distribution here it shows that is a bit less significant as the genre effect. As a result it was not considered for the algorithm.

2.4 Predicting algorithm

In relation to the previous section, it was decided to use movie, user and genre effect variables to create the algorithm. In this sense, it was decided to use the suggestion presented in (Irizarry, 2021), (Equation (1)):

$$Y_{u,i} = \mu + b_i + b_u + \sum_{k=1}^{K} x_{u,i}^{k} \beta_k + \varepsilon_{u,i} \tag{1}$$

with $x_{u,i}^{k}$ = 1 if $g_{u,i}$ is genre k
with $x_{u,i}^{k}$ = 0 if $g_{u,i}$ is not genre k

Where:
$y_{u,i}$: Actual rating
μ: average of all ratings
b_i: average ranking for movie$i$ or *bias*
b_u: user-specific effect$u$
b_k: genre effect$k$
$\sum_{k=1}^{K} x_{u,i}^{k}$: sum of each genre effect of a x rating
$\varepsilon_{u,i}$: independent errors sampled taking into account previous parameters.

When is considered as prediction, the $\varepsilon_{u,i}$ is out. So, the formula indicates that the average ($\hat{\mu}$) of all ratings is the simplest approach to predict a certain rating. However, there were movies that are rated higher than others and also users who rated extremely positive and negative than others (Figure 1). Therefore, to integrate the mentioned effect, b_i refers to the average ranking of movie$_i$ (Equation (2)). On the other side b_u represents the average rating of user$_u$ corrected by b_i (Equation (3)).

$$\hat{b}_i = \frac{1}{N} \sum_{i} Y_{u,i} - \hat{\mu} \tag{2}$$

$$\hat{b}_u = \frac{1}{N} \sum_{u} Y_{u,i} - \hat{\mu} - \hat{b}_i \tag{3}$$

2.4.1 Genre effect $\hat{b}_k$ and Sum

At difference with $\hat{b}_i$ and $\hat{b}_u$, there is not only 1 genre effect $\hat{b}_k$, instead it must be computed the sum of the 20 genre effect over a rating. This calculation was highly memory demanding and it was a huge handicap for this project, since it had to be subdivided the training dataset to be computed.

The $\hat{b}_k$ calculation was organised in the following way:

2.4.1.1 Subdivision

To decrease the memory RAM consumption to an acceptable performance, the data was subdivided through genre type, in other words, in 20 times, being each new dataset only movies of a corresponding genre.

```
#Due to hardware processing ( it was impossible to apply at 1 shot)
#1.- start  subdividing the dataset according to each genre
edx_<-list()
for (i in 1:length(genres_edx)){
  a<-as.data.frame(sapply(genres_edx[i], function(x) {
    str_detect(edx$genres, x)}))
  edx<-cbind(edx,a)
  edx_[[i]]<-edx%>%filter(!!sym(genres_edx[i])== TRUE)
  edx<-select(edx,-7)
}
rm(a)
save(edx_,file="Capstone/rda_data/edx_.rda")
load(file="Capstone/rda_data/edx_.rda")
```

2.4.1.2 Establish $\hat{b}_k$ for each genre

Afterwards, it was implemented The $\hat{b}_k$ formula (Equation (4)), which was exported as a list file.

$$\hat{b}_k = \frac{1}{N}\sum_k Y_{u,i} - \hat{\mu} - \hat{b}_i - \hat{b}_u \tag{4}$$

```
#2.- establish $hat{b}_k$ for each genre
mu <- mean(edx$rating)
b_i <- edx %>%
  group_by(movieId) %>%
  summarize(b_i = mean(rating - mu))
b_u <- edx %>%
  left_join(b_i, by="movieId") %>%
  group_by(userId) %>%
  summarize(b_u = mean(rating - b_i - mu))

b_k.list<-list()
for(i in 1:length(genres_edx)){
b_k <- edx_[[i]] %>%
  left_join(b_u, by="userId") %>%
  left_join(b_i, by="movieId")%>%
  summarize(b_k = mean(rating - b_i - b_u - mu))
b_k.list[[i]]<-b_k
}
save(b_k.list,file="Capstone/rda_data/b_k.list.rda")
load(file="Capstone/rda_data/b_k.list.rda")
rm(b_i,b_k,b_u,mu,edx_)
```

2.4.1.3 Calculation sum $\hat{b}_k$ for each rating

The aim was to create a new column in the training data, being the sum of each genre $\hat{b}_k$ value of each rating (Equation(5)). First, it was created a matrix (rows = ratings, columns = genre) with values of 1 if genre presented, and 0 when genre is not presented in a rating. Then it was multiplied with the $\hat{b}_k$, which was converted too in a matrix.

$$\sum_{k=1}^{K} x_{u,i}^{k} \hat{b}_k \tag{5}$$

with $x_{u,i}^{k}$ = 1 if $g_{u,i}$ is genre k
with $x_{u,i}^{k}$ = 0 if $g_{u,i}$ is not genre k

The result of this operation is a new matrix of 1 column which was combined with the training data. However, due to hardware processing, the data training data was divided 10 times (independently from the previous genre subdivision). The next script is an extract of all the process. Here is presented 1 of the 10 subdivisions.

```
#is done in this way to not overwhelm hardware processing
m_kvalues<-matrix(unlist(b_k.list))
#subdivide edx in 10 times
#subbdivision
edx_1<-edx[1:1000000,]
#matrix value genre
a_1<-as.matrix(sapply(genres_edx, function(x) {
  ifelse(str_detect(edx_1$genres, x)=='TRUE',1,0)}))
#sum of  each rating with genre b_k values
k_1<-a_1%*%m_kvalues
rm(edx_1,a_1)
#subbdivision
edx_2<-edx[1000001:2000000,]
#This is performed 10 times
```

Then, each new subdivision was appended to create again the original training data plus the new sum genre effect column, which is used in the Cross-validation section.

```
# append k values
k<-rbind(k_1,k_2,k_3,k_4,k_5,k_6,k_7,k_8,k_9,k_10)
rm(k_1,k_2,k_3,k_4,k_5,k_6,k_7,k_8,k_9,k_10)
#add k value to edx
edx$b_k<-k
rm(k)
```

2.5 Testing the algorithm

Finally, the algorithm was tested using the typical error loss expressed as the residual mean squared error (RMSE) (Equation (6)).

$$\text{RMSE} = \sqrt{\frac{1}{N}\sum_{u,i}\left(\hat{y}_{u,i} - y_{u,i}\right)^2} \tag{6}$$

It means the typical error that is made when it is predicted a rating. The importance of this formula is to observe the differences between the rating $y_{u,i}$ with the predicted ratings $\hat{y}_{u,i}$. Low values mean a better prediction of the model. Values over 1, for example, means that the typical error is larger than 1 score, so it is necessary to get values below 1.

2.6 Penalizations

Notwithstanding, before testing the algorithm it was decided to apply penalties over average ranking for movie $\widehat{b}_i$ and for the user-specific effect $\widehat{b}_u$. The idea is based that the *bias* and the *user-specific effect* are affected by the number of observations associated to them. So, at larger observations the penalization is lower. For this step, it was not considered to be applied on $\widehat{b}_k$, since the hardware processing was would be much more than it could cover the personal computer, and as the observation number for each genre is much larger than user and movies (except for "no genres listed movies"), a penalization is nonsense.

In this sense, to constraint the variability of the effect sizes, it was considered that for $\widehat{b}_i$ and $\widehat{b}_u$ values having few observations, it was better to be assigned values close or even 0, instead of the provided value. The penalization is represented by λ, which is a tuning parameter, being practically ignored when there were large observations. At a larger λ, shrinks more.

Thus, the minimization is represented in the following way (Equation (7)):

$$\frac{1}{N}\sum_{u,i}\left(y_{u,i}-\mu-b_i-b_u\right)^2+\lambda\left(\sum_i b_i^2+\sum_u b_u^2\right) \tag{7}$$

where to calculate the penalization to b_i (Equation (8)), and b_u (Equation (9)) are expressed like this:

$$\hat{b}_i(\lambda)=\frac{1}{\lambda+n_i}\sum_{u=1}^{n_i}\left(Y_{u,i}-\hat{\mu}\right) \tag{8}$$

$$\hat{b}_u(\lambda)=\frac{1}{\lambda+n_u}\sum_{u=1}^{n_i}\left(Y_{u,i}-\hat{\mu}-\hat{b}_i\right) \tag{9}$$

One important point is that the definition of the λ value had to be calculated. For this purpose it was selected the cross-validation method, which **must be applied on the training set**.

2.6.1 Cross-validation

The basic idea of cross-validation is to create an algorithm or to estimate a parameter, in this case λ, and to test it in the training data. Therefore, to be consistent with the machine learning methodology, the training data must be split between **training data** and **validate set data**. Then, to obtain more convinced results, this procedure has to occure several times over the original training data, creating the known *K-folds*.

On the other hand, **the test data, here validation dataset, it is not used at all**.

One important point is to apply the parameter equally across all the *K-folds*, before starting with a new parameter value. So for this project, it was tested the algorithm calculating RMSE, but with several provided parameter values.
This is expressed with this formula (Equation (10)):

$$\text{RMSE}(\lambda)=\frac{1}{K}\sum_{b=1}^{K}\text{RMSE}_b(\lambda) \tag{10}$$

In this sense, it was defined to create 3 *K-folds*, due to hardware processing, selecting the parameter λ that minimizes the k fold RMSE (Equation (11)):

$$Min\text{RMSE}(\lambda)=\frac{1}{3}\sum_{b=1}^{3}\text{RMSE}_b(\lambda) \tag{11}$$

2.7 Testing calibrated algorithm

Finally, after obtaining the optimal λ, it was added to the algorithm and reapplied the test, but instead of using the training data as in Cross-validation section, it was used the test data (**validation dataset**).

3 Results

This section is organized presenting the lambda value selection and the values of the typical error loss. The calculation of the $\sum_{k=1}^{K} x_{u,i}^{k} \hat{b}_k$ was already done in the Genre effect $\hat{b}_k$ and Sum section.

3.1 Choosing Lambda

First, it was created the *K-folds* with their corresponding training and validate set. The process was meticulously made, assuring to define properly the training and test set from the training set *edx*.

```
set.seed(1, sample.kind="Rounding") # if using R 3.5 or earlier, use `set.seed(1)`
# to create 3 training partition to identify the appropriate lambda value
edx_test_index <- createDataPartition(y = edx$rating, times = 3, p = 0.1, list = FALSE)
k<-3
#partition of the training dataset (edx) to apply cross-validation
edx_tr_list<-list()
edx_temp_list<-list()
edx_test_list<-list()
removed<-list()
for (i in 1:k){
edx_tr_list[[i]]<- edx[-edx_test_index[,i],]
edx_temp_list[[i]]<- edx[edx_test_index[,i],]
#Make sure userId and movieId in edx validation set are also in edx training set
edx_test_list[[i]] <- as.data.frame(edx_temp_list[i]) %>%
  semi_join(edx_tr_list[[i]], by = "movieId") %>%
  semi_join(edx_tr_list[[i]], by = "userId")
#Add rows removed from edx validation set back into edx training set
removed[[i]] <- anti_join(as.data.frame(edx_temp_list[i]),as.data.frame(edx_test_list[[i]]))
edx_tr_list[[i]] <- rbind(edx_tr_list[[i]], removed[[i]])
}
#remove temporal unnecessary data
rm(edx_temp_list,removed)
```

Secondly, it was applied the cross validation formula. To simplify the calculations it was tested 21 λ values, from 0 to 20. It is remarked that the RMSE included the $\sum_{k=1}^{K} x_{u,i}^{k} \hat{b}_k$ variable.

```
#Procedure is to test  possible lambda values,
#then it is created 3x11 possible RMSE values, using loop, sapply
#and matrix formulas.

lambdas <- seq(0, 10, 0.5)# 20 possible lambda values
rmses<-list()
for (j in 1:3) {
rmses[[j]] <- sapply(lambdas, function(l){
```

```
  mu <- mean(edx_tr_list[[j]]$rating)

  b_i <- edx_tr_list[[j]] %>%
    group_by(movieId) %>%
    summarize(b_i = sum(rating - mu)/(n()+l))

  b_u <- edx_tr_list[[j]] %>%
    left_join(b_i, by="movieId") %>%
    group_by(userId) %>%
    summarize(b_u = sum(rating - b_i - mu)/(n()+l))

  predicted_ratings <-
    edx_test_list[[j]] %>%
    left_join(b_i, by = "movieId") %>%
    left_join(b_u, by = "userId") %>%
    mutate(pred = mu + b_i + b_u + b_k) %>%# b_k it was
    #calculated as a mean value in the 2.4.1 section
    pull(pred)

  return(RMSE(predicted_ratings, edx_test_list[[j]]$rating))
})}
```

In this case, the procedure took a couple of minutes since it was creating at least 63 operations (3 *K-folds* x 21 λ values).

The next step was to get the overall RMSE according to each λ value, and then to select the optimal λ.

```
#RMSE calculation from the cross-validation step
RMSE_t<-rowMeans(matrix(cbind(rmses[[1]],rmses[[2]],rmses[[3]]),length(lambdas),3))
```

The next graph present the λ results (Figure 6):

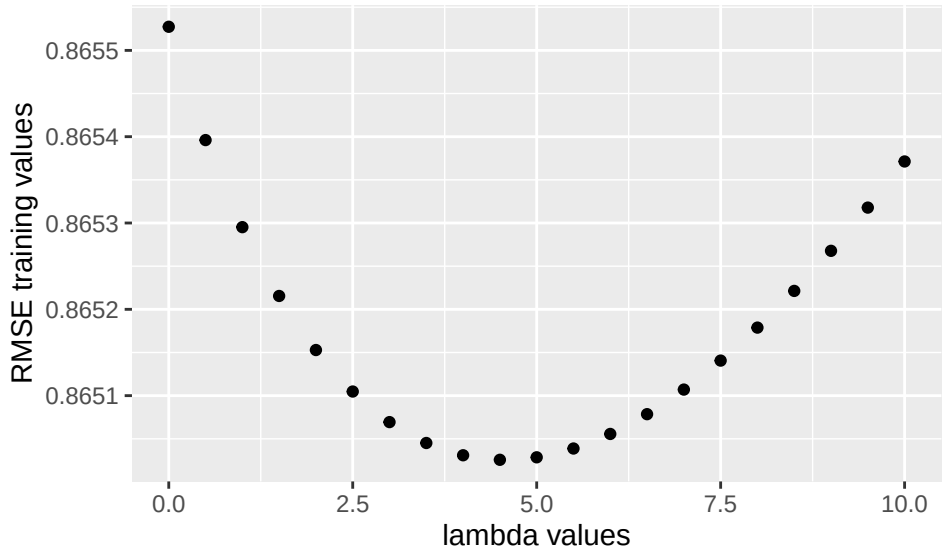

Figure 6: RMSE training values per lambda

As it can be seen, the optimal λ from the cross validation is:

```
lambda <- lambdas[which.min(RMSE_t)]
lambda# optimal lambda
```

```
## [1] 4.5
```

So the next step was to prove this lambda in the algorithm and to test it with the validation dataset with RMSE.

3.2 RMSE

The algorithm calculation is the following:

```
##Final calculation
mu <- mean(edx$rating) # average of all ratings
b_i <- edx %>% #average ranking for movie i
  group_by(movieId) %>%
  summarize(b_i = sum(rating - mu)/(n()+lambda))

b_u <- edx %>% #user-specific effect u
  left_join(b_i, by="movieId") %>%
  group_by(userId) %>%
  summarize(b_u = sum(rating - b_i - mu)/(n()+lambda))

#b_k
m_kvalues<-matrix(unlist(b_k.list))#b_k values from training data
#matrix value genre
val_k<-as.matrix(sapply(genres_edx, function(x) {
  ifelse(str_detect(validation$genres, x)=='TRUE',1,0)}))
#sum of  each rating  with genre b_k values
val_k<-val_k%*%m_kvalues
#add k value to edx
validation$b_k<-val_k
rm(val_k)

predicted_ratings <- #Prediction
  validation %>%
  left_join(b_i, by = "movieId") %>%
  left_join(b_u, by = "userId") %>%
  mutate(pred = mu + b_i + b_u+b_k) %>%# b_k it was calculated
  #as a mean value in the previous step
  pull(pred)
```

As it can be observed, it has 5 calculations, the first 4 are the variables $\hat{\mu}, \hat{b}_i, \hat{b}_u$ and $\sum_{k=1}^{K} x_{u,i}^{k} \hat{b}_k$, the last one is the prediction $\hat{y}_{u,i}$. Likewise, it was added the lambda value.

Finally, the last step was to test the algorithm.

```
RMSE(predicted_ratings, validation$rating) #Final outcome
```

```
## [1] 0.8647733
```

An acceptable prediction, since is below 1, being an acceptable result.

4 Conclusions

Summarizing, the obtained result satisfies the expectations of this project. It was applied a recommendation system to predict user rating of different movies from the MovieLens 10M dataset. The approach used 4 variables (average of all rankings, ranking of a movie and user-specific effect, and genre sum effect). Furthermore, it was added penalizations to the last variables as well, to incorporate specific characteristics of the movie streaming field. In the end, the result is trustworthy, having a loss of less than **0.8647733**, fulfilling the requirements for the first Capstone project.

In term of the difficulties experienced during the project elaboration, the memory RAM capacity can determine the performance and methodology of a machine learning approach. Indeed, this could be crucial for any project and for the present one it was a huge handicap. As a recommendation, if one wants to dedicate in the world of data science is mandatory to invest in a desktop computer with high memory disc and RAM capacity.

Finally, even though this project is based in a consistent machine learning approach (recommendation system), there are still data patterns that could be integrated to the algorithm. For example, it seems that ratings could be related to the presence of relevant actor/actresses, or by being a blockbuster. Considering them undoubtedly will help to improve the algorithm, being extremely useful to predict ratings in the streaming company or even for similar activities.

References

de Oliveira, D. F. (2021). Recommendations systems: Rating predictions with movielens data. https://github.com/dfedeoli/recommendation-system/.

F. M. Harper, J. A. K. (2015). Movielens 10m dataset readme. https://files.grouplens.org/datasets/movielens/ml-10m-README.html.

F. O. Isinkaye, Y. O. Folajimi, B. O. (2015). Recommendation systems: Principles, methods and evaluation. *Egyptian Informatics Journal*.

Irizarry, R. (2021). Introduction to data science: Data analysis and prediction algorithms with R. https://rafalab.github.io/dsbook/.