MACHINE LEARNING CASE- MOVIE RECOMENDATION

MOHAMED BOUNIT 2024-04-04

RMD, R and pdf files at : https://github.com/MOBOUN/BOUNGIT

SUMMARY

1. INTRODUCTION: OBJECTIVE, CONTEXT AND SCOPE

The aim of current project is to create a movie recommendation system using the 10M version of the MovieLens; we'll be using some tools we have learned throughout the courses, especially R coding, Data exploration and visualization and machine learning algorithms.

We will use the 10M version of the MovieLens dataset to make the computation feasible and easy to evaluate; the entire latest MovieLens dataset is larger and kept up to date on grouplens.org website.

The process to achieve this project can be summarized in the following steps:

- Extraction Data
- Pre-processing
- Data Exploration Solution
- Modeling and coding
- Results
- Conclusion

In addition to courses I learned on Edx platform, other internet sources were very helpful for me understanding each of the steps and I pointed the most used machine learning process and algorithms, and to implement blocs of code in R as well. The most used resources are pointed out in the references section of pdf report. My thoughts beyond the academic aspect of this project are open to discussion on section 9 of pdf report. The scope been defined clearly by the "capstone" instructions: finding the best model that minimizes RMSE, regularization included been one of the alternatives.

The Output to predict, rating, is categorical with 10 possible levels, thus we will test linear regression based algorithms and find out the most accurate.

2. DATA PREPROCESSING

Extraction: Code provided by the EDX staff to download an create MovieLens dataset.

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

library(tidyverse)
library(caret)
library(dplyr)
library(anytime)
```

```
library(lubridate)
library(ggplot2)
```

```
## Rows: 10,000,054
## Columns: 6
## $ userId    <int> 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, …
## $ movieId   <chr> "122", "185", "231", "292", "316", "329", "355", "356", "362…
## $ rating    <dbl> 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, …
## $ timestamp <int> 838985046, 838983525, 838983392, 838983421, 838983392, 83898…
## $ title     <chr> "Boomerang (1992)", "Net, The (1995)", "Dumb & Dumber (1994)…
## $ genres    <chr> "Comedy|Romance", "Action|Crime|Thriller", "Comedy", "Action…

####looking for incomplete entries (missing data)
sum(is.na(movies))## [1] 0

sum(is.na(ratings)) ## [1] 0

sum(is.na(movielens)) ## [1] 0

### "final_holdout_test" set, for validation, will be 10% of Movielens data, "edx" will
be used for the training and test

set.seed(1, sample.kind = "Rounding") #  actual version R 4.2.3

test_index <- caret::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 final_holdout_test set are in "edx" set

final_holdout_test <- temp %>%
    semi_join(edx, by = "movieId") %>%
    semi_join(edx, by = "userId")

  # Add rows removed from final_holdout_test back to edx set

removed <- anti_join(temp, final_holdout_test)
edx <- rbind(edx, removed)

#Removing files no more needed for the rest of the project
#rm(dl, test_index, temp, removed)

# filees not removed yet to allow code testing without downloading/unzip
#rm(movielens,ratings,movies)

###looking for incomplete entries (missing data)
sum(is.na(final_holdout_test))  ## [1] 0 sum(is.na(edx)) ## [1] 0
```

Training and validation sets Overview

```
glimpse(edx)

## Rows: 9,000,055
## Columns: 6
## $ userId    <int> 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 2, …
## $ movieId   <chr> "122", "185", "292", "316", "329", "355", "356", "362", "364…
## $ rating    <dbl> 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, …
## $ timestamp <int> 838985046, 838983525, 838983421, 838983392, 838983392, 83898…
## $ title     <chr> "Boomerang (1992)", "Net, The (1995)", "Outbreak (1995)", "S…
## $ genres    <chr> "Comedy|Romance", "Action|Crime|Thriller", "Action|Drama|Sci…

glimpse(final_holdout_test)

## Rows: 999,999
## Columns: 6
## $ userId    <int> 1, 1, 1, 2, 2, 2, 3, 3, 4, 4, 4, 5, 5, 5, 5, 5, 5, 5, 5, …
## $ movieId   <chr> "231", "480", "586", "151", "858", "1544", "590", "4995", "3…
## $ rating    <dbl> 5.0, 5.0, 5.0, 3.0, 2.0, 3.0, 3.5, 4.5, 5.0, 3.0, 3.0, 3.0, …
## $ timestamp <int> 838983392, 838983653, 838984068, 868246450, 868245645, 86824…
## $ title     <chr> "Dumb & Dumber (1994)", "Jurassic Park (1993)", "Home Alone …
## $ genres    <chr> "Comedy", "Action|Adventure|Sci-Fi|Thriller", "Children|Come…

##  Class :character   Class :character
##  Mode  :character   Mode  :character
```

No missing data found in any of the input and output files of the training and validation sets sum(is.na(movies)) = 0 sum(is.na(ratings))= 0 sum(is.na(movielens))= 0 sum(is.na(final_holdout_test))= 0 sum(is.na(edx))=0

Original "movielens" contains Rows:

10,000,054 Columns: 6 - userId 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1… - movieId 122, 185, 231, 292, 316, 329, 355, 356, 362, 3… - rating 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5… - timestamp 838985046, 838983525, 838983392, 838983421, 83… - title "Boomerang (1992)", "Net, The (1995)", "Dumb &… - genres "Comedy|Romance", "Action|Crime|Thriller", "Co…

The edx set contains

9000055 ratings provides by 69878 unique users for 10677 unique evaluated movies

Rows: 9,000,055 Columns: 6 observations

$ userId 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 2, 2, 2, 2, 2, 2, 2, 2,…

$movieId 122, 185, 292, 316, 329, 355, 356, 362, 364, 370, 377, 420, 466, 520, 539, 588, …

$ rating 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 3, 3, 5, 2, 3, 3,…

$ timestamp 838985046, 838983525, 838983421, 838983392, 838983392, 838984474, 838983653, 838…

$ title “Boomerang (1992)”, “Net, The (1995)”, “Outbreak (1995)”, “Stargate (1994)”, “St…

$ genres ”Comedy|Romance”, “Action|Crime|Thriller”, “Action|Drama|Sci-Fi|Thriller”, “Acti…

There are 69878 unique users given ratings to 10677 different films. the unique genres were counted `Drama` and `Comedy|Drama` are counted as 2 different genres. Same remark for `Comedy` and `Comedy|Drama` where we count `Comedy` 2 times etc ; “rating”, “genres” and title are in the character format The `userId` and `movieId` variables are `Integer` format in the original data set. These are actually labels and are converted to `factor` type in the following section.

Rating given by a user to a movie values From 0 to 5 stars stepping of 0.5.type is `numeric` Having in mind the fact that a time effect could be ovserved on rating, or should we look at this potential effect, we need to change “timestamp” format to be useful, I also loaded lubridate and anytime libraries for this aim.The `timestamp` variable is converted to `POSIXct` type, to be handle correctly as a `date` vector. The year is extracted to the `year` column..

The final_holdout_test set contains:

Rows: 999,999 records Columns: 6 observations

```
edx$userId <- as.factor(edx$userId) # change `userId` to `factor`.
edx$movieId <- as.factor(edx$movieId) # change `movieId` to `factor`.
edx$genres <- as.factor(edx$genres) # change `genres` to `factor`.
edx$timestamp <- as.POSIXct(edx$timestamp, origin = "1970-01-01") # change `timestamp to
`POSIXct`.
edx <- edx %>% mutate(year_rate = year(timestamp)) #  add year that the rate from
timestamp.
glimpse(edx)

## Rows: 9,000,055
## Columns: 7
## $ userId    <fct> 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 2, …
## $ movieId   <fct> 122, 185, 292, 316, 329, 355, 356, 362, 364, 370, 377, 420, …
## $ rating    <dbl> 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, …
## $ timestamp <dttm> 1996-08-02 11:24:06, 1996-08-02 10:58:45, 1996-08-02 10:57:…
## $ title     <chr> "Boomerang (1992)", "Net, The (1995)", "Outbreak (1995)", "S…
## $ genres    <fct> Comedy|Romance, Action|Crime|Thriller, Action|Drama|Sci-Fi|T…
## $ year_rate <dbl> 1996, 1996, 1996, 1996, 1996, 1996, 1996, 1996, 1996, 1996, …
```

3. DATA EXPLORATION

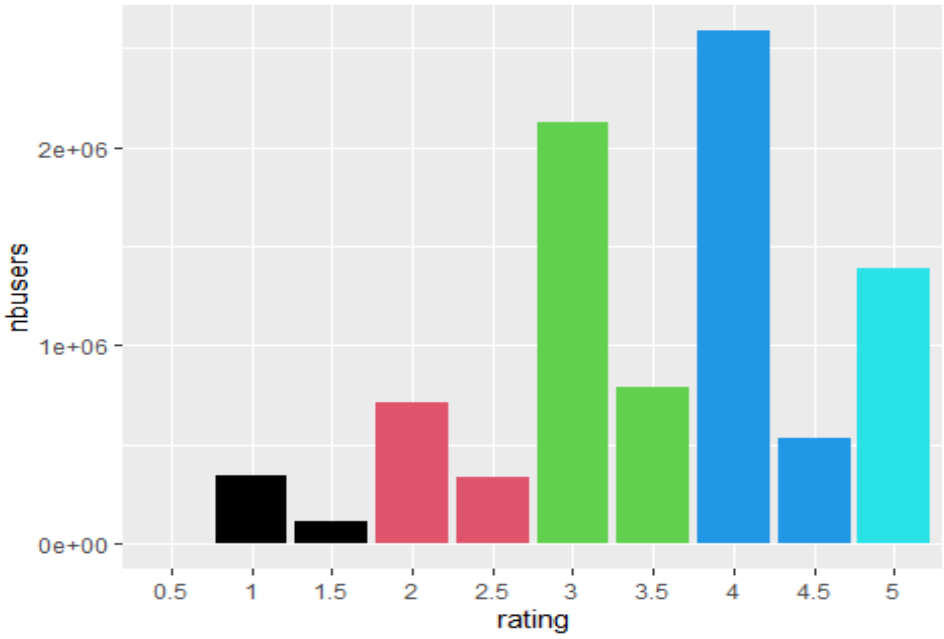
```
#the most rated title
Firsttitle <- edx%>% select(title,userId)%>%
  group_by(title)%>%summarise(nbuser=n())%>%
  arrange(desc(nbuser))
head(Firsttitle)
```

```
## # A tibble: 6 × 2
##   title                              nbuser
##   <chr>                               <int>
## 1 Pulp Fiction (1994)                 31362
## 2 Forrest Gump (1994)                 31079
## 3 Silence of the Lambs, The (1991)    30382
## 4 Jurassic Park (1993)                29360
## 5 Shawshank Redemption, The (1994)    28015
## 6 Braveheart (1995)                   26212
```

```
# Rating top down
distrib <- edx %>% select(rating,userId)%>%
  group_by(rating)%>%summarise(nbuser=n())%>%
  select(rating,nbuser)%>%
  arrange(desc(nbuser))
head(distrib,10)
```

```
## # A tibble: 10 × 2
##    rating  nbuser
##     <dbl>   <int>
##  1    4   2588430
##  2    3   2121240
##  3    5   1390114
##  4    3.5  791624
##  5    2    711422
##  6    4.5  526736
##  7    1    345679
##  8    2.5  333010
##  9    1.5  106426
## 10    0.5   85374
```

```
# Rating distribution using geom_bar

rat <- as.data.frame <- edx %>% select(rating,userId)%>%
  group_by(rating)%>%summarise(nbusers=n())%>%
  select(rating,nbusers) %>% mutate(rating=as.character(rating))
ggplot(rat,aes(x=rating,y=nbusers))+
geom_bar(stat='Identity',fill=rat$rating)
```



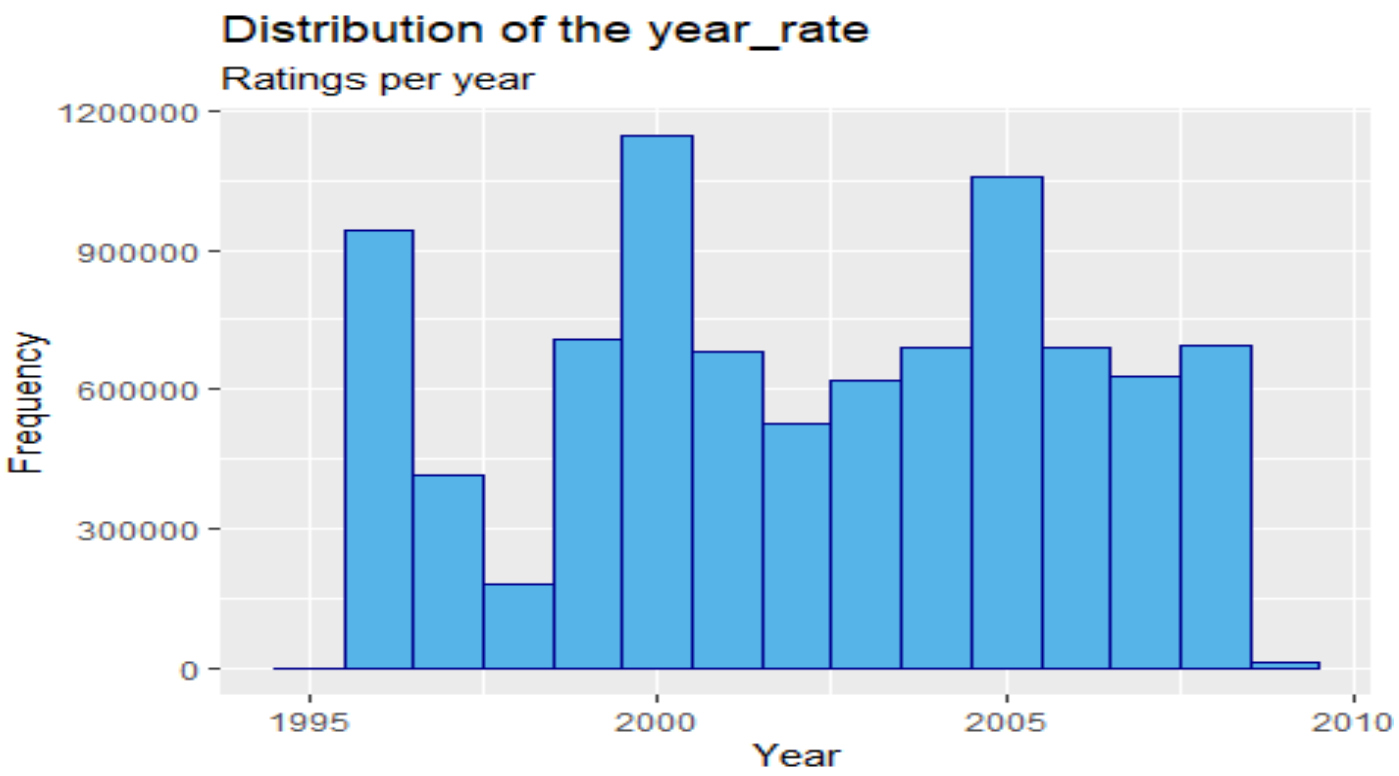
The most given rates are respectively: 4/3/5/3.5/2; More generally We notice left-skewed shape of rating distribution. there are more *likes* ratings than *bad ratings* ratings. Neuro-scientists or even psychology could explain better this behavior, furthermore find normal this shape, as people tend to recommend their own preferences to their friends and families. We can also assume that ranking a movie, after watching it entirely, means we took time for this, we were interested and even liked before the end. In the other hand, disliking a film could happen at some point while watching. The normal behavior when we don't like,is the to quit, without having a complete idea to evaluate, or the time and will to spend on commenting and evaluations.

People who have rated small values less than 1.5 could be interested on some aspects of a movie at the beggining(director or actors reputations, genre for the film or social media influenced, professionals of the movie industry, journalists, and everyone disappointed regarding his initial expectations. We can make more assumptions on the impact of the scale itself, as soon as there's clearly more rates with entire numbers than half rates, could we imagine the distribution withe INTEGERs from (1:5) or (1:10), behavioral or neurological science should have more insights

Analyzing the "year of release" effect on movie rating there's a record of 15 years from 1995 to 2009, These 2 limits of the range seem to be outliners, as far as few records were done, the ratings didn't include the hole year for 1995 AND 2009

```
# table(unique(edx$year_rate)) years check
edx %>% ggplot(aes(year_rate)) +
  geom_histogram(binwidth=1,color="darkblue",fill="#56B4E9") +
```

```
  labs(title = "Distribution of the year_rate",
       subtitle = "Ratings per year",
       x = "Year",
       y = "Frequency")
```



The shape observed from rating frequency to release year shows spikes on 1996 followed by decreasing numbers for 1997 and 1998, after that, at the 3rd year (after the spike), ratings start the rise. Same remark 2000 and 2005, It seems there's a five year cycle from spike to spike. 2 years after the left spike are decreasing, the growth starts from the 4th year. However, o confirm this hypothesis, more data is required; which is out of scope of this work. The rating seem to fluctuate less between 2 spikes

```
# Rating distribution top down
distyear <- edx %>% select(userId,year_rate)%>%
  group_by(year_rate)%>%summarise(yearcount=n())%>%
  arrange(desc(yearcount))
head(distyear,15)
```

```
## # A tibble: 15 × 2
##    year_rate yearcount
##        <dbl>     <int>
##  1      2000   1144349
##  2      2005   1059277
##  3      1996    942772
##  4      1999    709893
##  5      2008    696740
##  6      2004    691429
##  7      2006    689315
##  8      2001    683355
```

```
##  9      2007    629168
## 10      2003    619938
## 11      2002    524959
## 12      1997    414101
## 13      1998    181634
## 14      2009     13123
## 15      1995         2
```

```
sum(distyear$yearcount)
```

```
## [1] 9000055
```

Users versus rating average

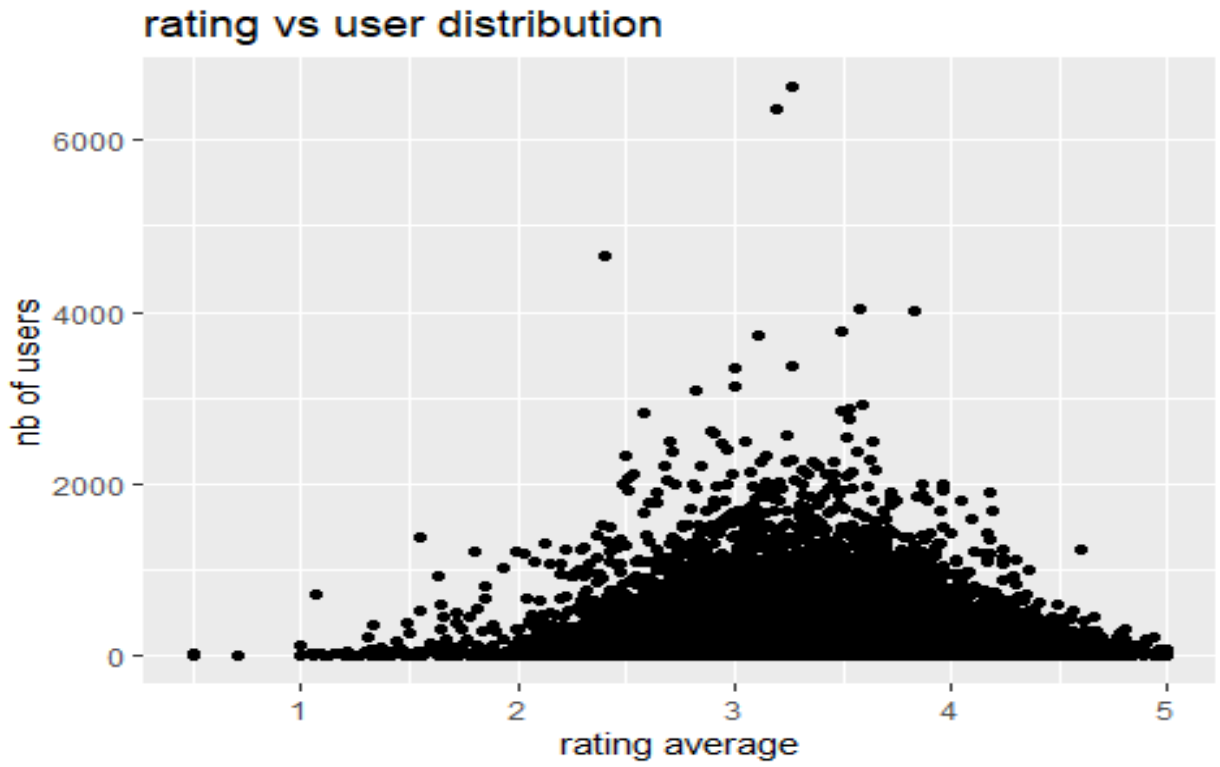
```
useAVG <- edx %>%mutate(userId=as.factor(userId))%>%
  group_by(userId) %>%
  summarize(Muser = mean(rating),nb_ratings=n())
#top down sorting by average rating
USR1 <- useAVG %>%  arrange(desc(Muser))
head(USR1,30)
```

```
## # A tibble: 30 × 3
##    userId Muser nb_ratings
##    <fct>  <dbl>      <int>
##  1 1          5         19
##  2 7984       5         17
##  3 11884      5         18
##  4 13027      5         29
##  5 13513      5         17
##  6 13524      5         20
##  7 15575      5         29
##  8 18965      5         49
##  9 22045      5         18
## 10 26308      5         17
## # … with 20 more rows
```

```
#top down sorting by average no ratings
USR2 <- useAVG %>%  arrange(desc(nb_ratings))
head(USR2,30)
```

```
## # A tibble: 30 × 3
##    userId Muser nb_ratings
##    <fct>  <dbl>      <int>
##  1 59269   3.26       6616
##  2 67385   3.20       6360
##  3 14463   2.40       4648
##  4 68259   3.58       4036
##  5 27468   3.83       4023
##  6 19635   3.50       3771
##  7 3817    3.11       3733
##  8 63134   3.27       3371
##  9 58357   3.00       3361
## 10 27584   3.00       3142
## # … with 20 more rows
```

```
qplot(useAVG$Muser, useAVG$nb_ratings,
      main = "rating vs user distribution",
      xlab = "rating average", ylab ="nb of users" )
```



The plot of average ratings to number of users shapes a flat normal distribution; right shifted to the rating range (2 to five 5 well centered); with small number of outlines that more than 6000 for an average of 3.3 stars with regards to the 2 tables above, users with regular 'high' rates are relatively rare: only one rated 5 for 19 times, this probably ensures this data has not been fabricated or scores manipulated by a robot. Secondly, the most frequently recorded users (4000 TO 7600 votes) give scores less than the global average of 3.5 stars.

Rating breakdown per genre

797 different genres were listed, with overlapping counts as one movie could have been assigned more than one specific genre (drama, comedy …), in fact most of the movies are hardly classified within a single genre (ie in one single word).

The figures show different conclusions depending whether we arrange the rating versus genre by the average (men(rating),or the total number of rating: "Animation|IMAX|Sci-Fi" for example is rated only 7 times with relatively high average "Action|Drama|Thriller|War" for example is rated only 480 times with relatively high average in the other hand the most rated genres are "Drama" and "Comedy", regardless of combined genres that may contain these 2 single classified genres.

In the other hand, the 733 296 ratings for Drama genre don't include all the occurrences of drama type when it's merged in combined category, such as "Comedy|Drama" and Comedy|Drama|Romance

In next section we'll see how does ratins differens with genres taken individually But first look at the most rated genres (actual, composed) and see nb of users that a rated a movie and the top down classification by average rating given to the genres(composed)

```
distgen <- edx %>%
  select(genres, rating) %>%
  group_by(genres) %>% summarize(avg = mean(rating), nbpg = n())

distgen%>% arrange(desc(avg)) %>% head(20)
```

```
## # A tibble: 20 × 3
##    genres                                              avg  nbpg
##    <fct>                                             <dbl> <int>
##  1 Animation|IMAX|Sci-Fi                              4.71     7
##  2 Drama|Film-Noir|Romance                            4.30  2989
##  3 Action|Crime|Drama|IMAX                            4.30  2353
##  4 Animation|Children|Comedy|Crime                     4.28  7167
##  5 Film-Noir|Mystery                                  4.24  5988
##  6 Crime|Film-Noir|Mystery                            4.22  4029
##  7 Film-Noir|Romance|Thriller                         4.22  2453
##  8 Crime|Film-Noir|Thriller                           4.21  4844
##  9 Crime|Mystery|Thriller                             4.20 26892
## 10 Action|Adventure|Comedy|Fantasy|Romance            4.20 14809
## 11 Crime|Thriller|War                                4.17  4595
## 12 Film-Noir|Mystery|Thriller                        4.16  4011
## 13 Adventure|Drama|Film-Noir|Sci-Fi|Thriller         4.15 13957
## 14 Adventure|Animation|Children|Comedy|Sci-Fi         4.15  3529
## 15 Adventure|Comedy|Romance|War                      4.13  5223
## 16 Adventure|Animation|Children|Comedy|Romance|Sci-Fi 4.12  1254
## 17 Comedy|Drama|Romance|Sci-Fi                       4.12  7593
## 18 Action|Crime|Drama|Film-Noir|Mystery              4.12  1103
## 19 Animation|Drama|War                               4.11  1039
## 20 Action|Drama|Thriller|War                         4.11   480
```

```
distgen%>% arrange(desc(nbpg)) %>% head(20)
```

```
## # A tibble: 20 × 3
##    genres                            avg   nbpg
##    <fct>                           <dbl>  <int>
##  1 Drama                            3.71 733296
##  2 Comedy                           3.24 700889
##  3 Comedy|Romance                   3.41 365468
##  4 Comedy|Drama                     3.60 323637
##  5 Comedy|Drama|Romance             3.65 261425
##  6 Drama|Romance                    3.61 259355
##  7 Action|Adventure|Sci-Fi          3.51 219938
##  8 Action|Adventure|Thriller        3.43 149091
##  9 Drama|Thriller                   3.45 145373
## 10 Crime|Drama                      3.95 137387
## 11 Drama|War                        3.98 111029
## 12 Crime|Drama|Thriller             3.78 106101
## 13 Action|Adventure|Sci-Fi|Thriller 3.54 105144
```

```
## 14 Action|Crime|Thriller              3.46 102259
## 15 Action|Drama|War                   3.92  99183
## 16 Action|Thriller                    3.27  96535
## 17 Action|Sci-Fi|Thriller             3.42  95280
## 18 Thriller                           3.53  94662
## 19 Horror|Thriller                    3.12  75000
## 20 Comedy|Crime                       3.53  73286
```

Count genres appearances with all occurrences of single word “Drama” “Comedy# ….etc.

We use ”gsub” and “strsplit” and word to extract individual genres for a movie, from the column genres actually given by Movielens, we look for every occurrence of the word “Drama” through genres column and do the same for other categories, then we put the data GENSINGLE Summarized by average rating and totals number of rating for each single genre. Notice the total ratings number of these 4 items is 11 489 056 is larger than the total count of ratings, because in this case, the same rating is counted more than one time, as much as the number of single categories of the rated movie.

```
#splitting "genres " column in to 4 separate columns containing single genres or NA

GENSINGLE <- distgen %>% mutate(
GEN0=gsub("[^[:alnum:]]","",word(strsplit(as.character(genres),"[|,]+",fixed=FALSE),start
=1,end=1)),
GEN2=gsub("[^[:alnum:]]","",word(strsplit(as.character(genres),"[|,]+",fixed=FALSE),start
=2,end=2)),
GEN3=gsub("[^[:alnum:]]","",word(strsplit(as.character(genres),"[|,]+",fixed=FALSE),start
=3,end=3)),
GEN4=gsub("[^[:alnum:]]","",word(strsplit(as.character(genres),"[|,]+",fixed=FALSE),start
=4,end=4)))
#GENSINGLE$GEN0

#Removing "c" fom the genres first column splitted
GENSINGLE <- GENSINGLE %>% mutate(GEN1 =
if_else(substr(GEN0,1,1)=="c",sub("c","",GEN0),GEN0))

#### TABLE GENRE, Nb RATINGS, and AVERAGE RATING
#pivot long to have single genre values in one column
SING <- GENSINGLE %>% pivot_longer (
    cols = c("GEN1", "GEN2","GEN3","GEN4"),
    values_to = "GENSING" ) %>% drop_na()%>%
    filter(nbpg>100)%>%
    group_by(GENSING) %>%
    summarize(AVG=mean(avg),totrates=sum(nbpg))
SING1<-SING %>% arrange(desc(totrates))
SING2 <-SING %>% arrange(desc(AVG))
head(SING1,20)

## # A tibble: 19 × 3
##    GENSING       AVG totrates
##    <chr>       <dbl>    <int>
##  1 Drama        3.63  3893965
##  2 Comedy       3.37  3530753
##  3 Action       3.31  2558792
```

```
##  4 Thriller     3.46  2115913
##  5 Adventure    3.39  1907011
##  6 Romance      3.50  1622005
##  7 Crime        3.55  1315542
##  8 SciFi        3.33  1284328
##  9 Fantasy      3.36   834280
## 10 Children     3.23   737391
## 11 Horror       3.24   682720
## 12 Mystery      3.55   545433
## 13 War          3.58   499329
## 14 Animation    3.48   466676
## 15 Musical      3.44   366175
## 16 Western      3.56   182664
## 17 FilmNoir     3.93   118337
## 18 Documentary  3.51    92462
## 19 IMAX         3.79     7386
```

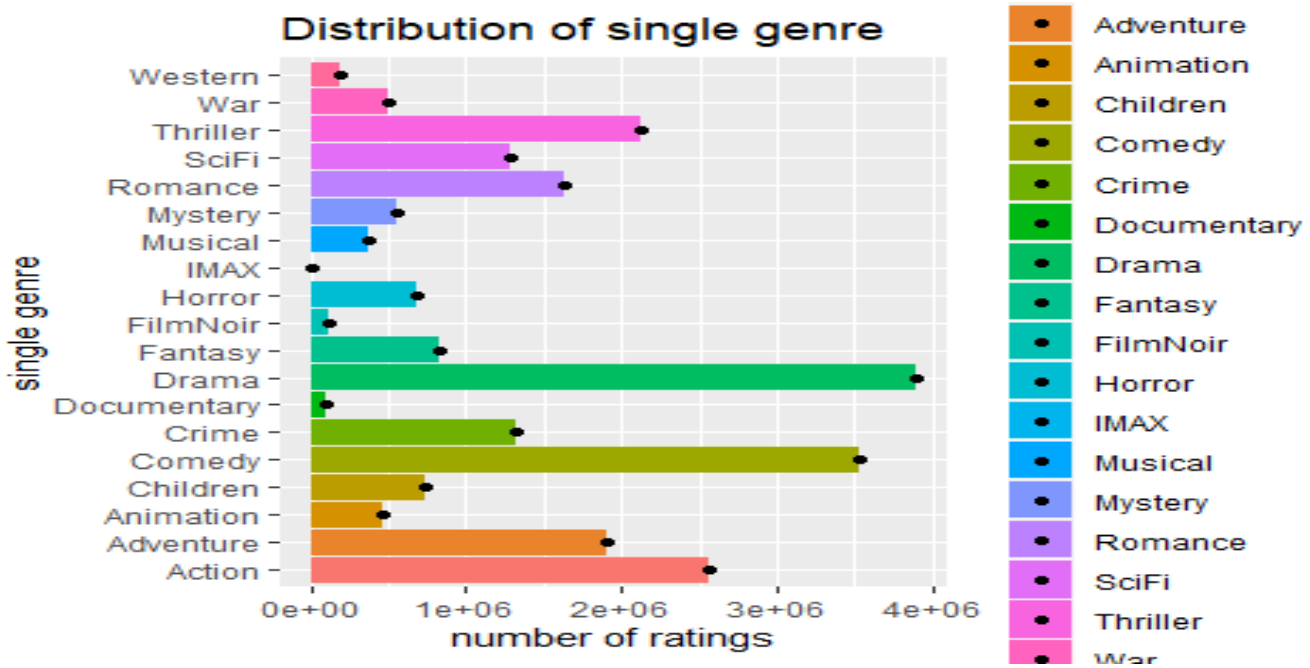
```
head(SING2,20)
```

```
## # A tibble: 19 × 3
##    GENSING       AVG totrates
##    <chr>       <dbl>    <int>
##  1 FilmNoir     3.93   118337
##  2 IMAX         3.79     7386
##  3 Drama        3.63  3893965
##  4 War          3.58   499329
##  5 Western      3.56   182664
##  6 Crime        3.55  1315542
##  7 Mystery      3.55   545433
##  8 Documentary  3.51    92462
##  9 Romance      3.50  1622005
## 10 Animation    3.48   466676
## 11 Thriller     3.46  2115913
## 12 Musical      3.44   366175
## 13 Adventure    3.39  1907011
## 14 Comedy       3.37  3530753
## 15 Fantasy      3.36   834280
## 16 SciFi        3.33  1284328
## 17 Action       3.31  2558792
## 18 Horror       3.24   682720
## 19 Children     3.23   737391
```

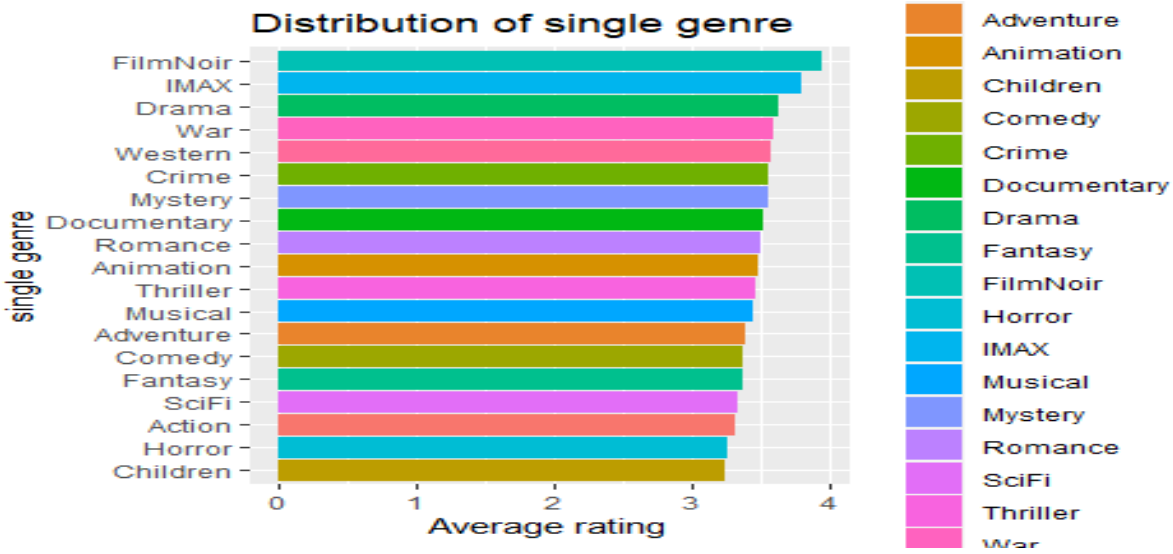
```
#### chart GENRE VS Nb RATINGS
```

Comparing rating distribution versus single genres above, we have 2 tables for top down sorting of single genres, once by average and the 2nd by ratings number now we have 19 “single genres”, with each movie belonging to one or a combination or 1 to 4 items, the rating given to this movie is counted the same for every genres it was belonged to drama, comedy, action and thriller, have the first 4 users respectively, with more than 2 000 000 votes In the other hand, Film noir, Imax and drama are the top 3 by average rating, but the drama genre is relatively significant (stable for prediction) as the number of ratings is much bigger.as for the 4 categories seen above.

```
# nb ratings vs single genres
SING %>% ggplot(aes(totrates, GENSING, fill= GENSING)) +
  geom_col()+
  labs(title = "Distribution of single genre",
       x = "number of ratings",
       y = "single genre") +
geom_point()
```




```
# average vs single genres
SING %>% ggplot(aes(AVG,y=reorder(GENSING,AVG),fill=GENSING)) +
  geom_col()+
labs(title = "Distribution of single genre",
       x = "Average rating",
       y = "single genre")
```



4. MODELING

The aim is to find out a learning model with the minimum error for prediction; we'll try to get RMSE less than 0.86490 that give right to 25 POINTS evaluation and seem to be a good achievement from previous submitted projects. Both training and validation sets are created above with the provided code from EDX staff. Rows: 999,999 records and Columns: 6 observations "final_holdout_test" will be kept unchanged for final estimation of RMSE only (Validation), with the last model chosen with the best RMSE on "test" set, The edx set contains initially 9000055 ratings provides by 69878 unique users for 10677 unique evaluated movies, will be split in 2 subsets: "training" and "test"

```
edx <- edx %>% select(userId, movieId, rating)
test_index <- createDataPartition(edx$rating, times = 1, p = .2, list = F)# Create the
index
training <- edx[-test_index, ] # Create Train set
test <- edx[test_index, ] # Create Test set
test <- test %>% # The same movieId and usersId appears in both set. (Not the same cases)
    semi_join(training, by = "movieId") %>%
    semi_join(training, by = "userId")

#glimpse(edx)
#glimpse(test)
```

Subset "test" has now 1 799 967 rows, "training" has 7 200 043 obs and edx 9 000 055 rows I created RMSEs dataframe used to collect gradually all the values for tested models with meodel name and RMSE value obtained

Baseline model

The simplest model is to use the most common rating from the *training* set to be predicted. This is the **baseline MODEL** model.

```
mu <- mean(training$rating) # Mean accross all movies.
model0_RMSE <- RMSE(test$rating, mu) # RMSE in test set.
model0_RMSE

## [1] 1.060704
```

The RMSE baseline model 1.060289 is greater than 1 and thus show the baseline model is weak for making prediction, more than 1 star error; we looked for better models in the following section. RMSEs for different models are collected in rmse_table we created with the code bellow:

```
rmse_table <- data_frame(Method = "Baseline MODEL", RMSE = model0_RMSE)
rmse_table %>% knitr::kable(caption = "RMSEs")
```

RMSEs

Method	RMSE
Baseline MODEL	1.060704

Movie effect Model

```
Mu <- mean(training$rating)

movieAVG <- training %>%mutate(movieId=as.factor(movieId))%>%
  group_by(movieId) %>%
  summarize(Mi = mean(rating - Mu))
PREDICTIONS <- test %>%
  left_join(movieAVG, by = "movieId") %>%
  mutate(pred = Mu + Mi) %>% .$pred
model1_RMSE <- RMSE(PREDICTIONS,test$rating)
model1_RMSE

## [1] 0.9437144

rmse_table <- rbind(rmse_table, data_frame(Method = "Movie Effect", RMSE = model1_RMSE))
rmse_table %>% knitr::kable(caption = "RMSEs")
```

RMSEs

Method	RMSE
Baseline MODEL	1.0607045
Movie Effect	0.9437144

User effect Model

```
Mu <- mean(training$rating)

userEFFAVG <- training %>%mutate(userId=as.factor(userId))%>%
  group_by(userId) %>%
  summarize(Mi = mean(rating - Mu))
#head(userEFFAVG)
PREDICTIONS <- test %>%
  left_join(userEFFAVG, by = "userId") %>%
  mutate(pred = Mu + Mi) %>% .$pred

model2_RMSE <- RMSE(PREDICTIONS,test$rating)
model2_RMSE

## [1] 0.9794339

rmse_table <- rbind(rmse_table, data_frame(Method = "User Effect", RMSE = model2_RMSE))
rmse_table %>% knitr::kable(caption = "RMSEs")
```

RMSEs

Method	RMSE
Baseline MODEL	1.0607045
Movie Effect	0.9437144
User Effect	0.9794339

User and Movie effect Model

User effect model and movie effect model both have still a big RMSE and weeak prediction power, whereas the error is bigger when using user effect alone.

The next step is going to try to get a new model with a better RMSE. using both user and movie effect. We take the user effect *Ui* and the movie effect *Mi* as predictors. Therefore, we are generating the next model to predict `rating` *y_hat_i*: *y_hat_i=Ui+Mi+epsilon*

```
Mu <- mean(training$rating)
movieAVG <- training %>%mutate(movieId=as.factor(movieId))%>%
  group_by(movieId) %>%
  summarize(Mi = mean(rating - Mu))
userAVG <- training %>% mutate(movieId=as.factor(movieId))%>%
  left_join(movieAVG, by = "movieId") %>%
  group_by(userId) %>%
  summarize(Ui = mean(rating - Mu - Mi))

#glimpse(movieAVG)
#glimpse(userAVG)
#glimpse(test)
PREDICTIONS <- test %>%
  left_join(movieAVG, by = "movieId") %>%
  left_join(userAVG, by = "userId") %>%
  mutate(pred = Mu + Mi + Ui) %>% .$pred

modelMU_RMSE <- RMSE(PREDICTIONS,test$rating)
modelMU_RMSE

## [1] 0.8661625

rmse_table <- rbind(rmse_table,data_frame(Method = "User & Movie Effect", RMSE =
modelMU_RMSE))
rmse_table %>% knitr::kable(caption = "RMSEs")
```

RMSEs

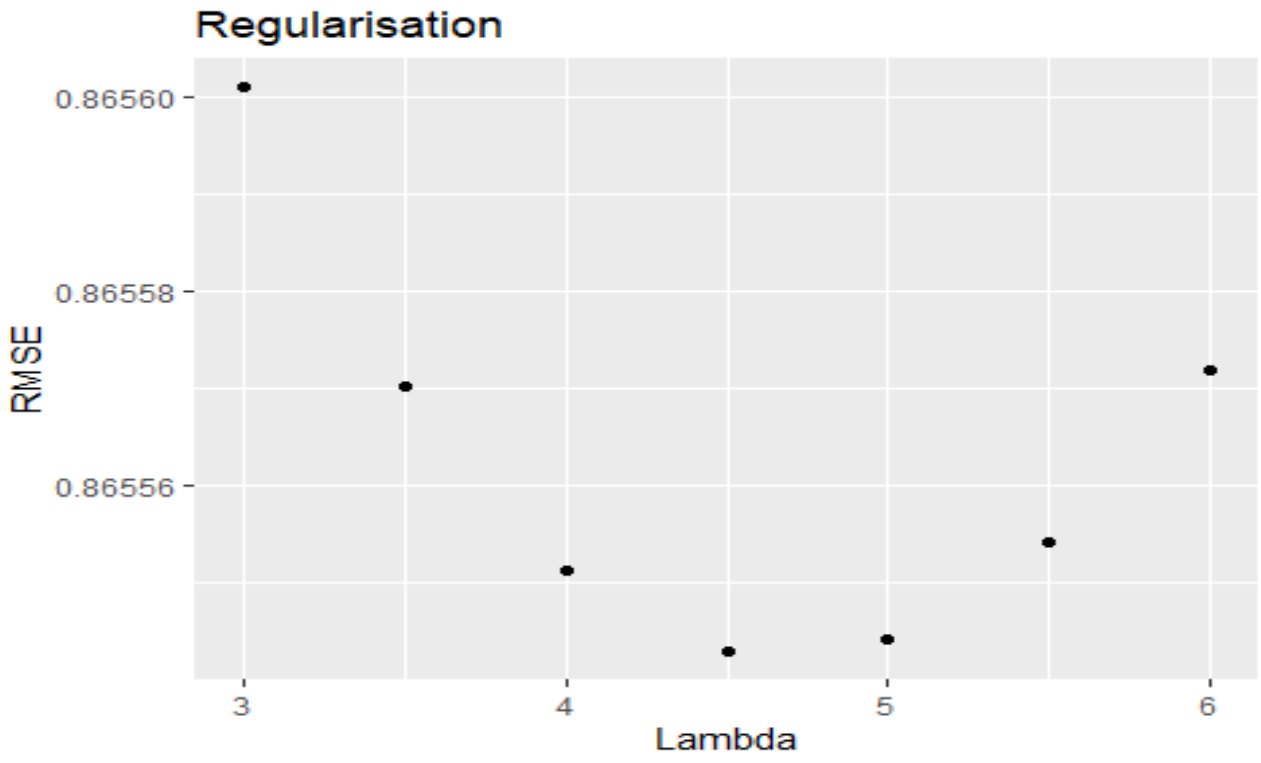
Method	RMSE
Baseline MODEL	1.0607045
Movie Effect	0.9437144
User Effect	0.9794339
User & Movie Effect	0.8661625

We obtained a better RMSE when using this model than when using user effect or movie effect alone. This is expect because the effect of both is obvious on the prediction for all reasons discussed above. B*ut* we are still away from our target, let's see if regularization will give us better results.

Regularization WITH MOVIE & USER EFFECT

The regularization will evaluate different values for *lambda*, and show us the value corresponding RMSE minimal. The best Lambda found from early runs is about 4.5, thus we set 3-6 range for the bellow sapply to go faster.

```
lambda_values <- seq(3,6,.5)
RMSE_function_reg <- sapply(lambda_values, function(l){
  mu <- mean(training$rating)
  Mi <- training %>%
    group_by(movieId) %>%
    summarize(Mi = sum(rating - mu)/(n()+l))
  Ui <- training %>%
    left_join(Mi, by="movieId") %>%
    group_by(userId) %>%
    summarize(Ui = sum(rating - Mi - mu)/(n()+l))
  predicted_ratings <- test %>%
    left_join(Mi, by = "movieId") %>%
    left_join(Ui, by = "userId") %>%
    mutate(pred = mu + Mi + Ui) %>% .$pred
    return(RMSE(predicted_ratings, test$rating))
})
qplot(lambda_values, RMSE_function_reg,
      main = "Regularisation",
      xlab = "Lambda", ylab ="RMSE" ) # Lambda vs RMSE
```



```
lambda_opt <- lambda_values[which.min(RMSE_function_reg)]# Lambda which minimizes RMSE
rmse_table <- rbind(rmse_table,
                    data_frame(Method = "User & Movie Effect Regularisation, test set ",
                    RMSE = min(RMSE_function_reg)))
rmse_table %>% knitr::kable(caption = "RMSEs")
```

RMSEs

Method	RMSE
Baseline MODEL	1.0607045
Movie Effect	0.9437144
User Effect	0.9794339
User & Movie Effect	0.8661625
User & Movie Effect Regularisation, test set	0.8655430

5. PREDICTIONS

The regularisation give as a RMSE than the first "User & Movie Effect" model. This is expected, this models takes in account the combined effects most important, gave an RMSE better than all previous models 0.8659116

Now let's make prediction using regularization outcome, we will estimate the LMSE when predictions are confronted to final_holdout_test results We need to change variables formats on *final_holdout_test* data set like we did for *edx* data set :

```
final_holdout_test <- final_holdout_test %>%
  mutate( userId = as.factor(userId),
      movieId=as.factor(movieId))
glimpse(final_holdout_test)
```

```
## Rows: 999,999
## Columns: 6
## $ userId    <fct> 1, 1, 1, 2, 2, 2, 3, 3, 4, 4, 4, 5, 5, 5, 5, 5, 5, 5, 5, 5, …
## $ movieId   <fct> 231, 480, 586, 151, 858, 1544, 590, 4995, 34, 432, 434, 85, …
## $ rating    <dbl> 5.0, 5.0, 5.0, 3.0, 2.0, 3.0, 3.5, 4.5, 5.0, 3.0, 3.0, 3.0, …
## $ timestamp <int> 838983392, 838983653, 838984068, 868246450, 868245645, 86824…
## $ title     <chr> "Dumb & Dumber (1994)", "Jurassic Park (1993)", "Home Alone …
## $ genres    <chr> "Comedy", "Action|Adventure|Sci-Fi|Thriller", "Children|Come…
```

We run the model above to estimate RMSE on final_holdout_test

```
## [1] 0.8663756
```

```
rmse_table_val <- data_frame(Method = "User & Movie Effect on final houldout test",
RMSE = val_RMSE)
rmse_table <- rbind(rmse_table,
                    data_frame(Method = "User & Movie Effect on final_houldout_test",
RMSE = val_RMSE))
```

```
rmse_table %>% knitr::kable(caption = "RMSEs")
```

RMSEs

Method	RMSE
Baseline MODEL	1.0607045
Movie Effect	0.9437144
User Effect	0.9794339
User & Movie Effect	0.8661625
User & Movie Effect Regularization, test set	0.8655430
User & Movie Effect on final_houldout_test	0.8663756

The RMSE 0.8665063 is worse than the previous obtained on test set , note training is 80% of edx and test 20%, means test counts 1.8M rows while final_holdout_test counts 1M; another issue with over fitting could explain this result as regard to Lambda stepping and range setting, we could change this to have the same size on test and validation subsets; plus probably a better optimizer (Lambda).

We can also build on current model but consider the whole edx subset for training. In fact, above, we understood the Movie and user effect, with regularization on "training" set was the best model to predict values on "test" set.

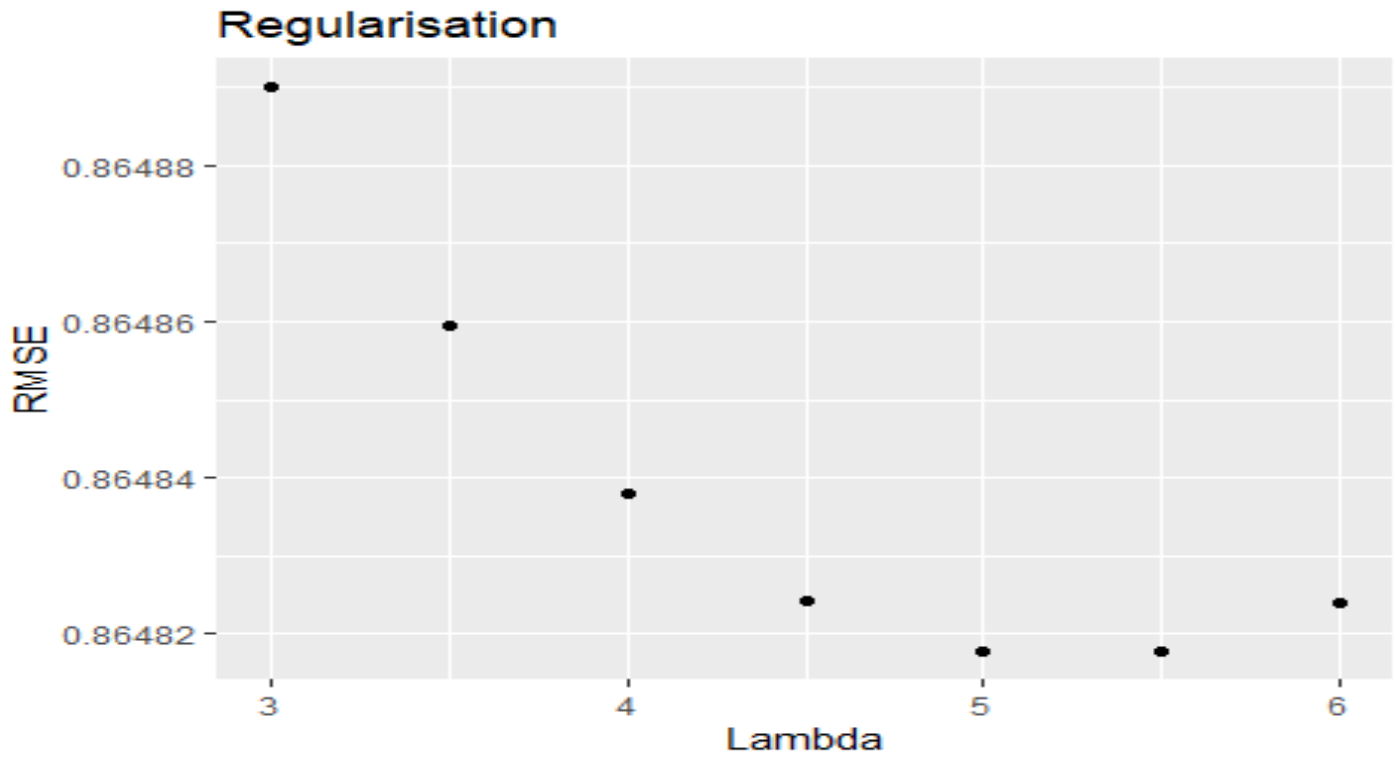
let's step back to this once, but now our training subset will be the hole "edx" and we make predictions and estimate with the best fitting model observed on edx set.

PREDICTIONS on final_holdout_test with parameters from edx set, model:

Regularized user & movie effect

```
lambda_values <- seq(3,6,.5)

RMSE_function_reg <- sapply(lambda_values, function(l){

  mu <- mean(edx$rating)

  Mi <- edx %>%
    group_by(movieId) %>%
    summarize(Mi = sum(rating - mu)/(n()+l))

  Ui <- edx %>%
    left_join(Mi, by="movieId") %>%
    group_by(userId) %>%
    summarize(Ui = sum(rating - Mi - mu)/(n()+l))

  predicted_ratings <- final_holdout_test %>%
    left_join(Mi, by = "movieId") %>%
    left_join(Ui, by = "userId") %>%
    mutate(pred = mu + Mi + Ui) %>% .$pred

  return(RMSE(predicted_ratings, final_holdout_test$rating))
```

```
})

qplot(lambda_values, RMSE_function_reg,
      main = "Regularisation",
      xlab = "Lambda", ylab ="RMSE" ) # Lambda vs RMSE
```




```
lambda_opt <- lambda_values[which.min(RMSE_function_reg)]
lambda_opt # Lambda which minimizes RMSE

## [1] 5

rmse_table <- rbind(rmse_table,
                    data_frame(Method = "User & Movie Effect Regularisation with edx
set",
                               RMSE = min(RMSE_function_reg)))
rmse_table %>% knitr::kable(caption = "RMSEs")
```

RMSEs

Method	RMSE
Baseline MODEL	1.0607045
Movie Effect	0.9437144
User Effect	0.9794339
User & Movie Effect	0.8661625
User & Movie Effect Regularisation, test set	0.8655430
User & Movie Effect on final_houldout_test	0.8663756
User & Movie Effect Regularisation with edx set	0.8648177

6. RESULTS

This is the best estimate we could find, and is on the target 0.8648177 < < 0.86490 We can observe that the better RMSE is obtained from the *User & Movie Effect* model. However, this RMSE *only obtained on the* training* set. When we move to the *final_holdout_test* data set, we obtain the worse RMSE (ignoring the baseline).the final test estimated error have probably suffered from over fitting while doing predictions with the regularization operation on training set (subset of edx) was not seen when we shifted to edx instead.

This confirms that parameters we choose from "edx" are more likely to give better predictions model, and that both user and effect, after regularization, led to the lowest RMSE; which is better yet on target. The processing time did not cause any issue, even with a standard student owned computer, the regularization algorithm took few minutes, the downloading/unzipping of movie Lens file took 5 to 10 minutes with a standard home WiFi connection

7. CONCLUSION

Both `userId` and `movieId` have predictive power to predict how a user will rate a movie. Even though movie effect seem bigger than user effect, we then needed to take both of them in account, and saw that regularization gave better precision.

This work allowed me to test a new approach on training and validation I didn't find through the literature I had time to view, this gave me good estimate: I split the edx set to training and test subset, I used them to choose a model, I then stepped back and applied the same model on the hole edx set, and estimated the LMSE on final_holdout_test. This means we can merge training and validations sets on a new training set (actually old set before split), without changing the test set initially appointed to this matter. This trick has limited the shrinking of regularization parameters.

8. TOUGHTS

Watching time effect is theoretically significant to test for eventual impact on rating a movie, weekends and holidays could be favorable for greater rates (at least on romance and family genres). Plus, the real world, a user could have seen a movie more than one time and gave different rates. Genre effect is also not integrated in the model because lack of time, knowing that genre effect is not entirely hidden by movie effect, I guess user/genre effect would raise probably more than user/movie effect. We can question the impact of the rating scale itself, smoother predictions could happen over a scale with whole integers (stars). Other aspects of recommendation system are not possible to explore with actual database, the age and gender of users, social conditions and beliefs are criteria for watching a movie, then evaluations. Film budget, duration, marketing effort deployed, director and actors reputation and history, collaborative influence with social networks are just as important regarding the evaluation of a movie. We all like to recommend a movie to relates and friends, and tend to better rate the movies we have seen. In the other hand our networks influences all our preferences. Moreover, the audience/number of watchers versus number of evaluations is a good criteria. We do not have the watchers here, and we know that not all watchers rate a movie they watched. Professionals and neurologists know certainly about other technical and behavioral features to include to the model and interpret its findings:

technology evolution in sound and image special effects for specific genres as SCI-FI. This means more collaboration and more efficient machine leaning models are actually used for rating prediction.

9. REFERENCES

This work was a good opportunity for me to review different modules early attended, and to revise related chapters on text book: *https://rafalab.dfci.harvard.edu/dsbook/*

Some items came to my understanding better with videos on YouTube channels with links bellow, I thanks all those who provide pretty didactically elements for them knowledge sharing. The valuable ones I remember are:

https://www.youtube.com/@designworld15 http://www.sthda.com/english/
http://www.zstatistics.com/videos/ https://www.youtube.com/@RProgramming101
https://www.geeksforgeeks.org/root-mean-square-error-in-r-programming/
https://towardsdatascience.com/predicting-the-survival-of-titanic-passengers-30870ccc7e8
https://www.youtube.com/@misraturp

https://github.com/MOBOUN/BOUNGIT