MovieLens

Alejandro Greco

10/10/2019

Contents

1 Introduction

This is a MovieLens Project to make a movie recommendation system based on movies data set provided in the Edx course.

The objective of this project is to obtain the lower possible Root Mean Squared Error (RMSE), lower than 0.8649, using the edx data provided validating with the validation set. The original data content about 10M registers but after EDX processing were created 2 data set, one named "edx" that is composed for about 9M observations and other named "validation" with about 1M registers both with 6 columns with not null variables.

2 Methods

In this project was evaluated 4 models to make the best recommendation possible with the lower value in loss function Root mean squared error (RMSE), the models are:

1 - Naive: A model using only the average of all ratings.

2 - Movies: A model using model 1 adding the movies because some movies are rated higher than others.

3 - Users: A model using model 2 adding the users because some users are very cranky and others love every movie. This implies that rating depends on the movies and the users.

4 - Genres: A model using model 3 adding the genre of the movie because some genres like sci-fi are more popular than others like documentaries.

Using Root mean squared error (RMSE) will would compare the efficiency of the models.

2.1 Data

This is the original data set transformation given edx set and validation set:

```
################################
# Create edx set, validation 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 1.3.0
```

```
##   ggplot2 3.2.1      purrr   0.3.3
##   tibble  2.1.3      dplyr   0.8.3
##   tidyr   1.0.0      stringr 1.4.0
##   readr   1.3.1      forcats 0.4.0
```

```
##   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-project.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
```

```
# 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(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(levels(movieId))[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")
```

```
## Warning in set.seed(1, sample.kind = "Rounding"): non-uniform 'Rounding' sampler
## used
```

```
# if using R 3.5 or earlier, use `set.seed(1)` instead
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)

################ end of the data given
```

2.2 Requirements

These packages are required for this project:

```
# Installation of packages

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

```
## Loading required package: car
```

```
## Loading required package: carData
```

```
##
## Attaching package: 'car'
```

```
## The following object is masked from 'package:dplyr':
##
##     recode
```

```
## The following object is masked from 'package:purrr':
##
##     some
```

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

```
## Loading required package: kableExtra
```

```
##
## Attaching package: 'kableExtra'
```

```
## The following object is masked from 'package:dplyr':
##
##     group_rows
```

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

```
## Loading required package: ggthemes
```

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

```
## Loading required package: rmarkdown
```

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

```
## Loading required package: tinytex
```

2.3 Data cleaning

First, it is suggested to create a copy of the edx and validation data set to protect the integrity of the original data.

```
edx_clean        <-  edx
validation_clean <-  validation
```

In the table below is the head of the data set.

```
head(edx_clean)
```

```
##   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
```

As you could observe, there are two columns hard to read, timestamp and the genres.

The timestamp column is not in date-time format and genres are multiple combinations. We will start with the timestamp changing its format.

```
edx_clean$timestamp_format <- as.POSIXct(edx_clean$timestamp, origin="1970-01-01")
validation_clean$timestamp_format <- as.POSIXct(validation_clean$timestamp, origin="1970-01-01")
```

On one hand, based on that new things are more attractive than the old ones, the year when the movie was made could affect the rate.

```
edx_clean$year_evaluated<-format(edx_clean$timestamp_format,"%Y")
validation_clean$year_evaluated<-format(validation_clean$timestamp_format,"%Y")
```

On the other hand, genres is other column to transform because some of them are more popular than others and for this reason is necessary to split one genre per row in a new column.

```
edx_clean$genre <- edx_clean$genres
edx_clean <- edx_clean %>% separate_rows(genre,sep = "\\|")

validation_clean$genre <- validation_clean$genres
validation_clean <- validation_clean %>% separate_rows(genre,sep = "\\|")
```

2.4 Data exploration and visualization

As you can notice bellow, there are 10677 uniques movies, 69878 uniques users that rate from 1 to 5 since 1995-01-09 08:46:49 to 2009-01-05 02:02:16 -03 and there is no null value. The first and last year were particulate because in 1995 were 2 evaluations and in 2009 where less evaluation than 2008. There are 20 uniques genres and 797 combinations of one or more because each movie could have more than one genre.

The size of the data is about 10 million rows, split into edx data with 9M and validation 1 million.

```
head(edx_clean, 10)
```

```
##        userId movieId rating timestamp           title
## 1...1       1     122      5 838985046 Boomerang (1992)
## 1...2       1     122      5 838985046 Boomerang (1992)
```

```
## 2...3       1     185      5 838983525  Net, The (1995)
## 2...4       1     185      5 838983525  Net, The (1995)
## 2...5       1     185      5 838983525  Net, The (1995)
## 4...6       1     292      5 838983421  Outbreak (1995)
## 4...7       1     292      5 838983421  Outbreak (1995)
## 4...8       1     292      5 838983421  Outbreak (1995)
## 4...9       1     292      5 838983421  Outbreak (1995)
## 5...10      1     316      5 838983392  Stargate (1994)
##                             genres    timestamp_format year_evaluated    genre
## 1...1                Comedy|Romance 1996-08-02 07:24:06           1996   Comedy
## 1...2                Comedy|Romance 1996-08-02 07:24:06           1996  Romance
## 2...3         Action|Crime|Thriller 1996-08-02 06:58:45           1996   Action
## 2...4         Action|Crime|Thriller 1996-08-02 06:58:45           1996    Crime
## 2...5         Action|Crime|Thriller 1996-08-02 06:58:45           1996 Thriller
## 4...6  Action|Drama|Sci-Fi|Thriller 1996-08-02 06:57:01           1996   Action
## 4...7  Action|Drama|Sci-Fi|Thriller 1996-08-02 06:57:01           1996    Drama
## 4...8  Action|Drama|Sci-Fi|Thriller 1996-08-02 06:57:01           1996   Sci-Fi
## 4...9  Action|Drama|Sci-Fi|Thriller 1996-08-02 06:57:01           1996 Thriller
## 5...10      Action|Adventure|Sci-Fi 1996-08-02 06:56:32           1996   Action
```

```
which(is.na(edx_clean))
```

```
## integer(0)
```

```
movies_by_year<-edx_clean %>%
                group_by(year_evaluated) %>%
                summarise(number_uniques_movies=n_distinct(movieId)
                          )
kable(movies_by_year)
```

year_evaluated	number_uniques_movies
1995	2
1996	1385
1997	1664
1998	2261
1999	3009
2000	3810
2001	4656
2002	5677
2003	6743
2004	7830
2005	8281
2006	8490
2007	9163
2008	9594
2009	3445

```
# Movies, Users and Genres
# The unique genres are 20 and there are 797 combinations of one or more.

resumen_unique <- edx_clean %>%
                  summarise(
                  Movies = n_distinct(movieId),
                  Users = n_distinct(userId),
                  Genres = n_distinct(genres),
                  Genre = n_distinct(genre)
```

```
                                          )
resumen_unique
```

```
##   Movies Users Genres Genre
## 1  10677 69878    797    20
```

```
#### Min time and max time
date_min<-min(edx_clean$timestamp_format)
date_max<-max(edx_clean$timestamp_format)

date_min
```

```
## [1] "1995-01-09 08:46:49 -03"
```

```
date_max
```

```
## [1] "2009-01-05 02:02:16 -03"
```

Because the size of the data set is too big to process it or to graphic it for some computers, I made a random sample of 666 rows to represent the data. For example, for 9 million rows, the size of a sample could be 16,610 with a confidence level of 99% and a confidence interval of 1%. This sample is just to explore graphically and will not be used for the models. I will use a 99% confidential level and a 5% margin of error.

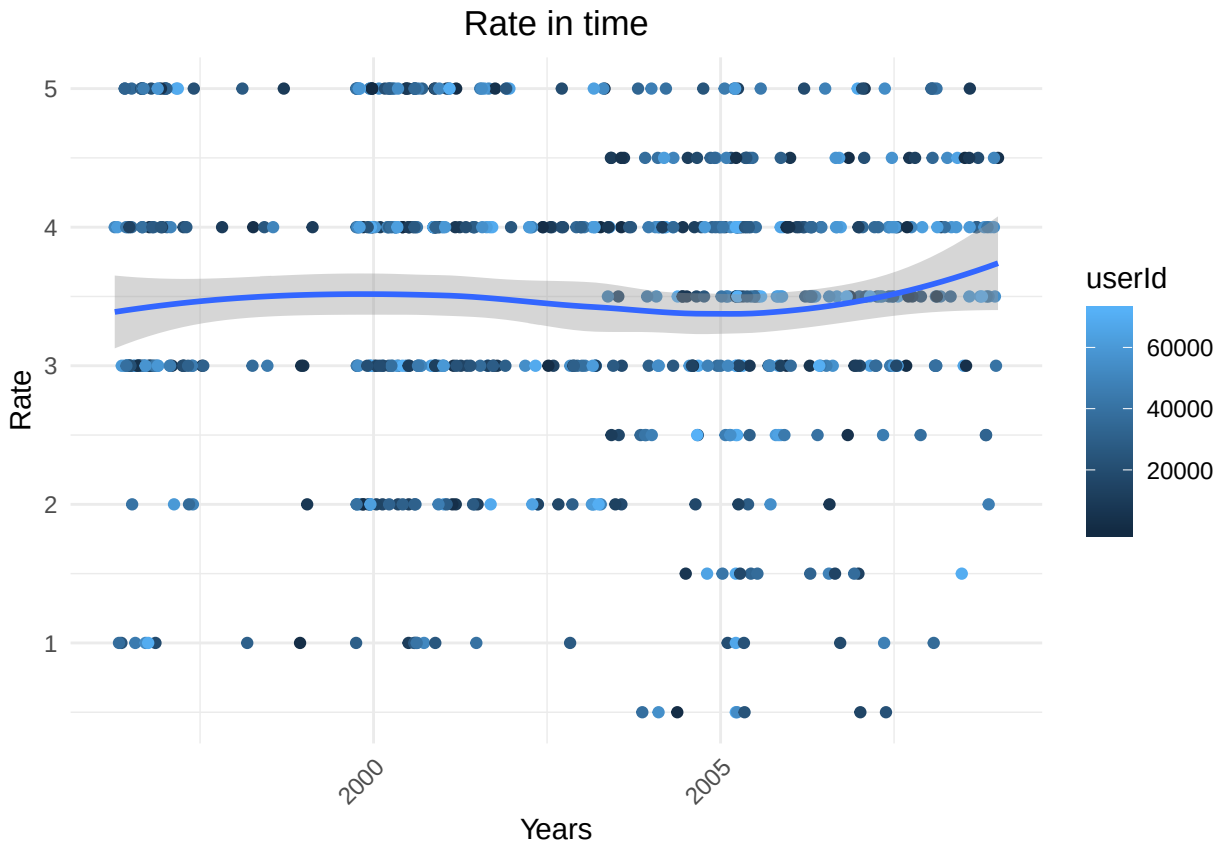
```
edx_sample<-edx_clean[sample(nrow(edx_clean),666),]
```

2.4.1 Time

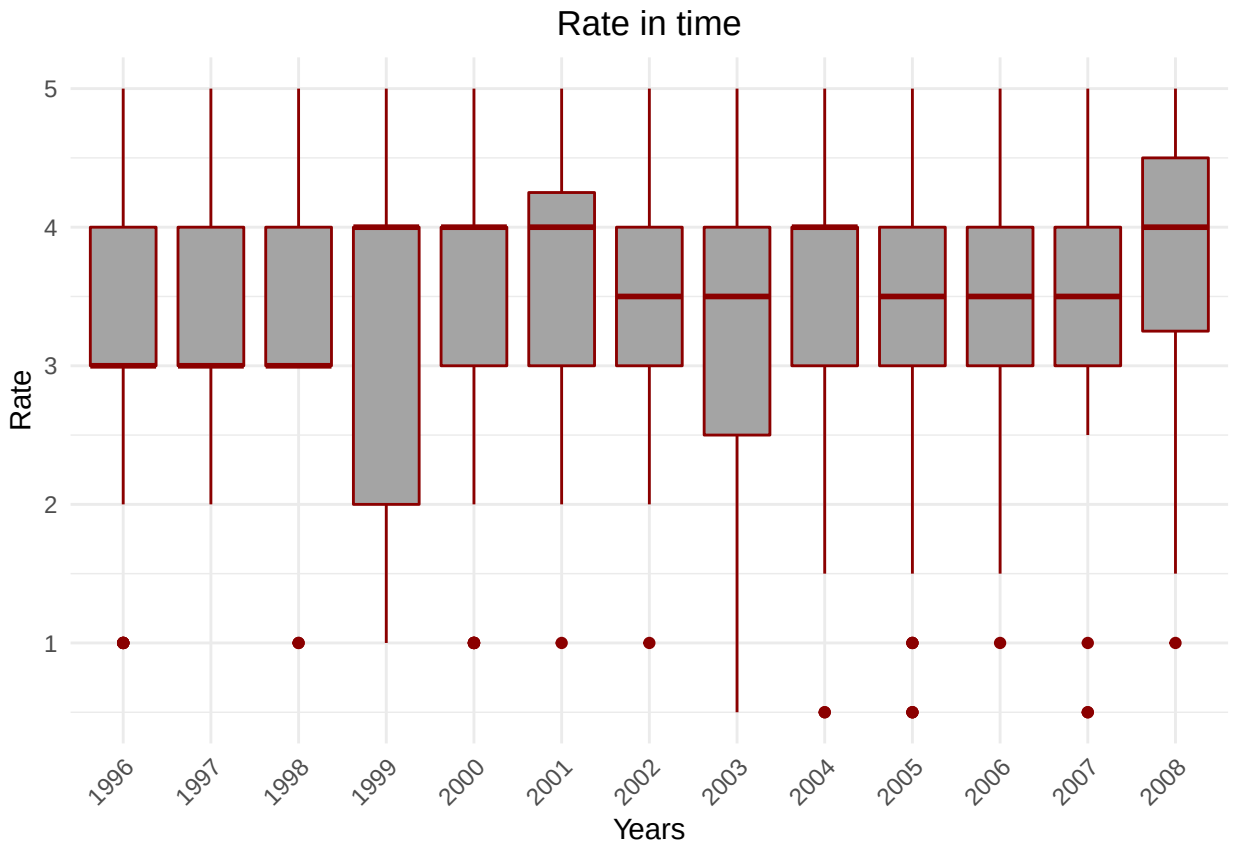
Using the sample to make a scatter plot and box plot to graphic time and rating, we can observe something interesting. The half-point ratings started about 2003 when we are guessing was enable and the rating looks very stable between 3 and 4 over the years.

```
sp_rating_in_time_temp <-  qplot(timestamp_format, rating, colour = userId,   data = edx_sample) +
                           geom_smooth() +
                           labs(x="Years", y="Rate")

sp_rating_in_time <- sp_rating_in_time_temp +
                     theme_minimal() +
                     ggtitle("Rate in time") +
                     theme(axis.text.x = element_text(angle = 45, hjust = 1),
                                         plot.title=element_text(hjust=0.5))

sp_rating_in_time
```

```
## `geom_smooth()` using method = 'loess' and formula 'y ~ x'
```




```
boxplot_year_ratting <- ggplot(edx_sample,
                               aes(group=year_evaluated,
                                   x=year_evaluated,
                                   y=rating)) +
                         geom_boxplot(fill='#A4A4A4', color="darkred") + theme_minimal()+
                         labs(x="Years", y="Rate")   +
                         ggtitle("Rate in time")+
                        theme(axis.text.x = element_text(angle = 45, hjust = 1),
                        plot.title=element_text(hjust=0.5))

boxplot_year_ratting
```




```
edx_clean$year_evaluated<-format(edx_clean$timestamp_format,"%Y")

date_min
```

```
## [1] "1995-01-09 08:46:49 -03"
```

```
date_max
```

```
## [1] "2009-01-05 02:02:16 -03"
```

Now, using the full data, we can notice that started in 1995-01-09 to 2009-01-05, the years 1995 and 2009 are incomplete and more of that, in the 2003 open to half points evaluation. The graph shows that every year increase the number of movies rated (2009 is incomplete).

```
year_mean_rating<-edx_clean %>%
                  group_by(year_evaluated) %>%
                  summarise(number_movies=n(),
                            avg_rating=mean(rating),
                            max_rating=max(rating),
                            min_rating=min(rating),
                            sd_rating=sd(rating)
                           )

year_mean_rating
```

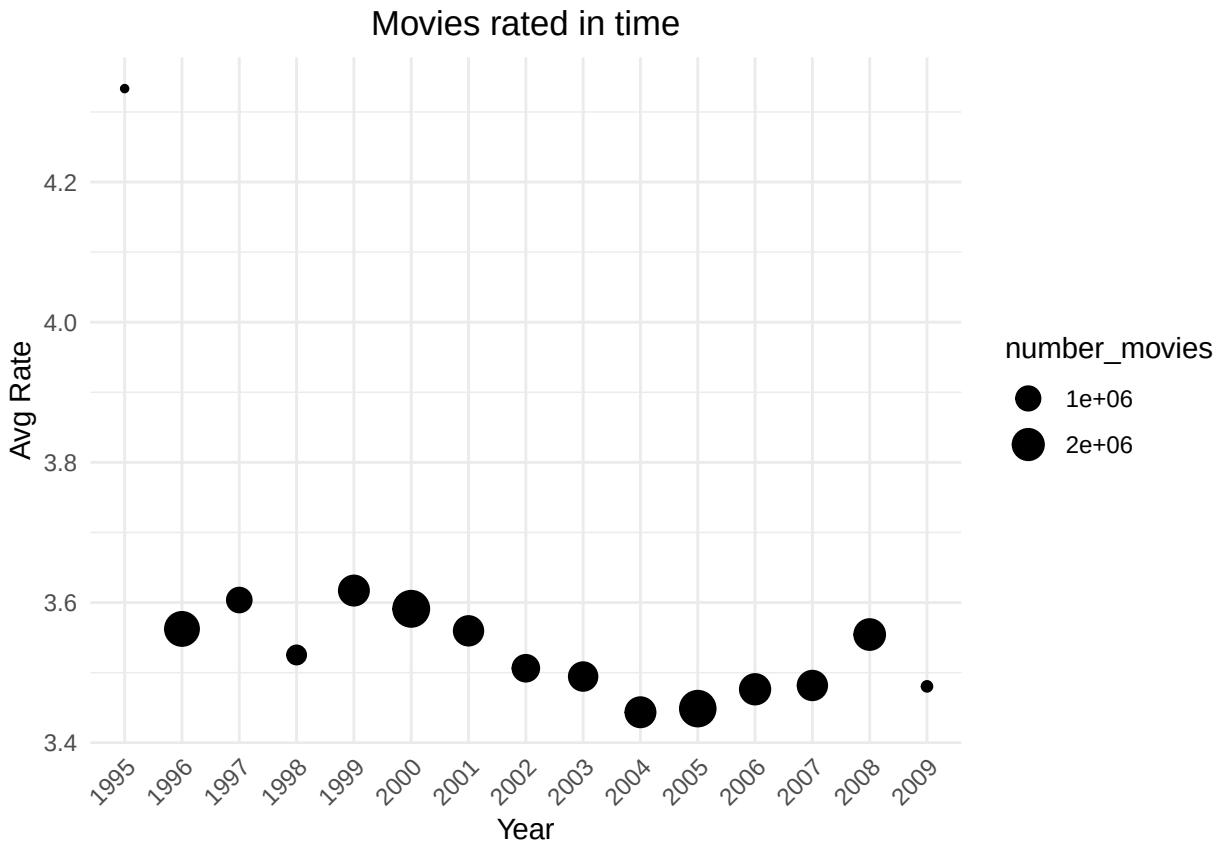
```
## # A tibble: 15 x 6
##    year_evaluated number_movies avg_rating max_rating min_rating sd_rating
##    <chr>                  <int>      <dbl>      <dbl>      <dbl>     <dbl>
##  1 1995                       6       4.33          5          3      1.03
```

```
##  2 1996                2480792       3.56          5        1       0.983
##  3 1997                1046181       3.60          5        1       1.00
##  4 1998                 466191       3.53          5        1       1.12
##  5 1999                1796389       3.62          5        1       1.12
##  6 2000                2879126       3.59          5        1       1.12
##  7 2001                1707984       3.56          5        1       1.11
##  8 2002                1303274       3.51          5        1       1.11
##  9 2003                1567057       3.49          5        0.5     1.05
## 10 2004                1799173       3.44          5        0.5     1.03
## 11 2005                2806522       3.45          5        0.5     1.03
## 12 2006                1852741       3.48          5        0.5     1.02
## 13 2007                1709265       3.48          5        0.5     1.03
## 14 2008                1921210       3.55          5        0.5     0.990
## 15 2009                  35512       3.48          5        0.5     0.993
```

```
attach(year_mean_rating)
year_mean_rating_sorted <- year_mean_rating[order(-number_movies),]

head(year_mean_rating_sorted)
```

```
## # A tibble: 6 x 6
##   year_evaluated number_movies avg_rating max_rating min_rating sd_rating
##   <chr>                  <int>      <dbl>      <dbl>      <dbl>     <dbl>
## 1 2000                 2879126       3.59          5        1       1.12
## 2 2005                 2806522       3.45          5        0.5     1.03
## 3 1996                 2480792       3.56          5        1       0.983
## 4 2008                 1921210       3.55          5        0.5     0.990
## 5 2006                 1852741       3.48          5        0.5     1.02
## 6 2004                 1799173       3.44          5        0.5     1.03
```

```
movies_time_size <- qplot(year_evaluated,avg_rating , data = year_mean_rating, size = number_movies)+
                    xlab("Year")+ylab("Avg Rate")  + theme_minimal()+
                    ggtitle("Movies rated in time")+
                    theme(axis.text.x = element_text(angle = 45, hjust = 1),
                          plot.title=element_text(hjust=0.5))

movies_time_size
```



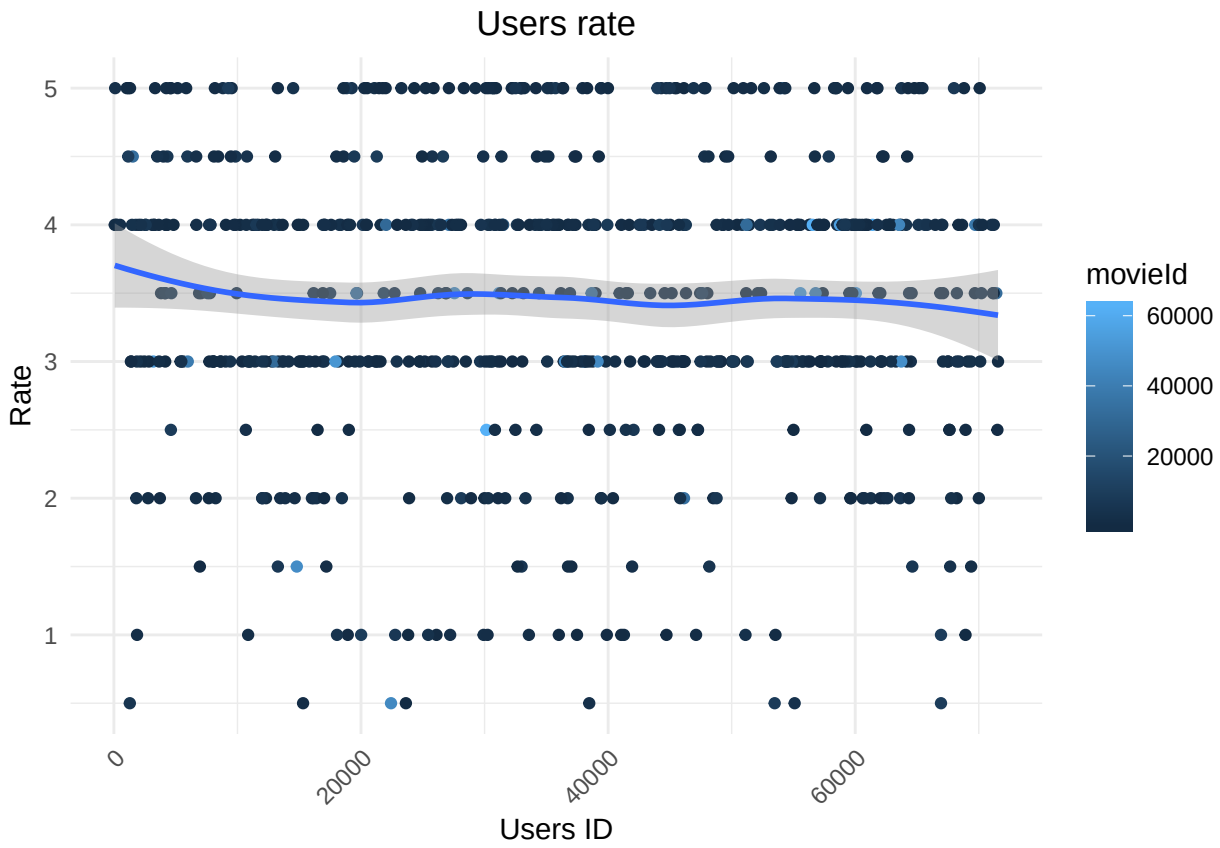

```
#detach(year_mean_rating)  #only if is necessary to run it more than one time because if is already att
```

2.4.2 User

Each person is a world, and they express themselves through the rating of the movies been some of them too good and others too mean. Additionally, some of them have rated more than 6 thousand movies and some of them just 10.

```
sp_users_rating_temp <- qplot(userId,
                              rating,
                              colour = movieId,
                              data = edx_sample) +
                        geom_smooth() +
                        labs(x="Users ID", y="Rate")

sp_users_rating <- sp_users_rating_temp +
                   theme_minimal() +
                   ggtitle("Users rate") +
                   theme(axis.text.x = element_text(angle = 45, hjust = 1),
                         plot.title=element_text(hjust=0.5))

sp_users_rating
```

```
## `geom_smooth()` using method = 'loess' and formula 'y ~ x'
```




```
###It was used the edx original data to calculate the unique number of movies rated by users easer.
nomber_movie_users_rated <- edx %>%
                            group_by(userId) %>%
                            summarise(unique_number_movies=n_distinct(movieId),
                                      avg_rating=mean(rating),
                                       sd_rating=sd(rating)
                                      )

nomber_movie_users_rated
```

```
## # A tibble: 69,878 x 4
##    userId unique_number_movies avg_rating sd_rating
##     <int>                <int>      <dbl>     <dbl>
##  1      1                   19       5        0
##  2      2                   17       3.29     0.920
##  3      3                   31       3.94     0.761
##  4      4                   35       4.06     1.16
##  5      5                   74       3.92     1.11
##  6      6                   39       3.95     1.10
##  7      7                   96       3.86     0.975
##  8      8                  727       3.39     0.777
##  9      9                   21       4.05     1.30
## 10     10                  112       3.83     0.879
## # … with 69,868 more rows
```

```
top <- head(arrange(nomber_movie_users_rated,desc(unique_number_movies)), n = 10)
bottom <- tail(arrange(nomber_movie_users_rated,desc(unique_number_movies)), n = 10)
```

```
kable(top)
```

userId	unique_number_movies	avg_rating	sd_rating
59269	6616	3.264586	0.6386508
67385	6360	3.197720	0.9568548
14463	4648	2.403615	0.6881860
68259	4036	3.576933	1.0513298
27468	4023	3.826871	0.7343875
19635	3771	3.498807	0.7783109
3817	3733	3.112510	0.5787978
63134	3371	3.268170	0.9568703
58357	3361	3.000744	0.7984822
27584	3142	3.001432	0.7187717

```
kable(bottom)
```

userId	unique_number_movies	avg_rating	sd_rating
57894	14	4.357143	1.0082081
62317	14	3.714286	0.8254203
63143	14	3.928571	0.8287419
68161	14	3.428571	1.3134971
68293	14	4.357143	0.8418974
71344	14	3.214286	1.0509023
15719	13	3.769231	1.2351684
50608	13	3.923077	1.4411534
22170	12	4.000000	0.7385489
62516	10	2.250000	1.1365151

```
#average standard deviation

sd_general_rating <- edx %>%  summarise( sd_general_srating=sd(rating) )
sd_general_rating
```

```
##   sd_general_srating
## 1           1.060331
```

```
avg_sd_top <- top  %>%  summarise( avg_sd_rating_top=mean(sd_rating) )
avg_sd_top
```

```
## # A tibble: 1 x 1
##   avg_sd_rating_top
##               <dbl>
## 1             0.790
```

```
avg_sd_bottom <- bottom  %>%  summarise( avg_sd_rating_bottom=mean(sd_rating) )
avg_sd_bottom
```

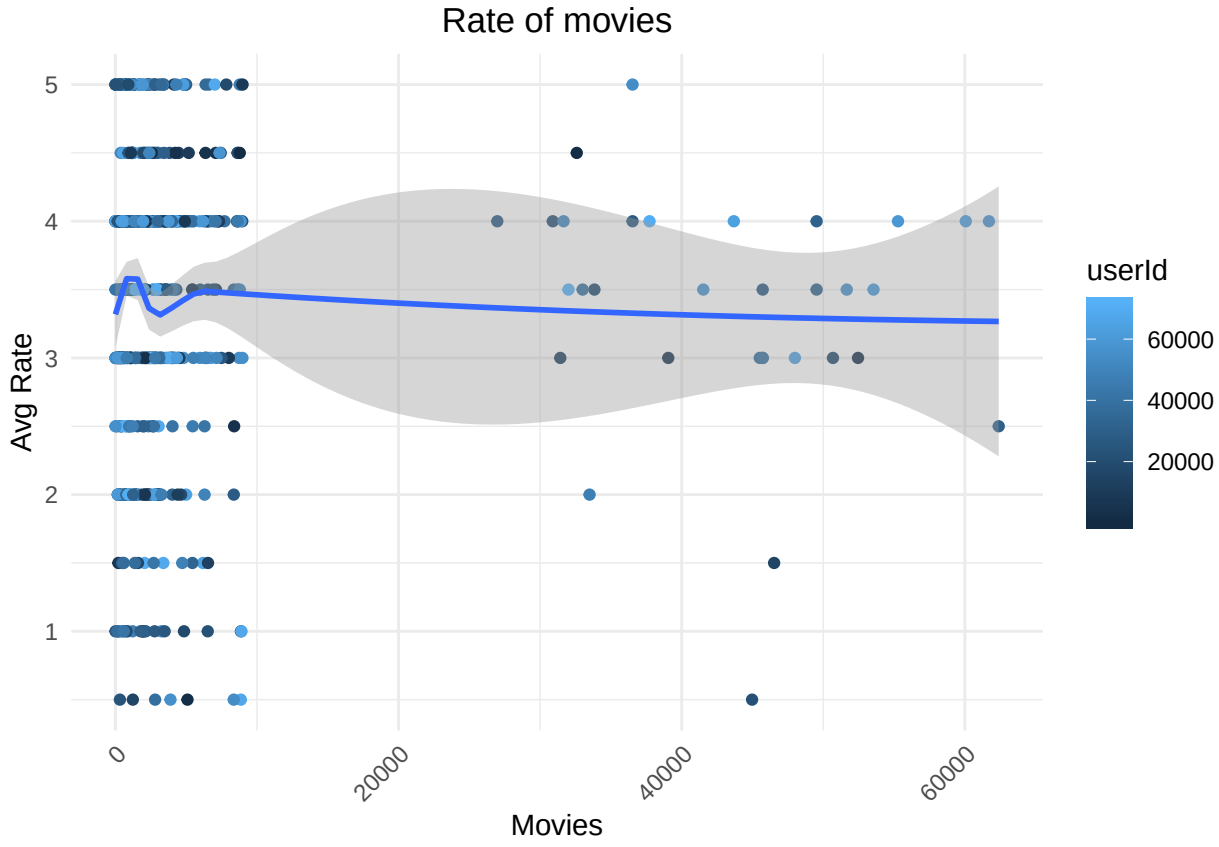
```
## # A tibble: 1 x 1
##   avg_sd_rating_bottom
##                  <dbl>
## 1                 1.04
```

2.4.3 Movies

Each movie is a piece of art, some of them are very famous but some others not, for this reason, the rating of the movie was definitely affected by itself. Here we can clearly observe that it is related to the rating with the movie, and the evaluator (users).

```
sp_movie_rating<-qplot(movieId,
                       rating,
                       colour = userId,
                       data = edx_sample) +
                  geom_smooth()+  theme_minimal() +
                  xlab("Movies")+ylab("Avg Rate")  +
                  ggtitle("Rate of movies")+
                  theme(axis.text.x = element_text(angle = 45, hjust = 1),
                        plot.title=element_text(hjust=0.5))

sp_movie_rating
```

```
## `geom_smooth()` using method = 'loess' and formula 'y ~ x'
```



2.4.4 Genre

The genre could affect the rate because some genres have more passionate followers that could rate higher or lower for example sci-fi.

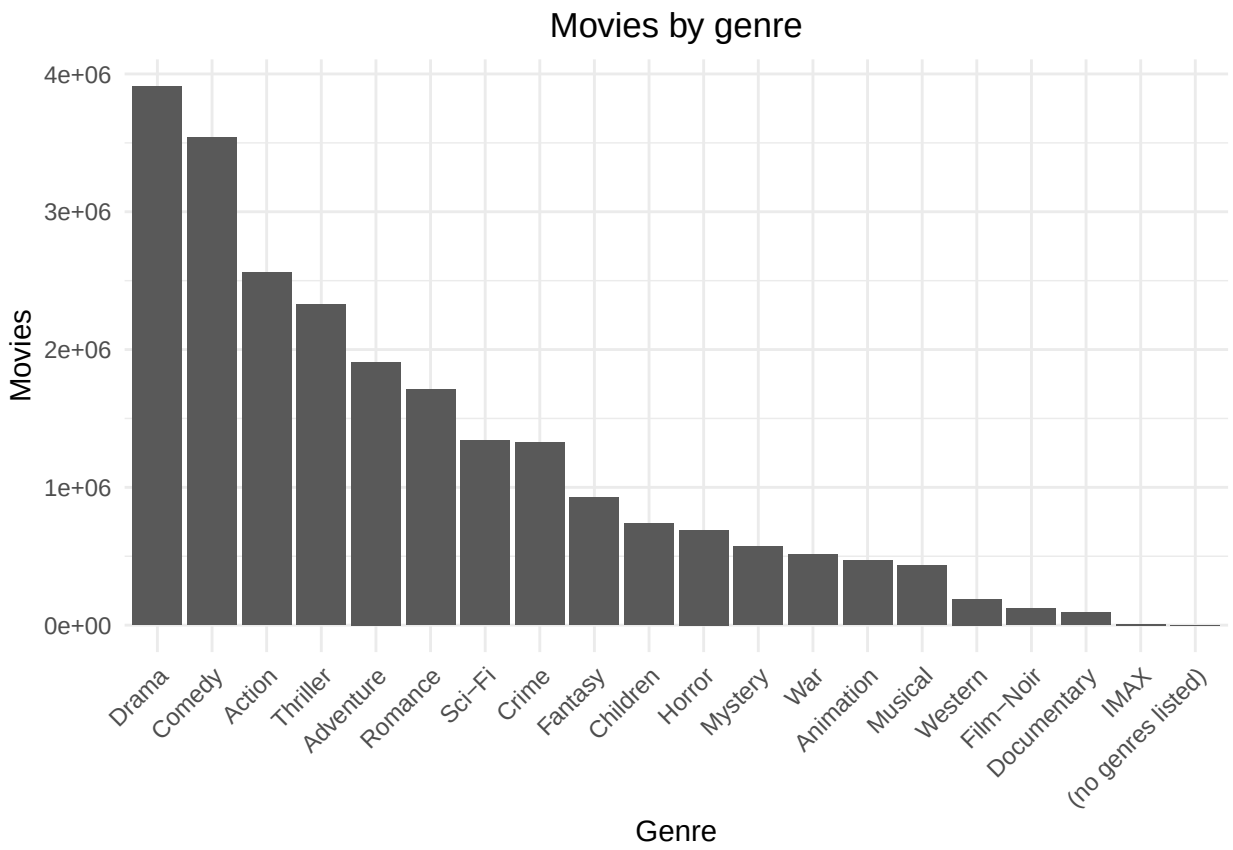
```
genre_mean_rating<-edx_clean %>%
                  group_by(genre) %>%
                  summarise(number_movies=n(),
                            avg_rating=mean(rating),
                            max_rating=max(rating),
                            min_rating=min(rating),
                            sd_rating=sd(rating))

attach(genre_mean_rating)
```

```
## The following objects are masked from year_mean_rating:
##
##     avg_rating, max_rating, min_rating, number_movies, sd_rating
```

```
genre_mean_rating_sorted<-genre_mean_rating[order(-number_movies),]

head(genre_mean_rating_sorted,10)
```

```
## # A tibble: 10 x 6
##    genre     number_movies avg_rating max_rating min_rating sd_rating
##    <chr>             <int>      <dbl>      <dbl>      <dbl>     <dbl>
##  1 Drama           3910127       3.67          5        0.5     0.995
##  2 Comedy          3540930       3.44          5        0.5     1.07
##  3 Action          2560545       3.42          5        0.5     1.07
##  4 Thriller        2325899       3.51          5        0.5     1.03
##  5 Adventure       1908892       3.49          5        0.5     1.05
##  6 Romance         1712100       3.55          5        0.5     1.03
##  7 Sci-Fi          1341183       3.40          5        0.5     1.09
##  8 Crime           1327715       3.67          5        0.5     1.01
##  9 Fantasy          925637       3.50          5        0.5     1.07
## 10 Children         737994       3.42          5        0.5     1.09
```

```
movie_genre_graph_temp<-ggplot(data=genre_mean_rating,
                               aes(x=reorder(genre,-number_movies),
                                   y=number_movies)) +
                        geom_bar(stat="identity") +
                        xlab("Genre")+
                        ylab("Movies")

movie_genre_graph<- movie_genre_graph_temp  +
                    theme_minimal()+
                    ggtitle("Movies by genre") +
                     theme(axis.text.x = element_text(angle = 45, hjust = 1),
                           plot.title=element_text(hjust=0.5))

movie_genre_graph
```




```
#detach(genre_mean_rating)  #only if is necessary to run it more than one time because if is already at

genre_sum<-edx_clean %>%
           group_by(genre) %>%
           summarize(count = n()) %>%
           arrange(desc(count))

genre_sum
```

```
## # A tibble: 20 x 2
##    genre                count
##    <chr>                <int>
##  1 Drama              3910127
##  2 Comedy             3540930
##  3 Action             2560545
##  4 Thriller           2325899
##  5 Adventure          1908892
##  6 Romance            1712100
##  7 Sci-Fi             1341183
##  8 Crime              1327715
##  9 Fantasy             925637
## 10 Children            737994
## 11 Horror              691485
## 12 Mystery             568332
## 13 War                 511147
## 14 Animation           467168
## 15 Musical             433080
```

```
## 16 Western             189394
## 17 Film-Noir           118541
## 18 Documentary          93066
## 19 IMAX                  8181
## 20 (no genres listed)       7
```

```
kable(resumen_unique)
```

Movies	Users	Genres	Genre
10677	69878	797	20

2.5 Insights gained

It is interesting how in this data every year more and more movies are rated and some users have 6,616 movies rated, one rate for each movie. This is the user with more rates but on average a movie is 100 minutes (1.66667 horas) for each movie, that represent 6,616,000 minutes or 11,026.67 horas, or 459.44 days, or 1.25875 years of continues watching. In a more realistic way, if we divide the total hours between 40 working hours per week we have 275.67 working weeks divided by 52 weeks per year, which means 5.30 working years without vacations. But this is just the number one, is interested how from this user to the 10Th drop the rates to half. This tops 10 users with more than 3,000 rates with a standard deviation on average of 0.79 vs the bottom of the user with less than 14 movies and more than 10 they have an average of SD of 1.042. for the whole data set is 1.06 that is much closer to the bottom than to the top.

As you could observe, the number of uniques movies is about 10 thousand but each movie is rated more than one time and for more than one user, and on top of that, each movie could have more than one genre making this data a matrix of information.

2.6 Modeling approach

This study considered 4 different approaches:

1 - The simple average of the rating;

2 - The previus model adding movies effect;

3 - The previus model adding users effect;

4 - The previous model adding genres effect.

All of them were evaluated using RMSE and the best option was the 4th as you could observe in the results.

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

2.6.1 Method 1, rating average

This method have the recommendation using just the rating.

```
mu_hat <- mean(edx_clean$rating)
mu_hat
```

```
## [1] 3.527019
```

```
rmse_1_mu <- RMSE(edx_clean$rating, mu_hat)
rmse_1_mu
```

```
## [1] 1.052262
```

```
methods_rmse_results <- data.frame(model="General rating average only", RMSE=rmse_1_mu)

kable(methods_rmse_results)
```

model	RMSE
General rating average only	1.052262

2.6.2 Method 2, movies effect

This method have the recommendation using the rating and considering the effect of each movie.

```
# first we need movie average
mu <- mean(edx_clean$rating)
mu
```

```
## [1] 3.527019
```

```
movie_mu <- edx_clean %>%
            group_by(movieId) %>%
            summarize(b_m = mean(rating - mu))
movie_mu
```

```
## # A tibble: 10,677 x 2
##    movieId    b_m
##      <dbl>  <dbl>
##  1       1  0.401
##  2       2 -0.322
##  3       3 -0.380
##  4       4 -0.663
##  5       5 -0.458
##  6       6  0.288
##  7       7 -0.168
##  8       8 -0.392
##  9       9 -0.529
## 10      10 -0.101
## # … with 10,667 more rows
```

```
predicted_ratings_2 <- mu + validation_clean %>%
                      left_join(movie_mu, by='movieId') %>%
                        pull(b_m)

model_2_rmse <- RMSE(predicted_ratings_2, validation_clean$rating)
model_2_rmse
```

```
## [1] 0.94107
```

```
methods_rmse_results <- methods_rmse_results %>%
                        add_row(model="Movies effect", RMSE=model_2_rmse)

kable(methods_rmse_results)
```

model	RMSE
General rating average only	1.052262
Movies effect	0.941070

2.6.3 Method 3, users effect

This method have the recommendation using the rating and considering the effect of each movie and users.

```
user_mu <- edx_clean %>%
          left_join(movie_mu, by='movieId') %>%
          group_by(userId) %>%
```

```
          summarize(b_u = mean(rating - mu - b_m))

predicted_ratings_3 <- validation_clean %>%
                       left_join(movie_mu, by='movieId') %>%
                       left_join(user_mu, by='userId') %>%
                       mutate(pred = mu + b_m + b_u) %>%
                       pull(pred)

model_3_rmse <- RMSE(predicted_ratings_3, validation_clean$rating)
model_3_rmse
```

```
## [1] 0.863366
```

```
methods_rmse_results <- methods_rmse_results %>%
                        add_row(model="Users effect", RMSE=model_3_rmse)

kable(methods_rmse_results)
```

model	RMSE
General rating average only	1.052262
Movies effect	0.941070
Users effect	0.863366

2.6.4 Method 4, genres effect

This method have the recommendation using the rating and considering the effect of each movie, users and genre.

```
genre_mu <- edx_clean %>%
            left_join(movie_mu, by='movieId') %>%
            left_join(user_mu, by='userId') %>%
            group_by(genre) %>%
            summarize(b_g = mean(rating - mu - b_m - b_u))
genre_mu
```

```
## # A tibble: 20 x 2
##    genre                    b_g
##    <chr>                  <dbl>
##  1 (no genres listed)  0.256
##  2 Action             -0.0118
##  3 Adventure          -0.0143
##  4 Animation          -0.0152
##  5 Children           -0.0242
##  6 Comedy              0.00161
##  7 Crime               0.0117
##  8 Documentary         0.0738
##  9 Drama               0.0157
## 10 Fantasy            -0.00442
## 11 Film-Noir           0.0359
## 12 Horror              0.00881
## 13 IMAX               -0.00533
## 14 Musical            -0.00838
## 15 Mystery             0.0173
## 16 Romance             0.000587
## 17 Sci-Fi             -0.0121
## 18 Thriller           -0.00265
```

```
## 19 War                 0.00521
## 20 Western            -0.00560
```

```
predicted_ratings_4 <- validation_clean %>%
                       left_join(movie_mu, by='movieId') %>%
                       left_join(user_mu, by='userId') %>%
                       left_join(genre_mu, by='genre') %>%
                       mutate(pred = mu + b_m + b_u + b_g) %>%
                       pull(pred)

model_4_rmse <- RMSE(predicted_ratings_4, validation_clean$rating)
model_4_rmse
```

```
## [1] 0.8632723
```

```
methods_rmse_results <- methods_rmse_results %>%
                          add_row(model="Genres effect", RMSE=model_4_rmse)

kable(methods_rmse_results)
```

model	RMSE
General rating average only	1.0522618
Movies effect	0.9410700
Users effect	0.8633660
Genres effect	0.8632723

3 Results

In conclusion, the best model is the number 4 with excellent RMSE of 0.8632723 Adding more variables to the model increase the complexity giving better predictions but it is no infinity. Every time a new dimension is added to the model improves the RMSE but in this study after the 3rd and 4Th, the improvement is very low because it is harder to predict better even with more dimensions. A movie with an average 100 minutes composed of scenes, actors, stories, sounds, special effects and expression of feelings that combined in a specific order and in a specif way could influence the watcher perception. These variables are hard to measure objectively making hard to improve more the recommendation system for movies.

```
kable(methods_rmse_results)
```

model	RMSE
General rating average only	1.0522618
Movies effect	0.9410700
Users effect	0.8633660
Genres effect	0.8632723

4 Conclusion

4.1 Summary of the report

In this project we could evaluated more than 10 thousands of movie data set of about 13 years of evaluation of almost 70 thousand users to demonstrate that despite of the fact that each movie is an expression of art, we use the rating of all the users had made, considering that some movies are more popular than others, that each user is unique and the genre of the movie to make a recommendation with an RMSE of 0.86327 that is very good based on the difficulty of making a recommendation to a variety of people.

4.2 Limitations

In general, the limitations of this project are the data that is limited in a 13-year movie rating sample of some users with a change of methodology in the rating adding half points in 2003 and the availability of hardware that is definitely a limitation taking a lot of time to make some calculations or graphics.

The universe of the data is not the only limitation, also the format of the data classification like the genre that could be better if a movie could be classified in just one genre and not in multiples. Like the music how could be classified as only pop? what is rock? the same happens with the movie because they are human expressions been considered as art and are not easy to classify in just one genre.

Additionally, the only movie variables we have are the title, genre and year in the title limiting our possibilities of analysis. Some information about the movie could be affecting rate like the duration of the movie, the number of scenes, actors, stories, sounds, special effects and expression of feelings by each scene and order of them.

4.3 Future work

This study riched great results but it still has great potential to improve. The above analysis is limited and we want to extend the discussion and make a few further recommendations.

From the movie perspective, the year when were rated it doesn't look like affected the user preference, but perhaps could be relevant related to the year when the movie appears in the catalog and when the movie was on theaters. Perhaps, with more details of the movies like actors, duration, synopsis, the number of scenes, stories, script, music and special effects could help not only to determinate a better recommendation but also could help to developed better movies.

From the user perspective, there are several studies that suggest that the genre of the person, age, environment and culture determines the behavior and preferences of the person that could affect the rate. Based

on this, could be interested to improve customer satisfaction to considered their genre, location, age and culture. The location is affecting the rate? by countries? or by cities? Is it more probably a better rate when it is new rather than if it is old? The people that were born in the '80s prefer more sci-fi than comedy? Are some actors better than others? This question could be answered in future works.