MovieLens Recommendation System Project

Harvardx's Data Science Professional Certificate

Otman Cherrati

7/16/2021

Contents

1 Intorduction :

This report is part of the final capstone course of HarvardX Data Science Professional certificate taught by the famous professor Rafael Irizzary. The report is on MovieLens data set. MovieLens is run by GroupLens Research, a research lab in the Department of Computer Science and Engineering at the University of Minnesota. MovieLens recommends movies for its users to watch, based on their film preferences using collaborative filtering of members movie ratings and movie reviews.

1.1 Object :

Our goal in this analysis is to create a movie recommendation system, using MovieLens 10M data set, the system must be capable of generate predictions of users rating based on a training datavset.

We will start by taking a look on the databset and then exploring the different variables to see if there is any trends that can help us make some assumptions. After cleaning the data we will start building and testing our models based on what we have learned in the exploratory analysis. The evaluation of the models is based on the RMSE - Root Mean Squared Error that should be at least lower than 0.87750.

$$\text{RMSE} = \sqrt{\frac{1}{n}\sum_{t=1}^{n} e_t^2}$$

Finally we will apply our final best model to the validation data set and discuss the results.

1.2 package used :

tidyverse
dplyr
ggplot2
stringr
data.table
caret
lubridate
Xgboost
knitr

1.3 Session-Information :

The output of sessionInfo() is placed here for reproducibility purposes.

```
## R version 4.0.5 (2021-03-31)
## Platform: x86_64-w64-mingw32/x64 (64-bit)
## Running under: Windows 10 x64 (build 19042)
##
## Matrix products: default
##
## locale:
## [1] LC_COLLATE=English_United States.1252
## [2] LC_CTYPE=English_United States.1252
## [3] LC_MONETARY=English_United States.1252
## [4] LC_NUMERIC=C
## [5] LC_TIME=English_United States.1252
```

```
##
## attached base packages:
## [1] stats     graphics  grDevices utils     datasets  methods   base
##
## other attached packages:
##  [1] knitr_1.33        xgboost_1.4.1.1   lubridate_1.7.10  data.table_1.14.0
##  [5] forcats_0.5.1     stringr_1.4.0     dplyr_1.0.5       purrr_0.3.4
##  [9] readr_1.4.0       tidyr_1.1.3       tibble_3.1.1      ggplot2_3.3.5
## [13] tidyverse_1.3.1
##
## loaded via a namespace (and not attached):
##  [1] tidyselect_1.1.0  xfun_0.22         lattice_0.20-41   haven_2.4.1
##  [5] colorspace_2.0-0  vctrs_0.3.7       generics_0.1.0    htmltools_0.5.1.1
##  [9] yaml_2.2.1        utf8_1.2.1        rlang_0.4.10      pillar_1.6.0
## [13] glue_1.4.2        withr_2.4.2       DBI_1.1.1         dbplyr_2.1.1
## [17] modelr_0.1.8      readxl_1.3.1      lifecycle_1.0.0   munsell_0.5.0
## [21] gtable_0.3.0      cellranger_1.1.0  rvest_1.0.0       evaluate_0.14
## [25] fansi_0.4.2       broom_0.7.6       Rcpp_1.0.6        scales_1.1.1
## [29] backports_1.2.1   jsonlite_1.7.2    fs_1.5.0          hms_1.0.0
## [33] digest_0.6.27     stringi_1.5.3     grid_4.0.5        cli_2.5.0
## [37] tools_4.0.5       magrittr_2.0.1    crayon_1.4.1      pkgconfig_2.0.3
## [41] Matrix_1.3-2      ellipsis_0.3.1    xml2_1.3.2        reprex_2.0.0
## [45] assertthat_0.2.1  rmarkdown_2.7     httr_1.4.2        rstudioapi_0.13
## [49] R6_2.5.0          compiler_4.0.5
```

1.4 Load the movilens data set :

To load the data set we run the code provided by Edx.

1.5 Dataset Description

We will use the 10M version of the MovieLens dataset. The 10 Millions dataset has been divided into two dataset: edx for training purpose and validation for the final validation phase.

The edx dataset contains approximately 9 Millions of rows, where every row represent a rating, with 69878 different users and 10677 movies.

We will focus in our exploration and analysis on Edx data set, the validation set has the same variables and will only be used in the final step.

2 Data Exploration :

2.1 Initial exploration :

The structure :
From the str() function we see the dimension, the names and class of the variables.

```
## Classes 'data.table' and 'data.frame':   9000055 obs. of  6 variables:
##  $ userId   : int  1 1 1 1 1 1 1 1 1 1 ...
##  $ movieId  : num  122 185 292 316 329 355 356 362 364 370 ...
##  $ rating   : num  5 5 5 5 5 5 5 5 5 5 ...
##  $ timestamp: int  838985046 838983525 838983421 838983392 838983392 838984474 838983653 838984885 83
##  $ title    : chr  "Boomerang (1992)" "Net, The (1995)" "Outbreak (1995)" "Stargate (1994)" ...
##  $ genres   : chr  "Comedy|Romance" "Action|Crime|Thriller" "Action|Drama|Sci-Fi|Thriller" "Action|Ac
##  - attr(*, ".internal.selfref")=<externalptr>
```

The six features of the data set are (same as validation) :
movieId: Unique ID for the movie.
userId: Unique ID for the user.
rating: A rating between 0 and 5 for the movie.
timestamp: Date and time the rating was given.
title: Movie title and Year the movie was released. (not unique).
genres: Genres associated with the movie.

Let's take a look on the first six rows of Edx data :

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

The Edx data contains 9000055 rows and 6 variables

Table 2: Number of rows and columns

rows	columns
9000055	6

Look for missing values :
we notice that there is no missing values in the edx dataset.

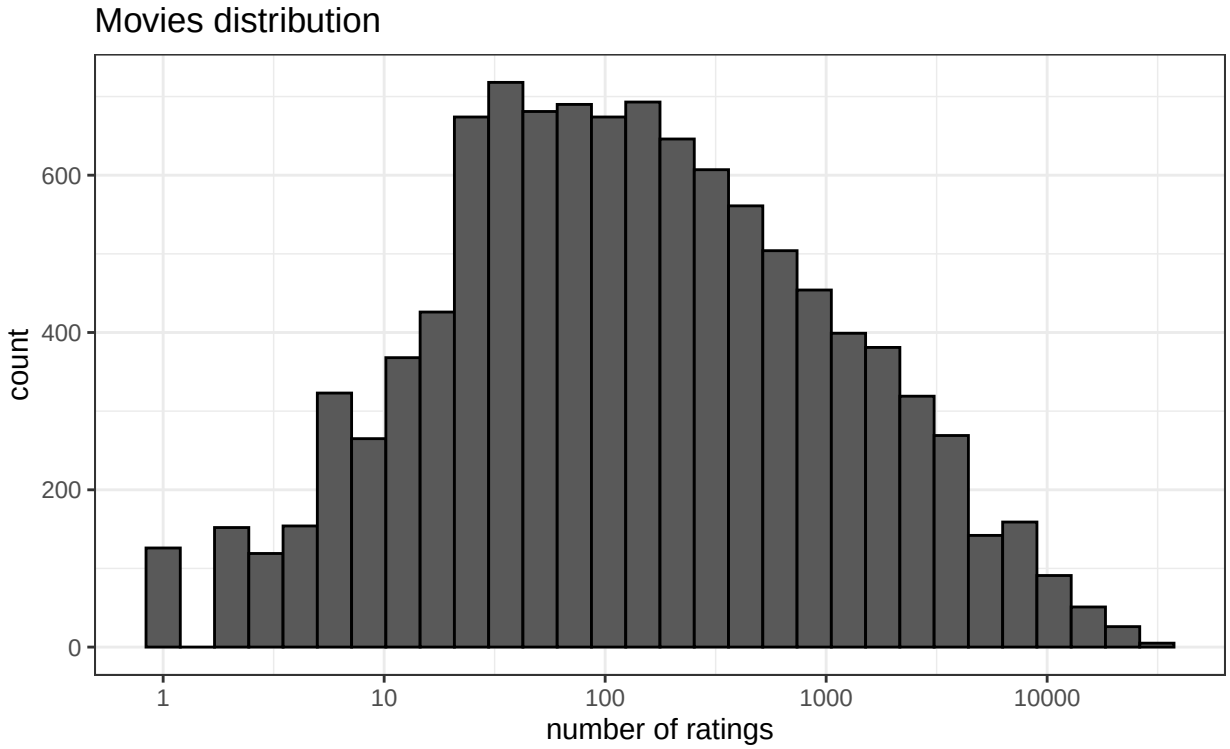
	Number of NAs
userId	0
movieId	0
rating	0
timestamp	0
title	0
genres	0

Number of movies and users :

Number of unique users	Number of unique movies
69878	10677

2.2 Variables Exploration

2.2.1 Movies :



We notice from the 10677 different movies in the edx set, some movies are rated more than the others, movies don't have the same popularity.

First six most rated movies

MovieID	Title	Number of ratings
296	Pulp Fiction (1994)	31362
356	Forrest Gump (1994)	31079
593	Silence of the Lambs, The (1991)	30382
480	Jurassic Park (1993)	29360
318	Shawshank Redemption, The (1994)	28015
110	Braveheart (1995)	26212

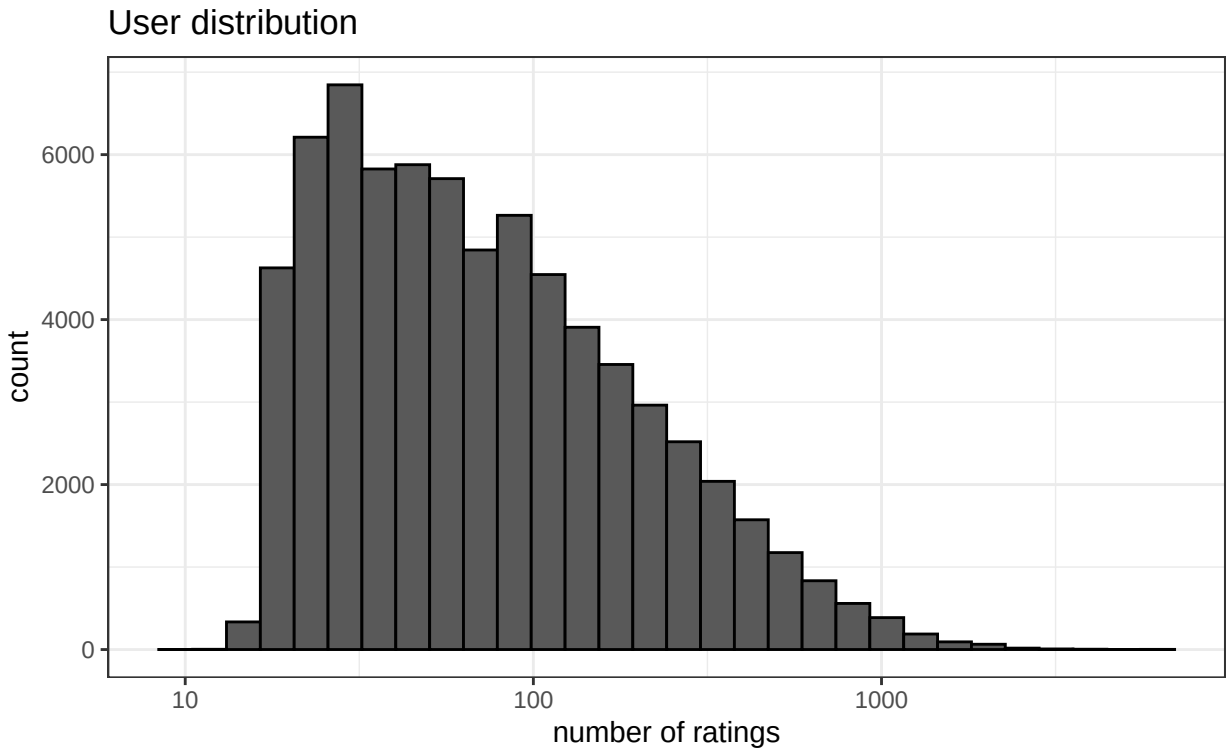
We notice that 90% of movies are rated at least 10 times:

movies which are rated at least 10 times
0.90128

Summary of rating per movies:
The average number of rating per movie is about 843.

Min.	1st Qu.	Median	Mean	3rd Qu.	Max.
1	30	122	842.94	565	31362

2.2.2 users



We notice that the user distribution is right skewed. Not every user is equally active. Some users have rated very few movie and others have rated a lot of movies.

Six users that rated the most :

UserID	Number of rating
59269	6616
67385	6360
14463	4648
68259	4036
27468	4023
19635	3771

Summary rating per users:
The average number rating per user is about 128 ratings

Min.	1st Qu.	Median	Mean	3rd Qu.	Max.
10	32	62	128.8	141	6616

95 % of the users rated more than 20 movies

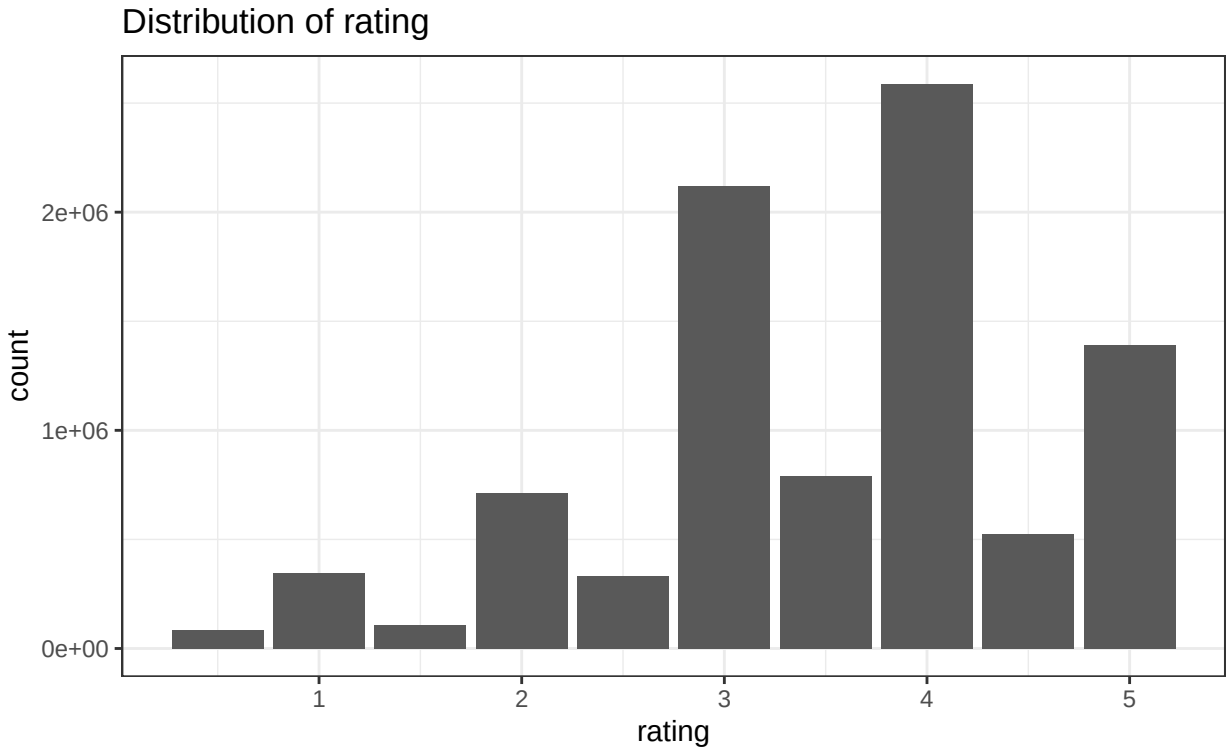
users who rated more than 20 movies
0.95034

2.2.3 Rating

Total number for each rating:

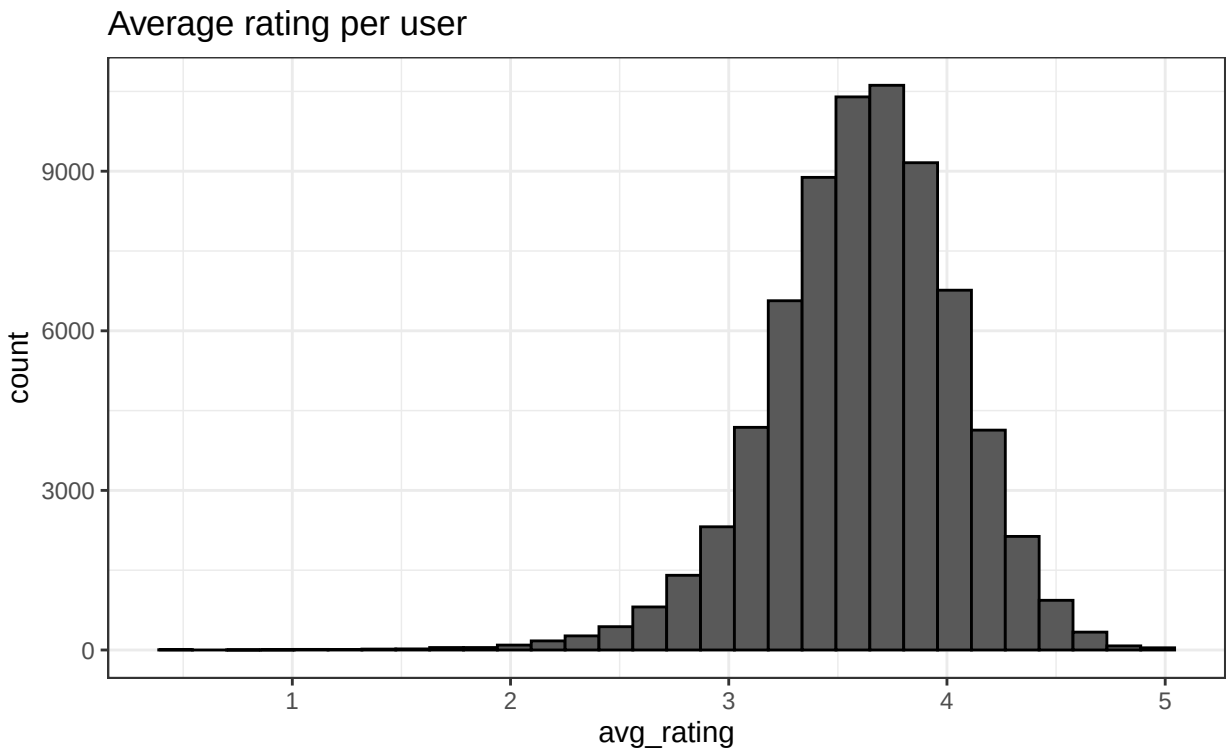
Rating	Total number
4.0	2588430
3.0	2121240
5.0	1390114
3.5	791624
2.0	711422
4.5	526736
1.0	345679
2.5	333010
1.5	106426
0.5	85374

From the 9000055 ratings in the edx data the rating range is between 0.5 and 5 with 10 possible choices.



We notice that some rating values are given more than the others, that most ratings get rounded value and that users have a general tendency to rate movies between 3 and 4.

Average rating per user :



The distribution is almost symmetric and users give on average rating of 3.6.

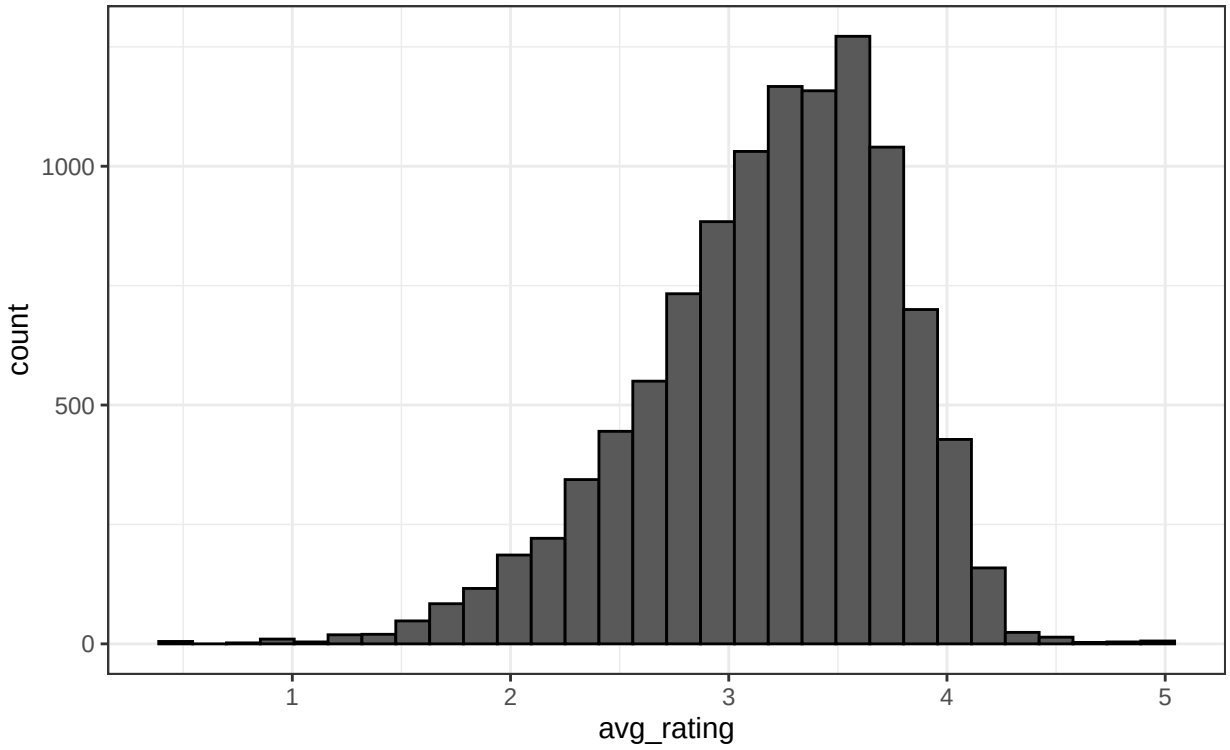
Summary of average rating per user :

Min.	1st Qu.	Median	Mean	3rd Qu.	Max.
0.5	3.3571	3.6351	3.6136	3.9026	5

93% of users have an average rating of at least 3

Users who give at least a 3 star
0.92817

Average rating per movie:



The distribution skewed to the left, this means that most movies get rating more than the average of average rating per movie.

Summary of average rating per movie

Min.	1st Qu.	Median	Mean	3rd Qu.	Max.
0.5	2.8443	3.2679	3.1917	3.6094	5

67% of movies have an average rating more than 3 stars :

Movie with average rating more than 3
0.6662

2.2.4 Date :

We will have to do some data processing to explore the date of release and date of rating.

2.2.4.1 Date of rating :
Convert timestamp to year of rating

That's how the data looks like now :

userId	movieId	rating	title	genres	rating_year
1	122	5	Boomerang (1992)	Comedy|Romance	1996
1	185	5	Net, The (1995)	Action|Crime|Thriller	1996
1	292	5	Outbreak (1995)	Action|Drama|Sci-Fi|Thriller	1996
1	316	5	Stargate (1994)	Action|Adventure|Sci-Fi	1996
1	329	5	Star Trek: Generations (1994)	Action|Adventure|Drama|Sci-Fi	1996
1	355	5	Flintstones, The (1994)	Children|Comedy|Fantasy	1996

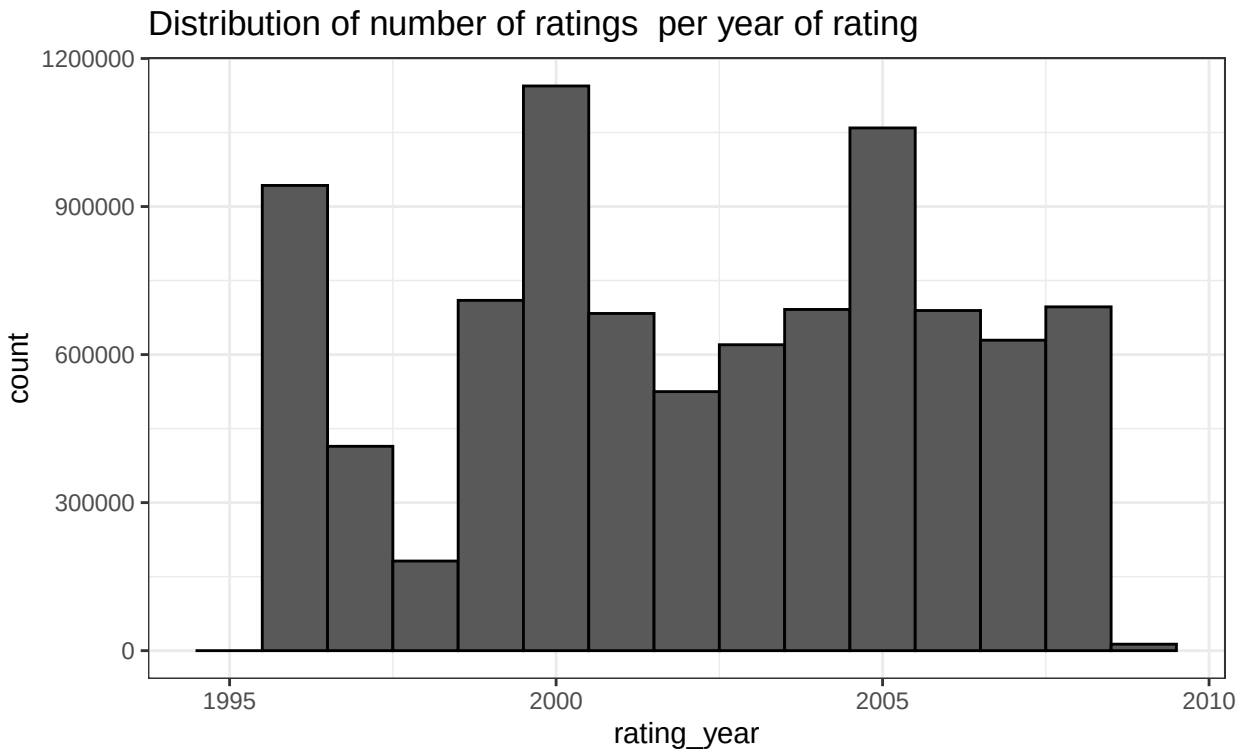
Summary of year of rating :

The rating starts in 1995, ends in 2009 and last more than 14 years.

Start	End	Period
1995	2009	14

The ten years with most number of ratings:

year of rating	number of ratings
1995	2
1996	942772
1997	414101
1998	181634
1999	709893
2000	1144349
2001	683355
2002	524959
2003	619938
2004	691429



We can see that the number of ratings per year has increased since 1995, has reached a maximum of 1144349 in 2002 and since then it has been around 60000.

2.2.4.2 Year of release:

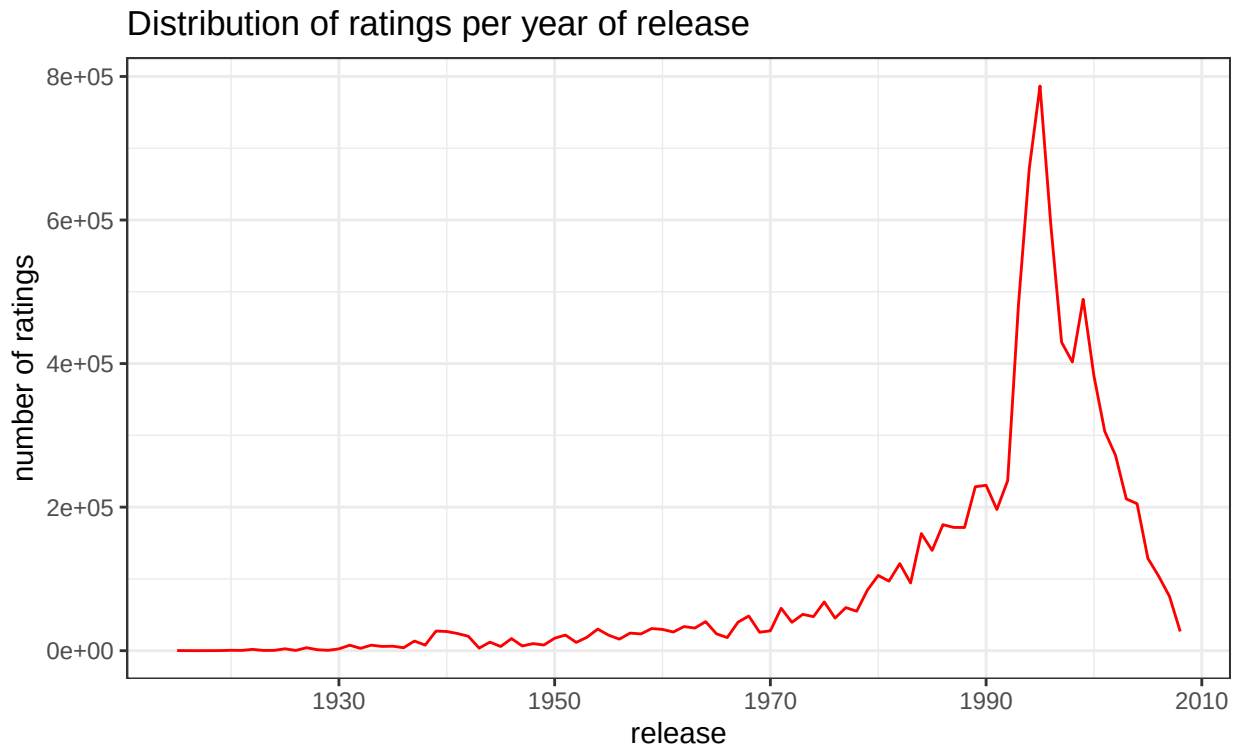
We'll extract the year of release from the title and take a look on the data :

userId	movieId	rating	title	release	genres	rating_year
1	122	5	Boomerang	1992	Comedy|Romance	1996
1	185	5	Net, The	1995	Action|Crime|Thriller	1996
1	292	5	Outbreak	1995	Action|Drama|Sci-Fi|Thriller	1996
1	316	5	Stargate	1994	Action|Adventure|Sci-Fi	1996
1	329	5	Star Trek: Generations	1994	Action|Adventure|Drama|Sci-Fi	1996
1	355	5	Flintstones, The	1994	Children|Comedy|Fantasy	1996

Summary year of relaese:

The first movie was released in 1915 and the last one 2008.

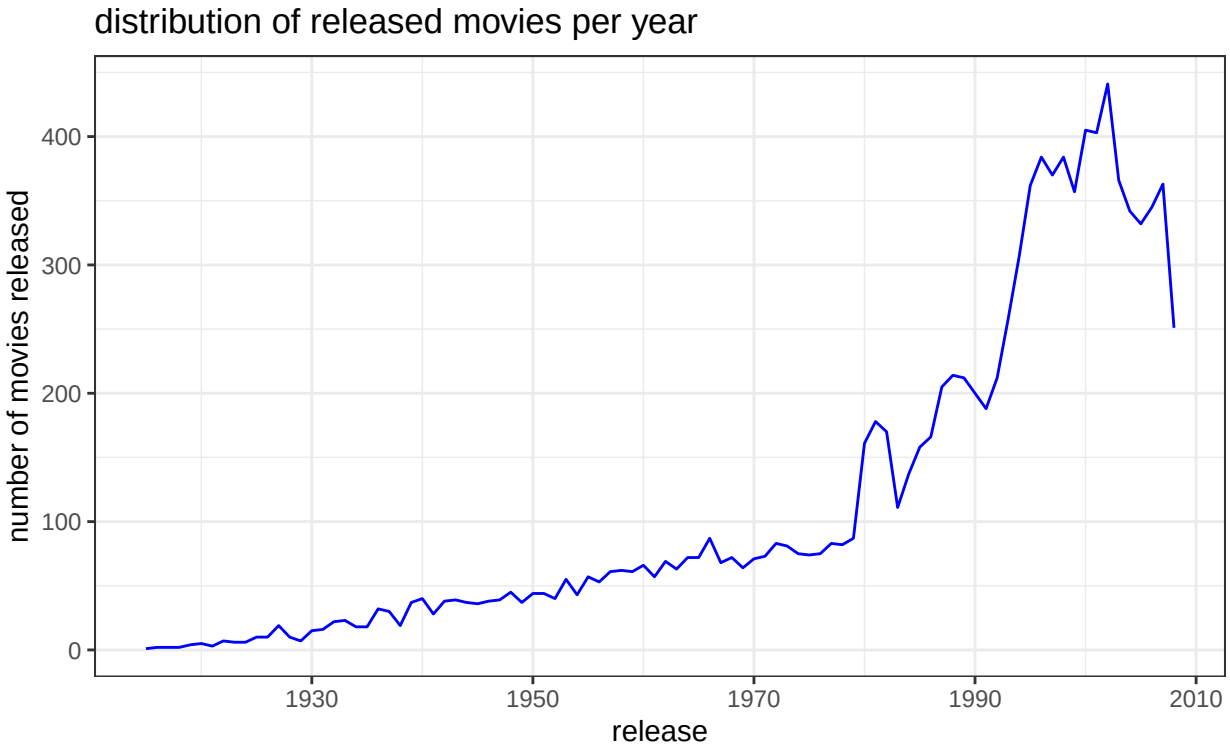
Start	End	period
1915	2008	93



The number of ratings has increased since the start of the movie making, it has reached its pic around 1995, but it has dropped since then.

The 10 years of release with the most rated movies:

year if release	number of rating
1995	786762
1994	671376
1996	593518
1999	489537
1993	481184
1997	429751
1998	402187
2000	382763
2001	305705
2002	272180



The number of movies released has increased since 1915 and it has reached the max in on 2002.

Number of released movies per year

The 10 years with most released movies :

release	n
2002	441
2000	405
2001	403
1996	384
1998	384
1997	370
2003	366
2007	363
1995	362
1999	357

2.2.5 Genre

Genres with most number of ratings (genre in attached format) :

genres	number of ratings
Drama	733296
Comedy	700889
Comedy|Romance	365468
Comedy|Drama	323637
Comedy|Drama|Romance	261425
Drama|Romance	259355

Number of genres combinations (genres attached) :

Number of genres
797

We will split the genres so as to have one genre in each row.

This is how the data looks now:

userId	movieId	rating	title	release	genres	rating_year
1	122	5	Boomerang	1992	Comedy	1996
1	122	5	Boomerang	1992	Romance	1996
1	185	5	Net, The	1995	Action	1996
1	185	5	Net, The	1995	Crime	1996
1	185	5	Net, The	1995	Thriller	1996
1	292	5	Outbreak	1995	Action	1996

Number of unique genres in our data :

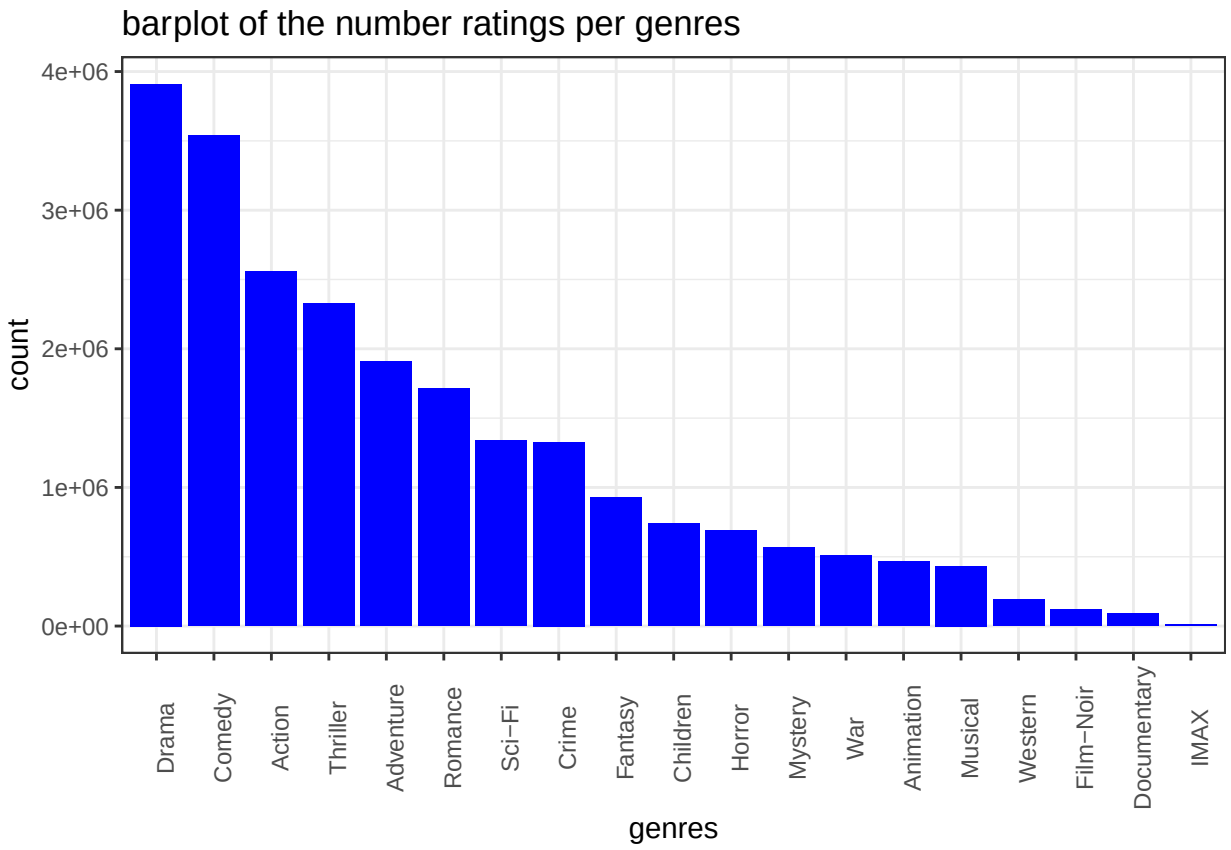
```
## The number of genres is 20
```

Number of ratings per each genre :

genres	number of ratings
Drama	3910127
Comedy	3540930
Action	2560545
Thriller	2325899
Adventure	1908892
Romance	1712100
Sci-Fi	1341183
Crime	1327715
Fantasy	925637
Children	737994
Horror	691485
Mystery	568332

genres	number of ratings
War	511147
Animation	467168
Musical	433080
Western	189394
Film-Noir	118541
Documentary	93066
IMAX	8181
(no genres listed)	7

We notice that the "no listed genre" is present just 8 times, compared to the other genrs it's almost insignificant, we can either assign it to the the most present genres (Drama and Comedy) or just remove it. We will remove it.



We notice that some genres get rated much more than the others, maybe it's because some genres are more popular than others. Including the genre in the analysis may give batter results.

3 Data preprocessing :

3.1 Data cleaning :

We will remove the title variable since it won't help us in our analysis.

The data is already cleaned, so now we will only encode genres to binary to make it suitable for some type of models.

3.2 Create train and test set :

The train set will be 80% of the edx data while the test set will be 20%.

First six rows and a subset of columns from the train set

userId	movieId	rating	Animation	Film-Noir	Western
1	185	5	0	0	0
1	316	5	0	0	0
1	329	5	0	0	0
1	355	5	0	0	0
1	364	5	1	0	0
1	377	5	0	0	0

First six rows and a subset of columns from the train set

userId	movieId	rating	Animation	Film-Noir	Western
1	122	5	0	0	0
1	292	5	0	0	0
1	356	5	0	0	0
1	362	5	0	0	0
1	370	5	0	0	0
1	466	5	0	0	0

4 Models building :

Machine learning algorithms in recommender systems typically fit into two categories: content-based systems and collaborative filtering systems. Modern recommender systems combine both approaches.

A) Content-based methods are based on the similarity of movie attributes. Using this type of recommender system, if a user watches one movie, similar movies are recommended.

B) With collaborative filtering, the system is based on past interactions between users and movies. With this in mind, the input for a collaborative filtering system is made up of past data of user interactions with the movies they watch.

In this project we will try some models starting with a naive approach based on the average rating, a 'content based' one based on the average movie, and start then using some collaborative methods. We will get the RMSE of the models on the test set. The one with the least RMSE will then be applied on the validation set to get the final RMSE result.

4.1 Naive model : just the mean

In this simple model we will make our predictions, by simply using the mean of all ratings as the estimate for every unknown rating.

The formula used for this model is :

$$Y_{u,i} = \hat{\mu} + \epsilon_{u,i}$$

with û the mean rating and upsilon a random error.

```
## The mean rating in the train set is 3.51236231036643
```

Table 29: Results

method	rmse
just the avg	1.0597

The rmse is 1.06 which is very big and that means that in average we are missing the true prediction by a whole star rating.

4.2 Movie effect model :

Some movies are rated more often than others. So in this model we will take into account the variability from movie to movie. A bias term b_i will be calculated for each movie to determine how much better or worse a given film is from the overall average.

$$Y_{u,i} = \hat{\mu} + b_i + \epsilon_{u,i}$$

Table 30: Results

method	rmse
just the avg	1.05968
movie_eff	0.94312

Using the movie effect the RMSE on the test set is better but still not good enough.

4.3 Movie plus User effect model :

Some users are positive and some have negative reviews because of their own personal liking/disliking regardless of movie. Users may have a tendency to rate movies higher or lower than the overall mean. So let's add this into the model. First we'll calculate the bias for each user. Then we'll combine the user bias with the movie bias to get the prediction.

$$Y_{u,i} = \hat{\mu} + b_i + b_u + \epsilon_{u,i}$$

Table 31: Results

method	rmse
just the avg	1.05968
movie_eff	0.94312
movie_user_eff	0.86547

With the movie_user combination the results are good, the rmse tells us that we are making good predictions, but can we do even better.

4.4 Movie, User and release year effect model :

The popularity of the movie genre depends strongly on the contemporary issues. So we should also add the release date bias b_l to our analysis.

$$Y_{u,i} = \hat{\mu} + b_i + b_u + b_l + \epsilon_{u,i}$$

Table 32: Results

method	rmse
just the avg	1.05968
movie_eff	0.94312
movie_user_eff	0.86547
movie_user_rel_eff	0.86513

The RMSE didn't change that much, let's add the year of rating effect.

4.5 Movie, User, release and rating year effect model :

The users mindset also has evolved over time. This can also effect the average rating of movies over the years. So we will add a rating date bias b_r to our model.

$$Y_{u,i} = \hat{\mu} + b_i + b_u + b_l + b_r + \epsilon_{u,i}$$

Table 33: Results

method	rmse
just the avg	1.05968
movie_eff	0.94312
movie_user_eff	0.86547
movie_user_rel_eff	0.86513
movie_user_year_eff	0.86507

We can see that using the average rating per year don't affect much the rmse.

4.6 Xgboost model : Extreme Gradient Boosting

XGBoost stands for "Extreme Gradient Boosting,it is an extension to gradient boosted decision trees (GBM) and specially designed to improve speed and performance.

XGBoost, Like other gradient boosting algorithms, operates on decision trees, models that construct a graph that examines the input under various "if" statements. Whether the "if" condition is satisfied influences the next "if" condition and eventual prediction. XGBoost progressively adds more and more "if" conditions to the decision tree to build a stronger model. XGBoost also works in a similar manner as Gradient Boosting model but introduces regularization terms to counter over_fitting..

4.6.1 Xgboost data processing :

From the last model we conclude that there is an impact of the movie effect and user effect on the rating, normally we would have encoded the user ID and movie ID to binaries, but this would have resulted in a very large data with thousands of columns since the number of movies and users is very large, so we won't be using movie and user ID in the model, but keep their effect we will add the averages of movies and users to their respective data sets (train and test) before deleting the ID's colums to let our model take advantage of these features.

Add averages to train set

The data will look like this (subset of the train set) : (The IDs will be dropped in the next step)

userId	movieId	rating	Animation	Film-Noir	Western	mov_avg	user_avg
1	185	5	0	0	0	3.1339	5
1	316	5	0	0	0	3.3481	5
1	329	5	0	0	0	3.3409	5
1	355	5	0	0	0	2.4817	5
1	364	5	1	0	0	3.7485	5

userId	movieId	rating	Animation	Film-Noir	Western	mov_avg	user_avg
1	377	5	0	0	0	3.5254	5

Add averages to test set

The data will look like this (subset of the test set) : (The IDs will be dropped in the next step)

userId	movieId	rating	Animation	Film-Noir	Western	mov_avg	user_avg
1	122	5	0	0	0	2.8124	5
1	292	5	0	0	0	3.4179	5
1	356	5	0	0	0	4.0294	5
1	362	5	0	0	0	3.4760	5
1	370	5	0	0	0	2.9863	5
1	466	5	0	0	0	2.9364	5

4.6.2 Define predictor and response variables in the datasets :

We remove non necessary features(ids), convert our train and test set predictors to an object data.matrix and we define the target variables using this piece of code :

```
train_x = data.matrix(train[, -c(1, 2, 3)])
train_y = train[,3]
test_x = data.matrix(test[, -c(1, 2, 3)])
test_y = test[, 3]
```

4.6.3 define final training and testing sets :

```
xgb_train = xgb.DMatrix(data = train_x, label = train_y)
xgb_test = xgb.DMatrix(data = test_x, label = test_y)
```

4.6.4 Define watchlist :

```
watchlist = list(train=xgb_train, test=xgb_test)
```

4.6.5 Fit XGBoost model and display training and testing data at each round to chose the best parameters :

```
## [1]  train-rmse:2.320063 test-rmse:2.317890
## [2]  train-rmse:1.738472 test-rmse:1.733779
## [3]  train-rmse:1.365744 test-rmse:1.358556
## [4]  train-rmse:1.139055 test-rmse:1.130040
## [5]  train-rmse:1.009342 test-rmse:0.999141
## [6]  train-rmse:0.939055 test-rmse:0.928067
## [7]  train-rmse:0.902358 test-rmse:0.891086
## [8]  train-rmse:0.883638 test-rmse:0.872262
## [9]  train-rmse:0.874126 test-rmse:0.862805
## [10] train-rmse:0.869237 test-rmse:0.858049
## [11] train-rmse:0.866703 test-rmse:0.855652
## [12] train-rmse:0.865289 test-rmse:0.854412
## [13] train-rmse:0.864444 test-rmse:0.853747
## [14] train-rmse:0.863876 test-rmse:0.853374
## [15] train-rmse:0.863457 test-rmse:0.853125
```

```
## [16] train-rmse:0.863190 test-rmse:0.852998
## [17] train-rmse:0.862925 test-rmse:0.852889
## [18] train-rmse:0.862675 test-rmse:0.852784
## [19] train-rmse:0.862529 test-rmse:0.852759
## [20] train-rmse:0.862422 test-rmse:0.852718
```

4.6.6 Define final model :

Using the optimal parameters from the Xgb model we define the final model.

4.6.7 Get predictions on test set :

Table 36: Results

method	rmse
just the avg	1.05968
movie_eff	0.94312
movie_user_eff	0.86547
movie_user_rel_eff	0.86513
movie_user_year_eff	0.86507
xgboost	0.85303

The rmse of the Xgboost is the best among all the models we have tried, so we will aplly it to the validation set.

5 Final results :

Apply the winning model for on validation set

Based on the results we have got in the previous section, we know that the Xgboost model did the best. In this final section we will apply it on the validation data set and get the final RMSE. Before this we will have to do some data pre_processing on the the validation data set to have it on the same structure as the training set.

5.1 Validation loading and cleaning :

There is no missing values in the validation data set.

```
##      userId             movieId            rating
## [1,] "| Number of NAs|" "| Number of NAs|" "| Number of NAs|"
## [2,] "|-------------:|" "|-------------:|" "|-------------:|"
## [3,] "|             0|" "|             0|" "|             0|"
##      timestamp          title              genres
## [1,] "| Number of NAs|" "| Number of NAs|" "| Number of NAs|"
## [2,] "|-------------:|" "|-------------:|" "|-------------:|"
## [3,] "|             0|" "|             0|" "|             0|"
```

We will extract the rating year, year of release and encode genres to binary in the validation data set. Then we will add the averages of rating per users and movies and remove the non needed columns.

5.2 Validation set preparation :

In this step we define the matrix of predictors and the target variable.

5.3 Run final model on validation set :

We run the final Xgboost model on the validation matrix.

```
## The RMSE of Xgboost applied on validation set is 0.841388939644233
```

The overall objective to training a machine learning algorithm of course resides in being able to generate predictions. The models used in this project have given different results with the Xgboost being the best but requiring more resources. The method based on the averages is the most light and doesn't consume much resources. After all the final model managed to perform a good result in the validation set.

6 Conclusion :

The project goal was to build a model that predict movie ratings. Our approach undertaken here relied on an initial exploratory analysis with data visualizations, then we start building models based on the assumptions made in the last part. Finally, we chose the model that performs the least RMSE and applied it on the validation set which produced an RMSE of 0.84 that is better than out target of 0.864.

While these results are promising, future work could improve both predictive performance and speed of the algorithm in the following ways :
-Use smooth function to model the date effect.
- Use matrix factorization.
- Use models that consume less time and memory.
- Use sampling techniques as it permit to work on large data set while guaranteeing generalizability.