Recommendation System - MovieLens Project

Luiz Guilherme Gomes Fregona

June, 2021

Contents

1. Introduction

In 2009, Netflix opened a public machine learning competition for the best recommendation system to predict user ratings for movies. In that competition, Netflix provided a dataset of 100,480,507 ratings that 480,189 users gave to 17,770 movies. For this project, we will be also creating a movie reccomendation system by using a small subset of the original data known as the "10M version of the MovieLens dataset". The data is open source, and can be found at the following website: https://grouplens.org/datasets/movielens/10m/

1.1. Problem Statement

- Three major characteristics must be taken into account while dealing with the present problem:
  - The first one is related to the dataset's nature, each outcome has a different set of predictors. Different users rate different numbers of movies and rate different movies as well. This is not usually found in machine learning problems.
  - The MovieLens dataset also reveals a myriad of biases, which are commonly presented in a movie review. Some bias has its origin at a personal level and usually are related to the background of each person. It means that reaching perfect accuracy would be impossible. However, major patterns can be found by studying major bias, and its effect on the overall accuracy of our model.
  - And finally, due to the extensive number of ratings, it is computationally expensive for most home computers to calculate complex algorithms. They will crackdown before being able to perform most machine learning algorithms.

1.2. Objetives

Considering all the issues pointed out in the previous topic, the goal of this project is to develop an algoritm that can predict ratings for movie i by user u efficiently, that takes into consideration most major bias, and uses the whole dataset as predictors for each rating estimated.

2. Pipeline

- The main steps in a data science project can change substantially between project, and it is always wise to define our goals and sources of information before setting it on stone. The pipeline decided for this project includes the following 4 steps:
  - 1º Data preparation: download, wrangle, and store the dataset(s) as a .rda file to be processed and analysed.
  - 2º Data analysis: explore the dataset in order to find any structure or meaninful relationship between features.
  - 3º Data preparation II: prepare the final dataset for modelling. In this step it is included:
    - data cleaning: get rid of unnecessary features for modeling.
    - data wrangling: format and name the dataset's colunms and rows with useful names.
  - 4º Modeling: create the model, test and validate it.
  - 5º Reporting: organize the finding in a clear .pdf and .html files.

3. Inicial Setup - Data Preparation

In this section we download and prepare the dataset for use in the data analysis step. The raw data were download from http://files.grouplens.org/datasets/movielens/ml-10m.zip, then separated into two objects: "ratings" and "movies". Afterwards, they were put together again using the "movieId" colunm from rating as reference. The new object "movielens" was then split into two parts, the training set called *edx* and the final evaluation set called *validation*.

3.1 Libraries

The following libraries were used in this project:

```
library(tidyverse)
library(caret)
library(data.table)
library(lubridate)
library(stringr)
```

3.2 Raw Data Loading

```
# MovieLens 10M dataset:
dl <- tempfile()
download.file("http://files.grouplens.org/datasets/movielens/ml-10m.zip", dl)
```

```
#Rating dataset
ratings <- fread(text = gsub("::", "\t", readLines(unzip(dl, "ml-10M100K/ratings.dat"))),
                 col.names = c("userId", "movieId", "rating", "timestamp"))
```

```
#Movies dataset
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 dataset - only movieIds from "ratings" were considered
movielens <- left_join(ratings, movies, by = "movieId")
```

3.3 Data Wrangling I

The following code generates the train and final hold-out test set. To do so, we randomly split the movielens set into two sets where 90% of it went to the training set (*edx*) and 10% to the evaluation set (*validation*). The validation set was used with only one purpose, evaluate our final reccomendation system.

```
# Validation set will be 10% of MovieLens data
set.seed(1, sample.kind="Rounding")
test_index <- createDataPartition(y = movielens$rating, times = 1, p = 0.1, list = FALSE)
```

```
#Segregate movielens dataset into edx and temp datasets
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)
edx <- rbind(edx, removed)
```

```
#Remove useless datasets and variables previously created
rm(dl, ratings, movies, test_index, temp, movielens, removed)
```

The code from all previous topics was provided by "HarvardX - Data Science: Capstone Project". It was not made by the author of this project.

3.4. Intermediate Dataset

In order to make the further analysis fast between days of work, the *edx* and *validation* set were saved as .rda files.

```
#Saving edx and validation set as .rda files
save(edx, file = "rda/edx.rda")
save(validation, file = "rda/validation.rda")
```

For data analysis, each time a day of work started, the *edx* dataset was loaded back to the system.

```
load("rda/edx.rda")
```

4. Data Analysis

"Explanation"

4.1. General properties

Before doing any advanced analysis, we need to undestand how the dataset is structured, and what are the predictors we are going to use. This information will help us build a better model.

```
#Dataset Dimensions
dim(edx)
```

```
## [1] 9000055       6
```

The edx dataset is composed by 6 colunms and 9000055 observations.

```
#Dataset Structure
str(edx, vec.len = 2)
```

```
## Classes 'data.table' and 'data.frame':   9000055 obs. of  6 variables:
##  $ userId   : int  1 1 1 1 1 ...
##  $ movieId  : num  122 185 292 316 329 ...
##  $ rating   : num  5 5 5 5 5 ...
##  $ timestamp: int  838985046 838983525 838983421 838983392 838983392 ...
##  $ title    : chr  "Boomerang (1992)" "Net, The (1995)" ...
##  $ genres   : chr  "Comedy|Romance" "Action|Crime|Thriller" ...
##  - attr(*, ".internal.selfref")=<externalptr>
```

- The six predictors are, respectively:
  - userId: integer - user identification
  - movieId: numeric - movie identification
  - rating: numeric - rating scores
  - timestamp: integer - seconds
  - title: character - title of movies followed by its release date
  - genres: character - multiple genres related to the movie

```
#Small sample of observations
head(edx)
```

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 dataset is organized in a tidy format, which means that each observation represent one rating of one movie by one user. Also, the predictor "timestamp" represent seconds since midnight UTC of January 1, 1970.

```
edx %>%
  summarise(n_users = n_distinct(userId),
            n_movies = n_distinct(movieId),
            n_rating = n_distinct(rating))
```

n_users	n_movies	n_rating
69878	10677	10

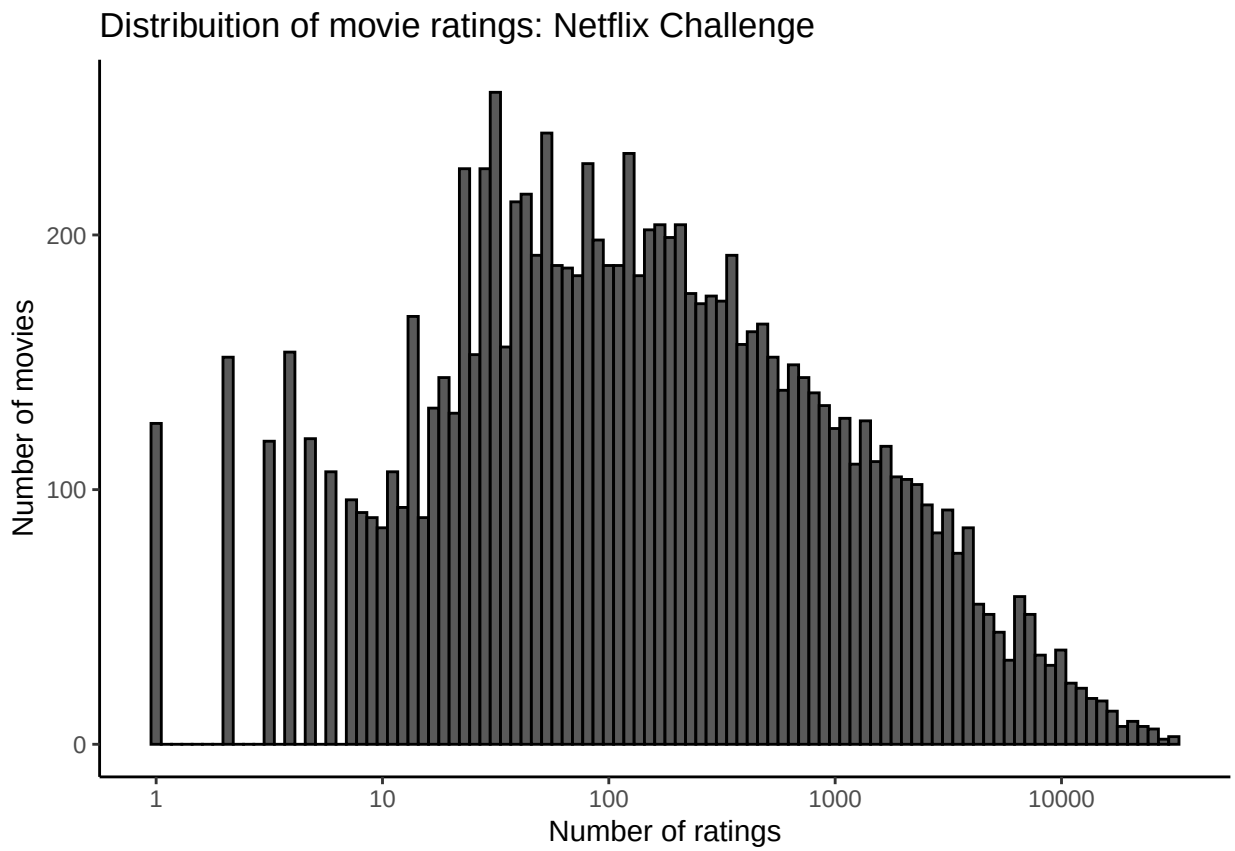
There are 69878 unique values of userIds, 10 different rating scores, and 10677 distinct movieIds. By doing a simple multiplication between number of unique users and movies, we can see that not all users rated all movies.

4.2. Data Exploration

4.2.1. Movies

The distribuition of some movies are more rated than others. It is obvious since there are blockbusters movies, which are watched by millions, and unsuccessful movies, not seen by a few.

```
#Number of times distinct movies were rated by users
edx %>%
  count(movieId) %>%
  ggplot(aes(n)) +
  geom_histogram(bins = 100, color = "black") +
  scale_x_log10() +
  ylab("Number of movies") +
  xlab("Number of ratings") +
  ggtitle("Distribuition of movie ratings: Netflix Challenge") +
  theme_classic()
```



The top 5 movies were rated more than 28.000 times, while the top least rated movies just once.

```
#Top 5 most rated movies
edx %>%
  group_by(title) %>%
  summarise(n=n()) %>%
  arrange(desc(n)) %>%
  top_n(5)
```

title	n
Pulp Fiction (1994)	31362
Forrest Gump (1994)	31079
Silence of the Lambs, The (1991)	30382
Jurassic Park (1993)	29360
Shawshank Redemption, The (1994)	28015

```
#Top 5 least rated movies
edx %>%
  group_by(title) %>%
  summarise(n=n()) %>%
  arrange(desc(n)) %>%
  slice_tail(n=5)
```

title	n
When Time Ran Out… (a.k.a. The Day the World Ended) (1980)	1
Where A Good Man Goes (Joi gin a long) (1999)	1
Won't Anybody Listen? (2000)	1
Young Unknowns, The (2000)	1
Zona Zamfirova (2002)	1

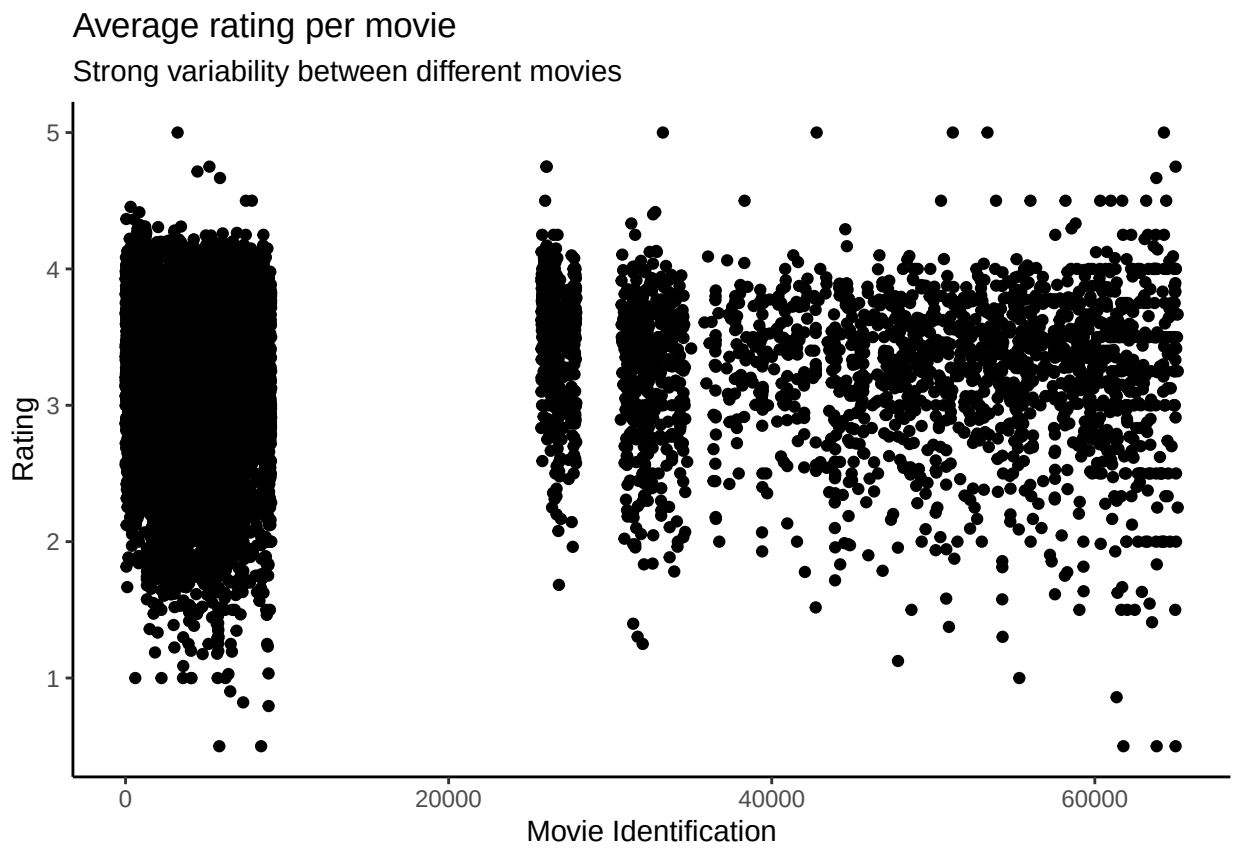
Most of the films are not rated by more than 1% of all users, and just a few get to be rated by more than 10% of all users.

```
#Proportion of users who rated a specific movie
edx %>%
  count(movieId) %>%
  mutate(p_users = case_when(
    n <= 700 ~ "less than 1%",
    n >= 5000 ~ "more than 10%",
    TRUE ~ "between 1% and 10%"
  )) %>%
  count(p_users)
```

p_users	n
between 1% and 10%	1955
less than 1%	8303
more than 10%	419

We can also see that the average rating of each movie varies substantially, and that rating between 2 and 4 are prevalent. That's a important caracteristic in our dataset, and need to be adressed in order to make better rating predictions.

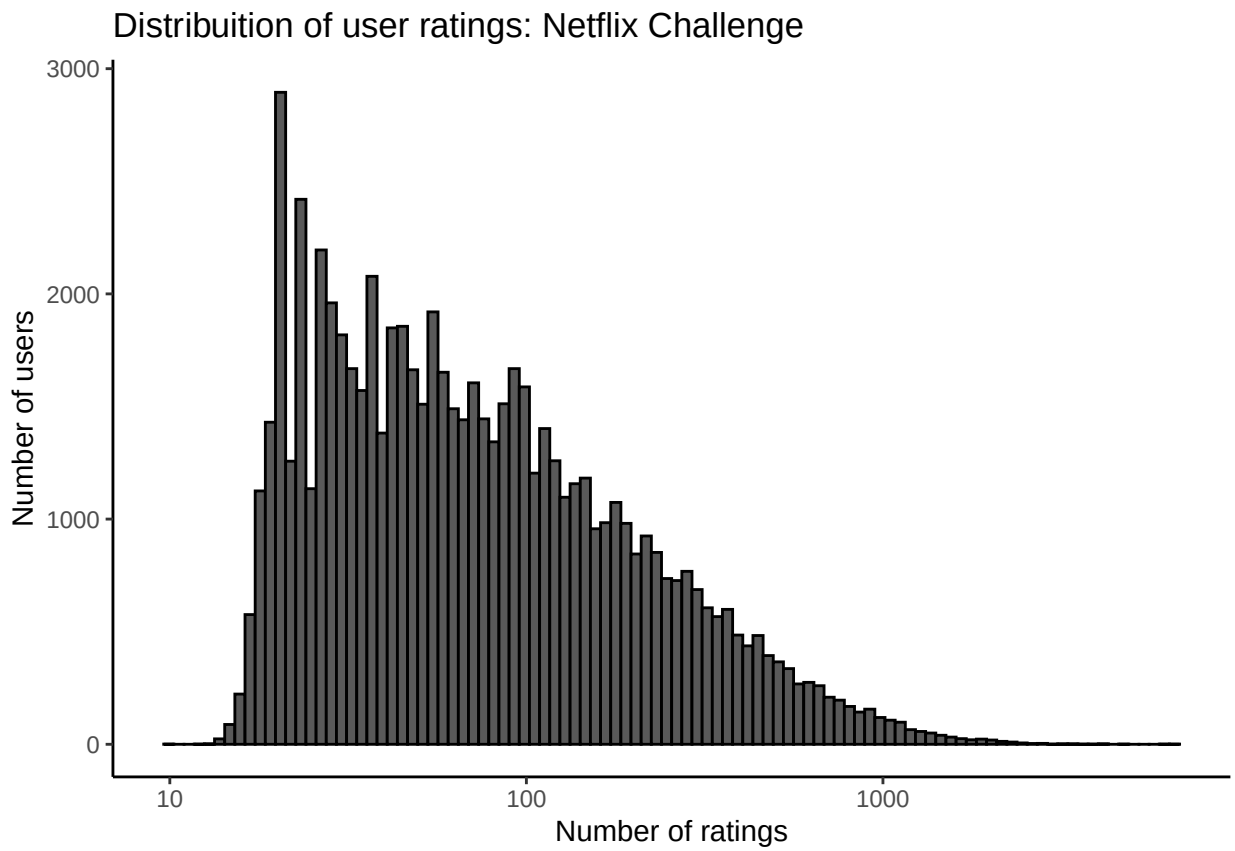
```
edx %>%
  group_by(movieId) %>%
  summarise(avg = mean(rating)) %>%
  ggplot(aes(movieId, avg)) +
  geom_point() +
  theme_classic() +
  ylab("Rating") +
  xlab("Movie Identification") +
  ggtitle("Average rating per movie", subtitle = "Strong variability between different movies")
```



4.2.2.Users

The distribuition . We can see that some users are more active than other.

```
edx %>%
  count(userId) %>%
  ggplot(aes(n)) +
  geom_histogram(bins = 100, color = "black") +
  scale_x_log10() +
  ylab("Number of users") +
  xlab("Number of ratings") +
  ggtitle("Distribuition of user ratings: Netflix Challenge") +
  theme_classic()
```



The top 5 users had rated more than 4.000 times, while the top least active users less than 15 times.

```
#Top 5 most active users
edx %>%
  group_by(userId) %>%
  summarise(n=n()) %>%
  arrange(desc(n)) %>%
  top_n(5)
```

userId	n
59269	6616
67385	6360
14463	4648
68259	4036
27468	4023

```
#Top 5 least active users
edx %>%
  group_by(userId) %>%
  summarise(n=n()) %>%
  arrange(desc(n)) %>%
  slice_tail(n=5)
```

userId	n
71344	14
15719	13
50608	13
22170	12
62516	10

4.2.3. Rating

5. Data Preparation II

The *edx* set was again divide into train and test set, in order to perform cross-validation.

6. Modeling

6.1. Evaluation Metric

6.2. Baseline Model

Model 1 - Timestamp

Model 2 - Timestamp + Genres

Model 3 - Timestamp + Genres + Movie Effect

Model 4 - Timestamp + Genres + Movie Effect + User's Effect

Model 5 - All + Regularization

Model 6 - All + Matrix Factorization

Model 7 - Knn + Regularization

Model 8 - Ensembles

Conclusion