HarvardX Capstone Project - MovieLens Prediction

Marcelo Argotti

6/20/2020

Contents

1. Introduction

1.1 Overview/Executive Summary

The following project is focused on the theme of recommendation systems, with the objective of predicting the ratings that users will attribute to a particular movie. The recommendation system will take into consideration the users' ratings to give specific suggestions for future movies. Those with the highest predicted ratings will then be recommended to the user.

The use of such predictive models is of practical relevance and was popularized by The "Netflix Challenge" that evaluated the quality of different proposed algorithms using the "typical error": the Root Mean Square Error (RMSE).

The same method will be used to determine the strength of a variety of prediction models using the 10M version of the MovieLens dataset. The goal will be to train a machine learning algorithm using the inputs from one subset (named `edx`) of the database to predict movie ratings in the `validation` set.

```
## load packages and libraries needed
# Note: this process could take a couple of minutes
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(data.table)) install.packages("data.table",
                                          repos = "http://cran.us.r-project.org")
if(!require(ggplot2)) install.packages("ggplot2",
                                       repos = "http://cran.us.r-project.org")
if(!require(lattice)) install.packages("lattice",
                                       repos = "http://cran.us.r-project.org")
if(!require(knitr)) install.packages("knitr",
                                     repos = "http://cran.us.r-project.org")
if(!require(kableExtra)) install.packages("kableExtra",
                                          repos = "http://cran.us.r-project.org")
if(!require(dplyr)) install.packages("dplyr",
                                     repos = "http://cran.us.r-project.org")
if(!require(lubridate)) install.packages("lubridate",
                                         repos = "http://cran.us.r-project.org")
if(!require(Matrix)) install.packages("Matrix",
                                      repos = "http://cran.us.r-project.org")
if(!require(recommenderlab)) install.packages("recommenderlab",
                                              repos = "http://cran.us.r-project.org")
```

1.2 Loading the Data

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


# 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(movieId),title = as.character(title),
         genres = as.character(genres))

movielens <- left_join(ratings, movies, by = "movieId")
```

```r
# 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
```

```r
# 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)
edx <- rbind(edx, removed)

rm(dl, ratings, movies, test_index, temp, movielens, removed)
```

2. Overview - Data Observation

First, let's take a glimpse of the edx and validation data set. * **Data structure of the training set (edX dataset)**

```r
class(edx)
```

```
## [1] "data.table" "data.frame"
```

```r
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, 1, ...
## $ movieId   <dbl> 122, 185, 292, 316, 329, 355, 356, 362, 364, 370, 377, 42...
## $ 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, 83...
## $ title     <chr> "Boomerang (1992)", "Net, The (1995)", "Outbreak (1995)",...
## $ genres    <chr> "Comedy|Romance", "Action|Crime|Thriller", "Action|Drama|...
```

- **Data structure of the test set (validation dataset)**

```r
class(validation)
```

```
## [1] "data.table" "data.frame"
```

```
glimpse(validation)
```

```
## 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   <dbl> 231, 480, 586, 151, 858, 1544, 590, 4995, 34, 432, 434, 8...
## $ 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....
## $ timestamp <int> 838983392, 838983653, 838984068, 868246450, 868245645, 86...
## $ title     <chr> "Dumb & Dumber (1994)", "Jurassic Park (1993)", "Home Alo...
## $ genres    <chr> "Comedy", "Action|Adventure|Sci-Fi|Thriller", "Children|C...
```

We can see that the edX dataset is made of 6 variables for a total of about 9,000,055 observations. Same as our training set (edx dataset) the validation set contains the same 6 features, but with a total of 999,999 occurences.

The outcome we want to predict is the 'rating' feature. Taking into account both datasets, here are their characteristics.

Quantitative features	Qualitative features
userID (numerical, discrete)	title(categorical, nominal)
movieID (numerical, discrete)	genres (categorical, nominal)
timestamp (data, discrete)	
rating (numerical, continuous)	

Let's take a look at the ratings (our dependent value, y) feature in order to count the occurrences of each of them.

```
unique(edx$rating)
```

```
##  [1] 5.0 3.0 2.0 4.0 4.5 3.5 1.0 1.5 2.5 0.5
```

It can be seen that there are only 10 continuous values of rating (in the rage of 0.5 to 5). Also, no user gives 0 as rating.

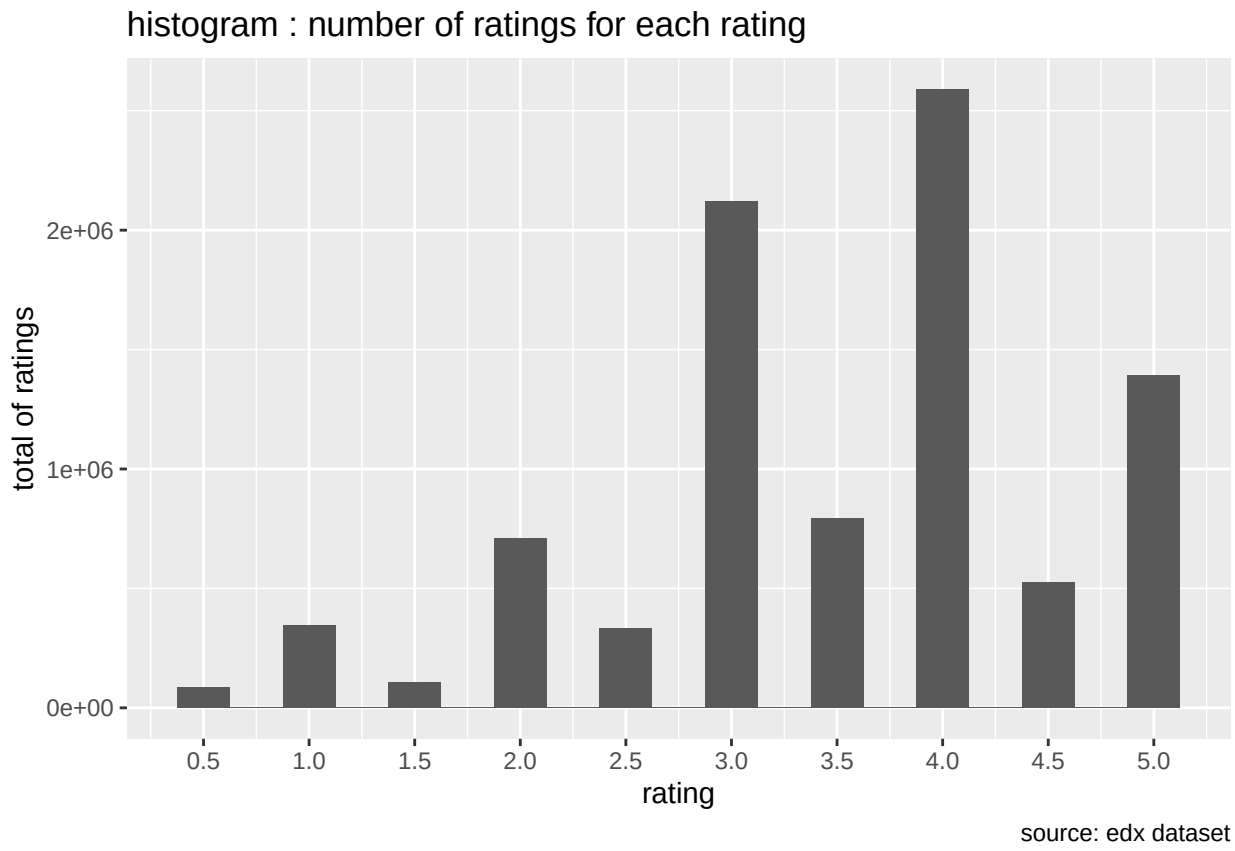
Now we will display and plot the ratings distribution of our training set.

```
#Rating plot distribution of our training set
ratings_distribution <- edx %>% group_by(rating) %>%
  summarize(ratings_sum = n()) %>% arrange(desc(ratings_sum))
ratings_distribution
```

```
## # A tibble: 10 x 2
##    rating ratings_sum
##     <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
```

```r
ggplot(edx, aes(x= edx$rating, fill = edx$rating)) +
  geom_histogram( binwidth = 0.25) +
  scale_x_continuous(breaks=seq(0, 5, by= 0.5)) +
  labs(x="rating", y="total of ratings", caption = "source: edx dataset") +
  ggtitle("histogram : number of ratings for each rating")
```



Most of the ratings are above 3.5 and the most common is 4. The histogram shows that half star ratings are less common than whole star ratings.

Now we are going to explore the **qualitative features** (genres and titles) of the edX dataset.

```r
# I tried to create a data frame that includes each single genre without their combinations using he 's

#We make a copy of the training dataset due to we want to build a matrix
#containing some features as factors. Also, we  want to observe unique
#or independent genres without been combined, in order to visualize which
#genre has more ratings, get a better understanding of the edx dataset,
#and identify outliers.
edx_copy <- edx
```

```r
genres <- edx_copy$genres %>% str_split(pattern = "\\|")
genres_unique <- genres %>% unlist() %>% unique()
genres_unique
```

```
##  [1] "Comedy"             "Romance"            "Action"
##  [4] "Crime"              "Thriller"           "Drama"
##  [7] "Sci-Fi"             "Adventure"          "Children"
## [10] "Fantasy"            "War"                "Animation"
## [13] "Musical"            "Western"            "Mystery"
## [16] "Film-Noir"          "Horror"             "Documentary"
## [19] "IMAX"               "(no genres listed)"
```

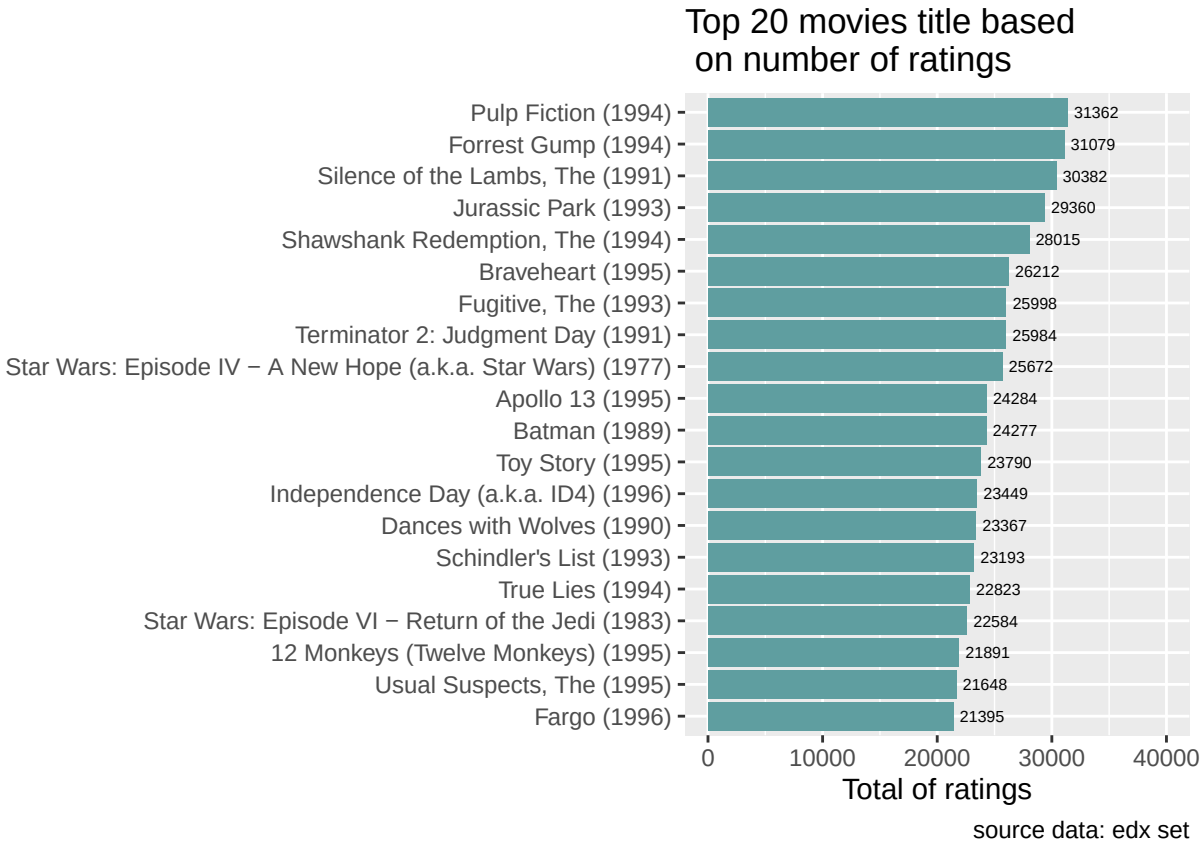
```r
# The following lines of code define a table containing independent genres
#as variables, each one of them are described only by 0s and 1s. The 0s will
#be treated as non genre matching the corresponding movie.
cols_edxcopy <- ncol(edx_copy)
for (i in seq_along(genres_unique)) {
  id <- grepl(pattern = genres_unique[i], edx_copy$genres)
  edx_copy[[cols_edxcopy + i]] <- 0
  edx_copy[[cols_edxcopy + i]][id] <- 1
}
names(edx_copy)[(cols_edxcopy + 1):ncol(edx_copy)] <- genres_unique


#After separating genres and transforming them into individual variables, I write a simple summary of h
topgen <- edx_copy %>% summarise(Comedy= sum(Comedy), Romance=sum(Romance),
                                 Action=sum(Action), Crime=sum(Crime),
                                 Thriller=sum(Thriller), Drama=sum(Drama),
                                 Sci_fi=sum(`Sci-Fi`),Adventure=sum(Adventure),
                                 Children=sum(Children), Fantasy=sum(Fantasy),
                                 War=sum(War), Animation=sum(Animation),
                                 Musical=sum(Musical), Western=sum(Western),
                                 Mystery=sum(Mystery), Film_noir=sum(`Film-Noir`),
                                 Horror=sum(Horror), Documentary=sum(Documentary),
                                 Imax=sum(IMAX),)
glimpse(sort(topgen))
```

```
## Rows: 1
## Columns: 19
## $ Imax        <dbl> 8181
## $ Documentary <dbl> 93066
## $ Film_noir   <dbl> 118541
## $ Western     <dbl> 189394
## $ Musical     <dbl> 433080
## $ Animation   <dbl> 467168
## $ War         <dbl> 511147
## $ Mystery     <dbl> 568332
## $ Horror      <dbl> 691485
## $ Children    <dbl> 737994
## $ Fantasy     <dbl> 925637
## $ Crime       <dbl> 1327715
## $ Sci_fi      <dbl> 1341183
```

```
## $ Romance     <dbl> 1712100
## $ Adventure   <dbl> 1908892
## $ Thriller    <dbl> 2325899
## $ Action      <dbl> 2560545
## $ Comedy      <dbl> 3540930
## $ Drama       <dbl> 3910127
```

The Drama, Comedy and Action genres has the top 3 movie ratings.

```
# We move on with the title features.
top_title <- edx %>% group_by(title) %>%
  summarize(count=n()) %>% top_n(20,count) %>%
  arrange(desc(count))

top_title %>%
  ggplot(aes(x=reorder(title, count), y=count)) +
  geom_bar(stat='identity', fill="cadetblue") + coord_flip(y=c(0, 40000)) +
  labs(x="", y="Total of ratings") +
  geom_text(aes(label= count), hjust=-0.15, size=2) +
  labs(title="Top 20 movies title based \n on number of ratings" ,
       caption = "source data: edx set")
```




```
# With the kable and head function we generate a simple table that contains
#the top 30 rated movies, including the genres associated to each one.
#It can be seen that the first 25 movies have been rated over 20.000
```

```
kable(head(edx %>% group_by(title,genres) %>%
             summarize(count=n()) %>%
             top_n(20,count) %>%
             arrange(desc(count)) ,30)) %>%
  kable_styling(bootstrap_options = "bordered",
                full_width = FALSE ,position ="center") %>%
  column_spec(1,bold = TRUE ) %>%
  column_spec(2,italic = TRUE) %>%
  column_spec(3,color = "brown")
```

title	*genres*
Pulp Fiction (1994)	*Comedy|Crime|Drama*
Forrest Gump (1994)	*Comedy|Drama|Romance|W*
Silence of the Lambs, The (1991)	*Crime|Horror|Thriller*
Jurassic Park (1993)	*Action|Adventure|Sci-Fi|Th*
Shawshank Redemption, The (1994)	*Drama*
Braveheart (1995)	*Action|Drama|War*
Fugitive, The (1993)	*Thriller*
Terminator 2: Judgment Day (1991)	*Action|Sci-Fi*
Star Wars: Episode IV - A New Hope (a.k.a. Star Wars) (1977)	*Action|Adventure|Sci-Fi*
Apollo 13 (1995)	*Adventure|Drama*
Batman (1989)	*Action|Crime|Sci-Fi|Thrille*
Toy Story (1995)	*Adventure|Animation|Child*
Independence Day (a.k.a. ID4) (1996)	*Action|Adventure|Sci-Fi|W*
Dances with Wolves (1990)	*Adventure|Drama|Western*
Schindler's List (1993)	*Drama|War*
True Lies (1994)	*Action|Adventure|Comedy|*
Star Wars: Episode VI - Return of the Jedi (1983)	*Action|Adventure|Sci-Fi*
12 Monkeys (Twelve Monkeys) (1995)	*Sci-Fi|Thriller*
Usual Suspects, The (1995)	*Crime|Mystery|Thriller*
Fargo (1996)	*Comedy|Crime|Drama|Thri*
Speed (1994)	*Action|Romance|Thriller*
Aladdin (1992)	*Adventure|Animation|Child*
Matrix, The (1999)	*Action|Sci-Fi|Thriller*
Star Wars: Episode V - The Empire Strikes Back (1980)	*Action|Adventure|Sci-Fi*
Seven (a.k.a. Se7en) (1995)	*Crime|Horror|Mystery|Thri*
American Beauty (1999)	*Drama*
Raiders of the Lost Ark (Indiana Jones and the Raiders of the Lost Ark) (1981)	*Action|Adventure*
Back to the Future (1985)	*Adventure|Comedy|Sci-Fi*
Mission: Impossible (1996)	*Action|Adventure|Mystery|*
Ace Ventura: Pet Detective (1994)	*Comedy*

Now we are going to explore the **quantitative features** (userId, movieId, timestamp) of the edX dataset. First, we are going to display the distinct number of users and distinct number of movies of our training set.

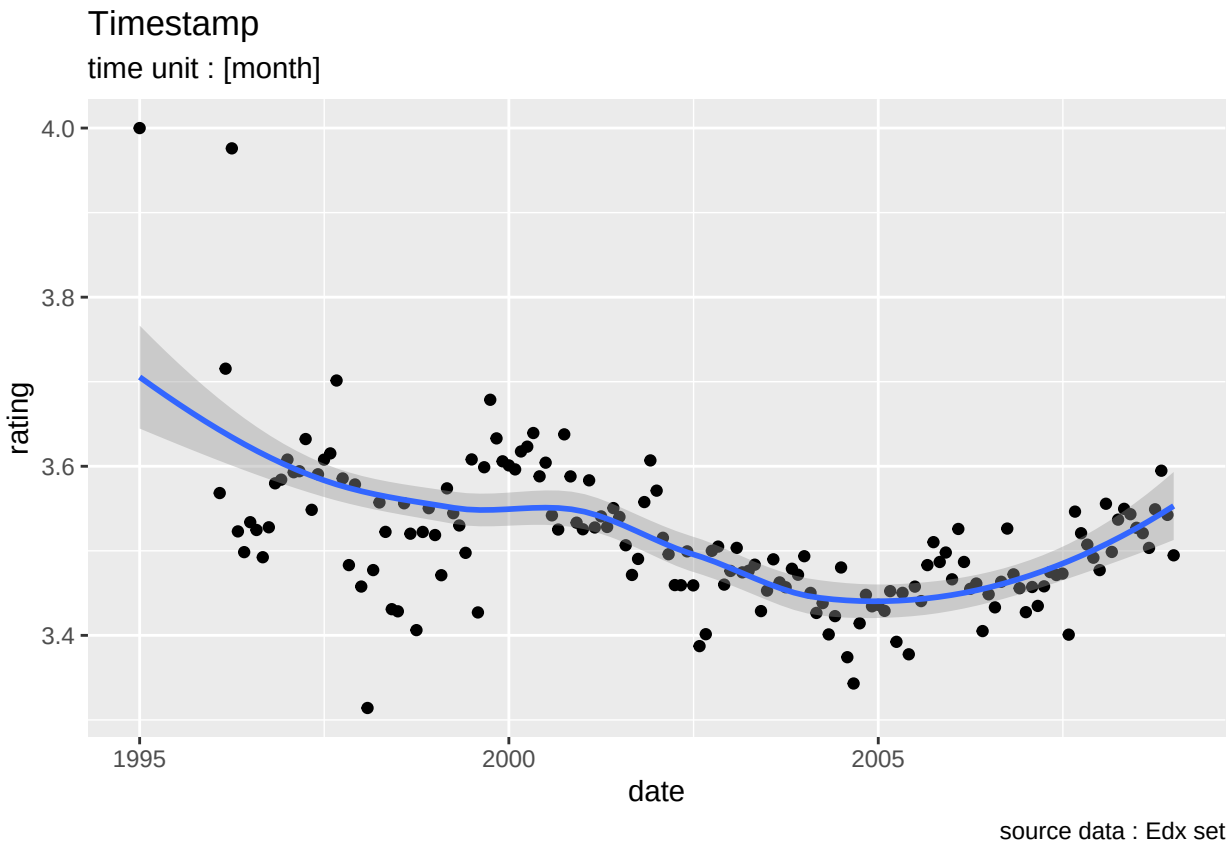
```
edx %>% summarize(distinct_users=n_distinct(userId),
                  distinct_movies=n_distinct(movieId))
```

```
##   distinct_users distinct_movies
## 1          69878           10677
```

The number of unique values for the userId variable is 69878 and 10677 for the movieId. It is clear that there are less movies provided for ratings that users who rated them.

In order to analyze for correlations we transform the 'timestamp' variable which represents the time in which the rating was done. The units of measurement are expressed in seconds. Thanks to the 'as_datetime' function we can transform each timestamp in the desired time format. Subsequently we create a scatter point plot of average ratings vs date. Also we make use of smooth geom in order to identify outliers or overplotting.

```
# Lets build a scatter point plot of average ratings vs date.
# we make use of smooth geom in order to identify outliers or overplotting.
edx %>%
  mutate(date = round_date(as_datetime(timestamp), unit = "month")) %>%
  group_by(date) %>% summarize(rating = mean(rating)) %>%
  ggplot(aes(date, rating)) + geom_point() + geom_smooth() +
  ggtitle("Timestamp")+
  labs(subtitle = "time unit : [month]",
       caption = "source data : Edx set")
```



Analyzing the trend we observe that the oldest the movie the highest the rating it receives. This may be due to the old movies are consider classics and they are rated better by the audience.

Now that we have a good insight of our datasets we can choose the appropriate algorithms and proceed with the models training.

In order to improve the results from our algorithms we will create a new dataframe where usersId and movieId will be treated as factors for some analysis purposes. We already have made a copy of our edx dataset, since we want to use more features, improve our model and keep unchanged our original training set (edx).

3. Data Preprocessing

Data preprocessing (e.g. filtered, transformed, cleansed, etc.) is an important step for any machine learning data analysis problem. Data preprocessing techniques include similarity measurements (such as Euclidian distance, cosine distance and Pearson coefficient), dimensionality-reduction, such as Principal Component Analysis (PCA) and Singular Value Decomposition (SVD).

As we mentioned earlier there are 2 main types of recommender systems. Content-based recommender systems (based on the description of an item) and Collaborative filtering systems (based on the similarity between items or users and finding the minimum).

We are going to use the collaborative filtering system by using the **Cosine similarity** which measure the distance between two vectors of an inner product space that measures the cosine of the angle between them. According to Gorakala and Usuelli (2015) it is defined as follow:

$similarity = cos(A, B) = A * B/||A|| * ||B||$

```
#Lets build a rating matrix using the SparseMatrix function
#We make a new copy of the edx dataset in order to preserve the organl
# class of userId and movieId that will be converted into numeric vectors

edx_copy2 <- edx

edx_copy2$userId <- as.factor(edx_copy2$userId)
edx_copy2$movieId <- as.factor(edx_copy2$movieId)

edx_copy2$userId <- as.numeric(edx_copy2$userId)
edx_copy2$movieId <- as.numeric(edx_copy2$movieId)

scattered_ratings <- sparseMatrix(i = edx_copy2$userId,
                                  j = edx_copy2$movieId,
                                  x = edx_copy2$rating,
                                  dims = c(length(unique(edx_copy2$userId)),
                                           length(unique(edx_copy2$movieId))),
                                  dimnames = list(paste(
                                    "user", 1:length(unique(edx_copy2$userId)),
                                    sep = ""),
                                    paste("movie", 1:length(unique(
                                      edx_copy2$movieId)), sep = "")))

rm(edx_copy2)
#Lets give a look on the first 30 users
scattered_ratings[1:30,1:30]
```

```
## 30 x 30 sparse Matrix of class "dgCMatrix"
##
## user1  . .   . . . . .   . . . . . . . . . . . . . .   .   . .   . . . . . . . .
## user2  . .   . . . . .   . . . . . . . . . . . . . .   .   . .   . . . . . . . .
## user3  . .   . . . . .   . . . . . . . . . . . . . .   .   . .   . . . . . . . .
```

```
## user4  . .   . . . . .   . . . . . . . . . . . . . .   .   3 .   . . . . . . . .
## user5  1 .   . . . . 3.0 . . . . . . . . . . . . . .   .   . .   . . 3 . . 3 . 5
## user6  . .   . . . . .   . . . . . . . . . . . . . .   .   . .   . . . . . . . .
## user7  . .   . . . . .   . . . . . . . . . . . . . .   .   . .   . . . . . . . .
## user8  . 2.5 . . 3 4 .   . . . . . . . . . 3 . . 3.5 .   . 2.5 . . . . . . . .
## user9  . .   . . . . .   . . . . . . . . . . . . . .   .   . .   . . . . . . . .
## user10 . .   . . . . 3.0 . . . . . . . . . 4 . . .   5 .   . . 3 . . . . .
## user11 . .   . . . . .   . . . . . . . . . . . . . .   .   . .   . . . . . . . .
## user12 . .   . . . . .   . . . . . . . . . . . . . .   .   . .   . . . . . . . .
## user13 . 3.0 . . . . .   . . . . . . . . . . . . . .   3.0 . .   4 . . . . . . .
## user14 3 3.0 . . . . .   . . . . . . . . . . . . . .   .   . .   . . . . . . . .
## user15 . .   . . . . .   . . . . . . . . . . . . . .   .   . .   . . . . . . 4 .
## user16 . .   . . . . .   . . . . . . . . . . . . . .   .   . .   . . . . . . . .
## user17 3 3.0 . . 1 . 2.5 . . 4 . . . . . . 4 . .   3.5 . .   4 . 4 . . . . .
## user18 . .   . . 3 . .   . . 4 . . 4 . . . 5 . .   .   4 .   . . . . . 5 . .
## user19 . .   . . . . .   . . . . . . . . . . . . . .   .   4 .   . . . . . . . .
## user20 5 .   . . . . 3.0 . . . . . . . . . 5 . .   .   4 .   . . . . . . . .
## user21 5 .   3 . . 3 4.0 . . . . . . . . . 5 . .   .   . .   . . . . . . . .
## user22 . .   . . . . .   . . . . . . . . . . . . . .   .   . .   . . . . . . . .
## user23 . .   3 . . . .   . . . . 2 . . . . . . . .   .   . .   . . . . . . . .
## user24 . .   . . . . .   . . . . . . . . . . . . . .   .   . .   . . . . . . . .
## user25 . .   . . . . .   . . . . . . . . . . . . . .   .   . .   . . . . . . . .
## user26 5 .   4 . . . .   . . . 5 . . . . . 5 . .   .   . .   . . . . . 5 . .
## user27 4 .   . . . 3 3.0 . . . . . . . . . 4 . .   .   . .   . . . 4 . . . .
## user28 . 3.0 4 . 3 . .   3 3 . . . . . . . . 2.0 .   . 3.0 4 5 . . . . . .
## user29 3 .   . . . . .   . . . . . . . . . . . . . .   .   . .   . . . . . . . .
## user30 4 3.0 3 . . 5 .   . . 3 2 . 4 . 4 5 . 4 . .   .   3 .   . 3 . . . . . .
```

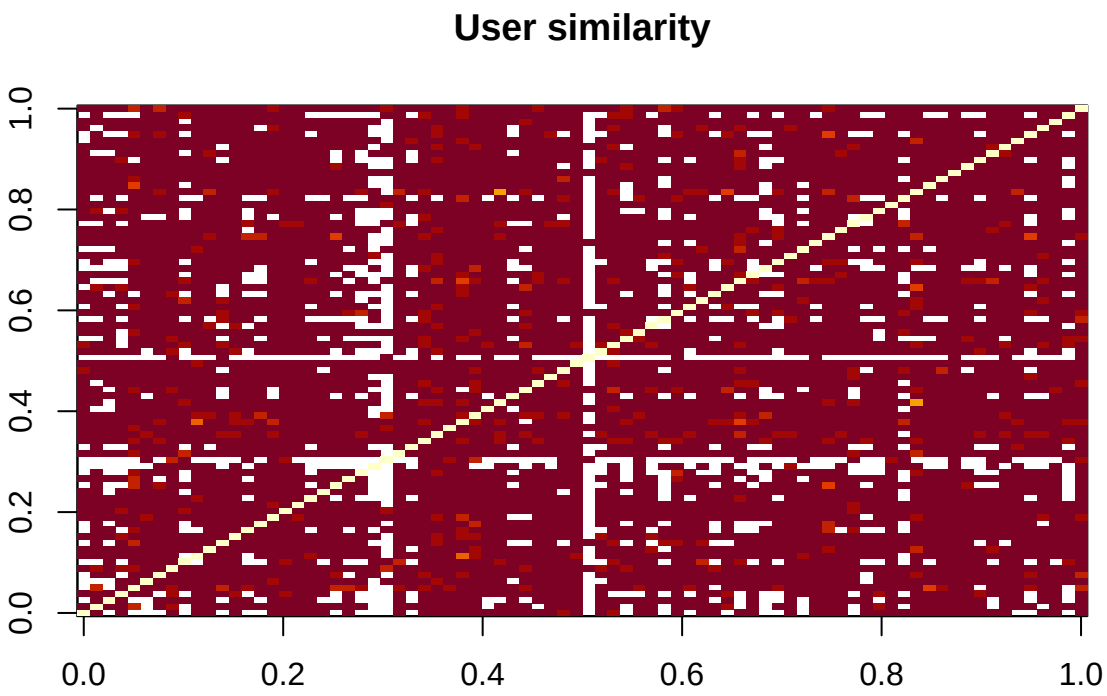
After observing scattered_ratings matrix, we could say there are similar ratings between certain users than others, and more similar ratings between certain movies than others, therefore we could evidence the existence of a group of users pattern or a group of movies pattern.

In order to visualize more efficiently what mentioned before we are going to convert the dgCMatrix (scattered_ratings) matrix into a recommenderlab matrix. After that, we will calculate the users and movies similarities using the **cosine similarity** and plot each one of them.

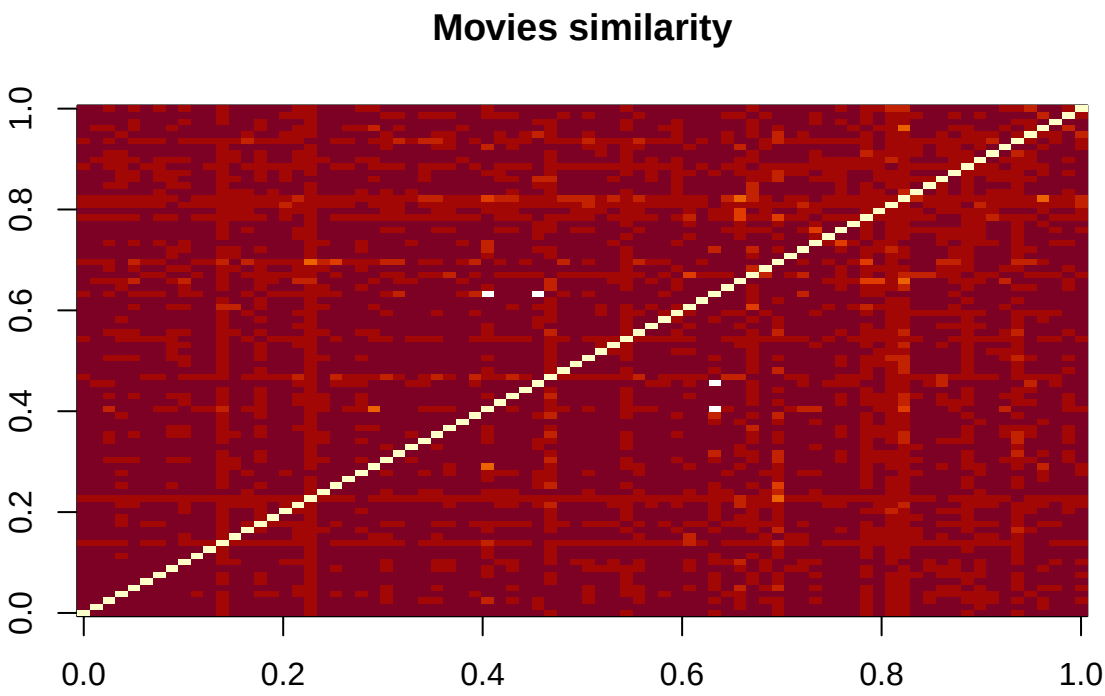
```
###### Convert the dgCMatrix (scattered_ratings) matrix into a recommenderlab matrix
# dgCMatrix into recommenderlab rating matrix
rating_matrix <- new("realRatingMatrix", data=scattered_ratings)
rating_matrix
```

```
## 69878 x 10677 rating matrix of class 'realRatingMatrix' with 9000055 ratings.
```

```
# userId similarity, first 80 users
user_similarity <- similarity(rating_matrix[1:80,],
                              method = "cosine",
                              which = "users")
image(as.matrix(user_similarity), main = "User similarity")
```




```
#movieId similarity, first 80 users
movie_similarity <- similarity(rating_matrix[,1:80],
                               method = "cosine",
                               which = "items")
image(as.matrix(movie_similarity), main = "Movies similarity")
```



In the images above, each row and each column corresponds to a user, and each cell corresponds to the similarity between two users . The more red the cell is, the more similar two users are.

4. Methos and Analysis

Next, we are going to explore the methodology over three different Machine Learning algorithms and present the metrics for each model performance evaluation. The measurement use to evaluate the model is the RMSE (Root Mean Square Error). RMSE is the standard deviation of the residuals (prediction errors).

```
# RMSE function for vectors of ratings and their corresponding predictors
RMSE <- function(true_ratings, predicted_ratings){
  sqrt(mean((true_ratings - predicted_ratings)^2))
}
```

As stated in Irizarry's (2019) Data Science book, we follow the same approach to build our linear regression model as the simplest possible recommendation system. Let's start by predicting the same rating for all movies regardless of user. We can use a model based approach that assumes the same rating for all movies and users with all the differences explained by random variation, like this:

4.1 Average Rating

```r
# Model 1: predict same rating for all movies regardless of users
# predict average rating of all movies
mu_hat <- mean(edx$rating)
mu_hat
```

```
## [1] 3.512465
```

```r
# calculate rmse of this naive approach
naive_rmse <- RMSE(validation$rating, mu_hat)
naive_rmse
```

```
## [1] 1.061202
```

```r
# add rmse results in a table
rmse_results <- data.frame(method = "Naive approach", RMSE = naive_rmse)
rmse_results %>% knitr::kable()
```

method	RMSE
Naive approach	1.061202

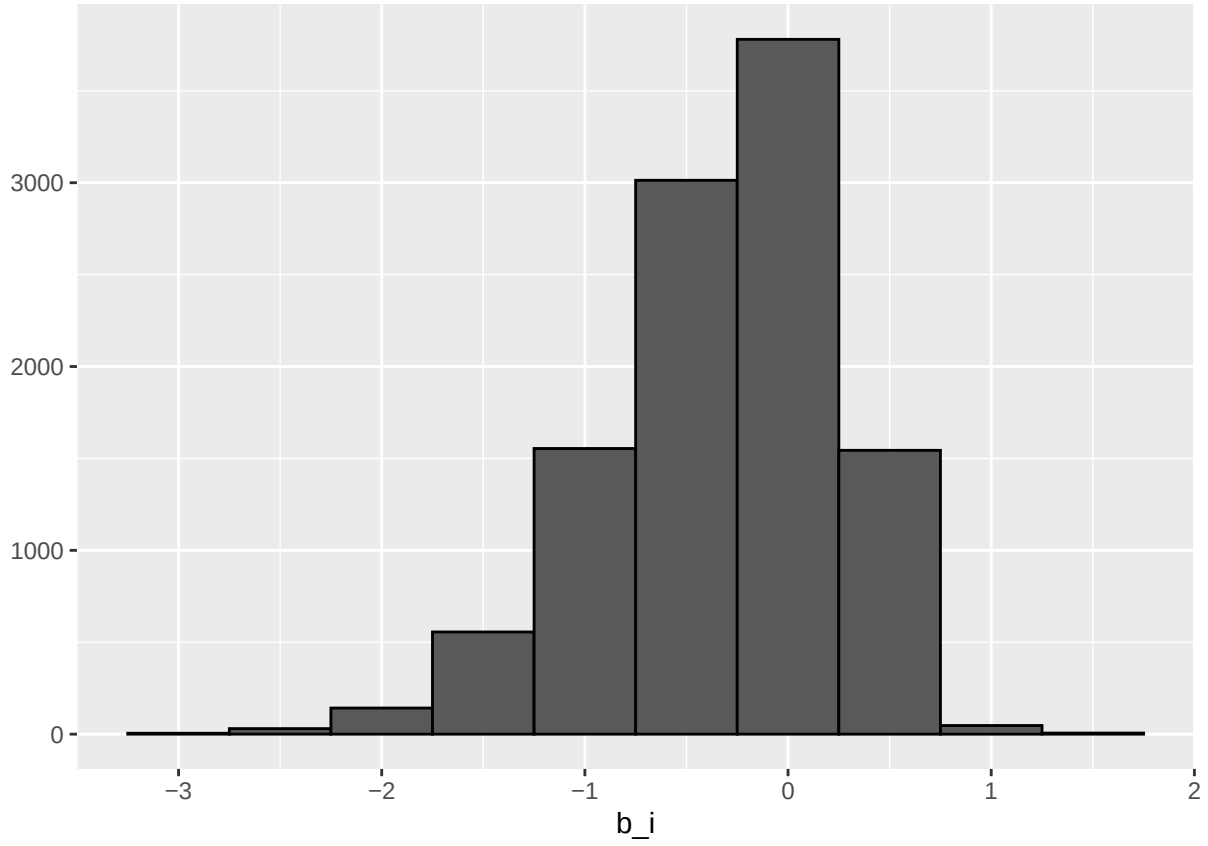
4.2 Movie Bias Effect

Based on the approach above, we will successively add to it different **effects** (also called 'bias') to represent average ranking by each effect.Thus, we know from the current data exploration that some movies are generally rated higher than others. We can augment our previous model by adding the term **bi** to represent average ranking for movie **i**, like this:

```r
# Model 2: modelling movie effect
# estimate movie bias 'b_i' for all movies
mu <- mean(edx$rating)
movie_avgs <- edx %>%
  group_by(movieId) %>%
  summarize(b_i = mean(rating - mu))
```

```
## `summarise()` ungrouping output (override with `.groups` argument)
```

```r
# plot these movie 'bias'
movie_avgs %>% qplot(b_i, geom ="histogram", bins = 10, data = ., color = I("black"))
```




```
# calculate predictions considering movie effect
predicted_ratings <- mu + validation %>%
  left_join(movie_avgs, by='movieId') %>%
  pull(b_i)
# calculate rmse after modelling movie effect
model_1_rmse <- RMSE(predicted_ratings, validation$rating)

rmse_results <- bind_rows(rmse_results,
                          data_frame(method="Movie effect model",
                                     RMSE = model_1_rmse))
```

```
## Warning: `data_frame()` is deprecated as of tibble 1.1.0.
## Please use `tibble()` instead.
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_warnings()` to see where this warning was generated.
```

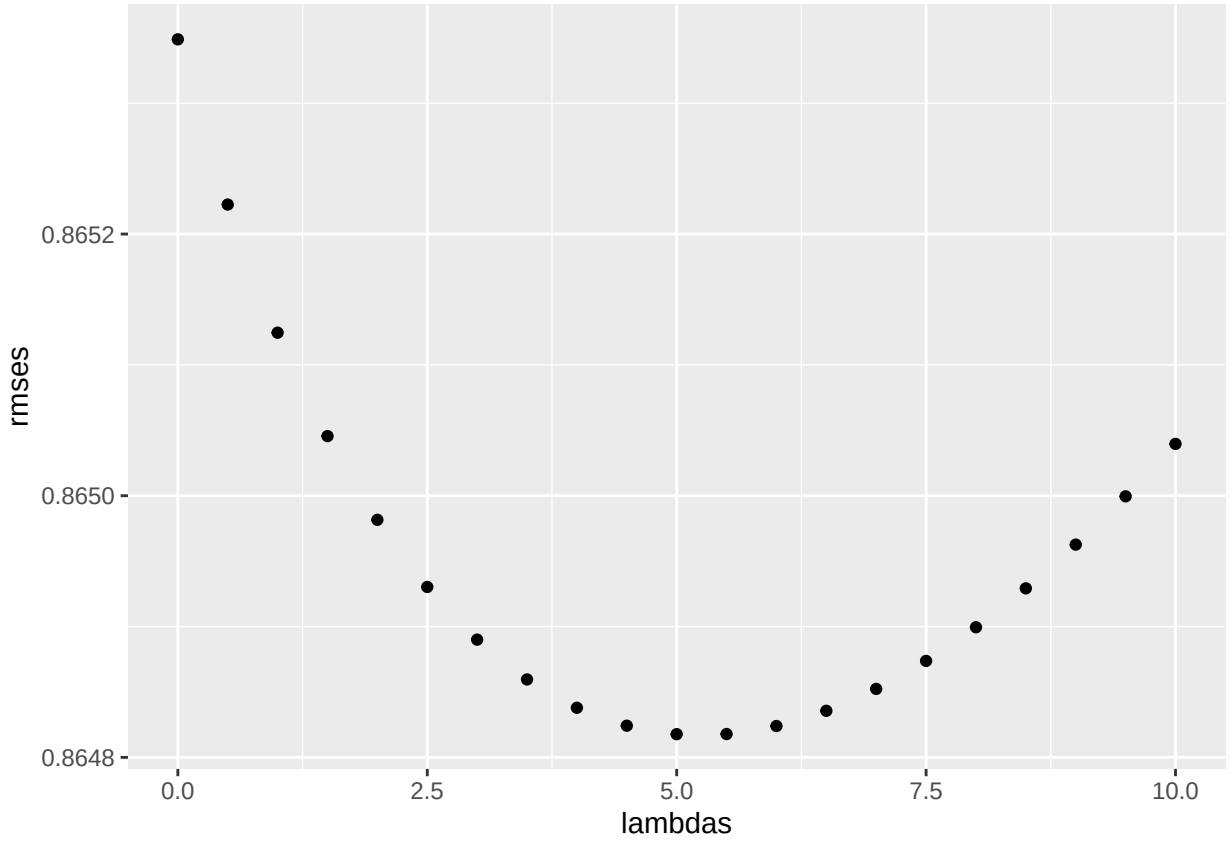
```
rmse_results %>% knitr::kable()
```

method	RMSE
Naive approach	1.0612018
Movie effect model	0.9439087

4.3 Regularization: Movie and User Effect

As seen in the data exploration section, not all movies have the same number of ratings. Some have a a lot of ratings and some have a few. The general idea behind this model is to constrain the total variability of the effect sizes. That is to say, penalize large estimates that are formed using small samples sizes and thus helps avoid the risk of overfitting.

```
# Model 3: regularizing movie + user effect model from previous models
# choosing the penalty term lambda
lambdas <- seq(0, 10, 0.5)

rmses <- sapply(lambdas, function(l){

  mu <- mean(edx$rating)

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

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

  predicted_ratings <-
    validation %>%
    left_join(b_i, by = "movieId") %>%
    left_join(b_u, by = "userId") %>%
    mutate(pred = mu + b_i + b_u) %>%
    pull(pred)

  return(RMSE(predicted_ratings, validation$rating))
})

qplot(lambdas, rmses)
```



We can see the RMSE is the lowest around 5.0 lambda. But lets check the data

```
# finding the lowest lambda point
lambda <- lambdas[which.min(rmses)]
lambda
```

```
## [1] 5
```

```
# calculate rmse after regularizing movie + user effect from previous models
rmse_results <- bind_rows(rmse_results,
                          data_frame(method="Regularized Movie + User effect model",
                                     RMSE = min(rmses)))
rmse_results %>% knitr::kable()
```

method	RMSE
Naive approach	1.0612018
Movie effect model	0.9439087
Regularized Movie + User effect model	0.8648177

5. Result and Conclusions

The RMSEs results for various models are as follows:

```
rmse_results <- bind_rows(rmse_results,
                          data_frame(method="Regularized Movie + User effect model",
                                     RMSE = min(rmses)))
rmse_results %>% knitr::kable()
```

method	RMSE
Naive approach	1.0612018
Movie effect model	0.9439087
Regularized Movie + User effect model	0.8648177
Regularized Movie + User effect model	0.8648177

- We explored three different models in this report. The first model acted as a base model to allow us to compare with subsequent models. As shown in the results table, each time we added a new variable to account for the variation, our model's accuracy was improved.
- It can be observed a continuous improvement of the model leading to a final RMSE value of 0.8642 with the Regularized Movie and User Effect model, which met the requirement of the project.