# edX Course PH125.9x - Capstone Project: MovieLens

*Marcos de Oliveira*

*May 18th, 2019*

# 1. Introduction

## 1.1 Objective

Aim of this project is to create a **recommendation system for Movies**, that could be used by a service such as Netflix. Given a certain user and movie, our model will estimate the rating this specific user would give to this specific movie. This could be used to recommend movies with best estimated ratings to specific users. Use cases of such algorithm can be to **increase user retention, increase his average ticket or purchase rate**.

Most effective models are categorized as ***Hybryd Recommendation System*** because they use both techniques:

- ***Collaborative Filtering***: based on user behaviour, and similar users; in general terms, we can state that the larger the amount of users which are rating items and number of ratings by each user, the better the accuracy that can be obtained by the model
- ***Content-based filtering***: based on item attributes (movie genres, present in this dataset, for example)

Both approaches have pros and cons, and are generally used in conjunction in more sophisticated recommendation systems to improve accuracy.

OUr model will be simpler, and it will try to capture user and movie bias to increase RMSE and accuracy, focusing on RMSE.

**Aim is to get to RMSE that is equal or less than *0.87750* when model is tested against validation data.**

## 1.2 Dataset

We will use a version of MovieLens dataset which has 10 million user reviews of movies. It contains a row for each user evaluation of each film, and columns that contain user and movie information, as well as ratings in a scale of 0 to 5 (expressed in integers or integer plus 0.5).

Code provided by Edx to generate Test and Validation sets did not work in my PC, in spite of attempts to use different suggestions provided in the forum, and long time wasted to try to solve this. Issues found: original code did not complete after more than 24 hours (many attempts); workaround resulted in memory allocation error (many attempts). Many users complained about same issues, probably a result of dataset being too large for our PC version.

After staff orientation, they finally allowed working with data files provided by Edx support teams in **Google Drive** (https://drive.google.com/drive/folders/1IZcBBX0OmL9wu9AdzMBFUG8GoPbGQ38D)

# 2. Methodology

## 2.1 Supervised Learning approach

Training and validation datasets were provided by staff, as explained above.

We will use supervised learning to create a model using a significant amount of data (~90% of observations) and use the rest of data to validate our model.

We have two datasets:

- *Ratings*: data set which will be used to create the model, with 9000055 observations of 6 variables
- *Validation*: data set which will be used to validate the model, with 999999 observations of the same 6 variables

Target variable is rating; this attribute will be used in ratings dataset to create the model, and in test dataset to calculate how effective our models are.

## 2.2 Process

In summary, process which will be used is:

- Load libraries and data
- Explore data (answering QUiz questions)
- Further exploit of data
- Build a simple model based on overall rating means
- Improve the model including *movie bias* for each movie
- Further improve model adding *user bias* for each user

# 3. Results

## 3.1 Load required libraries and data

```
library(tidyverse)
```

```
## Warning: package 'tidyverse' was built under R version 3.4.4
```

```
## -- Attaching packages ------------------------------ tidyverse 1.2.1 --
```

```
## v ggplot2 3.0.0     v purrr   0.2.4
## v tibble  1.4.2     v dplyr   0.7.6
## v tidyr   0.8.0     v stringr 1.3.0
## v readr   1.1.1     v forcats 0.3.0
```

```
## Warning: package 'ggplot2' was built under R version 3.4.4
```

```
## Warning: package 'purrr' was built under R version 3.4.4
```

```
## Warning: package 'dplyr' was built under R version 3.4.4
```

```
## Warning: package 'forcats' was built under R version 3.4.4
## -- Conflicts --------------------------------- tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```
library(caret)
```

```
## Warning: package 'caret' was built under R version 3.4.4
```

```
## Loading required package: lattice
```

```
##
## Attaching package: 'caret'
```

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

```
library(dplyr)
library(lubridate)
```

```
## Warning: package 'lubridate' was built under R version 3.4.4
```

```
##
## Attaching package: 'lubridate'
```

```
## The following object is masked from 'package:base':
##
##     date
```

```
library(ggplot2)
```

```
setwd("~/Marcos/Data Science/Cursos/Harvard Data Science Prof Cert/PH125.9x Capstone")
#setwd("~/Marcos/Pessoal/Capstone")
validation<-readRDS("validation.rds")
ratings<-readRDS("edx.rds")
```

## 3.2 Explore data (answering Quiz questions)

### 3.2.1 Q1 How many rows and columns are there in the edx dataset?

```
#Q1
nrow(ratings)
```

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

```
ncol(ratings)
```

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

### 3.2.2 Q2 How many zeros were given as ratings in the edx dataset?

```
#Q2
length(which(ratings$rating==0))
```

```
## [1] 0
```

```
#How many threes were given as ratings in the edx dataset?
length(which(ratings$rating==3))
```

```
## [1] 2121240
```

### 3.2.3 Q3 How many different movies are in the edx dataset?

```
#Q3
ratings %>%
  summarise(n_distinct(movieId))
```

```
##   n_distinct(movieId)
## 1               10677
```

### 3.2.4 Q4 How many different users are in the edx dataset?

```
#Q4
ratings%>%
  summarise(n_distinct(userId))
```

```
##   n_distinct(userId)
## 1              69878
```

### 3.2.5 Q5 How many movie ratings are in each of the following genres in the edx dataset?

### Drama, Comedy, Thriller, Romance

```
#Q5
Genres<-ratings%>%select(genres)
s<-split(Genres, sample(1:5, nrow(Genres), replace=T))
sep1 <- separate_rows(data = s[[1]], genres, sep = "\\|")
sep2 <- separate_rows(data = s[[2]], genres, sep = "\\|")
sep3 <- separate_rows(data = s[[3]], genres, sep = "\\|")
sep4 <- separate_rows(data = s[[4]], genres, sep = "\\|")
sep5 <- separate_rows(data = s[[5]], genres, sep = "\\|")
j<-do.call("rbind",list(sep1,sep2,sep3,sep4,sep5))
length(which(j$genres=="Drama"))
```

```
## [1] 3910127
```

```
length(which(j$genres=="Comedy"))
```

```
## [1] 3540930
```

```
length(which(j$genres=="Thriller"))
```

```
## [1] 2325899
```

```
length(which(j$genres=="Romance"))
```

```
## [1] 1712100
```

### 3.2.6 Q6 Which movie has the greatest number of ratings?

```
#Q6
ratings %>%
  group_by(title) %>%
  summarize(count=n()) %>%
  top_n(1,count)
```

```
## Warning: package 'bindrcpp' was built under R version 3.4.4
```

```
## # A tibble: 1 x 2
##   title               count
##   <chr>               <int>
## 1 Pulp Fiction (1994) 31362
```

### 3.2.7 Q7 What are the five most given ratings in order from most to least?

```
#Q7
ratings %>%
  group_by(rating) %>%
  summarize(count=n()) %>%
  top_n(5,count)%>%
  arrange(desc(count))
```

```
## # A tibble: 5 x 2
##   rating   count
##    <dbl>   <int>
## 1   4.00 2588430
## 2   3.00 2121240
## 3   5.00 1390114
## 4   3.50  791624
## 5   2.00  711422
```

### 3.2.8 Q8 True or False: In general, half star ratings are less common than whole star ratings (e.g., there are fewer ratings of 3.5 than there are ratings of 3 or 4, etc.)

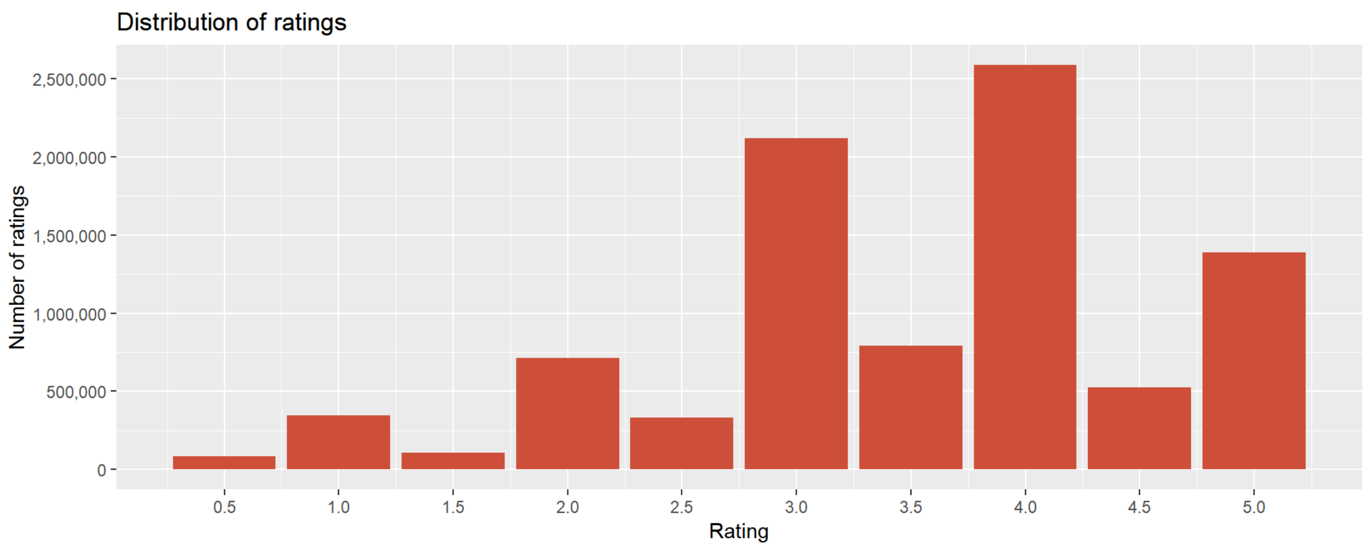
```
#Q8
ratings <- ratings %>%
  mutate(Integer=ifelse(round(rating) == rating,TRUE,FALSE))
summary(ratings$Integer)
```

```
##    Mode   FALSE    TRUE
## logical 1843170 7156885
```

## 3.3 Further data exploitation

### 3.3.1 Distribution of ratings

```
#Plot of count of each rating
ratings %>%
  group_by(rating) %>%
  summarize(count=n()) %>%
  ggplot(aes(x=rating,y=count))+
  geom_bar(fill="tomato3",stat="identity")+
  scale_x_continuous(breaks = pretty(ratings$rating, n = 10)) +
  scale_y_continuous(breaks = seq(0, 3000000, by = 500000),labels = scales::comma)+
  labs(title="Distribution of ratings",
       y="Number of ratings",
       x="Rating")
```



### 3.3.2 Rating along months

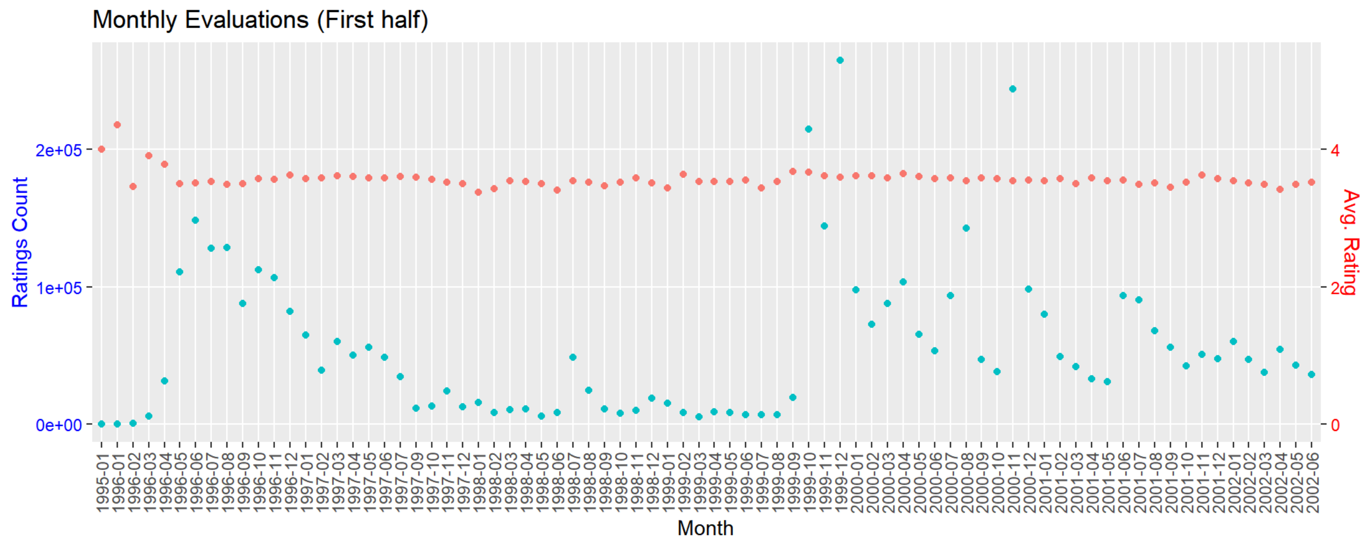
```
#Converting timestamp to posix and extracting day
ratings$date<-as_date(as_datetime(ratings$timestamp))
#Extracting month and year
ratings$monthyear<-format(ratings$date,"%Y-%m")
#Summarizing
monthly_summary <- ratings %>% group_by(monthyear) %>%
  summarize(count=n(),avgrating=mean(rating))
nrows<-nrow(monthly_summary)
maxcount<-max(monthly_summary$count)

maxmonthyear<-ratings$monthyear[which.max(monthly_summary$count)]
maxmonthavg<-monthly_summary$avgrating[which(monthly_summary$monthyear==maxmonthyear)]

#Finding out which were top 5 movies most rated in month with most ratings
maxmonth<-monthly_summary[which.max(monthly_summary$count),]$monthyear
ratings_maxmonth=ratings[which(ratings$monthyear==maxmonth),]

#Number of movies rated in month with max ratings
ratings_maxmonth %>%
  summarize(n_distinct(movieId))
```

```
##   n_distinct(movieId)
## 1                2867
```

```
#Top 5 movies in number of ratings in month with max count of ratings
ratings_maxmonth %>%
  group_by(title)%>%
  summarize(count=n())%>%
  top_n(5,count)%>%
  arrange(desc(count))
```

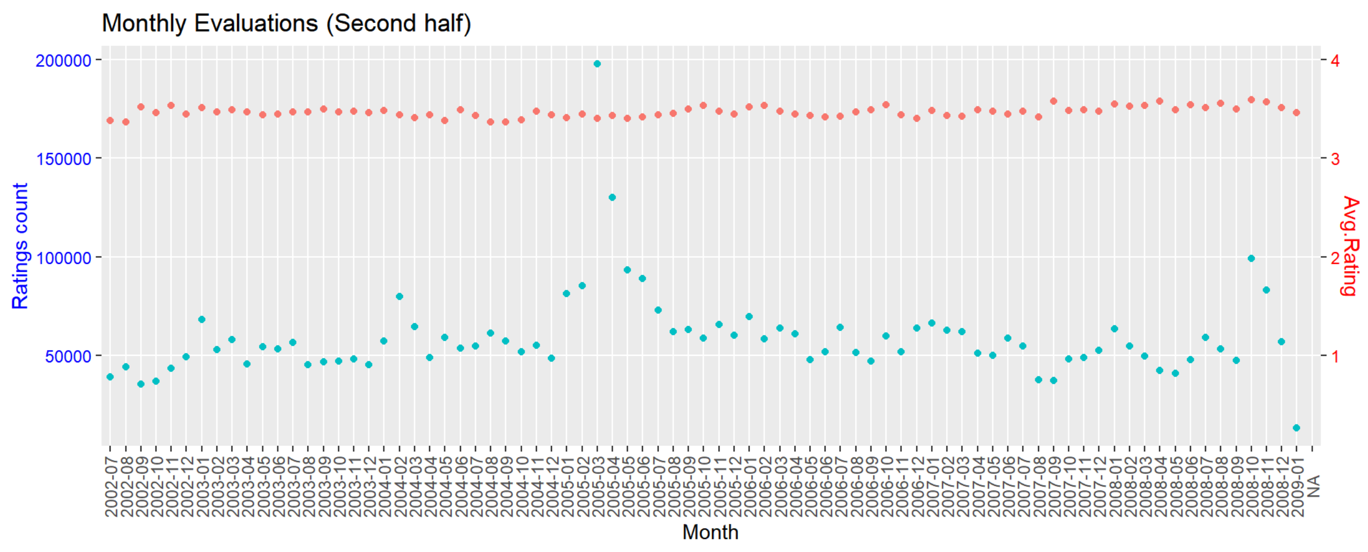
```
## # A tibble: 5 x 2
##   title                                                             count
##   <chr>                                                             <int>
## 1 Star Wars: Episode IV - A New Hope (a.k.a. Star Wars) (1977)       1118
## 2 Star Wars: Episode V - The Empire Strikes Back (1980)              1099
## 3 Matrix, The (1999)                                                  969
## 4 Raiders of the Lost Ark (Indiana Jones and the Raiders of the Los~   941
## 5 Star Wars: Episode VI - Return of the Jedi (1983)                   935
```

```
#Plotting Ratings count and average rating per month
#We will divide in two parts otherwise it will become cluttered
ScaleFactor<-50000
ggplot(monthly_summary[1:round(nrows/2),],aes(x=monthyear))+
  geom_point(aes(y=count, colour="red"))+
  geom_point(aes(y=avgrating*ScaleFactor, colour="black"))+
  labs(title="Monthly Evaluations (First half)",
       y="Evaluations",
       x="Month") +  # title and caption
  theme(axis.text.x = element_text(angle = 90, vjust=0.5),  # rotate x axis text
        panel.grid.minor = element_blank()) +  # turn off minor grid
  scale_y_continuous(name="Ratings Count",
                     sec.axis=sec_axis(~./ScaleFactor, name="Avg. Rating")) +
  theme(
    legend.position = "none",
    axis.title.y.left=element_text(color="blue"),
    axis.text.y.left=element_text(color="blue"),
    axis.title.y.right=element_text(color="red"),
    axis.text.y.right=element_text(color="red")
  )
```



```
ggplot(monthly_summary[round(nrows/2)+1:nrows,],aes(x=monthyear))+
  geom_point(aes(y=count, colour="red"))+
  geom_point(aes(y=avgrating*ScaleFactor, colour="black"))+
  labs(title="Monthly Evaluations (Second half)",
       y="Evaluations",
       x="Month") +  # title and caption
  theme(axis.text.x = element_text(angle = 90, vjust=0.5),  # rotate x axis text
        panel.grid.minor = element_blank()) +  # turn off minor grid
  scale_y_continuous(name="Ratings count",
                     sec.axis=sec_axis(~./ScaleFactor, name="Avg.Rating")) +
  theme(
    legend.position = "none",
    axis.title.y.left=element_text(color="blue"),
    axis.text.y.left=element_text(color="blue"),
    axis.title.y.right=element_text(color="red"),
    axis.text.y.right=element_text(color="red")
  )
```

```
## Warning: Removed 78 rows containing missing values (geom_point).

## Warning: Removed 78 rows containing missing values (geom_point).
```



### 3.3.3 Users who rate integers

```
#Overall Integer and non-Integer ratings
ratings %>%
  group_by(Integer) %>%
  summarize(count=n())
```

```
## # A tibble: 2 x 2
##   Integer   count
##   <lgl>     <int>
## 1 FALSE   1843170
## 2 TRUE    7156885
```

```
#Use all() function to check if all values in previously defined Integer column are TRUE per
 user
user_integer <- ratings%>%group_by(userId) %>%
  summarize(Integerrate=all(Integer))
summary(user_integer[,2])
```

```
##  Integerrate
##  Mode :logical
##  FALSE:25115
##  TRUE :44763
```

## 3.4 First model - Simple overall rating means

```
#Create RMSE generic function
RMSE = function(true, predicted){
  sqrt(mean((true - predicted)^2))
}

#Assign train and test sets (more mnemonic to supervised learning)
train<-ratings
test<-validation

#Create Model 1
#Simple means as prediction
means<-mean(train$rating)
error<-RMSE(test$rating,means)
#Store results in a dataframe for comparison
rmse_results<-data_frame(method="Average",RMSE=error)
rmse_results
```

```
## # A tibble: 1 x 2
##   method   RMSE
##   <chr>   <dbl>
## 1 Average  1.06
```

## 3.5 Second Model - Adding *Movie bias*

```
#Create Model 2 adding movie bias
#Group bymovie_id and clculate movie bias
movie_avg<-train %>%
  group_by(movieId) %>%
  summarize(movie_bias=mean(rating-means))

#add movie_bias to test dataset
predicted<-means+test %>%
  left_join(movie_avg,by='movieId')%>%
  pull(movie_bias)

error<-RMSE(test$rating,predicted)
rmse_results<-bind_rows(rmse_results,data_frame(method="MovieBias",RMSE=error))
rmse_results
```

```
## # A tibble: 2 x 2
##   method     RMSE
##   <chr>     <dbl>
## 1 Average   1.06
## 2 MovieBias 0.944
```

## 3.6 Third Model - Adding *User bias*

```
#Create Model 3 adding user bias
#On top of previous model we will extract user bias
user_avg<-train %>%
  left_join(movie_avg,by='movieId') %>%
  group_by(userId) %>%
  summarize(user_bias=mean(rating-means-movie_bias))

#Add movie and user biases to test dataset
predicted<-test %>%
  left_join(movie_avg,by='movieId')%>%
  left_join(user_avg,by='userId')%>%
  mutate(pred=means+movie_bias+user_bias)%>%
  .$pred

error<-RMSE(test$rating,predicted)
rmse_results<-bind_rows(rmse_results,data_frame(method="MovieandUserBias",RMSE=error))
rmse_results
```

```
## # A tibble: 3 x 2
##   method            RMSE
##   <chr>            <dbl>
## 1 Average          1.06
## 2 MovieBias        0.944
## 3 MovieandUserBias 0.865
```

```
# As can be seen, we achieved our target RMSE
```

### 3.6.1 Calculating Model Accuracy

```
#Calculate accuracy of predictions
#Rounded to .5
predictedrounded<-round(predicted/0.5)*0.5
mean(predictedrounded==test$rating)
```

```
## [1] 0.2479812
```

```
#Rounded to nearest integer as most of ratings are integers
predictedrounded<-round(predicted)
mean(predictedrounded==test$rating)
```

```
## [1] 0.3614204
```

# 4. Conclusion

Analysing data from *MovieLens* dataset, we faced initial difficulties, mainly because of size of data (10M records). Although this is probably not considered Big Data in terms of Volume (one of the five Vs), for the equipment we used it was probably too big. It took very long time to run script provided by staff (more than 24 hours after I aborted it).

Using other approaches suggested by staff and members in discussion forum, did not help either, because they all resulted in error messages, indicating mainly lack of memory.

This resulted in a loss of precious time and effort. After some discussion, staff decided to provide training and test sets divided.

After answering Quiz questions as initial investigation of data, we generated a few simple analysis for further exploitation. For example, we generated a few graphs showing **distribution of amount of ratings per rate**, and **number and average of ratings per month**. We also indicated **movies rated in the date with most ratings**, and the **5 titles which received largest amount of rates**. Finally, we **counted and tagged users who only provide integer ratings**, versus the ones who use also integer+0.5.

When building models, we used same approach as in the course, first creating a model using **simple average of ratings**, and improving it with ***MovieBias*** and ***UserBias***.

We were able to get to target rating (**lower than 0.87750**) with the third model, and we decided to stop because of lack of time.

Finally, accuracy of our model was low (about 25%), which we improved to about 36% by using integer rating only, since integer ratings are most common.