# HarvardX: PH125.9x Data Science

Khushnud Sapaev

January 5, 2020

## Table of Contents



## Title

MovieLens Rating Prediction Project

## Introduction

Nowadays, data-driven companies use recommendation systems which can be facilitated to predict which rating a particular user will give to a particular product. Machine learning algorithms are applied in recommendations systems for providing insightful meaningful predictions. This project is also based on that approach,users were assigned to give specific recommendations for the movies. The project focuses on creating a movie recommendation system using the 10M version of the MovieLens dataset compiled by the GroupLens Research.

This project aims to develop a machine learning algorithm that predicts user ratings (from 0.5 to 5 stars) using the data provided in the dataset (edx dataset) to predict movie ratings in the given validation set.The Root Mean Square Error (RMSE) will be used to evaluate the model performance. RMSE is a measure of how spread out the residuals are, it measures how concentrated the data is around the line of best fit. Models will be developed to compare RMSE in order to assess highest quality.The evaluation criteria for this algorithm is the RMSE expected to be lower than 0.8775.The best resulting model will be used to predict the movie ratings. The formula of the RMSE is as follows:

$$RMSE = \sqrt{\frac{1}{N}\sum_{u,i}(\hat{y}_{u,i} - y_{u,i})^2}$$

## Data preparation

```r
#############################################################
# Create edx set and validation set
# Note: this process could take a couple of minutes as tidyverse and caret
packages to be installed and files to be downloaded
# Loading libraries, if does not exist then installing first
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")
dl <- tempfile()
download.file("http://files.grouplens.org/datasets/movielens/ml-10m.zip", dl)
ratings <- read.table(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")
```

The MovieLens dataset will be splitted into 2 subsets: (1) edx, a training subset to train the algorithm, and (2) validation, a test subset to test the movie ratings.

```r
# The Validation subset will be 10% of the MovieLens data.
set.seed(1)
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 subset:
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)
```

# Methods and Analysis

## Data Analysis

To have a clear vision of the dataset columns a summary was run on both subsets. The complete cases show no NA’s in the subsets.

```
##      userId         movieId          rating        timestamp
##  Min.   :    1   Min.   :    1   Min.   :0.500   Min.   :7.897e+08
##  1st Qu.:18124   1st Qu.:  648   1st Qu.:3.000   1st Qu.:9.468e+08
##  Median :35738   Median : 1834   Median :4.000   Median :1.035e+09
##  Mean   :35870   Mean   : 4122   Mean   :3.512   Mean   :1.033e+09
##  3rd Qu.:53607   3rd Qu.: 3626   3rd Qu.:4.000   3rd Qu.:1.127e+09
##  Max.   :71567   Max.   :65133   Max.   :5.000   Max.   :1.231e+09
##     title              genres
##  Length:9000055     Length:9000055
##  Class :character   Class :character
##  Mode  :character   Mode  :character
##
##
##
```

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

```
##      userId         movieId          rating        timestamp
##  Min.   :    1   Min.   :    1   Min.   :0.500   Min.   :7.897e+08
##  1st Qu.:18096   1st Qu.:  648   1st Qu.:3.000   1st Qu.:9.467e+08
##  Median :35768   Median : 1827   Median :4.000   Median :1.035e+09
##  Mean   :35870   Mean   : 4108   Mean   :3.512   Mean   :1.033e+09
##  3rd Qu.:53621   3rd Qu.: 3624   3rd Qu.:4.000   3rd Qu.:1.127e+09
##  Max.   :71567   Max.   :65133   Max.   :5.000   Max.   :1.231e+09
##     title              genres
##  Length:999999      Length:999999
##  Class :character   Class :character
##  Mode  :character   Mode  :character
##
##
##
```

```
## [1] 999999
```

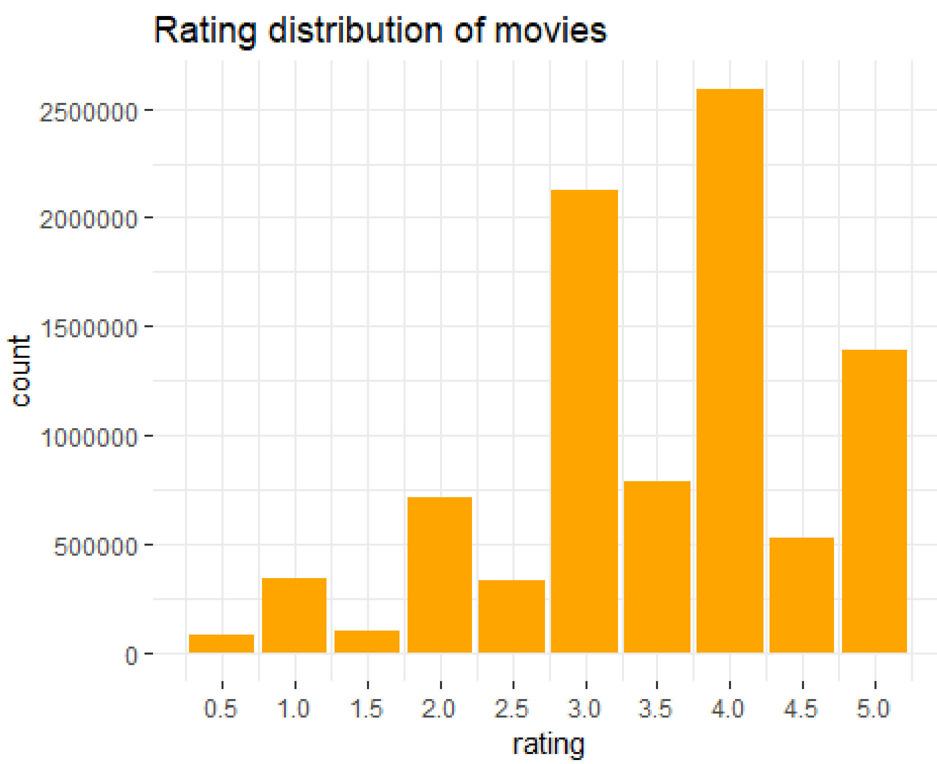
Drama and comedy genres outnumber the rest of the genres.

```
edx %>% group_by(genres) %>% summarize(count = n()) %>% arrange(desc(count))
```

```
## # A tibble: 797 x 2
##    genres                     count
##    <chr>                      <int>
##  1 Drama                     733296
##  2 Comedy                    700889
##  3 Comedy|Romance            365468
##  4 Comedy|Drama              323637
##  5 Comedy|Drama|Romance      261425
##  6 Drama|Romance             259355
##  7 Action|Adventure|Sci-Fi   219938
##  8 Action|Adventure|Thriller 149091
##  9 Drama|Thriller            145373
## 10 Crime|Drama               137387
## # ... with 787 more rows
```

Whole number and high number rates are preferred by the users. The distribution shows that rating 4 has the highest quantity followed by 3 and 5, the least common ratings are 1.5 and 0.5.

```
edx %>% group_by(rating) %>% summarize(count = n()) %>% arrange(desc(count))

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

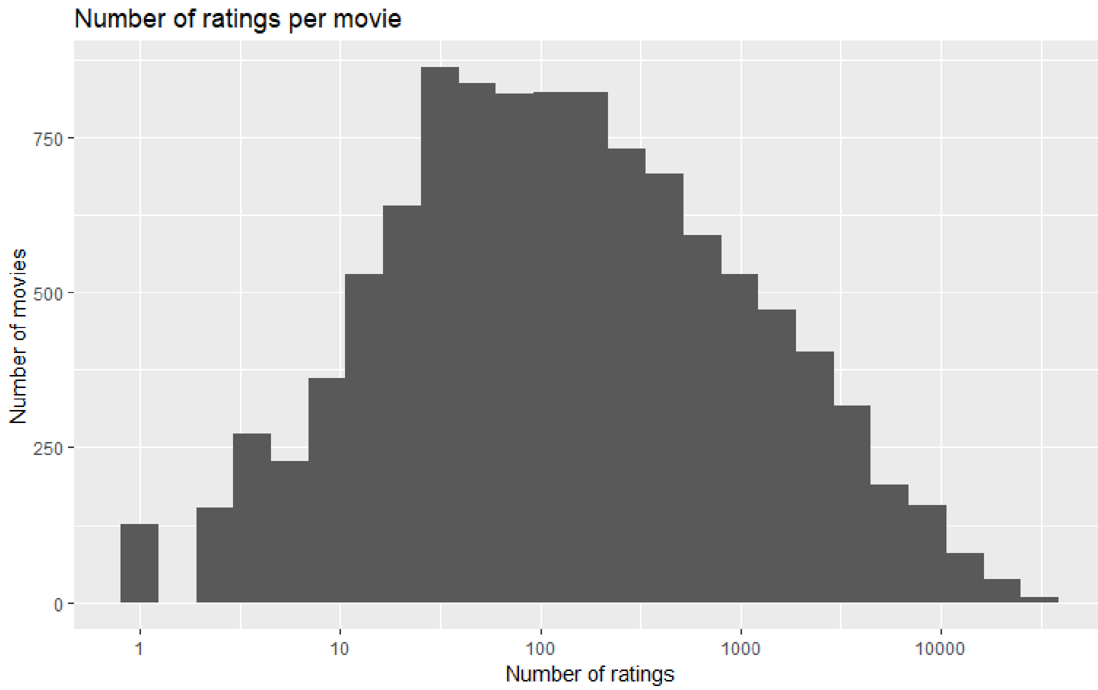



Edx subset has nearly 70,000 unique users and about 10,700 different movies.

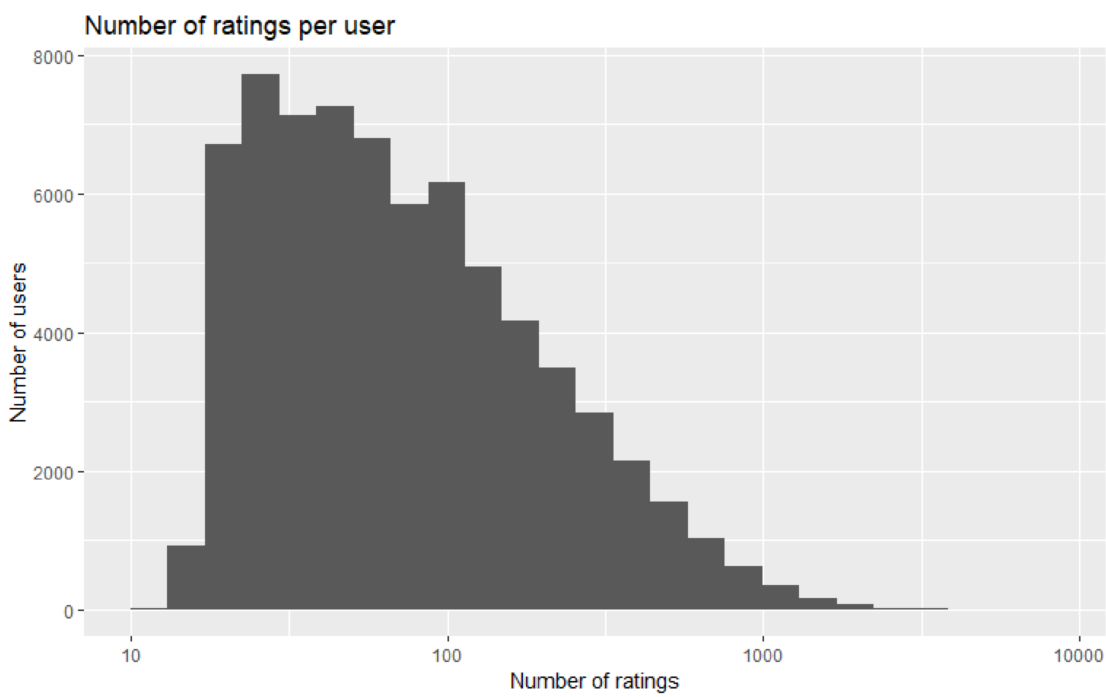
```
edx %>% summarize(Users = length(unique(userId)), Movies =
length(unique(movieId)))

##   Users Movies
## 1 69878  10677
```

Not the all movies have the sufficient number of ratings. The histogram shows that some movies have only one rating (126 movies fall in that category) or less than 10 ratings. Regularisation and a penalty term applied to the models can be useful in this project.

Regularization helps to choose preferred model complexity, so that model is better at predicting. Regularization is nothing but adding a penalty term to the objective function and control the model complexity using that penalty term.




The majority of users have rated between 30 and 100 movies, thus a user penalty term should be included in the models. More recent movies get more user ratings. Movies earlier than 1930 get few ratings, whereas later movies get considerably more ratings.




Both movies and users affect the prediction results so, as modeling approaches I will use movie and user effect model and regularized model as well.

## Movie and user effect model

The RMSE is lower than 0.8775. The model has some uncertainty as large errors can increase the RMSE.

```
# Mean rating of edx subset, movie and user averages
mu <- mean(edx$rating)
movie_avgs <- edx %>% group_by(movieId) %>% summarize(m_a = mean(rating - mu))
user_avgs <- edx %>% left_join(movie_avgs, by='movieId') %>% group_by(userId)
%>%
  summarize(u_a = mean(rating - mu - m_a))

# Testing and saving the RMSE result
predicted_ratings <- validation %>% left_join(movie_avgs, by='movieId') %>%
  left_join(user_avgs, by='userId') %>%
  mutate(predict = mu + m_a + u_a) %>% pull(predict)

movie_user_model_rmse <- RMSE(predicted_ratings, validation$rating)

# Checking the result
rmse_results <- tibble(method="Movie and user effect model", RMSE =
movie_user_model_rmse)
rmse_results %>% knitr::kable()
```

| method | RMSE |
|---|---|
| Movie and user effect model | 0.8653488 |

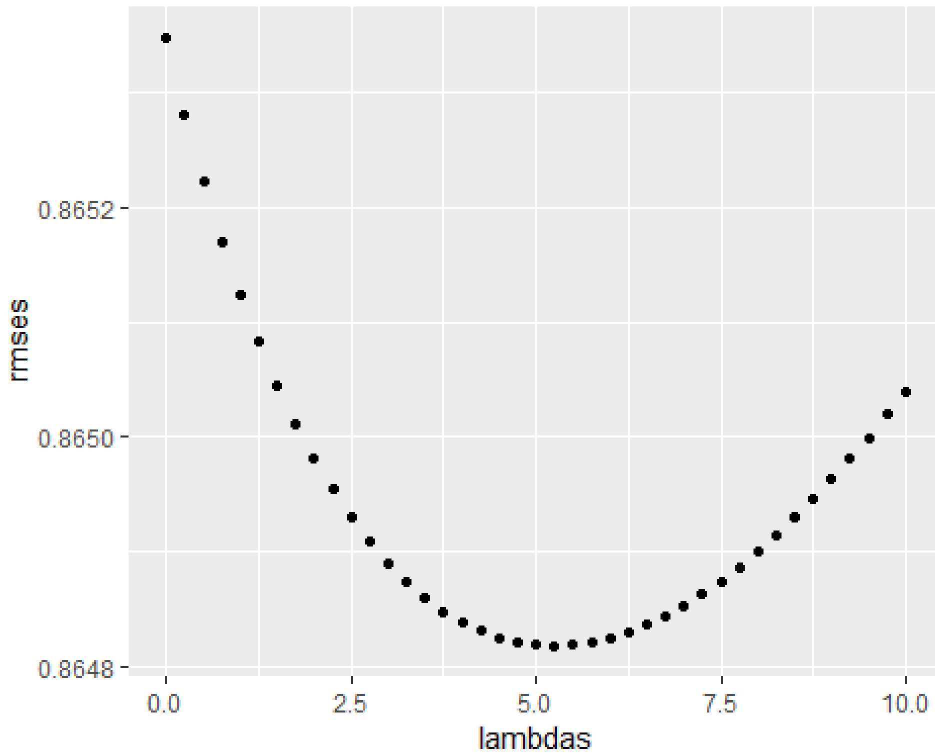
When making predictions, exact number and prediction is necessary, not an interval. For this the concept of regularization can be introduced to penalize large estimates from small sample sizes. Regularization is is a method applied to decrease the overfitting effect.

## Regularized movie and user effect model

The best result was achieved at lambda = 5.25. The RMSE is lower than the result from the previous model.

```
# lambda is a tuning parameter
lambdas <- seq(0, 10, 0.25)

# m_a and u_a for each lambda followed by prediction and testing the rating
rmses <- sapply(lambdas, function(l){
  mu <- mean(edx$rating)
  m_a <- edx %>% group_by(movieId) %>% summarize(m_a = sum(rating - mu)/(n()+l))
  u_a <- edx %>% left_join(m_a, by="movieId") %>% group_by(userId) %>%
    summarize(u_a = sum(rating - mu - m_a)/(n()+l))
  predicted_ratings <- validation %>% left_join(m_a, by = "movieId") %>%
    left_join(u_a, by = "userId") %>%
    mutate(predict = mu + m_a + u_a) %>% pull(predict)
  return(RMSE(predicted_ratings, validation$rating))
})
```

```r
# Visualization of RMSEs vs lambdas to select the optimal lambda
qplot(lambdas, rmses)
```




```r
# The optimal lambda is the one with the minimal RMSE
lambda <- lambdas[which.min(rmses)]

# Checking the result with the result from the previous model
rmse_results <- tibble(method="Regularized movie and user effect model", RMSE =
min(rmses))
rmse_results %>% knitr::kable()
```

| method | RMSE |
|---|---|
| Regularized movie and user effect model | 0.864817 |

## Conclusion

The machine learning algorithm built to predict movie ratings with MovieLens dataset. The regularized model including the effect of user is characterized by the lower RMSE value and is the optimal model to use for the current project. The final model for the project is the following:

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

The final model characterised by the RMSE value at 0.864817, lower than the evaluation criteria provided as a goal of the project.