HarvardX: PH125.9x Data Science
MovieLens Rating Prediction Project

Eddwin Cheteni

03/06/2020

1.0 Introduction

Recommendation systems use ratings that users have given to items to make specific recommendations. Companies that sell many products to many customers and allow these customers to rate their products, like eBay, are able to collect huge datasets that can be used to predict the rating a particular user will give to a specific item. Users are given recommendations to items which predicts a high rating score.

The same could be done for other items such as movies for instance in this case. Recommendation systems are one of the most commonly used models in machine learning algorithms. In fact the success of Netflix is said to be based on its strong recommendation system. The Netflix prize (open competition for the best collaborative filtering algorithm to predict user ratings for films, based on previous ratings without any other information about the users or films), in fact, represent the high importance of algorithm for products recommendation system.

2.0 Problem definition

The aim in this project is to train a machine learning algorithm that predicts user ratings (from 0.5 to 5 stars) using the inputs of a provided subset (edx dataset provided by the staff) to predict movie ratings in a provided validation set.

The value used to evaluate algorithm performance is the Root Mean Square Error, or RMSE. RMSE is one of the most used measure of the differences between values predicted by a model and the values observed. RMSE is a measure of accuracy, to compare forecasting errors of different models for a particular dataset, a lower RMSE is better than a higher one. The effect of each error on RMSE is proportional to the size of the squared error; thus larger errors have a disproportionately large effect on RMSE. Consequently, RMSE is sensitive to outliers.

Four models that will be developed will be compared using their resulting RMSE in order to assess their quality. The evaluation criteria for this algorithm is a RMSE expected to be lower than 0.8775. The function that computes the RMSE for vectors of ratings and their corresponding predictors will be the following:

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

Finally, the model that gives a lowest RMSE will be considered as the best model to predict the movie ratings.

3.0 Dataset

The code is provided in the edx capstone project overview [Create edx and Validation Sets]:
- https://grouplens.org/datasets/movielens/10m/ -
http://files.grouplens.org/datasets/movielens/ml-10m.zip

```
#Create test and validation sets
# Create edx set, validation set, and submission file
if(!require(tidyverse)) install.packages("tidyverse", repos = "http://cran.us
.r-project.org")
if(!require(caret)) install.packages("caret", repos = "http://cran.us.r-proje
ct.org")
# 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 <- 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(movieI
d))[movieId],
                                           title = as.character(title),
                                           genres = as.character(genres))
movielens <- left_join(ratings, movies, by = "movieId")
# Validation set will be 10% of MovieLens data
set.seed(1)
test_index <- createDataPartition(y = movielens$rating, times = 1, p = 0.1, l
ist = 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)
```

The above chunk of code gives a partition of the dataset for edx and validation our dataset. It also removes the unnecessary files from the working directory, which is always a good coding practice (‘always wash your hands after you touch objects’). The datasets were downloaded and saved as .rds file.

3.1 Data Exploration

Loading the data and necessary library to use during exploration:

Lets check the dimensions of edx and validation dataset:

```
## [1] 9000055       6

## [1] 999999      6
```

We can see that both datasets contains 6 columns and the validation set is 10% of the edx dataset.

The rating column in the validation set is removed so that we predict the ratings of the validation set as if they were unknown in order to compute our RMSE.

Lets remove the rating column from the validation set:

Once a clean dataset is available, one must inquire the dataset features and calculate the basic summary statistics.

```
## # A tibble: 6 x 6
##   userId movieId rating timestamp title                   genres
##    <int>   <dbl>  <dbl>     <int> <chr>                   <chr>
## 1      1     122      5 838985046 Boomerang (1992)        Comedy|Romance
## 2      1     185      5 838983525 Net, The (1995)         Action|Crime|Thr
iller
## 3      1     292      5 838983421 Outbreak (1995)         Action|Drama|Sci
-Fi|T~
## 4      1     316      5 838983392 Stargate (1994)         Action|Adventure
|Sci-~
## 5      1     329      5 838983392 Star Trek: Generations~ Action|Adventure
|Dram~
## 6      1     355      5 838984474 Flintstones, The (1994) Children|Comedy|
Fanta~
```

Each row represents a rating given by one user to one movie.

A summary of the edx set confirms that there are no missing values as shown below:

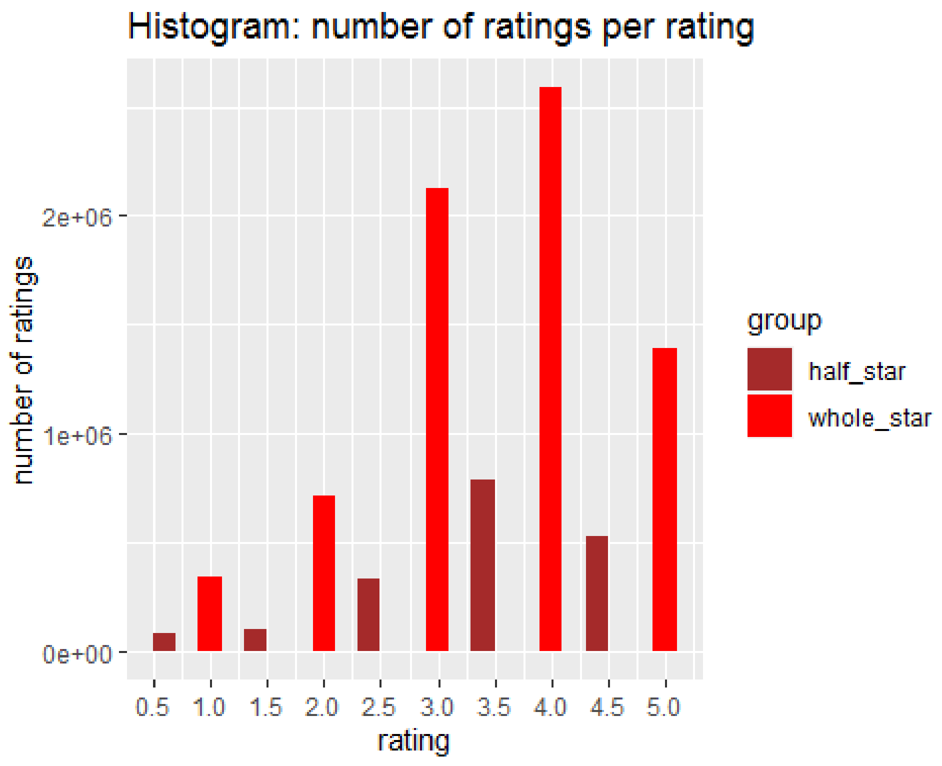
```
##      userId         movieId          rating        timestamp
##  Min.   :    1   Min.   :    1   Min.   :0.500   Min.   :7.897e+08
##  1st Qu.:18124   1st Qu.:  648   1st Qu.:3.000   1st Qu.:9.468e+08
##  Median :35738   Median : 1834   Median :4.000   Median :1.035e+09
##  Mean   :35870   Mean   : 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
##
##
##
```

We can see the number of unique users that provided ratings and check how many unique movies were rated. The total of unique movies and users in the edx subset is almost 70 000 unique users and approximately 10.700 different movies:

```
##                     Count
## Number of Reviews 9000055
## Distinct Users      69878
## Distinct Movies     10677
## Distinct Titles     10676
## Distinct Genres       797
```

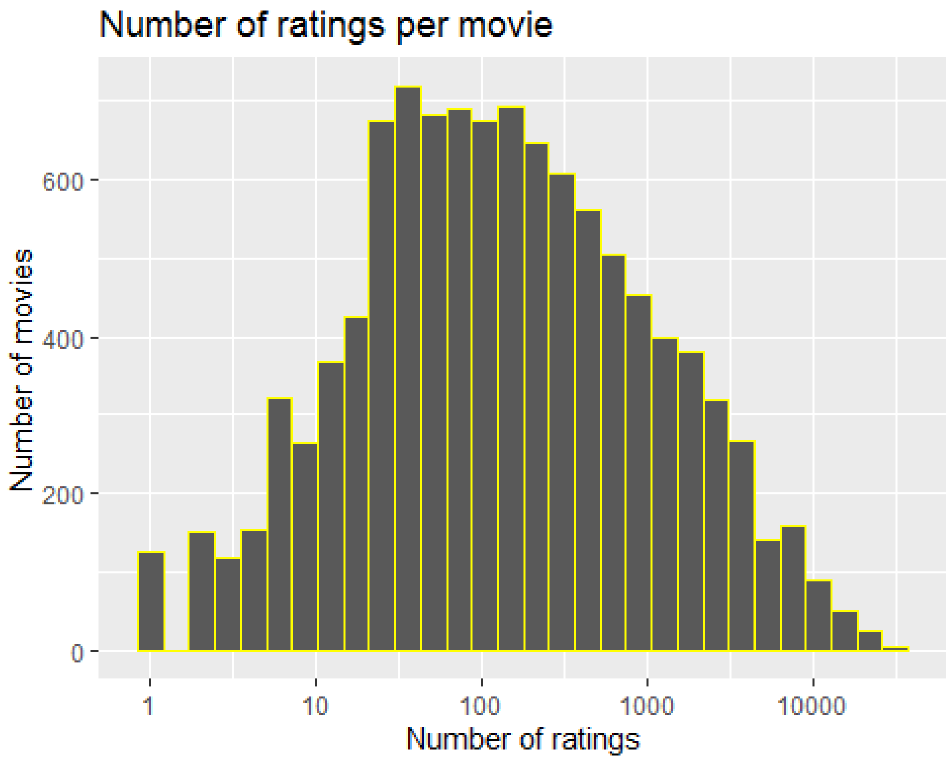
The first thing we notice is that some movies get rated more than others so we need to check their distribution. Another observation is that some users are more active than others at rating movies.

Users have a preference to rate movies rather higher than lower as shown by the distribution of ratings below. A 4 is the most common rating, followed by 3 and 5. A 0.5 is the least common rating. In general, half rating are less common than whole star ratings.



We can observe that some movies have been rated more often than others, while some have very few ratings and sometimes only one rating. This will be important for our model as very low rating numbers might results in untrustworthy estimate for our predictions.

Thus regularization and a penalty term will be applied to the models in this project. Regularization is a technique used to reduce the error by fitting a function appropriately on the given training set and avoid overfitting (the production of an analysis that corresponds too closely or exactly to a particular set of data, and may therefore fail to fit additional data or predict future observations reliably). Regularization is used for tuning the function by adding an additional penalty term in the error function. The additional term controls the excessively fluctuating function such that the coefficients do not take extreme values.

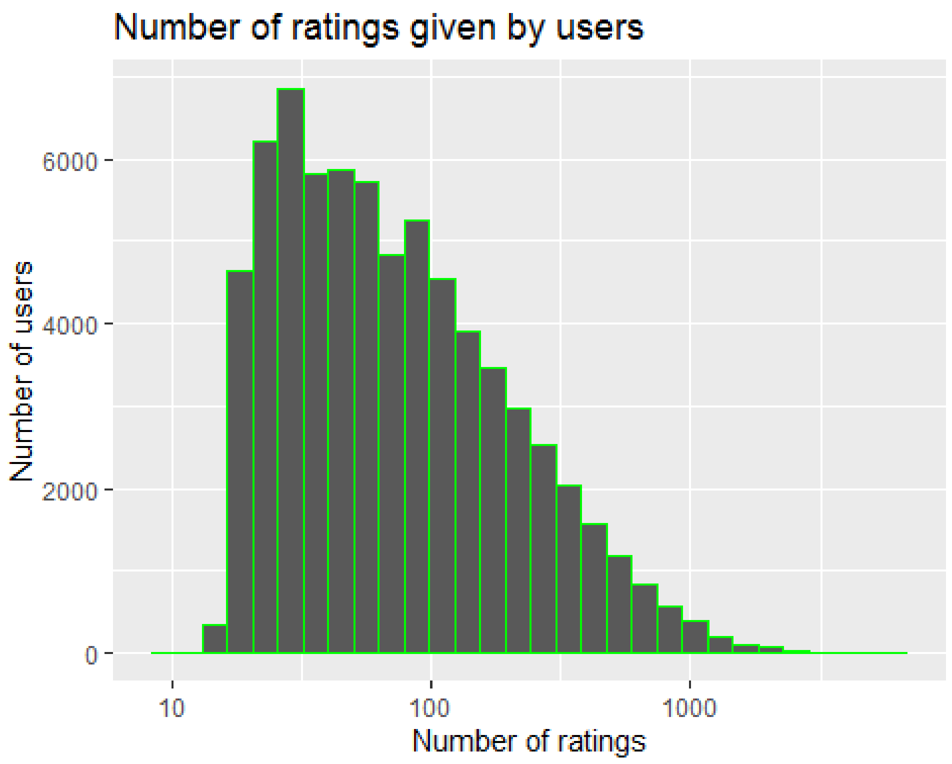


As more than 100 movies that were rated only once appear to be obscure, predictions of future ratings for them will be difficult.

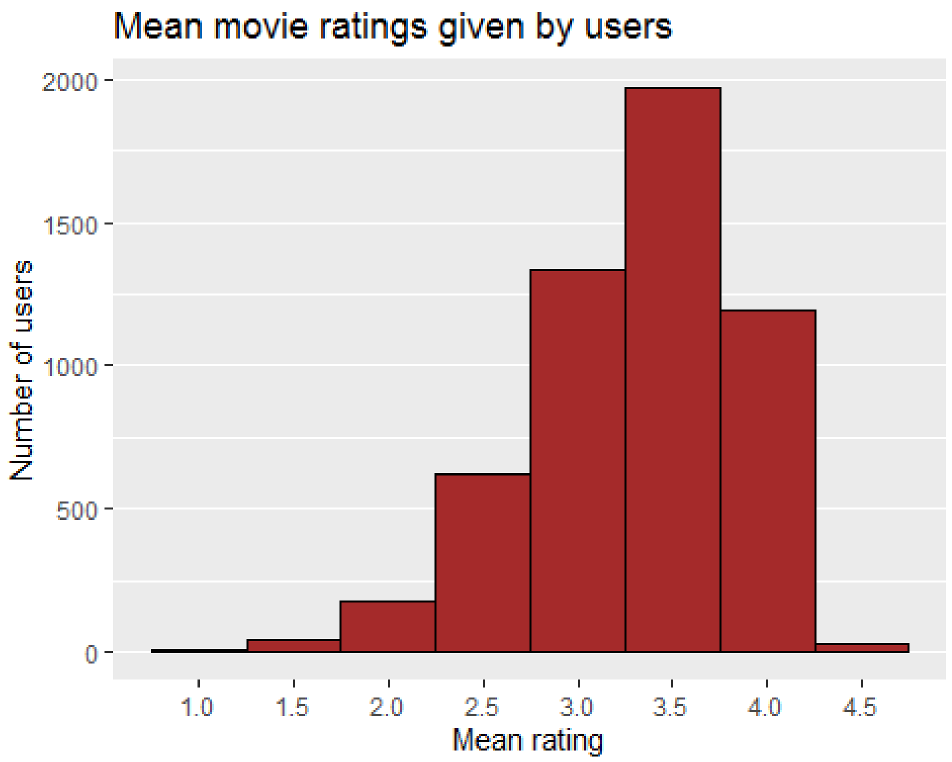
```
## # A tibble: 15 x 3
##    title                                                          rating n_
rating
##    <chr>                                                           <dbl>
<int>
##  1 1, 2, 3, Sun (Un, deuz, trois, soleil) (1993)                     2
1
##  2 100 Feet (2008)                                                   2
1
##  3 4 (2005)                                                          2.5
1
```

```
##  4 Accused (Anklaget) (2005)                                         0.5
1
##  5 Ace of Hearts (2008)                                              2
1
##  6 Ace of Hearts, The (1921)                                         3.5
1
##  7 Adios, Sabata (Indio Black, sai che ti dico: Sei un gran fig~     1.5
1
##  8 Africa addio (1966)                                               3
1
##  9 Aleksandra (2007)                                                 3
1
## 10 Bad Blood (Mauvais sang) (1986)                                   4.5
1
## 11 Battle of Russia, The (Why We Fight, 5) (1943)                    3.5
1
## 12 Bellissima (1951)                                                 4
1
## 13 Big Fella (1937)                                                  3
1
## 14 Black Tights (1-2-3-4 ou Les Collants noirs) (1960)               3
1
## 15 Blind Shaft (Mang jing) (2003)                                    2.5
1
```

We can observe that the majority of users have rated between 30 and 100 movies. So, a user penalty term need to be included later in our models.



Furthermore, users differ greatly in how critical they are with their ratings. Some users tend to give much lower star ratings and some users tend to give higher star ratings than average. The visualization below includes only users that have rated at least 120 movies.



```
## [1] 3.191762

## [1] 0.5713468
```

We find that the mean average movie rating is approximately 3.2 with a standard deviation 0.6

3.2 Methods and data analysis

3.2.1 Recommendation system using machine learning

Noticing that the movie release years are contained in the titles, we extract the year information using the `str_extract` function and save this in a title named `years` containing movie titles and their release years.

```
## # A tibble: 6 x 2
##   title                                  year
##   <chr>                                 <dbl>
## 1 "'burbs, The (1989)"                   1989
## 2 "'night Mother (1986)"                 1986
## 3 "'Round Midnight (1986)"               1986
## 4 "'Til There Was You (1997)"            1997
## 5 "\"Great Performances\" Cats (1998)"  1998
## 6 "*batteries not included (1987)"      1987
```

We join the year column onto our edx training set, and will later join it onto the validation set during testing:

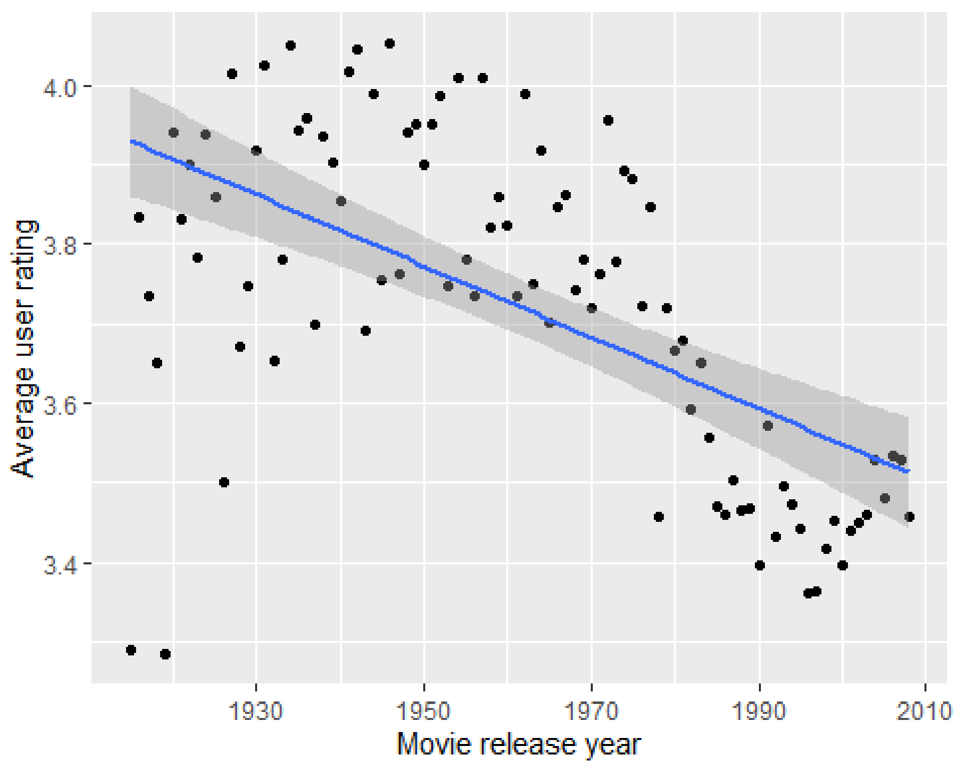
```
##   userId movieId rating timestamp                         title
## 1      1     122      5 838985046              Boomerang (1992)
## 2      1     185      5 838983525               Net, The (1995)
## 3      1     292      5 838983421               Outbreak (1995)
## 4      1     316      5 838983392               Stargate (1994)
## 5      1     329      5 838983392 Star Trek: Generations (1994)
## 6      1     355      5 838984474       Flintstones, The (1994)
##                           genres year
## 1                 Comedy|Romance 1992
## 2          Action|Crime|Thriller 1995
## 3  Action|Drama|Sci-Fi|Thriller 1995
## 4        Action|Adventure|Sci-Fi 1994
## 5 Action|Adventure|Drama|Sci-Fi 1994
## 6        Children|Comedy|Fantasy 1994
```

Before we begin building our algorithm to predict movie ratings, in order to train our algorithm we first split our training set edx_df into a training set containing 80% of entries and a test set containing 20%.

Lets remove the rating column from the test_set:

```
##   userId movieId timestamp                                title
## 1      1     122 838985046                     Boomerang (1992)
## 2      1     292 838983421                      Outbreak (1995)
## 3      1     356 838983653                  Forrest Gump (1994)
## 4      1     362 838984885              Jungle Book, The (1994)
## 5      1     370 838984596 Naked Gun 33 1/3: The Final Insult (1994)
## 6      1     466 838984679             Hot Shots! Part Deux (1993)
##                          genres year
## 1                Comedy|Romance 1992
## 2 Action|Drama|Sci-Fi|Thriller 1995
## 3     Comedy|Drama|Romance|War 1994
## 4   Adventure|Children|Romance 1994
## 5                 Action|Comedy 1994
## 6             Action|Comedy|War 1993
```

We examine the relationship between movie release year and average user rating in the plot below:



3.2.2 Loss function

We are going to use RMSE to represent the loss function:

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

3.2.3 Just the average rating model

```
## [1] 3.512482
```

If we predict all unknown ratings with μ we obtain the following RMSE. We create a results table with this naive approach to give:

```
## # A tibble: 1 x 2
##   Method             RMSE
##   <chr>             <dbl>
## 1 Just the average  1.06
```

3.2.4 Movie effects model

To improve above model we focus on the fact that, from experience, we know that some movies are just generally rated higher than others. Higher ratings are mostly linked to popular movies among users and the opposite is true for unpopular movies. We compute the estimated deviation of each movies' mean rating from the total mean of all movies μ. The resulting variable is called “b” (as bias) for each movie “i” b_i, that represents average ranking for movie i:

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

The histogram is left skewed, implying that more movies have negative effects. We can check how much our prediction improves once we use:

```
## # A tibble: 2 x 2
##   Model             RMSE
##   <chr>            <dbl>
## 1 Just the average 1.06
## 2 Movie Effect     0.944
```

We already seeing some improvement in RMSE but we can do much better than this.

3.2.5 Movie + User effect model

Let’s compute the average rating for user μ for those that have rated over 100 movies:

We can now construct predictors and see how much the RMSE improves:

```
## # A tibble: 3 x 2
##   Model                       RMSE
##   <chr>                      <dbl>
## 1 Just the average           1.06
## 2 Movie Effect Model         0.944
## 3 Movie + User Effect Model 0.866
```

3.2.6 Movie + User + Time effect model

```
## Joining, by = "title"
## Joining, by = "title"

## Joining, by = "userId"
```

We calculate the RMSE for our models:

```
## # A tibble: 4 x 2
##   Model                              RMSE
##   <chr>                             <dbl>
## 1 Just the Average                  1.06
## 2 Movie Effect Model                0.944
## 3 Movie + User Effect Model         0.866
## 4 Movie + User + Time Effect Model 0.866
```

3.3 Regularization

check the reduction in RMSE

```
## [1] 7.8
```

Despite the large movie to movie variation, our improvement in RMSE was only about 7.8%.

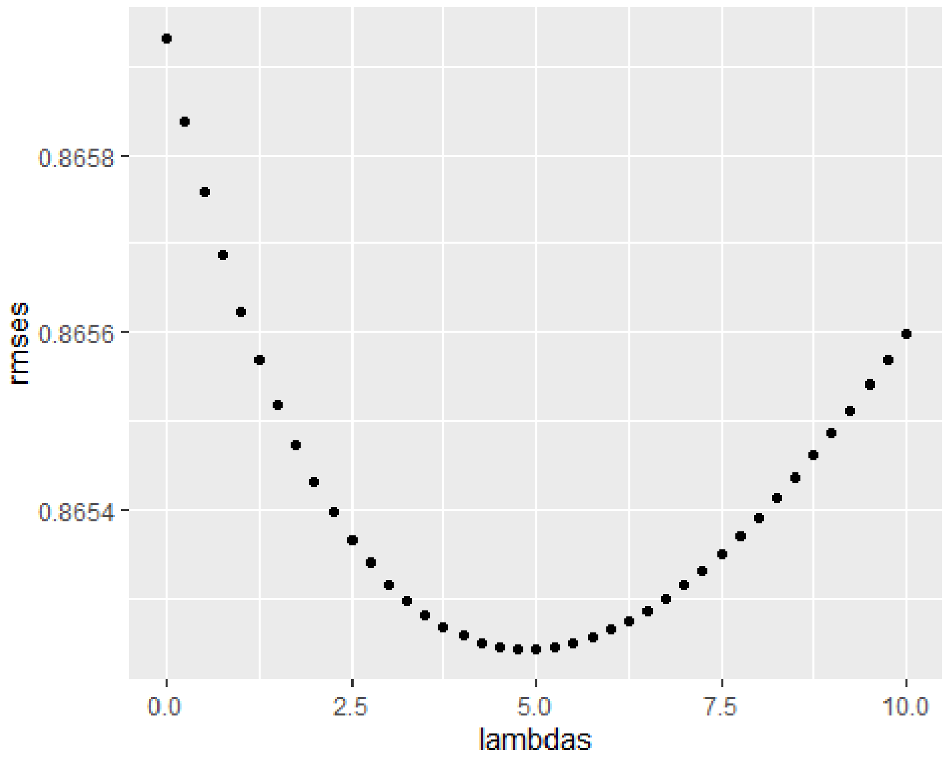
The penalized estimates did not provide a large improvement over the least squares estimates so we apply regularization effect to determine a better RMSE.

3.3.1 Choosing the penalty terms

We have noticed in our data exploration, some users are more actively participated in movie reviewing. There are also users who have rated very few movies (less than 30 movies). On the other hand, some movies are rated very few times (say 1 or 2). These are basically noisy estimates that we should not trust. Additionally, RMSE are sensitive to large errors. Large errors can increase our residual mean squared error. So we must put a penalty term to give less importance to such effect.

Note that λ is a tuning parameter. We can use cross-validation to choose it.

So estimates of b_{movie} and b_{user} are caused by movies with very few ratings and in some users that only rated a very small number of movies. Hence this can strongly influence the prediction. The use of the regularization permits to penalize these two aspects. We should find the value of lambda (that is a tuning parameter) that will minimize the RMSE. This shrinks the b_{movie} and b_{user} in case of small number of ratings.



For the full model, the optimal lambda is: 4.75

```
## [1] 4.75
```

The new results will be:

```
rmse_reg_model
```

```
## # A tibble: 5 x 2
##   Model                                   RMSE
##   <chr>                                  <dbl>
## 1 Just the Average                       1.06 
## 2 Movie Effect Model                     0.944
## 3 Movie + User Effect Model              0.866
## 4 Movie + User + Time Effect Model       0.866
## 5 Regularized Movie + User Effect Model 0.865
```

4.0 Results

4.1 RMSE overview

```
## [1] 0.07
```

The RMSE value for the our regularized model is shown below:

```
## # A tibble: 1 x 2
##   Model                                  RMSE
##   <chr>                                 <dbl>
## 1 Regularized Movie + User Effect Model 0.865
```

5.0 Concluding Remarks

We can affirm to have built a machine learning algorithm to predict movie ratings with MovieLens dataset.

The regularized model including the effect of user is characterized by the lower RMSE value and is hence the optimal model to use for the present project. The optimal model characterized by the lowest RMSE value (0.8652421) lower than the initial evaluation criteria (0.8775) given by the goal of the present project. This value of RMSE is over 19.9% with respect to the first model

We could also affirm that improvements in the RMSE could be achieved by adding other effect such as genre. Other different machine learning models could also improve the results further, but hardware limitations, as the RAM, are a constraint.

6.0 References

- https://github.com/johnfelipe/MovieLens-2
- https://github.com/cmrad/Updated-MovieLens-Rating-Prediction