MovieLens Project - Recommendation System

EXECUTIVE SUMMARY

This project constists in the creation of a Recommendation System using a sample from the publicly available MovieLense dataset included in the dslabs package. The final goal is to predict the rating that a user would give to a given movie. The sample consist of a set with 7 columns and 9M rows. Each row contains the rating of a movie by a given user, plus the title and genre of that movie. The process consisted in 4 steps:

1) Understanding of the data
2) Splitting the data set into test and train sets
3) Training and testing models
4) Final validation of the selected model in a separated set

The final model was created by applying Matrix Factorization using the "recosystem" library

Creation of the datasets. Code provided by edx:

```
## Loading required package: tidyverse

## -- Attaching packages --------------------------------------- tidyverse 1.3.0 --

## v ggplot2 3.3.3     v purrr   0.3.4
## v tibble  3.0.5     v dplyr   1.0.3
## v tidyr   1.1.3     v stringr 1.4.0
## v readr   1.4.0     v forcats 0.5.0

## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()

## Loading required package: caret

## Loading required package: lattice

##
## Attaching package: 'caret'

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

## Loading required package: data.table

##
## Attaching package: 'data.table'
## The following objects are masked from 'package:dplyr':
##
##     between, first, last

## The following object is masked from 'package:purrr':
##
##     transpose

## Warning in set.seed(1, sample.kind = "Rounding"): non-uniform 'Rounding' sampler
## used

## Joining, by = c("userId", "movieId", "rating", "timestamp", "title", "genres")
```

ANALYSIS

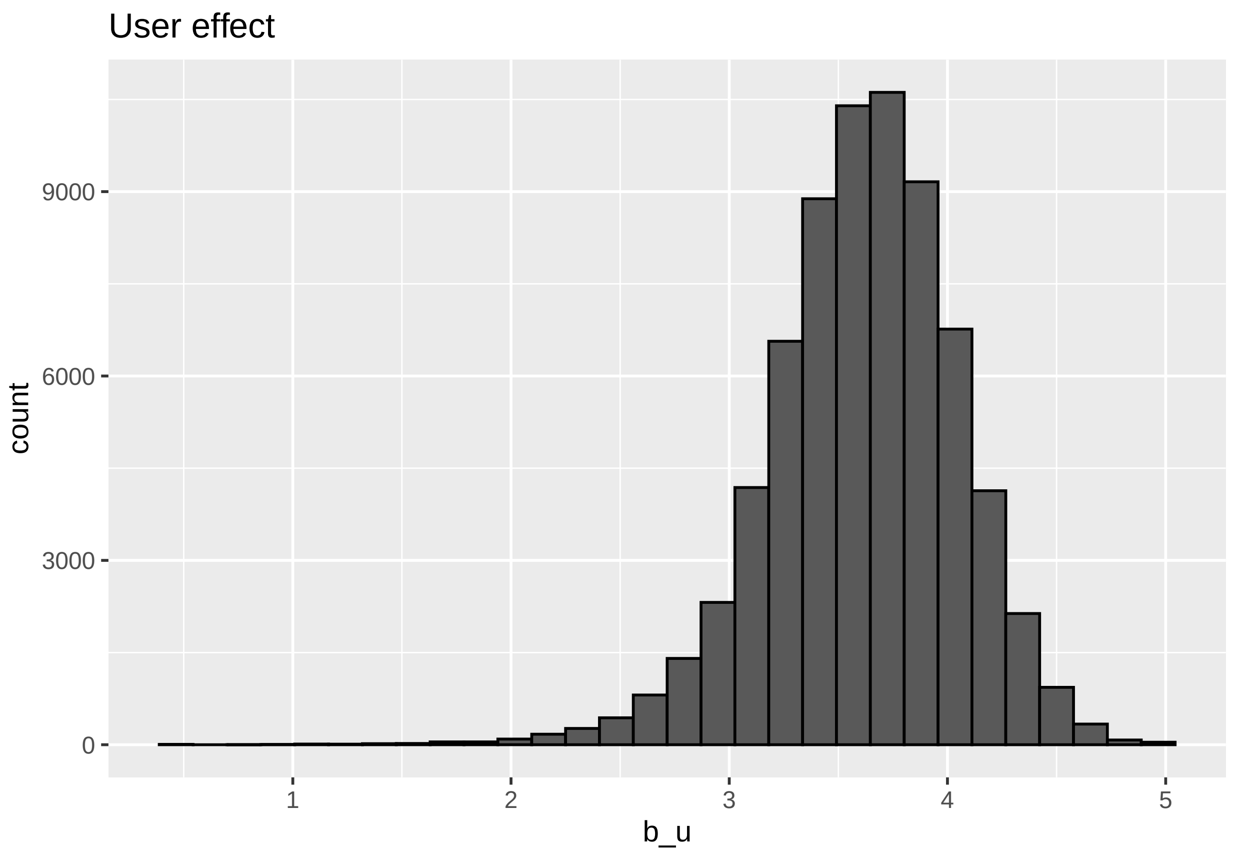
1) Understanding the data

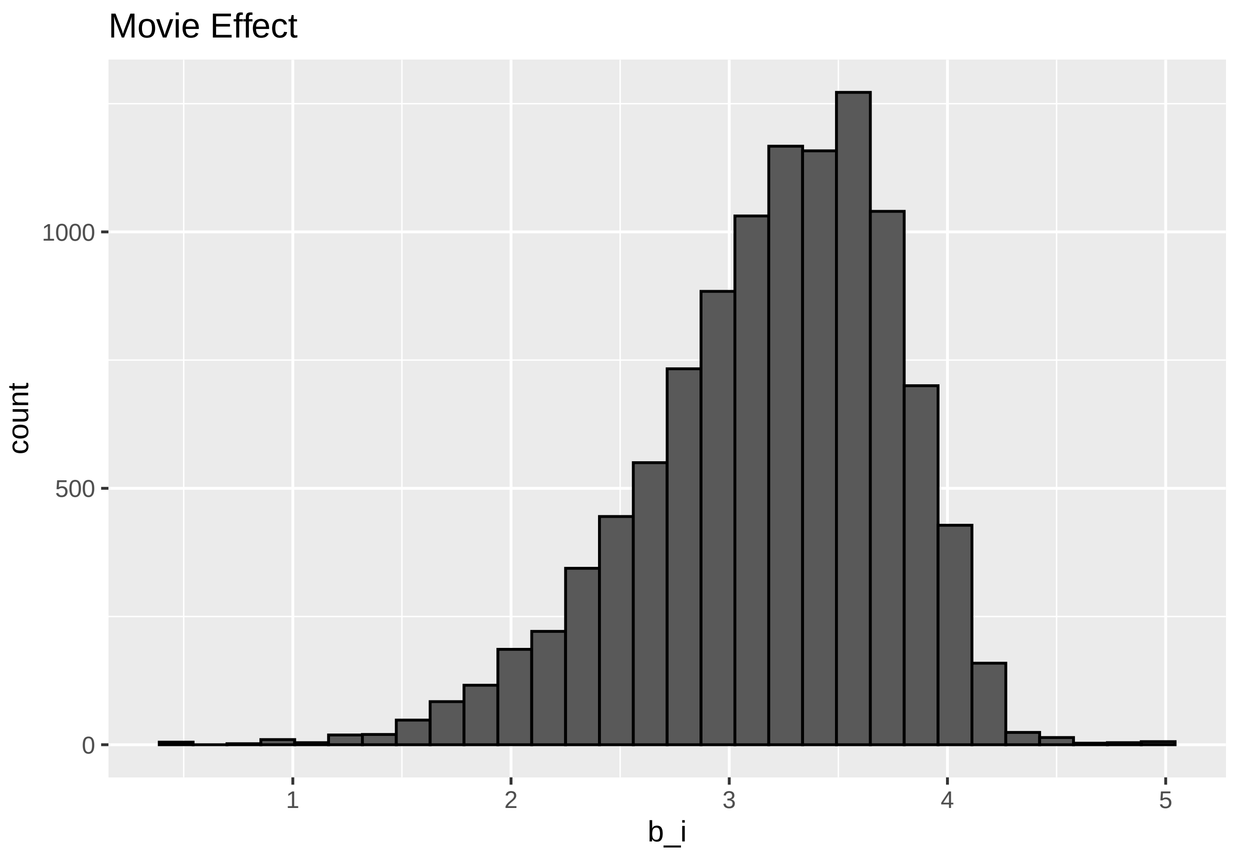
The first step is to quickly see what data and variables we have available

```
##    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
## 1:                Comedy|Romance
## 2:         Action|Crime|Thriller
## 3:  Action|Drama|Sci-Fi|Thriller
## 4:       Action|Adventure|Sci-Fi
## 5: Action|Adventure|Drama|Sci-Fi
## 6:       Children|Comedy|Fantasy
```

We know that we have several features to try to predict the rating: userID, MovieID, Title, Timestamp and Genre.

We want to see how strong is the user and movie effect on the ratings.


User effect
count
9000
6000
3000
0
1
2
3
4
5
b_u



We see that the user and movie effects are strong, and must be taken into consideration when building the model.

2) Splitting the data set into test and train sets

We will split the edx set into train set (20% of the edx set) and test set (80% of the edx set). Each model will used only these two datasets. The final model will be tested in the validation set created using the code provided by edx.

To compare different models or to see how well we're doing compared to some baseline, we need to quantify what it means to do well.

We build a function that computes this residual means squared error for a vector of ratings and their corresponding predictors

We will be comparing different approaches, We're going to create a table that's going to store the results that we obtain.

```
## [1] 1.466085
```

3) Training and testing models

We will use the methods used during the course, in which users and movies effects are taking into account for rating prediciton.

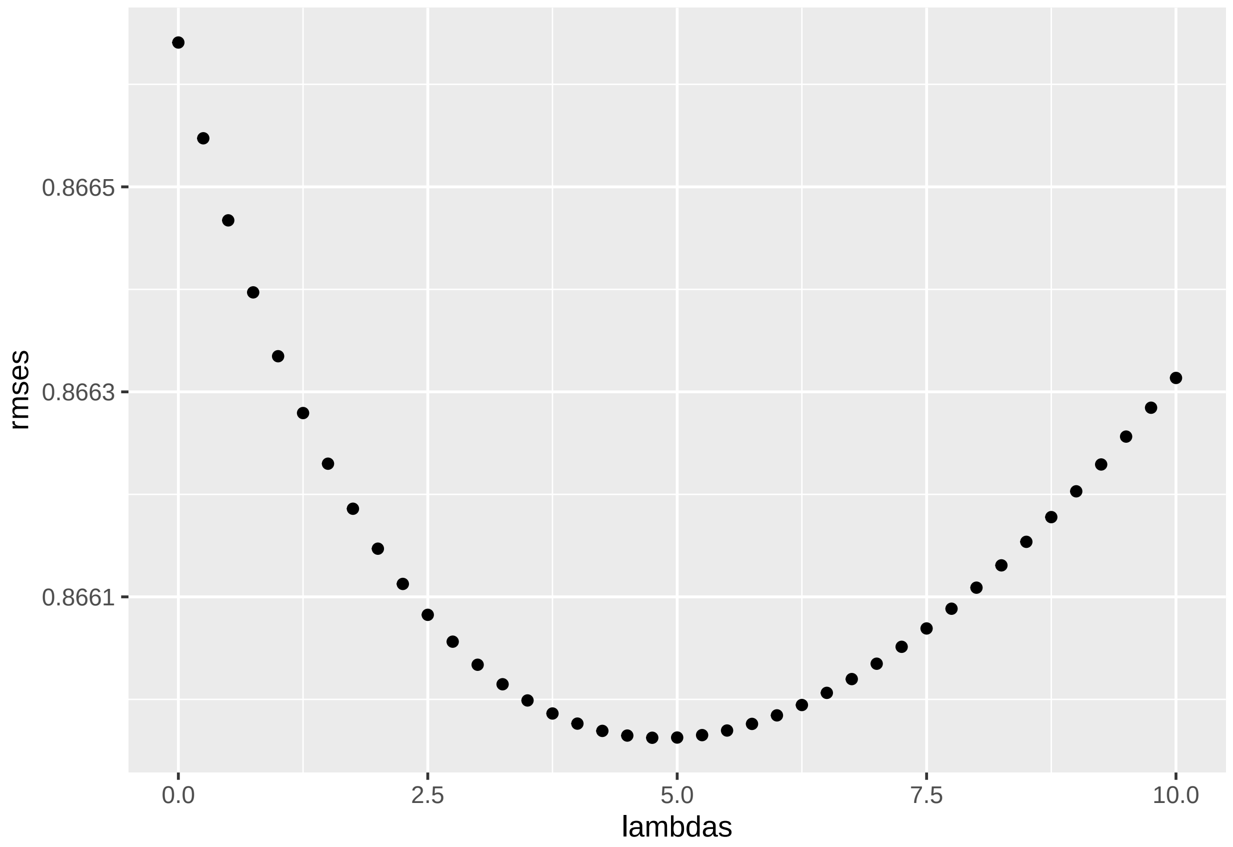
```
## Warning: ‘data_frame()‘ is deprecated as of tibble 1.1.0.
## Please use `tibble()` instead.
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_warnings()` to see where this warning was generated.
```

```
## # A tibble: 3 x 2
##   method                      RMSE
##   <chr>                      <dbl>
## 1 Just the average           1.06
## 2 Movie Effect Model         0.944
## 3 Movie + User Effects Model 0.867
```

The RMSE represents the error of our prediction. Using just the average, a prediction could "fail" by 1.06 points in the rating, which is very large.

By taking into account the user effect, this error decreases to .94, and when adding both user and movie effect it goes down to 0.866.

An additional method is to consider the size of the samples. This means that the variability is contrained by penalizing predictions that are estimated using small sample sizes. As we do not know how much to "penalize" the samples, we run a test with different values to find the optimal one, in which the rmse is minimized.



We see that the rmse is minimized at 4.75 value of lambda, being lambda de factor that penalizes the sample sizes.

We see that the RMSE decreases even more down to 0.865.

method	RMSE
Just the average	1.0605613
Movie Effect Model	0.9439868
Movie + User Effects Model	0.8666408
Regularized Movie + User Effect Model	0.8659626

Finally, we will try Matrix Factorization method.

There are "hidden" features more than the user and movie effect. Some users might like some time of movies because of many reasons: actors, genre, lenght, year, etc. We cannot get this only by looking at userid and movieid columns.

The Matrix Factorization method will allow us to create features that explain the variability. In this method, there are 2 main parameters that we can define: the number of factors and the number of iterations. Due to the heavy load of work for a home PC, I manually tested 3 different values of each and selected the optimal one. This resulted in 50 factors and 500 iterations.

```
## # A tibble: 5 x 2
##   method                                 RMSE
##   <chr>                                 <dbl>
## 1 Just the average                      1.06 
## 2 Movie Effect Model                    0.944
## 3 Movie + User Effects Model            0.867
## 4 Regularized Movie + User Effect Model 0.866
## 5 Matrix Factorization 50 dim           0.810
```

Using this method, the RMSE goes down to 0.809, the lowest of them all.

RESULTS

Here, we will see the final result of our model in the Validation set, which has not been used in any previous analysis.

4) Final validation of the selected model in a separated set

We selected the Matrix Factorization method as the optimal one. Now we must use this in the Validation set, to see how well it performs in new data.

The RMSE in this dataset is:

```
## [1] 0.8094506
```

We can see how the RMSE is still below 0.81 even in the validation set.

This means that the model is correct and reaches the goal of an RMSE below 0.8649

CONSLUSIONS

From the results of the different methods we can determine that Matrix Factorization is a powerful method useful when the data can have many features that explain the variability of the searched values.

As the existing variables in the dataset do not capture the whole variability (user id, movie id), it is necessary to use this method that allows us to ucapture patterns as factors to predict the searched values.