Capstone: Movielens Project

Suhaimi William Chan

9/15/2020

1. Introduction

Recommendation systems plays an important role in e-commerce and online streaming services, such as Netflix, YouTube, Amazon, etc. Making the right recommendation for the next product/service, music or movie increases user retention and satisfaction, leading to product/service sales, thus increasing profit growth. Companies competing for customer loyalty invest on recommendation systems that capture and analyze the user's preferences, and offer products/services with higher likelihood of purchase.

The economic impact of such company-customer relationship is very clear. Amazon is the largest online retail company by sales and part of its success comes from the recommendation systems and marketing based on user preferences. In 2006 Netflix offered a one million dollar prize for the person or group that could improve their recommendation system by at least 10%.

Most recommendation systems are based on a rating scale from 1 to 5 grades or stars, with 1 indicating lowest satisfaction and 5 is the highest satisfaction. Other indicators can also be used, such as comments posted on previously used items; videos, musics or links shared with friends; percentage of movie watched or music listened; web pages visited and time spent on each page; product category; and any other interactions with the company's web site or application can be used as a predictor.

The primary goal of recommendation systems is to help users find what they want based on their preferences and previous interactions, and predicting the rating for a new item. In this document, we create a movie recommendation system using the MovieLens dataset and applying the courses/lessons learned during the HarvardX's Data Science Professional Certificate program.

This document is structured as follows: Chapter 1 describes the dataset and summarizes the goal of the project and key steps that were performed. Chapter 2 we explain the method, process and techniques used, such as data cleaning, data exploration and visualization, any insights gained, and the modeling approach. Chapter 3 we present the modeling analysis. Chapter 4 We discuss the results and model performance. Chapter 5 We conclude with a brief summary of the report, its limitations and future work.

The goal of this movielens project is to predict the movie ratings by users as close as the true ratings in the validation set using RMSE. Our goal is to get our final predicted RMSE < 0.86490

We are going to use the following library:

```r
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")

library(tidyverse)
library(caret)
library(data.table)
```

2. Data wrangling/data preparation and data exploration

Movielens Project is 10M dataset from [Movielens] (http://files.grouplens.org/datasets/movielens/ml-10m.zip).
The zip file data were downloaded into our local computer system. There were two set of data files from the downloaded zip file. One is ratings.dat that contains userId, movieID, rating, and timestamp data. The other one is movies.dat that contains movieId, title, and genres data.

We stored the second data set of movies in data frame movies.
Then we stored the first data set of ratings in data frame movielens that we used left join to data frame movies by movieId.
Then we partitioned movielens into edx and temp with 90%/10% split, respectively.
We created a validation data set using data frame temp that semi join to edx by userId and movieId to make sure that userId and movieId in validation set are also in edx set. Then we add rows removed from validation set back into edx set.

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

First, we are going the check dimension of our edx data set.

```
## [1] 9000055       6
```

Second, we are going the check dimension of our validation data set.

```
## [1] 999999      6
```

After making sure the data source looks good, now we are going to split edx data set into our train_set and test_set with 90/10 ratio, respectively.

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

Now, we are going the check dimension of our train_set. (about 90% of edx data set)

```
## [1] 8100065       6
```

Then, we are going the check dimension of our test_set. (about 10% of edx data set)

```
## [1] 899990      6
```

To see a more complete data set, we are going to do our data exploration using edx data set, instead of using training set. We can see some examples of our edx data set with available columns.

```
##    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
```

First, we are going to check the classes of our edx data set columns

```
## Classes 'data.table' and 'data.frame':   9000055 obs. of  6 variables:
##  $ userId   : int  1 1 1 1 1 1 1 1 1 1 ...
##  $ movieId  : num  122 185 292 316 329 355 356 362 364 370 ...
##  $ rating   : num  5 5 5 5 5 5 5 5 5 5 ...
##  $ timestamp: int  838985046 838983525 838983421 838983392 838983392 838984474 838983653 838984885 83
##  $ title    : chr  "Boomerang (1992)" "Net, The (1995)" "Outbreak (1995)" "Stargate (1994)" ...
##  $ genres   : chr  "Comedy|Romance" "Action|Crime|Thriller" "Action|Drama|Sci-Fi|Thriller" "Action|Ad
##  - attr(*, ".internal.selfref")=<externalptr>
```

Now we can explore our edx data set. Let's check the number of unique movies and users in edx data set

```
##   n_users n_movies
## 1   69878    10677
```

Let's see some examples of top movie genre list in edx data set

```
## # A tibble: 6 x 2
##   genres                    n
##   <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
```

Let's see the number of some popular genre movies in edx data set

```
##    Action Adventure  Children    Comedy     Drama   Romance    Sci-Fi  Thriller
##   2560545   1908892    737994   3540930   3910127   1712100   1341183   2325899
```

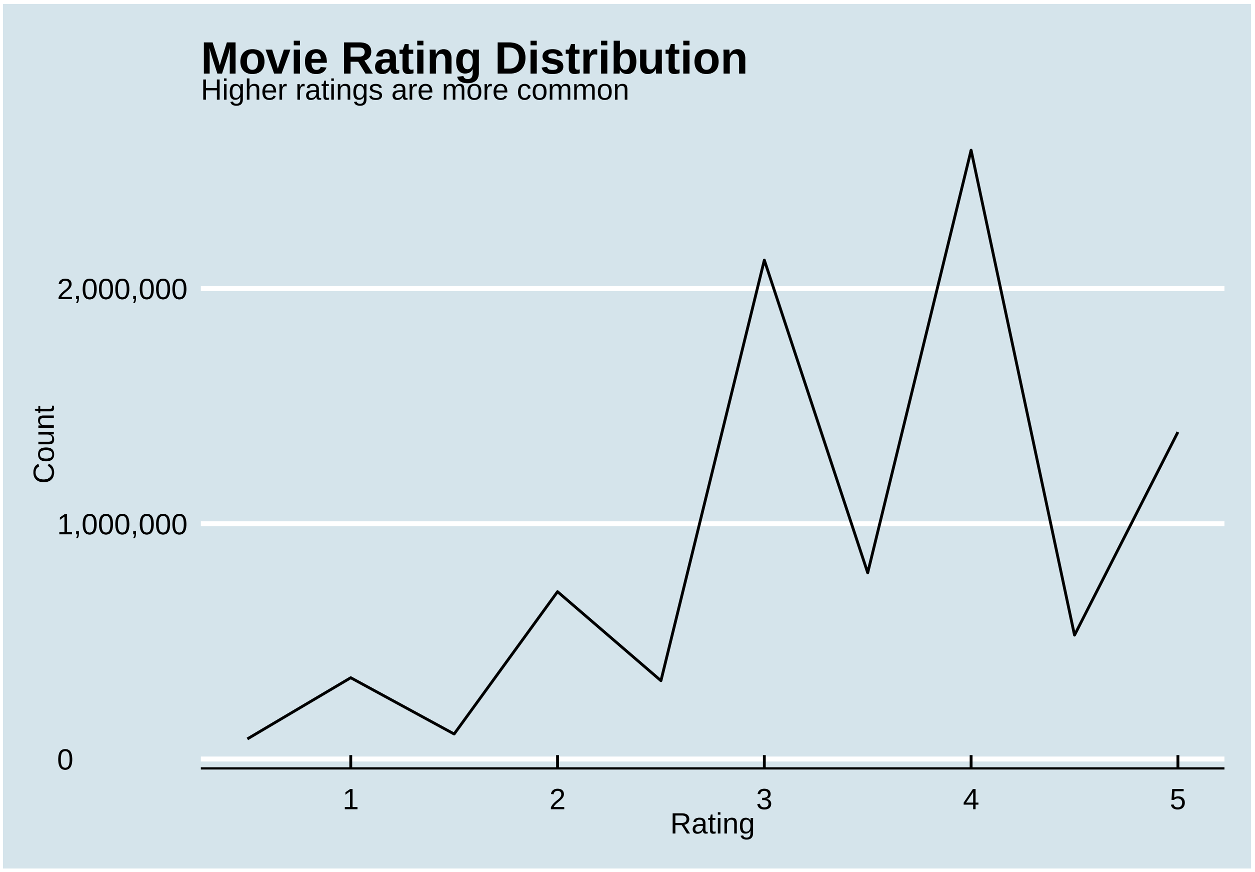
Let's check out some most rated movies in edx data set

```
## # A tibble: 10,677 x 3
## # Groups:   movieId [10,677]
##    movieId title                                                        count
##      <dbl> <chr>                                                        <int>
##  1     296 Pulp Fiction (1994)                                          31362
##  2     356 Forrest Gump (1994)                                          31079
##  3     593 Silence of the Lambs, The (1991)                             30382
##  4     480 Jurassic Park (1993)                                         29360
##  5     318 Shawshank Redemption, The (1994)                            28015
##  6     110 Braveheart (1995)                                           26212
##  7     457 Fugitive, The (1993)                                        25998
##  8     589 Terminator 2: Judgment Day (1991)                           25984
##  9     260 Star Wars: Episode IV - A New Hope (a.k.a. Star Wars) (1977) 25672
## 10     150 Apollo 13 (1995)                                            24284
## # ... with 10,667 more rows
```

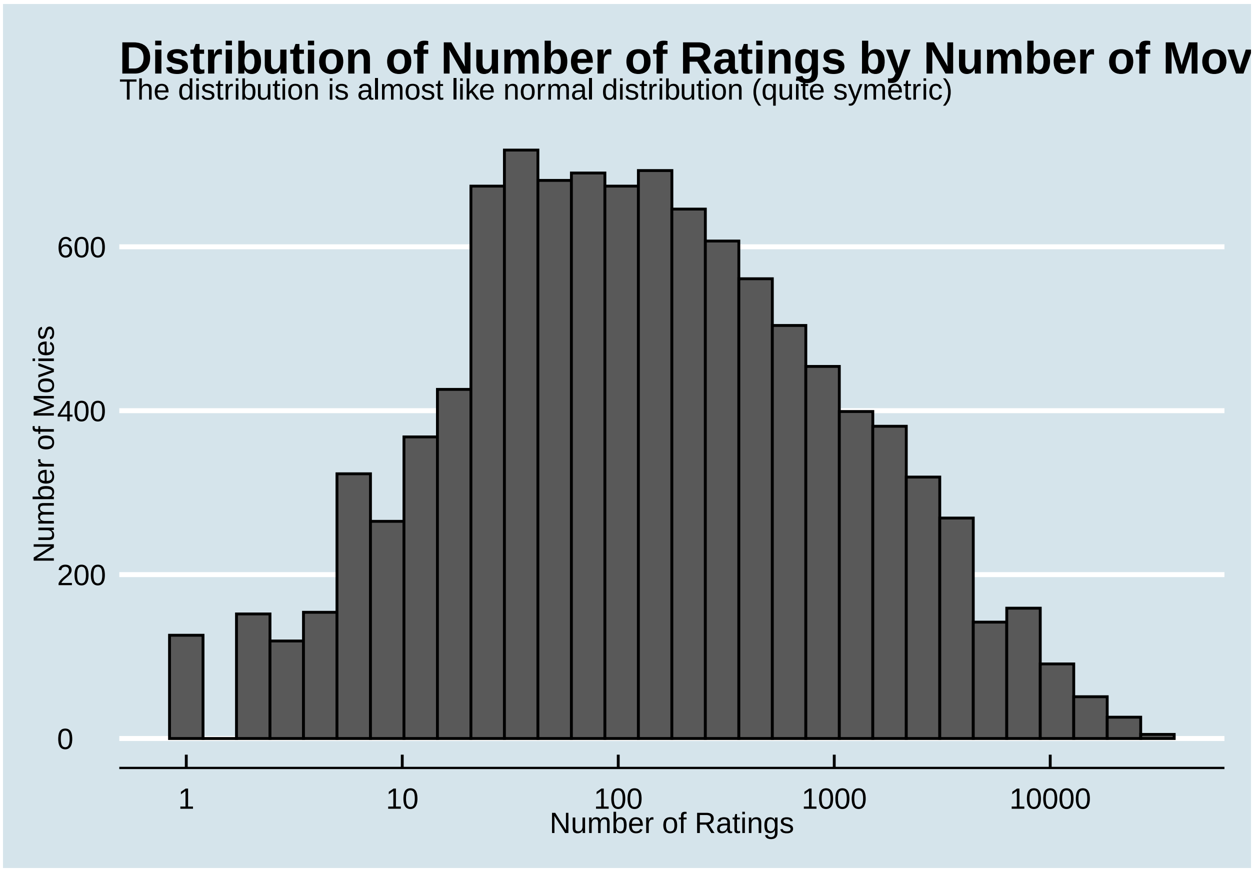
Let's check out the ratings of all movies in edx data set

```
## # A tibble: 10 x 2
##    rating       n
##     <dbl>   <int>
##  1    5   1390114
##  2    4.5  526736
##  3    4   2588430
##  4    3.5  791624
##  5    3   2121240
##  6    2.5  333010
##  7    2    711422
##  8    1.5  106426
##  9    1    345679
## 10    0.5   85374
```

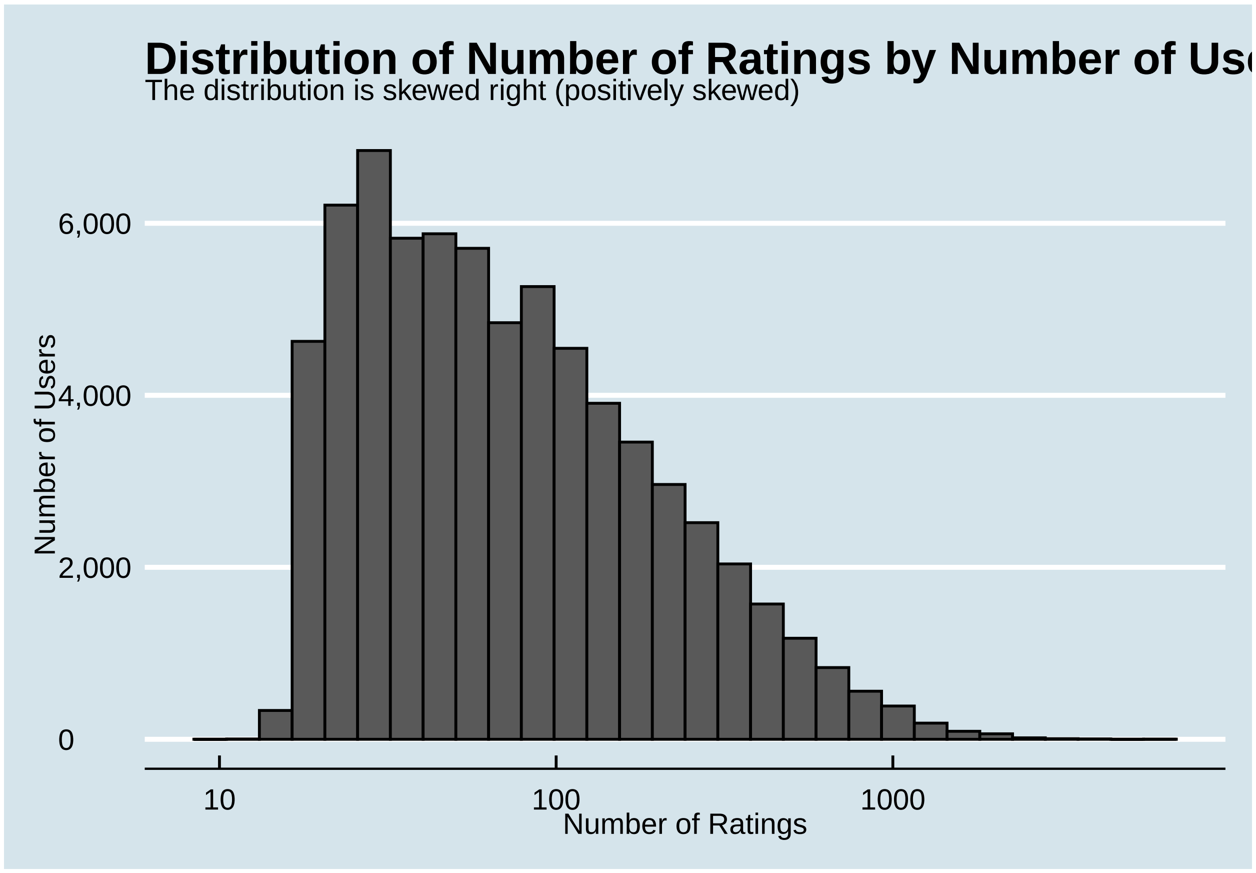
Let's visually see the ratings distribution of all movies in a chart



Let's visually see the distribution of number of ratings by number of movies



Let's visually see the distribution of number of ratings by number of users



3. Analysis

Now, we are going to create our model evaluation function which is also known as loss function: The residual mean square error (RMSE). Later, We will decide the best algorithm trained on the training set using the residual mean squared error (RMSE) with cross-validation on the validation set at the end.

We know that the estimate that minimizes the RMSE is the least squares estimate of mu and, in this case, is the average of all ratings in our training set:

```
## [1] 3.512456
```

The average of all ratings (mu) is 3.512456

Our first model is just the average model RMSE. The formula is Yu,i = mu + Eu,i We obtain the following RMSE:

```
## [1] 1.060054
```

Just the average model RMSE is 1.06 (our first model)

We are going to store all our RMSE goal in rmse_result tibble

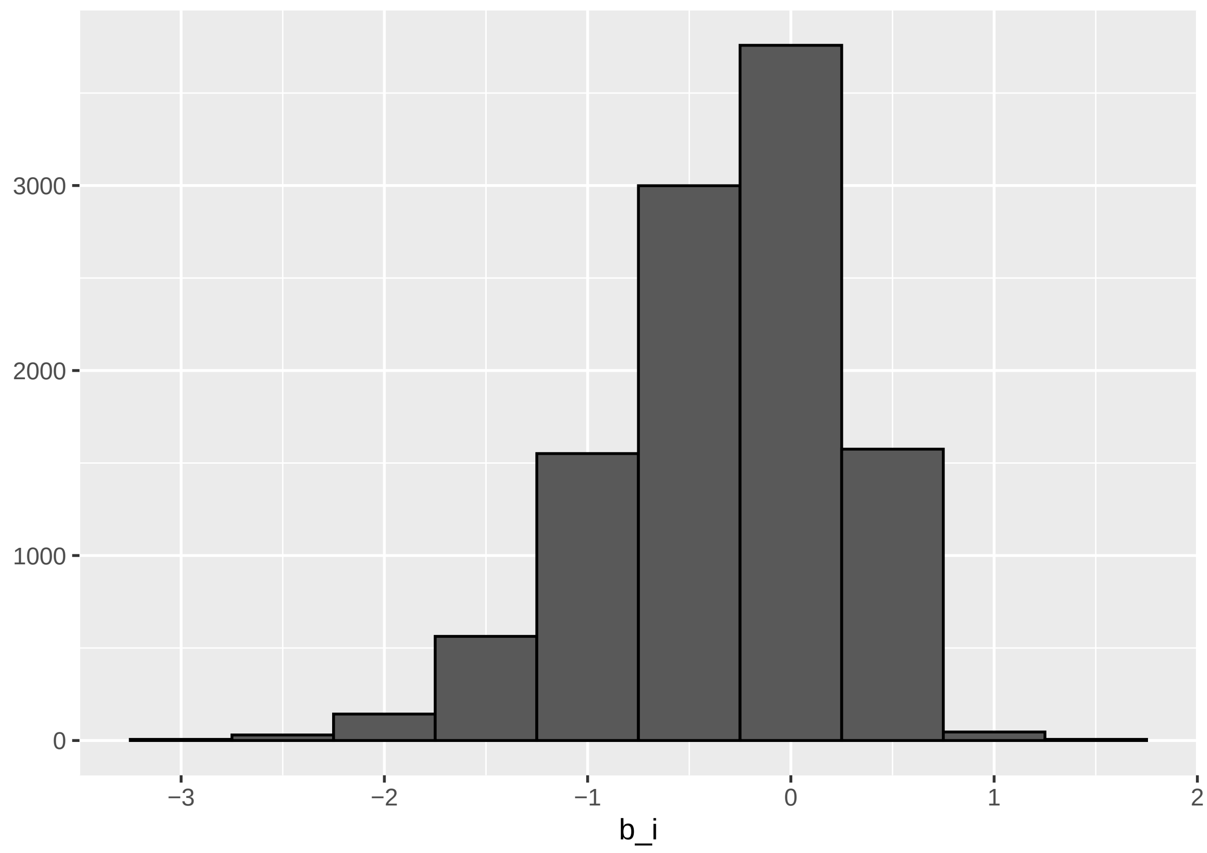
```
## # A tibble: 1 x 2
##   method      RMSE
##   <chr>      <dbl>
## 1 RMSE Goal 0.8649
```

We are going to store all our model RMSE in rmse_result tibble to compare to our RMSE goal of 0.86490

```
## # A tibble: 2 x 2
##   method               RMSE
##   <chr>               <dbl>
## 1 RMSE Goal        0.8649
## 2 Just the average 1.060054
```

Our second model: modeling movie effect formula will be adding movie bias as follow: Yu,i = mu + b_i + Eu,i

We can see that these estimates vary substantially in the following chart:



Remember mu = 3.5 so a b_i = 1.5 implies a perfect five star rating. Let’s see how much our prediction improves once we use yu,i = mu + b_i:

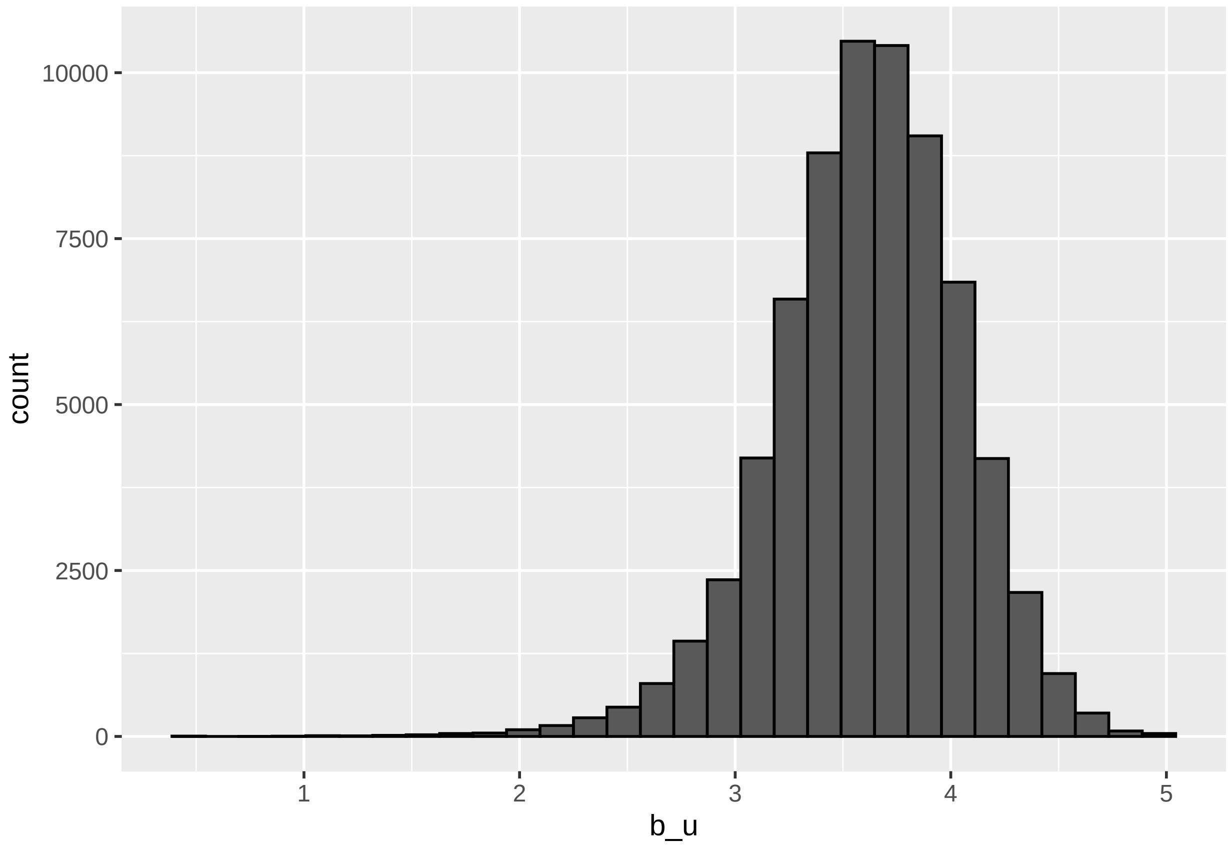
```
## [1] 0.9429615
```

The Movie effect model RMSE is 0.944 (our second model)

Here is the results of our method and RMSE so far:

```
## # A tibble: 3 x 2
##   method                  RMSE
##   <chr>                  <dbl>
## 1 RMSE Goal          0.8649
## 2 Just the average   1.060054
## 3 Movie effect model 0.9429615
```

For our third model, Movie + User effect model, let's start with computing the average rating for user u for those that have rated over 100 movies. Here is the chart plot:



Our third model: modeling movie + user effect formula will be adding user bias as follow: Yu,i = mu + b_i + b_u + Eu,i We will compute an approximation by computing mu and b_i and estimating b_u as the average of yu,i - mu - b_i. Then We can now construct predictors and see how much the RMSE improves using our User effect model:

```
## [1] 0.8646843
```

Movie + User Effect model RMSE is 0.8646843 (our third model)

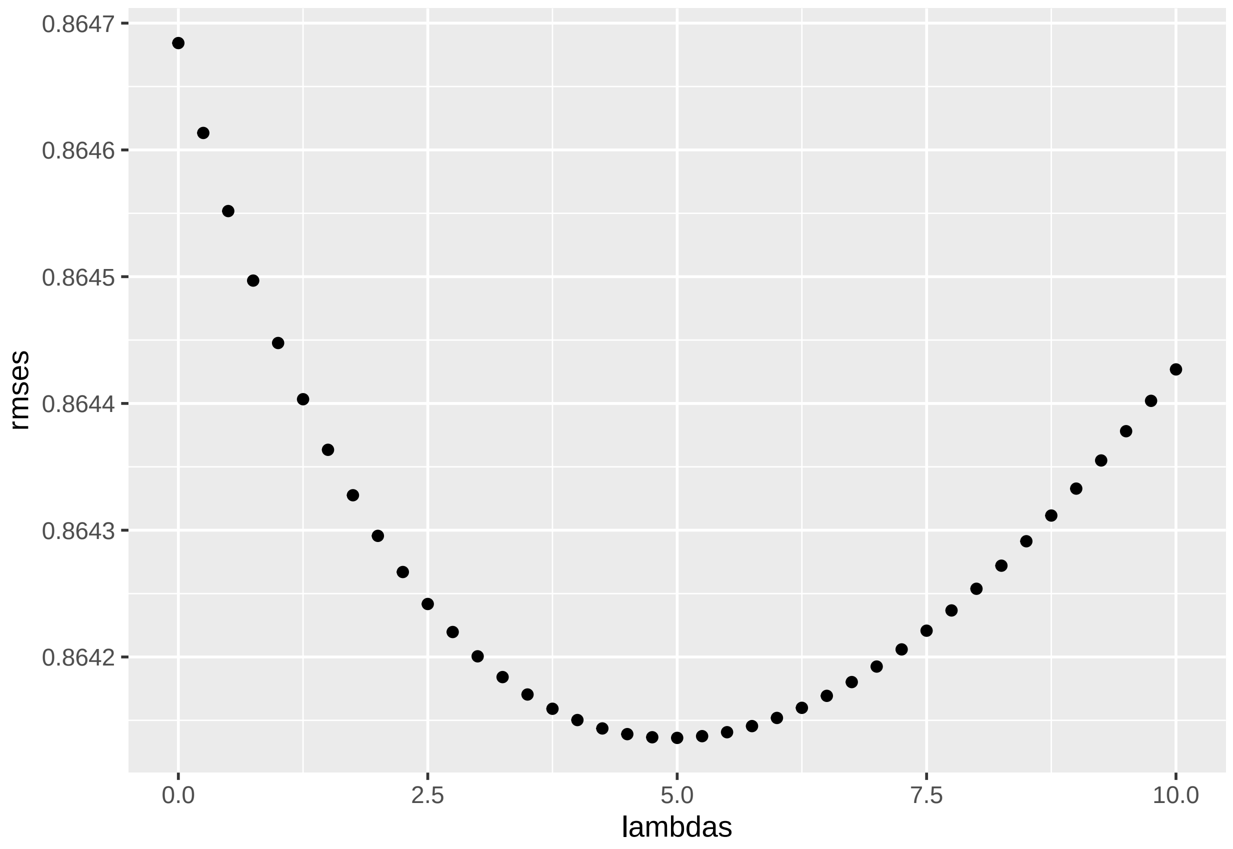
Here is the results of our method and RMSE so far:

```
## # A tibble: 4 x 2
##   method                         RMSE
##   <chr>                         <dbl>
## 1 RMSE Goal                 0.8649
## 2 Just the average          1.060054
## 3 Movie effect model        0.9429615
## 4 Movie + User Effect model 0.8646843
```

For our fourth model, we are going to add regularization to improve our predicted RMSE. We can use regularization for the estimate movie + user effects as well. We are minimizing: Sum ((yu,i - mu - bi - bu)^2 + lambda(sum(bi^2) + sum(bu^2)))

Our fourth model: Regularized Movie + User Effect Model

Choosing the best penalty terms. Lambda is a tuning parameter. We can use cross-validation to choose it to get our optimized lambda. Here is the plot of lambda by increment of 0.25 from 0 to 10, and optimized lambda value.



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

Using our training set and cross-validating it with our test set, we get our optimized lambda at 5.

Let's compute our regularized movie + user effect model.

```
## [1] 0.8641362
```

Regularized Movie + User Effect Model RMSE is 0.8641362 (our fourth model)

Here is the results of our method and RMSE so far:

```
## # A tibble: 5 x 2
##   method                                  RMSE
##   <chr>                                  <dbl>
## 1 RMSE Goal                            0.8649
## 2 Just the average                     1.060054
## 3 Movie effect model                   0.9429615
## 4 Movie + User Effect model            0.8646843
## 5 Regularized Movie + User Effect model 0.8641362
```

Our goal is final model RMSE < 0.86490, so we meet the goal with our final model RMSE as 0.8641362 < 0.86490

Let's compute our final regularized movie + user effect model on edx data set and cross-validate it using validation set.

```
## [1] 0.8648177
```

Here is the results of all our methods and RMSEs:

```
## # A tibble: 6 x 2
##   method                                                          RMSE
##   <chr>                                                          <dbl>
## 1 RMSE Goal                                                     0.8649
## 2 Just the average                                              1.060054
## 3 Movie effect model                                            0.9429615
## 4 Movie + User Effect model                                     0.8646843
## 5 Regularized Movie + User Effect model                         0.8641362
## 6 Final Regularized Movie + User Effect model (edx vs validation) 0.8648177
```

Our goal is final model RMSE < 0.86490, as 0.8648177 < 0.86490 so we meet the goal with our final model RMSE using edx data set with cross-validation using validation data set,

4. Results

Our first model: Just the average model has RMSE at 1.06 The formula is Yu,i = mu + Eu,i This sets as our baseline

Our second model: movie effect model has RMSE at 0.944
The formula is Yu,i = mu + b_i + Eu,i The Movie effect model RMSE has made a little improvement over our first baseline model, just the average model (RMSE 0.944 < RMSE 1.06)

Our third model: movie + user effect model has RMSE at 0.865
The formula is Yu,i = mu + b_i + b_u + Eu,i The Movie + User Effect model RMSE had made a better improvement over our second model, Movie Effect Model (RMSE 0.865 < RMSE 0.944)

Our fourth model: Regularized Movie + User Effect Model has RMSE at 0.864817 The formula is Sum ((yu,i - mu - bi - bu)^2 + lambda(sum(bi^2) + sum(bu^2))) Regularized Movie + User Effect Model RMSE has made the best improvement over our third model, Movie + User Effect Model (RMSE 0.864817 < RMSE 0.865)

Our goal is to have a final RMSE < 0.86490, so our final model, Regularized Movie + User Effect Model has met our goal with RMSE 0.864817 < goal RMSE 0.86490

Here is the results of all our methods and RMSEs:

```
## # A tibble: 6 x 2
##   method                                                          RMSE
##   <chr>                                                          <dbl>
## 1 RMSE Goal                                                     0.8649
## 2 Just the average                                              1.060054
## 3 Movie effect model                                            0.9429615
## 4 Movie + User Effect model                                     0.8646843
## 5 Regularized Movie + User Effect model                         0.8641362
## 6 Final Regularized Movie + User Effect model (edx vs validation) 0.8648177
```

Our goal is final model RMSE < 0.86490, as 0.864817 < 0.86490 so we meet the goal with our final model RMSE using edx data set with cross-validation using validation data set,

5. Conclusion

Summary of the results:

```
## # A tibble: 6 x 2
##   method                                                           RMSE
##   <chr>                                                           <dbl>
## 1 RMSE Goal                                                   0.8649
## 2 Just the average                                            1.060054
## 3 Movie effect model                                          0.9429615
## 4 Movie + User Effect model                                   0.8646843
## 5 Regularized Movie + User Effect model                       0.8641362
## 6 Final Regularized Movie + User Effect model (edx vs validation) 0.8648177
```

The Goal to beat RMSE 0.86490 has been met using Regularized Movie + User Effect Model. With that results, we can say that the final model can predict the movie ratings as close as the true ratings in the validation set.

Limitations

1. Some machine learning algorithms are computationally expensive to run in a commodity laptop. The required amount of memory to compute far exceeded the available memory in a commodity laptop, even with increased virtual memory.
2. Only two predictors are used, the movie and user data, not considering other features. Modern recommendation system models use many predictors, such as genres, bookmarks, playlists, etc.
3. The model works only for existing users, movies and rating values, so the algorithm must run every time a new user or movie is included, or when the rating changes. This is not an issue for small client base and a few thousand movies, but may become a concern for large data sets. The model should consider these changes and update the predictions as information changes.
4. There is no initial recommendation for a new user or for users that usually don't rate movies. Algorithms that uses several features as predictors can overcome this issue.

Future Work This report briefly describes simple models that predicts ratings. There are a few widely adopted approaches not discussed here: matrix-factorization, content-based and collaborative filtering.

The recommenderlab package implements these methods and provides an environment to build and test recommendation systems.

Besides recommenderlab, there are other packages for building recommendation systems available in The Comprehensive R Archive Network (CRAN) website.

References

Rafael A. Irizarry (2019), Introduction to Data Science: Data Analysis and Prediction Algorithms with R

https://www.edx.org/professional-certificate/harvardx-data-science

https://www.netflixprize.com/

https://grouplens.org/

https://movielens.org/

https://grouplens.org/datasets/movielens/latest/

https://grouplens.org/datasets/movielens/10m/

https://cran.r-project.org/web/packages/available_packages_by_name.html