Report on MovieLens

Jo Leung

28/12/2020

Introduction

This project showcases different techniques learnt from the HarvardX Data Science program. The objective of this project is to create a movie recommendation system using the MovieLens dataset, in particular the 10M version (for easier computation).

Different machine learning algorithms will be trained in one subset of the data to predict movie ratings in the validation set (which will NOT be used in the training process and is strictly used for evaluation only).

The edx set contains data that will be used for training and testing different machine learning algorithms.

The validation dataset is used for evaluation only, which will be used in this report to evaluate the accuracy of different algorithms.

Analysis

Data wrangling

```
##    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 first begin by exploring the dataset through data visualization and data wrangling, from the above summary of the dataset, we can see that the timestamp (the year when the movie is rated) is not in an intuitive format thus we begin by converting the timestamp to year format.

```
##    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_rated
## 1:                Comedy|Romance       1996
## 2:          Action|Crime|Thriller       1996
## 3:  Action|Drama|Sci-Fi|Thriller       1996
## 4:       Action|Adventure|Sci-Fi       1996
## 5: Action|Adventure|Drama|Sci-Fi       1996
## 6:       Children|Comedy|Fantasy       1996
```

Furthermore, the year in which the movie is released is included in the title column, which we can extract using regex and separate function.

Now that we have access the the release year of the movie and the year in which the movie was rated, we can add in a column "age_when_rated" which indicates the age of the movie when the rating was performed to see if it affects the average rating.

```
##    userId movieId rating timestamp                  title year_released
## 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_rated age_when_rated
## 1:                Comedy|Romance       1996              4
## 2:          Action|Crime|Thriller       1996              1
## 3:  Action|Drama|Sci-Fi|Thriller       1996              1
## 4:       Action|Adventure|Sci-Fi       1996              2
## 5: Action|Adventure|Drama|Sci-Fi       1996              2
## 6:       Children|Comedy|Fantasy       1996              2
```

Data visualization

After the data wrangling process, we can now learn more about the dataset through data visualization.

We first explore the number of distinct movies and users in the dataset.

```
##   n_users n_movies
## 1   69855    10588
```

We can also look at the mean and standard deviation of the ratings

```
## [1] 3.516445
```

```
## [1] 1.058554
```

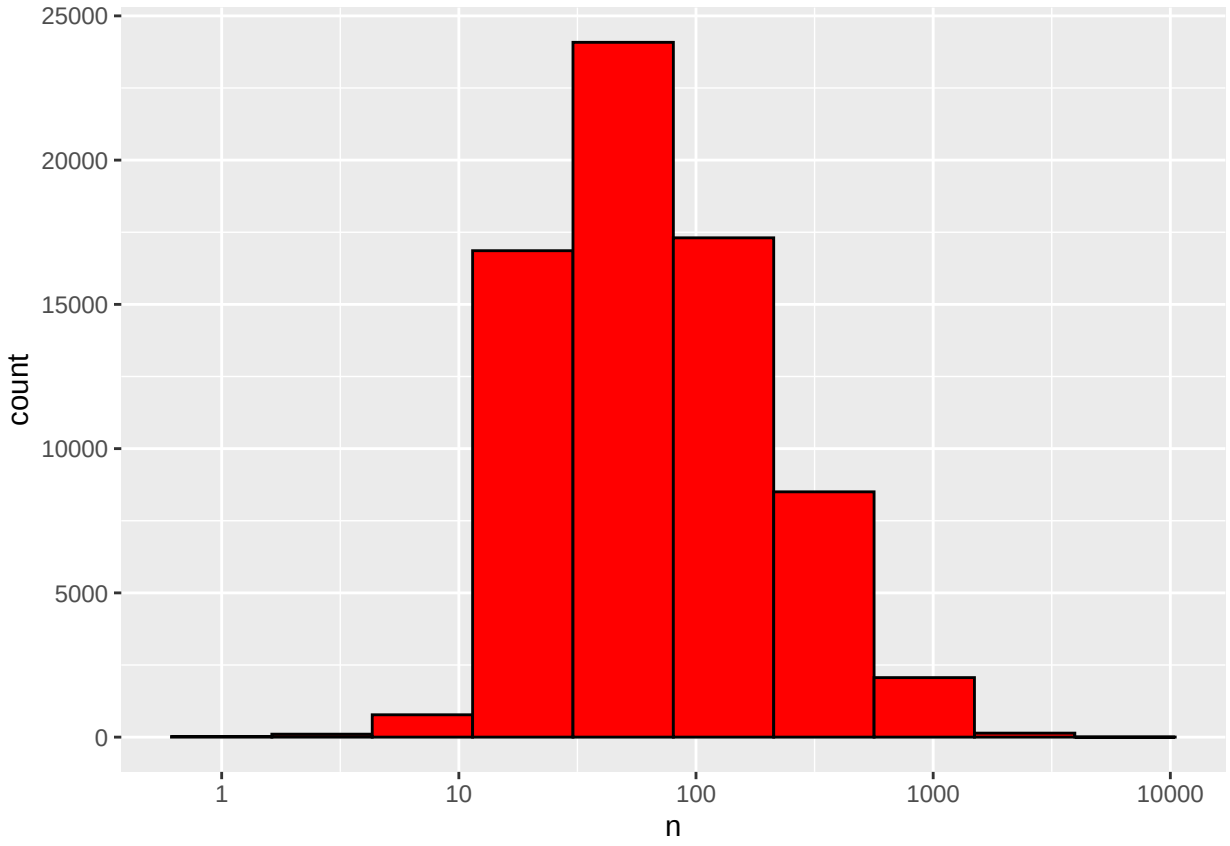
We can make a table of the 10 most rated movies

```
## `summarise()` ungrouping output (override with `.groups` argument)
```

```
## # A tibble: 10 x 2
##    title                                                count
##    <chr>                                                <int>
##  1 Pulp Fiction                                         31362
##  2 Forrest Gump                                         31079
##  3 Silence of the Lambs, The                            30382
##  4 Jurassic Park                                        29360
##  5 Shawshank Redemption, The                            28015
##  6 Braveheart                                           26212
##  7 Fugitive, The                                        26020
##  8 Terminator 2: Judgment Day                           25984
##  9 Star Wars: Episode IV - A New Hope (a.k.a. Star Wars) 25672
## 10 Batman                                               24585
```

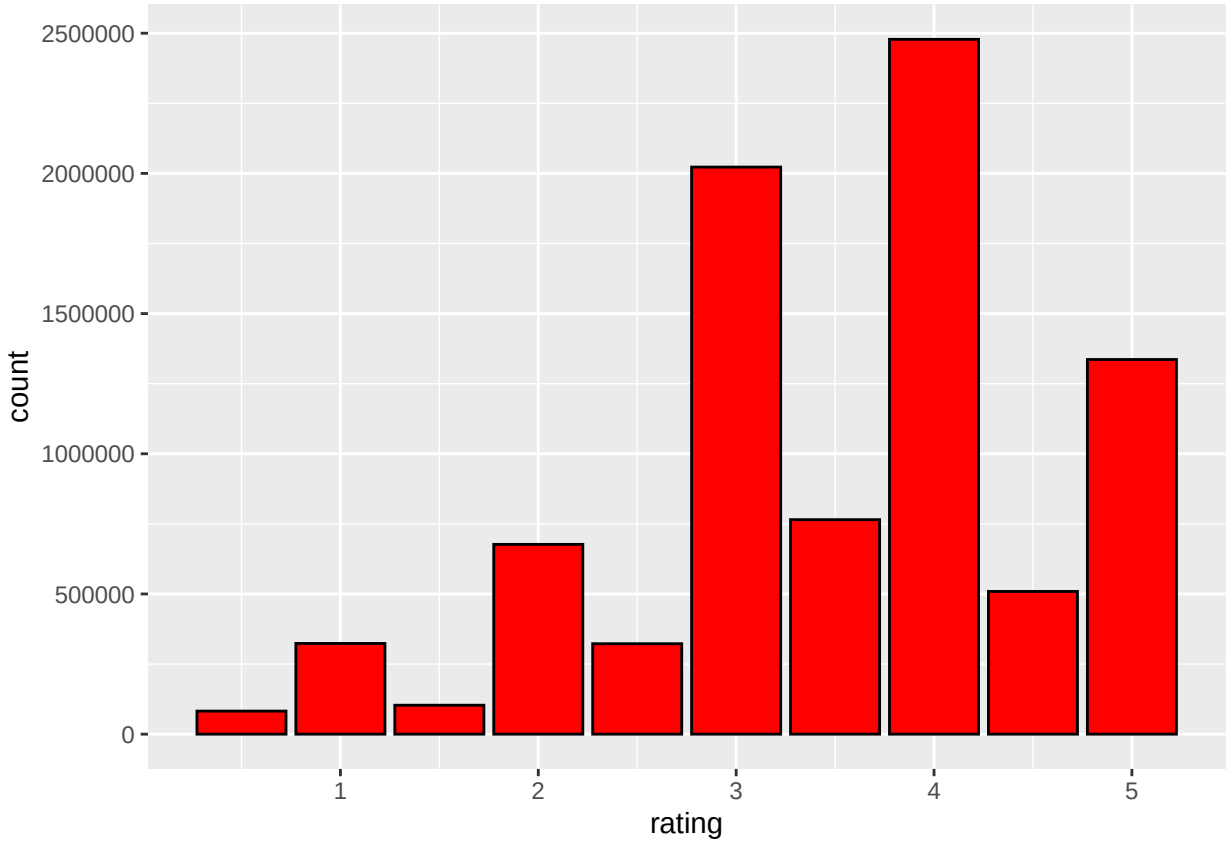
We can then look at how many ratings do users give to movies

```
## `summarise()` ungrouping output (override with `.groups` argument)
```



From the plot, we can see that it kind of follows a normal distribution, with most users giving just under 100 ratings.

We can then look at the rating distribution.



4 star is the most common rating given by user, another useful trend to observe is full star (1,2,3,4,5) ratings are more common than half star ratings (0.5, 1.5, 2.5, 3.5, 4.5)

Next, we look at genre distribution.

```
## `summarise()` ungrouping output (override with `.groups` argument)
```

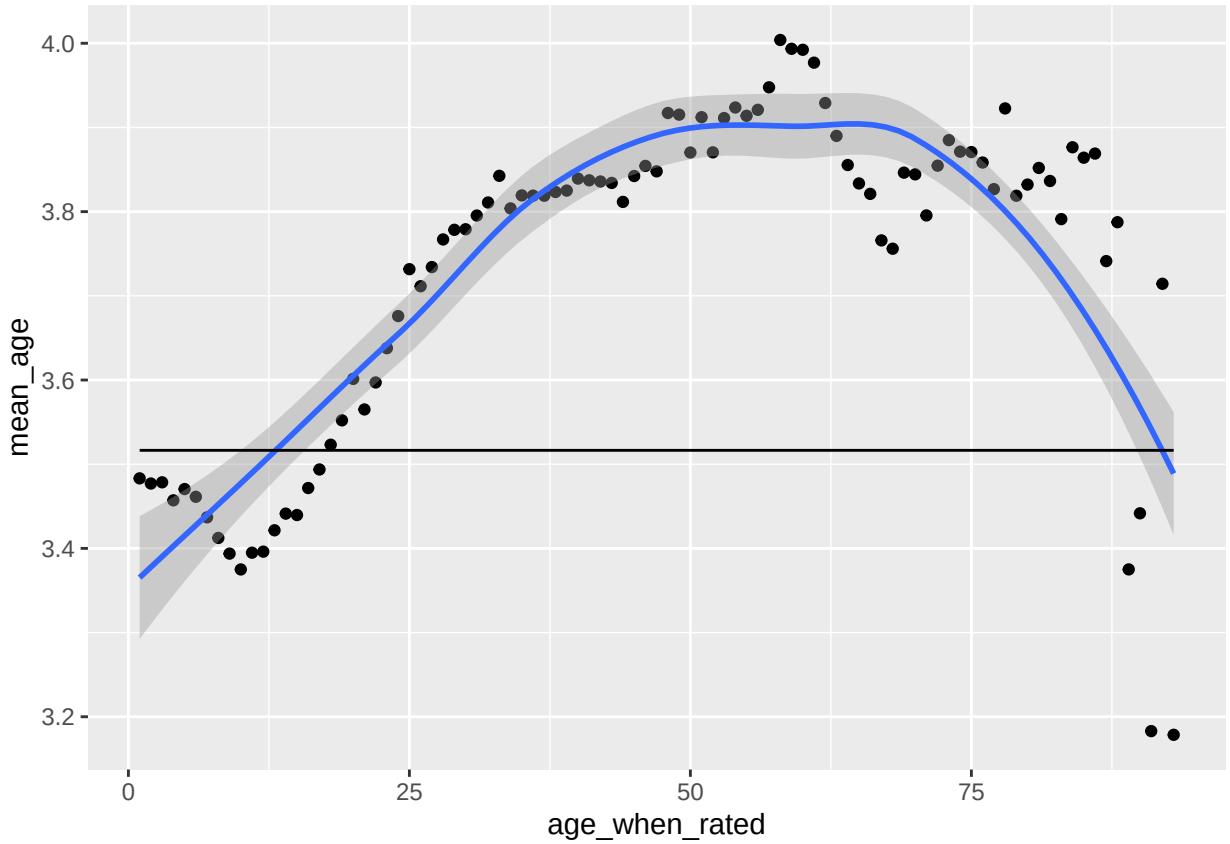
```
## # A tibble: 20 x 2
##    genres                  n
##    <chr>               <int>
##  1 Drama             3772120
##  2 Comedy            3385808
##  3 Action            2423024
##  4 Thriller          2193086
##  5 Adventure         1808971
##  6 Romance           1652625
##  7 Sci-Fi            1281377
##  8 Crime             1275413
##  9 Fantasy            889119
## 10 Children           708993
## 11 Horror             666093
## 12 Mystery            544948
## 13 War                496075
## 14 Animation          444420
## 15 Musical            423064
## 16 Western            186647
```

```
## 17 Film-Noir            116011
## 18 Documentary           87551
## 19 IMAX                   5017
## 20 (no genres listed)        7
```

We can see from the table that drama is the most common genre, whilst 7 movies does not have a genre associated with it.

Finally, we look at the age of movie when it's rated against the mean rating to investigate whether there's a correlation, the mean rating overall is also plotted as a horizontal line.

```
## `summarise()` ungrouping output (override with `.groups` argument)
```



An interesting trend that we can observe from the trend is typically 10-15 years after a movie is released, its rating goes up above the mean rating, this is likely due to popular movies being rated more and spread more, causing them to be rated more and receive higher ratings due to their reputation over the years. This finding inspires the use of an age model later on to predict the rating of the movies.

Model evaluation

We will be using the RMSE (root mean squared error) as our evaluation benchmark, the target RMSE that we are aiming for is < 0.86490, we will see how different models performs at the end.

```r
#define RMSE as our evaluation benchmark

RMSE <- function(true_rating, predicted_rating){
  sqrt(mean((true_rating - predicted_rating)^2))
}
```

The data is partitioned so 80% of the edx dataset is used for training and 20% of the edx dataset is used for testing.

The RMSE results presented in this report are results evaluated on the validation dataset which is NOT used for training in any way and it's only used for the evaluation presented here.

Model 1 | Mean Model This is a naive approach that simply uses the mena rating to predict ratings for all movies.

```r
model_mean <- mean(training_set$rating)

rmse_mean <- RMSE(validation$rating, model_mean)
```

```
## # A tibble: 1 x 2
##   method         RMSE
##   <chr>         <dbl>
## 1 Mean model 1.059507
```

Model 2 | Age Model This model uses the age of the movie when rated to predict rating (age bias)

```r
model_age <- training_set %>%
  group_by(age_when_rated) %>%
  summarize(b_a = mean(rating - model_mean))

predicted_age <- model_mean + validation %>%
  left_join(model_age, by = 'age_when_rated') %>%
  pull(b_a)

rmse_age <- RMSE(validation$rating, predicted_age)
```

```
## # A tibble: 2 x 2
##   method               RMSE
##   <chr>               <dbl>
## 1 Mean model       1.059507
## 2 Rating age model 1.050461
```

Model 3 | Movie Model This model uses the movidId to predict rating (movie bias)

```r
model_movie <- training_set %>%
  group_by(movieId) %>%
  summarize(b_m = mean(rating - model_mean))

predicted_movie <- model_mean + validation %>%
  left_join(model_movie, by = 'movieId') %>%
```

```
  pull(b_m)

rmse_movie <- RMSE(validation$rating, predicted_movie)
```

```
## # A tibble: 3 x 2
##   method                RMSE
##   <chr>                <dbl>
## 1 Mean model       1.059507
## 2 Rating age model 1.050461
## 3 MovieID model    0.9415110
```

Model 4 | User Model This model uses the userId to predict rating (user bias)

```
model_user <- training_set %>%
  group_by(userId) %>%
  summarize(b_u = mean(rating - model_mean))

predicted_user <- model_mean + validation %>%
  left_join(model_user, by = 'userId') %>%
  pull(b_u)

rmse_user <- RMSE(validation$rating, predicted_user)
```

```
## # A tibble: 4 x 2
##   method                RMSE
##   <chr>                <dbl>
## 1 Mean model       1.059507
## 2 Rating age model 1.050461
## 3 UserID model     0.9763781
## 4 MovieID model    0.9415110
```

Model 5 | Genre Model We then attempt to use the genre to predict rating (genre bias)

```
model_genre <- training_set %>%
  group_by(genres) %>%
  summarize(b_g = mean(rating - model_mean))

predicted_genre <- model_mean + validation %>%
  left_join(model_genre, by = 'genres') %>%
  pull(b_g)

rmse_genre <- RMSE(validation$rating, predicted_genre)
```

```
## # A tibble: 5 x 2
##   method                RMSE
##   <chr>                <dbl>
## 1 Mean model       1.059507
## 2 Rating age model 1.050461
## 3 Genre model      1.015847
## 4 UserID model     0.9763781
## 5 MovieID model    0.9415110
```

Model 6 | Release Date Model We also attempt to use the movie's release date to predict rating (release date bias)

```
model_release <- training_set %>%
  group_by(year_released) %>%
  summarize(b_r = mean(rating - model_mean))

predicted_release <- model_mean + validation %>%
  left_join(model_release, by = 'year_released') %>%
  pull(b_r)

rmse_release <- RMSE(validation$rating, predicted_release)
```

```
## # A tibble: 6 x 2
##   method                  RMSE
##   <chr>                  <dbl>
## 1 Mean model          1.059507
## 2 Rating age model    1.050461
## 3 Release date model 1.047883
## 4 Genre model         1.015847
## 5 UserID model        0.9763781
## 6 MovieID model       0.9415110
```

As shown from the table, when only considering one bias for prediction, Movie model has the lowest RMSE, followed by User Model.

The worst performing model is the mean model which is the simplest and naive so it is expected that the performance isnt't as good as the other models.

We now move on to combine different bias to form more complex models.

Model 7 | Movie + User Model We first consider using both MovieId and UserId to predict rating.

```
user_effect <- training_set %>%
  left_join(model_movie, by='movieId') %>%
  group_by(userId) %>%
  summarize(b_u = mean(rating - model_mean - b_m))

predicted_m_u <- validation %>%
  left_join(model_movie, by = 'movieId') %>%
  left_join(user_effect, by = 'userId') %>%
  mutate(pred = model_mean + b_m + b_u)

rmse_m_u <- RMSE(validation$rating, predicted_m_u$pred)
```

```
## # A tibble: 7 x 2
##   method                     RMSE
##   <chr>                     <dbl>
## 1 Mean model             1.059507
## 2 Rating age model       1.050461
## 3 Release date model     1.047883
## 4 Genre model            1.015847
## 5 UserID model           0.9763781
## 6 MovieID model          0.9415110
## 7 MovieID + UserID model 0.8640119
```

Model 8 | Rating Age + Release Date Model We then consider combining rating age and release date to predict rating.

```
release_effect <- training_set %>%
  left_join(model_age, by='age_when_rated') %>%
  group_by(year_released) %>%
  summarize(b_r = mean(rating - model_mean - b_a))

predicted_a_r <- validation %>%
  left_join(model_age, by = 'age_when_rated') %>%
  left_join(release_effect, by = 'year_released') %>%
  mutate(pred = model_mean + b_a + b_r)

rmse_a_r <- RMSE(validation$rating, predicted_a_r$pred)
```

```
## # A tibble: 8 x 2
##   method                              RMSE
##   <chr>                              <dbl>
## 1 Mean model                       1.059507
## 2 Rating age model                 1.050461
## 3 Rating Age + Release date model  1.048308
## 4 Release date model               1.047883
## 5 Genre model                      1.015847
## 6 UserID model                     0.9763781
## 7 MovieID model                    0.9415110
## 8 MovieID + UserID model           0.8640119
```

From the table, we can see that the movie + user model greatly improved the RMSE, whereas the age + release date model didn't help predict ratings.

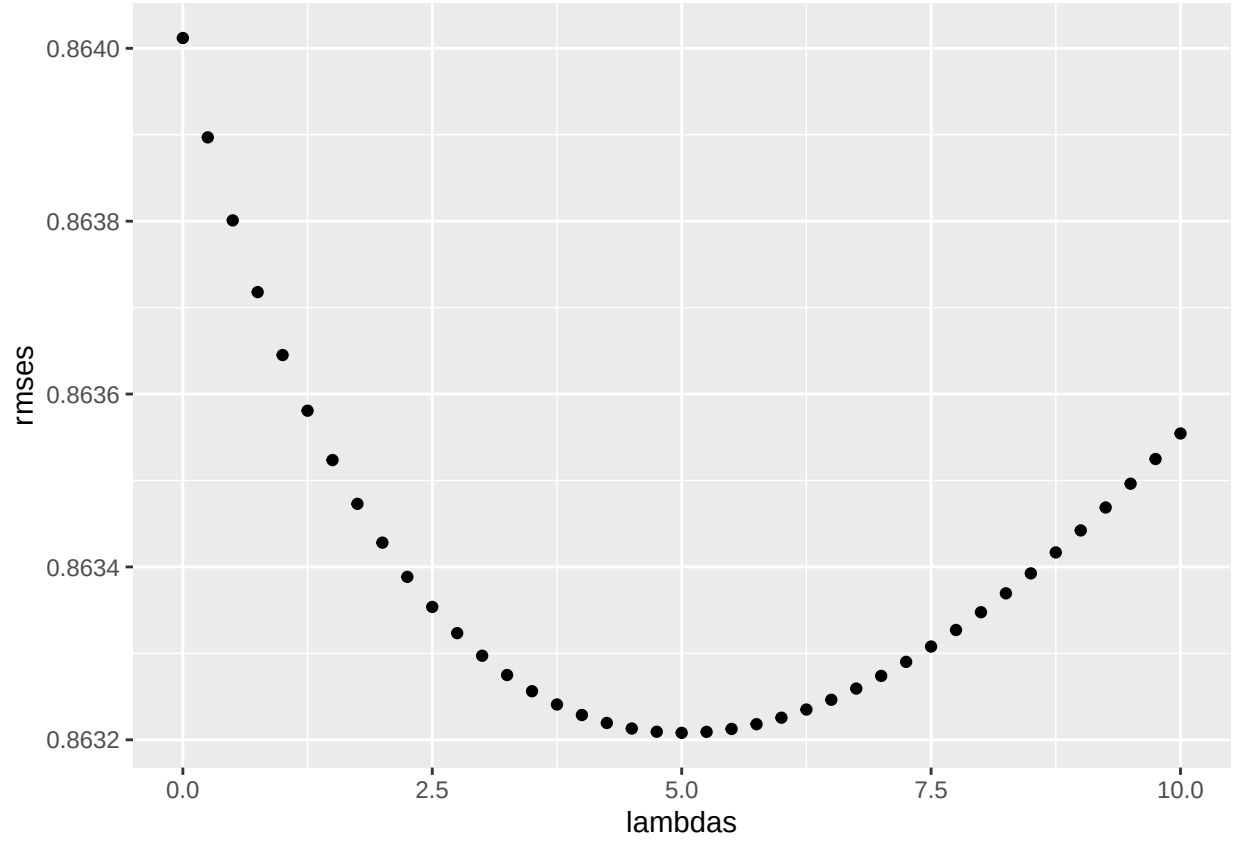
We then consider using regularization to help improve our algorithm.

Regularization is the concept of constraining the total variability of the effect sizes by penalizing large estimates that come from small sample sizes.

Model 9 | Regularized approach using Movie + User We will consider a reularized model using movieId and userid.

```
lambdas <- seq(0, 10, 0.25)
rmses <- sapply(lambdas, function(l){

  mean <- mean(training_set$rating)

  b_m <- training_set %>%
    group_by(movieId) %>%
    summarize(b_m = sum(rating - mean)/(n()+l))

  b_u <- training_set %>%
    left_join(b_m, by='movieId') %>%
    group_by(userId) %>%
    summarize(b_u = sum(rating - mean - b_m)/(n()+l))

  rpred_m_u <- validation %>%
```

```
    left_join(b_m, by='movieId') %>%
    left_join(b_u, by='userId') %>%
    mutate(pred = mean + b_m + b_u) %>%
    .$pred

  return(RMSE(validation$rating, rpred_m_u))
})

qplot(lambdas, rmses)
```




```
rmse_rm_u <- min(rmses)
```

```
## # A tibble: 9 x 2
##   method                                    RMSE
##   <chr>                                    <dbl>
## 1 Mean model                            1.059507
## 2 Rating age model                      1.050461
## 3 Rating Age + Release date model       1.048308
## 4 Release date model                    1.047883
## 5 Genre model                           1.015847
## 6 UserID model                          0.9763781
## 7 MovieID model                         0.9415110
## 8 MovieID + UserID model                0.8640119
## 9 Regularized Movie + User Effect Model 0.8632081
```

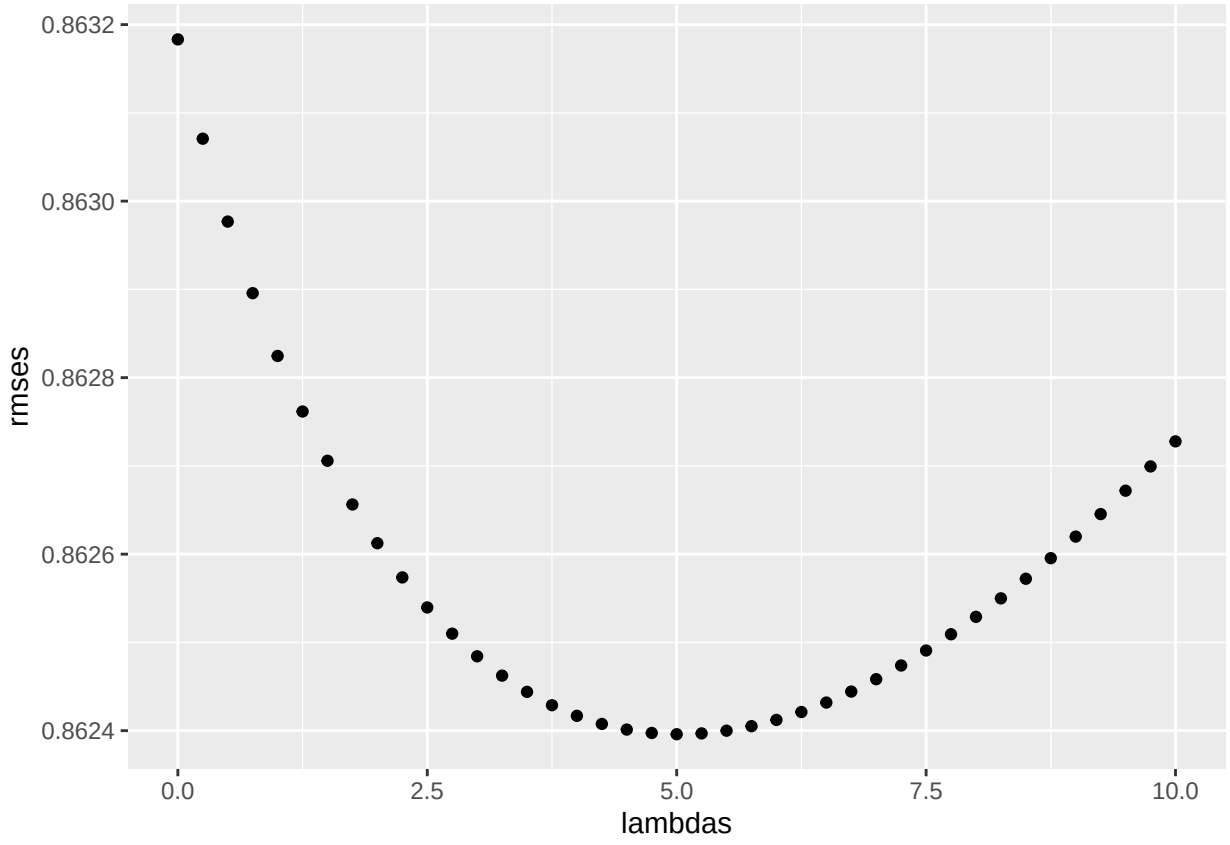
The regularized movie + user model again improved our RMSE.

Model 10 | Final model: Regularized combined bias model We will now construct our final regularized model using all the bias which consists of:

MovieID, UserID, Genre, Release date, Rating age

```
lambdas <- seq(0, 10, 0.25)
rmses <- sapply(lambdas, function(l){

  mean <- mean(training_set$rating)

  b_m <- training_set %>%
    group_by(movieId) %>%
    summarize(b_m = sum(rating - mean)/(n()+l))

  b_u <- training_set %>%
    left_join(b_m, by='movieId') %>%
    group_by(userId) %>%
    summarize(b_u = sum(rating - mean - b_m)/(n()+l))

  b_a <- training_set %>%
    left_join(b_m, by='movieId') %>%
    left_join(b_u, by='userId') %>%
    group_by(age_when_rated) %>%
    summarize(b_a = sum(rating - mean - b_m - b_u)/(n()+l))

  b_r <- training_set %>%
    left_join(b_m, by='movieId') %>%
    left_join(b_u, by='userId') %>%
    left_join(b_a, by='age_when_rated') %>%
    group_by(year_released) %>%
    summarize(b_r = sum(rating - mean - b_m - b_u - b_a)/(n()+l))

  b_g <- training_set %>%
    left_join(b_m, by='movieId') %>%
    left_join(b_u, by='userId') %>%
    left_join(b_a, by='age_when_rated') %>%
    left_join(b_r, by='year_released') %>%
    group_by(genres) %>%
    summarize(b_g = sum(rating - mean - b_m - b_u - b_a - b_r)/(n()+l))

  rpred_m_u <- validation %>%
    left_join(b_m, by='movieId') %>%
    left_join(b_u, by='userId') %>%
    left_join(b_a, by='age_when_rated') %>%
    left_join(b_r, by='year_released') %>%
    left_join(b_g, by='genres') %>%
    mutate(pred = mean + b_m + b_u + b_a + b_r + b_g) %>%
    .$pred

  return(RMSE(validation$rating, rpred_m_u))
})
```

```
qplot(lambdas, rmses)
```




```
rmse_rcombined <- min(rmses)
```

```
## # A tibble: 10 x 2
##    method                                    RMSE
##    <chr>                                    <dbl>
##  1 Mean model                            1.059507
##  2 Rating age model                      1.050461
##  3 Rating Age + Release date model       1.048308
##  4 Release date model                    1.047883
##  5 Genre model                           1.015847
##  6 UserID model                          0.9763781
##  7 MovieID model                         0.9415110
##  8 MovieID + UserID model                0.8640119
##  9 Regularized Movie + User Effect Model 0.8632081
## 10 Regularized combined bias Model       0.8623960
```

Results

Here again are the final RMSE results using the different algorithms. Our final model is a regularized model using a combination of different bias to predict rating, and it gives use the lowest RMSE and achieved our aim of < 0.86490.

```
## # A tibble: 10 x 2
##    method                                    RMSE
##    <chr>                                    <dbl>
##  1 Mean model                             1.059507
##  2 Rating age model                       1.050461
##  3 Rating Age + Release date model        1.048308
##  4 Release date model                     1.047883
##  5 Genre model                            1.015847
##  6 UserID model                           0.9763781
##  7 MovieID model                          0.9415110
##  8 MovieID + UserID model                 0.8640119
##  9 Regularized Movie + User Effect Model 0.8632081
## 10 Regularized combined bias Model        0.8623960
```

Conclusion

These algorithms are the fundamental of recommendation systems which is used widely to make specific recommendation to users, the goal is to reduce RMSE as much as possible as using a 5-star rating system, a RMSE > 1 is not a good prediction by any means.

Using the skills we learnt from the course, we are able to build a model that can achieve a RMSE of 0.8623960.