# MovieLens Project

John B Read

03/01/2022

## Introduction

This project is undertaken to accomplish a requirement of the capstone project for the HarvardX Data Science course. This project builds upon previous modules within the course that have explored movie ratings and the algorithms used by movie service providers to generate recommendations for customers (see Chapter 33.7 for a discussion of how this has been done by Netflix). Using a dataset comprising movie rating (MovieLens), this project demonstrates how Machine Learning is a useful tool for predicting movie ratings by users when dataset size is unwieldy. In developing a predictive algorithm, the project will subsequently test and validate the algorithm against the true values within the dataset using Root-Mean-Square Error (RMSE).

## Methods

This project incorporates the MovieLens 10M dataset. Which according to GroupLens (2022, para. 1), is a "Stable benchmark dataset. 10 million ratings and 100,000 tag applications applied to 10,000 movies by 72,000 users. Released 1/2009." As discussed in the introduction, this dataset will be used to test and validate a Machine Learning algorithm for predicting movie ratings. To do this, the dataset will be split. First, into a training section (90%) that has 9,000,055 rows and 6 columns. Then, into a test section (10%) that has 999,999 rows and 6 columns.

```
glimpse(capstone)
```

```
## Rows: 9,000,055
## Columns: 6
## $ userId    <int> 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 2, ~
## $ movieId   <dbl> 122, 185, 292, 316, 329, 355, 356, 362, 364, 370, 377, 420, ~
## $ rating    <dbl> 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, ~
## $ timestamp <int> 838985046, 838983525, 838983421, 838983392, 838983392, 83898~
## $ title     <chr> "Boomerang (1992)", "Net, The (1995)", "Outbreak (1995)", "S~
## $ genres    <chr> "Comedy|Romance", "Action|Crime|Thriller", "Action|Drama|Sci~
```

```
glimpse(validation)
```

```
## Rows: 999,999
## Columns: 6
## $ userId    <int> 1, 1, 1, 2, 2, 2, 3, 3, 4, 4, 4, 5, 5, 5, 5, 5, 5, 5, 5, 5, ~
## $ movieId   <dbl> 231, 480, 586, 151, 858, 1544, 590, 4995, 34, 432, 434, 85, ~
## $ rating    <dbl> 5.0, 5.0, 5.0, 3.0, 2.0, 3.0, 3.5, 4.5, 5.0, 3.0, 3.0, 3.0, ~
## $ timestamp <int> 838983392, 838983653, 838984068, 868246450, 868245645, 86824~
## $ title     <chr> "Dumb & Dumber (1994)", "Jurassic Park (1993)", "Home Alone ~
## $ genres    <chr> "Comedy", "Action|Adventure|Sci-Fi|Thriller", "Children|Come~
```

With these two data sections, Root-Mean-Square Error (RMSE) will be used to test and validate the accuracy of the algorithm. There are multiple ways that an Algorithm can be modeled. This project will first incorporate a simple algorithm that assumes that movie ratings will remain the same across the reviewers. This can be understood as the following equation:

$$Y_{u,i} = \mu + \varepsilon_{u,i} \tag{1}$$

"where $\varepsilon_{u,i}$ independent errors sampled from the same distribution centered at 0 and $\mu$ the "true" rating for all movies. We know that the estimate that minimizes the RMSE is the least squares estimate of $\mu$" (Irizarry, 2019, chp. 33.7.4).

```
mu_hat <- mean(capstone$rating)
naive_rmse <- RMSE(validation$rating, mu_hat)
naive_rmse
```

```
## [1] 1.061202
```

This algorithm appears to overfit with an RMSE of over 1. So a more advanced algorithm will be applied to attempt to gain a lower RMSE. This second algorithm incorporates a movies effect (or bias) as denoted by b_i. This new algorithm equation appears as:

$$Y_{u,i} = \mu + b_i + \varepsilon_{u,i} \tag{2}$$

This equation could be run as a linear model. However, Irizarry (2019) cautions that due to large amount of effect/bias within the dataset this method will take an exorbitant amount of time to run. As such, we create b_i (fit) for each movie rank and then predict ratings for $\mu$ and b_i.

```
fit <- capstone %>%
   group_by(movieId) %>%
   summarize(fit = mean(rating - mu_hat))

ratingpredict <- validation %>%
   left_join(fit, by='movieId') %>%
   mutate(pred = mu_hat + fit) %>%
   pull(pred)

RMSE(validation$rating, ratingpredict)
```

```
## [1] 0.9439087
```

This algorithm can be further fine-tuned through the inclusion of user effects/bias. This can be accomplished through the inclusion of the user specific effect/bias variable b_u (user_avgs).

$$Y_{u,i} = \mu + b_i + b_u + \varepsilon_{u,i} \tag{3}$$

The variable b_u will be calculated as the average of:

$$y_{u,i} - \hat{\mu} - \hat{b}_i \tag{4}$$

```
user_avgs <- capstone %>%
   left_join(fit, by='movieId') %>%
   group_by(userId) %>%
   summarize(user_avgs = mean(rating - mu_hat - fit))
```

```r
ratingpredict <- validation %>%
   left_join(fit, by='movieId') %>%
   left_join(user_avgs, by='userId') %>%
   mutate(pred = mu_hat + fit + user_avgs) %>%
   pull(pred)

RMSE(validation$rating, ratingpredict)
```

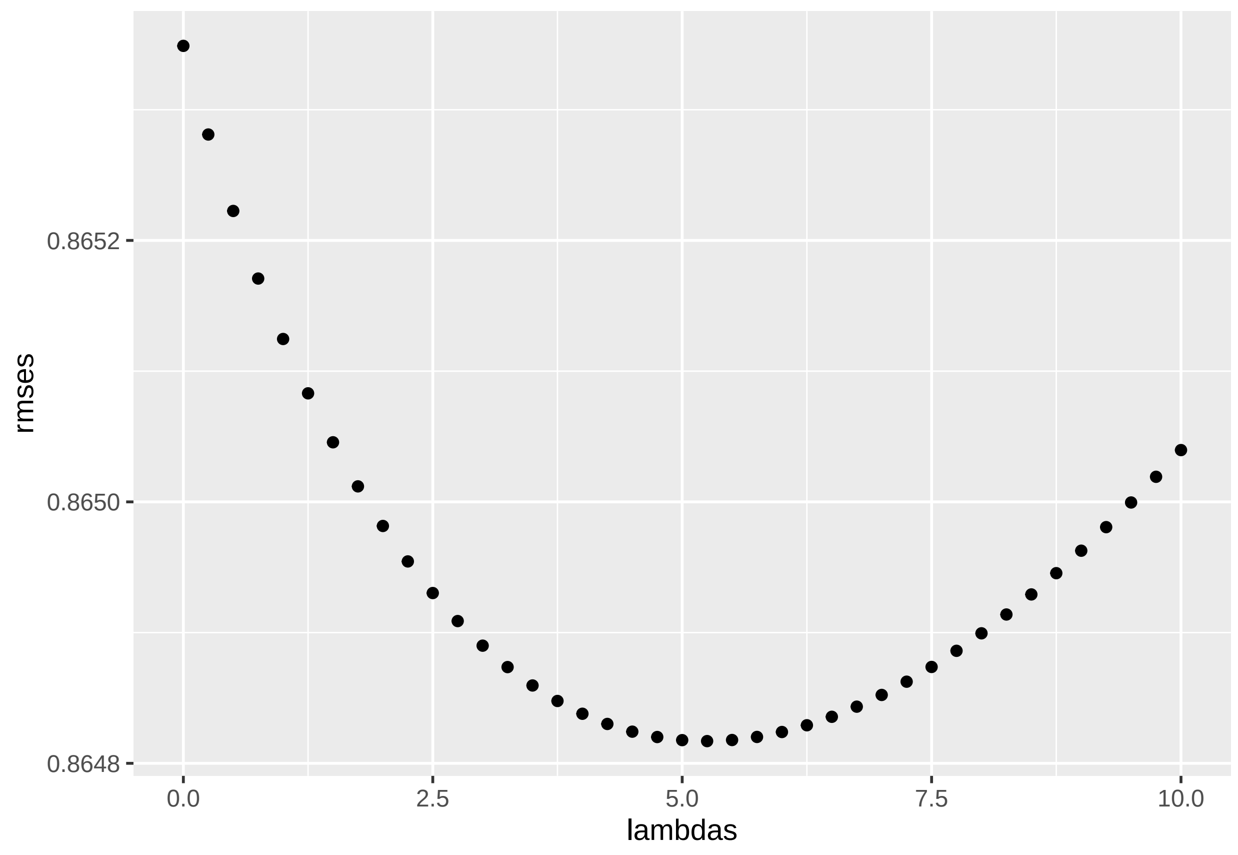
```
## [1] 0.8653488
```

While a significant improvement in RMSE was observed with the more advanced algorithm that included both movie and user effect/bias. The algorithm can be further tailored through regularisation. As Irizarry (2019) notes, "regularization permits us to penalize large estimates that are formed using small sample sizes" (chp. 33.9.2). This allows for for movies with few reviews to be penalised such that the effect on b_i is reduced by ratings with high uncertainty. Similarly, regularization allows for penalisation of effects on b_u. Combining these through regularization provides the following algorithm:

$$\sum_{u,i} (y_{u,i} - \mu - b_i - b_u)^2 + \lambda \left( \sum_i b_i^2 + \sum_u b_u^2 \right) \tag{5}$$

In this equation the lambda ($\lambda$) is identified through cross-validation and RMSE is calculated with each lambda. Then a plot is generated and lowest RMSE identified.

```r
lambdas <- seq(from=0, to=10, by=0.25)
qplot(lambdas, rmses)
```



```r
min(rmses)
```

```
## [1] 0.864817
```

The lambda for the corresponding minimum RMSE is 5.25.

## Results

After deriving the lambda that corresponds to the minimum RMSE, the algorithm with regularisation of movie and user effects/bias was run with lambda 5.25 (see below) which provided an RMSE of 0.864817.

```r
l <- lambdas[which.min(rmses)]

fit1 <- capstone %>%
   group_by(movieId) %>%
   summarize(fit1 = sum(rating - mu_hat)/(n()+l))

user_avgs1 <- capstone %>%
   left_join(fit1, by="movieId") %>%
   group_by(userId) %>%
   summarize(user_avgs1 = sum(rating - fit1 - mu_hat)/(n()+l))

ratingpredict1 <-
   validation %>%
   left_join(fit1, by = "movieId") %>%
   left_join(user_avgs1, by = "userId") %>%
   mutate(pred = mu_hat + fit1 + user_avgs1) %>%
   pull(pred)

RMSE(ratingpredict1, validation$rating)
```

```
## [1] 0.864817
```

| Algorithm | RMSE |
|---|---|
| First Model | 1.0612 |
| Movie Bias | 0.9439 |
| User Bias | 0.8654 |
| Regularised Movie/User Bias | 0.8648 |

## Conclusion

As a requirement of the Capstone HarvardX Data Science course, this report demonstrates how incorporating Machine Learning to predict movie ratings by user can be effective. In evaluating the MovieLens 10M dataset through the construction of and use of multiple predictive algorithms (while accounting for effects/bias of both ratings and users and through the incorporation of regularisation), this report demonstrates that RMSE can be reduced and that the algorithm can be effective in predicting a users movie ratings. However, this report is not without limitations especially as it required the use of regularisation. The dataset used has inherent issues especially pertaining to how some movies have an extreme under and over-representation of number of reviews. The dataset would be benefited by more reviews of under-represented movies to provide the algorithm a more balanced dataset. Future research should continue to evaluate how algorithms can be further fine-tuned as well as may be benefited by generating algorithms that are limited to each genre of movie as this may demonstrate greater predictive power.

## References

GroupLens. (2022). MovieLens 10M Dataset. GroupLens. https://grouplens.org/datasets/movielens/10m/
Irizarry, R. A. (2019). Introduction to Data Science. CRC Press. https://rafalab.github.io/dsbook/index.html