edX HarvardX: PH125.8x | MovieLens Project

Nicole Yong

8/26/2019

OVERVIEW

INTRODUCTION

This MovieLens project is created to fulfil the coursework requirements of the capstone module in edX HarvardX: PH125.9x (Data Science Professional Certificate).

MovieLens Dataset

The MovieLens dataset, collected by GroupLens Research, includes movie rating datasets from the MovieLens website (http://movielens.org).

For the purpose of this project, the MovieLens 10M Dataset is used. This dataset includes 10 million ratings applied to 10,000 movies by 72,000 users. It was released in 2009.

This dataset can be retrieved in the following links:

- https://grouplens.org/datasets/movielens/10m/
- http://files.grouplens.org/datasets/movielens/ml-10m.zip

Goal of MovieLens Project

The goal of this project is to create a machine learning algorithm that predicts movie ratings in the validation dataset as if they were unknown.

RMSE (Root Mean Squared Error) value is used to evaluate the effectiveness of the models. RMSE is the standard deviation of the residuals (prediction errors). Residuals are a measure of how far from the regression line data points are. Hence, a lower RMSE value indicates a better fit.

Key Steps Taken

1. Data Gathering & Loading the Dataset

- Load the dataset from data source
- Split the dataset into train and test dataset

2. Data Preprocessing

- Explore both the train and test dataset
- Check for variables that require preprocessing (e.g. missing values, wrong format)

3. Data Exploration

- Gain insights on the dataset

- Explore the relationships between different variables

4. Model Selection and Training

- Test out different models
- Improve models' performance using insights from data exploration

METHODS & ANALYSIS OF MOVIELENS DATASET

Load required libraries

```
## -- Attaching packages --------------------------------------------------------- tidyverse 1.2.1 --

## v ggplot2 3.2.0     v purrr   0.3.2
## v tibble  2.1.3     v dplyr   0.8.3
## v tidyr   0.8.3     v stringr 1.4.0
## v readr   1.3.1     v forcats 0.4.0

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

## Loading required package: lattice

##
## Attaching package: 'caret'

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

##
## Attaching package: 'data.table'

## The following objects are masked from 'package:dplyr':
##
##     between, first, last

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

##
## Attaching package: 'lubridate'

## The following objects are masked from 'package:data.table':
##
##     hour, isoweek, mday, minute, month, quarter, second, wday,
##     week, yday, year
```

```
## The following object is masked from 'package:base':
##
##     date
```

```
## NOTE: Either Arial Narrow or Roboto Condensed fonts are required to use these themes.
```

```
##       Please use hrbrthemes::import_roboto_condensed() to install Roboto Condensed and
```

```
##       if Arial Narrow is not on your system, please see http://bit.ly/arialnarrow
```

```
## Loading required package: RColorBrewer
```

Print session information

To help with code reproducibility, print version information about R, the OS and attached or loaded packages.

```
## R version 3.6.1 (2019-07-05)
## Platform: x86_64-w64-mingw32/x64 (64-bit)
## Running under: Windows 10 x64 (build 17134)
##
## Matrix products: default
##
## locale:
## [1] LC_COLLATE=English_Singapore.1252  LC_CTYPE=English_Singapore.1252
## [3] LC_MONETARY=English_Singapore.1252 LC_NUMERIC=C
## [5] LC_TIME=English_Singapore.1252
##
## attached base packages:
## [1] stats     graphics  grDevices utils     datasets  methods   base
##
## other attached packages:
##  [1] wordcloud_2.6      RColorBrewer_1.1-2 hrbrthemes_0.6.0
##  [4] lubridate_1.7.4    data.table_1.12.2  caret_6.0-84
##  [7] lattice_0.20-38    forcats_0.4.0      stringr_1.4.0
## [10] dplyr_0.8.3        purrr_0.3.2        readr_1.3.1
## [13] tidyr_0.8.3        tibble_2.1.3       ggplot2_3.2.0
## [16] tidyverse_1.2.1
##
## loaded via a namespace (and not attached):
##  [1] Rcpp_1.0.1         class_7.3-15       assertthat_0.2.1
##  [4] zeallot_0.1.0      digest_0.6.20      ipred_0.9-9
##  [7] foreach_1.4.4      R6_2.4.0           cellranger_1.1.0
## [10] plyr_1.8.4         backports_1.1.4    stats4_3.6.1
## [13] evaluate_0.14      httr_1.4.0         pillar_1.4.2
## [16] gdtools_0.1.9      rlang_0.4.0        lazyeval_0.2.2
## [19] readxl_1.3.1       rstudioapi_0.10    extrafontdb_1.0
## [22] rpart_4.1-15       Matrix_1.2-17      rmarkdown_1.14
## [25] splines_3.6.1      extrafont_0.17     gower_0.2.1
## [28] munsell_0.5.0      broom_0.5.2        compiler_3.6.1
## [31] modelr_0.1.4       xfun_0.9           pkgconfig_2.0.2
## [34] htmltools_0.3.6    nnet_7.3-12        tidyselect_0.2.5
## [37] prodlim_2018.04.18 codetools_0.2-16   crayon_1.3.4
```

```
## [40] withr_2.1.2        ModelMetrics_1.2.2 MASS_7.3-51.4
## [43] recipes_0.1.6      grid_3.6.1         Rttf2pt1_1.3.7
## [46] nlme_3.1-140       jsonlite_1.6       gtable_0.3.0
## [49] magrittr_1.5       scales_1.0.0       cli_1.1.0
## [52] stringi_1.4.3      reshape2_1.4.3     timeDate_3043.102
## [55] xml2_1.2.0         vctrs_0.2.0        generics_0.0.2
## [58] lava_1.6.5         iterators_1.0.10   tools_3.6.1
## [61] glue_1.3.1         hms_0.5.0          survival_2.44-1.1
## [64] yaml_2.2.0         colorspace_1.4-1   rvest_0.3.4
## [67] knitr_1.23         haven_2.1.1
```

STEP 1: DATA GATHERING & LOADING THE DATASET

Load MovieLens dataset

```r
dl <- tempfile()
download.file("http://files.grouplens.org/datasets/movielens/ml-10m.zip", dl)

ratings <- fread(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(movieId))[movieId],
                                           title = as.character(title),
                                           genres = as.character(genres))

movielens <- left_join(ratings, movies, by = "movieId")
```

Create train (edx) and test (validation) sets

Validation set will be 10% of MovieLens dataset

```r
set.seed(1)
test_index <- createDataPartition(y = movielens$rating, times = 1, p = 0.1, list = FALSE)
edx <- movielens[-test_index,]
temp <- movielens[test_index,]
```

Ensure userId and movieId in validation set are also in edx set

```r
validation <- temp %>%
  semi_join(edx, by = "movieId") %>%
  semi_join(edx, by = "userId")
```

Add rows removed from validation set back into edx set

```r
removed <- anti_join(temp, validation)
edx <- rbind(edx, removed)
rm(dl, ratings, movies, test_index, temp, movielens, removed)
```

STEP 2: DATA PREPROCESSING

Take a look at the train (edx) dataset

```r
glimpse(edx)
```

```
## Observations: 9,000,061
## Variables: 6
## $ userId    <int> 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1...
## $ movieId   <dbl> 122, 185, 231, 292, 316, 329, 355, 356, 362, 364, 37...
## $ rating    <dbl> 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5...
## $ timestamp <int> 838985046, 838983525, 838983392, 838983421, 83898339...
## $ title     <chr> "Boomerang (1992)", "Net, The (1995)", "Dumb & Dumbe...
## $ genres    <chr> "Comedy|Romance", "Action|Crime|Thriller", "Comedy",...
```

```r
summary(edx)
```

```
##      userId         movieId          rating        timestamp
##  Min.   :    1   Min.   :    1   Min.   :0.500   Min.   :7.897e+08
##  1st Qu.:18122   1st Qu.:  648   1st Qu.:3.000   1st Qu.:9.468e+08
##  Median :35743   Median : 1834   Median :4.000   Median :1.035e+09
##  Mean   :35869   Mean   : 4120   Mean   :3.512   Mean   :1.033e+09
##  3rd Qu.:53602   3rd Qu.: 3624   3rd Qu.:4.000   3rd Qu.:1.127e+09
##  Max.   :71567   Max.   :65133   Max.   :5.000   Max.   :1.231e+09
##     title              genres
##  Length:9000061     Length:9000061
##  Class :character   Class :character
##  Mode  :character   Mode  :character
##
##
##
```

```r
head(edx)
```

```
##   userId movieId rating timestamp                         title
## 1      1     122      5 838985046              Boomerang (1992)
## 2      1     185      5 838983525               Net, The (1995)
## 3      1     231      5 838983392          Dumb & Dumber (1994)
## 4      1     292      5 838983421               Outbreak (1995)
## 5      1     316      5 838983392               Stargate (1994)
## 6      1     329      5 838983392 Star Trek: Generations (1994)
##                          genres
## 1                Comedy|Romance
## 2         Action|Crime|Thriller
## 3                        Comedy
## 4  Action|Drama|Sci-Fi|Thriller
## 5       Action|Adventure|Sci-Fi
## 6 Action|Adventure|Drama|Sci-Fi
```

Insights

1. No missing variables from summary table
2. Timestamp variable needs to be processed (converted to year)

```
edx <- mutate(edx, timestamp_year = year(as_datetime(timestamp)))
```

3. Genres variable needs to be processed (separate movies with multiple genres into individual rows per genre)

```
edx <- edx %>% separate_rows(genres, sep ="\\|")
```

Take a look at the test (validation) dataset

```
glimpse(validation)
```

```
## Observations: 999,993
## Variables: 6
## $ userId    <int> 1, 2, 2, 3, 3, 3, 4, 4, 4, 5, 5, 5, 5, 5, 5, 5, 6, 6...
## $ movieId   <dbl> 588, 1210, 1544, 151, 1288, 5299, 380, 435, 480, 477...
## $ rating    <dbl> 5.0, 4.0, 3.0, 4.5, 3.0, 3.0, 3.0, 3.0, 5.0, 3.0, 3....
## $ timestamp <int> 838983339, 868245644, 868245920, 1133571026, 1133571...
## $ title     <chr> "Aladdin (1992)", "Star Wars: Episode VI - Return of...
## $ genres    <chr> "Adventure|Animation|Children|Comedy|Musical", "Acti...
```

```
summary(validation)
```

```
##      userId         movieId          rating        timestamp
##  Min.   :    1   Min.   :    1   Min.   :0.500   Min.   :7.897e+08
##  1st Qu.:18127   1st Qu.:  653   1st Qu.:3.000   1st Qu.:9.468e+08
##  Median :35719   Median : 1835   Median :4.000   Median :1.036e+09
##  Mean   :35878   Mean   : 4121   Mean   :3.512   Mean   :1.033e+09
##  3rd Qu.:53649   3rd Qu.: 3633   3rd Qu.:4.000   3rd Qu.:1.127e+09
##  Max.   :71567   Max.   :65133   Max.   :5.000   Max.   :1.231e+09
##     title              genres
##  Length:999993      Length:999993
##  Class :character   Class :character
##  Mode  :character   Mode  :character
##
##
##
```

```
head(validation)
```

```
##   userId movieId rating  timestamp
## 1      1     588    5.0  838983339
## 2      2    1210    4.0  868245644
## 3      2    1544    3.0  868245920
```

```
## 4      3     151    4.5 1133571026
## 5      3    1288    3.0 1133571035
## 6      3    5299    3.0 1164885617
##                                                 title
## 1                                      Aladdin (1992)
## 2       Star Wars: Episode VI - Return of the Jedi (1983)
## 3 Lost World: Jurassic Park, The (Jurassic Park 2) (1997)
## 4                                      Rob Roy (1995)
## 5                            This Is Spinal Tap (1984)
## 6                       My Big Fat Greek Wedding (2002)
##                                     genres
## 1 Adventure|Animation|Children|Comedy|Musical
## 2                     Action|Adventure|Sci-Fi
## 3     Action|Adventure|Horror|Sci-Fi|Thriller
## 4                    Action|Drama|Romance|War
## 5                              Comedy|Musical
## 6                              Comedy|Romance
```

Insights

1. No missing variables from summary table
2. Timestamp variable needs to be processed (converted to year)

```r
validation <- mutate(validation, timestamp_year = year(as_datetime(timestamp)))
```

3. Genres variable needs to be processed (separate movies with multiple genres into individual rows per genre)

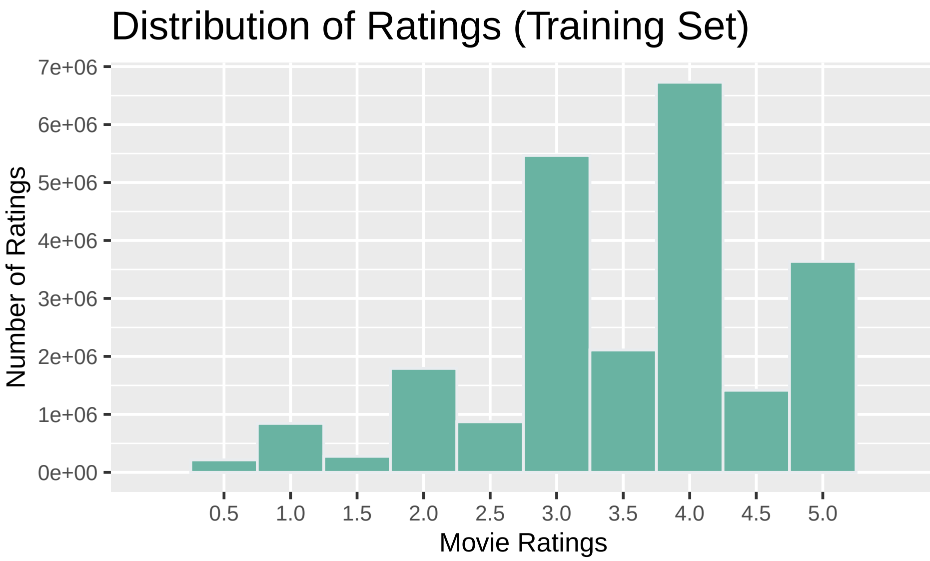
```r
validation <- validation %>% separate_rows(genres, sep ="\\|")
```

STEP 3: DATA EXPLORATION

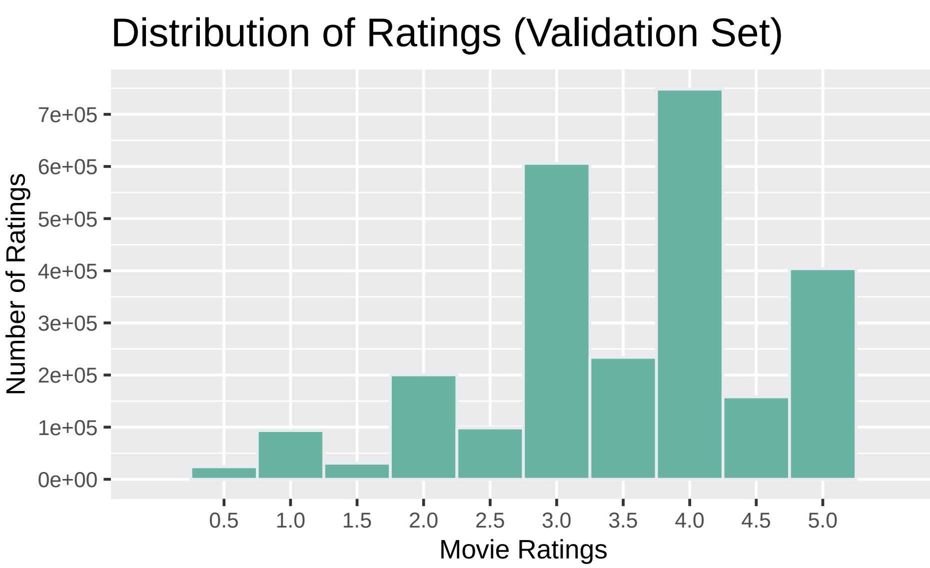
Explore number of unique movies and users in edx training set

- There are 69878 unique users
- There are 10677 unique movies

Explore distribution of ratings in edx training set



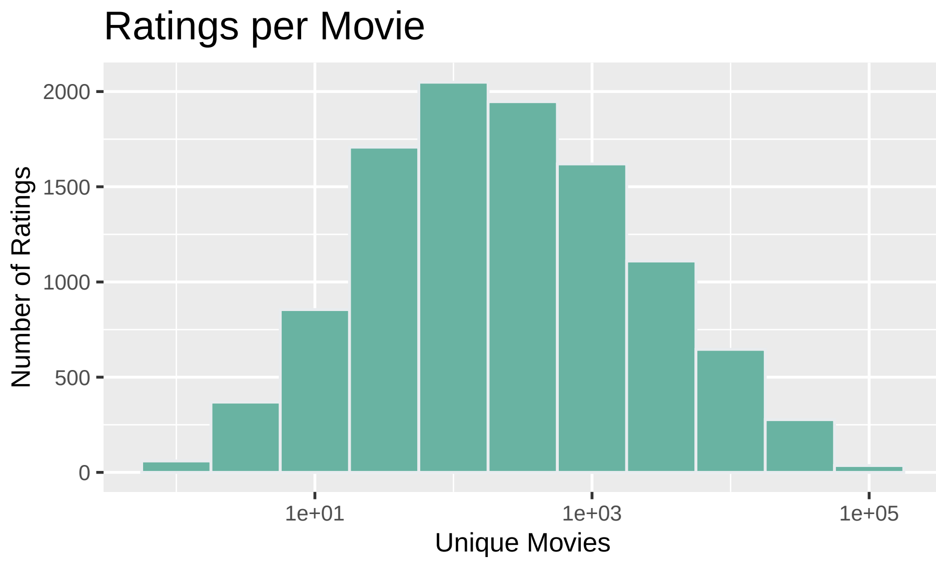
Explore distribution of ratings in validation set



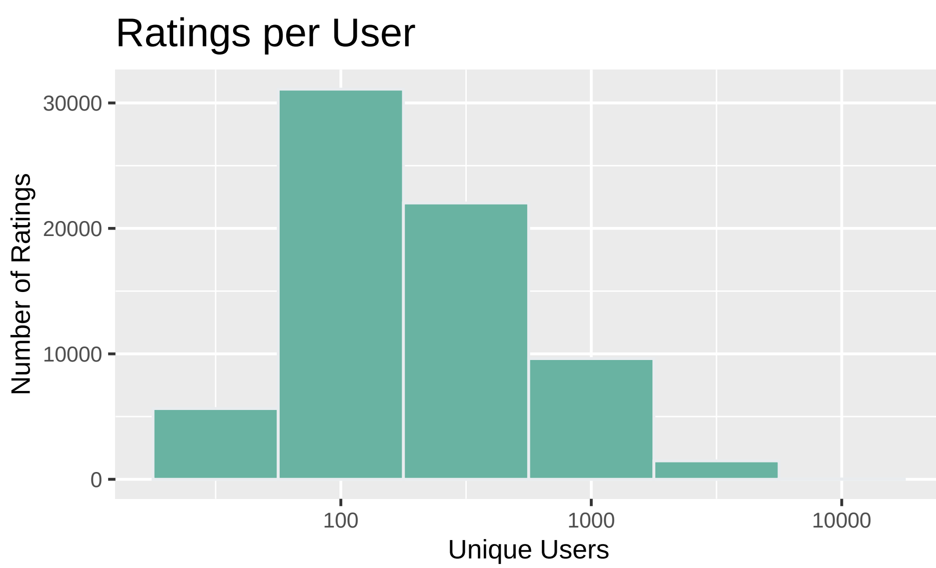
Insights

- Both train and test datasets have similar distribution of ratings
- More full ratings are given compared to half (0.5) ratings

Explore number of ratings per unique movie



Explore number of ratings per unique user



Insights

- Certain movies have a higher number of ratings compared to others
- There are some movies with very few ratings
- Certain users rate more movies than other users

Explore the genres of movies rated

Insights

- Certain genres of movies garner more movie ratings
- However, note that a single movie can have multiple genres

STEP 4: MODEL SELECTION AND TRAINING

Define RMSE (Root Mean Squared Error)

```
RMSE <- function(true_ratings, predicted_ratings){
  sqrt(mean((true_ratings - predicted_ratings)^2))
}
```

Part 4.1 - Baseline RMSE

To compute the baseline RMSE with the average rating computed from the training dataset

```
mu <- mean(edx$rating)
baseline_RMSE <- RMSE(edx$rating, mu)
```

Test the baseline model against the validation (test) dataset
Add baseline RMSE result to a table for later comparison

Model	RMSE
Baseline RMSE	1.052443

Part 4.1 Insights

- Baseline RMSE is **1.0524433**
- The goal of the subsequent models is to perform better than the baseline RMSE
- To improve subsequent models using insights gained from data exploration

Data Exploration Insights

- The top 3 ratings given by users are 4.0, 3.0 and 5.0 respectively
- 0.5 (or half) ratings are less popular than whole ratings
- Potential movie effect: Different movies have different ratings frequency and different ratings
- Potential user effect: Certain users have rated more movies than other users

Part 4.2 - Movie Effect

- Recap: Some movies have a higher (than mean) rating
- Account for estimated deviation of each movie's mean rating from total average mean

Account for Movie Effect

```
movie_mu <- edx %>%
  group_by(movieId) %>%
  summarize(b_i = mean(rating - mu))
```

Test the adjusted model
Add Movie Effect RMSE results to a table for later comparison

Model	RMSE
Baseline RMSE	1.0524433
Movie Effect	0.9411063

Part 4.2 Insights

RMSE has improved to **0.9411063**

Part 4.3 - User Effect

- Recap: Different users rate movies differently
- Account for movie effect and user effect in the predicted ratings

```
user_mu <- edx %>%
  left_join(movie_mu, by='movieId') %>%
  group_by(userId) %>%
  summarize(b_u = mean(rating - mu - b_i))
```

Test the adjusted model
Add User Effect RMSE results to a table for later comparison

Model	RMSE
Baseline RMSE	1.0524433
Movie Effect	0.9411063
Movie Effect & User Effect	0.8635899

Part 4.3 Insights

RMSE has improved to **0.8635899**

Part 4.4 - Lambda (Regularization rate)

- The lambda (regularization parameter) reduces overfitting of the model
- It reduces the variance of the model's estimated regression parameters
- However, it adds bias to the estimate
- Therefore, try out various lambda values and select the lambda value that produces the smallest RMSE value

```
lambdas <- seq(0, 10, 0.25)

rmses <- sapply(lambdas, function(l){

  mu <- mean(edx$rating)

  b_i <- edx %>%
    group_by(movieId) %>%
    summarize(b_i = sum(rating - mu)/(n()+l))

  b_u <- edx %>%
    left_join(b_i, by="movieId") %>%
    group_by(userId) %>%
    summarize(b_u = sum(rating - b_i - mu)/(n()+l))

  predicted_ratings <-
    validation %>%
    left_join(b_i, by = "movieId") %>%
    left_join(b_u, by = "userId") %>%
    mutate(pred = mu + b_i + b_u) %>%
    pull(pred)

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

Add Regularized Model RMSE results to a table for later comparison

Part 4.4 Insights

RMSE has improved to **0.8630292**

STEP 5: EVALUATION

The RMSE values produced by the different models are reproduced in the table below

Model	RMSE
Baseline RMSE	1.0524433
Movie Effect	0.9411063
Movie Effect & User Effect	0.8635899
Regularized Model	0.8630292

RESULTS

Modeling Results & Model Performance

To recap, the goal of this project is to create a machine learning algorithm that predicts movie ratings in the validation dataset as if they were unknown.

RMSE value is used to evaluate the effectiveness of the models. A lower RMSE value indicates a better fit.

In summary, the lowest RMSE value is **0.8630292** , achieved from the **Regularized Model**.

CONCLUSION

Brief Summary

This report explained the steps in creating a machine learning algorithm to predict movie ratings in the validation dataset from the MovieLens dataset.

The various machine learning algorithms trained on the training dataset was then used to predict the movie ratings on the validation dataset. Ultimately, iterating subsequent models helped to improve the RMSE result.

Limitations and Future Work

For the purpose of this project, the MovieLens dataset used was a subset of the full dataset.
To improve results of predicted ratings in future work, one can utilize the full MovieLens dataset, which includes variables like *tags* that have been assigned by users. This informative variable is likely to improve model results.

Github

This is the link to the GitHub repository containing the reports in PDF format, Rmd format and R script:
https://github.com/Nicole-Yong/HarvardX-MovieLens