MovieLens Final

Benjamin Khoo

2024-09-19

Introduction

The aim of this project is to design a machine learning algorithm to create a movie recommendation system using the MovieLens dataset. This is a dataset containing 9000055 observations of 6 variables. The first few lines of code have been provided to generate the edx dataset and the final holdout test set.

Run provided code to load MovieLens dataset and create final holdout test set.

```
library(tidyverse)
library(caret)
# MovieLens 10M dataset:
# https://grouplens.org/datasets/movielens/10m/
# http://files.grouplens.org/datasets/movielens/ml-10m.zip
options(timeout = 120)
dl <- "ml-10M100K.zip"
if(!file.exists(dl))
  download.file("https://files.grouplens.org/datasets/movielens/ml-10m.zip", dl)
ratings_file <- "ml-10M100K/ratings.dat"
  unzip(dl, ratings_file)
movies_file <- "ml-10M100K/movies.dat"
if(!file.exists(movies_file))
  unzip(dl, movies_file)
ratings <- as.data.frame(str_split(read_lines(ratings_file), fixed("::"), simplify = TRUE),
                         stringsAsFactors = FALSE)
colnames(ratings) <- c("userId", "movieId", "rating", "timestamp")
ratings <- ratings %>%
  mutate(userId = as.integer(userId),
         movieId = as.integer(movieId),
         rating = as.numeric(rating),
         timestamp = as.integer(timestamp))
movies <- as.data.frame(str_split(read_lines(movies_file), fixed("::"), simplify = TRUE),
                        stringsAsFactors = FALSE)
colnames(movies) <- c("movieId", "title", "genres")
movies <- movies %>%
  mutate(movieId = as.integer(movieId))
movielens <- left_join(ratings, movies, by = "movieId")
# Final hold-out test set will be 10% of MovieLens data
set.seed(1, sample.kind="Rounding") # if using R 3.6 or later
# set.seed(1) # if using R 3.5 or earlier
test_index <- createDataPartition(y = movielens$rating, times = 1, p = 0.1, list = FALSE)
```

```
edx <- movielens[-test_index,]
temp <- movielens[test_index,]
# Make sure userId and movieId in final hold-out test set are also in edx set
final_holdout_test <- temp %>%
  semi_join(edx, by = "movieId") %>%
  semi_join(edx, by = "userId")
# Add rows removed from final hold-out test set back into edx set
removed <- anti_join(temp, final_holdout_test)
edx <- rbind(edx, removed)
rm(dl, ratings, movies, test_index, temp, movielens, removed)
```

```
# Installing package format R to keep code tidy
install.packages("formatR", repos = "http://cran.us.r-project.org")
```

```
## Installing package into 'C:/Users/benkh/AppData/Local/R/win-library/4.3'
## (as 'lib' is unspecified)
```

```
library(formatR)
```

[Methods] Data visualisation and exploration

The aim of the following code is to visualise the data set, preliminary exploration and check for NA values.

```
edx %>%
    as_tibble()
```

```
## # A tibble: 9,000,055 x 6
##    userId movieId rating timestamp title                                genres
##     <int>   <int>  <dbl>     <int> <chr>                                <chr>
##  1      1     122      5 838985046 Boomerang (1992)                     Comed~
##  2      1     185      5 838983525 Net, The (1995)                      Actio~
##  3      1     292      5 838983421 Outbreak (1995)                      Actio~
##  4      1     316      5 838983392 Stargate (1994)                      Actio~
##  5      1     329      5 838983392 Star Trek: Generations (1994)        Actio~
##  6      1     355      5 838984474 Flintstones, The (1994)              Child~
##  7      1     356      5 838983653 Forrest Gump (1994)                  Comed~
##  8      1     362      5 838984885 Jungle Book, The (1994)              Adven~
##  9      1     364      5 838983707 Lion King, The (1994)                Adven~
## 10      1     370      5 838984596 Naked Gun 33 1/3: The Final Insult (1~ Actio~
## # i 9,000,045 more rows
```

```
# The dataset consists of 9,000,055 rows and 6 columns
edx %>%
    summarise_all(~sum(is.na(.)))
```

```
##   userId movieId rating timestamp title genres
## 1      0       0      0         0     0      0
```

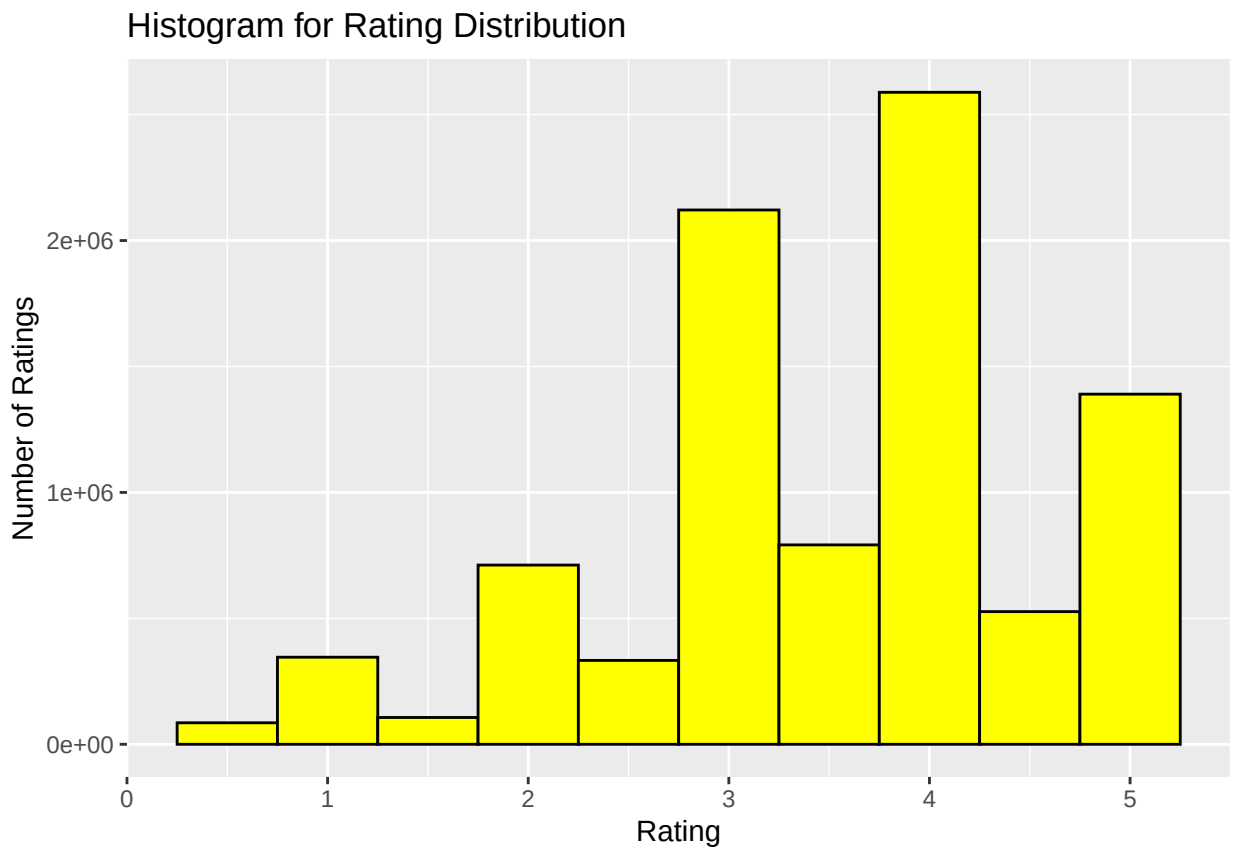
```r
# No NA values are noted in the dataset
edx %>%
    group_by(title) %>%
    summarize(total_ratings = n())
```

```
## # A tibble: 10,676 x 2
##    title                                                      total_ratings
##    <chr>                                                              <int>
##  1 "\"Great Performances\" Cats (1998)"                                   4
##  2 "'Round Midnight (1986)"                                              45
##  3 "'Til There Was You (1997)"                                          269
##  4 "'burbs, The (1989)"                                                1343
##  5 "'night Mother (1986)"                                               196
##  6 "*batteries not included (1987)"                                     438
##  7 "...All the Marbles (a.k.a. The California Dolls) (1981)"             21
##  8 "...And God Created Woman (Et Dieu... créa la femme) (1956)"          70
##  9 "...And God Spoke (1993)"                                             21
## 10 "...And Justice for All (1979)"                                      550
## # i 10,666 more rows
```

```r
edx %>%
    group_by(rating) %>%
    summarize(count = n()) %>%
    arrange(desc(count))
```

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

```r
# This table shows the movies with the highest ratings.
edx %>%
    group_by(movieId) %>%
    ggplot(aes(rating)) + geom_histogram(binwidth = 0.5, fill = "yellow", col = "black") +
    ggtitle("Histogram for Rating Distribution") + ylab("Number of Ratings") + xlab("Rating")
```



The dataset consists of 6 columns and 9,000,055 ratings. The columns are userId, movieId, rating, timestamp, title and genre. There are no NA values in this dataset. In this dataset, there are 10676 movies, which have a rating of 0.5-5. There is a right skew for ratings and that the most common rating is 4.

[Methods] Transforming year of movie and and year of rating to year format, then deriving age of movie at time of rating

The lubridate package was then used to determine the year of rating from the timestamp, followed by extraction of movie year using stringr package. The age of movie at the point of rating was then calculated by subtracting the year movie was released from the year of rating . Negative age of movie values were set to 0.

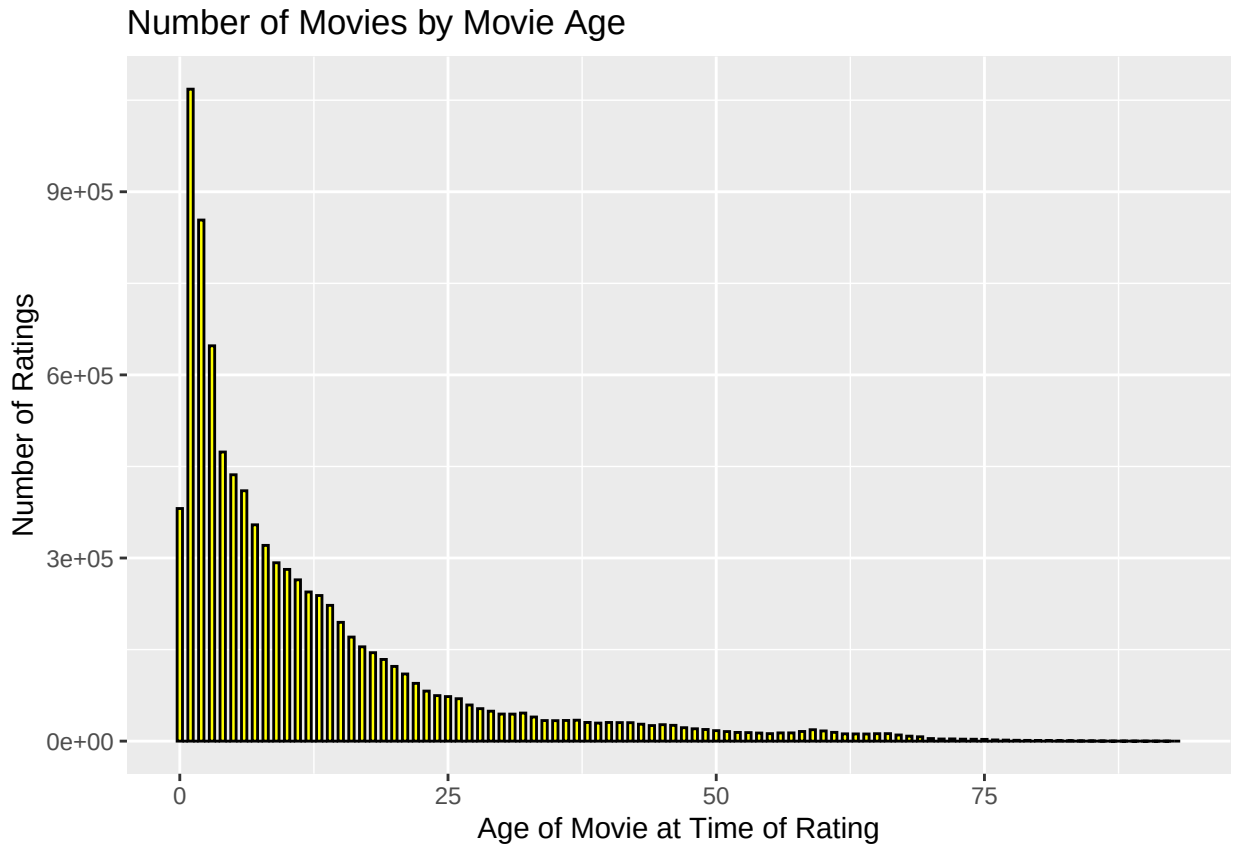
```
library(lubridate)
edx <- edx %>%
    mutate(year_rating = as_datetime(edx$timestamp))
edx <- edx %>%
    mutate(year_rating = year(year_rating))
# Extract movie year and calculate movie age at time of rating
edx <- edx %>%
    mutate(year_movie = substr(edx$title, nchar(edx$title) - 4, nchar(edx$title) -
        1))
range(edx$year_movie)
```

```
## [1] "1915" "2008"
```

```
edx <- edx %>%
    mutate(age_movie = year_rating - as.numeric(year_movie))
# As year of rating cannot precede year of movie, negative values of movie age
# at year of rating were set to 0.
range(edx$age_movie)
```

```
## [1] -2 93
```

```
edx <- edx %>%
    mutate(age_movie = ifelse(age_movie < 0, 0, age_movie))
```



This shows that most movies are rated almost immediately after they are released with a left skew.

[Methods] Edx dataset divided into test and train sets

```
set.seed(1, sample.kind = "Rounding")
```

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

```
test_index <- createDataPartition(y = edx$rating, times = 1, p = 0.1, list = FALSE)
edx_train <- edx[-test_index,]
edx_test <- edx[test_index,]
```

[Methods] Calculate baseline RMSE using the mean rating of the training set as the average, mu.

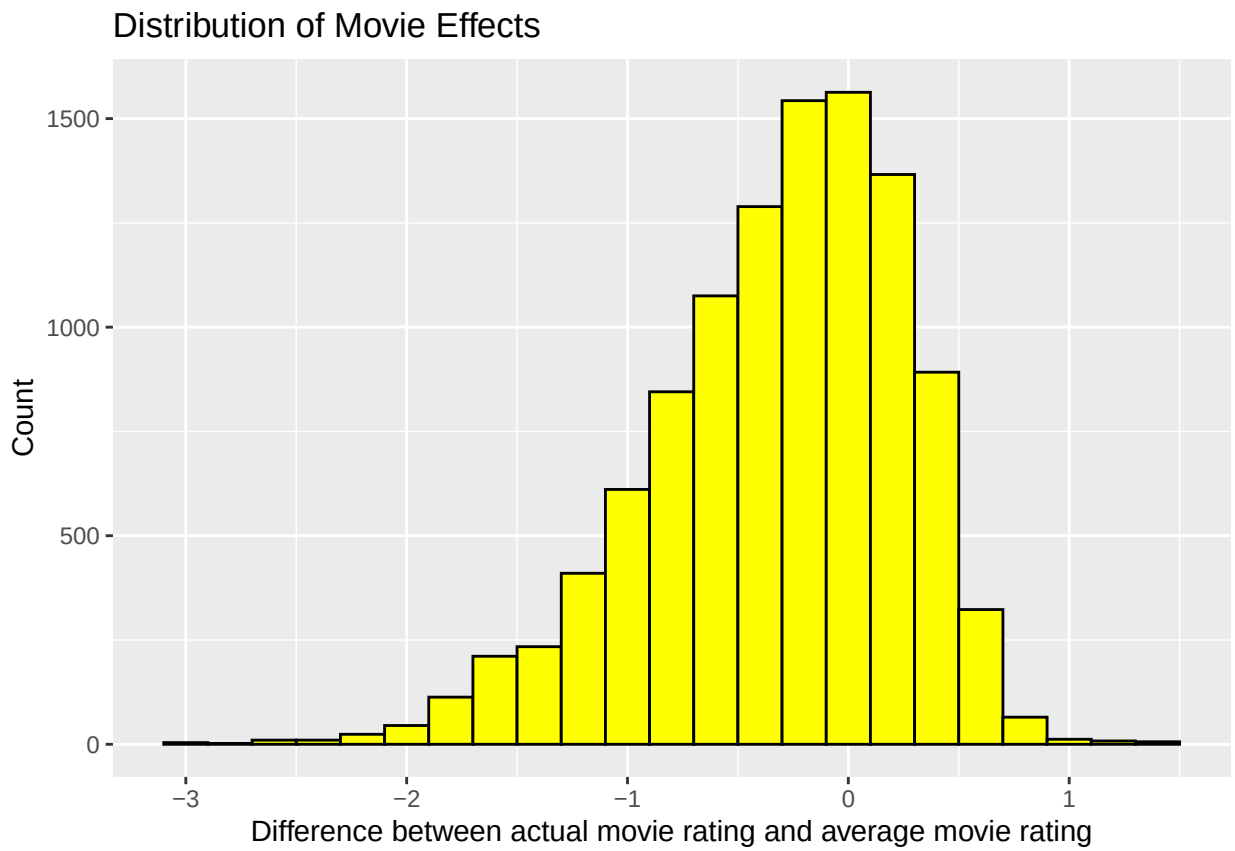
```
RMSE <- function(actual_rating, predicted_rating){
  sqrt(mean((actual_rating - predicted_rating)^2))
}
mu <- mean(edx_train$rating)
naive_RMSE <- RMSE(edx_test$rating, mu)
naive_RMSE
```

```
## [1] 1.060056
```

[Results] Model 1: Average movie rating effects then remove NA to allow model to run

It was recognised that during the joining process, some NAs may be generated thus returning NA when model was run. 17 NAs were identified and were recoded as the average rating. The RMSE obtained by this model was 0.943.

```
average_movie_rating <- edx_train %>%
    group_by(movieId) %>%
    summarize(b_i = mean(rating - mu))
```



This graph shows the difference between the average movie rating for each movie and the average overall rating. It is noted that there can be significant variability for this.

```
model_1 <- edx_test %>%
    left_join(average_movie_rating, by = "movieId") %>%
    mutate(predicted_rating = mu + b_i) %>%
    pull(predicted_rating)
RMSE_AMR <- RMSE(edx_test$rating, model_1)
RMSE_AMR
```

```
## [1] NA
```

```
sum(is.na(model_1))
```

```
## [1] 17
```

```
# Noted 17 NAs were generated during the join procedure.
summary(edx_test %>%
    left_join(average_movie_rating, by = "movieId") %>%
    pull(b_i))
```

```
##      Min.   1st Qu.    Median      Mean   3rd Qu.      Max.      NA's
## -3.012457 -0.293959  0.080178  0.000379  0.363172  1.487543        17
```

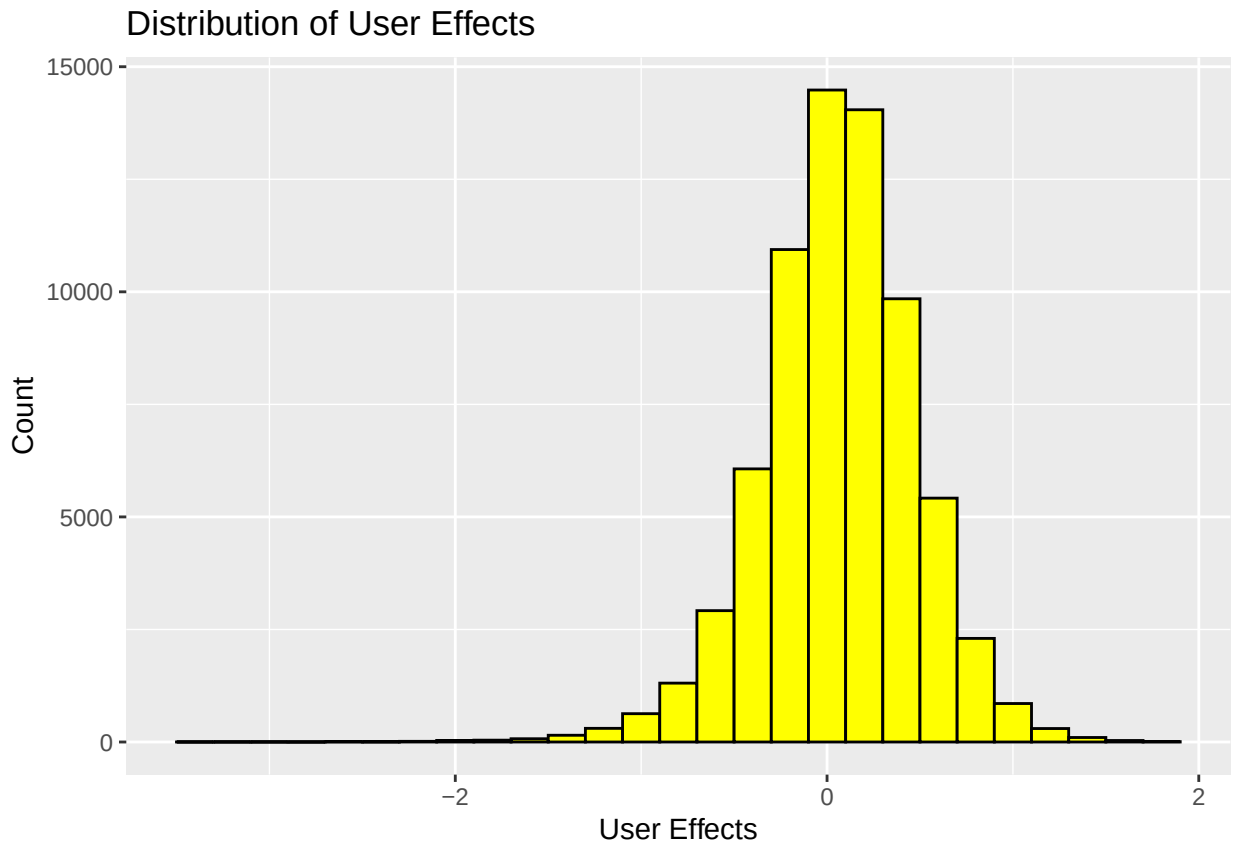
```
model_1[is.na(model_1)] <- mu
RMSE_model_1 <- RMSE(edx_test$rating, model_1)
RMSE_model_1
```

```
## [1] 0.9429666
```

[Results] Model 2

Average movie rating effects + user effects

```
user_average_movie_rating <- edx_train %>%
    left_join(average_movie_rating, by = "movieId") %>%
    group_by(userId) %>%
    summarize(b_u = mean(rating - mu - b_i))
```



User effects are introduced that reflect the biases that individual users have for their movie preferences. This appears to be normally distributed and clustered around 0.

```
model_2 <- edx_test %>%
    left_join(average_movie_rating, by = "movieId") %>%
    left_join(user_average_movie_rating, by = "userId") %>%
    mutate(predicted_rating = mu + b_i + b_u) %>%
    pull(predicted_rating)
```

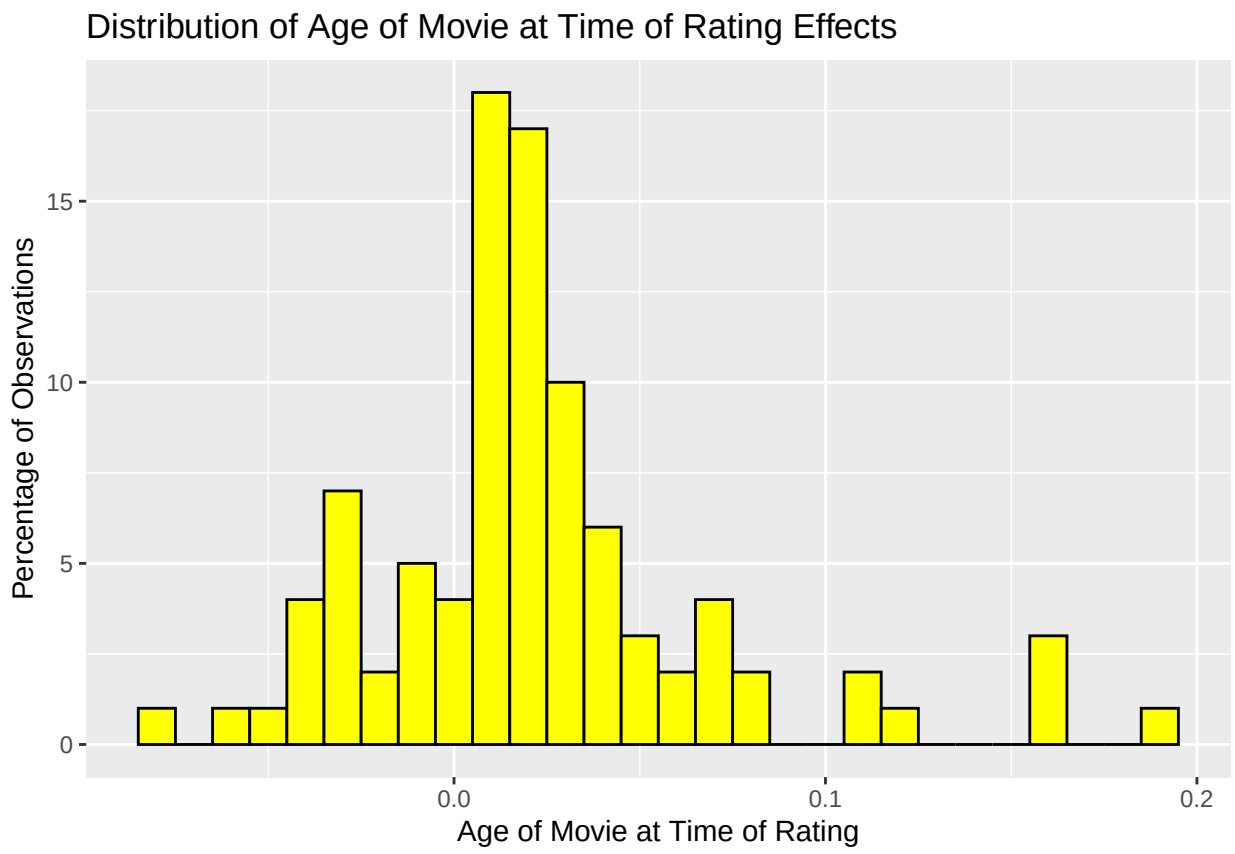
```
model_2[is.na(model_2)] <- mu
RMSE_model_2 <- RMSE(edx_test$rating, model_2)
RMSE_model_2
```

```
## [1] 0.8646915
```

Adding user effects to the average movie rating effects led to an improvement in the RMSE to 0.865.

[Results] Model 3: Average movie rating effects + user effects + age of movie at time of rating effects

```
age_average_movie_rating <- edx_train %>%
    left_join(average_movie_rating, by = "movieId") %>%
    left_join(user_average_movie_rating, by = "userId") %>%
    group_by(age_movie) %>%
    summarize(b_a = mean(rating - mu - b_i - b_u))
```



This shows that there are biases that incorporate variation among the ratings for films, depending on the number of years that has passed from film release to rating. This model attempts to account for those differences.

```
model_3 <- edx_test %>%
    left_join(average_movie_rating, by = "movieId") %>%
    left_join(user_average_movie_rating, by = "userId") %>%
    left_join(age_average_movie_rating, by = "age_movie") %>%
    mutate(predicted_rating = mu + b_i + b_u + b_a) %>%
    pull(predicted_rating)
model_3[is.na(model_3)] <- mu
RMSE_model_3 <- RMSE(edx_test$rating, model_3)
RMSE_model_3
```

```
## [1] 0.8642672
```

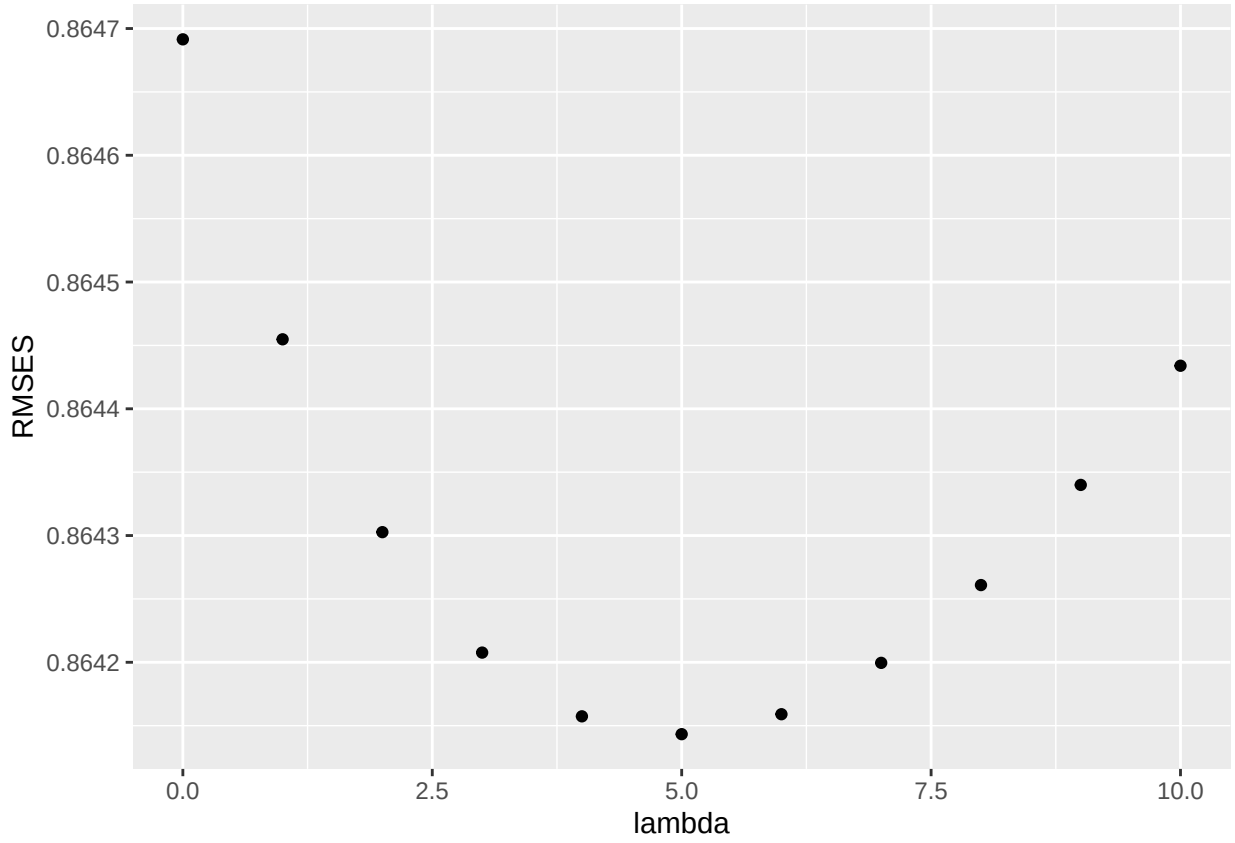
Adding age of movie at time of rating effects to Model 2 led to an improvement in the RMSE to 0.864.

[Results] Model 4: Adding regularisation to Model 2 and finding the optimal lambda

Regularisation is a process which helps to constrain the variability of the effect sizes and thus improve model prediction ability.

```
lambda <- seq(0, 10, 1)
RMSES <- sapply(lambda, function(l) {
    mu <- mean(edx_train$rating)
    b_i <- edx_train %>%
        group_by(movieId) %>%
        summarize(b_i = sum(rating - mu)/(n() + l))
    b_u <- edx_train %>%
        left_join(b_i, by = "movieId") %>%
        group_by(userId) %>%
        summarize(b_u = sum(rating - mu - b_i)/(n() + l))
    predicted_rating <- edx_test %>%
        left_join(b_i, by = "movieId") %>%
        left_join(b_u, by = "userId") %>%
        mutate(pred = mu + b_i + b_u) %>%
        pull(pred)
    predicted_rating[is.na(predicted_rating)] <- mu
    return(RMSE(edx_test$rating, predicted_rating))
})
qplot(lambda, RMSES)
```

```
## Warning: `qplot()` was deprecated in ggplot2 3.4.0.
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_lifecycle_warnings()` to see where this warning was
## generated.
```

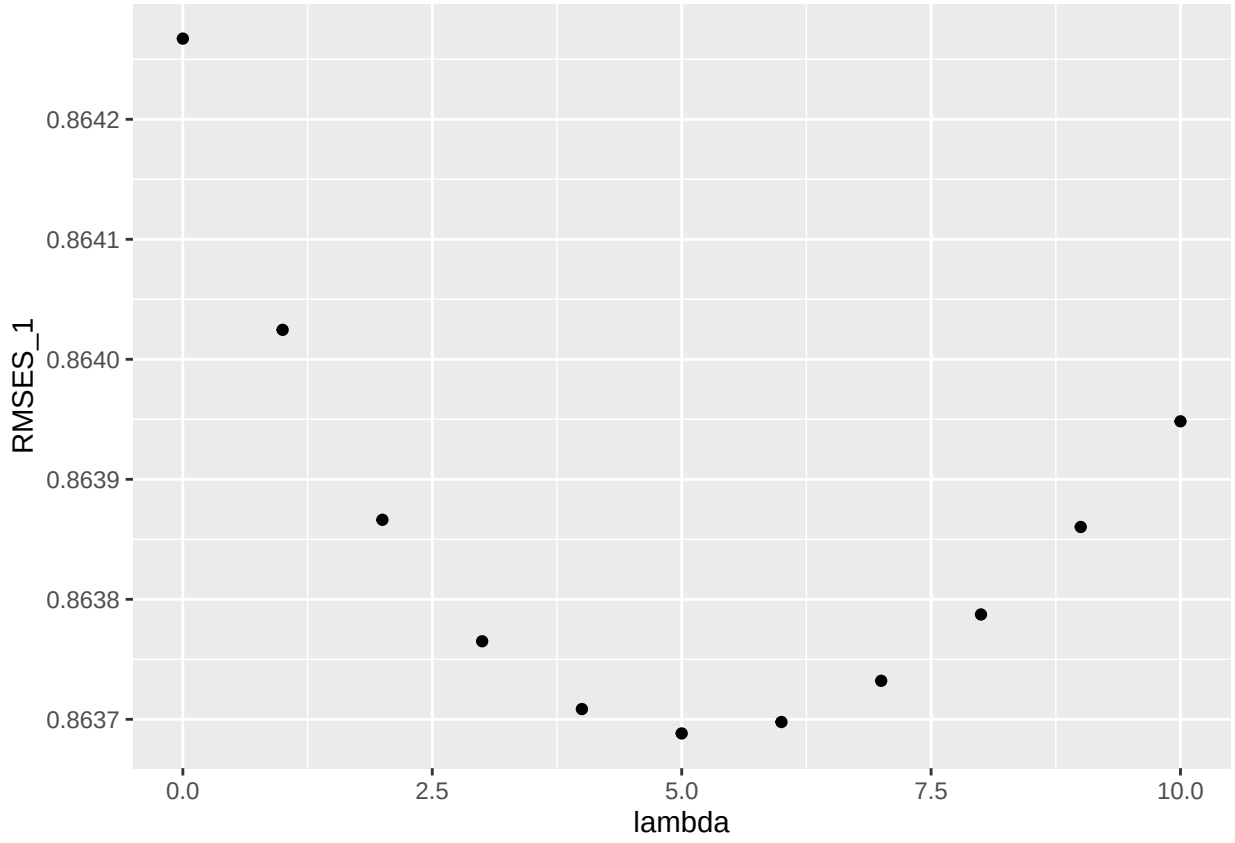



```
RMSE_model_4 <- min(RMSES)
```

[Results] Model 5: Adding regularisation to average movie rating effects + user effects + age of movie at time of rating effects

```
lambda <- seq(0, 10, 1)
RMSES_1 <- sapply(lambda, function(l) {
    mu <- mean(edx_train$rating)
    b_i <- edx_train %>%
        group_by(movieId) %>%
        summarize(b_i = sum(rating - mu)/(n() + l))
    b_u <- edx_train %>%
        left_join(b_i, by = "movieId") %>%
        group_by(userId) %>%
        summarize(b_u = sum(rating - mu - b_i)/(n() + l))
    b_a <- edx_train %>%
        left_join(b_i, by = "movieId") %>%
        left_join(b_u, by = "userId") %>%
        group_by(age_movie) %>%
        summarize(b_a = sum(rating - mu - b_i - b_u)/(n() + l))
    predicted_rating <- edx_test %>%
        left_join(b_i, by = "movieId") %>%
        left_join(b_u, by = "userId") %>%
```

```
        left_join(b_a, by = "age_movie") %>%
        mutate(pred = mu + b_i + b_u + b_a) %>%
        pull(pred)
    predicted_rating[is.na(predicted_rating)] <- mu
    return(RMSE(edx_test$rating, predicted_rating))
})
qplot(lambda, RMSES_1)
```




```
lambda <- 5
b_i <- edx_train %>%
    group_by(movieId) %>%
    summarize(b_i = sum(rating - mu)/(n() + lambda))
b_u <- edx_train %>%
    left_join(b_i, by = "movieId") %>%
    group_by(userId) %>%
    summarize(b_u = sum(rating - mu - b_i)/(n() + lambda))
b_a <- edx_train %>%
    left_join(b_i, by = "movieId") %>%
    left_join(b_u, by = "userId") %>%
    group_by(age_movie) %>%
    summarize(b_a = sum(rating - mu - b_i - b_u)/(n() + lambda))
predicted_rating <- edx_test %>%
    left_join(b_i, by = "movieId") %>%
    left_join(b_u, by = "userId") %>%
    left_join(b_a, by = "age_movie") %>%
```

```
    mutate(pred = mu + b_i + b_u + b_a) %>%
    pull(pred)
predicted_rating[is.na(predicted_rating)] <- mu
RMSE_model_5 <- RMSE(edx_test$rating, predicted_rating)
```

[Results] Mutating the final holdout test set to include new columns that include the age of movie at time of rating.

```
# Mutate final_holdout_test to include age_movie column, set negative values to
# 0
final_holdout_test <- final_holdout_test %>%
    mutate(year_rating = as_datetime(final_holdout_test$timestamp))
final_holdout_test <- final_holdout_test %>%
    mutate(year_rating = year(year_rating))
final_holdout_test <- final_holdout_test %>%
    mutate(year_movie = substr(final_holdout_test$title, nchar(final_holdout_test$title) -
        4, nchar(final_holdout_test$title) - 1))
final_holdout_test <- final_holdout_test %>%
    mutate(age_movie = year_rating - as.numeric(year_movie))
final_holdout_test <- final_holdout_test %>%
    mutate(age_movie = ifelse(age_movie < 0, 0, age_movie))
```

[Results] Calculating RMSE using the final_holdout_test set, using the previously dervied Model 5 which was trained on the edx_train set. This model includes average movie rating effects + user effects + age of movie at time of rating effects with regularisation.

```
lambda <- 5
final_mu <- mean(edx$rating)
final_b_i <- edx %>%
    group_by(movieId) %>%
    summarize(final_b_i = sum(rating - final_mu)/(n() + lambda))
final_b_u <- edx %>%
    left_join(final_b_i, by = "movieId") %>%
    group_by(userId) %>%
    summarize(final_b_u = sum(rating - final_mu - final_b_i)/(n() + lambda))
final_b_a <- edx %>%
    left_join(final_b_i, by = "movieId") %>%
    left_join(final_b_u, by = "userId") %>%
    group_by(age_movie) %>%
    summarize(final_b_a = sum(rating - final_mu - final_b_i - final_b_u)/(n() + lambda))
Final_model_5 <- final_holdout_test %>%
    left_join(final_b_i, by = "movieId") %>%
    left_join(final_b_u, by = "userId") %>%
    left_join(final_b_a, by = "age_movie") %>%
    mutate(pred = final_mu + final_b_i + final_b_u + final_b_a) %>%
    pull(pred)
Final_model_5[is.na(predicted_rating)] <- final_mu
```

```
Final_RMSE_model_5 <- RMSE(final_holdout_test$rating, Final_model_5)
Final_RMSE_model_5
```

```
## [1] 0.864346
```

Conclusion

The initial model training process showed improvement when more parameters were added to the model, and also with the addition of regularization. The final validation model showed that when the best model (Model 5) was tested on the final test set, the RMSE calculated was 0.8643. All the RMSE readings from the training and validation sets are given below.

The development of a well tuned model for the movie rating can lead to improved user experience and thus benefits for companies that distribute movies e.g. Netflix and Disney. Alternatives approaches to machine learning such as matrix factorisation may be considered but these may require increased computation power.

```
options(digits=5)
results <- tibble(Model_Type = c("Baseline", "Training", "Training", "Training",
                  "Training", "Training", "Validation"),
                    Model = c("Baseline RMSE", "Average Rating Effects",
                                    "Average Rating + User Effects",
                                    "Average Rating + User + Age Effects",
                                    "Average Rating + User + Regularization",
                                    "Average Rating + User + Age + Regularization",
                                    "Final RMSE on Final Test Set"),
                       RMSE_Calculated = c(naive_RMSE, RMSE_model_1, RMSE_model_2,
                                           RMSE_model_3, RMSE_model_4,
                                           RMSE_model_5, Final_RMSE_model_5))
```

Executive Summary

The aim of this project is to design a machine learning algorithm to create a movie recommendation system using the MovieLens dataset, which may be found at the reference below. This is a dataset containing 9000055 observations of 6 variables. Code has been provided to generate the edx dataset and the final test set.

The dataset was first visualised and checked for NA values. This showed that there was significant variability. Age of movie at time of rating was a new column derived from the difference between the year a movie was produced and the year of rating.

The dataset (edx) was then partitioned for test and training purposes without involving the final holdout test set. The baseline root mean square deviation (RMSE) was then calculated using the test set and used to determine model performance, a common measure used to determine model performance in machine learning. Care was taken to remove NA values. The average movie rating effects, user effects and age of movie effects were found to influence the rating. These were then added to successive models as follows to improve the model performance.

Model 1: Average movie rating effects

Model 2: Average movie rating effects + user effects

Model 3: Average movie rating effects + user effects + age of movie at time of rating effects

Model 4: Model 2 + regularization

Model 5: Model 3 + regularization

This led to sequential improvement in RMSE score during the model training process using the training set. For Model 4 and 5, regularization was used to improve the model performance. A lambda of 5 showed the most optimal model performance for regularization. Model 5 had the lowest RMSE of 0.8637 on the training set. Following this, we tested the final model (Model 5) against the final test set (final_holdout_test), and a RMSE of 0.8643 was calculated.

```
install.packages("tibble", repos = "http://cran.us.r-project.org")
library(tibble)
options(pillar.sigfig = 5)
results
```

```
## # A tibble: 7 x 3
##   Model_Type Model                                        RMSE_Calculated
##   <chr>      <chr>                                                  <dbl>
## 1 Baseline   Baseline RMSE                                        1.0601
## 2 Training   Average Rating Effects                               0.94297
## 3 Training   Average Rating + User Effects                        0.86469
## 4 Training   Average Rating + User + Age Effects                  0.86427
## 5 Training   Average Rating + User + Regularization               0.86414
## 6 Training   Average Rating + User + Age + Regularization         0.86369
## 7 Validation Final RMSE on Final Test Set                        0.86435
```

Table: Summary of RMSE generated during model development process.

References

1. Introduction to Data Science. Rafael A Irizarry. 2019. https://rafalab.dfci.harvard.edu/dsbook/
2. OpenAI. (2024). ChatGPT 3.5.
3. MovieLens 10M Dataset. https://grouplens.org/datasets/movielens/10m/