# MovieLens Ratings Prediction Project Report

Rachel Cooray

06 June 2024

# Table of Contents

# Introduction

This project aims to predict movie ratings using the MovieLens 10M dataset, employing various models and evaluating them based on metrics like RMSE and NRMSE to enhance recommendation system accuracy. The MovieLens dataset is used for recommender systems and collaborative filtering. It contains a large number of ratings given by users to movies, along with movie metadata such as titles, genres, and release years. The project aims to build and evaluate several models for predicting movie ratings based on user behaviour and movie characteristics. The goal is to identify the most effective model for predicting movie ratings, which can be used to improve recommendation systems. This report outlines the steps taken, including exploratory data analysis, train-test split, model building, evaluation, and conclusions, using the MovieLens 10M dataset, which contains millions of ratings from users on various movies.

# Loading and pre processing

The MovieLens dataset was obtained from the GroupLens website (https://grouplens.org/datasets/movielens/10m/). The dataset includes ratings, movie metadata, and user information.

The preprocessing steps involved downloading and unzipping the dataset, loading the ratings and movie data using data.table::fread and readLines, joining the ratings and movie datasets to create a unified dataset, and splitting the data into training and validation sets while ensuring that users and movies in the validation set are also present in the training set.

# Exploratory Data Analysis (EDA)

The EDA process included examining the dataset structure using glimpse, summarizing key statistics such as unique users, movies, genres, and ratings, and analyzing the distribution of movie ratings and their frequencies.

It also involved visualizing the distribution of ratings over different years, exploring the distribution of ratings across different movie genres, and calculating the age of movies to analyze how it affects ratings.

## Summary Statistics

### Training Set (edx)

```
> summary(edx)
     userId         movieId          rating        timestamp        
 Min.   :    1   Min.   :    1   Min.   :0.500   Min.   :7.897e+08  
 1st Qu.:18124   1st Qu.:  648   1st Qu.:3.000   1st Qu.:9.468e+08  
 Median :35738   Median : 1834   Median :4.000   Median :1.035e+09  
 Mean   :35870   Mean   : 4122   Mean   :3.512   Mean   :1.033e+09  
 3rd Qu.:53607   3rd Qu.: 3626   3rd Qu.:4.000   3rd Qu.:1.127e+09  
 Max.   :71567   Max.   :65133   Max.   :5.000   Max.   :1.231e+09  
    title              genres         
 Length:9000055     Length:9000055    
 Class :character   Class :character  
 Mode  :character   Mode  :character  
```

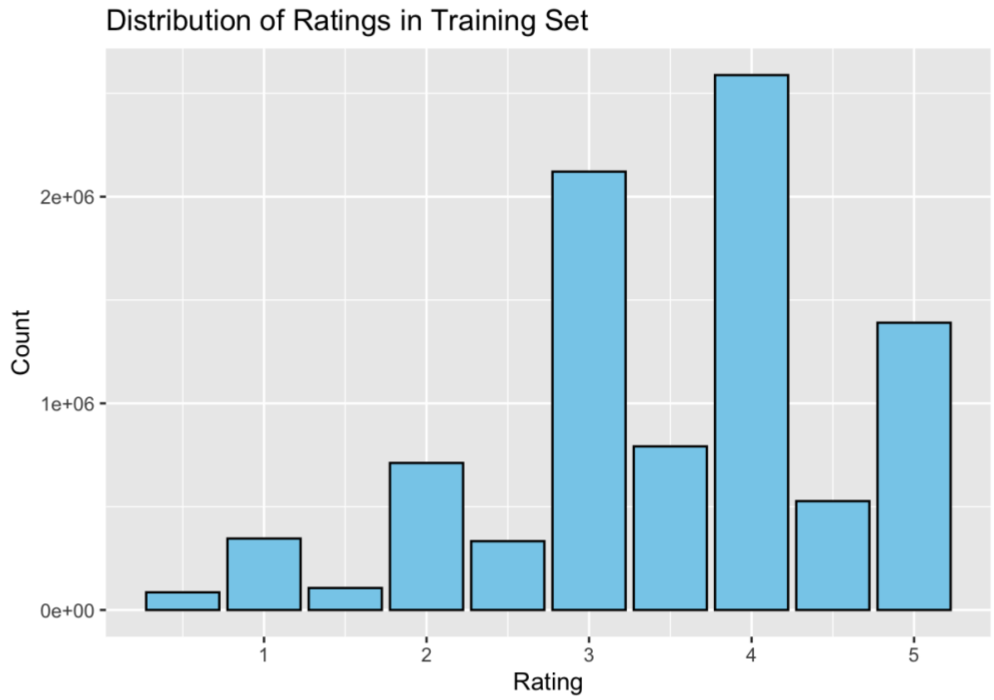
The edx dataset contains userId (a unique identifier for users), movieId (a unique identifier for movies), rating (user rating for a movie on a scale of 0.5 to 5 in 0.5 increments), timestamp (timestamp of the rating), title (title of the movie), and genres (genres of the movie).

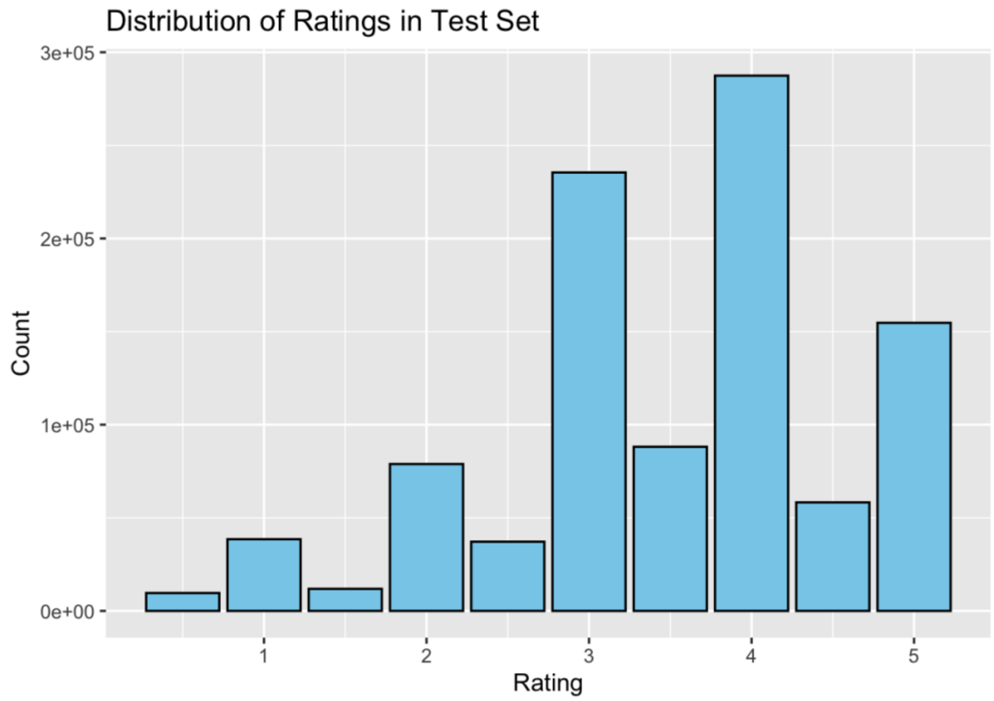
### Final Holdout Test Set

```
> summary(final_holdout_test)
     userId         movieId          rating        timestamp        
 Min.   :    1   Min.   :    1   Min.   :0.500   Min.   :7.897e+08  
 1st Qu.:18096   1st Qu.:  648   1st Qu.:3.000   1st Qu.:9.467e+08  
 Median :35768   Median : 1827   Median :4.000   Median :1.035e+09  
 Mean   :35870   Mean   : 4108   Mean   :3.512   Mean   :1.033e+09  
 3rd Qu.:53621   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:999999      Length:999999     
 Class :character   Class :character  
 Mode  :character   Mode  :character  
```

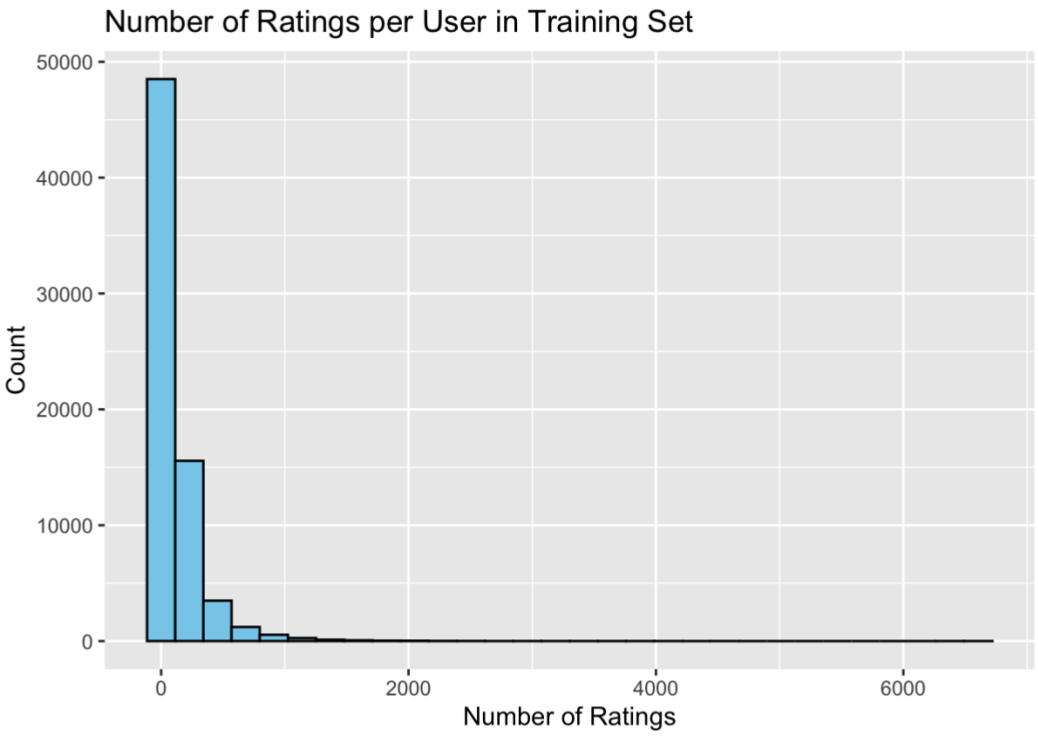
## Visualisation Plots

## Distribution of Ratings

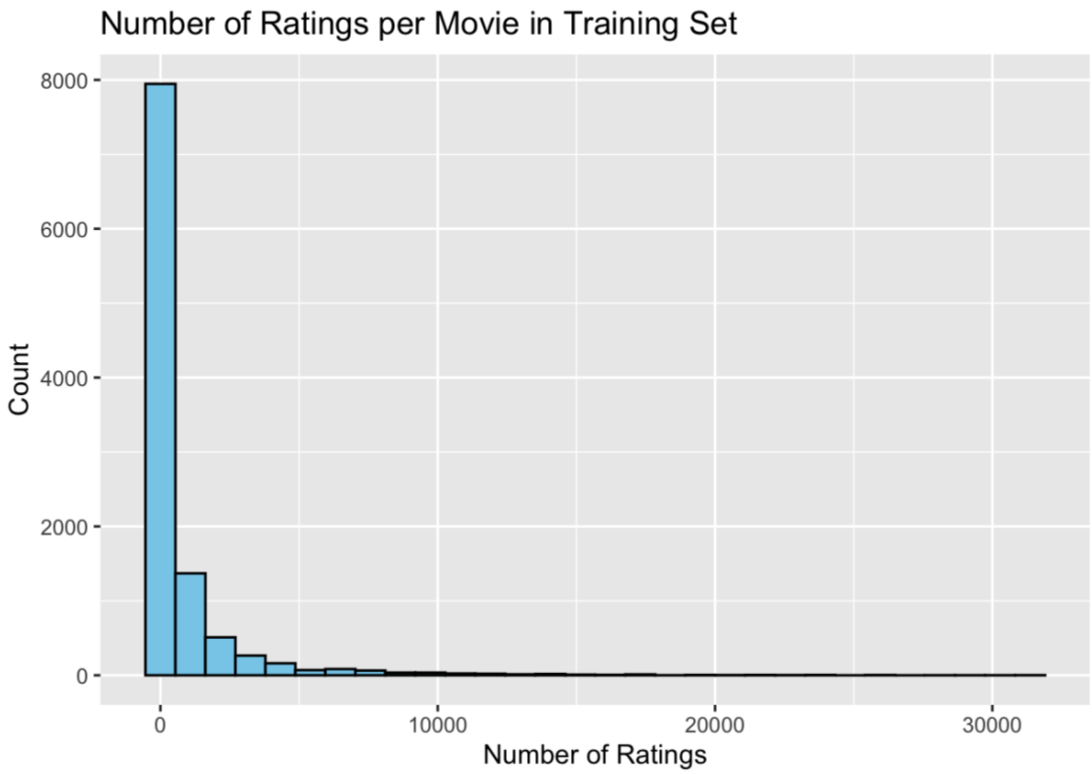


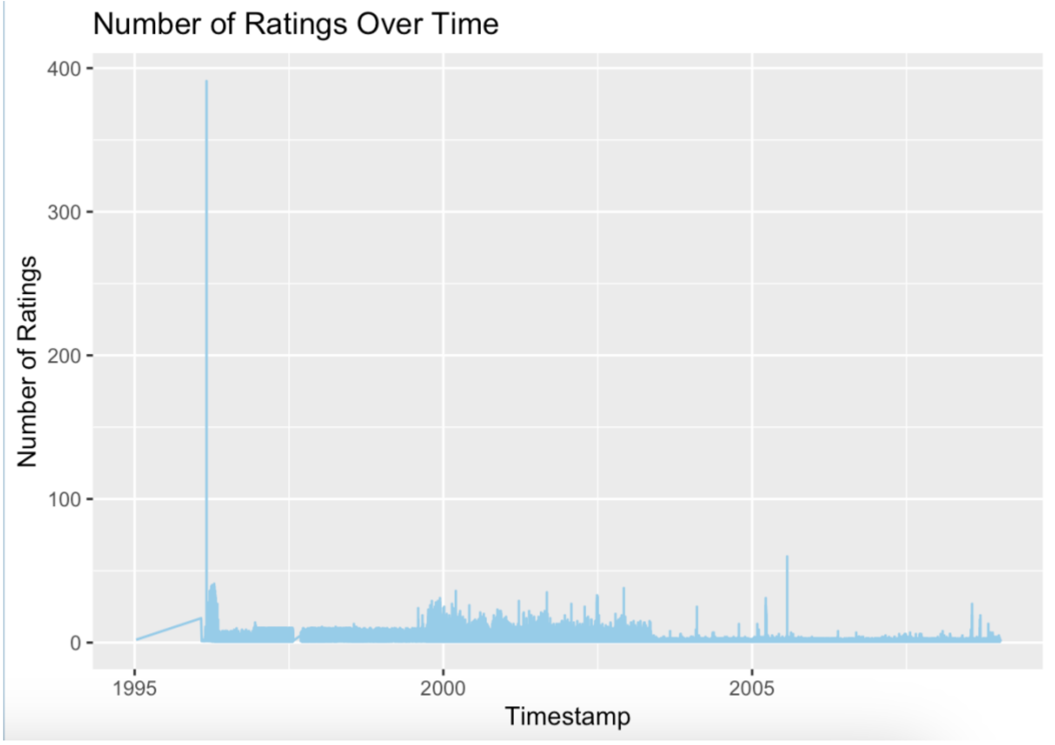

## Number of Ratings per User




## Number of Ratings per Movie




## Number of Ratings Over Time




## Train-Test Split

The dataset was split into a training set (edx) and a final holdout test set (final_holdout_test). The final holdout test set comprises 10% of the original data. Ensuring that the userId and movieId in the final holdout test set are also present in the training set, we maintain consistency in our model evaluation.

## Model building and evaluation

### Model Types Explored

1. Baseline Models
   **Mean Model -** Predicts the mean rating for all entries.
   **Median Model -** Predicts the median rating for all entries.

2. Movie Effects Model
   Incorporates movie-specific biases to adjust the rating predictions.

3. User Effects Model
   Incorporates user-specific biases to adjust the rating predictions.

4. Movie Age Effects Model
   Incorporates the age of the movie to adjust the rating predictions.

5. Regularized Movie and User Effects Model
   Combines both movie-specific and user-specific biases and applies regularization to avoid overfitting.

## Evaluation Metrics

### Root Mean Squared Error (RMSE)

Root Mean Squared Error (RMSE) is a widely used metric to measure the average prediction error of a model in predicting continuous numeric outcomes. In the context of this project, RMSE measures the difference between predicted and actual movie ratings. It is calculated by taking the square root of the average of squared differences between predicted and actual ratings.

RMSE provides a measure of the typical magnitude of the errors produced by the model. Lower values of RMSE indicate better model performance, as they signify that the model's predictions are closer to the actual ratings.

### Normalized Root Mean Squared Error (NRMSE)

Normalized Root Mean Squared Error (NRMSE) is a variation of RMSE that normalizes the error to the range of ratings, making it more interpretable across different datasets. NRMSE scales the RMSE to the range of the actual ratings, providing a percentage error relative to the span of the observed ratings. This makes NRMSE useful for comparing models across different datasets with varying rating scales.

## Results

The results of the model evaluations are summarized as follows.

| Model Type | RMSE | NRMSE |
|---|---|---|
| Baseline Model - Mean | 1.061202 | 0.1247115 |
| Baseline Model - Median | 1.168016 | 0.148448 |
| Movie Effects Model | 0.9439087 | 0.09864637 |
| User Effects Model | 0.978336 | 0.217408 |
| Movie Age Effects Model | 1.197495 | 0. 266099 |
| Regularized Movie and User Effects Model | 0.868537 | 0.1966755 |

## The RMSE returned by testing the algorithm on the final_holdout_test set

The best performing model, the **Regularized Movie and User Effects Model**, achieved an **RMSE of 0.868537** on the final holdout test set. This model combines the movie-specific and user-specific biases and applies regularization to achieve better performance.

# Conclusion

The evaluation results suggest that models incorporating both user and movie effects, along with movie age effects and regularization, tend to perform better in predicting movie ratings. The best-performing model achieved a validation RMSE of 0.868537, indicating its ability to make more accurate predictions compared to simpler baseline models. These findings show the importance of considering both user-specific preferences and movie characteristics when designing effective recommendation systems.

This project involved developing models to predict movie ratings using the MovieLens 10M dataset. After performing exploratory data analysis (EDA) and splitting the data into training and test sets, various models were built and evaluated. The best performance was achieved by the Regularized Movie and User Effects Model, which accurately incorporates both user and movie biases with regularization to prevent overfitting. The final RMSE of 0.868537 demonstrates the effectiveness of this approach in predicting movie ratings.

Overall, this project provides valuable insights into the factors influencing movie ratings and demonstrates various techniques for improving rating prediction accuracy in collaborative filtering systems.

# References

MovieLens 10M Dataset. Available from
https://grouplens.org/datasets/movielens/10m/

# Appendix

## Complete Code

```
# Load necessary packages if not already installed
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(glmnet)) install.packages("glmnet", repos = "http://cran.us.r-project.org")

# Load required libraries
library(tidyverse)
library(caret)
library(glmnet)
```

```
# 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)

# Define file paths for ratings and movies and unzip the dataset if necessary
ratings_file <- "ml-10M100K/ratings.dat"
if(!file.exists(ratings_file))
  unzip(dl, ratings_file)

movies_file <- "ml-10M100K/movies.dat"
if(!file.exists(movies_file))
  unzip(dl, movies_file)

# Load ratings data
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))

# Load movies data
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))

# Merge ratings and movies data
movielens <- left_join(ratings, movies, by = "movieId")

# Set seed for reproducibility
set.seed(1, sample.kind="Rounding")

# Create a train-test split
```

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

# Create a final holdout test set
final_holdout_test <- temp %>%
  semi_join(edx, by = "movieId") %>%
  semi_join(edx, by = "userId")

# Update training set after creating the final holdout test set
removed <- anti_join(temp, final_holdout_test)
edx <- rbind(edx, removed)

# Clean up unnecessary objects from the environment
rm(dl, ratings, movies, test_index, temp, movielens, removed)


# Exploratory Data Analysis (EDA)

# Summary statistics
summary(edx)
summary(final_holdout_test)

# Distribution of ratings
ggplot(edx, aes(x = rating)) +
  geom_bar(fill = "skyblue", color = "black") +
  labs(title = "Distribution of Ratings in Training Set",
       x = "Rating",
       y = "Count")

ggplot(final_holdout_test, aes(x = rating)) +
  geom_bar(fill = "skyblue", color = "black") +
  labs(title = "Distribution of Ratings in Test Set",
       x = "Rating",
       y = "Count")

# Number of ratings per user
ratings_per_user <- edx %>%
  group_by(userId) %>%
  summarise(num_ratings = n())

ggplot(ratings_per_user, aes(x = num_ratings)) +
  geom_histogram(fill = "skyblue", color = "black", bins = 30) +
  labs(title = "Number of Ratings per User in Training Set",
       x = "Number of Ratings",
```

```
      y = "Count")

# Number of ratings per movie
ratings_per_movie <- edx %>%
  group_by(movieId) %>%
  summarise(num_ratings = n())

ggplot(ratings_per_movie, aes(x = num_ratings)) +
  geom_histogram(fill = "skyblue", color = "black", bins = 30) +
  labs(title = "Number of Ratings per Movie in Training Set",
      x = "Number of Ratings",
      y = "Count")

# Convert timestamp to Date format
edx <- edx %>%
  mutate(timestamp = as.POSIXct(timestamp, origin = "1970-01-01"))

# Number of ratings over time
ratings_over_time <- edx %>%
  count(timestamp) %>%
  ggplot(aes(x = timestamp, y = n)) +
  geom_line(color = "skyblue") +
  labs(title = "Number of Ratings Over Time",
      x = "Timestamp",
      y = "Number of Ratings")

ratings_over_time


# Function to calculate NRMSE
nrmse <- function(predicted, actual) {
  rmse <- sqrt(mean((actual - predicted)^2))
  (rmse - min(actual)) / (max(actual) - min(actual))
}

# Model Building and Evaluation

# Baseline Model - Mean and Median
mean_rating <- mean(edx$rating)
median_rating <- median(edx$rating)
mean_rating
median_rating

# Baseline model using mean rating
baseline_mean <- rep(mean_rating, nrow(final_holdout_test))
```

```
# Baseline model using median rating
baseline_median <- rep(median_rating, nrow(final_holdout_test))

# Calculate RMSE and NRMSE for baseline models
rmse_baseline_mean <- sqrt(mean((final_holdout_test$rating - baseline_mean)^2))
nrmse_baseline_mean <- nrmse(baseline_mean, final_holdout_test$rating)

rmse_baseline_median <- sqrt(mean((final_holdout_test$rating - baseline_median)^2))
nrmse_baseline_median <- nrmse(baseline_median, final_holdout_test$rating)

cat("Baseline Model - Mean:\n")
cat("RMSE:", rmse_baseline_mean, "\n")
cat("NRMSE:", nrmse_baseline_mean, "\n\n")

cat("Baseline Model - Median:\n")
cat("RMSE:", rmse_baseline_median, "\n")
cat("NRMSE:", nrmse_baseline_median, "\n\n")

##############

# Movie Effects Model - Incorporating movie-specific biases (b_i)
movie_mean_rating <- edx %>%
  group_by(movieId) %>%
  summarise(mean_rating = mean(rating))

overall_mean_rating <- mean(edx$rating)

movie_effects_model <- edx %>%
  left_join(movie_mean_rating, by = "movieId") %>%
  group_by(movieId) %>%
  summarise(b_i = mean(mean_rating - overall_mean_rating))

final_holdout_test_movie_effects <- final_holdout_test %>%
  left_join(movie_effects_model, by = "movieId") %>%
  mutate(b_i = ifelse(is.na(b_i), 0, b_i)) %>%
  select(userId, movieId, rating, b_i)

rmse_movie_effects <- sqrt(mean((final_holdout_test_movie_effects$rating -
(mean(edx$rating) + final_holdout_test_movie_effects$b_i))^2))
nrmse_movie_effects <- nrmse((mean(edx$rating) +
final_holdout_test_movie_effects$b_i), final_holdout_test$rating)

cat("Movie Effects Model:\n")
cat("RMSE:", rmse_movie_effects, "\n")
cat("NRMSE:", nrmse_movie_effects, "\n\n")
```

```
# Calculate overall mean rating
overall_mean_rating <- mean(edx$rating)

# User Effects Model - Incorporating user-specific biases (b_u)
# Calculate user-specific biases (b_u)
user_mean_rating <- edx %>%
  group_by(userId) %>%
  summarise(mean_rating = mean(rating))

user_effects_model <- edx %>%
  left_join(user_mean_rating, by = "userId") %>%
  group_by(userId) %>%
  summarise(b_u = mean(mean_rating - overall_mean_rating))

# Apply user-specific biases to the final holdout test set
final_holdout_test_user_effects <- final_holdout_test %>%
  left_join(user_effects_model, by = "userId") %>%
  mutate(b_u = ifelse(is.na(b_u), 0, b_u)) %>%
  select(userId, movieId, rating, b_u)

# Calculate RMSE and NRMSE for user effects model
rmse_user_effects <- sqrt(mean((final_holdout_test_user_effects$rating -
(overall_mean_rating + final_holdout_test_user_effects$b_u))^2))
nrmse_user_effects <- rmse_user_effects / (max(final_holdout_test$rating) -
min(final_holdout_test$rating))

cat("User Effects Model:\n")
cat("RMSE:", rmse_user_effects, "\n")
cat("NRMSE:", nrmse_user_effects, "\n\n")

# Movie Age Effects Model - Incorporating movie age effects (b_a)
edx <- edx %>%
  mutate(timestamp = as.Date(timestamp, origin = "1970-01-01"))

movie_age_effects_model <- edx %>%
  group_by(movieId) %>%
  summarise(b_a = as.numeric(difftime(mean(timestamp), as.Date("1970-01-01"),
units = "days")))

final_holdout_test_movie_age_effects <- final_holdout_test %>%
  left_join(movie_age_effects_model, by = "movieId") %>%
  mutate(b_a = ifelse(is.na(b_a), 0, b_a)) %>%
  select(userId, movieId, rating, b_a)

rmse_movie_age_effects <- sqrt(mean((final_holdout_test_movie_age_effects$rating -
(mean(edx$rating) + final_holdout_test_movie_age_effects$b_a))^2))
```

```
nrmse_movie_age_effects <- nrmse((mean(edx$rating) + final_holdout_test_movie_age_effects$b_a), final_holdout_test$rating)

cat("Movie Age Effects Model:\n")
cat("RMSE:", rmse_movie_age_effects, "\n")
cat("NRMSE:", nrmse_movie_age_effects, "\n\n")

# Regularize movie and user effects

# Calculate overall mean rating
overall_mean_rating <- mean(edx$rating)

# Calculate movie-specific biases (b_i)
movie_effects_model <- edx %>%
  group_by(movieId) %>%
  summarise(b_i = mean(rating) - overall_mean_rating)

# Calculate user-specific biases (b_u)
user_effects_model <- edx %>%
  group_by(userId) %>%
  summarise(b_u = mean(rating) - overall_mean_rating)

# Combine movie and user effects in the training set
edx_with_effects <- edx %>%
  left_join(movie_effects_model, by = "movieId") %>%
  left_join(user_effects_model, by = "userId") %>%
  mutate(predicted_rating = overall_mean_rating + b_i + b_u)

# Handle missing values
edx_with_effects <- edx_with_effects %>%
  mutate(b_i = ifelse(is.na(b_i), 0, b_i),
         b_u = ifelse(is.na(b_u), 0, b_u),
         predicted_rating = ifelse(is.na(predicted_rating), overall_mean_rating, predicted_rating))

# Apply movie and user effects to the final holdout test set
final_holdout_test_with_effects <- final_holdout_test %>%
  left_join(movie_effects_model, by = "movieId") %>%
  left_join(user_effects_model, by = "userId") %>%
  mutate(b_i = ifelse(is.na(b_i), 0, b_i),
         b_u = ifelse(is.na(b_u), 0, b_u),
         predicted_rating = overall_mean_rating + b_i + b_u)

# Calculate RMSE and NRMSE for movie and user effects model
rmse_movie_user_effects <- sqrt(mean((final_holdout_test_with_effects$rating - final_holdout_test_with_effects$predicted_rating)^2))
```

```
nrmse_movie_user_effects <- rmse_movie_user_effects / (max(final_holdout_test$rating) - min(final_holdout_test$rating))

cat("Movie and User Effects Model:\n")
cat("RMSE:", rmse_movie_user_effects, "\n")
cat("NRMSE:", nrmse_movie_user_effects, "\n\n")
```