Harvard Capstone Project - Movie recommendation system

Karen Antonia Paul

2024-05-30

The Structure:

- Section 1: Introduction
- Section 2: Data Extraction, Data Preparation and Data Exploration
- Section 3: Modeling Analysis
- Section 4: Final Model and Results
- Section 5: Conclusion

Introduction

This project builds a movie recommendation system using the MovieLens dataset and learnings from HarvardX's Data Science Professional Certificate program.

Our Goal: Predicting Movie Ratings with Accuracy. This project aims to predict movie ratings as accurately as possible compared to actual ratings in the validation set. We'll use Root Mean Squared Error (RMSE) to measure accuracy. Our target is to achieve a final predicted RMSE lower than 0.86490.

Method : After examining the data, we need to come up with the best approach to create a model. The final hold-out test set will only be used for validating the final model, till then we will use 20% of edx data set as the test data set.

Data Extraction, Data Preparation and Data Exploration

Load Required Packages & Libraries

Note: this process could take a couple of minutes

```
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(knitr)) install.packages("knitr", repos = "http://cran.us.r-project.org")
if(!require(ggthemes)) install.packages("ggthemes", repos = "http://cran.us.r-project.org")

library(tidyverse)
library(caret)
library(knitr)
library(ggthemes)
```

Data Extraction

```
# MovieLens 10M dataset:
# https://grouplens.org/datasets/movielens/10m/
# http://files.grouplens.org/datasets/movielens/ml-10m.zip

options(timeout = 500)

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"
if(!file.exists(ratings_file)) unzip(dl, ratings_file)

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

Data Preparation

```
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")

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)

#Test set will be 20% of the edx set
set.seed(1, sample.kind="Rounding")
test_index <- createDataPartition(edx$rating
              , times = 1, p = 0.2, list = FALSE)

train_edx <- edx[-test_index,]
test_edx_1 <- edx[test_index,]

# Make sure userId and movieId in test set are also in train set

test_edx <- test_edx_1 %>%
            semi_join(train_edx, by="userId") %>%
            semi_join(train_edx, by="movieId")

# Add rows removed from test set back into train set

removed_data <- anti_join(test_edx_1,test_edx)
train_edx <- rbind(train_edx,removed_data)

rm(removed_data, test_edx_1, test_index)

goal_rmse <- 0.86490
```

Data Exploration

Exploring the data set to see the data class and to check if there are null values

```
head(edx)
```

```
##   userId movieId rating timestamp                         title
## 1      1     122      5 838985046              Boomerang (1992)
## 2      1     185      5 838983525               Net, The (1995)
## 4      1     292      5 838983421               Outbreak (1995)
## 5      1     316      5 838983392               Stargate (1994)
## 6      1     329      5 838983392 Star Trek: Generations (1994)
## 7      1     355      5 838984474       Flintstones, The (1994)
##                           genres
## 1                 Comedy|Romance
## 2           Action|Crime|Thriller
## 4  Action|Drama|Sci-Fi|Thriller
## 5         Action|Adventure|Sci-Fi
## 6 Action|Adventure|Drama|Sci-Fi
## 7         Children|Comedy|Fantasy
```

```
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
##
##
##
```

```
# How many rows and columns are there in the edx dataset?
dimensions <- dim(edx)
names(dimensions) <- c("Number of Rows", "Number of Columns")
dimensions
```

```
##    Number of Rows Number of Columns
##           9000055                 6
```

Deep dive into movie ratings

```
#Number of movies in the edx dataset
count_of_movies <- edx %>%
                   summarise(n_distinct(movieId)) %>%
                   kable(col.names = c("Number of Movies in the Edx Dataset"))
count_of_movies
```

Number of Movies in the Edx Dataset
10677

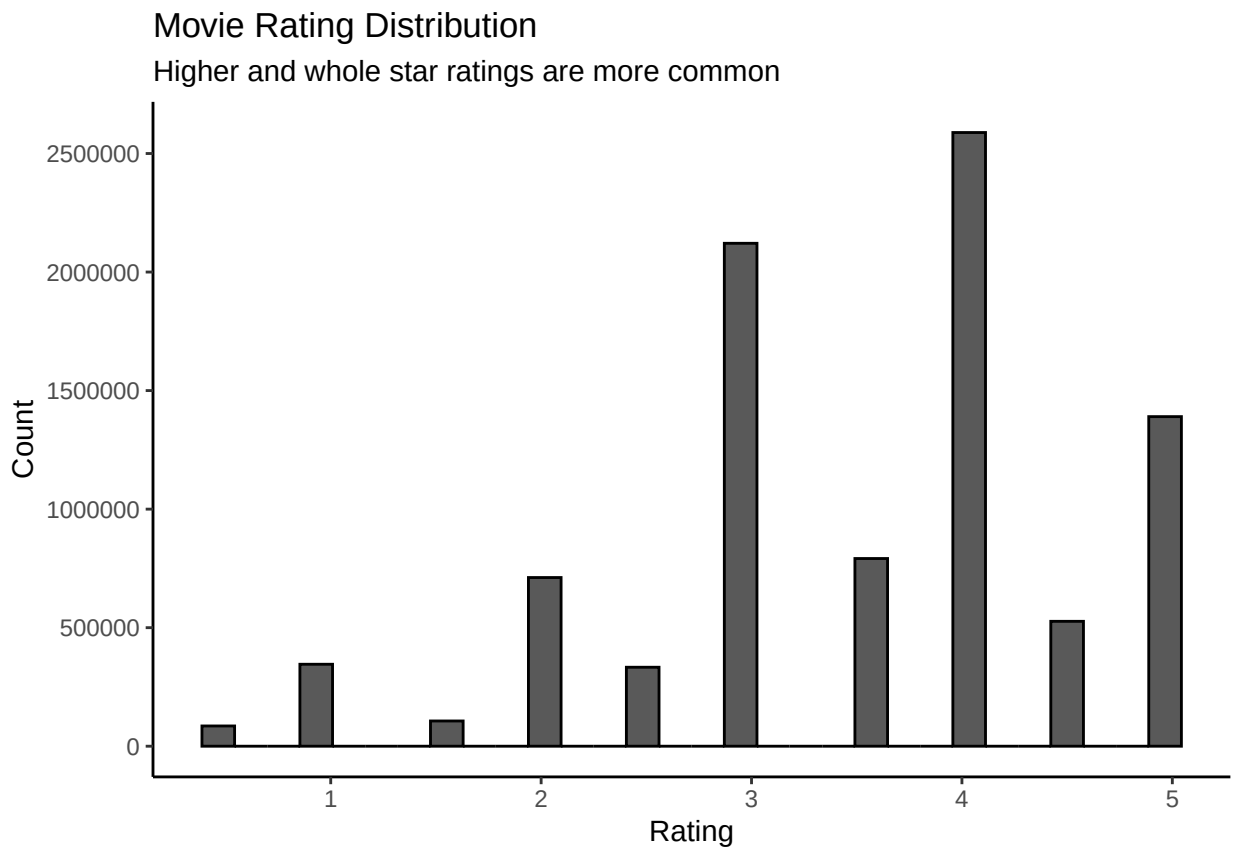
```
# Movie with the greatest number of ratings
highest_rated_movie <- edx %>%
                       group_by(title) %>%
                       summarise(count_of_ratings = n()) %>%
                       arrange(desc(count_of_ratings)) %>%
                       slice(1) %>%
                       kable(col.names = c("Highest Rated Movie", "Number of Ratings"))
highest_rated_movie
```

Highest Rated Movie	Number of Ratings
Pulp Fiction (1994)	31362

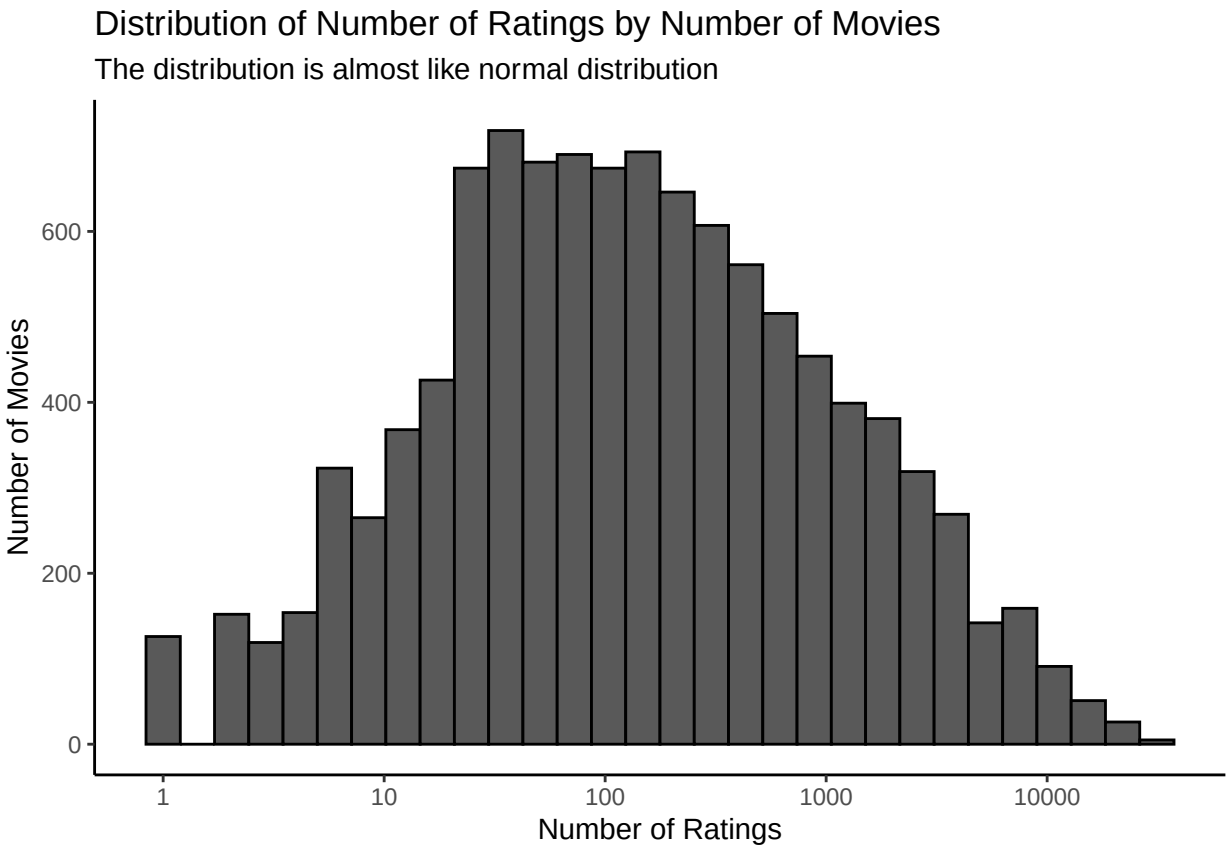
```
#Number of zeros given as ratings in the edx dataset
number_of_zero_ratings <- sum(edx$rating == 0)%>%
                          kable(col.names = c("Number of Zero Ratings in the Edx Dataset"))
number_of_zero_ratings
```

Number of Zero Ratings in the Edx Dataset
0

```
# Which ratings are more common?
edx %>%
    ggplot(aes(rating)) +
    geom_histogram(color = "black") +
    scale_y_continuous(breaks = c(seq(0, 3000000, 500000))) +
    ggtitle("Movie Rating Distribution",
    subtitle = "Higher and whole star ratings are more common") +
    labs(x = "Rating", y = "Count") +
    theme_classic()
```




```
# Distribution of number of ratings per movies
edx %>%
    group_by(movieId) %>%
    summarise(count=n()) %>%
    ggplot(aes(x=count))+
    geom_histogram(color = "black")+
    scale_x_log10()+
    ggtitle("Distribution of Number of Ratings by Number of Movies",
    subtitle = "The distribution is almost like normal distribution") +
    labs(x="Number of Ratings",  y ="Number of Movies") +
    theme_classic()
```



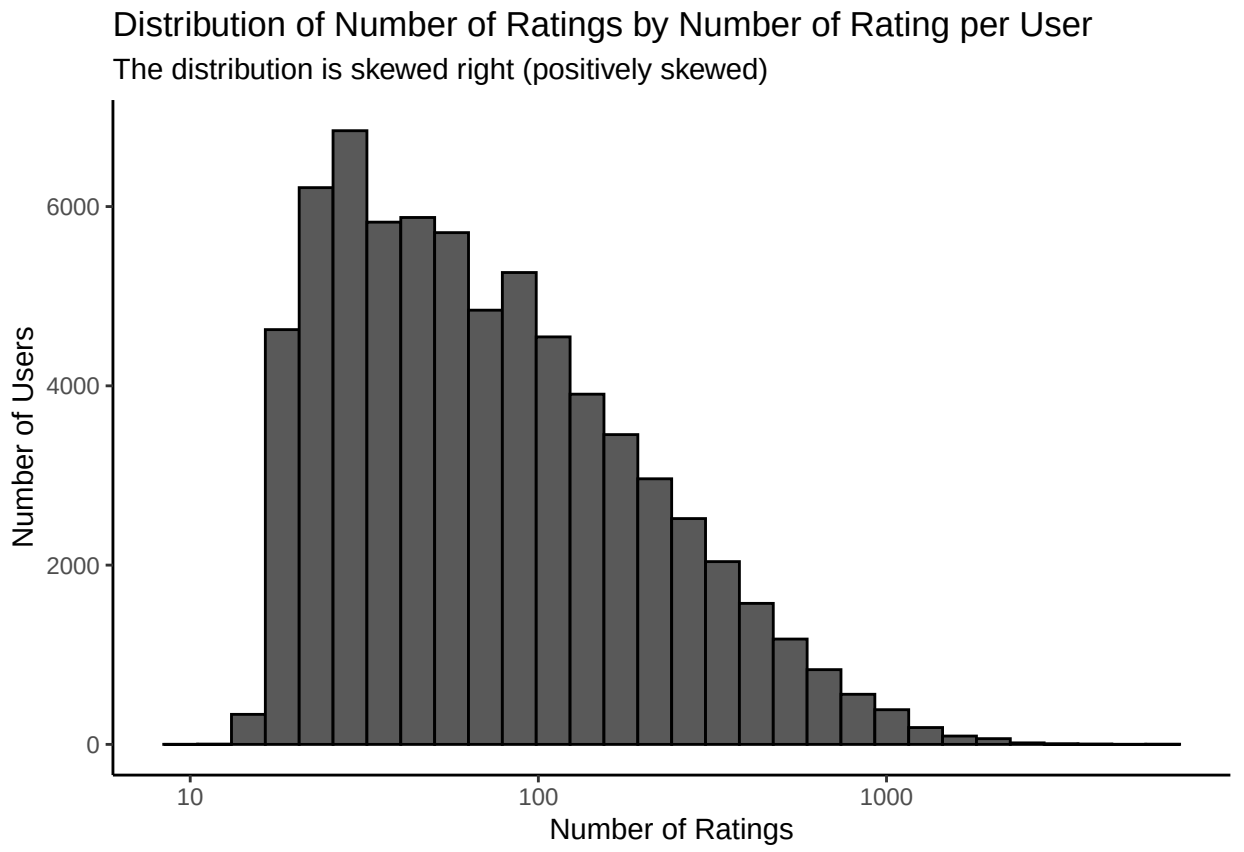
It can be observed that there are some movies which has had only one rating. This uneven spread of ratings means we need to adjust our model to account for it.

Deep dive into users who are rating

```
#Number of users are in the edx dataset
users_who_are_rating <- edx %>%
                        summarise(n_distinct(userId)) %>%
                        kable(col.names = c("Number of Users Who Are Rating"))
users_who_are_rating
```

Number of Users Who Are Rating
69878

```
# Distribution of number of ratings by number of users
edx %>%
    group_by(userId) %>%
    summarise(n=n()) %>%
    ggplot(aes(n)) +
    geom_histogram(color = "black") +
    scale_x_log10() +
    ggtitle("Distribution of Number of Ratings by Number of Rating per User",
    subtitle="The distribution is skewed right (positively skewed)") +
    labs (x="Number of Ratings", y = "Number of Users") +
    theme_classic()
```



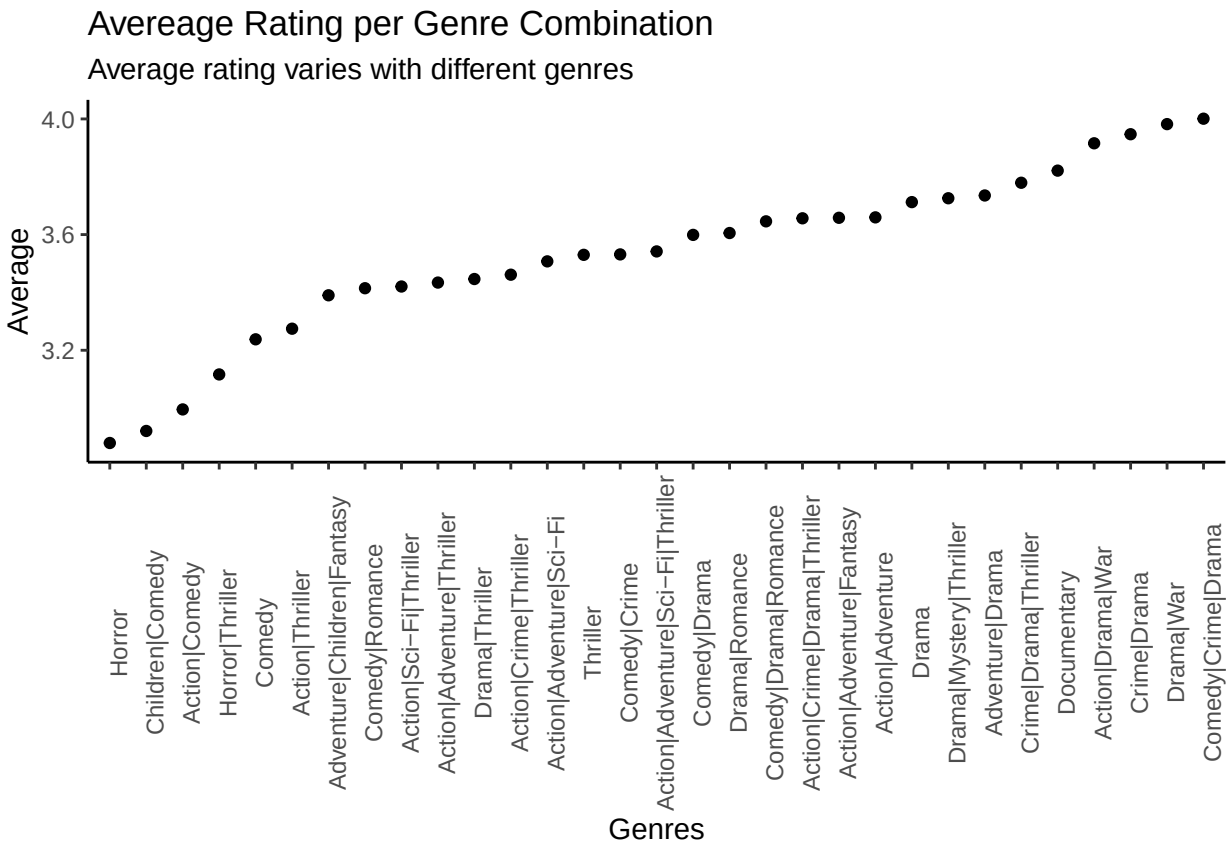
The distribution is skewed right. This means most users tend to rate a moderate number of movies, while a smaller group rates significantly more.

Deep dive into movie genres

```
# How many different genres are there in the data, and the average rating for this genre
edx %>%
    separate_rows(genres, sep = "\\|") %>%
    group_by(genres) %>%
    summarize(count = n(), avg_rating = mean(rating)) %>%
    kable(col.names = c("Distinct Genres","Number of Ratings","Average Ratings"))
```

Distinct Genres	Number of Ratings	Average Ratings
(no genres listed)	7	3.642857
Action	2560545	3.421405
Adventure	1908892	3.493544
Animation	467168	3.600644
Children	737994	3.418715
Comedy	3540930	3.436908
Crime	1327715	3.665925
Documentary	93066	3.783487
Drama	3910127	3.673131
Fantasy	925637	3.501946
Film-Noir	118541	4.011625
Horror	691485	3.269815
IMAX	8181	3.767693
Musical	433080	3.563305
Mystery	568332	3.677001
Romance	1712100	3.553813
Sci-Fi	1341183	3.395743
Thriller	2325899	3.507676
War	511147	3.780813
Western	189394	3.555918

```
# Average rating per genre combinations with more than 50,000 ratings
edx %>%
    group_by(genres) %>%
    summarize(n = n(), average = mean(rating), se = sd(rating)/sqrt(n())) %>%
    filter(n >= 50000) %>%
    mutate(genres = reorder(genres, average)) %>%
    ggplot(aes(x = genres, y = average, ymin = average - 2*se, ymax = average + 2*se)) +
    geom_point() +
    theme_classic() +
    ggtitle("Avereage Rating per Genre Combination",
    subtitle="Average rating varies with different genres") +
    labs (x="Genres", y = "Average") +
    theme(axis.text.x = element_text(angle = 90))
```



The above graph might show a correlation between genres and average ratings. This doesn't necessarily mean the genre itself causes the higher ratings. There could be other factors at play, like popularity or critical acclaim of movies within a genre.

Modeling Analysis

To recommend movies, we use a method that considers how wrong our predictions are (RMSE) and a technique (regularization) to avoid overfitting. Our goal is to make the predictions as accurate as possible, aiming for an RMSE lower than goal RMSE 0.86490. After examining the above data, we can see that the best approach to create this model will be to use an iterative approach.

$$RMSE = \sqrt{\frac{1}{N}\sum_{u,i}(\hat{y}_{u,i} - y_{u,i})^2}$$

Basic Model

This approach predicts the same rating for every movie. It uses the overall average rating from all the movies in the training data.

The formula used for this model is:

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

- $\hat{\mu}$ represents the overall average rating across all movies in the dataset
- $\varepsilon_{i,u}$ represents the random error

```
# Calculate the average of all movies
mu_hat <- mean(train_edx$rating)

# Predict the RMSE on the test set
RMSE_basic <- RMSE(test_edx$rating, mu_hat)

# Save prediction into data frame
rmse_results <- data_frame(Model = "Average movie rating model"
                , RMSE = RMSE_basic
                , Difference_from_goal = round(goal_rmse - RMSE_basic,4))
rmse_results %>% kable()
```

Model	RMSE	Difference_from_goal
Average movie rating model	1.059904	-0.195

Movie Effect Model

When we looked closely at the data, we noticed that some movies didn't have many ratings. These movies with very few ratings also tended to be lesser-known ones.

The formula used for this model is:

$$Y_{u,i} = \hat{\mu} + b_i + \epsilon_{u,i}$$

- $\hat{\mu}$ represents the overall average rating across all movies in the dataset
- $\varepsilon_{i,u}$ represents the random error
- b_i represents the movie bias

```
# Calculate the average of all movies
mu_hat <- mean(train_edx$rating)

# Calculate the average by movie
movies <- train_edx %>%
          group_by(movieId) %>%
          summarize(b_i = mean(rating - mu_hat))

# Predict the RMSE on the test set
RMSE_movies_model <- test_edx %>%
                     left_join(movies, by='movieId') %>%
                     mutate(pred = mu_hat + b_i) %>% pull(pred)

RMSE_movies_result <- RMSE(test_edx$rating,RMSE_movies_model)

# Adding the results to the results data set
rmse_results <- bind_rows(rmse_results
                , data_frame(Model ="Movie Effect Model"
                , RMSE = RMSE_movies_result
                , Difference_from_goal = round(goal_rmse - RMSE_movies_result,4)))
rmse_results %>% kable()
```

Model	RMSE	Difference_from_goal
Average movie rating model	1.0599043	-0.1950
Movie Effect Model	0.9437429	-0.0788

Movie & User Effect Model

In addition to the movie effect, we also saw a user effect. Some people rated movies less often, and their ratings tended to be lower overall.

The formula used for this model is:

$$Y_{u,i} = \hat{\mu} + b_i + b_u + \epsilon_{u,i}$$

- $\hat{\mu}$ represents the overall average rating across all movies in the dataset
- $\varepsilon_{i,u}$ represents the random error
- b_i represents the movie bias
- b_u represents the user bias

```
# Calculate the average of all movies
mu_hat <- mean(train_edx$rating)

# Calculate the average by movie
movies <- train_edx %>%
          group_by(movieId) %>%
          summarize(b_i = mean(rating - mu_hat))

# Calculate the average by user
users <- train_edx %>%
         left_join(movies, by='movieId') %>%
         group_by(userId) %>%
         summarize(b_u = mean(rating - mu_hat - b_i))

# Compute the predicted ratings on test set

RMSE_movies_users_model <- test_edx %>%
                           left_join(movies, by='movieId') %>%
                           left_join(users, by='userId') %>%
                           mutate(pred = mu_hat + b_i + b_u) %>%
                           pull(pred)

RMSE_movies_users_result <- RMSE(test_edx$rating, RMSE_movies_users_model)

# Adding the results to the results data set
rmse_results <- bind_rows(rmse_results
                , data_frame(Model ="Movie + User Effect Model"
                , RMSE = RMSE_movies_users_result
                , Difference_from_goal = round(goal_rmse - RMSE_movies_users_result,4)))
rmse_results %>% kable()
```

Model	RMSE	Difference_from_goal
Average movie rating model	1.0599043	-0.1950
Movie Effect Model	0.9437429	-0.0788
Movie + User Effect Model	0.8659319	-0.0010

Movie, User & Genre Effect Model

In addition to the movie and user effect, we also saw a variation based on the different genres

The formula used for this model is:

$$Y_{u,i} = \hat{\mu} + b_i + b_u + b_r + \epsilon_{u,i}$$

- $\hat{\mu}$ represents the overall average rating across all movies in the dataset
- $\varepsilon_{i,u}$ represents the random error
- b_i represents the movie bias
- b_u represents the user bias
- b_r represents the genre bias

```
# Calculate the average of all movies
mu_hat <- mean(train_edx$rating)

# Calculate the average by movie
movies <- train_edx %>%
          group_by(movieId) %>%
          summarize(b_i = mean(rating - mu_hat))

# Calculate the average by user
users <- train_edx %>%
         left_join(movies, by='movieId') %>%
         group_by(userId) %>%
         summarize(b_u = mean(rating - mu_hat - b_i))

# Calculate the average by genre
genres <- train_edx %>%
          left_join(movies, by='movieId') %>%
          left_join(users, by = 'userId') %>%
          group_by(genres) %>%
          summarize(b_r = mean(rating - mu_hat - b_i - b_u))

# Compute the predicted ratings on test data set

RMSE_movies_users_genre_model <- test_edx %>%
                                 left_join(movies, by='movieId') %>%
                                 left_join(users, by='userId') %>%
                                 left_join(genres, by='genres') %>%
                                 mutate(pred = mu_hat + b_i + b_u + b_r) %>%
                                 pull(pred)

RMSE_movies_users_genre_result <- RMSE(test_edx$rating, RMSE_movies_users_genre_model)

# Adding the results to the results data set
rmse_results <- bind_rows(rmse_results
                , data_frame(Model ="Movie + User + Genre Effect Model"
                , RMSE = RMSE_movies_users_genre_result
                , Difference_from_goal = round(goal_rmse - RMSE_movies_users_genre_result,4)))
rmse_results %>% kable()
```

Model	RMSE	Difference_from_goal
Average movie rating model	1.0599043	-0.1950
Movie Effect Model	0.9437429	-0.0788
Movie + User Effect Model	0.8659319	-0.0010
Movie + User + Genre Effect Model	0.8655941	-0.0007

Regularized Movie & User Effect Model

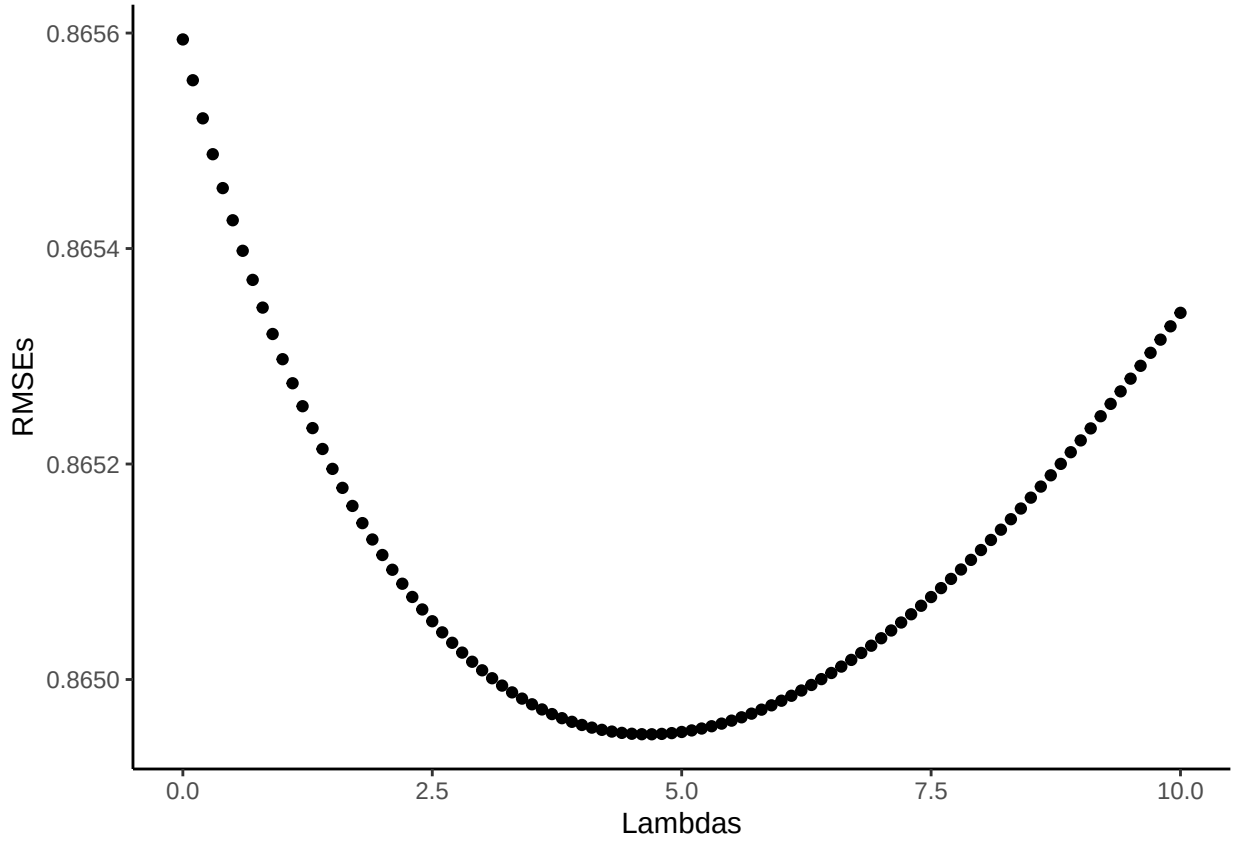
Given the observed influence of movie, user and genre effects on our predictions from the previous models, a crucial step becomes tuning the model to optimize its generalization performance. This necessitates the application of regularization techniques.

Regularization introduces a penalty term into the loss function we aim to minimize during model training. This penalty term specifically targets large values of the movie bias terms b_i and user bias terms b_u. The strength of this penalty is controlled by a hyperparameter known as lambda.

In simpler terms, the model is discouraged from assigning overly large positive or negative values to movie, user and genre effects. This prevents the model from becoming overly reliant on specific features in the training data (overfitting) and enhances its ability to generalize well to unseen data.

```
# Calculate the average of all movies
mu_hat <- mean(train_edx$rating)

# Define a table of lambdas
lambdas <- seq(0, 10, 0.1)

# Compute the predicted ratings on final_holdout_test data set using different values of lambda
rmses <- sapply(lambdas, function(lambda){

# Calculate the average by movie
b_i <- train_edx %>%
       group_by(movieId) %>%
       summarize(b_i = sum(rating - mu_hat) / (n() + lambda))

# Calculate the average by user
b_u <- train_edx %>%
       left_join(b_i, by='movieId') %>%
       group_by(userId) %>%
       summarize(b_u = sum(rating - mu_hat - b_i) / (n() + lambda))

 # Calculate the average by genre
b_r <- train_edx %>%
       left_join(movies, by='movieId') %>%
       left_join(users, by = 'userId') %>%
       group_by(genres) %>%
       summarize(b_r = sum(rating - mu_hat - b_i - b_u)/ (n() + lambda))

# Compute the predicted ratings on test data set
RMSE_regularized_movies_users_model <- test_edx %>%
                                       left_join(b_i, by='movieId') %>%
                                       left_join(b_u, by='userId') %>%
                                       left_join(b_r, by='genres') %>%
                                       mutate(pred = mu_hat + b_i + b_u + b_r) %>%
```

```
                                    pull(pred)

# Predict the RMSE on the test data set
return(RMSE(test_edx$rating, RMSE_regularized_movies_users_model))
})

# Get the lambda value that minimize the RMSE
min_lambda <- lambdas[which.min(rmses)]

# plot the result of lambdas
df <- data.frame(RMSE = rmses, Lambdas = lambdas)
      ggplot(df, aes(lambdas, rmses)) +
      geom_point() +
      theme_classic()  +
      labs(y = "RMSEs",x = "Lambdas")
```




```
# Predict the RMSE on the final_holdout_test set
RMSE_regularized_movies_users_result <- min(rmses)
RMSE_regularized_movies_users_result
```

```
## [1] 0.8649492
```

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

```
## [1] 4.7
```

The optimal Lambda is 4.7. At this value of Lambda, RMSE is minimized to **0.8649**

```
# Adding the results to the results data set
rmse_results <- bind_rows(rmse_results
                , data_frame(Model ="Regularized Movie + User Based Model"
                , RMSE = RMSE_regularized_movies_users_result
                , Difference_from_goal = round(goal_rmse - RMSE_regularized_movies_users_result,4)))
rmse_results %>% kable()
```

Model	RMSE	Difference_from_goal
Average movie rating model	1.0599043	-0.1950
Movie Effect Model	0.9437429	-0.0788
Movie + User Effect Model	0.8659319	-0.0010
Movie + User + Genre Effect Model	0.8655941	-0.0007
Regularized Movie + User Based Model	0.8649492	0.0000

Final Model

The analysis indicates that the regularized model achieved the lowest RMSE (0.8649) which is our target. Having identified the optimal lambda parameter, we will now proceed with building the final model.

This model will be constructed using the entire edx dataset as the training data and validated using the final_holdout_test dataset.

```
# Calculate the average of all movies
mu_hat <- mean(edx$rating)

# Calculate the average by movie
b_i <- edx %>%
       group_by(movieId) %>%
       summarize(b_i = sum(rating - mu_hat) / (n() + best_lambda))

# Calculate the average by user
b_u <- edx %>%
       left_join(b_i, by='movieId') %>%
       group_by(userId) %>%
       summarize(b_u = sum(rating - b_i - mu_hat) / (n() + best_lambda))

# Compute the predicted ratings on final_holdout_test data set
Final_RMSE_regularized_movies_users_genre_model <- final_holdout_test %>%
                                                   left_join(b_i, by='movieId') %>%
                                                   left_join(b_u, by='userId') %>%
                                                   mutate(pred = mu_hat + b_i + b_u) %>%
                                                   pull(pred)

# Predict the RMSE on the final_holdout_test data set
Final_RMSE_regularized_movies_users_genre_result <- (RMSE(final_holdout_test$rating, Final_RMSE_regulari

# Adding the results to the results data set
rmse_results <- bind_rows(rmse_results
                , data_frame(Model ="Final regularized Movie + User + Genre Based Model"
                , RMSE = Final_RMSE_regularized_movies_users_genre_result
                , Difference_from_goal = round(goal_rmse - Final_RMSE_regularized_movies_users_genre_res
rmse_results %>% knitr::kable()
```

Model	RMSE	Difference_from_goal
Average movie rating model	1.0599043	-0.1950
Movie Effect Model	0.9437429	-0.0788
Movie + User Effect Model	0.8659319	-0.0010
Movie + User + Genre Effect Model	0.8655941	-0.0007
Regularized Movie + User Based Model	0.8649492	0.0000
Final regularized Movie + User + Genre Based Model	0.8648208	0.0001

Conclusion

Our project employed a machine learning algorithm for movie rating prediction using the MovieLens dataset. The refined model incorporates movie, user and genre effects, achieving a significant reduction in RMSE compared to the baseline model. This optimized model exhibits an RMSE of 0.8648, exceeding the initial target benchmark of 0.86490, thus establishing itself as the preferred choice for our project.

Limitation

While further improvements to RMSE might be achievable by incorporating additional factors like release year, this project prioritizes achieving the defined target of 0.86490. Therefore, the model focused on movie and user effects to deliver a performant solution within the project scope.

Future Work

Here we explored basic models for predicting ratings. There are some advanced techniques to consider next:

- Matrix Factorization: Breaks down user-item interactions to understand what influences ratings.
- Content-Based Filtering: Recommends similar items users liked before, based on item features.
- Collaborative Filtering: Recommends items similar users enjoyed, leveraging their preferences.

The recommenderlab package lets you try these methods and build more powerful recommendation systems.