Keywords:

Recommendation System - Data Science Project for Movies

GroupLens-MovieLens (ML-10) Dataset

Charles A. Hulebak

Introduction

In the field of data science, machine learning algorithms play a crucial role in extracting insights from large datasets, with recommendation systems being one of the most widely used applications. These systems are designed to suggest products, services, or content to users based on their previous behaviors or preferences.

The MovieLens dataset, created by the GroupLens research lab at the University of Minnesota, has been a popular dataset used in recommender system research. The dataset includes user ratings for movies, allowing the development of systems that predict user preferences based on past ratings. In this project, we focus on building a movie recommendation system using the MovieLens (ML-10) dataset, which consists of 10 million ratings.

Our primary objective is to develop a collaborative filtering recommendation system, leveraging both the 10M and 32M datasets to assess the impact of a larger dataset on recommendation accuracy. By minimizing the Root Mean Squared Error (RMSE), we aim to enhance the accuracy of predictions and optimize the recommendations.[4]

Analysis Approach

Collaborative Filtering

Collaborative filtering is a method commonly used in recommendation systems, where items (in this case, movies) are recommended based on the similarity between users or items. In this project, we use user-item collaborative filtering, where the system recommends movies based on the preferences of users with similar preferences.

Loss Function - RMSE:

The performance of the recommendation system is evaluated using the Root Mean Squared Error (RMSE). RMSE is a widely used metric for regression tasks, and it calculates the square root of the average squared differences between the predicted ratings and the actual ratings. Lower RMSE values indicate more accurate predictions.

The formula for RMSE is:

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

Where: $y_{u,i}$ is the actual rating of movie i by user u, $\hat{y}_{u,i}$ is the predicted rating, N is the number of ratings.

Data Preparation

The data was initially provided in .dat format, but for the purposes of this project, we transitioned the files to .csv format to streamline the analysis. We used the 10M dataset for initial testing, followed by the 32M dataset to compare how the increased data volume would affect the recommendation accuracy.

Data preprocessing steps involved: 1. Loading the movie ratings and metadata from CSV files. 2. Data cleaning: Removing any missing or invalid data entries. 3. Splitting the data into training, test, and validation sets (90/10 split). 4. Feature extraction: Extracting features such as movie genres, user preferences, and ratings.

```
# Import Libraries (and install as necessary)
library(tidyverse)
library(caret)
library(data.table)

# if(!require(dplyr))install.packages("dplyr", repos = "http://cran.us.r-project.org")
library(dplyr, warn.conflicts = FALSE)
options(dplyr.summarise.inform = FALSE)
```

```
# Setup: locate local zip + detect root folder
zip_path <- file.path("data", "ml-10M100K.zip")
stopifnot(file.exists(zip_path))

zip_contents <- unzip(zip_path, list = TRUE)
ml_root <- unique(sub("/.*", "", zip_contents$Name))
ml_root <- ml_root[nzchar(ml_root)]

stopifnot(length(ml_root) == 1)
ml_root <- ml_root[[1]]
```

```
# Process to access the file from source
# url <- "https://files.grouplens.org/datasets/movielens/ml-10m.zip"
# dl  <- tempfile(fileext = ".zip")
# download.file(url, dl, mode = "wb", quiet = TRUE)
```

```
# Path to local zip (adjust if your Rmd is in a different folder)
zip_path <- file.path("data", "ml-10M100K.zip")

if (!file.exists(zip_path)) {
  stop("Can't find the MovieLens zip at: ", normalizePath(zip_path, winslash = "/"),
       "\nExpected: data/ml-10M100K.zip (relative to the Rmd working directory).")
}

# Tiny helper: detect top-level folder inside zip (ml_root)
unzip(zip_path, list = TRUE) |> head()
```

	Name <chr>	Length <dbl>	Date <dttm>
1	ml-10M100K/	0	2016-02-16 11:06:00
2	ml-10M100K/allbut.pl	753	2009-01-05 23:06:00
3	ml-10M100K/movies.dat	522197	2009-01-05 17:02:00
4	ml-10M100K/ratings.dat	265105635	2009-01-05 17:08:00
5	ml-10M100K/README.html	11563	2016-01-29 11:38:00
6	ml-10M100K/split_ratings.sh	1304	2016-02-16 11:06:00

6 rows

```
ml_root <- unique(sub("/.*", "", zip_contents$Name))
ml_root <- ml_root[nzchar(ml_root)]  # drop ""

if (length(ml_root) != 1) {
  stop("Unexpected zip structure. Top-level entries found: ", paste(ml_root, collapse = ", "))
}

ratings_path <- file.path(ml_root, "ratings.dat")
ratings <- data.table::fread(
  text = gsub("::", "\t", readLines(unz(zip_path, ratings_path))),
  col.names = c("userId", "movieId", "rating", "timestamp")
)
```

```
# Movies
movies_path <- file.path(ml_root, "movies.dat")

movies <- stringr::str_split_fixed(
  readLines(unz(zip_path, movies_path)),
  "\\::", 3
)

colnames(movies) <- c("movieId", "title", "genres")

movies <- dplyr::as_tibble(movies) %>%
  dplyr::mutate(
    movieId = as.numeric(movieId),
    title   = as.character(title),
    genres  = as.character(genres)
  )
```

MovieLens Data Exploration

```
# Join the datafiles and define as the movieLens
movielens <- left_join(ratings, movies, by = "movieId")

# Create the test index, which will encompass 10% of the MovieLens data, define training and test
data
set.seed(1, sample.kind="Rounding") # if R 3.5 or earlier, use `set.seed(1)`
```

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

```
test_index <- createDataPartition(y = movielens$rating,
                                  times = 1, p = 0.1, list = FALSE)
train_data <- movielens[-test_index,] #renamed from test_data
temp_data <- movielens[test_index,]   #renamed from temp_data

# Name the validation data by creating joins with specified training data
test_data <- temp_data %>%
  semi_join(train_data, by = "movieId") %>%
  semi_join(train_data, by = "userId")

# Add the removed rows from the validation set into the train data set
removed <- anti_join(temp_data, test_data)

train_data <- rbind(train_data, removed)

rm(test_index, temp_data, removed)

head(movielens)
```

userId <int>	movieId <dbl>	rating <dbl>	timestamp <int>	title <chr>
1	122	5	838985046	Boomerang (1992)
1	185	5	838983525	Net, The (1995)
1	231	5	838983392	Dumb & Dumber (1994)
1	292	5	838983421	Outbreak (1995)
1	316	5	838983392	Stargate (1994)
1	329	5	838983392	Star Trek: Generations (1994)

6 rows | 1-5 of 6 columns

```
# Train data - Data sample
head(train_data)
```

userId <int>	movieId <dbl>	rating <dbl>	timestamp <int>	title <chr>
1	122	5	838985046	Boomerang (1992)
1	185	5	838983525	Net, The (1995)
1	292	5	838983421	Outbreak (1995)
1	316	5	838983392	Stargate (1994)
1	329	5	838983392	Star Trek: Generations (1994)
1	355	5	838984474	Flintstones, The (1994)

6 rows | 1-5 of 6 columns

**Number of Unique Users, Movies, and Genres*

```
# Distinct/Unique Users
UniqueUsers <- length(unique(train_data$userId))
UniqueUsers
```

```
## [1] 69878
```

```
# Number of Movies within the dataset (variables between 1 or more genre combinations)
UniqueMovie <- length(unique(train_data$movieId))
UniqueMovie
```

```
## [1] 10677
```

```
# Number of Unique genres
tibble(count = str_count(train_data$genres, fixed("|")),
  genres = train_data$genres) %>%
  group_by(count, genres) %>%
  summarise('Quantity of Movies by Genre' = n()) %>%
  head(20)
```

count <int>	genres <chr>	Quantity of Movies by Genre <int>
0	(no genres listed)	7
0	Action	24482
0	Adventure	2276
0	Animation	329
0	Children	745
0	Comedy	700889
0	Crime	3197
0	Documentary	70041
0	Drama	733296
0	Fantasy	86

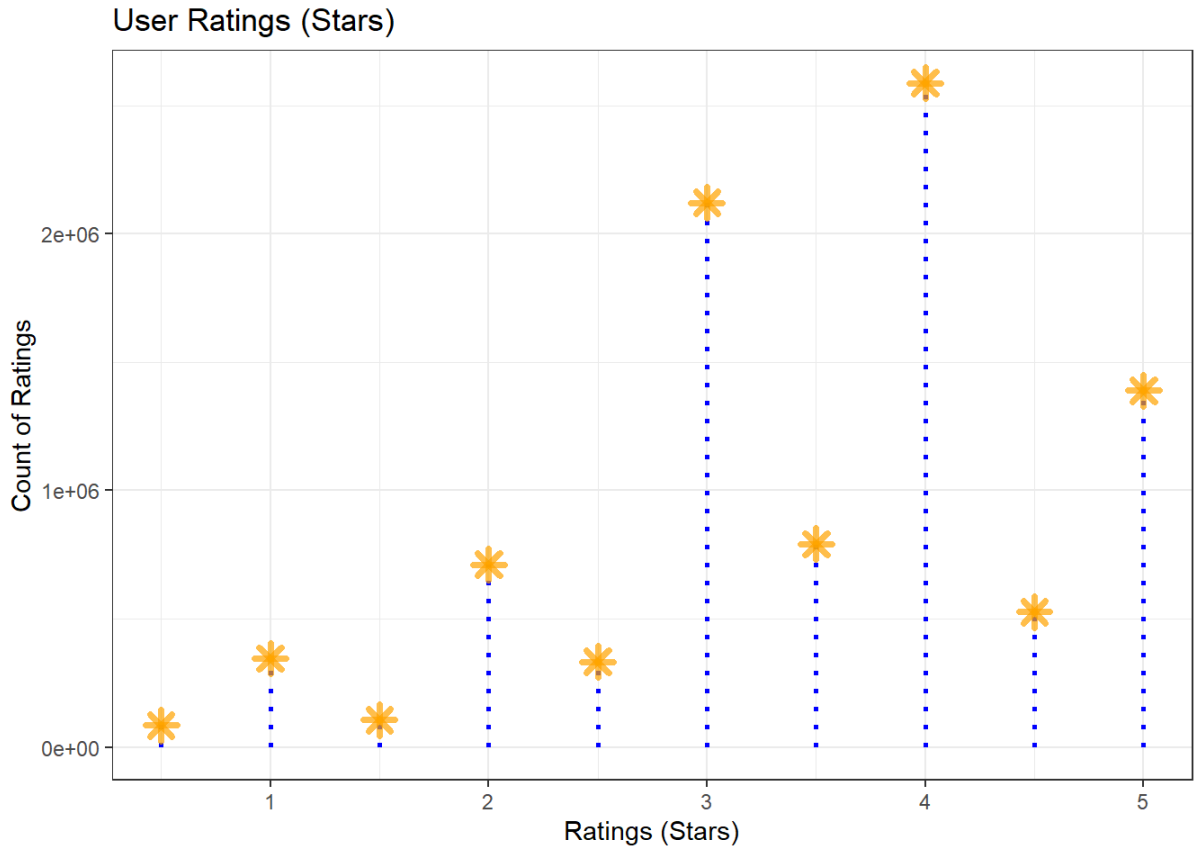
1-10 of 20 rows Previous **1** 2 Next

The results indicate 69,878 users, 10,677 movies, and 19 unique genres.

Movie Rating Distribution (1-5)

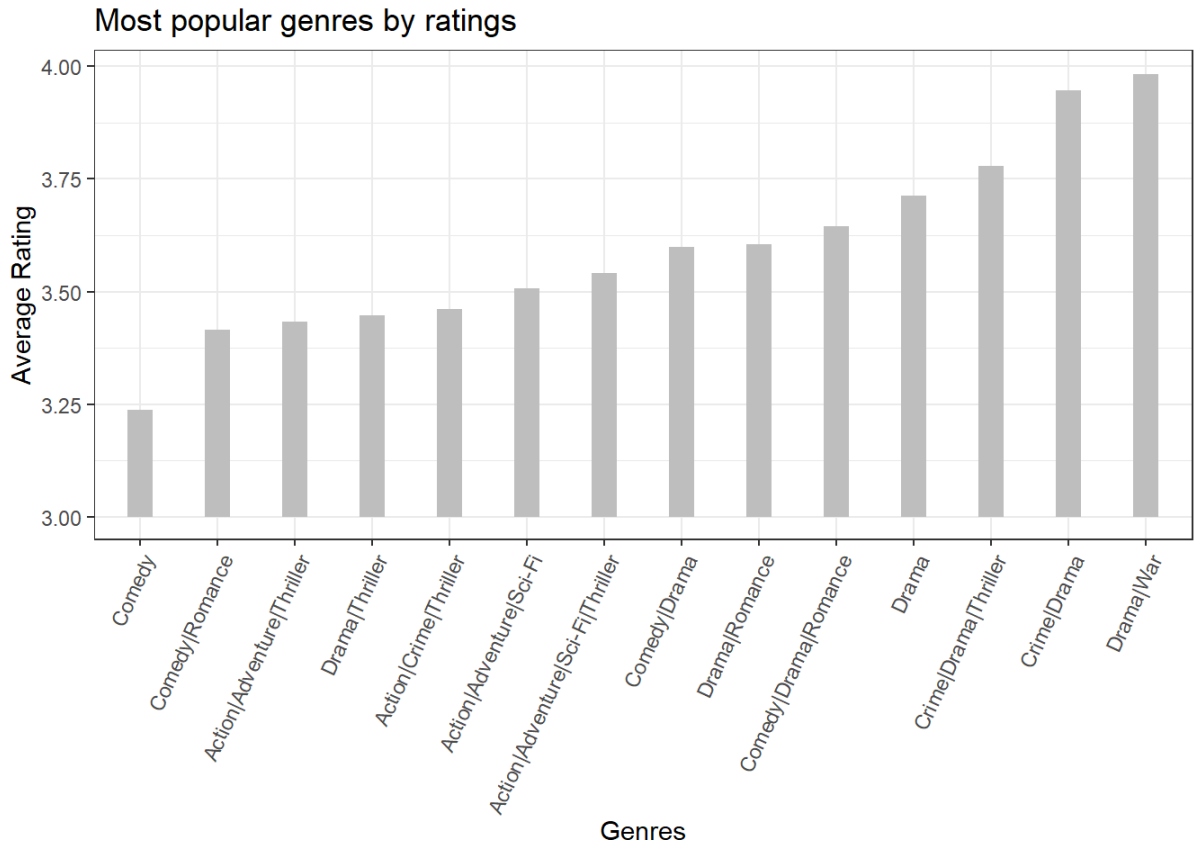
```
# Ratings Distribution and Rating data visualization
library(ggplot2)

train_data %>% group_by(rating) %>%
  summarise(count=n()) %>%
  ggplot(aes(x = rating, y = count)) +
  ggtitle("User Ratings (Stars)") +
  xlab("Ratings (Stars)") +
  ylab("Count of Ratings") +
  geom_segment(aes(x=rating, xend=rating, y=0, yend=count),
               size=1, color="blue", linetype = "dotted") +
  geom_point(size=3, color="orange", fill=alpha("orange", 0.3),
             alpha=0.7, shape=8, stroke=2) +
  theme_bw()
```



Most Popular Genres by Ratings

```
train_data %>% group_by(genres) %>%
  summarize(n = n(), avg = mean(rating), se = sd(rating)/sqrt(n())) %>%
  filter(n >= 100000) %>%
  mutate(genres = reorder(genres, avg)) %>%
  ggplot(aes(x = genres, y = avg, ymin = avg - 2*se, ymax = avg + 2*se)) +
  geom_segment(aes(xend=genres, yend=3), size = 5, color="gray") +
  ggtitle("Most popular genres by ratings") +
  xlab("Genres") +
  ylab("Average Rating") +
  theme_bw() +
  theme(axis.text.x = element_text(angle = 65, hjust = 1))
```



Data Analysis and ML Recommender Development

Creating the Recommender Model

The objective of the ML recommendation system is to generate a model that will make the best (highest rated) recommendations possible. This can be achieved by using regression loss algorithms and tuning model parameters to make predictions.

```
# Define Root Mean Squared Error (RMSE)
RMSE <- function(true_ratings, predicted_ratings){
  sqrt(mean((true_ratings - predicted_ratings)^2))
}

# Creating the model
movie_Id <- train_data %>%
     dplyr::count(movieId) %>%
     top_n(5) %>%
     pull(movieId)
```

```
## Selecting by n
```

```
table_1 <- train_data  %>%
     filter(userId %in% c(13:20)) %>%
     filter(movieId %in% movie_Id) %>%
     select(userId, title, rating) %>%
     spread(title, rating)
table_1
```

userId <int>	Forrest Gump (1994) <dbl>	Jurassic Park (1993) <dbl>	Pulp Fiction (1994) <dbl>
13	*NA*	*NA*	4
16	*NA*	3	*NA*
17	*NA*	*NA*	*NA*
18	*NA*	3	5
19	4	1	*NA*

5 rows | 1-4 of 6 columns

Recommender System Optimization

Through the process of removing null values higher accuracy can be achieved and minimizing loss. RMSE is increased with values other than the mean.

```
# Timestamp
train_data <- train_data %>% select(userId, movieId, rating, title, genres)
test_data <- test_data %>% select(userId, movieId, rating, title, genres)

# Initial: Define RMSE and determine the average
mu <- mean(train_data$rating)
rmse_mu <- RMSE(test_data$rating, mu)
rmses <- data_frame(method = "Average", RMSE = rmse_mu)
```

```
## Warning: `data_frame()` was deprecated in tibble 1.1.0.
## ℹ Please use `tibble()` instead.
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_lifecycle_warnings()` to see where this warning was
## generated.
```

```
rmses
```

method <chr>	RMSE <dbl>
Average	1.061202

1 row

```
# Create the predictor from the training set ratings
# Simplest model $$Y_{u,i}=\mu+\epsilon_{u.i}$$
mu_hat <- mean(train_data$rating)
mu_hat
```

```
## [1] 3.512465
```

```
# Tuning the model for the best RSME (lowest value)
# Movie effects bias, Least Squares Method, $$Y_{u,i}=\mu + b_i + \epsilon_{u,i}$$
m_e_bias <- train_data %>%
  group_by(movieId) %>%
  summarize(b_i = mean(rating - mu))
head(m_e_bias)
```

movieId <dbl>	b_i <dbl>
1	0.4151725
2	-0.3070658
3	-0.3654817
4	-0.6481659
5	-0.4437933
6	0.3028191

6 rows

```
movie_bias_predict <- mu + test_data %>%
  left_join(m_e_bias, by = 'movieId') %>%
  .$b_i
rmse_average <- RMSE(movie_bias_predict, test_data$rating)
MBErmses <- bind_rows(rmses,
                          tibble(method="Movie Bias Effect",
                                     RMSE = rmse_average))
MBErmses
```

method <chr>	RMSE <dbl>
Average	1.0612018
Movie Bias Effect	0.9439087

2 rows

```
# User Effects Bias $$Y_{u,i}=\mu+b_i+b_u+\epsilon_{u,i}$$
u_e_bias <- train_data %>%
  left_join(m_e_bias, by = 'movieId') %>%
  group_by(userId) %>%
  summarize(b_u = mean(rating - mu - b_i))

u_e_predict <- test_data %>%
  left_join(m_e_bias, by='movieId') %>%
  left_join(u_e_bias, by='userId') %>%
  mutate(pred = mu + b_i + b_u) %>%
  .$pred

model_results_u_e <- RMSE(u_e_predict, test_data$rating)
MBUErmses <- bind_rows(rmses,
                           tibble(method="User Bias Effect",
                                      RMSE = model_results_u_e))
MBUErmses
```

method <chr>	**RMSE** <dbl>
Average	1.0612018
User Bias Effect	0.8653488

2 rows

```
# Genre Effects Bias $$Y_{u,i}=\mu+b_i+b_u+\epsilon_{u,i}$$
b_g_bias <- train_data %>%
  left_join(m_e_bias, by = 'movieId') %>%
  left_join(u_e_bias, by = 'userId') %>%
  group_by(genres) %>%
  summarize(b_g = mean(rating - mu - b_i - b_u))

b_g_predict <- test_data %>%
  left_join(m_e_bias, by = 'movieId') %>%
  left_join(u_e_bias, by='userId') %>%
  left_join(b_g_bias, by='genres') %>%
  mutate(pred = mu + b_i + b_u + b_g) %>%
  .$pred

model_results_g_e <- RMSE(b_g_predict, test_data$rating)
MBGErmses <- bind_rows(rmses,
                           tibble(method="Genres Bias Effect",
                                      RMSE = model_results_g_e))
MBGErmses
```

method <chr>	**RMSE** <dbl>
Average	1.0612018
Genres Bias Effect	0.8649469

2 rows

Regularization

Following the recommender system tuning and selection of the most effective method using the genre effects model, the model can be further improved with regularization.

Initial Movie Results

The next step includes analyzing the initial data to review the results.

```
# 5 best and worst ranked movies according to the movie bias
movie_titles <- movielens %>%
  select(movieId, title) %>%
  distinct()

# 5 best movies
# Using movie effects bias only, showing larger error rate
best <- m_e_bias %>% left_join(movie_titles, by="movieId") %>%
  arrange(desc(b_i)) %>%
  pull(title)
best[1:5]
```

```
## [1] "Hellhounds on My Trail (1999)"
## [2] "Satan's Tango (Sátántangó) (1994)"
## [3] "Shadows of Forgotten Ancestors (1964)"
## [4] "Fighting Elegy (Kenka erejii) (1966)"
## [5] "Sun Alley (Sonnenallee) (1999)"
```

```
# 5 worst movies
# Using movie effects bias only, showing larger error rate
worst <- m_e_bias %>% left_join(movie_titles, by="movieId") %>%
  arrange(b_i) %>%
  pull(title)

worst[1:5]
```

```
## [1] "Besotted (2001)"
## [2] "Hi-Line, The (1999)"
## [3] "Accused (Anklaget) (2005)"
## [4] "Confessions of a Superhero (2007)"
## [5] "War of the Worlds 2: The Next Wave (2008)"
```

Regularization Results

```
# Regularization: $$\frac{1}{N}\sum_{u,i}({y}_{u,i} - {μ} - {b}_{i})^2 + \lambda \sum_{i} {b}_{i}^
2$$
# Define lambdas penalty to tune the model, least squares
lambdas <- seq(0, 10, 0.25)

# Define RMSE as a function of mu, b_i, b_u, and predicted ratings
lambda_tune <- sapply(lambdas, function(l){

  mu <- mean(train_data$rating)

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

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

  b_g <- train_data %>%
  left_join(b_i, by = 'movieId') %>%
  left_join(b_u, by = 'userId') %>%
  group_by(genres) %>%
  summarize(b_g = mean(rating - mu - b_i - b_u)/(n()+l))

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

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

```
# Optimizing Lambda
best_lambda <- lambdas[which.min(lambda_tune)]
best_lambda
```

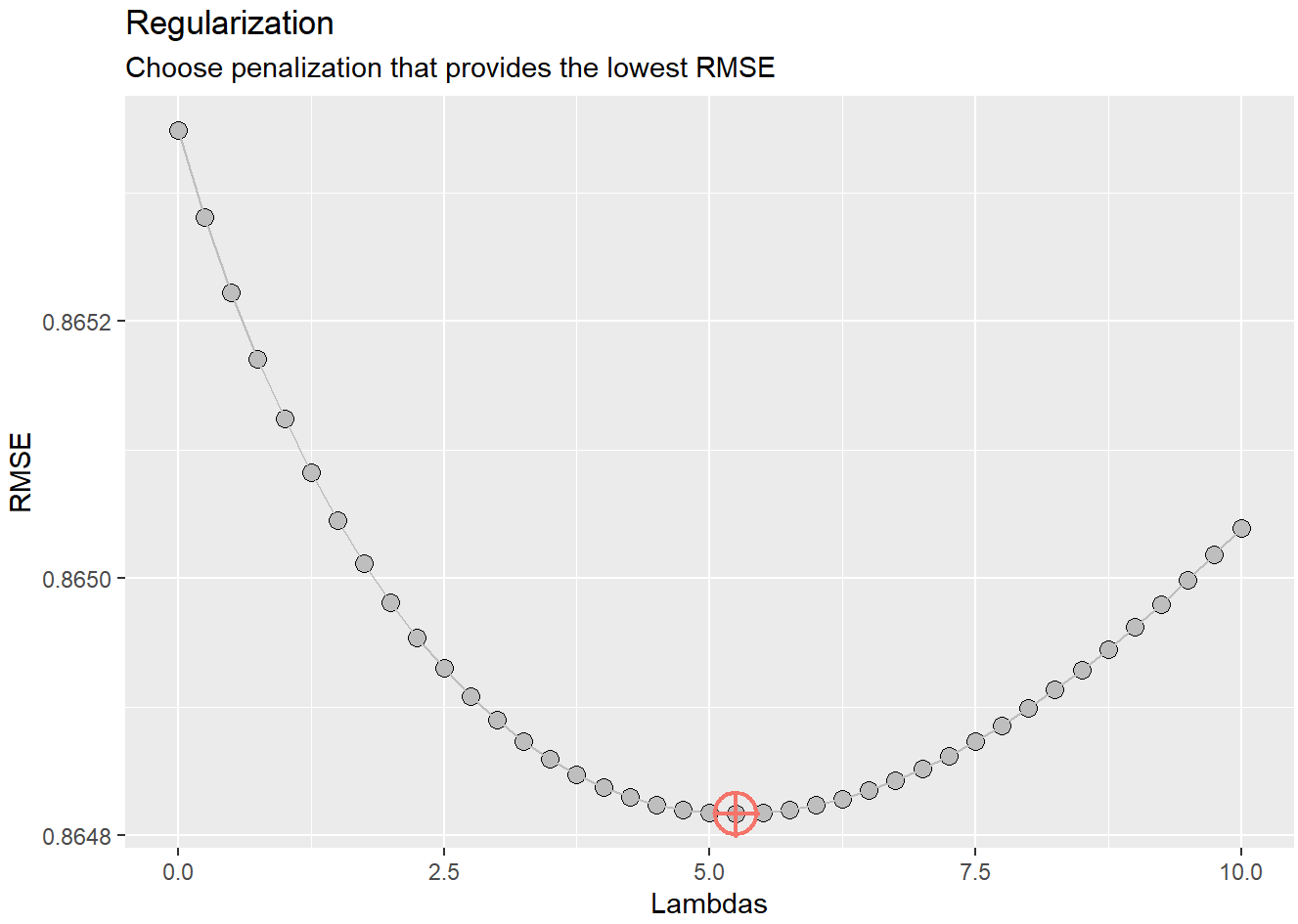
```
## [1] 5.25
```

```
best_penalty <- min(lambda_tune)
best_penalty
```

```
## [1] 0.8648165
```

```
reg_tune_lambda <- tibble(Lambda = lambdas, RMSE = lambda_tune) %>%
  ggplot(aes(x = Lambda, y = RMSE)) +
  geom_point(shape=21, color="black", fill="grey", size=3) +
  geom_line(color="grey") +
  geom_point(aes(y = best_penalty, x = best_lambda, color = 'red'),
             shape = 10, size = 7, stroke = 1.2) +
  ggtitle("Regularization",
          subtitle = "Choose penalization that provides the lowest RMSE") +
  xlab("Lambdas") +
  ylab("RMSE") +
  theme(legend.position = 'none')

plot(reg_tune_lambda)
```



The model is slightly improved with regularization, showing a lower RSME:

```
# Regularized model RSME
rmses <- bind_rows(rmses,
                   tibble(method="Regularization",
                                RMSE = min(lambda_tune)))
rmses
```

method	**RMSE**
<chr>	<dbl>
Average	1.0612018

method	RMSE
<chr>	<dbl>
Regularization	0.8648165

2 rows

Regularized Recommender System Results (Highest model accuracy)

```
# Choose the minimum lambda
lambda <- lambdas[which.min(lambda_tune)]

# Take the mean of mu
mu <- mean(train_data$rating)

# Movie effect bias
b_i <- train_data %>%
  group_by(movieId) %>%
  summarize(b_i = sum(rating - mu)/(n()+lambda))

# User effect bias
b_u <- train_data %>%
  left_join(b_i, by="movieId") %>%
  group_by(userId) %>%
  summarize(b_u = sum(rating - b_i - mu)/(n()+lambda))

# Genre effect bias
b_g <- train_data %>%
  left_join(b_i, by = 'movieId') %>%
  left_join(b_u, by = 'userId') %>%
  group_by(genres) %>%
  summarize(b_g = mean(rating - mu - b_i - b_u)/(n()+lambda))

# Prediction
predict_reg <- train_data %>%
  left_join(b_i, by = "movieId") %>%
  left_join(b_u, by = "userId") %>%
  left_join(b_g, by = "genres") %>%
  mutate(pred = mu + b_i + b_u + b_g) %>%
  pull(pred)
```

Top 5 Movies Overall

```
# Top 5 movies
train_data %>%
  left_join(b_i, by = "movieId") %>%
  left_join(b_u, by = "userId") %>%
  left_join(b_g, by = "genres") %>%
  mutate(pred = mu + b_i + b_u + b_g) %>%
  arrange(-pred) %>%
  group_by(title) %>%
  distinct(title) %>%
  head(5)
```

title <chr>
Shawshank Redemption, The (1994)
Usual Suspects, The (1995)
Schindler's List (1993)
Seven Samurai (Shichinin no samurai) (1954)
Godfather, The (1972)

5 rows

Top 5 Action Movies

```
# Genres can be filtered using the second string in this chunk (i.e Action, Comedy, Documentary, a
nd others listed in the genres category.  Action is shown
movielens %>%
  filter(str_detect(genres,"Action")) %>%
  left_join(b_i, by = "movieId") %>%
  left_join(b_u, by = "userId") %>%
  left_join(b_g, by = "genres") %>%
  mutate(pred = mu + b_i + b_u + b_g) %>%
  arrange(-pred) %>%
  distinct(title) %>%
  head(5)
```

title <chr>
Seven Samurai (Shichinin no samurai) (1954)
Raiders of the Lost Ark (Indiana Jones and the Raiders of the Lost Ark) (1981)
Star Wars: Episode IV - A New Hope (a.k.a. Star Wars) (1977)
Star Wars: Episode V - The Empire Strikes Back (1980)
Matrix, The (1999)

5 rows

Conclusion

The process of building the recommendation system began with exploring the data content, then recognizing how this data could be used to better filter and create a recommendation system. Utilizing a variety of movie bias effects and regularization, this project demonstrated an iterative approach to making small improvements to the RMSE, which resulted in significant improvements to the recommendations.

Future Work

While the current recommendation system performs well with the 10M dataset, there is always room for improvement. Future work could explore additional optimization techniques, such as incorporating matrix factorization to reduce dimensionality and improve performance further.

In addition, there is a larger dataset available with 32M reviews, which can improve the overall rating recommendations using a larger sample size.

References

1. “MovieLens 10M Dataset”, GroupLens, 2019. [online] Available: grouplens.org/datasets/movielens/10m/.

Keywords:

Big data, correlation, data analysis, data science, data visualization, large data set, machine learning, movie effects, movie recommendation, recommender system, regression loss, Residual Mean Squared Error (RMSE), statistics.