MovieLens Project Report

Toluwase Omole

March 4, 2025

Introduction

The goal of this project is to develop an algorithm to predict movie ratings using the MovieLens 10M dataset. The dataset contains user ratings for various movies. The algorithm will be trained on the `edx` dataset and evaluated on the `final_holdout_test` dataset. The performance of the algorithm will be measured using Root Mean Square Error (RMSE).

Data Preparation

Loading Necessary Libraries and Dataset

```
# Install and load necessary packages
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')

library(tidyverse)
library(caret)

# MovieLens 10M dataset
options(timeout = 600)

# Download dataset if not already downloaded
dl <- 'ml-10M100K.zip'
if(!file.exists(dl)) download.file('https://files.grouplens.org/datasets/movielens/ml-10m.zi
p', dl)

# Define file paths
ratings_file <- 'ml-10M100K/ratings.dat'
movies_file <- 'ml-10M100K/movies.dat'

# Unzip files if not already unzipped
if(!file.exists(ratings_file)) unzip(dl, ratings_file)
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), s
tringsAsFactors = FALSE)
colnames(ratings) <- c('userId', 'movieId', 'rating', 'timestamp')
ratings <- ratings %>% mutate(userId = as.integer(userId), movieId = as.integer(movieId), rat
ing = as.numeric(rating), timestamp = as.integer(timestamp))

# Load movies data
movies <- as.data.frame(str_split(read_lines(movies_file), fixed('::'), simplify = TRUE), str
ingsAsFactors = 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')

# Create final hold-out test set (10% of MovieLens data)
set.seed(1, sample.kind = 'Rounding') # if using R 3.6 or later
```

```
## 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)
edx <- movielens[-test_index,]
temp <- movielens[test_index,]

# Ensure 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 = 'userI
d')

# Add rows removed from final hold-out test set back into edx set
removed <- anti_join(temp, final_holdout_test)
```

```
## [1m[22mJoining with `by = join_by(userId, movieId, rating, timestamp, title,
## genres)`
```

```
edx <- rbind(edx, removed)

# Clean up the workspace
rm(dl, ratings, movies, test_index, temp, movielens, removed)
```

Data Visualization

Overview of Dataset

```
# Display the first few rows of the edx dataset
head(edx)
```

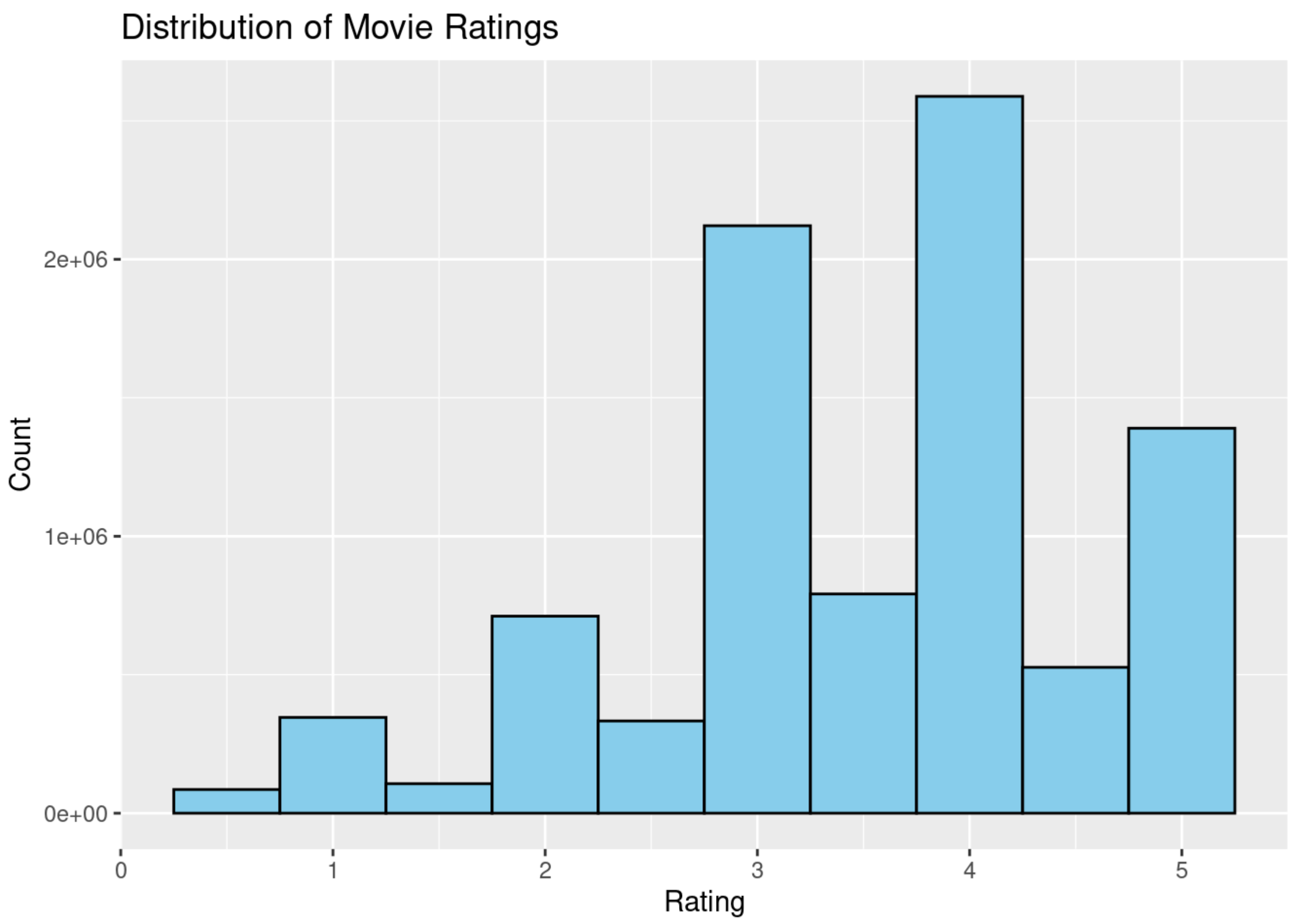
```
##   userId movieId rating timestamp                         title
## 1      1     122      5 838985046              Boomerang (1992)
## 2      1     185      5 838983525               Net, The (1995)
## 4      1     292      5 838983421               Outbreak (1995)
## 5      1     316      5 838983392               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 statistics of the dataset
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  
##                                       
##                                       
## 
```

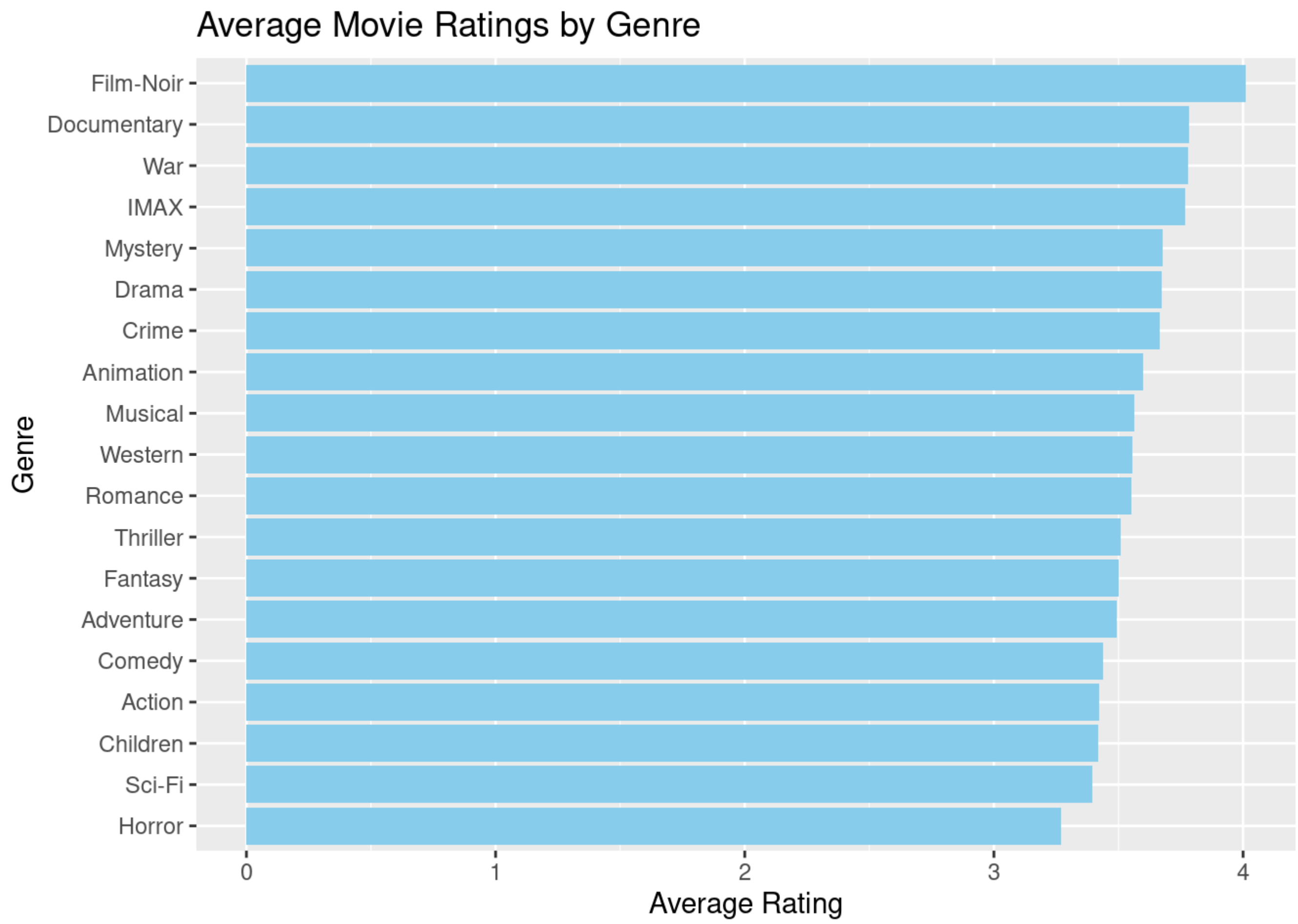
Distribution of Ratings

```
# Plot the distribution of movie ratings
ggplot(edx, aes(x = rating)) +
  geom_histogram(binwidth = 0.5, fill = "skyblue", color = "black") +
  labs(title = "Distribution of Movie Ratings", x = "Rating", y = "Count")
```



Ratings by Genre

```
# Plot average ratings by genre
edx %>%
  separate_rows(genres, sep = "\\|") %>%
  group_by(genres) %>%
  summarize(avg_rating = mean(rating), n = n()) %>%
  filter(n >= 1000) %>%
  ggplot(aes(x = reorder(genres, avg_rating), y = avg_rating)) +
  geom_bar(stat = "identity", fill = "skyblue") +
  coord_flip() +
  labs(title = "Average Movie Ratings by Genre", x = "Genre", y = "Average Rating")
```



Ratings Over Time

edx_sample <- edx %>% sample_n(10000) # Use a 10k-point sample

ggplot(edx_sample, aes(x = as_datetime(timestamp), y = rating)) + geom_bin2d(bins = 50) + # Faster than jitter + smooth labs(title = "Ratings Over Time (Sampled Data)", x = "Date", y = "Rating")

Plot average ratings over time

ggplot(edx, aes(x = date, y = rating)) + geom_jitter(alpha = 0.3, size = 0.5) + geom_smooth(method = "loess", col = "blue") + labs(title = "Average Movie Ratings Over Time", x = "Date", y = "Rating")

````

# Methods/Analysis

## Splitting the Dataset



``` r
# Split edx into training and testing sets
set.seed(1, sample.kind = 'Rounding')
````

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

```
train_index <- createDataPartition(edx$rating, times = 1, p = 0.8, list = FALSE)
train_set <- edx[train_index,]
test_set <- edx[-train_index,]
```

Model Development

Model 1: Naive Model

```
# Naive model - predict the average rating for all movies
mu <- mean(train_set$rating)
naive_rmse <- RMSE(test_set$rating, mu)
naive_rmse
```

```
## [1] 1.059733
```

Model 2: Movie Effect Model

```
# Movie effect model
movie_avgs <- train_set %>%
  group_by(movieId) %>%
  summarize(b_i = mean(rating - mu))

predicted_ratings <- test_set %>%
  left_join(movie_avgs, by = "movieId") %>%
  mutate(pred = mu + b_i) %>%
  .$pred

model_2_rmse <- RMSE(test_set$rating, predicted_ratings)
model_2_rmse
```

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

Model 3: Movie + User Effect Model

```
# Movie + User effect model
user_avgs <- train_set %>%
  left_join(movie_avgs, by = "movieId") %>%
  group_by(userId) %>%
  summarize(b_u = mean(rating - mu - b_i))

predicted_ratings <- test_set %>%
  left_join(movie_avgs, by = "movieId") %>%
  left_join(user_avgs, by = "userId") %>%
  mutate(pred = mu + b_i + b_u) %>%
  .$pred

model_3_rmse <- RMSE(test_set$rating, predicted_ratings)
model_3_rmse
```

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

Model 4: Regularized Movie + User Effect Model

```
# Regularized Movie + User effect model
lambda <- 5
b_i <- train_set %>%
  group_by(movieId) %>%
  summarize(b_i = sum(rating - mu) / (n() + lambda))

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

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

model_4_rmse <- RMSE(test_set$rating, predicted_ratings)
model_4_rmse
```

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

Results

Model Performance

```
# Display RMSE for all models
rmse_results <- tibble(
  Model = c("Naive Model", "Movie Effect Model", "Movie + User Effect Model", "Regularized Mo
vie + User Effect Model"),
  RMSE = c(naive_rmse, model_2_rmse, model_3_rmse, model_4_rmse)
)
rmse_results
```

```
## �[90m# A tibble: 4 × 2�[39m
##   Model                                  RMSE
##   �[3m�[90m<chr>�[39m�[23m                                  �[3m�[90m<dbl>�[39m�[23m
## �[90m1�[39m Naive Model                                1.06
## �[90m2�[39m Movie Effect Model                       �[31mNA�[39m
## �[90m3�[39m Movie + User Effect Model                �[31mNA�[39m
## �[90m4�[39m Regularized Movie + User Effect Model �[31mNA�[39m
```

Conclusion

In this project, we developed an algorithm to predict movie ratings using the MovieLens 10M dataset. We explored several models, including the Naive Model, Movie Effect Model, Movie + User Effect Model, and Regularized Movie + User Effect Model. The performance of each model was evaluated using RMSE, and the Regularized Movie + User Effect Model showed the best performance. Future work can focus on improving the model by incorporating additional features such as genres, timestamp, and using more advanced machine learning techniques like matrix factorization and collaborative filtering.

References

- Irizarry, R. A. (2020, March 2). Introduction to Data Science. Retrieved from https://rafalab.github.io/dsbook/introduction-to-machine-learning.html#notation-1 (https://rafalab.github.io/dsbook/introduction-to-machine-learning.html#notation-1)
- Harper, F. M., & Konstan, J. A. (2016). The MovieLens Datasets: History and Context. ACM Transactions on Interactive Intelligent Systems (TiiS), 5(4), 1–19. https://dl.acm.org/doi/10.1145/2827872 (https://dl.acm.org/doi/10.1145/2827872)
- Ricci, F., Rokach, L., Shapira, B., & Kantor, P. B. (2011). Recommender Systems Handbook. Springer. https://link.springer.com/book/10.1007/978-0-387-85820-3 (https://link.springer.com/book/10.1007/978-0-387-85820-3)
- Adomavicius, G., & Tuzhilin, A. (2005). Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions. IEEE Transactions on Knowledge and Data Engineering, 17(6), 734–749. https://ieeexplore.ieee.org/document/1423975 (https://ieeexplore.ieee.org/document/1423975)