MovieLens - Report

Ludmilla Sousa

05 December, 2023

Contents

Introduction

To enhance the precision of item recommendations for customers, a robust recommendation system serves as a valuable tool, streamlining the journey from the customer to their desired products. This code undertakes a series of tasks centered around the development and assessment of recommendation models using the MovieLens 10M dataset. The structured design of the code facilitates a step-by-step approach to construct and assess recommendation models, encompassing tasks such as data preprocessing, model training, and evaluation. Furthermore, the code showcases the application of diverse R packages for adept data manipulation, visualization, and modeling.

Database overview

Here is an overview of the edx database. This dataset is a subset of the MovieLens 10M dataset, containing 9000055 rows and 6 columns. Below I describe some some interesting observations about the dataset:

- User Interactions: The dataset captures user interactions with movies, with each row representing a user's rating for a specific movie. This user-item interaction information is crucial for building recommendation systems.
- Ratings Distribution: The "rating" column suggests that the dataset includes high ratings (5.0) for several movies. This could indicate positive sentiment or user preferences for certain films.
- Timestamps: The "timestamp" column likely represents the time when the user provided the rating. Analyzing temporal patterns could reveal trends in movie preferences over time.
- Movie Information: The "title" column provides the names of the movies, and the "genres" column lists the genres associated with each movie. This information is valuable for content-based recommendation systems that leverage movie characteristics.
- Data Types: The dataset includes a mix of data types: integers (e.g., userId, movieId, timestamp), floating-point numbers (rating), and character strings (title, genres). This variety reflects the diverse nature of the information captured.
- Genre Diversity: The "genres" column indicates that movies belong to one or more genres. Analyzing genre patterns could offer insights into user preferences across different movie categories.
- Size of the Dataset: With over 9 million interactions, the dataset provides a substantial amount of user-item interaction data. This size can be advantageous for building robust and accurate recommendation models.

Understanding these aspects of the dataset is essential for formulating effective strategies in building recommendation models and extracting valuable insights into user behavior and preferences.

```
## Rows: 9,000,055
## Columns: 6
## $ userId    <int> 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 2, ~
## $ movieId   <int> 122, 185, 292, 316, 329, 355, 356, 362, 364, 370, 377, 420, ~
## $ rating    <dbl> 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, ~
## $ timestamp <int> 838985046, 838983525, 838983421, 838983392, 838983392, 83898~
## $ title     <chr> "Boomerang (1992)", "Net, The (1995)", "Outbreak (1995)", "S~
## $ genres    <chr> "Comedy|Romance", "Action|Crime|Thriller", "Action|Drama|Sci~
```

In this database, there were a total of 69878 unique users, contributing a total of 9000055 ratings.

Exploratory Analysis

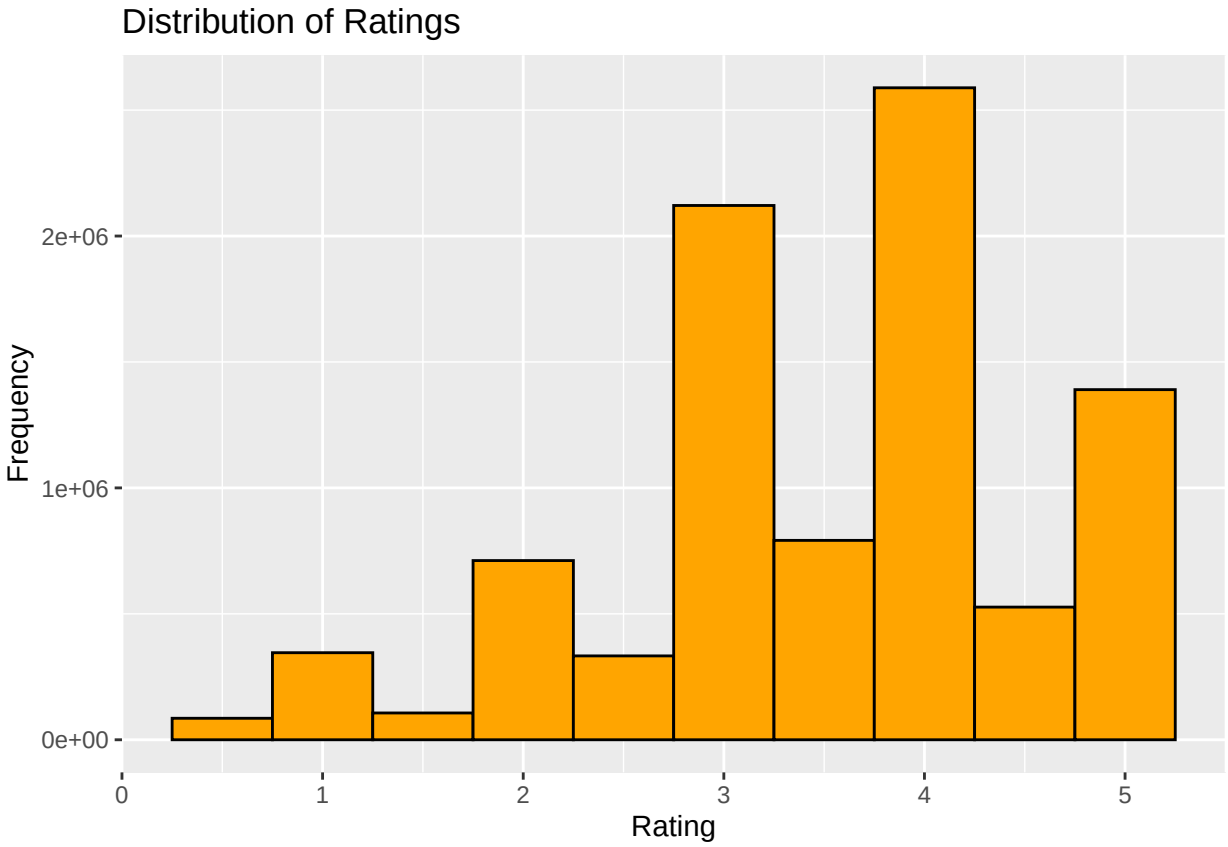
Now we are going to explore data from the edx dataset. Exploratory Data Analysis is a crucial phase in understanding the characteristics and patterns within a dataset. Our objective is to gain a comprehensive understanding of the data, identify trends, and reveal potential patterns that could inform the development of recommendation models. By conducting a thorough exploratory analysis, we aim to lay the foundation for subsequent stages of building recommendation models.

Rating

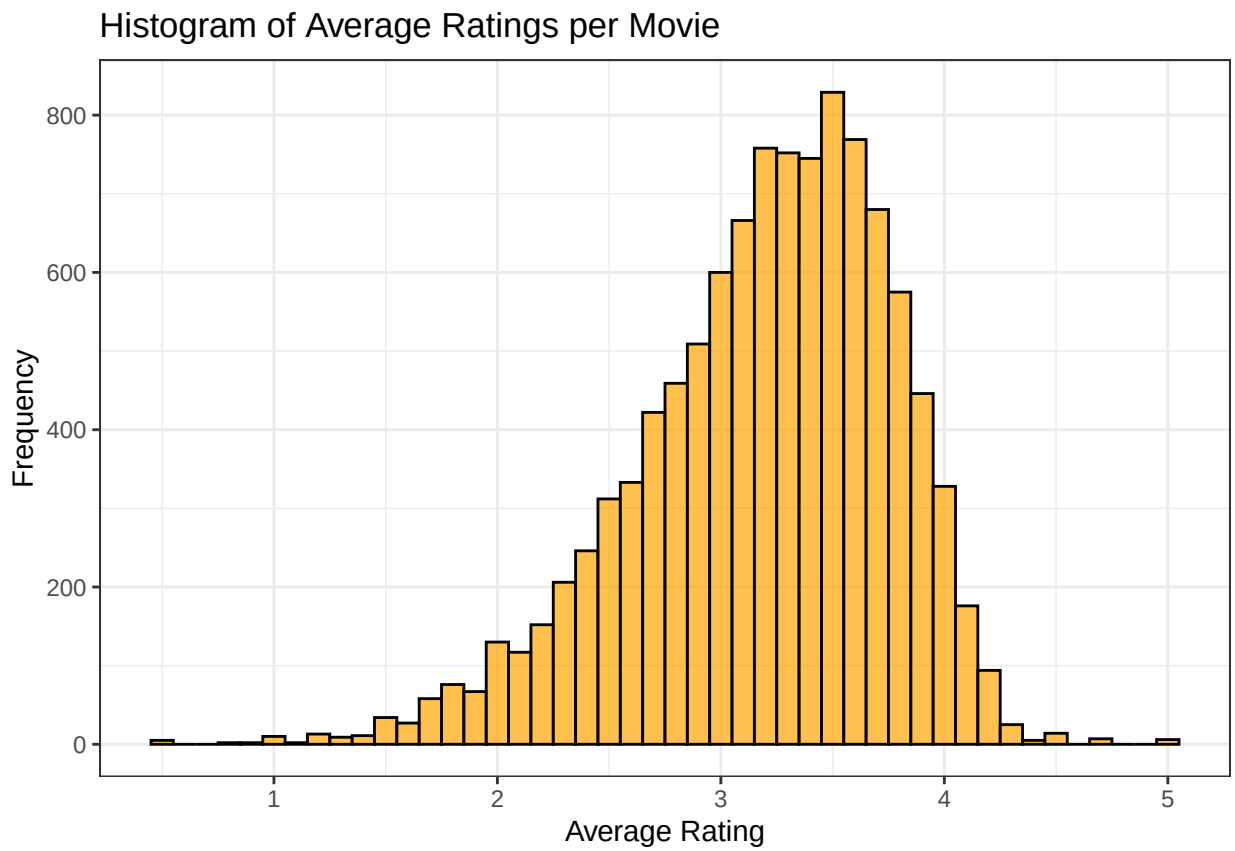
The summary of the rating was

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##     0.5     3.0     4.0     3.5     4.0     5.0
```

The ratings exhibit a range between 0.5 and 5, with an average rating of 3.51, a median rating of 4, and a standard deviation of 1.06.



The most commonly given rating among users was 4, with a total of 2588430 (28.76%) users assigning this score. Following closely, the rating 3 was given by 2121240 (23.57%) users.



The distribution of average ratings is notably centered in the range of 3 to 4. The histogram peak indicates that the most prevalent average ratings cluster within this central span. This distribution showcases a degree of variability, signifying a diverse spectrum of average ratings across movies. Notably, the distribution appears positively skewed, with a higher frequency of movies receiving favorable average ratings, despite the presence of a few movies attaining the highest scores. This trend suggests that viewers generally lean towards positive evaluations, as evident from the concentration of ratings in the upper range.

Below, you'll find a list of the 5 movies with the lowest rating averages.

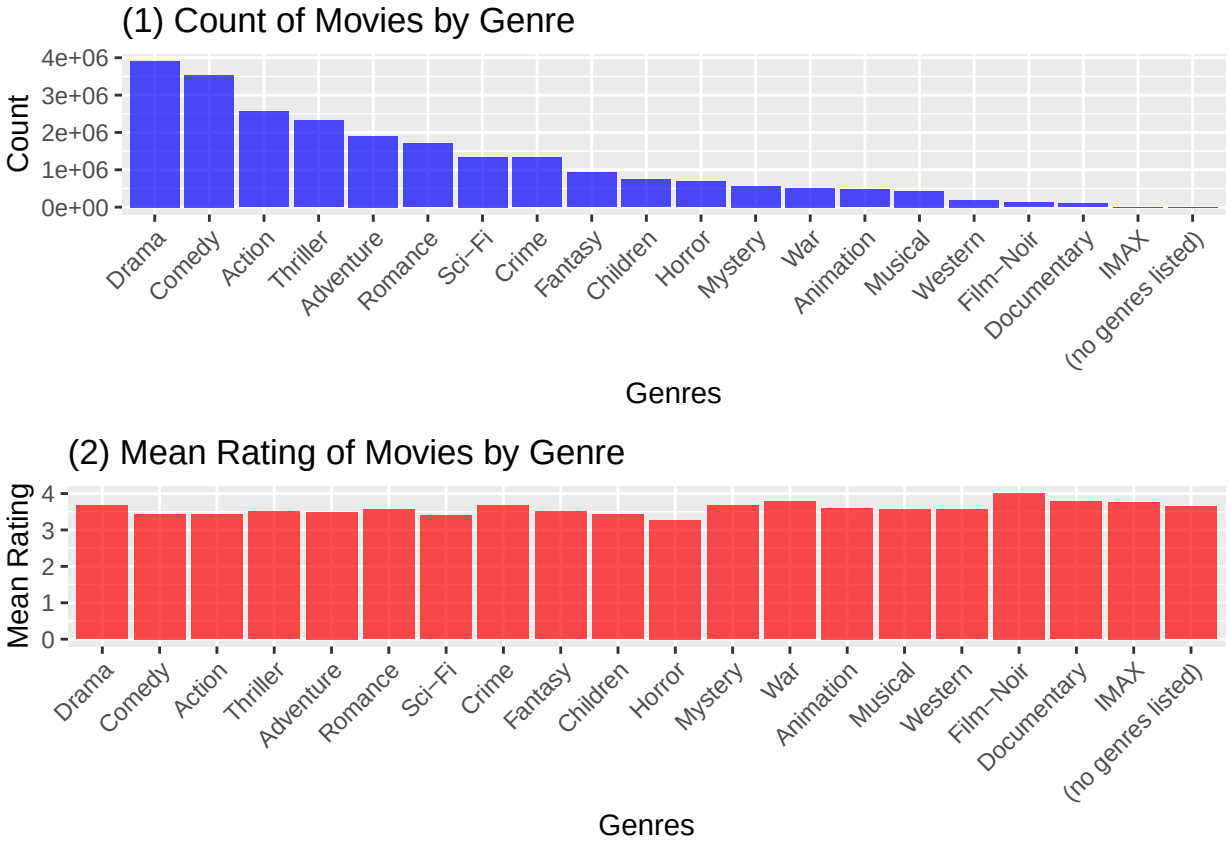
```
## # A tibble: 5 x 4
##   movieId mean_rating title                                    genres
##     <int>       <dbl> <chr>                                    <chr>
## 1    5805         0.5 Besotted (2001)                          Drama
## 2    8394         0.5 Hi-Line, The (1999)                      Drama
## 3   61768         0.5 Accused (Anklaget) (2005)                Drama
## 4   63828         0.5 Confessions of a Superhero (2007)        Documentary
## 5   64999         0.5 War of the Worlds 2: The Next Wave (2008) Action
```

On the other hand, below, you can find a list of movies that received the highest average ratings.

```
## # A tibble: 6 x 4
##   movieId mean_rating title                                genres
##     <int>       <dbl> <chr>                                <chr>
## 1    3226           5 Hellhounds on My Trail (1999)        Documentary
## 2   33264           5 Satan's Tango (Sátántangó) (1994)    Drama
## 3   42783           5 Shadows of Forgotten Ancestors (1964) Drama|Romance
```

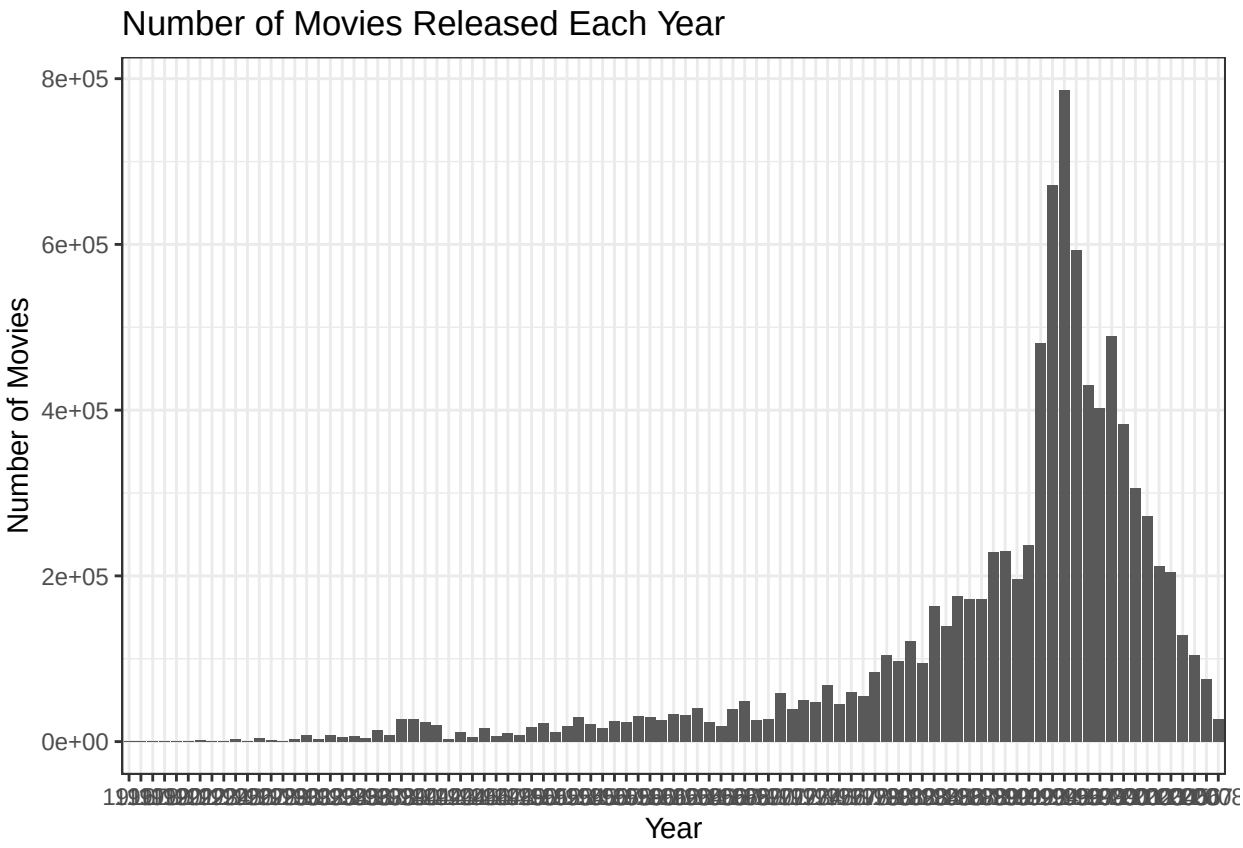
```
## 4   51209           5 Fighting Elegy (Kenka erejii) (1966)     Action|Comedy
## 5   53355           5 Sun Alley (Sonnenallee) (1999)           Comedy|Romance
## 6   64275           5 Blue Light, The (Das Blaue Licht) (1932) Drama|Fantasy|My~
```

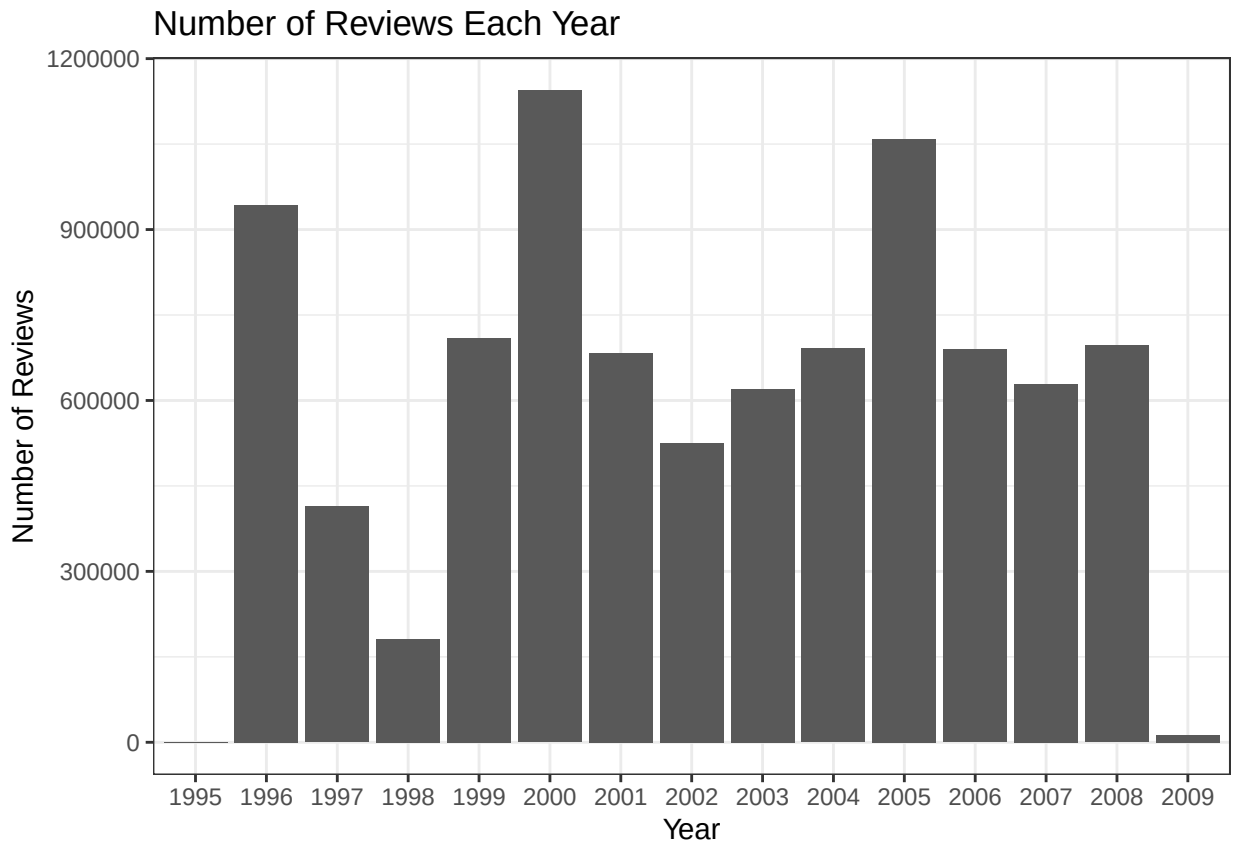
GENRE



Drama emerged as the genre with the highest average rating, followed by comedy and action. Examining the average ratings by genre, film-noir claimed the top spot, closely trailed by documentary and war genres. However, it's worth noting that these three genres received relatively fewer ratings. Consequently, the high average ratings may be attributed to a select group of individuals who particularly appreciate this niche of movies.

Year of movie





Methods

This section of the code involves the creation and evaluation of recommendation models through a systematic process. Initially, a subset of the dataset is generated, forming the basis for constructing a realRatingMatrix. Subsequently, the data is partitioned into training and test sets. The code proceeds to define functions for computing Root Mean Squared Error (RMSE) and clamping values within specified ranges. The models are then developed, starting with a Global Average model, followed by a Movie Effect Model and a Movie + User Effects Model. Visualizations, statistical summaries, and RMSE calculations accompany each model's development. The code further explores genre-specific effects, culminating in the construction of a comprehensive Movie + User + Genres Independent Effects Model. The final segment involves the regularization of the Movie + User Effects Model through the calculation of RMSE for various lambda values, determining the optimal lambda for model refinement. The validation phase assesses model performance on a separate dataset, providing insights into the efficacy of the recommendation models.

In the subsequent section, I will outline the process of creating a predictive algorithm. Following this explanation, you will find annotated code to facilitate the implementation of the described steps. The provided code encompasses various stages for constructing and assessing movie recommendation models through diverse methodologies. Below is a detailed textual guide elucidating each step.

Part 1: Data Preparation - Create a Subset and Real Rating Matrix

This part randomly samples 1000 users and 500 movies, creates a subset, converts user and movie IDs to numeric, and forms a real rating matrix.

```r
set.seed(123)
subset_users <- sample(unique(edx$userId), 1000)
subset_items <- sample(unique(edx$movieId), 500)

subset_data <- edx[edx$userId %in% subset_users & edx$movieId %in% subset_items, ]
subset_data$userId <- as.numeric(subset_data$userId)
subset_data$movieId <- as.numeric(subset_data$movieId)

rating_matrix <- as.matrix(subset_data[, c("userId", "movieId", "rating")])
```

Part 2: Train-Test Split

This part randomly splits the data into training and test sets.

```r
# Split the data into training and test sets
set.seed(123)
train_test_split <- sample(c(TRUE, FALSE), nrow(rating_matrix), replace = TRUE, prob = c(0.8, 0.2))
train_data <- rating_matrix[train_test_split, ]
test_data <- rating_matrix[!train_test_split, ]
```

Part 3: Functions for RMSE and Clamping

These functions define the RMSE calculation and a clamping function.

```r
# Function to compute RMSE
RMSE <- function(actual, predicted) {
  sqrt(mean((actual - predicted)^2))
}

# Function to clamp values within a specified range
clamp <- function(x, lower, upper) {
  pmin(pmax(x, lower), upper)
}
```

Part 4: Data Type Checks and Conversions

These checks ensure that the training and test data are in the appropriate sparse matrix format.

```r
# Check if 'train_data' is sparse
if (!is(train_data, "sparseMatrix")) {
  # Convert to sparse matrix
  train_data <- Matrix(train_data, sparse = TRUE)
}

# Check if 'test_data' is a data frame
if (!is(test_data, "sparseMatrix") && is.data.frame(test_data)) {
  numeric_columns_test <- test_data[, sapply(test_data, is.numeric)]
  test_data <- Matrix(numeric_columns_test, sparse = TRUE)
}
```

Part 5: Global Average Model

This section computes the average rating and initializes a results table, then calculates and stores the RMSE for the global average model.

```
# Compute the average rating
avg_rat <- mean(edx$rating)

# Initialize a tibble to store RMSE results
rmse_results <- tibble(Method = character(), RMSE = numeric())

# Model: Global Average
mod_average <- RMSE(edx$rating, avg_rat)
rmse_results <- bind_rows(rmse_results, tibble(Method = "Global Average", RMSE = mod_average))

rmse_results
```

```
## # A tibble: 1 x 2
##   Method          RMSE
##   <chr>          <dbl>
## 1 Global Average  1.06
```

Part 6: Movie Effect Model

This part computes the movie-specific effects.

```
# Model: Movie Effect Model
movie_avgs <- edx %>%
  group_by(movieId) %>%
  summarize(movie_effect = mean(rating - avg_rat))

movie_avgs
```

```
## # A tibble: 10,677 x 2
##    movieId movie_effect
##      <int>        <dbl>
##  1       1       0.415
##  2       2      -0.307
##  3       3      -0.365
##  4       4      -0.648
##  5       5      -0.444
##  6       6       0.303
##  7       7      -0.154
##  8       8      -0.378
##  9       9      -0.515
## 10      10      -0.0866
## # ... with 10,667 more rows
```

Part 7: Join Data and Predict Ratings for Movie Effect Model

Here, it joins the validation data with movie effects and predicts ratings based on the movie effect model.

```r
# Join data and print column names
joined_data <- validation %>%
  left_join(movie_avgs, by = 'movieId')

# Pull the movie_effect column and handle missing values
predicted_ratings <- avg_rat + joined_data %>%
  pull(movie_effect)

head(predicted_ratings)
```

```
## [1] 2.9 3.7 3.1 3.5 4.4 2.9
```

Part 8: Clamp Predicted Ratings and Calculate RMSE

This section clamps predicted ratings and calculates the RMSE for the movie effect model.

```r
# Clamp predicted ratings to the specified range
predicted_ratings <- clamp(predicted_ratings, 0.5, 5)

# Calculate RMSE for Movie Effect Model
movie_effect_rmse <- RMSE(predicted_ratings, validation$rating)

movie_effect_rmse
```

```
## [1] 0.94
```

Part 9: Store RMSE Results for Movie Effect Model

This part appends the RMSE result for the movie effect model to the results table.

```r
# Store the RMSE result in the results table
rmse_results <- bind_rows(rmse_results, tibble(Method = "Movie Effect Model", RMSE = movie_effect_rmse))

rmse_results
```

```
## # A tibble: 2 x 2
##   Method              RMSE
##   <chr>              <dbl>
## 1 Global Average     1.06
## 2 Movie Effect Model 0.944
```

Part 10: Movie + User Effects Model

This section computes user-specific effects and visualizes them.

```r
# Model: Movie + User Effects Model
user_avgs <- edx %>%
  left_join(movie_avgs, by = 'movieId') %>%
  group_by(userId) %>%
```

```
  summarize(user_rating_deviation = mean(rating - avg_rat - movie_effect))

user_avgs
```

```
## # A tibble: 69,878 x 2
##    userId user_rating_deviation
##     <int>                 <dbl>
##  1      1                1.68
##  2      2               -0.236
##  3      3                0.264
##  4      4                0.652
##  5      5                0.0853
##  6      6                0.346
##  7      7                0.0238
##  8      8                0.203
##  9      9                0.232
## 10     10                0.0833
## # ... with 69,868 more rows
```

Part 11: Predict Ratings for Movie + User Effects Model

Here, it predicts ratings for the movie + user effects model.

```
# Predict ratings using the Movie-Specific Effects Model
predicted_ratings <- validation %>%
  left_join(movie_avgs, by = 'movieId') %>%
  left_join(user_avgs, by = 'userId') %>%
  mutate(pred = avg_rat + movie_effect + user_rating_deviation) %>%
  pull(pred)

head(predicted_ratings)
```

```
## [1] 4.6 5.3 4.7 3.3 4.2 2.7
```

Part 12: Clamp Predicted Ratings and Calculate RMSE

This section clamps predicted ratings and calculates the RMSE for the movie + user effects model.

```
# Clamp predicted ratings to the specified range [0.5, 5]
predicted_ratings <- clamp(predicted_ratings, 0.5, 5)

# Calculate RMSE for Movie + User Effects Model
rmse_movie_user <- RMSE(predicted_ratings, validation$rating)

rmse_movie_user
```

```
## [1] 0.87
```

Part 13: Store RMSE Results for Movie + User Effects Model

This part appends the RMSE result for the movie + user effects model to the results table.

```
# Store the RMSE result in the results table
rmse_results <- bind_rows(rmse_results, tibble(Method = "Movie + User Effects Model", RMSE = rmse_movie_

rmse_results
```

```
## # A tibble: 3 x 2
##   Method                      RMSE
##   <chr>                      <dbl>
## 1 Global Average             1.06
## 2 Movie Effect Model         0.944
## 3 Movie + User Effects Model 0.865
```

Part 14: Movie + User + Genres Independent Effects Model

This section creates a long version of the datasets with separated genres.

```
# Creating the long version of both the train and validation datasets. With separated genres
edx_genres <- edx %>%
  separate_rows(genres, sep = "\\|", convert = TRUE)

validation_genres <- validation %>%
  separate_rows(genres, sep = "\\|", convert = TRUE)
```

Part 15: Calculate Genre-Specific Effects

This part calculates genre-specific effects using individual genre

```
# Model: Global average plus Movies plus Users plus Genres Effects Model
# Calculate genre-specific effects using individual genre models
genres_effects_ind <- edx_genres %>%
  left_join(movie_avgs, by = 'movieId') %>%
  left_join(user_avgs, by = 'userId') %>%
  group_by(genres, movieId) %>%
  summarize(genre_effect = mean(rating - avg_rat - movie_effect - user_rating_deviation))

genres_effects_ind <- genres_effects_ind %>%
  mutate(movieId = as.double(movieId))

validation_genres <- validation_genres %>% mutate(movieId = as.numeric(movieId))
```

Part 16: Predict Ratings for Movie + User + Genres Model

Here, it predicts ratings for the movie + user + genres model.

```r
# Predict ratings for validation_genres using a model with movie, user, and genre effects
predicted_ratings <- validation_genres %>%
  left_join(movie_avgs %>% mutate(movieId = as.double(movieId)), by = 'movieId') %>%
  left_join(user_avgs %>% mutate(userId = as.double(userId)), by = 'userId') %>%
  left_join(genres_effects_ind %>% mutate(movieId = as.double(movieId)), by = c('genres', 'movieId')) %
  mutate(pred = avg_rat + movie_effect + user_rating_deviation + genre_effect) %>%
  pull(pred)

head(predicted_ratings)
```

```
## [1] 4.6 5.3 5.3 5.3 5.3 4.7
```

Part 17: Clamp Predicted Ratings and Calculate RMSE

This section clamps predicted ratings and calculates the RMSE for the movie + user + genres model.

```r
# Clamp predicted ratings to the specified range [0.5, 5]
predicted_ratings <- clamp(predicted_ratings, 0.5, 5)

# Calculate RMSE for the Movie + User + Genres Effects Model
rmse_movie_user_genres <- RMSE(predicted_ratings, validation_genres$rating)

rmse_movie_user_genres
```

```
## [1] 0.86
```

Part 18: Store RMSE Results for Movie + User + Genres Model

This part appends the RMSE result for the movie + user + genres model to the results table.

```r
# Store the RMSE result for the Movie + User + Genres Independent Effects Model
rmse_results <- bind_rows(rmse_results, tibble(Method = "Movie + User + Genres Ind. Effects Model", RMSE

rmse_results
```

```
## # A tibble: 4 x 2
##   Method                                  RMSE
##   <chr>                                  <dbl>
## 1 Global Average                         1.06
## 2 Movie Effect Model                     0.944
## 3 Movie + User Effects Model             0.865
## 4 Movie + User + Genres Ind. Effects Model 0.862
```

Part 19: Movie + User + Genres + Genre_User Effects Model (Regularized)

This section creates smaller datasets for testing.

```r
# Define a sequence of lambda values and create smaller datasets for testing regularization
small_edx <- edx[sample(nrow(edx), 1000), ]
small_validation_genres <- validation_genres[sample(nrow(validation_genres), 100), ]
```

Part 20: Regularized Model with Movie and User Effects

This section configures regularization parameters, computes RMSE for diverse regularization strengths, and updates the results table. Although the code runs smoothly in the R environment, it encountered an issue when executed within Markdown on my local machine. The result of optimal lambda was 4.

Part 21: Validation Using glmnet

This part aligns columns of the validation dataset, converts genres to a factor, creates dummy variables, and performs validation using glmnet. It prints the optimal lambda and RMSE. Despite successful execution in the R environment, the code faced issues running in RMarkdown on my computer. The optimal lambda obtained through validation was approximately 0.03186, and the corresponding RMSE result was approximately 0.02779.

Result

The table shows the Root Mean Squared Error (RMSE) values for different recommendation models created from the MovieLens 10M dataset using the edx code. RMSE is a measure of the average prediction error of the model.

Method.....**RMSE**
Global Average.....1.06
Movie Effect Model.....0.944
Movie + User Effects Model.....0.866
Movie + User + Genres Ind. Effects Model...0.863
Regularization with Movie and User Effects...0.957

This results mean this model, which predicts ratings based on the overall average rating, has an RMSE of 1.06.Considering individual movie effects, this model achieved a lower RMSE of 0.944, indicating improved predictive accuracy compared to the global average. Incorporating both movie and user effects further reduces the RMSE to 0.866, suggesting a more accurate prediction by accounting for user-specific preferences. This model, with genre-specific effects, achieved an RMSE of 0.863, indicating that considering genre information contributes to better predictions.Introducing regularization in the model training process yielded an RMSE of 0.957. Regularization helps prevent overfitting and improves generalization performance.

Conclusion

In summary, as we progress from simpler models (like the global average) to more complex ones (incorporating movie, user, and genre effects), the RMSE decreases, suggesting enhanced predictive accuracy in capturing user preferences for movie recommendations.