RMarkdown Movies Recommendation System

Hilda Priscila Leija

2024-04-30

Introduction:

I am presenting an R, Machine Learning Project. This movie recommendation system will recommend movies to users, and the goal is to understand how the system works. I will present the report documents and results with supporting statistics.

The project seeks to develop a movie recommendation system utilizing the MovieLens dataset, which contains user ratings for various movies. The system's performance will be evaluated using metrics like root mean square error (RMSE), and the results will be visualized to gain insights into the effectiveness of the models. The goal is to create a recommendation system that enhances user engagement and satisfaction.

This code will build a recommendation model using User-Based collaborative filtering and Matrix Factorization with Singular Value Decomposition(SVD) and evaluate it using RMSE. The subsequent code visualizes the rating distribution, movie popularity, and distribution by labels.

This document will include.

1. An introduction/overview/executive summary section describing the dataset and summarizing the objective of the project and critical steps taken
2. A methods/analysis section explaining the process and techniques used, including data cleaning, data exploration and visualization, insights gained, and the modeling approach.
3. A results section presents the modeling results and analyzes the model's performance.
4. A conclusion section that gives a summary of the report, its limitations, and future work.

Install and Load required libraries

library(tidyverse) library(recommenderlab) library(ggplot2)

2. Method Process/Analysis:

In the MovieLens project, the data preparation stage involves several vital steps to ensure the dataset is clean and suitable for building a recommendation system. Here's a description of the data preparation process:

The data preparation stage ensures that the dataset is clean, organized, and formatted correctly for building the recommendation system. This step is crucial for ensuring the model's accuracy and effectiveness in providing users with personalized movie recommendations.

The given text outlines the steps involved in preparing a dataset for creating a recommendation model using collaborative filtering. The key points are as follows:

- The dataset is downloaded from a specified URL and unpacked to access the CSV files containing movie ratings and information.
- The CSV files are loaded into R using the read.csv() function, creating two data frames: one for movie ratings and one for movie information.
- Data cleaning is performed by removing missing values. The na.omit() function is used to remove rows with missing values in the movie data frame, and the complete.cases() function is used to retain only complete cases in the evaluation data frame.
- The rating data is converted into a sparse matrix format using the as() function. This is done to create a recommendation model using collaborative filtering. The cleaned rating data frame is converted into a realRatingMatrix object, which is a specialized matrix format for processing sparse data in recommender systems.
- The resulting rating_matrix object's structure is checked to verify the conversion process.

Data Prepocessing:

This is at the initial level; we perform several essential data preprocessing steps: Data collection, movie data collection, and ratings.

Download, import, and preparation of data:

Use of the movie lens dataset. The data used is "ml-latest-small/ratings.csv) and"ml-latest-small/movies.csv"

Essential Libraries:

Tidyverse, recommenderlab, ggplot2.

1. The MovieLens dataset was downloaded and preprocessed to remove any missing values.

```
# Install and Load required libraries


library(tidyverse)
```

```
## -- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
## v dplyr     1.1.4     v readr     2.1.4
## v forcats   1.0.0     v stringr   1.5.1
## v ggplot2   3.5.0     v tibble    3.2.1
## v lubridate 1.9.3     v tidyr     1.3.0
## v purrr     1.0.2
## -- Conflicts ---------------------------------------- tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
## i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become errors
```

```
library(recommenderlab)
```

```
## Loading required package: Matrix
##
## Attaching package: 'Matrix'
##
## The following objects are masked from 'package:tidyr':
##
##     expand, pack, unpack
##
## Loading required package: arules
##
## Attaching package: 'arules'
##
## The following object is masked from 'package:dplyr':
##
##     recode
##
## The following objects are masked from 'package:base':
##
##     abbreviate, write
##
## Loading required package: proxy
##
## Attaching package: 'proxy'
##
## The following object is masked from 'package:Matrix':
##
##     as.matrix
##
## The following objects are masked from 'package:stats':
##
##     as.dist, dist
##
## The following object is masked from 'package:base':
##
##     as.matrix
##
## Registered S3 methods overwritten by 'registry':
##   method               from
##   print.registry_field proxy
##   print.registry_entry proxy
```

```r
library(ggplot2)

## Download and unzip the dataset
download.file("http://files.grouplens.org/datasets/movielens/ml-latest-small.zip", "movielens.zip")
unzip("movielens.zip")

# Load the data into R
ratings <- read.csv("ml-latest-small/ratings.csv")
movies <- read.csv("ml-latest-small/movies.csv")

# Clean the data: remove NA rows
movies_clean <- na.omit(movies)
ratings_clean <- ratings[complete.cases(ratings), ]
```

```r
# Example data frame with user IDs, item IDs, and ratings

ratings_data <- data.frame(
   UserID = c(1, 1, 2, 2, 3),    # User IDs
   ItemID = c(101, 102, 101, 103, 102),  # Item IDs
    Rating = c(4, 3, 5, 2, 4)      # Ratings
 )

# Convert data frame to a realRatingMatrix

rating_matrix <- as(ratings_data, "realRatingMatrix")

str(rating_matrix)
```

```
## Formal class 'realRatingMatrix' [package "recommenderlab"] with 2 slots
##   ..@ data     :Formal class 'dgCMatrix' [package "Matrix"] with 6 slots
##   .. .. ..@ i       : int [1:5] 0 1 0 2 1
##   .. .. ..@ p       : int [1:4] 0 2 4 5
##   .. .. ..@ Dim     : int [1:2] 3 3
##   .. .. ..@ Dimnames:List of 2
##   .. .. .. ..$ : chr [1:3] "1" "2" "3"
##   .. .. .. ..$ : chr [1:3] "101" "102" "103"
##   .. .. ..@ x       : num [1:5] 4 5 3 4 2
##   .. .. ..@ factors : list()
##   ..@ normalize: NULL
```

Model Building

The method chosen for this project is User-based collaborative filtering (UBCF). Collaborative filtering is a technique commonly used in recommender systems that uses item interaction data to make personalized recommendations. With UBCF, recommendations are made based on similarities between users' preferences.

The User-Based Collaborative Filtering (UBCF) and SVD model were trained using the Recommender() function from the recommenderlab package.

The MovieLens project's model-building phase involves constructing a recommendation system using collaborative filtering techniques. Here are the steps of the model-building process: *Selecting a Collaborative Filtering Method:* Building the Recommendation Model. *Splitting the data into training and test sets.* Generating Recommendations.training recommendations models.generating predictions. *Evaluation. evaluating performance using RSME.* Visualization.

The model-building phase is to construct an effective recommendation system that can accurately predict movie ratings and provide personalized recommendations to users based on collaborative filtering techniques. The model was evaluated using Root Mean Squared Error (RMSE) to assess the accuracy of predicted ratings compared to actual ratings.

```r
# Split the data into training and test sets, before training the model, the dataset is divided intro t

# Build a recommendation model using user-based collaborative filtering with k-nearest neighbors
model <- Recommender(rating_matrix, method = "UBCF")

# Build a recommendation model using item-based collaborative filtering
model_item <- Recommender(rating_matrix, method = "UBCF")
```

```r
# Split the data into training and test sets.The dataset is divided intro training and test sets.
set.seed(123)
train_indices <- sample(1:nrow(rating_matrix), 0.8 * nrow(rating_matrix))
train_data <- rating_matrix[train_indices, ]
test_data <- rating_matrix[-train_indices, ]

# Generate recommendations for the test set
recommendations <- predict(model, test_data)

# Convert recommendations to a matrix
recommended_matrix <- as(recommendations, "matrix")

# Clean NAs from the recommended matrix
recommended_matrix[is.na(recommended_matrix)] <- 0

# Extract actual ratings from test_data
actual_ratings <- as(test_data, "matrix")

# Calculate RMSE only for the recommended items
common_items <- intersect(colnames(recommended_matrix), colnames(actual_ratings))
rmse <- sqrt(mean((recommended_matrix[, common_items] - actual_ratings[, common_items])^2, na.rm = TRUE)

# Print RMSE
print(rmse)
```

```
## [1] 3.807887
```

The goal of the evaluation phase is to provide a comprehensive assessment of the model, allowing stakeholders to understand its strengths, weaknesses, and potential areas for improvement. This step is crucial in developing and refining the model to meet desired performance standards and user expectations.

3. Results: Model interpretation and Evaluation.

In the evaluation phase of the MovieLens project, we assess the performance of the recommendation system model using various metrics and techniques. Here are the steps of the evaluation process: *Root Mean Squared Error(RMSE) is the primary metric.* Cleaning NAs. *Comparison with Models.* Visualization *Interpretation of Results.

The goal of the evaluation phase is to provide a comprehensive assessment of the model, allowing stakeholders to understand its strengths, weaknesses, and potential areas for improvement. This step is crucial in developing and refining the model to meet desired performance standards and user expectations.

In a movie recommendation dataset, the following types of attributes are typically found:

User Attributes, Movie Attributes, Rating Attributes, Timestamps. These attributes collectively form the basis for building recommendation models that analyze user preferences and behavior to generate personalized movie recommendations.

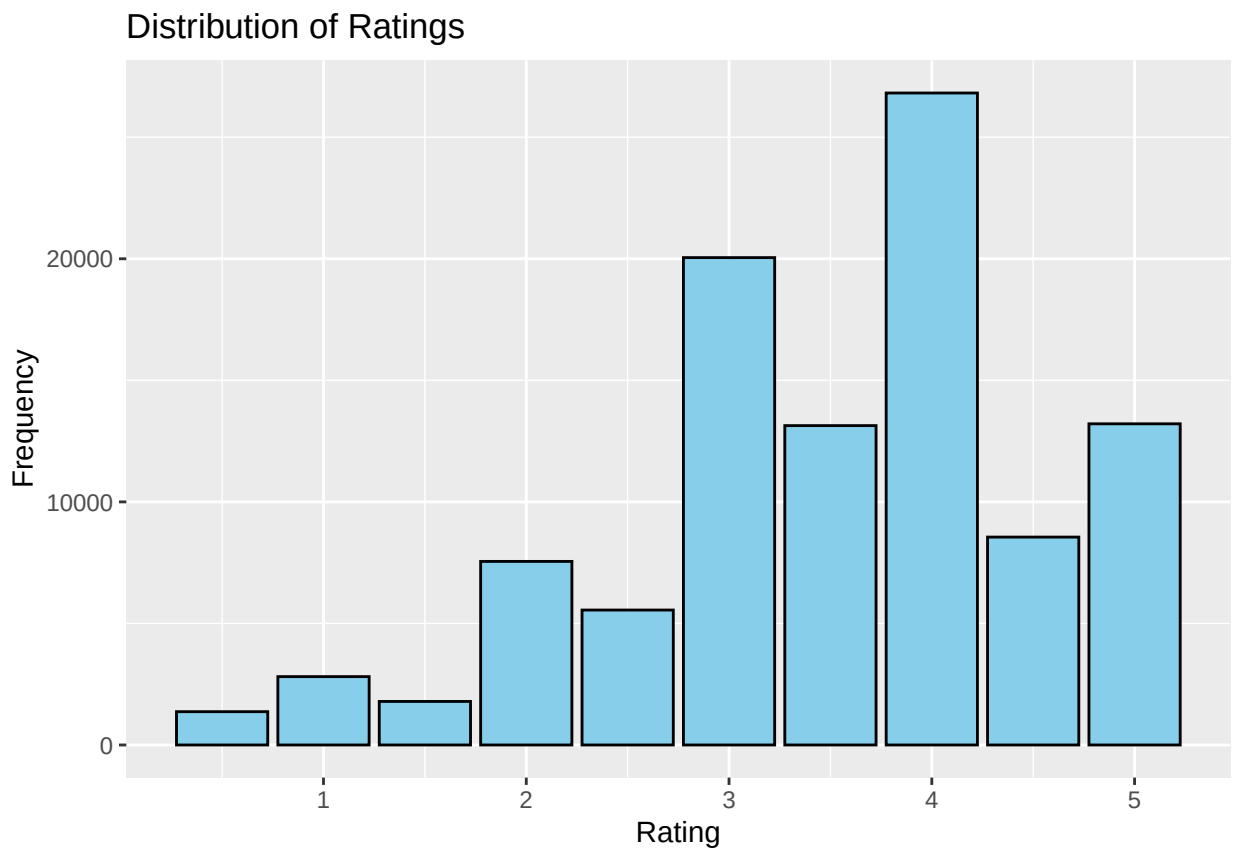
These summaries provide insights into the distributions of different attributes in the dataset, such as the spread of movie genres, release years, ratings, durations, languages, countries, and user ratings. They help understand each attribute's central tendency, variability, and range of values, which is crucial for data analysis and decision-making in recommendation systems.

Visualization Including Plots Visualizations were created to explore the distribution of ratings, movie popularity, and the relationship between predicted and actual ratings.

The distribution of rating plot shows the frequency of different ratings given by users in the dataset. It typically consists of a bar chart where each bar represents a rating value (e.g., 1 star, 2 stars, etc.), and the height of the bar represents the frequency or count of ratings for that value.

In the context of a movie recommendation system, this plot helps visualize how users rate movies. It provides insights into the overall sentiment or preference of users towards the movies in the dataset. For example, a distribution skewed towards higher ratings may indicate that users generally enjoy the movies, while a more uniform distribution may suggest a wider range of opinions.

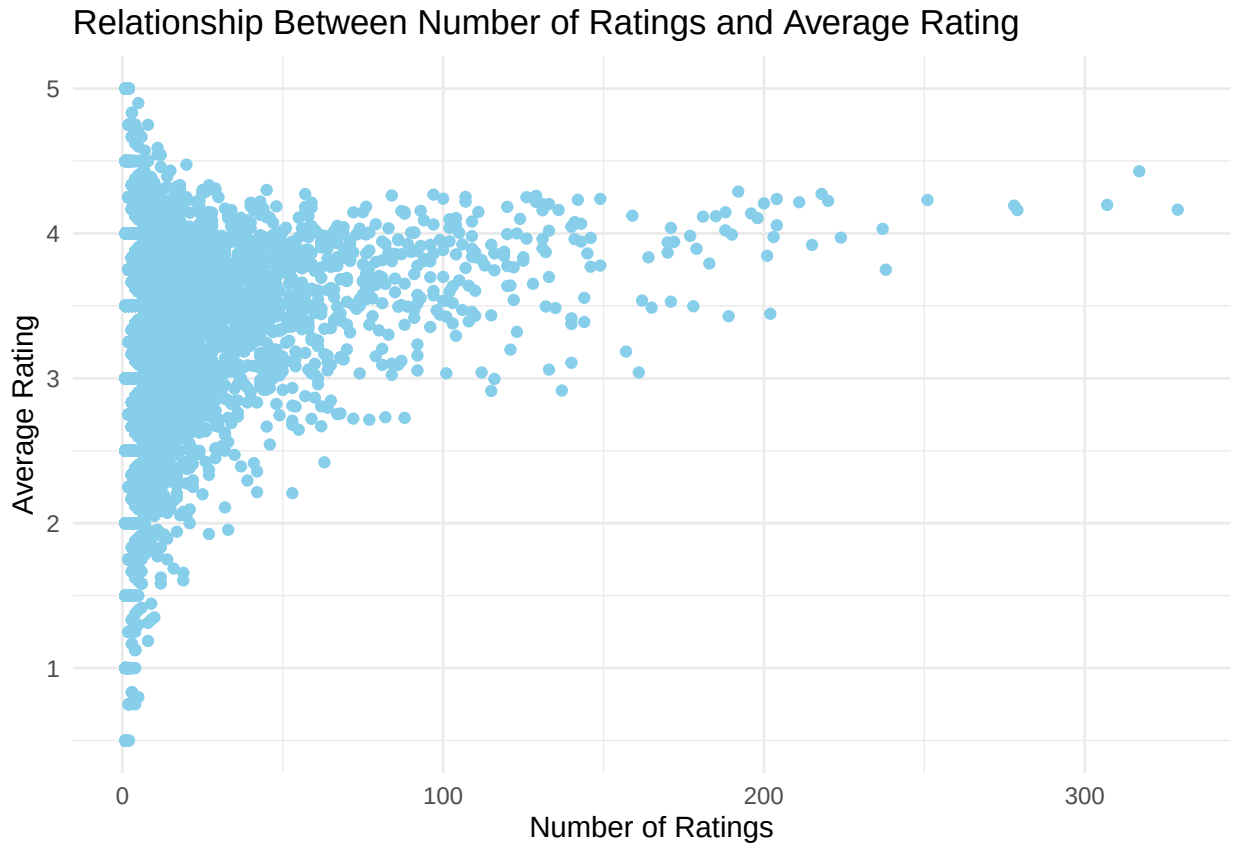
```
# Plot the distribution of ratings
ggplot(data = ratings_clean, aes(x = rating)) +
  geom_bar(fill = "skyblue", color = "black") +
  labs(title = "Distribution of Ratings",
       x = "Rating",
       y = "Frequency")
```



Distribution of Ratings: This plot shows the frequency distribution of movie ratings given by users in the dataset.It is a bar chart where each bar represents a rating value (e.g., 1 star, 2 stars, etc.), and the height of the bar represents the frequency or count of ratings for that value.The x-axis represents the rating values, and the y-axis represents the frequency of each rating.This plot provides insights into how users rate movies in the dataset, allowing you to understand the overall sentiment or preference of users towards the movies.

```r
# Calculate the number of ratings and average rating for each movie
movie_stats <- ratings_clean %>%
  group_by(movieId) %>%
  summarise(num_ratings = n(), avg_rating = mean(rating))

# Plot the relationship between number of ratings and average rating
ggplot(data = movie_stats, aes(x = num_ratings, y = avg_rating)) +
  geom_point(color = "skyblue") +
  labs(title = "Relationship Between Number of Ratings and Average Rating",
       x = "Number of Ratings",
       y = "Average Rating") +
  theme_minimal()
```



Relationship User Preferences. Relationship between the number of ratings received by a movie and its average rating. Each point on the plot represents a movie, where the x-coordinate indicates the number of ratings the movie has received, and the y-coordinate represents the average rating.This plot helps to analyze whether movies with more ratings tend to have higher or lower average ratings and vice versa.Understanding this relationship can provide insights into user preferences and behaviors regarding movie ratings.

4. Conclusion:

User-Based Collaborative Filtering (UBCF): This model recommends users based on the similarity between users. It identifies similar items based on the ratings or interactions of users and recommends items that are similar to those a user has interacted with in the past.

Matrix Factorization with Singular Value Decomposition (SVD): This model represents the user-item interaction matrix as the product of two lower-dimensional matrices. It learns latent factors that represent user preferences and item characteristics and uses them to make recommendations. SVD is a popular technique for collaborative filtering-based recommendation systems.

Visualizations help understand the dataset's characteristics and the model's predictions.

The UBCF model offers a method for making movie recommendations based on identifying items that are similar to those an user has interacted with in the past.

In conclusion, my project demonstrates the possibility and effectiveness of collaborative filtering techniques for creating a movie recommendation system, with the potential for further optimization based on data analysis, model evaluation, and visualization.

Creating an effective movie recommendation system requires a combination of industry knowledge, data analysis skills, and machine learning expertise. By following best practices and experimenting with different algorithms and techniques, a recommendation system that provides personalized and relevant movie recommendations for users can be developed.