Final project: Anime Recommendation System with Rating Prediction

Yagna sri madepally

2024-04-30

INTRODUCTION:

In this project, I aim to develop an anime recommendation system with rating prediction using two datasets: Anime and Rating.

Source: https://www.kaggle.com/datasets/CooperUnion/anime-recommendations-database

Anime.csv: This dataset contains 7 columns and 12,294 rows. Each row represents an anime entry with the following attributes:

- anime_id: Unique identifier for each anime on myanimelist.net. • name: Full name of the anime.
- genre: Comma-separated list of genres associated with the anime.
- type: Type of anime (e.g., movie, TV series, OVA, music, special, etc.).
- episodes: Number of episodes (1 if it's a movie).
- rating: Average rating of the anime, ranging from 1 to 10.
- members: Number of community members associated with the anime.

Rating.csv: This dataset consists of 3 columns and 46,986 rows. It represents user ratings for various anime, including:

- user_id: Randomly generated unique user identifier.
- anime_id: The ID of the anime that the user has rated.
- rating: The user's rating for the anime on a scale of 1 to 10.(-1 if watched but not rated)

The objective of this project is to build a recommendation system that can suggest anime titles to users based on their preferences.

Example of inspiration: Netflix Recommendation System

A similar recommendation system can be found on platforms like Netflix, where users are provided with personalized suggestions based on their viewing history and preferences. For instance, Netflix's "Users also watched" feature recommends titles that are similar to

the ones users have previously watched or rated positively. This enhances the user experience by offering relevant content tailored to individual tastes.

Step 1: Data Preparation/collection

Loading the required packages

```
library(tidyverse)

library(e1071) #Using the e1071 package to calculate the skewness

library(treemap) #for creating a treemap

library(rsample)   # for resampling procedures

library(caret)  # for fitting KNN models

library(recipes)    # for feature engineering

library(tidymodels) #for tidy and unified modeling workflow

library(reshape2) #for reshaping and restructuring data

library(Matrix) #Matrix library for sparse and dense matrix classes and
methods

library(stringr) #for string manipulation functions

library(class) #for various classification methods including k-Nearest
Neighbors (kNN)

library(knitr)

library(DT)
```

Reading the data

```
library(readr)
Anime<- read.csv('C:/Users/yagna/Desktop/MSIS/581/FINAL_PROJECT/anime.csv')
Rating<-read.csv('C:/Users/yagna/Desktop/MSIS/581/FINAL_PROJECT/ratings.csv')
```

Exploring the Anime and Rating datasets

```
dim(Anime)

## [1] 12294     7

dim(Rating)

## [1] 46986     3
```

Glimpse of anime dataset

```
glimpse(Anime)
```

```
## Rows: 12,294
## Columns: 7
## $ anime_id <int> 32281, 5114, 28977, 9253, 9969, 32935, 11061, 820, 15335,
154…
## $ name     <chr> "Kimi no Na wa.", "Fullmetal Alchemist: Brotherhood",
"Gintam…
## $ genre    <chr> "Drama, Romance, School, Supernatural", "Action,
Adventure, D…
## $ type     <chr> "Movie", "TV", "TV", "TV", "TV", "TV", "TV", "OVA",
"Movie", …
## $ episodes <chr> "1", "64", "51", "24", "51", "10", "148", "110", "1",
"13", "…
## $ rating   <dbl> 9.37, 9.26, 9.25, 9.17, 9.16, 9.15, 9.13, 9.11, 9.10,
9.11, 9…
## $ members  <int> 200630, 793665, 114262, 673572, 151266, 93351, 425855,
80679,…
```

Glimpse of rating dataset

```
glimpse(Rating)
```

```
## Rows: 46,986
## Columns: 3
## $ user_id  <int> 1, 1, 1, 1, 1, 1, 1, 1, 1, 2, 3, 3, 3, 3, 3, 3, 3, 3, 3,
3, 3…
## $ anime_id <int> 20, 24, 79, 226, 241, 355, 356, 442, 487, 16417, 20, 154,
170…
## $ rating   <int> -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, 8, 6, 9, 10, 9,
6, 7,…
```

summary of anime and rating data sets

```
summary(Anime)
```

```
##     anime_id         name               genre               type
##  Min.   :    1   Length:12294       Length:12294       Length:12294
##  1st Qu.: 3484   Class :character   Class :character   Class :character
##  Median :10260   Mode  :character   Mode  :character   Mode  :character
##  Mean   :14058
##  3rd Qu.:24795
##  Max.   :34527
##
##    episodes             rating          members
##  Length:12294       Min.   : 1.670   Min.   :      5
##  Class :character   1st Qu.: 5.880   1st Qu.:    225
##  Mode  :character   Median : 6.570   Median :   1550
##                     Mean   : 6.474   Mean   :  18071
##                     3rd Qu.: 7.180   3rd Qu.:   9437
##                     Max.   :10.000   Max.   :1013917
##                     NA's   :230
```

```
summary(Rating)
```

```
##     user_id         anime_id         rating      
##  Min.   :  1.0   Min.   :    1   Min.   :-1.000  
##  1st Qu.:155.0   1st Qu.: 2581   1st Qu.: 5.000  
##  Median :282.0   Median : 9736   Median : 7.000  
##  Mean   :270.1   Mean   :10922   Mean   : 5.904  
##  3rd Qu.:392.0   3rd Qu.:16782   3rd Qu.: 9.000  
##  Max.   :500.0   Max.   :34240   Max.   :10.000
```

STEP 2: DATA CLEANING

Let's remove the duplicate values in Anime dataset and compare the no of rows:

```
Anime_dup=unique(Anime)
dim(Anime_dup)
```

```
## [1] 12294     7
```

we can see that the no of rows are: 12,294 which means we have zero duplicate values

Let's check for duplicate values in Rating dataset:

```
Rating_dup=unique(Rating)
dim(Rating_dup)
```

```
## [1] 46986     3
```

It has zero duplicate entries among 46,986 entries.

Identifying the null values in Anime and Rating datasets:

```
Anime_null_values <-colSums(is.na(Anime_dup))
Rating_null_values <-colSums(is.na(Rating_dup))
```

Transposed view of Anime dataset null values:

```
transposed_data <- t(Anime_null_values)

kable(transposed_data)
```

anime_id	name	genre	type	episodes	rating	members
0	0	0	0	0	230	0

Transposed view of Rating dataset null values:

```
transposed_data <-t(Rating_null_values)

kable(transposed_data)
```

user_id	anime_id	rating
0	0	0

Ratings dataset has zero null values.

The number of null values in Anime_dup dataset is relatively small compared to the overall dataset size and dropping them doesn't significantly affect the analysis

Dropping the null values in Anime:

```
Anime_df <- Anime_dup[complete.cases(Anime_dup), ]
```

Lets check for null values again:

```
Anime_null_values <-colSums(is.na(Anime_df))

transposed_data<-t(Anime_null_values)


kable(transposed_data)
```

anime_id	name	genre	type	episodes	rating	members
0	0	0	0	0	0	0

The null values are zero in Anime_df dataset

Now let us merge the datasets:

```
merged_data <- merge(Anime_df, Rating_dup, by = "anime_id")

glimpse(merged_data)
```

```
## Rows: 46,986
## Columns: 9
## $ anime_id <int> 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1, 1…
## $ name     <chr> "Cowboy Bebop", "Cowboy Bebop", "Cowboy Bebop", "Cowboy
Bebop…
## $ genre    <chr> "Action, Adventure, Comedy, Drama, Sci-Fi, Space",
"Action, A…
## $ type     <chr> "TV", "TV", "TV", "TV", "TV", "TV", "TV", "TV", "TV",
"TV", "…
## $ episodes <chr> "26", "26", "26", "26", "26", "26", "26", "26", "26",
"26", "…
## $ rating.x <dbl> 8.82, 8.82, 8.82, 8.82, 8.82, 8.82, 8.82, 8.82, 8.82,
8.82, 8…
## $ members  <int> 486824, 486824, 486824, 486824, 486824, 486824, 486824,
48682…
## $ user_id  <int> 384, 363, 460, 259, 226, 173, 296, 21, 81, 189, 446, 292,
221…
## $ rating.y <int> 8, 10, 8, -1, 8, 7, 8, 9, 8, 7, 9, 10, -1, 10, 7, 8, 8,
10, 8…
```

Renaming the columns in merged_data:

```
ColNames <- c("Anime_ID", "Name", "Genre", "Type", "Episodes", "Rating",
"Members", "User_ID", "User_Rating")

colnames(merged_data) <- ColNames

glimpse(merged_data)
```

```
## Rows: 46,986
## Columns: 9
## $ Anime_ID    <int> 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1, 1…
## $ Name        <chr> "Cowboy Bebop", "Cowboy Bebop", "Cowboy Bebop",
"Cowboy Be…
## $ Genre       <chr> "Action, Adventure, Comedy, Drama, Sci-Fi, Space",
"Action…
## $ Type        <chr> "TV", "TV", "TV", "TV", "TV", "TV", "TV", "TV", "TV",
"TV"…
## $ Episodes    <chr> "26", "26", "26", "26", "26", "26", "26", "26", "26",
"26"…
## $ Rating      <dbl> 8.82, 8.82, 8.82, 8.82, 8.82, 8.82, 8.82, 8.82, 8.82,
8.82…
## $ Members     <int> 486824, 486824, 486824, 486824, 486824, 486824,
486824, 48…
## $ User_ID     <int> 384, 363, 460, 259, 226, 173, 296, 21, 81, 189, 446,
292, …
## $ User_Rating <int> 8, 10, 8, -1, 8, 7, 8, 9, 8, 7, 9, 10, -1, 10, 7, 8,
8, 10…
```

We have many special characters in the Anime Name. ex: Wolf`&quot` s Rain Lets remove the special characters from the Name

```
# Defining patterns to remove
patterns <- c("&quot;", "\\.hack//", "&#039;", "A's", "I's", "&amp;")

# Iterate over patterns and remove them from the Name column
for (pattern in patterns) {
  merged_data$Name <- gsub(pattern, "", merged_data$Name)
}
```

Capitalize every starting letter in the Name column

```
merged_data$Name <- str_to_title(merged_data$Name)

glimpse(merged_data$Name)
```

```
##  chr [1:46986] "Cowboy Bebop" "Cowboy Bebop" "Cowboy Bebop" "Cowboy Bebop"
...
```

STEP 3: Analysis

1. Anime Categories

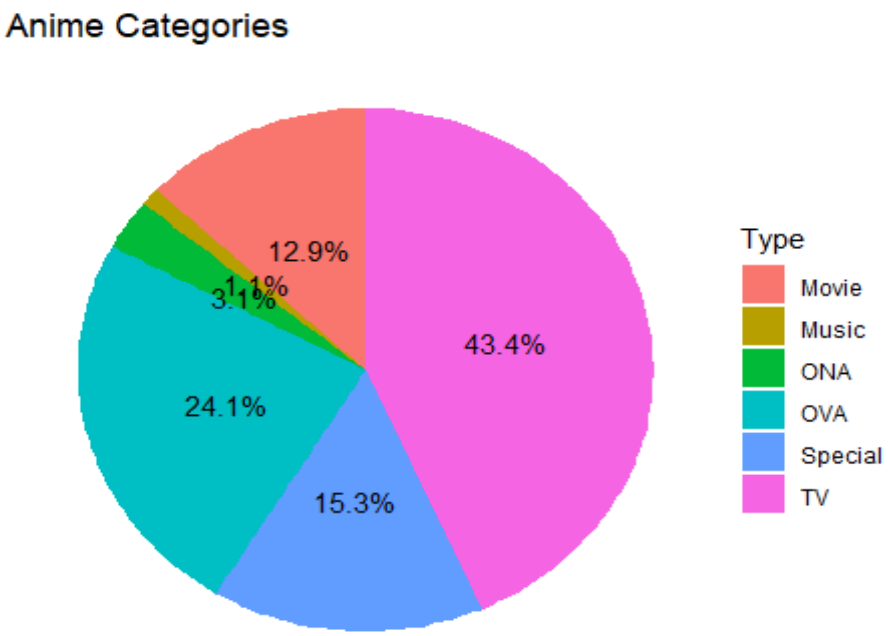
```r
Anime_types<-merged_data %>%
  group_by(Type) %>%
  summarise(Total= n_distinct(Anime_ID)) %>%
  arrange(desc(Total))


# Calculate the percentage of each category
Anime_types <- Anime_types %>%
  mutate(Percentage = (Total / sum(Total)) * 100)
datatable(Anime_types)
```

	Type	Total	Percentage
1	TV	1895	43.43341737336695
2	OVA	1052	24.11184964473986
3	Special	667	15.28764611505845
4	Movie	564	12.92688517075407
5	ONA	136	3.117121246848499
6	Music	49	1.12308044923218

```r
# Converting 'Type' variable to categorical using factor()
Anime_types$Type <- factor(Anime_types$Type)
str(Anime_types)
```

```
## tibble [6 × 3] (S3: tbl_df/tbl/data.frame)
##  $ Type      : Factor w/ 6 levels "Movie","Music",..: 6 4 5 1 3 2
##  $ Total     : int [1:6] 1895 1052 667 564 136 49
##  $ Percentage: num [1:6] 43.43 24.11 15.29 12.93 3.12 ...
```

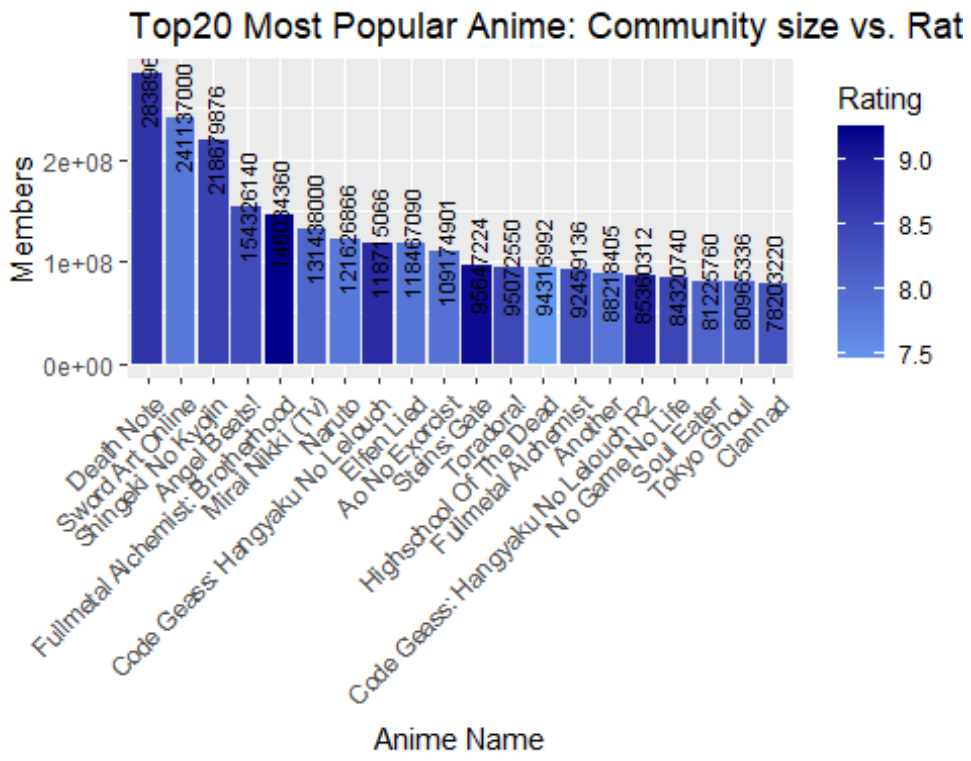
```r
# Plotting a pie chart
ggplot(Anime_types, aes(x = "", y = Percentage, fill = Type, label =
paste0(round(Percentage, 1), "%"))) +
  geom_bar(stat = "identity", width = 1) +
  coord_polar("y", start = 0) +
  labs(title = "Anime Categories", fill = "Type", y = "Percentage") +
  geom_text(position = position_stack(vjust = 0.5), color = "black") +
  theme_void() +
  theme(legend.position = "right")
```



From the plot we can say that majority (43.4%) of anime is aired on TV.

2. Most Popular Anime: Community size vs. Ratings

```
# Filtering the data to select the top 20 rows based on "Members"
top_20 <- merged_data %>%
  group_by(Name) %>%
  summarise(Total_Members = sum(Members),
            Rating = mean(Rating, na.rm = TRUE)) %>%
  top_n(20, Total_Members) %>%
  arrange(desc(Total_Members))


# Creating the bar plot of top 20 popular anime
ggplot(data = top_20,
       mapping = aes(x = reorder(Name, -Total_Members), y =
Total_Members,fill = Rating)) +  # Reorder anime names based on Members
  geom_bar(stat = "identity") +
  geom_text(aes(label = Total_Members), vjust = 0.5, color = "black", size =
3, angle = 90) +  # Add labels inside the bar
  labs(x = "Anime Name", y = "Members") +
  scale_fill_gradient(low = "cornflowerblue", high = "darkblue")+
  ggtitle("Top20 Most Popular Anime: Community size vs. Ratings")+
  theme(axis.text.x = element_text(angle = 45, hjust = 1))  # Rotate x-axis
labels for better readability
```



3. Exploring the different genres

```
glimpse(merged_data$Genre)
```

```
##  chr [1:46986] "Action, Adventure, Comedy, Drama, Sci-Fi, Space" ...
```

We can see multiple genres are separated by a ",". now lets split the genre of each row seperated by a comma(,)

```
# Extract distinct anime
distinct_genres <- merged_data %>%
  select(Anime_ID, Genre) %>%
  distinct()

glimpse(distinct_genres)
```

```
## Rows: 4,363
## Columns: 2
## $ Anime_ID <int> 1, 5, 6, 7, 8, 15, 16, 17, 18, 19, 20, 22, 24, 25, 26,
27, 28…
## $ Genre    <chr> "Action, Adventure, Comedy, Drama, Sci-Fi, Space",
"Action, D…
```

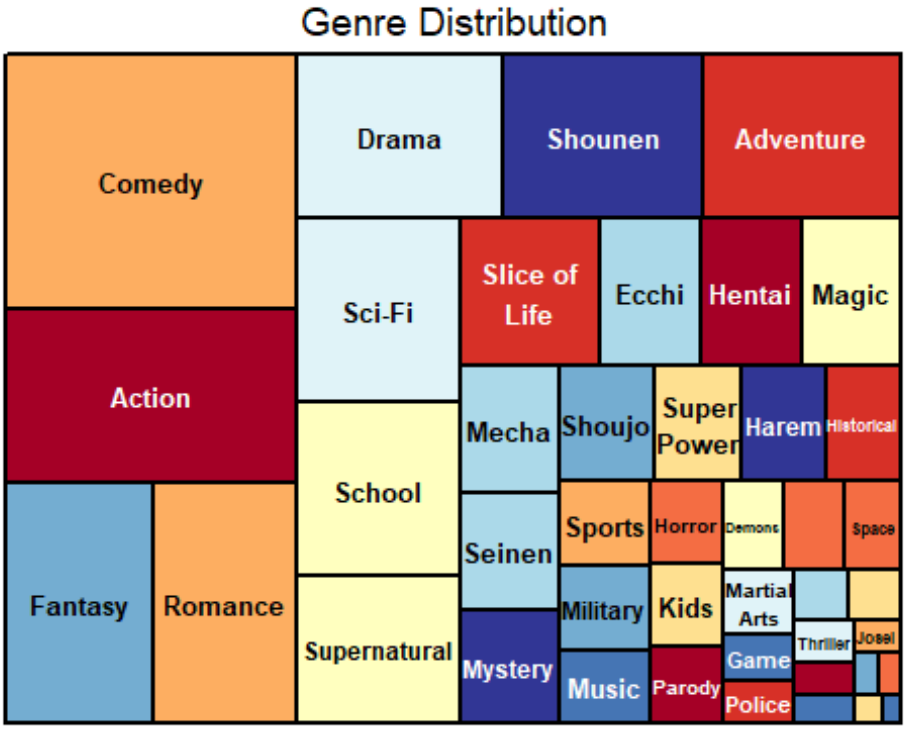
```
# Split the string in each row to extract individual genres
genre_list <- strsplit(distinct_genres$Genre, ", ")

# Convert the list of lists into a single vector
all_genres <- unlist(genre_list)
```

```r
# Counting the occurrences of each genre
genre_counts <- table(all_genres)

# Converting the result into a dataframe
genre_df <- as.data.frame(genre_counts)

# Renaming the columns
colnames(genre_df) <- c("Genre", "Count")

genre_df <- genre_df %>%
  arrange(desc(Count))

glimpse(genre_df)
```

```
## Rows: 43
## Columns: 2
## $ Genre <fct> Comedy, Action, Fantasy, Romance, Drama, Shounen, Adventure,
Sci…
## $ Count <int> 1984, 1351, 952, 929, 912, 892, 869, 812, 760, 651, 551,
407, 39…
```

We have 43 different genre in our merged dataset

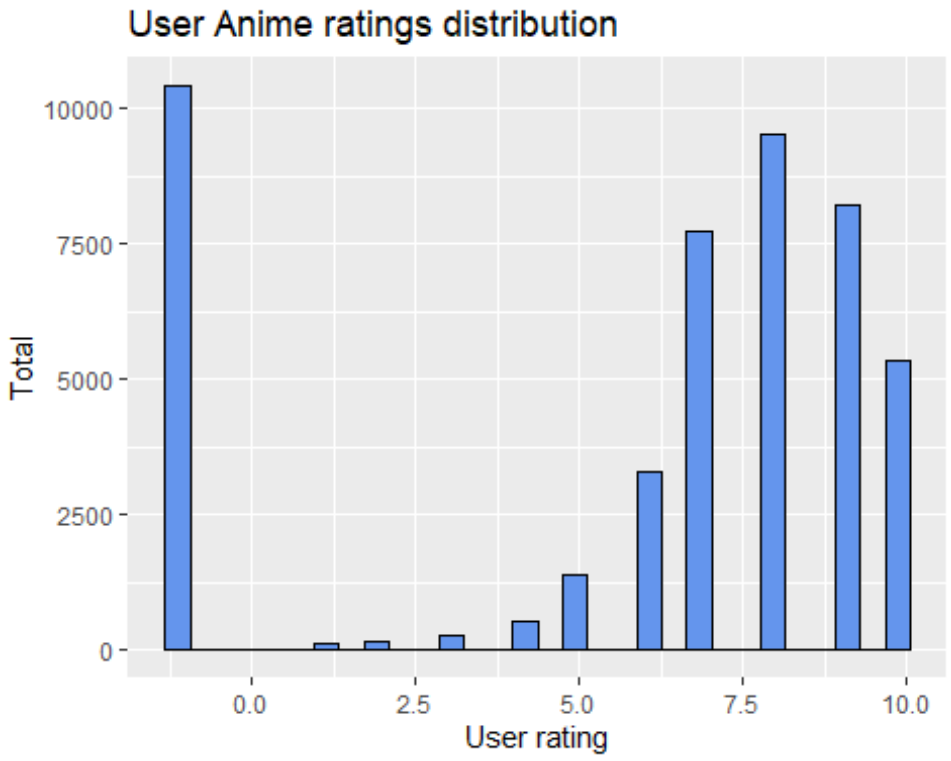
```r
# Creating the treemap plot

treemap(genre_df,
        index = "Genre",  # Specify the index column
        vSize = "Count",  # Specify the column for the size of the tiles
        title = "Genre Distribution",  # Title of the treemap
        fontsize.title = 14,  # Font size for the title
        fontsize.labels = 10,  # Font size for the labels
        fontsize.labels.min = 6,  # Minimum font size for labels
        align.labels = list(c("center", "center")),  # Alignment of labels
        palette = "RdYlBu",  # Color palette for the tiles
        width = 10,  # Width of the plotting area
        height = 10  # Height of the plotting area
)
```



4: Finding potential outliers

Finding the distribution of the user rating to identify any potential outliers.

```
# Sort merged_data by User_Rating in descending order
merged_data <- merged_data %>%
  arrange(desc(User_Rating))

#generating a histogram of user rating

ggplot() +
  geom_histogram(data = merged_data, aes(x = User_Rating), fill =
"cornflowerblue", color = "black") +
  labs(title = "User Anime ratings distribution", x = "User rating", y =
"Total")

## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



We can see that -1 is a outlier in user ratings. -1(If the user has watched the anime but haven't rated it)

```
skewness(merged_data$User_Rating)
```

```
## [1] -0.9636827
```

The skewness is -0.96<0 proves that distribution is negatively skewed.

STEP 4: Modeling

Before proceeding let us make sure to consider only valid scenarios.

Lets replace -1 (non rating) in User_rating column with the possible predicted rating.i.e,Let's predict user ratings for anime they haven't rated.

```
#Identify unrated anime
unrated_anime <- merged_data %>%
  filter(User_Rating == -1)

glimpse(unrated_anime)
```

```
## Rows: 10,426
## Columns: 9
## $ Anime_ID    <int> 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 5, 5, 5, 5, 5,
5, 5…
## $ Name        <chr> "Cowboy Bebop", "Cowboy Bebop", "Cowboy Bebop",
"Cowboy Be…
```

```
## $ Genre       <chr> "Action, Adventure, Comedy, Drama, Sci-Fi, Space",
"Action…
## $ Type        <chr> "TV", "TV", "TV", "TV", "TV", "TV", "TV", "TV", "TV",
"TV"…
## $ Episodes    <chr> "26", "26", "26", "26", "26", "26", "26", "26", "26",
"26"…
## $ Rating      <dbl> 8.82, 8.82, 8.82, 8.82, 8.82, 8.82, 8.82, 8.82, 8.82,
8.82…
## $ Members     <int> 486824, 486824, 486824, 486824, 486824, 486824,
486824, 48…
## $ User_ID     <int> 259, 221, 285, 270, 274, 302, 218, 13, 328, 375, 54,
342, …
## $ User_Rating <int> -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -
1, -1…
```

We can see that we have 10,426 unrated entries in our data frame.

```
# Predict ratings for unrated anime
predicted_ratings <- unrated_anime %>%
  group_by(Name) %>%
  summarise(Predicted_Rating = mean(Rating))

# Merge predicted ratings with merged_data based on the anime name
merged_data <- merge(merged_data, predicted_ratings, by = "Name", all.x =
TRUE)

# Replace "-1" with predicted ratings
merged_data$User_Rating[merged_data$User_Rating == -1] <-
merged_data$Predicted_Rating[merged_data$User_Rating == -1]

# Remove the temporary column used for prediction
merged_data <- subset(merged_data, select = -c(Predicted_Rating))

merged_data <- merged_data %>%
  arrange(desc(User_Rating))

# View updated dataset
summary(merged_data)
```

```
##      Name              Anime_ID        Genre              Type
##  Length:46986       Min.   :    1   Length:46986       Length:46986
##  Class :character   1st Qu.: 2581   Class :character   Class :character
##  Mode  :character   Median : 9736   Mode  :character   Mode  :character
##                     Mean   :10922
##                     3rd Qu.:16782
##                     Max.   :34240
##    Episodes            Rating        Members            User_ID
##  Length:46986       Min.   :2.00   Min.   :     43   Min.   :  1.0
##  Class :character   1st Qu.:7.30   1st Qu.:  56139   1st Qu.:155.0
##  Mode  :character   Median :7.70   Median : 134739   Median :282.0
```

```
##                     Mean   :7.68   Mean   : 200627   Mean   :270.1
##                     3rd Qu.:8.16   3rd Qu.: 284846   3rd Qu.:392.0
##                     Max.   :9.37   Max.   :1013917   Max.   :500.0
##   User_Rating
##  Min.   : 1.000
##  1st Qu.: 7.000
##  Median : 8.000
##  Mean   : 7.796
##  3rd Qu.: 9.000
##  Max.   :10.000
```

We have handled all the outliers.

```
dim(merged_data)

## [1] 46986     9
```

Merged_data includes columns: User_ID, Anime_ID, User_Rating

```
colnames(merged_data)

## [1] "Name"        "Anime_ID"    "Genre"       "Type"        "Episodes"
## [6] "Rating"      "Members"     "User_ID"     "User_Rating"
```

splitting the dataset into training(70%) and testing(30%)

```
merged_data$Liked <- ifelse(merged_data$User_Rating >= 7, 1, 0) #Creating
Liked column where if rating is >7 then its values is 1 and if <1 then its
value is 0
train_index <- createDataPartition(merged_data$Liked, p = 0.7, list = FALSE)
train_data <- merged_data[train_index, ]
test_data <- merged_data[-train_index, ]

# Preprocess the data
train_X <- train_data[, c("User_ID", "Anime_ID")]
train_y <- train_data$Liked
test_X <- test_data[, c("User_ID", "Anime_ID")]
test_y <- test_data$Liked
```

Training the kNN model using caret's train function

```
model_knn <- train(x = train_X, y = train_y, method = "knn", trControl =
trainControl(method = "cv", number = 5))

## Warning in train.default(x = train_X, y = train_y, method = "knn",
trControl =
## trainControl(method = "cv", : You are trying to do regression and your
outcome
## only has two possible values Are you trying to do classification? If so,
use a
## 2 level factor as your outcome column.
```

```r
# Converting train_y to a factor with two levels
train_y <- as.factor(train_y)

test_y <- as.factor(test_y)

# Make predictions on the test data
test_predictions <- predict(model_knn, newdata = test_X)

# Evaluate accuracy
accuracy <- mean(test_predictions == test_y)
cat("Accuracy:", accuracy, "\n")
```

```
## Accuracy: 0.8251862
```

This shows that our model is 83% accurate. It indicates that the kNN model performs reasonably well on the test data.

Anime recommendation function:

```r
recommend_similar_anime <- function(anime_id, n) {
  # Find the row corresponding to the input anime ID in the test data
  input_anime_row <- test_data[test_data$Anime_ID == anime_id, ]

  # Check if the input anime ID is present in the test data
  if (nrow(input_anime_row) == 0) {
    print("Anime not found in the dataset.")
    return(NULL)
  }

  # Extract features of the input anime
  input_features <- input_anime_row[, c("Genre", "Rating", "Members")]

  #Calculate similarity between input anime and all other anime based on
features
  #lapply is a function in R used to apply a given function to each element
of a list or vector, and returns a list containing the results.

  anime_similarity <- lapply(unique(merged_data$Anime_ID), function(id) {
    anime_row <- merged_data[merged_data$Anime_ID == id, ]
    if (nrow(anime_row) > 0) {
      # Calculating the similarity score based on the intersection of genres
      similarity <- sum(input_features$Genre %in% anime_row$Genre) /
length(input_features$Genre)
      return(c(id, similarity))
    } else {
      return(NULL)
    }
  })
```

```
  # Remove NULL values and convert to matrix
  anime_similarity <- matrix(unlist(anime_similarity), ncol = 2, byrow =
TRUE)
  anime_similarity <- anime_similarity[complete.cases(anime_similarity), ]
  colnames(anime_similarity) <- c("Anime_ID", "Similarity")

  # Order by similarity and get top n similar anime
  similar_anime <- anime_similarity[order(-anime_similarity[, "Similarity"]),
"Anime_ID"][1:n]

  return(similar_anime)
}
```

Function to map anime names to anime IDs:

```
get_anime_id <-function(anime_name) {
  anime_id <- merged_data$Anime_ID[merged_data$Name == anime_name]
  if (length(anime_id) == 0) {
    print("Anime name not found in the dataset.")
    return(NULL)
  }
  return(anime_id)
}
```

Function to retrieve anime names

```
get_anime_names <- function(anime_ids) {
  anime_names <- unique(merged_data$Name[merged_data$Anime_ID %in%
anime_ids])
  return(anime_names)
}
```

Example:

```
query_anime_name <- "Naruto"
```

Replace with the anime name of the anime you want recommendations for.

```
query_anime_name<-str_to_title(query_anime_name) #Handling case sensitive
inputs.
query_anime_id<-get_anime_id(query_anime_name)
recommendations <- recommend_similar_anime(query_anime_id, 10) #Lets take 10
similar anime suggestions

## Warning in test_data$Anime_ID == anime_id: longer object length is not a
## multiple of shorter object length

recommended_names <- get_anime_names(recommendations)

cat(paste("Anime recommendations similar to", query_anime_name, "are:\n",
paste(recommended_names, collapse = "\n"), "\n"))
```

outputs:

```
Anime recommendations similar to Naruto are:

07-Ghost
11eyes
Boruto: Naruto The Movie
Naruto
Naruto: Shippuuden Movie 4 - The Lost Tower
Naruto Soyokazeden Movie: Naruto To Mashin To Mitsu No Onegai Dattebayo!!
Naruto: Shippuuden Movie 3 - Hi No Ishi Wo Tsugu Mono
Boruto: Naruto The Movie - Naruto Ga Hokage Ni Natta Hi
Naruto X Ut
Naruto Shippuuden: Sunny Side Battle
```

```
Anime recommendations similar to Death Note are:
 07-Ghost
11eyes
11eyes: Momoiro Genmutan
91 Days
Aa! Megami-Sama! (Tv)
Aa! Megami-Sama! Movie
Accel World
Accel World Ex
Death Note
Death Note Rewrite
```