INDIRA GANDHI DELHI TECHNICAL UNIVERSITY FOR WOMEN

IT Workshop Project

Designing A Recommender system for Animes To Increase User Engagement And Generate More Revenue

Submitted to:

Mr.Santanoo Pattnaik

Submitted by:

Medhavi Bhardwaj(03201192022)

Medhavi Singh(03301192022)

Monya Mehta(03401192022)

Index

SYNOPSIS:

Introduction:

Anime is a popular type of animation, that originated in Japan that has become widely populated throughout the world in the last decade. Animes cover a large variety of themes and genres including action, romance, sci-fi, fantasy, comedy, etc. Animes are widely known for their unique art style and have helped in educating the Western demographic about East Asian culture.

Anime, short for "animation," refers to a distinctive style of animated entertainment that originated in Japan. It encompasses a wide range of genres, including action, adventure, romance, science fiction, fantasy, horror, and more. Anime is not limited to a specific age group and caters to viewers of all ages, from children to adults.

Situation:

There are several recommendation systems being used in the anime industry to help users discover new anime titles based on their preferences. The most commonly used include collaborative filtering, content-based filtering, hybrid approaches, Natural Language Processing (NLP), and Deep Learning Models.

Aim/Objective (Goal):

Trying out a particular combination of parameters to build a recommendation system that will recommend new animes to the user by analyzing their watch history and preferences. Thus, the revenue of the streaming platforms will increase due to increased user engagement, higher customer ratings, and user satisfaction.

Proposed outcomes of the project:

To increase customer satisfaction and thus generate more customer engagements and subscriptions and increase the revenue generation of companies.

What are Recommendation Systems?

Recommendation systems analyze user preferences, behavior, and historical data to provide personalized recommendations. By understanding individual tastes and preferences, these systems can suggest relevant content, products, or services, enhancing the user experience.

With the vast amount of content available today, it can be challenging for users to navigate and find what they're looking for. Recommendation systems can suggest relevant content based on user preferences and browsing patterns, enabling users to discover new and interesting items they may have otherwise missed. This helps users explore a wider range of options and diversify their interests.

In e-commerce, recommendation systems can significantly impact sales by suggesting products that are likely to be of interest to customers. By presenting personalized recommendations, these systems can drive user engagement, encourage repeat visits, and increase the likelihood of making a purchase. This leads to higher customer satisfaction and improved conversion rates.

Recommendation systems streamline the search and discovery process by providing tailored suggestions. Users can save time and effort that would otherwise be spent manually searching through a large volume of options. By offering relevant recommendations, these systems simplify decision-making and make the overall user experience more efficient.

Let's explore some commonly used recommendation systems:

Content-Based Filtering:

Content-based filtering recommends items to users based on the characteristics and features of the items themselves. It analyzes the attributes or metadata of items a user has previously +has watched several action movies, a content-based recommender might suggest other action movies based on genre, actors, or directors.

Context-Aware Recommendation:

Context-aware recommendation systems take into account various contextual factors, such as time, location, device, and user demographics, to provide more personalized recommendations. By considering contextual information, these systems can adapt recommendations to specific situations and provide more relevant suggestions to users.

Collaborative Filtering:

Collaborative filtering is one of the most widely used recommendation techniques. It operates on the principle of finding similarities between users or items based on their past behaviour or preferences.

METHODOLOGY:

1. Data extraction:

The dataset used was extracted from Kaggle.

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

It consists of the user ratings and the user preferences for various animes and the user's watch history.

Data Description

In the given dataset, the independent parameters (also known as features or predictors) are:

1. *Anime ID:* A unique identifier for each anime.
2. *Name*: The name of the anime.
3. *Genre*: The genre(s) associated with the anime.
4. *Episode*: The number of episodes in the anime.
5. *Type*: The type of the anime (e.g., TV series, movie, OVA).
6. *Members*: The number of members who have added the anime to their list.

The dependent parameter (also known as the target variable or response variable) is:

Rating: The rating given to the anime by users.

In this dataset, the goal is to predict the rating of an anime based on its independent parameters.

2. Data Preparation:

Data preprocessing plays a crucial role in recommendation systems as it involves preparing and transforming raw data into a suitable format for analysis and generating accurate recommendations. Here are some key steps involved in data preprocessing for a recommendation system:

1. Data Cleaning: Data cleaning involves handling missing values, outliers, and inconsistencies in the dataset. Missing values can be imputed or treated using techniques like mean imputation, regression imputation, or deletion of incomplete records. Outliers can be identified and handled by methods such as removing them or replacing them with more appropriate values. Inconsistencies or errors in the data can be corrected through techniques like data validation and data profiling.

2. Data Transformation: Data transformation involves converting and normalizing data to make it suitable for subsequent analysis.

3. Feature Engineering: Feature engineering involves selecting, creating, or deriving relevant features from the raw data to enhance the recommendation system's performance. Feature engineering aims to capture meaningful patterns and relationships between users, items, and context that can improve the accuracy and personalization of recommendations.

4. Data Representation: In recommendation systems, data representation is often done in the form of a user-item matrix. The matrix represents the interactions or preferences of users for items. Rows correspond to users, columns correspond to items, and the entries represent the strength of interaction (e.g., ratings, clicks, purchase history).

5. Data Splitting: To evaluate the performance of recommendation models, the dataset is typically split into training, validation, and testing subsets. The training set is used to train the model, the validation set is used for model selection and hyperparameter tuning, and the testing set is used to assess the final model's performance.

3.Data visualization

It is the process of representing data in graphical or visual formats to facilitate understanding, analysis, and communication of information. It plays a crucial role in making complex data more accessible, identifying patterns, trends, and insights, and effectively conveying information to stakeholders. Here are some key aspects of data visualization

1. User Profiles: Visualizing user profiles can provide a summary of their preferences, behavior, and interactions. This can include visual representations of demographic information, past purchases, ratings, or browsing history. Interactive charts or graphs can be used to display this information, allowing users to explore their profiles and gain insights into their preferences.

2. Item Catalog: Visualization techniques can be applied to showcase the item catalog making it visually appealing and easier to navigate. Interactive visualizations, such as grid layouts, tree maps, or tag clouds, can help users explore the available items, filter or sort them based on different criteria, and discover new products or content.

3. Recommendation Explanations: One challenge with recommendation systems is the lack of transparency in how recommendations are generated. Data visualization can be employed to provide explanations for the recommendations, helping users understand the underlying factors and decision-making process. This can include visualizations of similarity metrics, user-item interactions, or feature importance, allowing users to see why certain recommendations were made.

4. User Feedback: Visualizing user feedback, such as ratings, reviews, or comments, can provide valuable insights into the quality and effectiveness of recommendations. Sentiment analysis can be applied to analyze textual feedback and generate visualizations that summarize user sentiments towards recommended items.

5. Performance Metrics: Visualization techniques can be used to present performance metrics of the recommendation system. This can include visualizing metrics like precision, recall, click-through rates, or conversion rates over time, allowing system administrators to monitor and assess the effectiveness of the recommendation algorithms.

6. Personalization Control: Providing users with control over personalization levels can enhance their engagement with the recommendation system. Visualizing the impact of personalization settings on the recommendations can help users understand and adjust their preferences accordingly. This can be done through interactive sliders or toggles that show the changes in recommended items based on different personalization levels.

4. Making the Model

Algorithm Used:

The algorithm used in our project is user based collaborative filtering (UBCF) with cosine similarity.

Collaborative Filtering (CF) is a popular technique in recommender systems that predicts a user's preferences or interests by collecting and analyzing information from similar users. User-Based Collaborative Filtering (UBCF) is one of the approaches used in CF.

UBCF focuses on finding users who have similar tastes and preferences to the target user and leveraging their opinions to make recommendations. It assumes that users who have agreed on certain items in the past will agree on other items in the future.

Here's a general overview of how UBCF works:

1. User Similarity Calculation: The first step is to measure the similarity between the target user and other users in the system. Similarity can be calculated using various metrics, such as cosine similarity or Pearson correlation coefficient. These metrics compare the rating patterns or behaviors of users to determine their similarity.

2. Neighborhood Selection: Once the similarity between users is calculated, a subset of similar users, often called the "neighborhood," is selected. The neighborhood can be defined based on a fixed number of most similar users or a similarity threshold.

3. Rating Prediction: After selecting the neighborhood, the next step is to predict the target user's rating for items they have not yet rated. This is typically done by taking a weighted average of the ratings of similar users for those items. The weights are often based on the similarity scores calculated in the previous step.

4. Top-N Recommendations: Finally, the system generates a list of top-N recommendations by selecting the items with the highest predicted ratings for the target user. These recommendations can be personalized and tailored to the individual user's preferences.

UBCF has some advantages, such as its simplicity and the ability to provide recommendations for new or less popular items. However, it also has limitations, including the cold start problem (difficulty in providing recommendations for new users with little to no history) and scalability issues when dealing with large datasets.

To implement UBCF, you would typically need a dataset containing user-item interactions or ratings. The similarity calculation and recommendation generation steps can be implemented using various programming languages and libraries, such as Python with libraries like scikit-learn or TensorFlow.

In User-Based Collaborative Filtering (UBCF), cosine similarity is commonly used to measure the similarity between users based on their rating vectors.

Code and Explanation:

The code performs collaborative filtering-based recommendation for anime using the “recommender lab” package in R. Let's go through the code step by step:

a)Importing necessary libraries

The code imports several libraries that are required for performing the recommendation and manipulating the data. These libraries include string, reshape2, data.table, recommenderlab, ggplot2, and dplyr.

```r
{r}
library(stringi)
library(reshape2)
library(data.table)
library(recommenderlab)
library(ggplot2)
library(dplyr)
#install.packages("plotly")
library(plotly)
```

```
Registered S3 method overwritten by 'data.table':
  method           from
  print.data.table
data.table 1.14.8 using 4 threads (see ?getDTthreads).  Latest news:
r-datatable.com

Attaching package: ‘data.table’

The following objects are masked from ‘package:reshape2’:

    dcast, melt

Loading required package: Matrix
Loading required package: arules

Attaching package: ‘arules’
```

b) Reading the datasets:

The `read.csv` function is used to read the data from these CSV files into R data frames. The `anime` data frame will contain information about the anime, and the `ratings` data frame will contain user ratings for the anime.

```{r}
animes = read.csv("C:\\Users\\hp\\Downloads\\archive (1)\\anime.csv")
ratings = read.csv("C:\\Users\\hp\\Downloads\\archive (1)\\rating.csv")
```

c) Replacing -1 with NA

The code removes any ratings that are less than 0 to NA

This step is done to eliminate invalid or missing ratings.

```{r}
ratings <- ratings %>% mutate(rating = ifelse(rating == -1, NA, rating))
```

d) Limiting the number of ratings:

The code limits the number of ratings to 50,000 by using the line ratings <- head (ratings, 50000)`.

This step is done to reduce the computational complexity and runtime for the recommendation process.

```{r}
ratings <- head(ratings,50000)
```

e) Making Correlation Matrix

```{r}
merged_data <- merge(animes, ratings, by = "anime_id")
# Select the variables of interest
subset_data <- merged_data[c("rating.x", "rating.y", "episodes",
"members")]

# Calculate the correlation coefficients
subset_data <- sapply(subset_data, as.numeric)
correlation <- cor(subset_data, use = "complete.obs")

# Extract the lower triangle of the correlation matrix
coefficients <- correlation[lower.tri(correlation)]

# Print the corellation matrix
print(correlation)
```

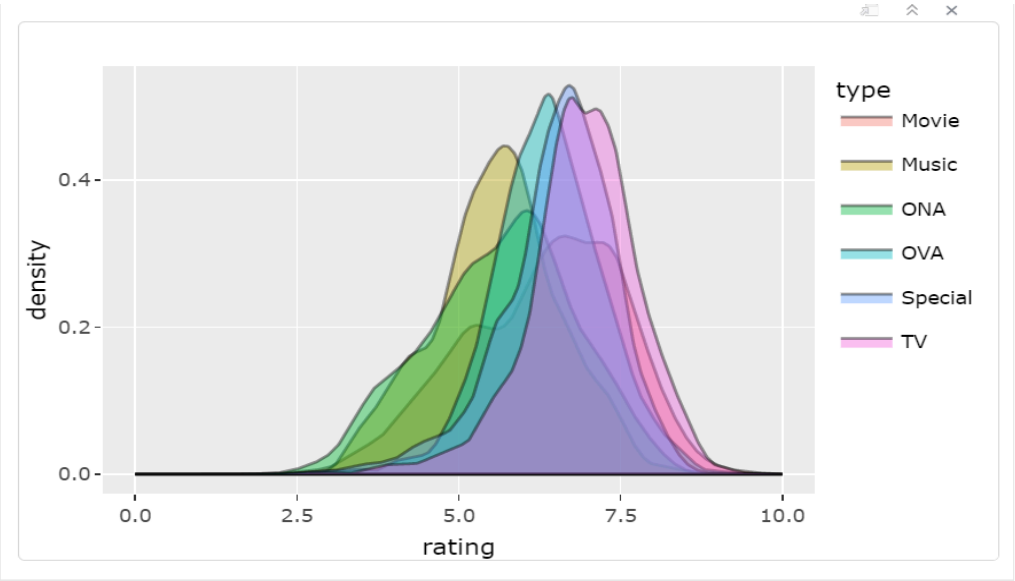
```
           rating.x   rating.y   episodes   members
rating.x 1.0000000 0.39863745 0.15778404 0.5341852
rating.y 0.3986375 1.00000000 0.06199281 0.2207851
episodes 0.1577840 0.06199281 1.00000000 0.2723102
members  0.5341852 0.22078515 0.27231020 1.0000000
```

f) Analysis And Inferences

How do ratings vary by anime type?

```{r}
g <- animes %>%
  filter(!is.na(rating)) %>%
  ggplot(aes(rating, group = type)) +
  geom_density(aes(fill = type), alpha = .4) +
  xlim(0, 10)

ggplotly(g)
```

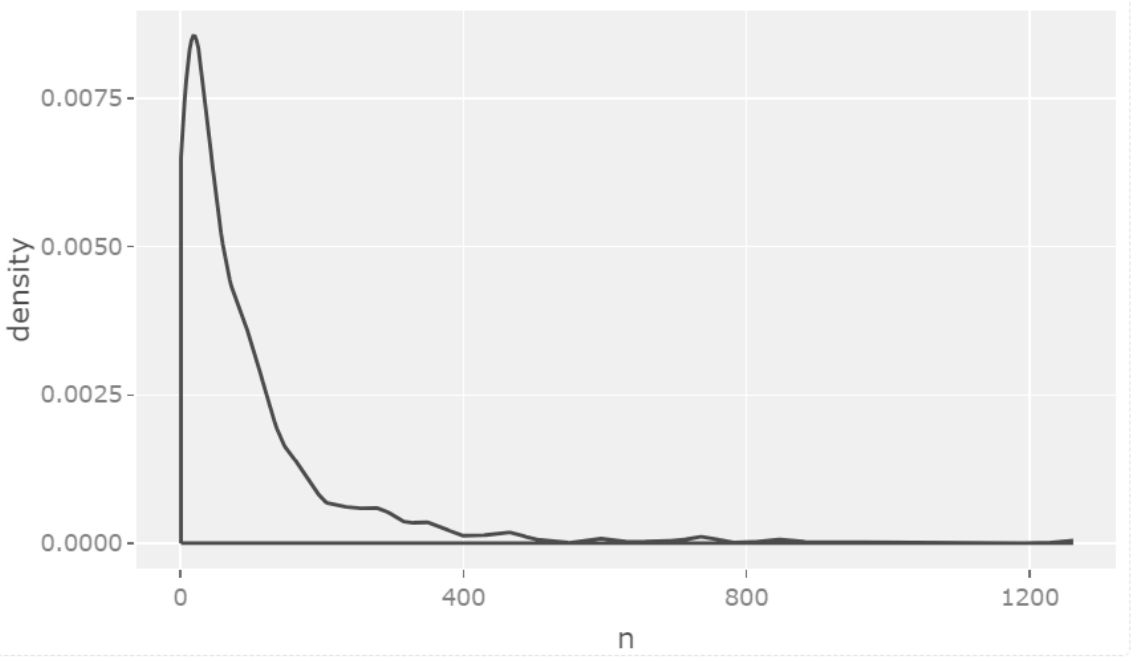


The special type of animes have the highest ratings peaking at about 7 to 7.2 out of 10, indicating nearly 50% of the people enjoy that particular genre the most.The TV genre seems to have the least ratings with only 20% of the viewers rating it around 5 out of 10. This data can be used by the company to determine which genre is the most watched, accordingly adjust their content to decide which new contents should be introduced.

How many animes do people rate?

```{r}
g <- ratings %>%
  count(user_id) %>%
  #filter(n < 500) %>%
  ggplot(aes(n)) +
  geom_density()

ggplotly(g)
```



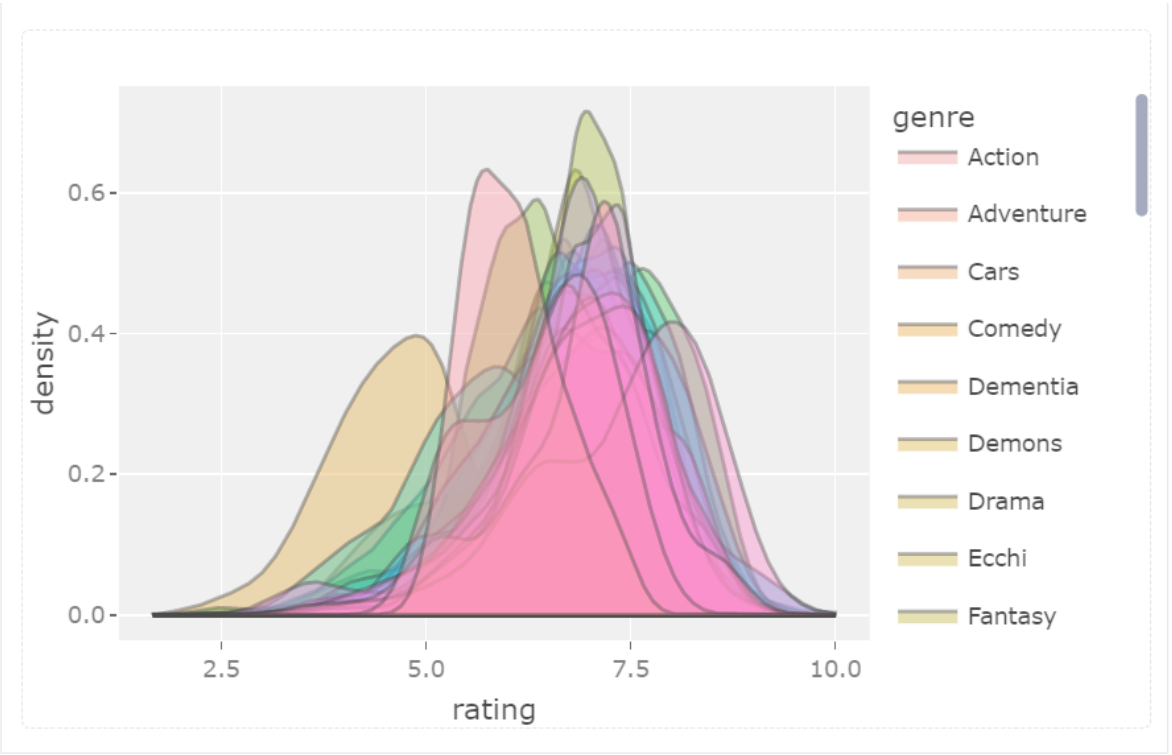
This indicates the number of anime episodes watched and subsequently rated by all the people who watched them. More than 75% people watch around 50 anime episodes on average. The percentage of people rating animes subsequently decreases, as the number of anime episodes increases. So the company can use this information to alter the number of recommendations given to a person at a time, so that the duration of the recommended anime doesn't exceed the free time the viewer has in hand. Most people watch anime for relaxation and recreation and their preferences may change according to the amount of free time they get.

Are ratings consistent across genres?

```r
{r}
animes %>%
  mutate(genre = map(genre, ~ strsplit(.x, ", ") %>% unlist())) %>%
  unnest(genre) ->
  shows_by_genre

g <- shows_by_genre %>%
  ggplot(aes(rating, group = genre)) +
  geom_density(aes(fill = genre), alpha = .4)

ggplotly(g)
```



The Harem genre has a very solid peak around 7.0.

The Yuri genre has a significantly lower mode, and is also skewed right.

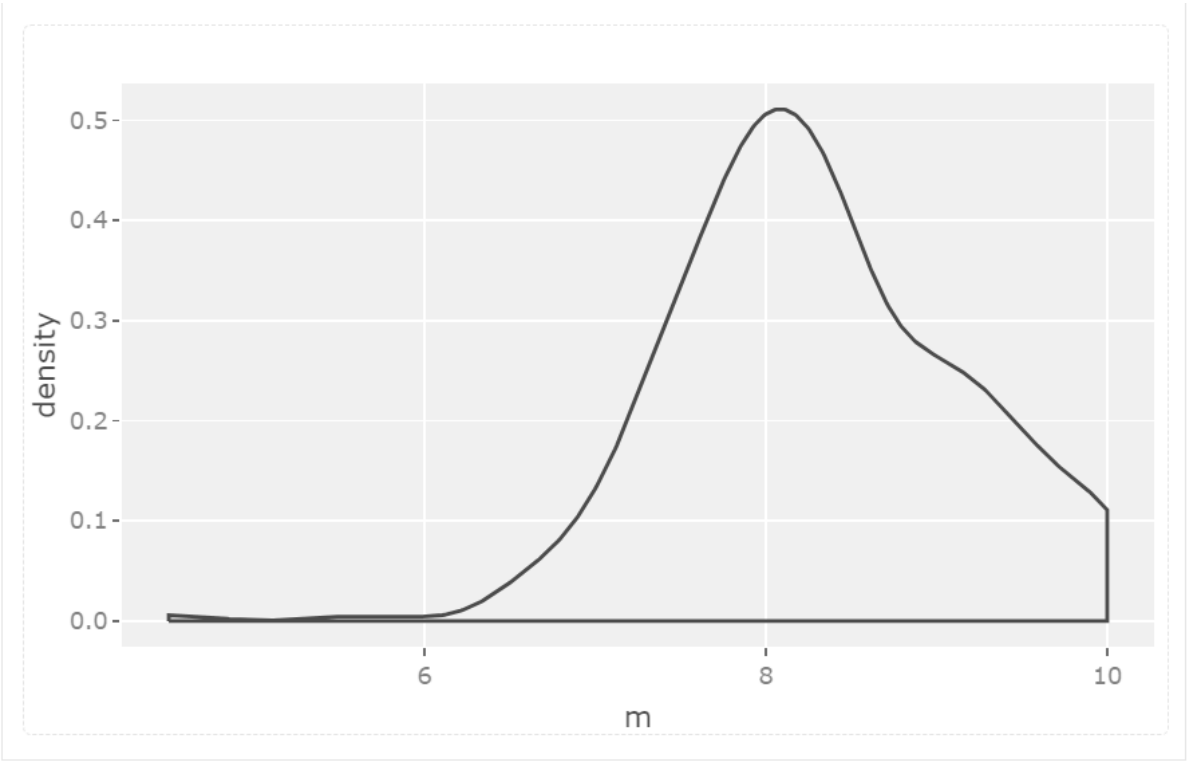
Shows in the NA genre are very low rated; presumably, they lack a genre because people don’t care about them.

The Dementia genre is very low rated.

What is the common ratings given?

```r
g <- ratings %>%
  group_by(user_id) %>%
  summarise(m = mean(rating)) %>%
  ggplot(aes(m)) +
  geom_density()

ggplotly(g)
```



Over 50% of the people have given a rating of about 8 out of 10 to the animes they've watched. This indicates that the people generally have liked the animes they’ve viewed and the streaming platform has good performance. This can be used as a metric to improve the streaming platforms performance by giving users personalized recommendations which increases the chances of them giving high rating if the recommended animes are similar to the ones they have already enjoyed.

h)Data Preparation

i)Removing the NA values

```{r}
ratings <- na.omit(ratings)
```

ii)Creating a rating matrix:

- The `dcast` function from the `reshape2` library is used to transform the `ratings` data frame into a wide format rating matrix, where each row represents a user and each column represents an anime.

- The `mean` aggregation function is used to calculate the average rating for each user-anime combination.

- The resulting rating matrix is stored in the variable `ratingmat`.

- The matrix is converted to the `realRatingMatrix` class, which is required by the `recommenderlab` package.

- The line `ratingmat = ratingmat[rowCounts(ratingmat) > 50,]` removes users who have rated fewer than 50 anime. This step helps ensure that there is enough data for reliable recommendations.

```{r}
ratingmat = dcast(ratings, user_id~anime_id ,fun.aggregate = mean,
value.var = "rating", na.rm=TRUE)
ratingmat = as.matrix(ratingmat[,-1])
ratingmat = as(ratingmat, "realRatingMatrix")
#removing users who have rated fewer than 50 anime
ratingmat = ratingmat[rowCounts(ratingmat) >50,]
```

```
Warning: The dcast generic in data.table has been passed a data.frame
and will attempt to redirect to the reshape2::dcast; please note that
reshape2 is deprecated, and this redirection is now deprecated as
well. Please do this redirection yourself like
reshape2::dcast(ratings). In the next version, this warning will
become an error.
```

iii) Splitting the data into training and test sets:

- The line `train = ratingmat[0:150]` creates a training set by selecting the first 150 users from the `ratingmat` matrix.

- This step is necessary to train the collaborative filtering recommender.

```{r}
train = ratingmat[0:150]
test = ratingmat[150:217]
```

i) Building the recommendation model:

1.Making the model

- The line `rec <- Recommender(train, method = "UBCF")` creates a user-based collaborative filtering recommender using the training data.

- The `"UBCF"` method indicates that the recommender should use user-based collaborative filtering.

```{r}
rec <- Recommender(train, method = "UBCF")
```

2. Generating recommendations:

- The line `pre <- predict(rec, ratingmat[151], n = 10)` generates recommendations for a specific user.

- The `predict` function is used to predict ratings for unrated items for the user with the ID 151.

- The `n = 10` argument specifies that the recommender should return the top 10 recommendations.

- The resulting recommendations are stored in the variable `pre` as a list.

```{r}
pre <- predict(rec, test, n = 10)
pre = as(pre, "list")
```

3. Converting recommendations to a data frame:

- The line `pre = as(pre, "list")` converts the recommendations to a list format.

- The following lines convert the list of recommendations to a data frame called `recommendations_df`.

- The column names of the data frame are set as "anime_id".

```{r}
recommendations_df <- data.frame(user_id = character(),
                                 anime_id = character(),
                                 stringsAsFactors = FALSE)
user_id<-1
user_recommendations <- pre[[user_id]]
user_df <- data.frame(user_id = rep(user_id, length(user_recommendations
)),anime_id = user_recommendations,stringsAsFactors = FALSE)
recommendations_df <- rbind(recommendations_df, user_df)
```

4. Joining with anime information:

- The line `names=left_join(recommendation_df,animes, by="anime_id")` joins the `Top_10_df` data frame with the `animes` data frame based on the "anime_id" column.

- This step adds additional information about the recommended anime, such as title, genre, etc., to the recommendations data frame.

```r
recommendations_df$anime_id <- as.integer(recommendations_df$anime_id)

top10 <- left_join( recommendations_df,animes, by = "anime_id")

top10
```

	user_id	anime_id	name	genre	type	episodes
1	1	16774	Inferno Cop	Action, Police	ONA	13
2	1	762	Bleach: Memories in the Rain	Action, Adventure, Drama, Shounen, Supernatural	Special	1
3	1	820	Ginga Eiyuu Densetsu	Drama, Military, Sci-Fi, Space	OVA	110
4	1	302	Mirai Shounen Conan	Adventure, Drama, Sci-Fi	TV	26
5	1	4073	Kyouryuu Daisensou Aizenborg	Action, Mecha	TV	39
6	1	6893	Oni Chichi	Hentai	OVA	2
7	1	10380	Oni Chichi: Re-birth	Hentai	OVA	1
8	1	11879	Oni Chichi: Re-born	Hentai	OVA	2
9	1	17909	Uchouten Kazoku	Comedy, Drama, Fantasy, Slice of Life	TV	13
10	1	18229	Gatchaman Crowds	Adventure, Sci-Fi	TV	12

Overall, this code reads the anime and rating data, processes and filters the data, builds a collaborative filtering recommender model, generates recommendations for a specific user, and then combines the recommendations with anime information for better understanding and presentation.

Outcomes:

Inferences From Data Visualization:

1.)The percentage of people who have rated animes by type and genre indicate the customer engagement and satisfaction levels of the people using the streaming platform to watch anime. This data accumulated can be used to improve the recommendations so that the users get more personalized recommendations, thus increasing the chances that they will enjoy the recommended animes. This will improve the streaming platform's performance and generate revenue.

2.)Similarly, the number of people rating animes indicate the viewer profiles and their preferences would indicate the time durations of animes they rated very high. Recommendations can thus be tailored to give the required number of animes of a particular time duration which matches with the viewers' free time period. Users would be able to enjoy animes they could completely watch during their free time if and when they have it.

3.) Similarly, consistency of ratings across genres can be used to decide what types of new animes are to be introduced on streaming platforms and recommended to the viewers thus increasing the number of subscriptions and the number of people watching and rating animes and thus increasing the performance levels of the streaming platform.

Limitations:

1)Accuracy:

Recommender systems are growing progressively more popular in online retail because of their ability to offer personalized experiences to unique users. Mean Average Precision at K (MAP@K) is typically the metric of choice for evaluating the performance of recommender systems. However, the use of additional diagnostic metrics and visualizations can offer deeper and sometimes surprising insights into a model's performance.

In collaborative filtering recommender systems user's preferences are expressed as ratings for items, and each additional rating extends the knowledge of the system and affects the system's recommendation accuracy. In general, the more ratings are elicited from the users, the more effective the recommendations are.

2)Personalization

Personalization is a great way to assess if a model recommends many of the same items to different users. It is the dissimilarity (1- cosine similarity) between user's lists of recommendations.

A high personalization score indicates user's recommendations are different, meaning the model is offering a personalized experience to each user.

Areas of Improvement:

1)Cannot handle fresh items

The prediction of the model for a given (user, item) pair is the dot product of the corresponding embeddings. So, if an item is not seen during training, the system can't create an embedding for it and can't query the model with this item. This issue is often called the cold-start problem.

2)Hard to include side features for query/item

Side features are any features beyond the query or item ID. For movie recommendations, the side features might include country or age. Including available side features improves the quality of the model.

To generalize WALS, augment the input matrix with features by defining a block matrix A, where:

- Block (0, 0) is the original feedback matrix
- Block (0, 1) is a multi-hot encoding of the user features.
- Block (1, 0) is a multi-hot encoding of the item features.

References:

1.)https://www.google.com/url?sa=t&source=web&rct=j&url=https://www.data-mania.com/blog/how-to-build-a-recommendation-engine-in-r/%23:~:text%3DThe%2520recommendation%2520package%2520in%2520R,Cosine%2520similarity&ved=2ahUKEwjhilikwY__AhVWU2wGHTMYDp4QFnoECBMQBQ&usg=AOvVaw1UzFVb8DLfaERJhEqMTbvl

2.)https://www.sciencedirect.com/science/article/abs/pii/S0747563216307518

3.)Maryam Khanian Najafabadi, Mohd Naz'ri Mahrin, Suriayati Chuprat, Haslina Md Sarkan, Improving the accuracy of collaborative filtering recommendations using clustering and association rules mining on implicit data, Computers in Human Behavior, Volume 67, 2017,

4.)https://developers.google.com/machine-learning/recommendation/collaborative/summary