FINAL PROJECT REPORT

Building a Movie Recommendation Engine

DS5110: Introduction to Data Management & Processing

Authors:

Pranali Bhosale

Pranita Deshpande

Rasagnya Reddy Avala

Amaranath Balaji Prithivirajan

Table of Contents

1. SUMMARY

Problem Setting & Objective:

The aim of this project was to build a movie recommendation engine using the TMDB datasets as well as Full MovieLens datasets. Popular streaming websites such as Netflix, Prime Video, Hulu etc. use some sort of an algorithm to recommend shows and movies to their users, based on their specific behavior.

In this project we have attempted to build 4 types of movie recommendation engines - weighted-rating based, sentiment based, content-based and user-based.

Data:

Two datasets were used in this project. They are as follows:

- TMDB Datasets:-
  [SOURCE:- https://www.kaggle.com/tmdb/tmdb-movie-metadata][1]
  This dataset gives details about 5000 movies from TMDB. Here we have two data tables as follows.
  - TMDB_5000_credits: This data table contains basic information such as movie ID, title, cast and crew. Movie ID is the primary key here.

- ❑ TMDB_5000_movies: This data table has 20 variables giving us additional details such as genre, budget, language information, popularity and so on. This table will have a combined primary key. Movie ID column from the previous table acts as a foreign key in this table.

- ❑ MovieLens Datasets:-
  [SOURCE:- http://grouplens.org/datasets/][2]
  This dataset describes 5-star rating and free text-tagging activity from MovieLens webpage. It contains 27753444 ratings and 1108997 tag applications across 58098 movies. This dataset is used in the Collaborative Filtering model.

2. METHODS

The first step in the project was to get the data into a tidy format.

Data Preprocessing:

Two datasets are used for the purpose of this study, the dataset primarily used was the TMDB dataset. The dataset is split into two datasets in separate files credits and the movies files. There was a total of 4803 observations, 20 variables in the movies dataset and 4 variables in the credits dataset. The challenge in tidying these datasets was the presence of JSON values in the variables, there were five such variables (genres, keywords, production_company, production_countries and spoken_languages) and two such variables (cast, crew) in the credits dataset. Other preprocessing such as type-conversion of the necessary variables, generating custom datasets was done on the fly as exploratory data analysis and modeling needs warranted. For the purpose of extracting JSON data 'jsonlite' package was used. To format the output tables better using HTML and CSS when needed 'kableextra' package was used.

The other dataset that was used was the 'movielens' dataset namely the 'ratings_small' dataset which consists of 44989 observations and 10 variables. This was primarily used for the purpose of collaborative filtering model to get user ratings. Preliminary data preprocessing steps like removing NaN values, type-conversion of variables alone was carried on the columns.

Exploratory Data Analysis:

In this section we explored the TMDB dataset and to look for patterns and insights.
Some of the interesting conclusions drawn from these graphs were as follows:

- ❑ 94% of the movies in this dataset have English as their original language. French is the next most common language.
- ❑ From 2010 onwards, a greater number of action, drama, thriller and comedy movies were released. While romance, Crime and Horror were the least common. Comedy and Drama are genres most associated with the loss of revenue.
- ❑ Altogether, TMBD uses 9,813 keywords to describe its movies. Some of the most commonly used Keywords are woman director, independent film etc. There are 412 movies with no keywords.

- ❑ Highest vote count was received by “Inception” movie with 13752 votes. While the highest vote average of 8.5 was received by “The Shawshank redemption”.

Weighted Ratings:

We researched about multiple Ratings for movies recommendations and later finalized on the below two recommendations ratings:

I. **IMDb rating**:

Some recommendation engines use the IMDB weighted rating formula. The IMDb has a “Top 250” and “Bottom 100” list of movies based only on the ratings of “regular voters”.

It is also derived based on the user rating formula given by,

$$W = \frac{R \cdot v + C \cdot m}{v + m}$$

Where, W = weighted rating
R = average rating of the movie(1 to 10)
v = number of votes for the movie
m = min. Votes needed to be in Top 250
C = the mean vote across the dataset

II. **Tomatometer rating**:

The tomatometer score is based on the opinions of hundreds of TV and Film critics.(1 to 5) We considered the regular voters as the film critics to get the tomatometer ratings. It is calculated only after a film receives at least 5 reviews. It is not based on a formula.

- Certified fresh <- consistent positive tomatometer score of 75% or higher
- Fresh <- if more than 60% of ratings are positive
- Rotten <- if less than 60% of ratings are positive

We generated the Top 250 and Bottom 100 lists for IMDb alongside their Tomatometer ratings.

Modeling:

In this project we have implemented 4 types of recommendation systems:-

I. **Popularity-based** engines: Recommender based on chosen Genre and Language (using IMDB weighted rating)
II. **Sentiment-based:** Recommender based on similarity of movie plot
III. **Content-based** engine: Recommendations are based on similarities between elements of a movie like genre, actor or director

IV. **Collaborative filtering** engine: providing movie recommendations tailored to a user by considering other users with similar tastes and preferences.

Taking a deeper look at the recommendation engines.

I. Popularity based engine: *Recommender based on chosen Genre and Language*

This is a simple recommender based on the IMDB weighted ratings mentioned earlier. It takes the Genre and Language as an input and displays the top 5 movies accordingly. English has been set as the default language in case the user only enters the genre.

Below is the result for Genre = 'Comedy' and Language = 'French'

id	title	genres	vote_average	vote_count	original_language	original_title	weighted_rating
194	Amélie	Comedy	7.8	3310	fr	Le fabuleux destin d'Amélie Poulain	7.628205
304410	The Bélier Family	Comedy	7.0	766	fr	La Famille Bélier	6.704209
11687	The Visitors	Comedy	7.1	420	fr	Les visiteurs	6.627843
48034	Little White Lies	Comedy	7.1	380	fr	Les petits mouchoirs	6.602669
8265	Welcome to the Sticks	Comedy	6.8	615	fr	Bienvenue chez les ch'tis	6.534026

Figure 1: Popularity based recommender output

II. Sentiment-based: *Recommender based on similarity of movie plot*

This next recommendation engine uses text mining and sentiment analysis techniques. The movies dataset contains a variable called 'overview' which has the movie plots stored in string type. The first step was to start by converting this variable into a tidy format, filtering out numbers, counting the total words per movie and getting the tf_idf values. Next, sentiment analysis using NRC lexicon was performed. And finally a recommender function was created which takes any sentiment as its input and outputs the top 5 movies which most closely match this sentiment based on the words found in its plot.

Here is the result from the recommender when 'joy' sentiment was given as the input.

id	title	sentiment	n
12429	Ponyo	joy	9
14736	Love & Basketball	joy	9
4523	Enchanted	joy	8
41144	To Save A Life	joy	8
824	Moulin Rouge!	joy	7

Figure 2: Sentiment-based recommender output

III. Content based recommender: Content-based recommenders treat recommendation as a user-specific classification problem and learn a classifier for the user's likes and dislikes based on product features. In this model actors, genres and directors were chosen to be considered as the similarity factor. In order to recommend movies based on content a function has been composed that takes a movie name as an input and returns top 5 recommendations similar to that movie. If in any case, a movie has the same similarity count then we use "weightedrating" to sort them.

Below is the result for 2 popular movies Titanic and Avatar. For Titanic the top 5 movies were recommended based on genre_1. Movies that had same value of "similarcount" were sorted based on "weightedrating."

id	title	similarcount	weightedrating	actor_1	actor_2	actor_3	director	genre_1	genre_2	genre_3	same_director	same_a1	same_a2	same_a3	same_g1	same_g2	same_g3
597	Titanic	7	8.087234	Kate Winslet	Leonardo DiCaprio	Frances Fisher	James Cameron	Drama	Romance	Thriller	1	1	1	1	1	1	1
4148	Revolutionary Road	4	12.246885	Leonardo DiCaprio	Kate Winslet	Michael Shannon	Sam Mendes	Drama	Romance	NA	0	1	1	0	1	1	0
905	Pandora's Box	3	94.572814	Louise Brooks	Fritz Kortner	Francis Lederer	G.W. Pabst	Drama	Thriller	Romance	0	0	0	0	1	1	1
11889	Iris	3	90.057165	Judi Dench	Jim Broadbent	Kate Winslet	Richard Eyre	Drama	Romance	NA	0	1	0	0	1	1	0
9709	My Summer of Love	3	74.416482	Natalie Press	Emily Blunt	Paddy Considine	Paweł Pawlikowski	Drama	Thriller	Romance	0	0	0	0	1	1	1
11065	O	3	60.732502	Mekhi Phifer	Josh Hartnett	Andrew Keegan	Tim Blake Nelson	Drama	Romance	Thriller	0	0	0	0	1	1	1

id	title	similarcount	weightedrating	actor_1	actor_2	actor_3	director	genre_1	genre_2	genre_3	same_director	same_a1	same_a2	same_a3	same_g1	same_g2	same_g3
19995	Avatar	7	7.576591	Sam Worthington	Zoe Saldana	Sigourney Weaver	James Cameron	Action	Adventure	Fantasy	1	1	1	1	1	1	1
18823	Clash of the Titans	4	7.590009	Sam Worthington	Liam Neeson	Ralph Fiennes	Louis Leterrier	Adventure	Fantasy	Action	0	1	0	0	1	1	1
27549	Beastmaster 2: Through the Portal of Time	3	197.821605	Marc Singer	Kari Wuhrer	Sarah Douglas	Sylvio Tabet	Action	Adventure	Fantasy	0	0	0	0	1	1	1
381902	The Monkey King 2	3	153.991228	Aaron Kwok	Gong Li	William Feng	Soi Cheang	Action	Adventure	Fantasy	0	0	0	0	1	1	1
9288	Dungeons & Dragons: Wrath of the Dragon God	3	110.774291	Bruce Payne	Mark Dymond	Clemency Burton-Hill	Gerry Lively	Action	Adventure	Fantasy	0	0	0	0	1	1	1
168705	BloodRayne	3	39.510393	Kristanna Loken	Ben Kingsley	Michelle Rodriguez	Uwe Boll	Action	Adventure	Fantasy	0	0	0	0	1	1	1

Figure 3: Content-based recommender output

IV. Collaborative filtering engine: Collaborative filtering is a recommendation technique. It provides recommendation to a user by considering the preferences of other users in the same pool or problem under study. This project makes use of 'Item Based Collaborative filtering' which is one of the two major categories of Collaborative filtering. The project makes use of 'recommenderlab' which is based on the research paper by Michael Hahsler[4]. The framework makes use of two main objects one a 'Recommender' object which takes in a 'ratingMatrix' , once we format the dataset as a 'ratingMatrix' and call the 'IBCF' tag which does 'Item Based Collaborative Filtering'. The similarity between the product under consideration and related products is considered to make recommendation. As part of this project the differences are elucidated by comparing the corresponding recommendations of a same item here a movie. 'recommenderlab' package provides an inbuilt method to achieve this.

3. RESULTS & DISCUSSION

The results have been displayed on the shiny platform. Shiny is an open source web application for R by R studio. It is interesting to display results of movie recommendations on an interactive platform to provide user-based recommendations. Shiny serves this purpose to communicate results via interactive visualizations and texts. Also, the ease of use and share-ability of the app are added features for a person not knowing the code to view the results. In-spite of some limitations of the app with few non-interactive functions, we successfully implemented a dashboard with 5 types of movie recommendations with interactive user preferences. (https://pranitad.shinyapps.io/movierecommender/)

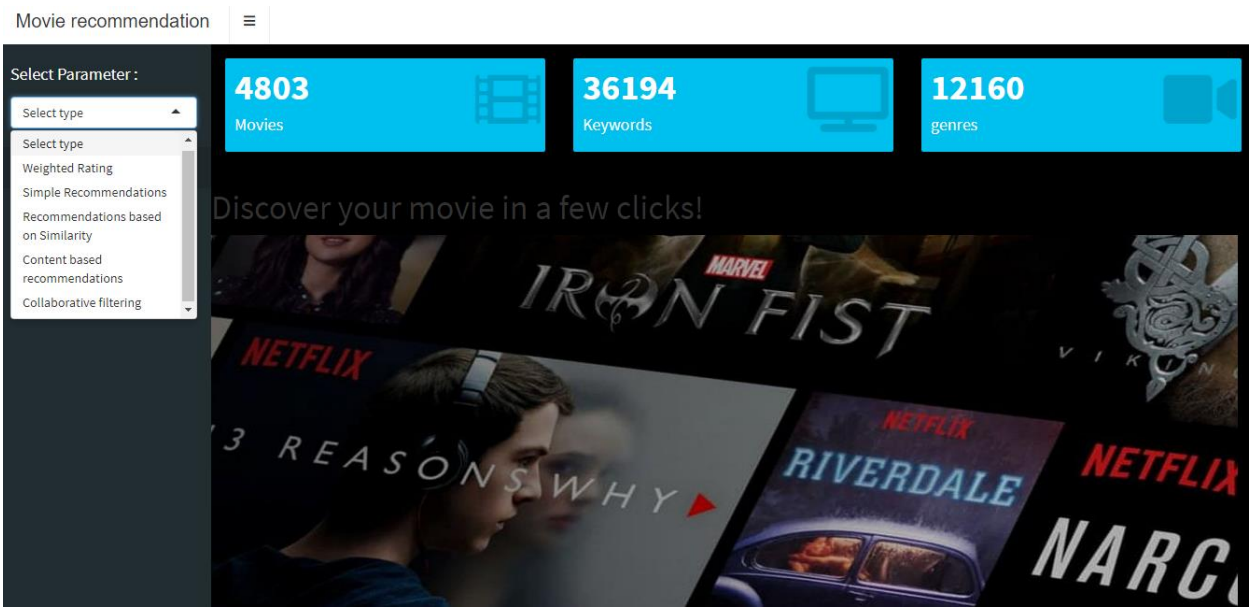


Figure 4: Shiny Interactive Dashboard

In conclusion, weighted rating seems to be a good measure to be used for movie comparison and analysis. First two recommenders are simple engines that do not take into account the user specific data, rather focus on the ratings and popularity. NCR proves to be a better sentiment analysis measure. ICBF model gives more precise recommendations. Although content-based recommender makes recommendations specific to a user, this technique requires domain knowledge and has limited ability to expand on existing interests. We have replaced the performance metrics with more suitable metrics for our classification problem, like TPR, FPR, Precision and Recall along with the use of similarity matrix like Pearson correlation matrix.

A real time movie recommendation could be built if we keep on updating the recent movies and have the user information database.

4. STATEMENT OF CONTRIBUTIONS

All group members contributed towards topic selection, making important decisions and report-writing.

Pranali Bhosale: In charge of exploratory data analysis, popularity-based recommender, sentiment-based recommender, tidying, editing and formatting of project reports and presentation.
Amaranath Balaji Prithivirajan: Carried out exploratory data analysis, research, tidying data, collaborative filtering engine, contributing for project presentation and report.
Rasagnya Reddy Avala: Implemented Content based recommendation engine, performed exploratory data analysis, contributed towards preparation of project presentation and report.
Pranita Deshpande: Performed tidying and pre-processing of data, implementing project plan for using weighted rating-based model, research, coding and successful implementation of Shiny dashboard, documenting project proposals, presentation and report.

5. REFERENCES

[1] Dataset Source

- **http://grouplens.org/datasets/**
- **https://www.kaggle.com/tmdb/tmdb-movie-metadata**

[2] IMDb and Tomatometer ratings

- **https://en.wikipedia.org/wiki/IMDb**
- **https://www.rottentomatoes.com/about**

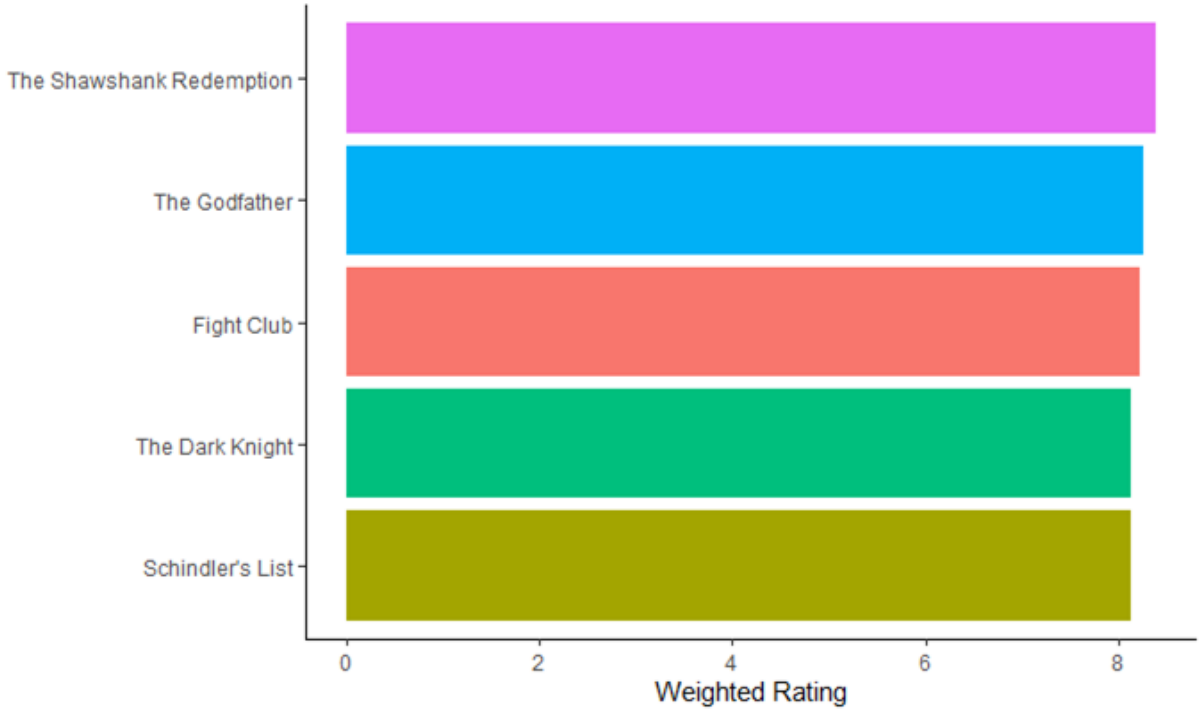
[3] Content based

- **https://www.kdnuggets.com/2019/11/content-based-recommender-using-natural-language-processing-nlp.html**

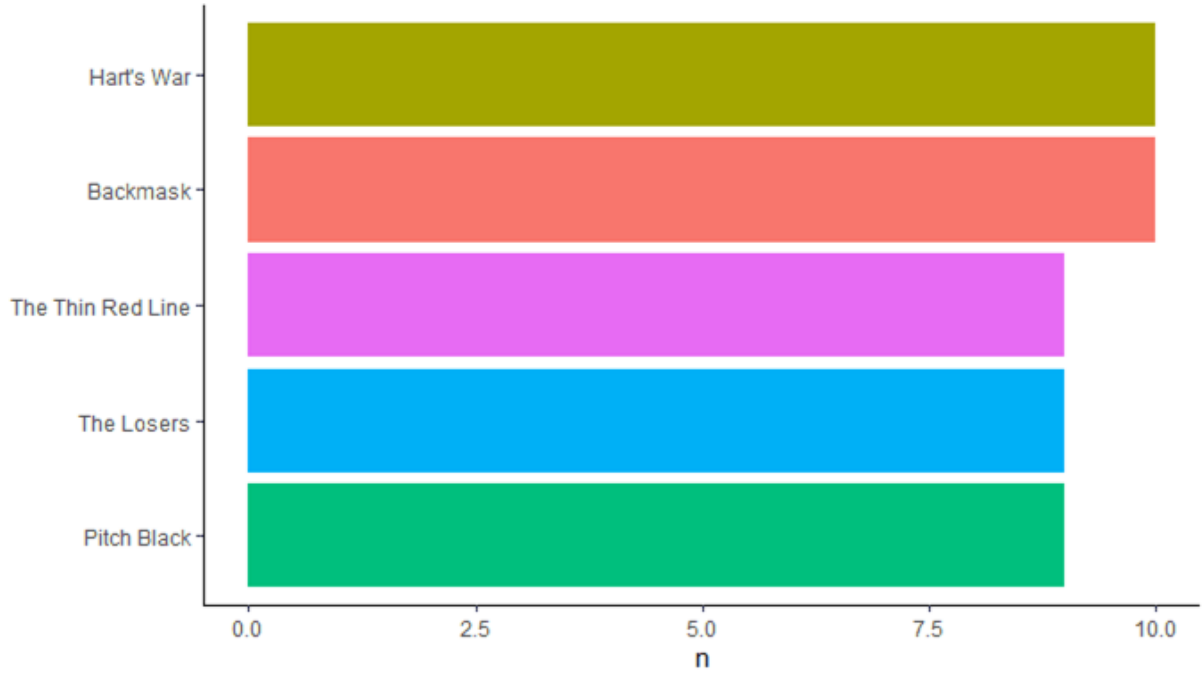
[4] recommenderlab: A Framework for Developing and Testing Recommendation Algorithms, Michael Hahsler

6. APPENDIX

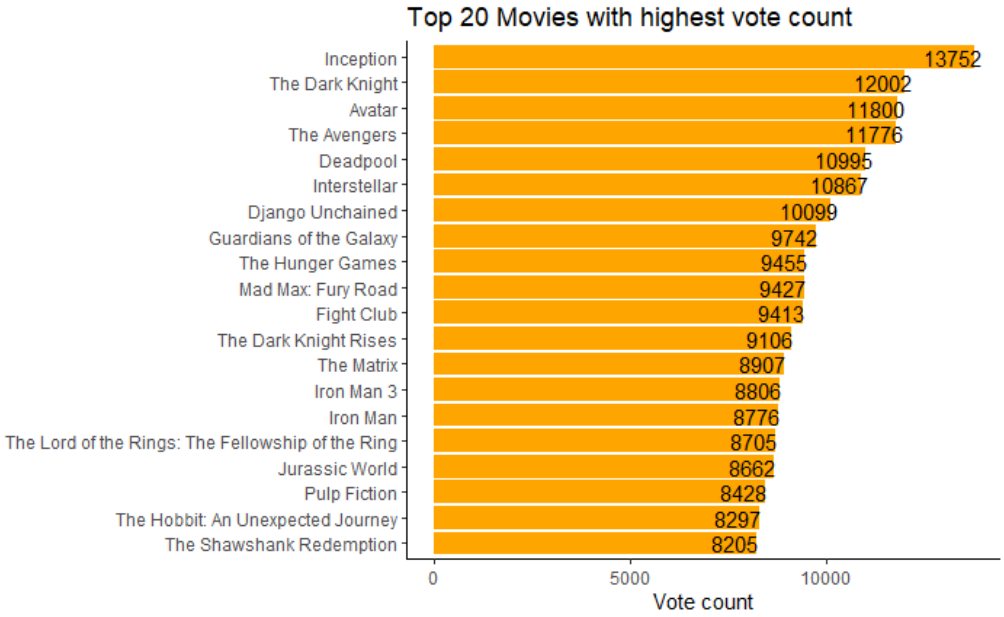
Below is the output from recommender based on chosen Genre and Language: Top 5 recommended English Drama movies

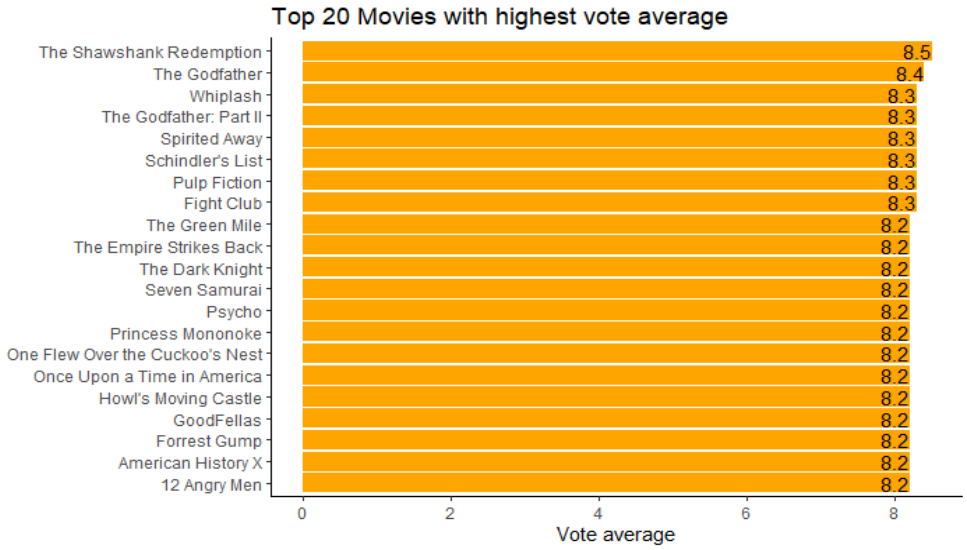


Below is the output from recommender based on similarity of movie plot: Top 5 recommended for "Fear" sentiment

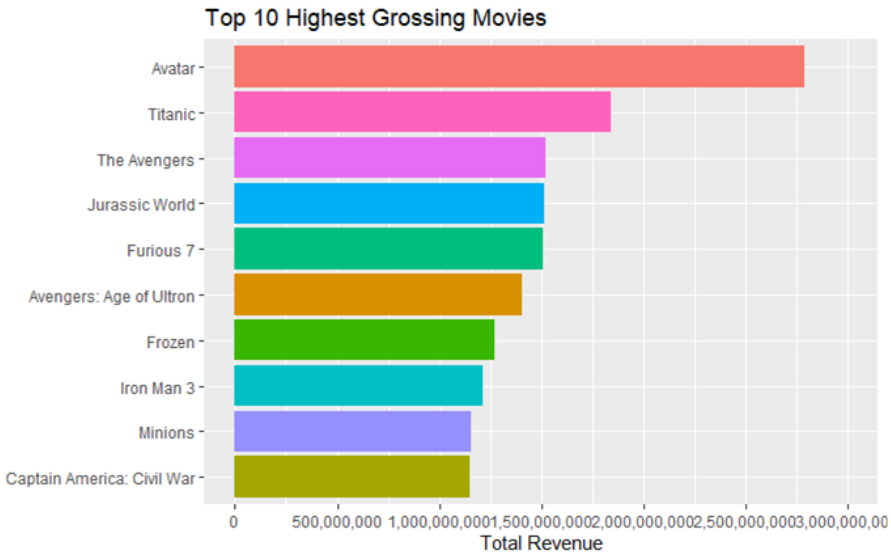


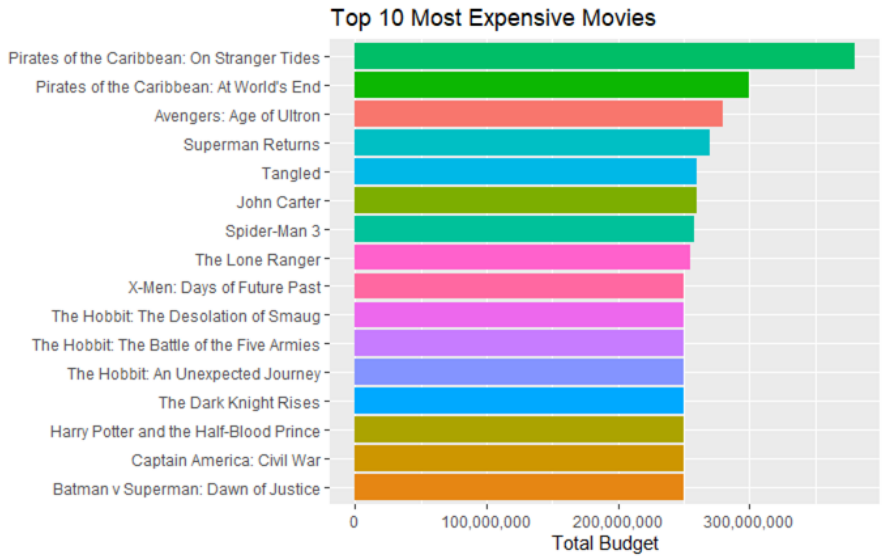
Top Rated Movies:



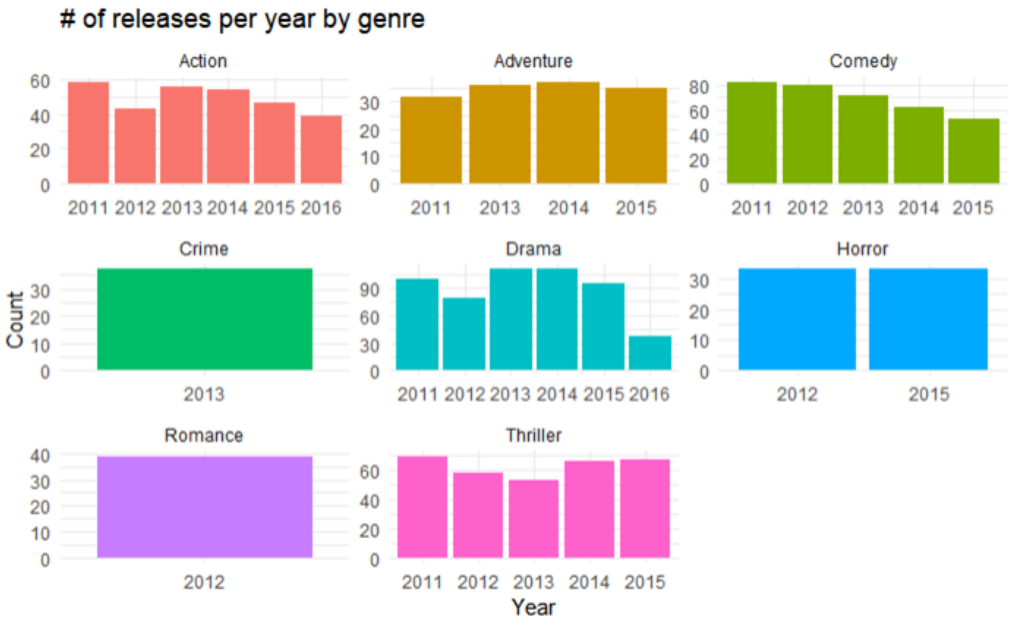


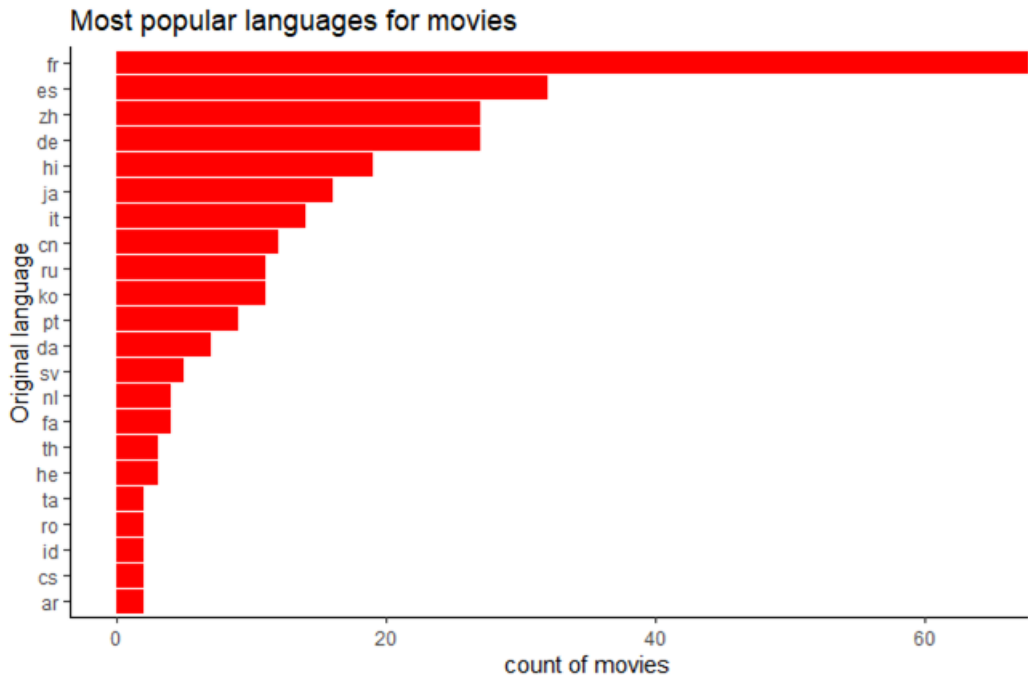
Budget & Revenue:

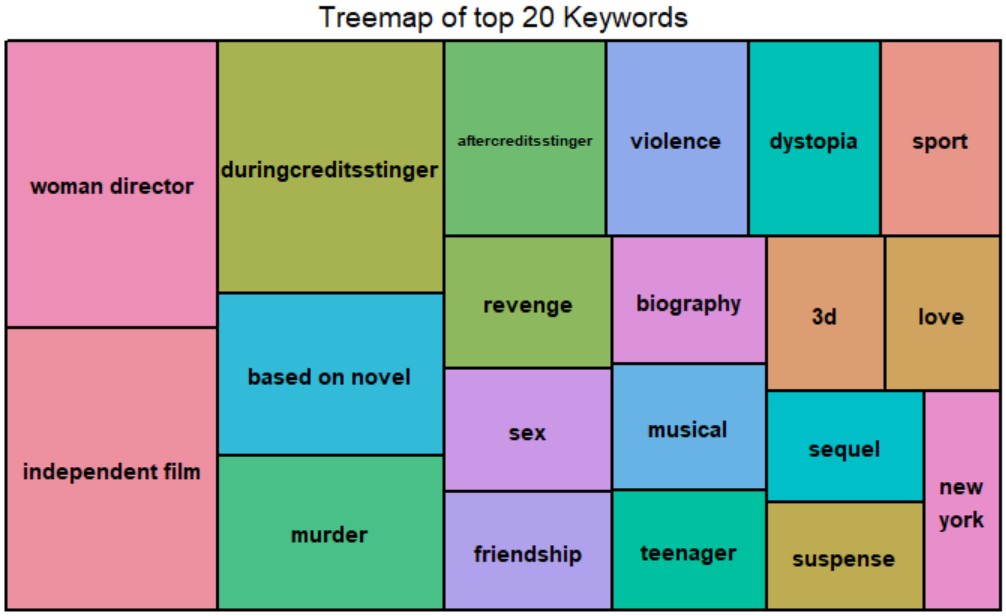




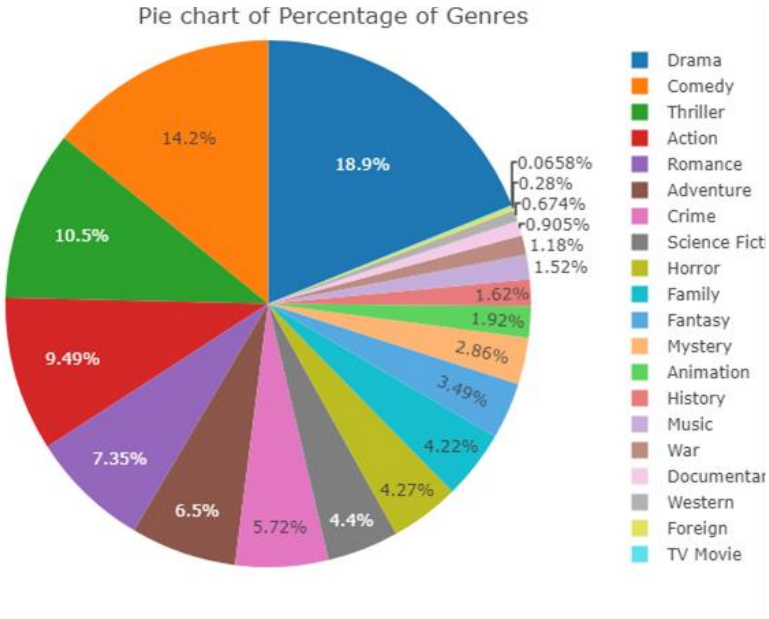
Releases per Year by Genre:




Most popular languages for movies
Original language
fr
es
zh
de
hi
ja
it
cn
ru
ko
pt
da
sv
nl
fa
th
he
ta
ro
id
cs
ar
0
20
40
60
count of movies


Treemap of top 20 Keywords
woman director
duringcreditsstinger
aftercreditsstinger
violence
dystopia
sport
revenge
biography
3d
love
based on novel
sex
musical
sequel
new york
independent film
murder
friendship
teenager
suspense

Word cloud of genres that have caused losses



Important parts of the Code:

```
---
title: "Building a Movie Recommendation Engine"
author: "Amaranath Balaji Prithivirajan",”Pranali Bhosale”,”Pranita Deshpande”,”Rasagnya Reddy Avala”
---

#IMDb and Tomatometer readings
C <- mean(movies$vote_average)
m <- quantile(movies$vote_count, 0.75)

movies$wtd_rating <- (movies$vote_average*movies$vote_count + C*m)/(movies$vote_count + m)

movies %>% top_n(20, wt=wtd_rating) %>%
  ggplot(aes(x=reorder(title, wtd_rating), y=wtd_rating)) +
  geom_bar(stat='identity', fill="blue") + coord_flip(y=c(0,10)) +
  labs(x="", y="Weighted Rating") +
  geom_text(aes(label=round(wtd_rating, 2)), hjust=-0.1, size=3) +
  scale_y_continuous(breaks=seq(0, 10, by=1)) +
  geom_text(aes(label=paste("Votes:", vote_count, "Vote Average:", vote_average)), y=2.3, size=3,
col="yellow")

#simple recommender
genres250 <- left_join(genres250, movies %>% select(id, wtd_rating), by="id")

Genre_lang_recommend <- function(Genre, Language="en") {
  genres250 %>% filter(original_language==Language & genres==Genre) %>%
    arrange(desc(wtd_rating)) %>% top_n(5)}

wav<-movies %>% top_n(20, wt=wtd_rating)
wavg<- movies %>% top_n(20, wt=wtd_rating) %>%
  ggplot(aes(x=reorder(title, wtd_rating), y=wtd_rating)) +
  geom_bar(stat='identity', fill="blue") + coord_flip(y=c(0,10)) +
  labs(x="", y="Weighted Rating") +
  geom_text(aes(label=round(wtd_rating, 2)), hjust=-0.1, size=3) +
  scale_y_continuous(breaks=seq(0, 10, by=1)) +
  geom_text(aes(label=paste("Votes:", vote_count, "Vote Average:", vote_average)), y=2.3, size=3,
col="yellow")

#Recommender based on sentiment of plot
sentiments_plot <- tidy_plots%>%
  inner_join(get_sentiments("nrc"), by=c("word"="word")) %>%
  count(id, title, word, sentiment, sort=TRUE)
sentiment_recommend <- function(feeling){
  sentiments_plot %>%
```

```
      filter(sentiment ==  feeling) %>%
      count(id, title, sentiment, sort = TRUE) %>%
      group_by(id, title) %>%
      dplyr::select(id, title, sentiment,n)%>%
      top_n(10) %>%
      print()
}

***#Collaborative filtering***
#loading the ratings of movies
ratings_small <- read_csv( "ratings_small.csv", na="NA", col_types =
                        cols(
                          userId = col_character(),
                          movieId = col_character(),
                          rating = col_double(),
                          timestamp = col_double()
                        ))

#loading the movie names and movie ids
movie_names <- read_csv("movies_metadata.csv", na="NA", col_types =
                      cols_only(
                        id = col_character(),
                        title = col_character()
                      ))

#filtering the ratings dataset to include only the movies which we have in our TMDB dataset
ratings_small<-ratings_small%>%filter(movieId %in% movie_names$id)

#Formatting the timestap variable
ratings_small$timestamp <- as.POSIXct(ratings_small$timestamp, tz="UTC", origin='1970-01-01')

#preprocessing to create a rating matrix with users as the rows and with movies as columns and
removing timestamp
rating_mat <- ratings_small %>% select(-timestamp) %>%
  spread(movieId, rating) %>%
  remove_rownames %>%
  column_to_rownames(var="userId")

#converting the data to matrix
rating_mat <- as.matrix(rating_mat)

dimnames(rating_mat) <- list(user= rownames(rating_mat), item = colnames(rating_mat))

#creating a "realRatingMatrix" to create a collaborative filtering model
```

```
movies_matrix <- as(rating_mat, "realRatingMatrix")
class(movies_matrix)

#threshold for getting movies with only a minimum number of ratings as 50- helps us to get
#data that have some validity
movie_ratings <- movies_matrix[rowCounts(movies_matrix) > 50,
                      colCounts(movies_matrix) > 50]

#sampling data to create training and testing data
sampled_data<- sample(x = c(TRUE, FALSE),
                size = nrow(movie_ratings),
                replace = TRUE,
                prob = c(0.8, 0.2))
training_data <- movie_ratings[sampled_data, ]
testing_data <- movie_ratings[!sampled_data, ]

#using the default settings to carry out Item Based Collaborative Filter
recommendation_system <- recommenderRegistry$get_entries(dataType ="realRatingMatrix")
#recommendation_system$IBCF_realRatingMatrix$parameters
recommen_model <- Recommender(data = training_data,
                      method = "IBCF",
                      parameter = list(k = 30))
#recommen_model
#class(recommen_model)



#items to recommend to each user
top_recommendations <- 10
predicted_recommendations <- predict(object = recommen_model,
                           newdata = testing_data,
                           n = top_recommendations)
predicted_recommendations
```

 # Define UI for application that plots the graph

```
ui <- dashboardPage(skin = "black",
              dashboardHeader(title = "Movie recommendation system",titleWidth =250),

              dashboardSidebar(
```

```
                selectInput(inputId = "plot_type", label=h4("Select Parameter :"), choices = c("Select
type","Weighted Rating","Simple Recommendations","Recommendations based on Similarity", "Content
based recommendations", "Collaborative filtering") ),
                sidebarMenu(
                  menuItem("Dashboard", tabName = "dashboard", icon = icon("database")),
                  menuItem("Graphs", tabName = "dashboard2", icon = icon("line-chart")),
                  tags$style(".mystyle{color=black;}"))
                # ,
                #
                #
                # column(12, h2(" Inputs"))

                ),


            dashboardBody(
            #    tags$head(
            #      tags$link(rel="stylesheet", type="text/css", href="custom.css")
            # )
                tabItems(
                  tabItem(tabName = "dashboard",
                          fluidRow(
                            valueBox(4803, h4("Movies"), icon = icon("film")),
                            valueBox(36194, h4("Keywords"), icon = icon("tv")),
                            valueBox(12160, h4("genres"), icon = icon("video"))),
                          fluidRow(
                            h1("Discover your movie in a few clicks!"),
                            tags$img(src='movie.jpg', height='500', width='1100', align ='center')
                          ),tags$head(tags$style(HTML('.content-wrapper {background-color:black;}')))
                  ),
                  tabItem(tabName = "dashboard2",
                    fluidRow(box(title = "Select input",background =
"aqua",uiOutput("inputwidget")),height = 2),
                          fluidRow(
                            plotOutput("graph1" , height=500 , width = 1000)),
                    tags$head(tags$style(HTML('.content-wrapper {background-color:black;}')))
                  )
                )))

server <- function(input, output){
  output$inputwidget<- renderUI({
    if(input$plot_type == "Recommendation type"){

    }
```

```
      else if(input$plot_type == "Weighted Rating"){
      selectInput(inputId = "select", "select",choices = c("Top","Bottom"))}
      else if(input$plot_type == "Simple Recommendations"){
         selectInput(inputId = "select", "select",choices = c("Action", "Animation",
         "Crime","Drama", "Fantasy","Horror", "Mystery","Science Fiction","War",
         "Adventure","Comedy","Documentary","Family","History","Music","Romance",
"Thriller","Western"))}
      else if(input$plot_type == "Content based recommendations"){
         selectInput(inputId = "select", "select",choices = c("Avatar", "The Dark Knight",
         "Inside Out","The Godfather", "Fight Club","Spirited Away", "The Shining","Inception","Star
Wars",
         "Titanic"))}
      else if(input$plot_type == "Recommendations based on Similarity"){
         selectInput(inputId = "select", "select",choices = c("Positive","Surprise","Joy",
                                          "Anger","Disgust","Fear","Negative"))}
      else if(input$plot_type == "Collaborative filtering"){
         selectInput(inputId = "select", "select",choices = c("User1","User2","User3",
                                          "User4","User5"))}
   })

   output$graph1 <- renderPlot({
      if (input$plot_type == "Weighted Rating")
      {if(input$select =="Top"){
         movies %>% top_n(15, wt=wtd_rating) %>%
            ggplot(aes(x=reorder(title, wtd_rating), y=wtd_rating)) +
            geom_bar(stat='identity', fill="deepskyblue2") + coord_flip(y=c(0,10)) +
            labs(x="Top Movies", y="Weighted Rating") +
            geom_text(aes(label=round(wtd_rating, 2)), hjust=-0.1, size=4) +
            scale_y_continuous(breaks=seq(0, 10, by=1)) +
            geom_text(aes(label=paste("Votes:", vote_count, "Vote Average:", vote_average)), y=2.3,
size=3, col="white")
      }
         else if(input$select =="Bottom"){
            movies %>% top_n(-15, wt=wtd_rating) %>%
            ggplot(aes(x=reorder(title, wtd_rating), y=wtd_rating)) +
            geom_col(fill="deepskyblue2") + coord_flip(y=c(0,9)) +
            labs(x="Bottom Movies", y="IMDB Weighted Rating") +
            geom_text(aes(label=round(wtd_rating, 2)), hjust=-0.1, size=4) +
            scale_y_continuous(breaks=seq(0, 9, by=1)) +
            geom_text(aes(label=paste("Votes:", vote_count, "Vote Average:", vote_average)),
                  y=2.3, size=3, col="white")
      }
         }
      else if (input$plot_type == "Simple Recommendations")
```

```
{
  if(input$select =="Comedy"){
    genres250 %>% filter(genres=="Comedy") %>%
      arrange(desc(wtd_rating)) %>% top_n(10) %>%
      ggplot(aes(x=title, y=wtd_rating)) +
      geom_col(fill="deepskyblue2") + coord_flip(y=c(0,9)) +
      labs(x="Movies", y="IMDB Weighted Rating") +
      geom_text(aes(label=round(wtd_rating, 2)), hjust=-0.1, size=4) +
      scale_y_continuous(breaks=seq(0, 9, by=1)) +
      geom_text(aes(label=paste("Votes:", vote_count, "Vote Average:", vote_average)),
            y=2.3, size=3, col="white")
  }

  else if(input$select =="Drama"){
    genres250 %>% filter(genres=="Drama") %>%
      arrange(desc(wtd_rating)) %>% top_n(10) %>%
      ggplot(aes(x=title, y=wtd_rating)) +
      geom_col(fill="deepskyblue2") + coord_flip(y=c(0,9)) +
      labs(x="Movies", y="IMDB Weighted Rating") +
      geom_text(aes(label=round(wtd_rating, 2)), hjust=-0.1, size=3) +
      scale_y_continuous(breaks=seq(0, 9, by=1)) +
      geom_text(aes(label=paste("Votes:", vote_count, "Vote Average:", vote_average)),
            y=2.3, size=3, col="white")
  }
  else if(input$select =="Action"){
    genres250 %>% filter(genres=="Action") %>%
      arrange(desc(wtd_rating)) %>% top_n(10) %>%
      ggplot(aes(x=title, y=wtd_rating)) +
      geom_col(fill="deepskyblue2") + coord_flip(y=c(0,9)) +
      labs(x="Movies", y="IMDB Weighted Rating") +
      geom_text(aes(label=round(wtd_rating, 2)), hjust=-0.1, size=3) +
      scale_y_continuous(breaks=seq(0, 9, by=1)) +
      geom_text(aes(label=paste("Votes:", vote_count, "Vote Average:", vote_average)),
            y=2.3, size=3, col="white")
  }
  else if(input$select =="Animation"){
    genres250 %>% filter(genres=="Animation") %>%
      arrange(desc(wtd_rating)) %>% top_n(10) %>%
      ggplot(aes(x=title, y=wtd_rating)) +
      geom_col(fill="deepskyblue2") + coord_flip(y=c(0,9)) +
      labs(x="Movies", y="IMDB Weighted Rating") +
      geom_text(aes(label=round(wtd_rating, 2)), hjust=-0.1, size=3) +
      scale_y_continuous(breaks=seq(0, 9, by=1)) +
      geom_text(aes(label=paste("Votes:", vote_count, "Vote Average:", vote_average)),
```

```
            y=2.3, size=3, col="white")
}
else if(input$select =="Crime"){
   genres250 %>% filter(genres=="Crime") %>%
      arrange(desc(wtd_rating)) %>% top_n(10) %>%
      ggplot(aes(x=title, y=wtd_rating)) +
      geom_col(fill="deepskyblue2") + coord_flip(y=c(0,9)) +
      labs(x="Movies", y="IMDB Weighted Rating") +
      geom_text(aes(label=round(wtd_rating, 2)), hjust=-0.1, size=3) +
      scale_y_continuous(breaks=seq(0, 9, by=1)) +
      geom_text(aes(label=paste("Votes:", vote_count, "Vote Average:", vote_average)),
            y=2.3, size=3, col="white")
}
else if(input$select =="Fantasy"){
   genres250 %>% filter(genres=="Fantasy") %>%
      arrange(desc(wtd_rating)) %>% top_n(10) %>%
      ggplot(aes(x=title, y=wtd_rating)) +
      geom_col(fill="deepskyblue2") + coord_flip(y=c(0,9)) +
      labs(x="Movies", y="IMDB Weighted Rating") +
      geom_text(aes(label=round(wtd_rating, 2)), hjust=-0.1, size=3) +
      scale_y_continuous(breaks=seq(0, 9, by=1)) +
      geom_text(aes(label=paste("Votes:", vote_count, "Vote Average:", vote_average)),
            y=2.3, size=3, col="white")
}
else if(input$select =="Horror"){
   genres250 %>% filter(genres=="Horror") %>%
      arrange(desc(wtd_rating)) %>% top_n(10) %>%
      ggplot(aes(x=title, y=wtd_rating)) +
      geom_col(fill="deepskyblue2") + coord_flip(y=c(0,9)) +
      labs(x="Movies", y="IMDB Weighted Rating") +
      geom_text(aes(label=round(wtd_rating, 2)), hjust=-0.1, size=3) +
      scale_y_continuous(breaks=seq(0, 9, by=1)) +
      geom_text(aes(label=paste("Votes:", vote_count, "Vote Average:", vote_average)),
            y=2.3, size=3, col="white")
}
else if(input$select =="Mystery"){
   genres250 %>% filter(genres=="Mystery") %>%
      arrange(desc(wtd_rating)) %>% top_n(10) %>%
      ggplot(aes(x=title, y=wtd_rating)) +
      geom_col(fill="deepskyblue2") + coord_flip(y=c(0,9)) +
      labs(x="Movies", y="IMDB Weighted Rating") +
      geom_text(aes(label=round(wtd_rating, 2)), hjust=-0.1, size=3) +
      scale_y_continuous(breaks=seq(0, 9, by=1)) +
      geom_text(aes(label=paste("Votes:", vote_count, "Vote Average:", vote_average)),
```

```
            y=2.3, size=3, col="white")
}
else if(input$select =="Science Fiction"){
   genres250 %>% filter(genres=="Science Fiction") %>%
      arrange(desc(wtd_rating)) %>% top_n(10) %>%
      ggplot(aes(x=title, y=wtd_rating)) +
      geom_col(fill="deepskyblue2") + coord_flip(y=c(0,9)) +
      labs(x="Movies", y="IMDB Weighted Rating") +
      geom_text(aes(label=round(wtd_rating, 2)), hjust=-0.1, size=3) +
      scale_y_continuous(breaks=seq(0, 9, by=1)) +
      geom_text(aes(label=paste("Votes:", vote_count, "Vote Average:", vote_average)),
            y=2.3, size=3, col="white")
}
else if(input$select =="War"){
   genres250 %>% filter(genres=="War") %>%
      arrange(desc(wtd_rating)) %>% top_n(10) %>%
      ggplot(aes(x=title, y=wtd_rating)) +
      geom_col(fill="deepskyblue2") + coord_flip(y=c(0,9)) +
      labs(x="Movies", y="IMDB Weighted Rating") +
      geom_text(aes(label=round(wtd_rating, 2)), hjust=-0.1, size=3) +
      scale_y_continuous(breaks=seq(0, 9, by=1)) +
      geom_text(aes(label=paste("Votes:", vote_count, "Vote Average:", vote_average)),
            y=2.3, size=3, col="white")
}
else if(input$select =="Adventure"){
   genres250 %>% filter(genres=="Adventure") %>%
      arrange(desc(wtd_rating)) %>% top_n(10) %>%
      ggplot(aes(x=title, y=wtd_rating)) +
      geom_col(fill="deepskyblue2") + coord_flip(y=c(0,9)) +
      labs(x="Movies", y="IMDB Weighted Rating") +
      geom_text(aes(label=round(wtd_rating, 2)), hjust=-0.1, size=3) +
      scale_y_continuous(breaks=seq(0, 9, by=1)) +
      geom_text(aes(label=paste("Votes:", vote_count, "Vote Average:", vote_average)),
            y=2.3, size=3, col="white")
}
else if(input$select =="Documentary"){
   genres250 %>% filter(genres=="Documentary") %>%
      arrange(desc(wtd_rating)) %>% top_n(10) %>%
      ggplot(aes(x=title, y=wtd_rating)) +
      geom_col(fill="deepskyblue2") + coord_flip(y=c(0,9)) +
      labs(x="Movies", y="IMDB Weighted Rating") +
      geom_text(aes(label=round(wtd_rating, 2)), hjust=-0.1, size=3) +
      scale_y_continuous(breaks=seq(0, 9, by=1)) +
      geom_text(aes(label=paste("Votes:", vote_count, "Vote Average:", vote_average)),
```

```
            y=2.3, size=3, col="white")
}
else if(input$select =="Family"){
   genres250 %>% filter(genres=="Family") %>%
      arrange(desc(wtd_rating)) %>% top_n(10) %>%
      ggplot(aes(x=title, y=wtd_rating)) +
      geom_col(fill="deepskyblue2") + coord_flip(y=c(0,9)) +
      labs(x="Movies", y="IMDB Weighted Rating") +
      geom_text(aes(label=round(wtd_rating, 2)), hjust=-0.1, size=3) +
      scale_y_continuous(breaks=seq(0, 9, by=1)) +
      geom_text(aes(label=paste("Votes:", vote_count, "Vote Average:", vote_average)),
            y=2.3, size=3, col="white")
}
else if(input$select =="History"){
   genres250 %>% filter(genres=="History") %>%
      arrange(desc(wtd_rating)) %>% top_n(10) %>%
      ggplot(aes(x=title, y=wtd_rating)) +
      geom_col(fill="deepskyblue2") + coord_flip(y=c(0,9)) +
      labs(x="Movies", y="IMDB Weighted Rating") +
      geom_text(aes(label=round(wtd_rating, 2)), hjust=-0.1, size=3) +
      scale_y_continuous(breaks=seq(0, 9, by=1)) +
      geom_text(aes(label=paste("Votes:", vote_count, "Vote Average:", vote_average)),
            y=2.3, size=3, col="white")
}
else if(input$select =="Music"){
   genres250 %>% filter(genres=="Music") %>%
      arrange(desc(wtd_rating)) %>% top_n(10) %>%
      ggplot(aes(x=title, y=wtd_rating)) +
      geom_col(fill="deepskyblue2") + coord_flip(y=c(0,9)) +
      labs(x="Movies", y="IMDB Weighted Rating") +
      geom_text(aes(label=round(wtd_rating, 2)), hjust=-0.1, size=3) +
      scale_y_continuous(breaks=seq(0, 9, by=1)) +
      geom_text(aes(label=paste("Votes:", vote_count, "Vote Average:", vote_average)),
            y=2.3, size=3, col="white")
}
else if(input$select =="Romance"){
   genres250 %>% filter(genres=="Romance") %>%
      arrange(desc(wtd_rating)) %>% top_n(10) %>%
      ggplot(aes(x=title, y=wtd_rating)) +
      geom_col(fill="deepskyblue2") + coord_flip(y=c(0,9)) +
      labs(x="Movies", y="IMDB Weighted Rating") +
      geom_text(aes(label=round(wtd_rating, 2)), hjust=-0.1, size=3) +
      scale_y_continuous(breaks=seq(0, 9, by=1)) +
      geom_text(aes(label=paste("Votes:", vote_count, "Vote Average:", vote_average)),
```

```
                y=2.3, size=3, col="white")
  }
  else if(input$select =="Thriller"){
    genres250 %>% filter(genres=="Thriller") %>%
      arrange(desc(wtd_rating)) %>% top_n(10) %>%
      ggplot(aes(x=title, y=wtd_rating)) +
      geom_col(fill="deepskyblue2") + coord_flip(y=c(0,9)) +
      labs(x="Movies", y="IMDB Weighted Rating") +
      geom_text(aes(label=round(wtd_rating, 2)), hjust=-0.1, size=3) +
      scale_y_continuous(breaks=seq(0, 9, by=1)) +
      geom_text(aes(label=paste("Votes:", vote_count, "Vote Average:", vote_average)),
            y=2.3, size=3, col="white")
  }
  else if(input$select =="Western"){
    genres250 %>% filter(genres=="Western") %>%
      arrange(desc(wtd_rating)) %>% top_n(10) %>%
      ggplot(aes(x=title, y=wtd_rating)) +
      geom_col(fill="deepskyblue2") + coord_flip(y=c(0,9)) +
      labs(x="Movies", y="IMDB Weighted Rating") +
      geom_text(aes(label=round(wtd_rating, 2)), hjust=-0.1, size=3) +
      scale_y_continuous(breaks=seq(0, 9, by=1)) +
      geom_text(aes(label=paste("Votes:", vote_count, "Vote Average:", vote_average)),
            y=2.3, size=3, col="white")
  }
  }

else if (input$plot_type == "Content based recommendations")
{
  if(input$select =="Avatar"){
    similar_content_recommendation("Avatar")%>%
      ggplot(aes(x=title, y=wtd_rating)) +
      geom_col(fill="deepskyblue2") + coord_flip() +
      labs(x="Movies", y="weighted rating")

  }
  else if(input$select =="The Dark Knight"){
    similar_content_recommendation("The Dark Knight")%>%
      ggplot(aes(x=title, y=wtd_rating)) +
      geom_col(fill="deepskyblue2") + coord_flip() +
      labs(x="Movies", y="weighted rating")

  }
  else if(input$select =="Inside Out"){
    similar_content_recommendation("Inside Out")%>%
```

```
      ggplot(aes(x=title, y=wtd_rating)) +
      geom_col(fill="deepskyblue2") + coord_flip() +
      labs(x="Movies", y="weighted rating")

}
else if(input$select =="The Godfather"){
   similar_content_recommendation("The Godfather")%>%
      ggplot(aes(x=title, y=wtd_rating)) +
      geom_col(fill="deepskyblue2") + coord_flip() +
      labs(x="Movies", y="weighted rating")

}
else if(input$select =="Fight Club"){
   similar_content_recommendation("Fight Club")%>%
      ggplot(aes(x=title, y=wtd_rating)) +
      geom_col(fill="deepskyblue2") + coord_flip() +
      labs(x="Movies", y="weighted rating")

}
else if(input$select =="Spirited Away"){
   similar_content_recommendation("Spirited Away")%>%
      ggplot(aes(x=title, y=wtd_rating)) +
      geom_col(fill="deepskyblue2") + coord_flip() +
      labs(x="Movies", y="weighted rating")

}
else if(input$select =="The Shining"){
   similar_content_recommendation("The Shining")%>%
      ggplot(aes(x=title, y=wtd_rating)) +
      geom_col(fill="deepskyblue2") + coord_flip() +
      labs(x="Movies", y="weighted rating")

}
else if(input$select =="Inception"){
   similar_content_recommendation("Inception")%>%
      ggplot(aes(x=title, y=wtd_rating)) +
      geom_col(fill="deepskyblue2") + coord_flip() +
      labs(x="Movies", y="weighted rating")

}
else if(input$select =="Star Wars"){
   similar_content_recommendation("Star Wars")%>%
      ggplot(aes(x=title, y=wtd_rating)) +
      geom_col(fill="deepskyblue2") + coord_flip() +
```

```
        labs(x="Movies", y="weighted rating")

  }
  else if(input$select =="Titanic"){
    similar_content_recommendation("Titanic")%>%
      ggplot(aes(x=title, y=wtd_rating)) +
      geom_col(fill="deepskyblue2") + coord_flip() +
      labs(x="Movies", y="weighted rating")

  }
}

else if (input$plot_type == "Recommendations based on Similarity")
{
  if(input$select =="Positive"){
  sent<-sentiment_recommend("positive")
  sent[(1:10),] %>%
    ggplot(aes(x=title, y=n)) +
    geom_col(fill="deepskyblue2") + coord_flip() +
    labs(x="Movies", y="Count")

}
else if(input$select =="Surprise"){
  sent<-sentiment_recommend("surprise")
  sent[(1:10),] %>%
    ggplot(aes(x=title, y=n)) +
    geom_col(fill="deepskyblue2") + coord_flip(y=c(0,9)) +
    labs(x="Movies", y="Count")

}
  else if(input$select =="Joy"){
    sent<-sentiment_recommend("joy")
    sent[(1:10),] %>%
      ggplot(aes(x=title, y=n)) +
      geom_col(fill="deepskyblue2") + coord_flip(y=c(0,9)) +
      labs(x="Movies", y="Count")
  }
  else if(input$select =="Anger"){
    sent<-sentiment_recommend("anger")
    sent[(1:10),] %>%
      ggplot(aes(x=title, y=n)) +
      geom_col(fill="deepskyblue2") + coord_flip(y=c(0,9)) +
      labs(x="Movies", y="Count")
  }
```

```
   else if(input$select =="Fear"){
      sent<-sentiment_recommend("fear")
      sent[(1:10),] %>%
         ggplot(aes(x=title, y=n)) +
         geom_col(fill="deepskyblue2") + coord_flip(y=c(0,9)) +
         labs(x="Movies", y="Count")
   }
   else if(input$select =="Disgust"){
      sent<-sentiment_recommend("disgust")
      sent[(1:10),] %>%
         ggplot(aes(x=title, y=n)) +
         geom_col(fill="deepskyblue2") + coord_flip(y=c(0,9)) +
         labs(x="Movies", y="Count")
   }
   else if(input$select =="Negative"){
      sent<-sentiment_recommend("negative")
      sent[(1:10),] %>%
         ggplot(aes(x=title, y=n)) +
         geom_col(fill="deepskyblue2") + coord_flip(y=c(0,9)) +
         labs(x="Movies", y="Count")
   }
}
else if(input$plot_type == "Collaborative filtering")
{
   if(input$select =="User1"){
      user1 <- predicted_recommendations@items[[1]]
      movies_user1 <- predicted_recommendations@itemLabels[user1]
      movies_user2 <- movies_user1
      for (index in 1:10){
         movies_user2[index] <- as.character(subset(movie_names,
                                    movie_names$id == movies_user1[index])$title)
      }
      movies_user2<-as.tibble(movies_user2)
      movies_user2$n<-(1:10)
      movies_user2 %>%
         ggplot(aes(x=value,y=n)) +
         geom_col(fill="deepskyblue2") + coord_flip() +
         labs(x="Movies", y="Count")

   }
  else if(input$select =="User2"){
      user1 <- predicted_recommendations@items[[2]]
      movies_user1 <- predicted_recommendations@itemLabels[user1]
      movies_user2 <- movies_user1
```

```
    for (index in 1:10){
      movies_user2[index] <- as.character(subset(movie_names,
                                  movie_names$id == movies_user1[index])$title)
    }
    movies_user2<-as.tibble(movies_user2)
    movies_user2$n<-(1:10)
    movies_user2 %>%
      ggplot(aes(x=value,y=n)) +
      geom_col(fill="deepskyblue2") + coord_flip() +
      labs(x="Movies", y="Count")

}
 else if(input$select =="User3"){
    user1 <- predicted_recommendations@items[[3]]
    movies_user1 <- predicted_recommendations@itemLabels[user1]
    movies_user2 <- movies_user1
    for (index in 1:10){
      movies_user2[index] <- as.character(subset(movie_names,
                                  movie_names$id == movies_user1[index])$title)
    }
    movies_user2<-as.tibble(movies_user2)
    movies_user2$n<-(1:10)
    movies_user2 %>%
      ggplot(aes(x=value,y=n)) +
      geom_col(fill="deepskyblue2") + coord_flip() +
      labs(x="Movies", y="Count")

 }
 else if(input$select =="User4"){
    user1 <- predicted_recommendations@items[[4]]
    movies_user1 <- predicted_recommendations@itemLabels[user1]
    movies_user2 <- movies_user1
    for (index in 1:10){
      movies_user2[index] <- as.character(subset(movie_names,
                                  movie_names$id == movies_user1[index])$title)
    }
    movies_user2<-as.tibble(movies_user2)
    movies_user2$n<-(1:10)
    movies_user2 %>%
      ggplot(aes(x=value,y=n)) +
      geom_col(fill="deepskyblue2") + coord_flip() +
      labs(x="Movies", y="Count")

 }
```

```
        else if(input$select =="User5"){
          user1 <- predicted_recommendations@items[[5]]
          movies_user1 <- predicted_recommendations@itemLabels[user1]
          movies_user2 <- movies_user1
          for (index in 1:10){
            movies_user2[index] <- as.character(subset(movie_names,
                                        movie_names$id == movies_user1[index])$title)
          }
          movies_user2<-as.tibble(movies_user2)
          movies_user2$n<-(1:10)
          movies_user2 %>%
            ggplot(aes(x=value,y=n)) +
            geom_col(fill="deepskyblue2") + coord_flip() +
            labs(x="Movies", y="Count")

        }
      }

    })

}
# Run the app ----
shinyApp(ui = ui, server = server)
```