Sentiment Analysis on IMDB Movie Review

Sanjana Rayarala
Artificial Intelligence
Illinois Institute of Technology
srayarala@hawk.iit.edu

Sai Dheeraj Surampally
Artificial Intelligence
Illinois Institute of Technology
ssurampally@hawk.iit.edu

Rahul Narahari
Artificial Intelligence
Illinois Institute of Technology
rnarahari@hawk.iit.edu

1. Abstract

This project employs R for sentiment analysis on IMDB movie reviews. It includes data exploration, cleaning, and visualization, followed by the development and evaluation of machine learning models (Naive Bayes, Logistic Regression, SVM) to showcase its effectiveness in classifying sentiment in movie reviews. The project provides insights into distinguishing features, such as frequent terms and review length, and emphasizes the importance of data preprocessing in natural language processing tasks.

Keywords: Sentiment Analysis, IMDB Dataset, Naïve Bayes model, Support Vector Machine, Logistic Regression Model, Data Preprocessing, Machine Learning Models, Natural Language Processing

2. INTRODUCTION

Sentiment analysis, a burgeoning field within natural language processing, plays a pivotal role in deciphering and categorizing subjective information from textual data. In an era dominated by digital content, understanding public sentiment expressed in reviews, comments, and social media has become increasingly essential. This project focuses on sentiment analysis within the context of IMDB movie reviews, utilizing the powerful capabilities of the R programming language and various libraries. By delving into the rich IMDB dataset, the project aims to not only categorize sentiments but also explore the intricacies of textual patterns that distinguish positive and negative sentiments in the realm of movie critiques.

The initial stages of the project involve a meticulous exploration of the IMDB dataset, assessing its structure, and addressing potential biases and duplicate entries. An understanding of the class distribution of sentiments aids in gauging the balance between positive and negative reviews. Rigorous data cleaning, including the removal of duplicate entries and text preprocessing, becomes imperative to ensure the reliability and quality of the dataset. This sets the stage for subsequent analyses and model training.

The project's focus extends beyond model development to encompass a holistic exploration of the textual features that contribute to sentiment classification. By visualizing sentiment label distributions and employing word clouds to highlight frequently occurring terms, the project seeks to unravel the nuances of language that contribute to the overall sentiment in IMDB movie reviews. The culmination lies in the development and evaluation of machine learning models, including Naive Bayes, Logistic Regression, and Support Vector Machine, with a keen emphasis on the logistic regression model's notable accuracy in classifying sentiments. Through this exploration, the project aims to contribute insights into the dynamics of sentiment analysis methodologies and their real-world applications, particularly the domain of cinematic critic.

Questions

1. How accurately can movie reviews be classified as "positive" or "negative" based on their textual content?

2. What text preprocessing techniques are most effective in preparing the text data for sentiment analysis?

3. Are there specific keywords or phrases that are strongly associated with "positive" sentiment in movie reviews?

4. What linguistic and textual patterns differentiate "positive" reviews from "negative" ones in the dataset?

5. What is the impact of review length on sentiment classification accuracy? Is there an optimal review length for accurate sentiment analysis?

3.PROPOSED METHODOLOGY

The methodology for this sentiment analysis project comprises a comprehensive two-fold approach, encompassing data preprocessing and machine learning model development. In the initial phase, the IMDB dataset undergoes meticulous exploration and cleaning. This includes a thorough examination of its structure, identification, and removal of duplicate entries, and the assessment of class distribution to detect and mitigate potential biases. The text data undergoes preprocessing steps such as converting to lowercase, removing special characters, and eliminating stopwords to enhance the quality of textual information. These preprocessing efforts are pivotal in ensuring a clean and standardized dataset for subsequent analysis.

Following data preprocessing, the project transitions into the machine learning model development phase. Three distinct models are considered: Naive Bayes, Logistic Regression, and Support Vector Machine (SVM). The dataset is split into training and testing sets to facilitate model training and evaluation. The Naive Bayes model is chosen for its simplicity and efficiency in handling text data, while Logistic Regression and SVM provide more sophisticated approaches. The evaluation process involves the generation of confusion matrices to assess model accuracy, and Receiver Operating Characteristic (ROC) curves are employed to visualize the models' performance. The logistic regression model showcases promising accuracy, leading to its selection as the primary sentiment analysis model for IMDB movie reviews. The trained model is then saved for potential future use, where an API is defined to which POST requests are made to classify whether the review falls into Positive/Negative category. This two-phase methodology aims to combine meticulous data preprocessing with advanced machine learning techniques to achieve accurate sentiment analysis on IMDB movie reviews.

4. Problem Statement:

The project addresses the challenge of effectively analyzing and interpreting the vast and growing volume of textual movie reviews on platforms like IMDb. In the realm of digital content where user-generated reviews significantly influence public perception and decision-making, there is a critical need for an automated system that can accurately classify these reviews into relevant sentiment categories. This task involves dealing with the complexities of natural language, which includes understanding context, nuances, and varied expressions of human emotions in text. The goal is to develop a robust sentiment analysis application that can process large datasets of movie reviews, categorizing them into 'positive' or 'negative' sentiments. This process requires meticulous data preprocessing, feature extraction, and the application of machine learning models capable of handling text data. An added dimension of the project is to ensure the model's unbiased performance despite the subjective nature of movie reviews. The final aim is to deploy this model as an accessible API, providing a practical tool for filmmakers, marketers, recommendation systems, and movie enthusiasts to gauge public sentiment. This would not only enhance the understanding of audience reception but also aid in decision-making processes related to the film industry.

Challenges:

Addressing potential biases, ensuring a balanced dataset, and handling any missing or duplicate data.

Optimizing the preprocessing steps for textual data to enhance the quality of input for machine learning models.

5. Data:

5.1. Data Properties:

Dataset Source: IMDB Movie Reviews Dataset, Kaggle ([IMDB Dataset of 50K Movie Reviews (kaggle.com)]())

Data Structure:

The dataset is a structured table with rows and columns.

Variables include "review" for textual content and "sentiment" for sentiment labels.

Data Size:

The dataset comprises a total of 49,528 entries(reviews), each described by 2 attributes(sentiment).

5.2. Data Preprocessing, Cleaning, and Wrangling:

Handling Missing Values:

Identify and handle missing values if present in any of the columns.

Duplicate Entries:

Detect and address any duplicated rows to ensure data integrity.

Text Data Preprocessing:

Convert text to lowercase to standardize the format.

Remove special characters, numbers, and punctuation to clean the text.

Utilize the tm library for additional text processing, including the removal of stop words.

Exploratory Data Analysis (EDA):

Explore the distribution of sentiment labels to understand the balance between positive and negative reviews.

Generate visualizations, such as bar plots and word clouds, to gain insights into the textual patterns.

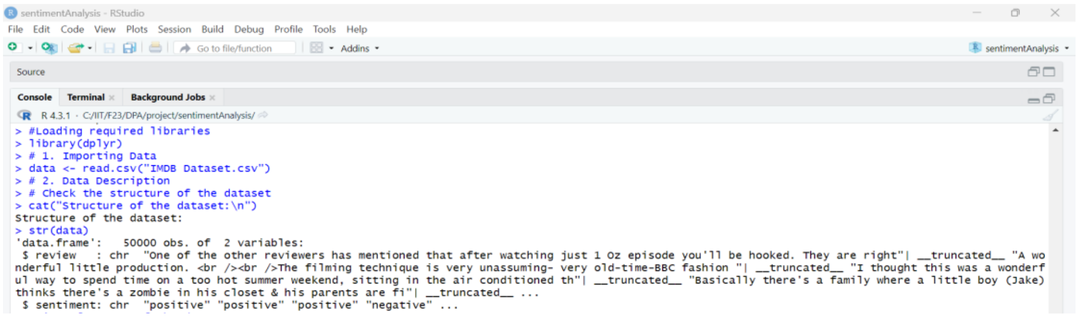
6. EXPLORATORY DATA ANALYSIS:

6.1. Data Description:

- Purpose: To understand the structure and characteristics of the IMDB dataset.

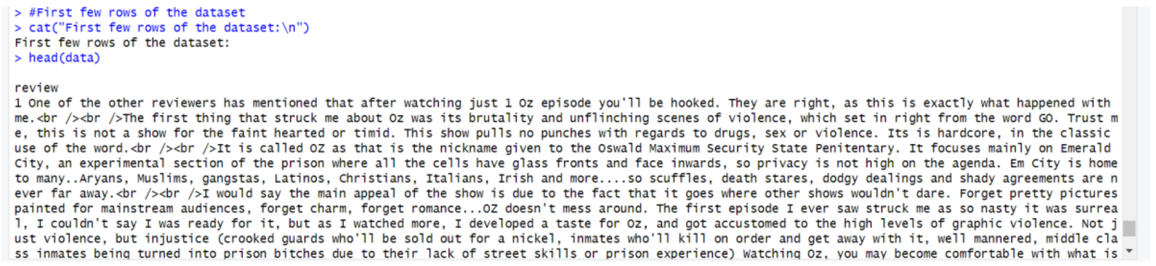
- Actions:

-> Used `str(data)` to explore the dataset's structure and variable types.


sentimentAnalysis - RStudio
File Edit Code View Plots Session Build Debug Profile Tools Help
Go to file/function
Addins
sentimentAnalysis
Source
Console
Terminal
Background Jobs
R 4.3.1 · C:/IIT/F23/DPA/project/sentimentAnalysis/

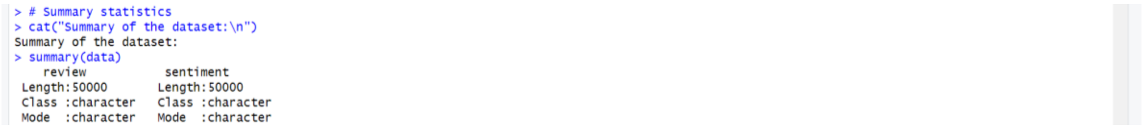
```
> #Loading required libraries
> library(dplyr)
> # 1. Importing Data
> data <- read.csv("IMDB Dataset.csv")
> # 2. Data Description
> # Check the structure of the dataset
> cat("Structure of the dataset:\n")
Structure of the dataset:
> str(data)
'data.frame':	50000 obs. of  2 variables:
 $ review   : chr  "One of the other reviewers has mentioned that after watching just 1 Oz episode you'll be hooked. They are right"| __truncated__ "A wo
nderful little production. <br /><br />The filming technique is very unassuming- very old-time-BBC fashion "| __truncated__ "I thought this was a wonderf
ul way to spend time on a too hot summer weekend, sitting in the air conditioned th"| __truncated__ "Basically there's a family where a little boy (Jake)
thinks there's a zombie in his closet & his parents are fi"| __truncated__ ...
 $ sentiment: chr  "positive" "positive" "positive" "negative" ...
```

->Displayed the first few rows of the dataset using `head(data)` for a quick overview.



```
> #First few rows of the dataset
> cat("First few rows of the dataset:\n")
First few rows of the dataset:
> head(data)

review
1 One of the other reviewers has mentioned that after watching just 1 Oz episode you'll be hooked. They are right, as this is exactly what happened with
me.<br /><br />The first thing that struck me about Oz was its brutality and unflinching scenes of violence, which set in right from the word GO. Trust m
e, this is not a show for the faint hearted or timid. This show pulls no punches with regards to drugs, sex or violence. Its is hardcore, in the classic
use of the word.<br /><br />It is called OZ as that is the nickname given to the Oswald Maximum Security State Penitentary. It focuses mainly on Emerald
City, an experimental section of the prison where all the cells have glass fronts and face inwards, so privacy is not high on the agenda. Em City is home
to many..Aryans, Muslims, gangstas, Latinos, Christians, Italians, Irish and more....so scuffles, death stares, dodgy dealings and shady agreements are n
ever far away.<br /><br />I would say the main appeal of the show is due to the fact that it goes where other shows wouldn't dare. Forget pretty pictures
painted for mainstream audiences, forget charm, forget romance...OZ doesn't mess around. The first episode I ever saw struck me as so nasty it was surrea
l, I couldn't say I was ready for it, but as I watched more, I developed a taste for Oz, and got accustomed to the high levels of graphic violence. Not j
ust violence, but injustice (crooked guards who'll be sold out for a nickel, inmates who'll kill on order and get away with it, well mannered, middle cla
ss inmates being turned into prison bitches due to their lack of street skills or prison experience) Watching Oz, you may become comfortable with what is
```

-> Generated summary statistics with `summary(data)` to understand key statistical measures.



```
> # Summary statistics
> cat("Summary of the dataset:\n")
Summary of the dataset:
> summary(data)
    review           sentiment        
 Length:50000       Length:50000      
 Class :character   Class :character  
 Mode  :character   Mode  :character  
```

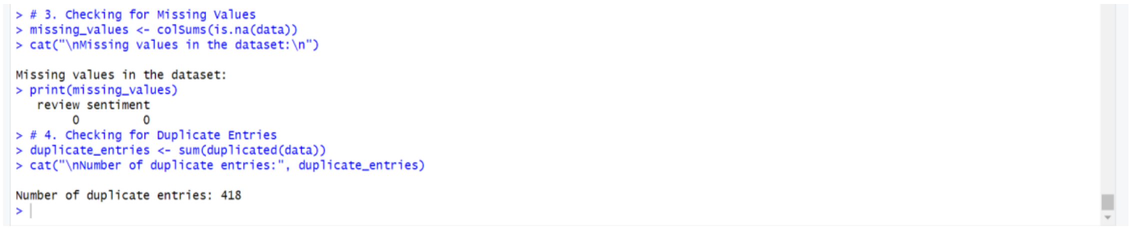
6.2. Checking for Missing Values and Duplicate Entries:

- Purpose: To ensure data quality by identifying and addressing missing and duplicate entries.

- Actions:

->Calculated and printed missing values using `colSums(is.na(data))`.

-> Identified and removed duplicate entries, displaying them using `duplicated(data)` and then `data[!duplicated(data), ]`.



```
> # 3. Checking for Missing Values
> missing_values <- colSums(is.na(data))
> cat("\nMissing values in the dataset:\n")

Missing values in the dataset:
> print(missing_values)
   review sentiment 
        0         0 
> # 4. Checking for Duplicate Entries
> duplicate_entries <- sum(duplicated(data))
> cat("\nNumber of duplicate entries:", duplicate_entries)

Number of duplicate entries: 418
> 
```

No. of duplicate entries were 418.

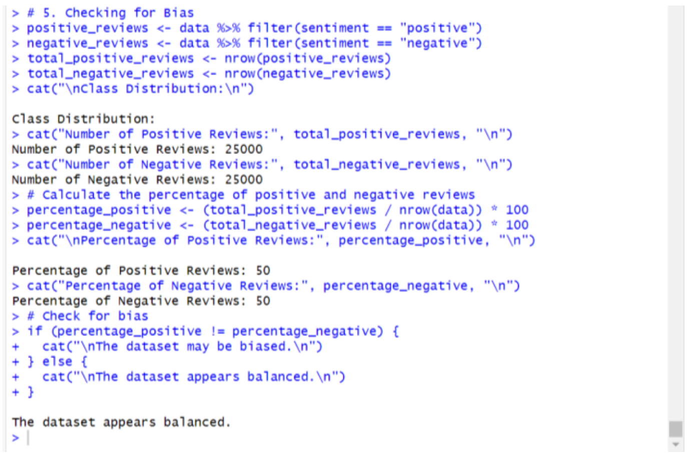
6.3.Checking for Bias:

-Purpose: To assess the distribution of sentiment labels and detect potential bias.

-Actions:

-> Filtered positive and negative reviews and calculated their percentages in the dataset.

-> Checked for bias by comparing the percentages of positive and negative reviews.



```
> # 5. Checking for Bias
> positive_reviews <- data %>% filter(sentiment == "positive")
> negative_reviews <- data %>% filter(sentiment == "negative")
> total_positive_reviews <- nrow(positive_reviews)
> total_negative_reviews <- nrow(negative_reviews)
> cat("\nClass Distribution:\n")

Class Distribution:
> cat("Number of Positive Reviews:", total_positive_reviews, "\n")
Number of Positive Reviews: 25000
> cat("Number of Negative Reviews:", total_negative_reviews, "\n")
Number of Negative Reviews: 25000
> # Calculate the percentage of positive and negative reviews
> percentage_positive <- (total_positive_reviews / nrow(data)) * 100
> percentage_negative <- (total_negative_reviews / nrow(data)) * 100
> cat("\nPercentage of Positive Reviews:", percentage_positive, "\n")

Percentage of Positive Reviews: 50
> cat("Percentage of Negative Reviews:", percentage_negative, "\n")
Percentage of Negative Reviews: 50
> # Check for bias
> if (percentage_positive != percentage_negative) {
+   cat("\nThe dataset may be biased.\n")
+ } else {
+   cat("\nThe dataset appears balanced.\n")
+ }

The dataset appears balanced.
>
```

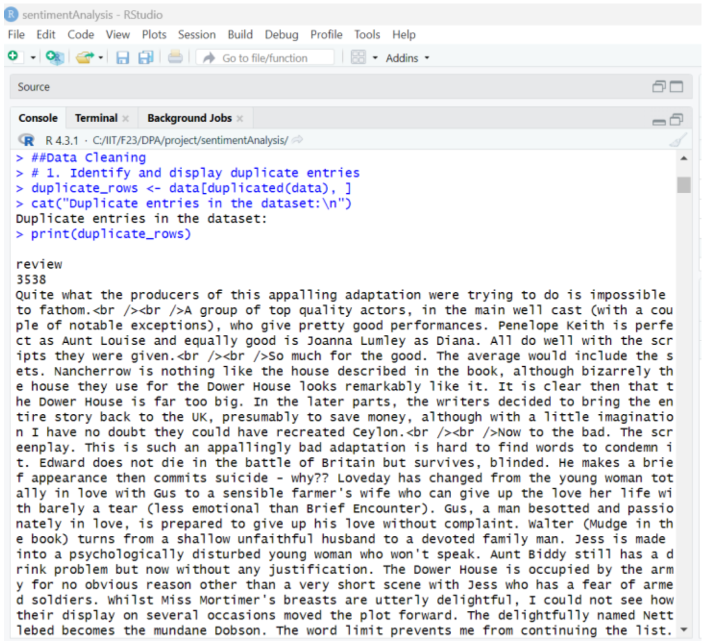
The dataset appears to be balanced.

6.4. Data Cleaning:

- Purpose: To enhance the quality of text data for sentiment analysis.

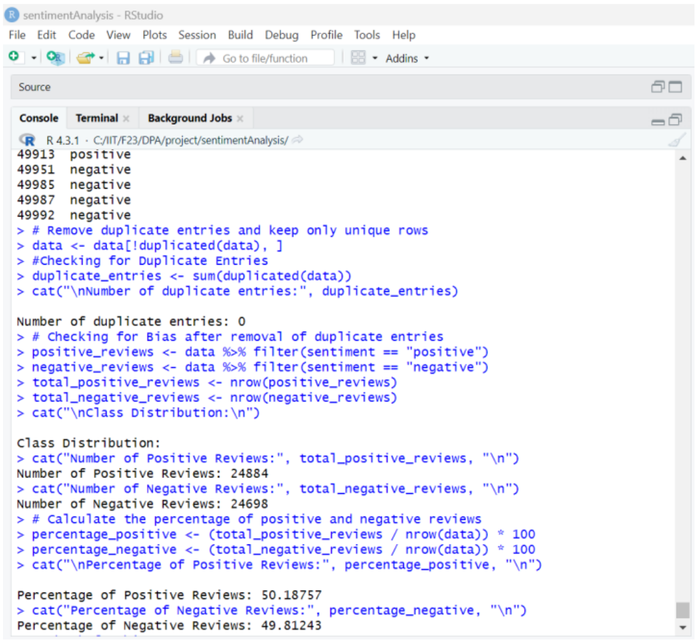
- Actions:

-> Removing duplicate entries.

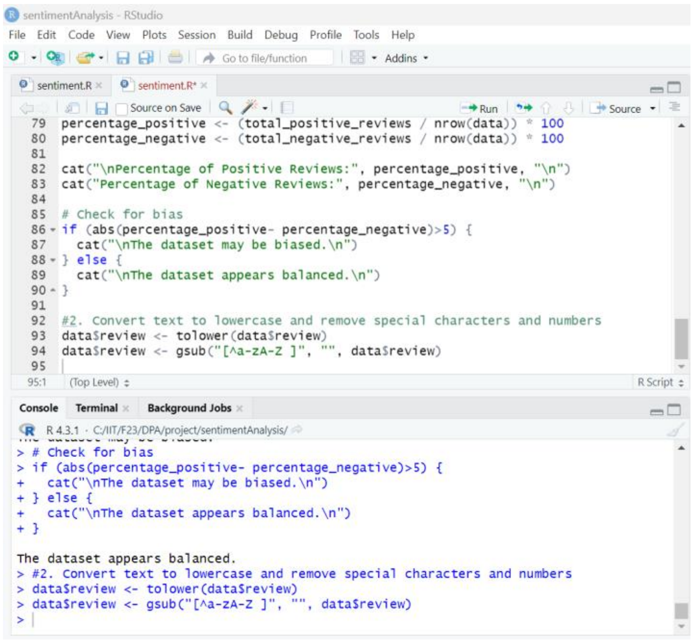


sentimentAnalysis - RStudio
File Edit Code View Plots Session Build Debug Profile Tools Help
Go to file/function
Addins
Source
Console
Terminal
Background Jobs
R 4.3.1 · C:/IIT/F23/DPA/project/sentimentAnalysis/

```
> ##Data Cleaning
> # 1. Identify and display duplicate entries
> duplicate_rows <- data[duplicated(data), ]
> cat("Duplicate entries in the dataset:\n")
Duplicate entries in the dataset:
> print(duplicate_rows)

review
3538
Quite what the producers of this appalling adaptation were trying to do is impossible
to fathom.<br /><br />A group of top quality actors, in the main well cast (with a cou
ple of notable exceptions), who give pretty good performances. Penelope Keith is perfe
ct as Aunt Louise and equally good is Joanna Lumley as Diana. All do well with the scr
ipts they were given.<br /><br />So much for the good. The average would include the s
ets. Nancherrow is nothing like the house described in the book, although bizarrely th
e house they use for the Dower House looks remarkably like it. It is clear then that t
he Dower House is far too big. In the later parts, the writers decided to bring the en
tire story back to the UK, presumably to save money, although with a little imaginatio
n I have no doubt they could have recreated Ceylon.<br /><br />Now to the bad. The scr
eenplay. This is such an appallingly bad adaptation is hard to find words to condemn i
t. Edward does not die in the battle of Britain but survives, blinded. He makes a brie
f appearance then commits suicide - why?? Loveday has changed from the young woman tot
ally in love with Gus to a sensible farmer's wife who can give up the love her life wi
th barely a tear (less emotional than Brief Encounter). Gus, a man besotted and passio
nately in love, is prepared to give up his love without complaint. Walter (Mudge in th
e book) turns from a shallow unfaithful husband to a devoted family man. Jess is made
into a psychologically disturbed young woman who won't speak. Aunt Biddy still has a d
rink problem but now without any justification. The Dower House is occupied by the arm
y for no obvious reason other than a very short scene with Jess who has a fear of arme
d soldiers. Whilst Miss Mortimer's breasts are utterly delightful, I could not see how
their display on several occasions moved the plot forward. The delightfully named Nett
lebed becomes the mundane Dobson. The word limit prevents me from continuing the list.
```


sentimentAnalysis - RStudio
File Edit Code View Plots Session Build Debug Profile Tools Help
Go to file/function
Addins
Source
Console
Terminal
Background Jobs
R 4.3.1 · C:/IIT/F23/DPA/project/sentimentAnalysis/

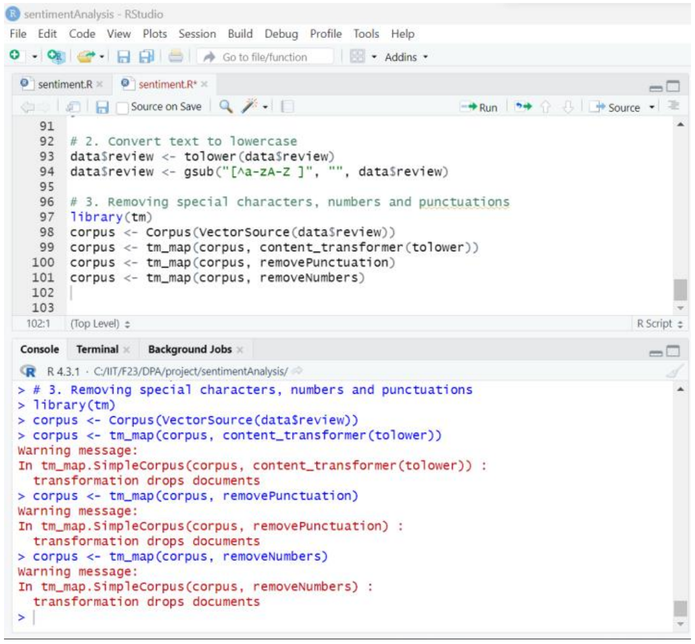
```
49913  positive
49951  negative
49985  negative
49987  negative
49992  negative
> # Remove duplicate entries and keep only unique rows
> data <- data[!duplicated(data), ]
> #Checking for Duplicate Entries
> duplicate_entries <- sum(duplicated(data))
> cat("\nNumber of duplicate entries:", duplicate_entries)

Number of duplicate entries: 0
> # Checking for Bias after removal of duplicate entries
> positive_reviews <- data %>% filter(sentiment == "positive")
> negative_reviews <- data %>% filter(sentiment == "negative")
> total_positive_reviews <- nrow(positive_reviews)
> total_negative_reviews <- nrow(negative_reviews)
> cat("\nClass Distribution:\n")

Class Distribution:
> cat("Number of Positive Reviews:", total_positive_reviews, "\n")
Number of Positive Reviews: 24884
> cat("Number of Negative Reviews:", total_negative_reviews, "\n")
Number of Negative Reviews: 24698
> # Calculate the percentage of positive and negative reviews
> percentage_positive <- (total_positive_reviews / nrow(data)) * 100
> percentage_negative <- (total_negative_reviews / nrow(data)) * 100
> cat("\nPercentage of Positive Reviews:", percentage_positive, "\n")

Percentage of Positive Reviews: 50.18757
> cat("Percentage of Negative Reviews:", percentage_negative, "\n")
Percentage of Negative Reviews: 49.81243
```


sentimentAnalysis - RStudio
File Edit Code View Plots Session Build Debug Profile Tools Help
Go to file/function
Addins
sentiment.R
sentiment.R*
Source on Save
Run
Source
95:1
(Top Level)
R Script
Console
Terminal
Background Jobs
R 4.3.1 · C:/IIT/F23/DPA/project/sentimentAnalysis/

```r
percentage_positive <- (total_positive_reviews / nrow(data)) * 100
percentage_negative <- (total_negative_reviews / nrow(data)) * 100

cat("\nPercentage of Positive Reviews:", percentage_positive, "\n")
cat("Percentage of Negative Reviews:", percentage_negative, "\n")

# Check for bias
if (abs(percentage_positive- percentage_negative)>5) {
  cat("\nThe dataset may be biased.\n")
} else {
  cat("\nThe dataset appears balanced.\n")
}

#2. Convert text to lowercase and remove special characters and numbers
data$review <- tolower(data$review)
data$review <- gsub("[^a-zA-Z ]", "", data$review)

```

```
> # Check for bias
> if (abs(percentage_positive- percentage_negative)>5) {
+   cat("\nThe dataset may be biased.\n")
+ } else {
+   cat("\nThe dataset appears balanced.\n")
+ }

The dataset appears balanced.
> #2. Convert text to lowercase and remove special characters and numbers
> data$review <- tolower(data$review)
> data$review <- gsub("[^a-zA-Z ]", "", data$review)
> 
```

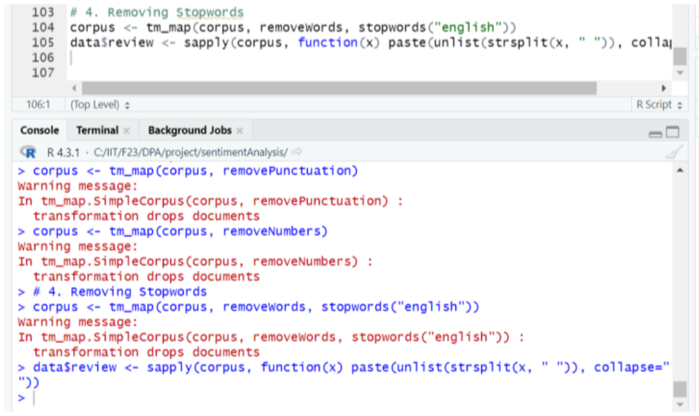
-> Converted text to lowercase and removed special characters and numbers and punctuations.


sentimentAnalysis - RStudio
File Edit Code View Plots Session Build Debug Profile Tools Help
Go to file/function
Addins
sentiment.R
sentiment.R*
Source on Save
Run
Source
102:1
(Top Level)
R Script
Console
Terminal
Background Jobs
R 4.3.1 · C:/IIT/F23/DPA/project/sentimentAnalysis/

```
# 2. Convert text to lowercase
data$review <- tolower(data$review)
data$review <- gsub("[^a-zA-Z ]", "", data$review)

# 3. Removing special characters, numbers and punctuations
library(tm)
corpus <- Corpus(VectorSource(data$review))
corpus <- tm_map(corpus, content_transformer(tolower))
corpus <- tm_map(corpus, removePunctuation)
corpus <- tm_map(corpus, removeNumbers)
```

```
> # 3. Removing special characters, numbers and punctuations
> library(tm)
> corpus <- Corpus(VectorSource(data$review))
> corpus <- tm_map(corpus, content_transformer(tolower))
Warning message:
In tm_map.SimpleCorpus(corpus, content_transformer(tolower)) :
  transformation drops documents
> corpus <- tm_map(corpus, removePunctuation)
Warning message:
In tm_map.SimpleCorpus(corpus, removePunctuation) :
  transformation drops documents
> corpus <- tm_map(corpus, removeNumbers)
Warning message:
In tm_map.SimpleCorpus(corpus, removeNumbers) :
  transformation drops documents
> |
```

-> Used the `tm` library for additional text processing, removing stopwords.



```
# 4. Removing Stopwords
corpus <- tm_map(corpus, removeWords, stopwords("english"))
data$review <- sapply(corpus, function(x) paste(unlist(strsplit(x, " ")), colla
```

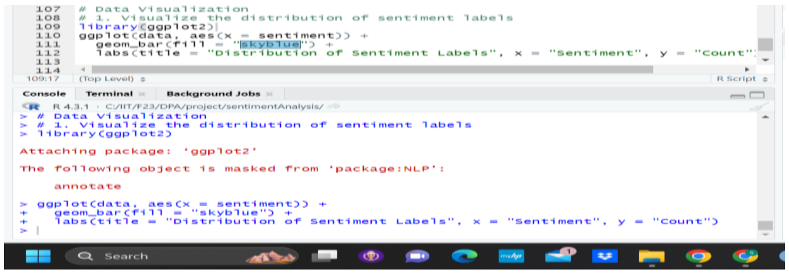
```
> corpus <- tm_map(corpus, removePunctuation)
Warning message:
In tm_map.SimpleCorpus(corpus, removePunctuation) :
  transformation drops documents
> corpus <- tm_map(corpus, removeNumbers)
Warning message:
In tm_map.SimpleCorpus(corpus, removeNumbers) :
  transformation drops documents
> # 4. Removing Stopwords
> corpus <- tm_map(corpus, removeWords, stopwords("english"))
Warning message:
In tm_map.SimpleCorpus(corpus, removeWords, stopwords("english")) :
  transformation drops documents
> data$review <- sapply(corpus, function(x) paste(unlist(strsplit(x, " ")), collapse="
"))
> |
```

6.5. Data Visualization:

- Purpose: To visually represent the distribution of sentiment labels, explore textual patterns.

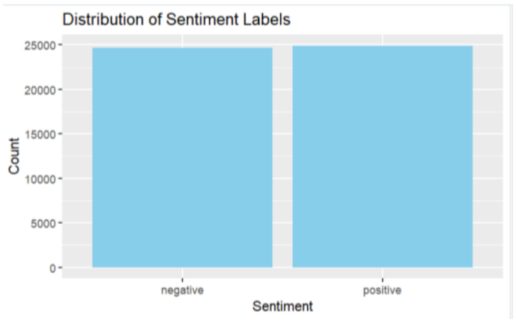
- Actions:

-> Created a bar plot with `ggplot2` to visualize the distribution of sentiment labels.



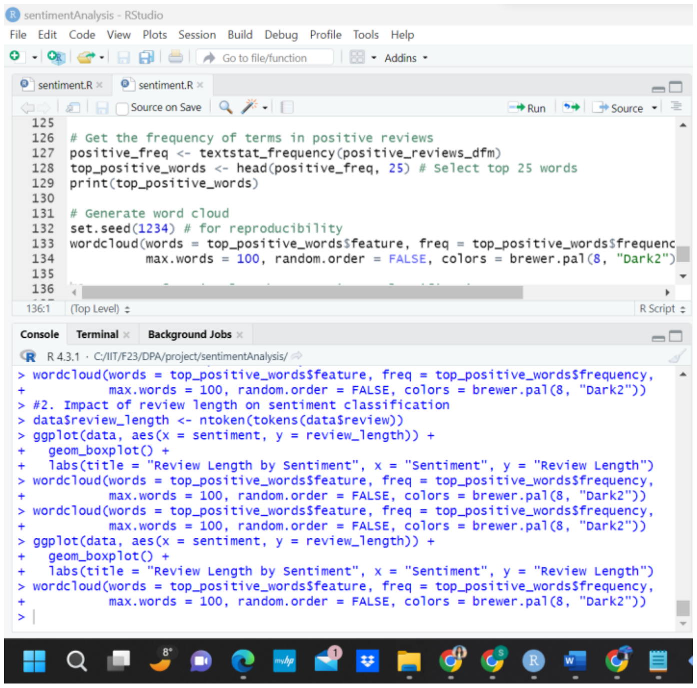
```
# Data Visualization
# 1. Visualize the distribution of sentiment labels
library(ggplot2)
ggplot(data, aes(x = sentiment)) +
  geom_bar(fill = "skyblue") +
  labs(title = "Distribution of Sentiment Labels", x = "Sentiment", y = "Count"
```

```
> # Data Visualization
> # 1. Visualize the distribution of sentiment labels
> library(ggplot2)

Attaching package: 'ggplot2'

The following object is masked from 'package:NLP':

    annotate

> ggplot(data, aes(x = sentiment)) +
+   geom_bar(fill = "skyblue") +
+   labs(title = "Distribution of Sentiment Labels", x = "Sentiment", y = "Count")
> |
```



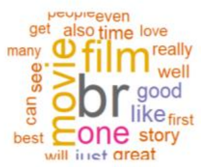
The plot illustrates the distribution of sentiment labels in a dataset. There are two bars, each representing the count of reviews classified into two categories: "negative" and "positive". The heights of the bars appear to be almost identical, suggesting that the dataset is relatively balanced in terms of sentiment distribution.

-> Generated word clouds to highlight frequently occurring terms in positive reviews.


sentimentAnalysis - RStudio
File Edit Code View Plots Session Build Debug Profile Tools Help
Go to file/function
Addins
sentiment.R
sentiment.R
Source on Save
Run
Source
136:1
(Top Level)
R Script
Console
Terminal
Background Jobs
R 4.3.1 · C:/IIT/F23/DPA/project/sentimentAnalysis/

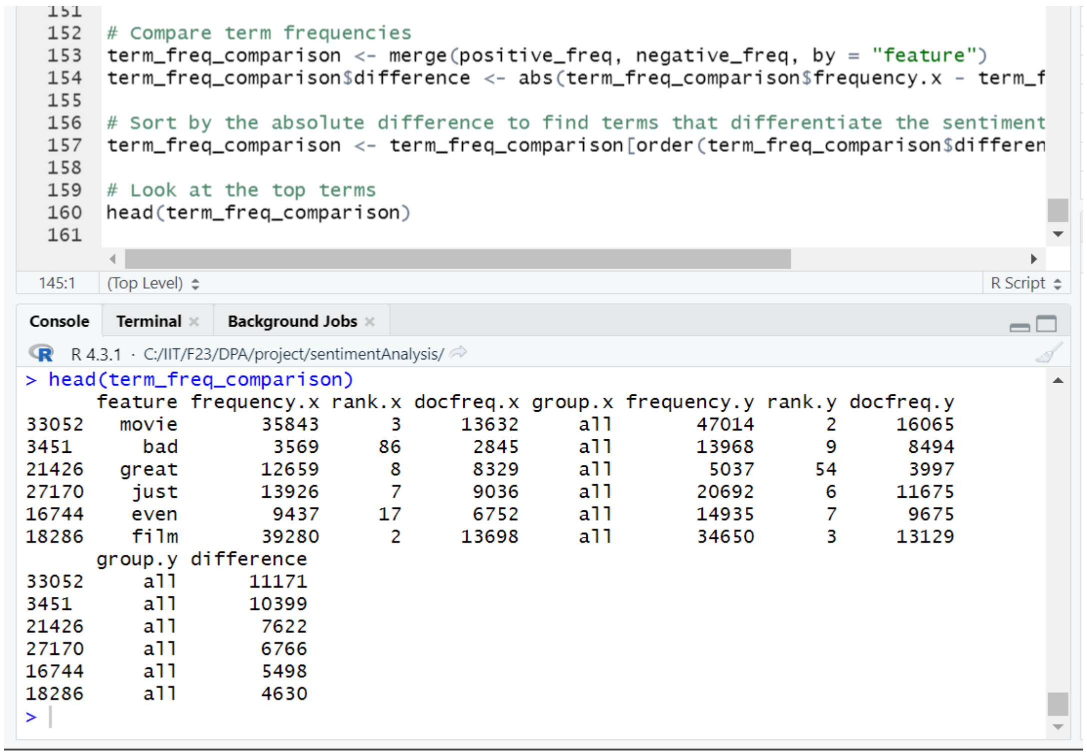
```

# Get the frequency of terms in positive reviews
positive_freq <- textstat_frequency(positive_reviews_dfm)
top_positive_words <- head(positive_freq, 25) # Select top 25 words
print(top_positive_words)

# Generate word cloud
set.seed(1234) # for reproducibility
wordcloud(words = top_positive_words$feature, freq = top_positive_words$frequenc
          max.words = 100, random.order = FALSE, colors = brewer.pal(8, "Dark2")
```

```
> wordcloud(words = top_positive_words$feature, freq = top_positive_words$frequency,
+           max.words = 100, random.order = FALSE, colors = brewer.pal(8, "Dark2"))
> #2. Impact of review length on sentiment classification
> data$review_length <- ntoken(tokens(data$review))
> ggplot(data, aes(x = sentiment, y = review_length)) +
+   geom_boxplot() +
+   labs(title = "Review Length by Sentiment", x = "Sentiment", y = "Review Length")
> wordcloud(words = top_positive_words$feature, freq = top_positive_words$frequency,
+           max.words = 100, random.order = FALSE, colors = brewer.pal(8, "Dark2"))
> wordcloud(words = top_positive_words$feature, freq = top_positive_words$frequency,
+           max.words = 100, random.order = FALSE, colors = brewer.pal(8, "Dark2"))
> ggplot(data, aes(x = sentiment, y = review_length)) +
+   geom_boxplot() +
+   labs(title = "Review Length by Sentiment", x = "Sentiment", y = "Review Length")
> wordcloud(words = top_positive_words$feature, freq = top_positive_words$frequency,
+           max.words = 100, random.order = FALSE, colors = brewer.pal(8, "Dark2"))
> |
```



In this particular word cloud, the most prominent words are "movie", "film", "one", and "story", which suggests these terms are frequently used in the dataset from which the word cloud was generated. This dataset likely pertains to movie reviews or discussions where such terms would be common. Other notable words include "good", "great", "best", "love", and "like", which might indicate positive sentiment in the reviews. The presence of words like "time", "see", "well", and "really" could be part of common phrases used in movie critiques or recommendations.

-> Generate term frequencies



```
# Compare term frequencies
term_freq_comparison <- merge(positive_freq, negative_freq, by = "feature")
term_freq_comparison$difference <- abs(term_freq_comparison$frequency.x - term_f

# Sort by the absolute difference to find terms that differentiate the sentiment
term_freq_comparison <- term_freq_comparison[order(term_freq_comparison$differen

# Look at the top terms
head(term_freq_comparison)
```

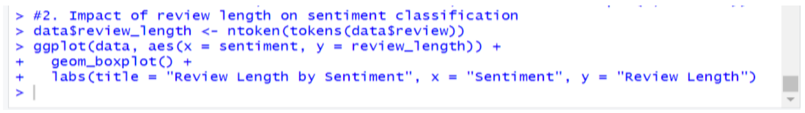
145:1 (Top Level) R Script

Console Terminal Background Jobs

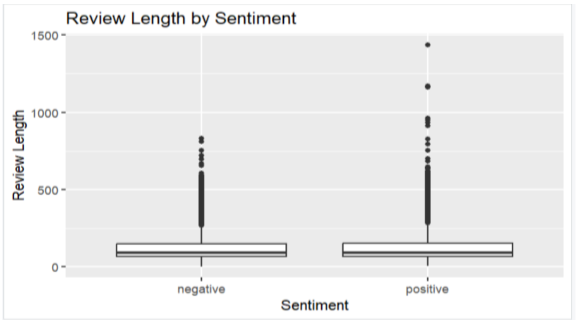
R 4.3.1 · C:/IIT/F23/DPA/project/sentimentAnalysis/

```
> head(term_freq_comparison)
      feature frequency.x rank.x docfreq.x group.x frequency.y rank.y docfreq.y
33052   movie       35843      3     13632     all       47014      2     16065
3451      bad        3569     86      2845     all       13968      9      8494
21426   great       12659      8      8329     all        5037     54      3997
27170    just       13926      7      9036     all       20692      6     11675
16744    even        9437     17      6752     all       14935      7      9675
18286    film       39280      2     13698     all       34650      3     13129
      group.y difference
33052     all      11171
3451      all      10399
21426     all       7622
27170     all       6766
16744     all       5498
18286     all       4630
> 
```

-> Used a box plot to explore the impact of review length on sentiment classification



```
> #2. Impact of review length on sentiment classification
> data$review_length <- ntoken(tokens(data$review))
> ggplot(data, aes(x = sentiment, y = review_length)) +
+   geom_boxplot() +
+   labs(title = "Review Length by Sentiment", x = "Sentiment", y = "Review Length")
> 
```



The box plot displays the distribution of review lengths for two categories of sentiment: negative and positive. Box plots are a standard way of representing data based on a five-number summary: minimum, first quartile (Q1), median, third quartile (Q3), and maximum.

In both categories, the box represents the interquartile range (IQR), which is the range between the first quartile (25th percentile) and the third quartile (75th percentile). The line inside the box indicates the median review length. The "whiskers" extend from the box to the smallest and largest values within 1.5 times the IQR from the quartiles, indicating the range of typical data points. Points beyond the whiskers are considered outliers, which are unusually long or short reviews compared to the rest of the data.

From the plot, we can observe the following:

- Both negative and positive reviews have a similar range of lengths, with the medians also appearing quite similar. This suggests that the sentiment of a review does not necessarily correlate with its length.
- There is a comparable spread of lengths in both categories, as indicated by the similar IQRs.
- There are outliers in both categories, with some reviews being significantly longer than the majority. These are represented by individual points beyond the whiskers.

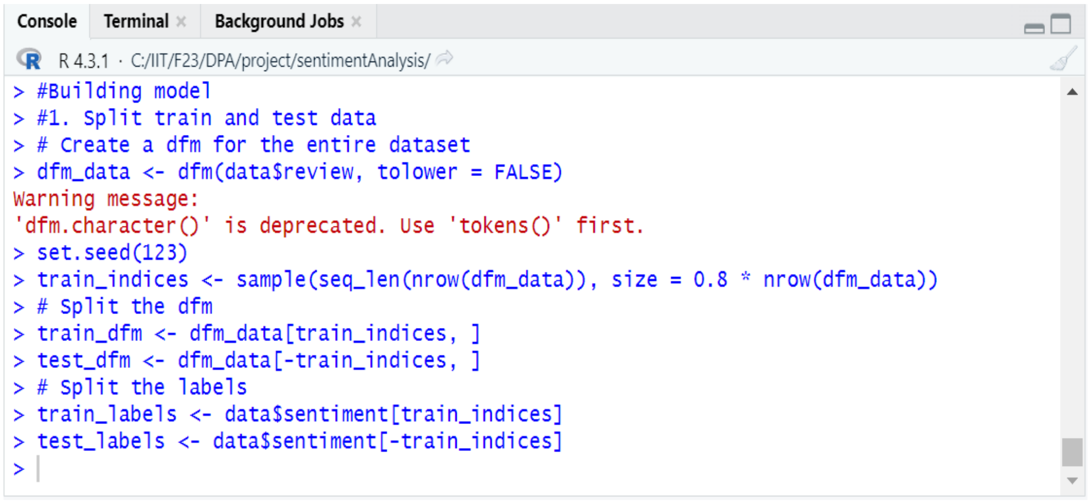
The plot suggests that review length alone may not be a distinguishing feature between positive and negative sentiments in this dataset.

7. MODELING AND ANALYSIS:

7.1. Train-Test Split:

The dataset is split into training and testing sets to facilitate model training and evaluation.

A common split ratio is 80% for training and 20% for testing.


Console Terminal Background Jobs
R 4.3.1 · C:/IIT/F23/DPA/project/sentimentAnalysis/

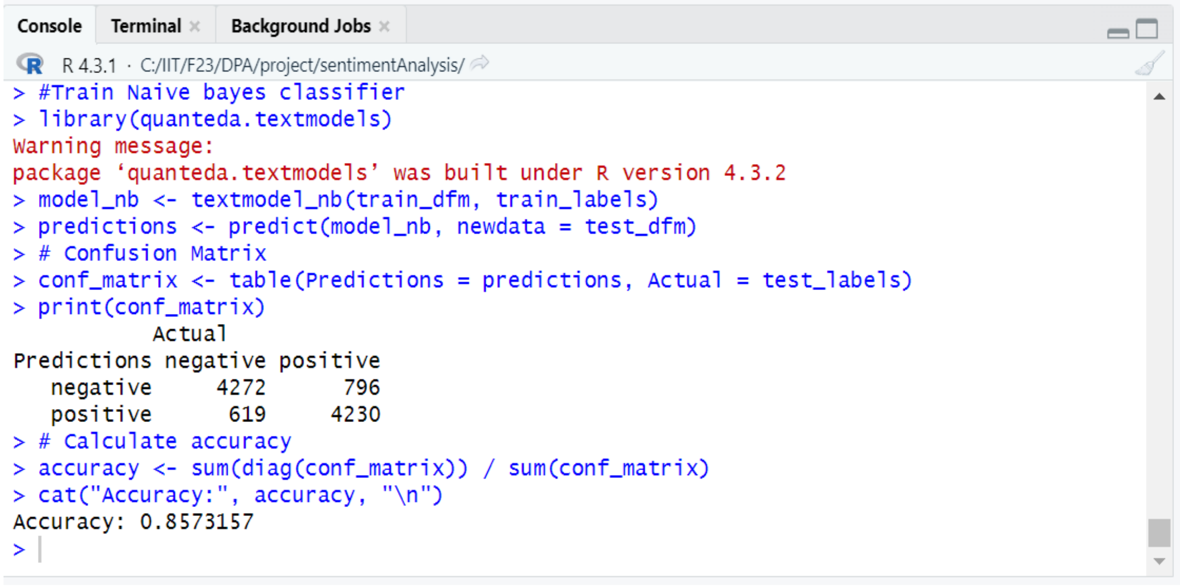
```
> #Building model
> #1. Split train and test data
> # Create a dfm for the entire dataset
> dfm_data <- dfm(data$review, tolower = FALSE)
Warning message:
'dfm.character()' is deprecated. Use 'tokens()' first.
> set.seed(123)
> train_indices <- sample(seq_len(nrow(dfm_data)), size = 0.8 * nrow(dfm_data))
> # Split the dfm
> train_dfm <- dfm_data[train_indices, ]
> test_dfm <- dfm_data[-train_indices, ]
> # Split the labels
> train_labels <- data$sentiment[train_indices]
> test_labels <- data$sentiment[-train_indices]
> 
```

7.2. Modeling:

7.2.1. Naive Bayes:

The textmodel_nb function from the quanteda.textmodels library is used to train the Naive Bayes model.

Naive Bayes is well-suited for text classification tasks due to its simplicity and efficiency.

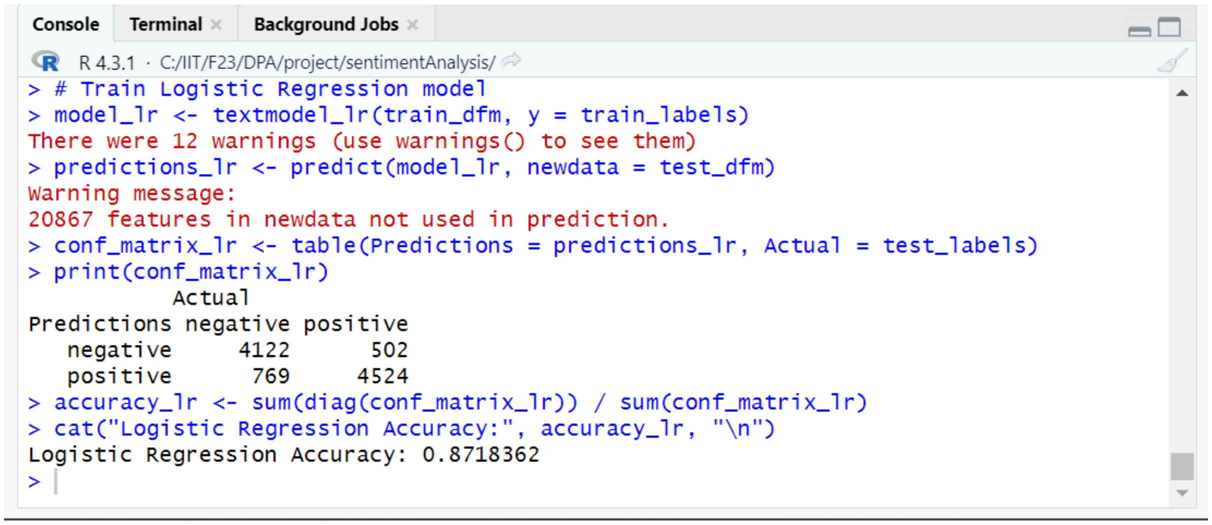

Console Terminal Background Jobs
R 4.3.1 · C:/IIT/F23/DPA/project/sentimentAnalysis/

```
> #Train Naive bayes classifier
> library(quanteda.textmodels)
Warning message:
package ‘quanteda.textmodels’ was built under R version 4.3.2
> model_nb <- textmodel_nb(train_dfm, train_labels)
> predictions <- predict(model_nb, newdata = test_dfm)
> # Confusion Matrix
> conf_matrix <- table(Predictions = predictions, Actual = test_labels)
> print(conf_matrix)
           Actual
Predictions negative positive
   negative     4272      796
   positive      619     4230
> # Calculate accuracy
> accuracy <- sum(diag(conf_matrix)) / sum(conf_matrix)
> cat("Accuracy:", accuracy, "\n")
Accuracy: 0.8573157
> 
```

7.2.2. Logistic Regression:

The textmodel_lr function from the quanteda.textmodels library is used to train the Logistic Regression model.

Logistic Regression is a common choice for binary classification tasks and is suitable for text analysis.

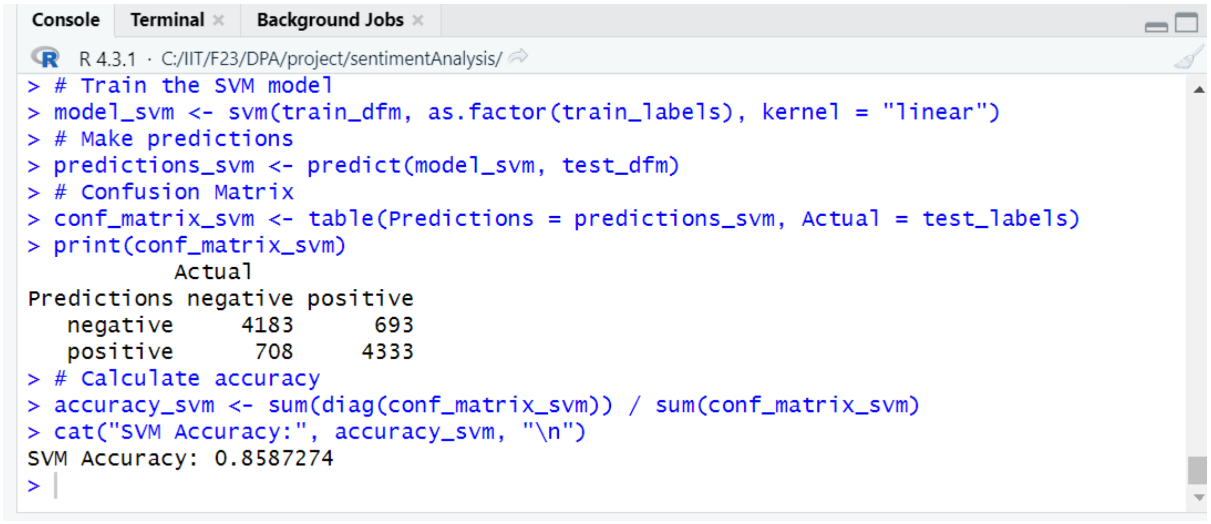


```
> # Train Logistic Regression model
> model_lr <- textmodel_lr(train_dfm, y = train_labels)
There were 12 warnings (use warnings() to see them)
> predictions_lr <- predict(model_lr, newdata = test_dfm)
Warning message:
20867 features in newdata not used in prediction.
> conf_matrix_lr <- table(Predictions = predictions_lr, Actual = test_labels)
> print(conf_matrix_lr)
           Actual
Predictions negative positive
   negative     4122      502
   positive      769     4524
> accuracy_lr <- sum(diag(conf_matrix_lr)) / sum(conf_matrix_lr)
> cat("Logistic Regression Accuracy:", accuracy_lr, "\n")
Logistic Regression Accuracy: 0.8718362
> 
```

7.2.3. Support Vector Machine (SVM):

The svm function from the e1071 library is used to train the SVM model.

The choice of the kernel (linear, radial, etc.) is crucial and may depend on the nature of the data.



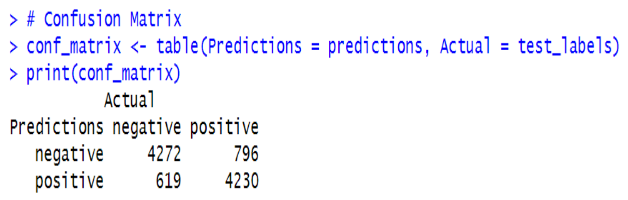
```
> # Train the SVM model
> model_svm <- svm(train_dfm, as.factor(train_labels), kernel = "linear")
> # Make predictions
> predictions_svm <- predict(model_svm, test_dfm)
> # Confusion Matrix
> conf_matrix_svm <- table(Predictions = predictions_svm, Actual = test_labels)
> print(conf_matrix_svm)
           Actual
Predictions negative positive
   negative     4183      693
   positive      708     4333
> # Calculate accuracy
> accuracy_svm <- sum(diag(conf_matrix_svm)) / sum(conf_matrix_svm)
> cat("SVM Accuracy:", accuracy_svm, "\n")
SVM Accuracy: 0.8587274
> 
```

7.3. Analysis:

7.3.1. Confusion Matrix:

Confusion matrices are generated for each model, showing the counts of true positives, true negatives, false positives, and false negatives.

Confusion matrix for Naïve Bayes model:



```
> # Confusion Matrix
> conf_matrix <- table(Predictions = predictions, Actual = test_labels)
> print(conf_matrix)
           Actual
Predictions negative positive
   negative     4272      796
   positive      619     4230
```

The confusion matrix displays a summary of the performance of the Naive Bayes classification model. It shows how well the model has predicted sentiment labels (negative or positive) compared to the actual sentiment labels from the test data (`test_labels`). Here's a breakdown of the matrix:

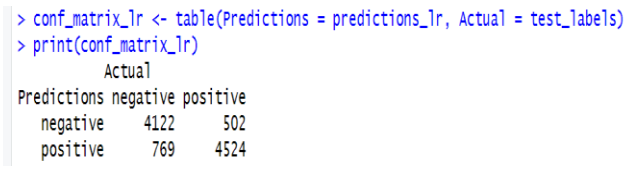
- True Negatives (TN): The model correctly predicted 4272 reviews as negative.

- False Positives (FP): The model incorrectly predicted 796 reviews as positive when they were actually negative.

- False Negatives (FN): The model incorrectly predicted 619 reviews as negative when they were actually positive.

- True Positives (TP): The model correctly predicted 4230 reviews as positive.

Confusion matrix for Logistic Regression model:



```
> conf_matrix_lr <- table(Predictions = predictions_lr, Actual = test_labels)
> print(conf_matrix_lr)
           Actual
Predictions negative positive
   negative     4122      502
   positive      769     4524
```

The provided confusion matrix is a summary of the prediction results from a Logistic Regression model used for classification. Here's the detailed explanation:

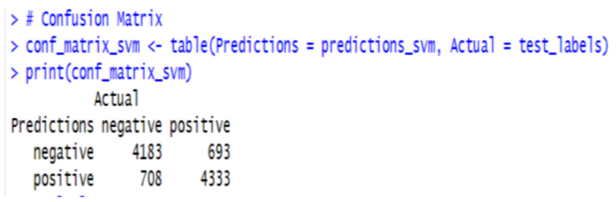
- True Negatives (TN): The model correctly predicted 4122 reviews as negative.

- False Positives (FP): The model incorrectly predicted 502 reviews as positive when they were actually negative.

- False Negatives (FN): The model incorrectly predicted 769 reviews as negative when they were actually positive.

- True Positives (TP): The model correctly predicted 4524 reviews as positive.

Confusion matrix for Support Vector Machine model:



```
> # Confusion Matrix
> conf_matrix_svm <- table(Predictions = predictions_svm, Actual = test_labels)
> print(conf_matrix_svm)
           Actual
Predictions negative positive
   negative     4183      693
   positive      708     4333
```

The confusion matrix presented summarizes the results of a Support Vector Machine (SVM) model used for binary classification. It shows the number of correct and incorrect predictions compared to the actual labels:

- True Negatives (TN): 4183 negative reviews were correctly identified as negative.

- False Positives (FP): 693 negative reviews were incorrectly identified as positive.

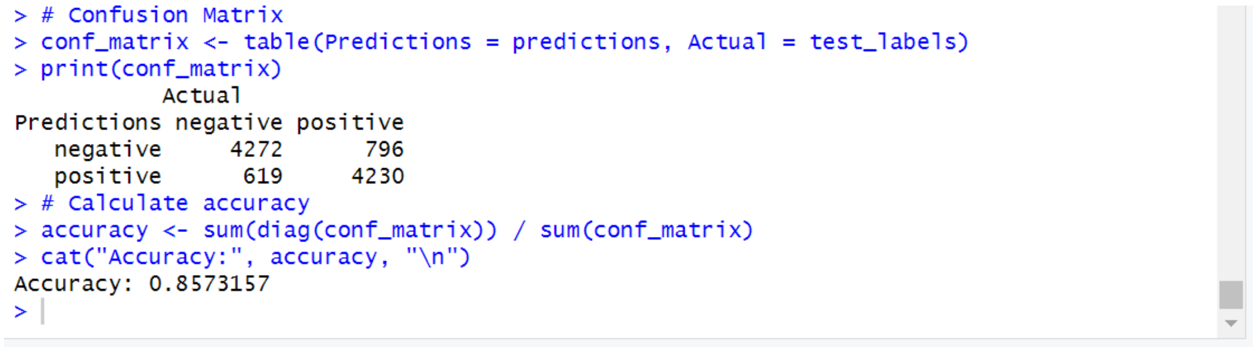
- False Negatives (FN): 708 positive reviews were incorrectly identified as negative.

- True Positives (TP): 4333 positive reviews were correctly identified as positive.

7.3.2. Accuracy Calculation:

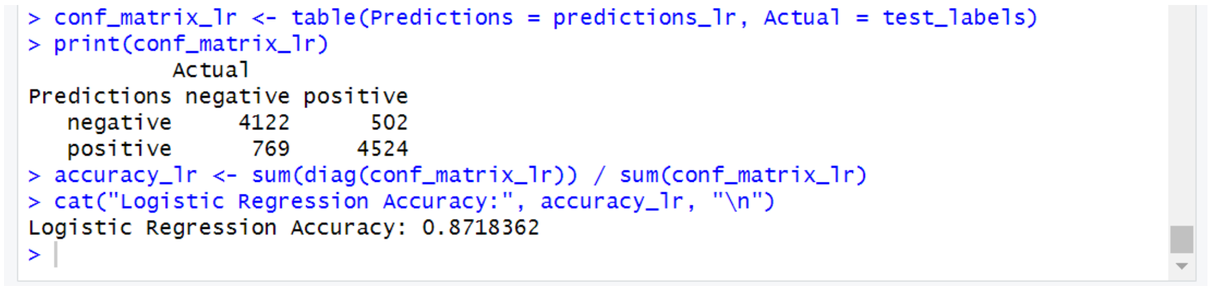
The overall accuracy of each model is calculated by summing the diagonal elements of the confusion matrix and dividing by the total number of instances.

Accuracy of Naïve Bayes model:



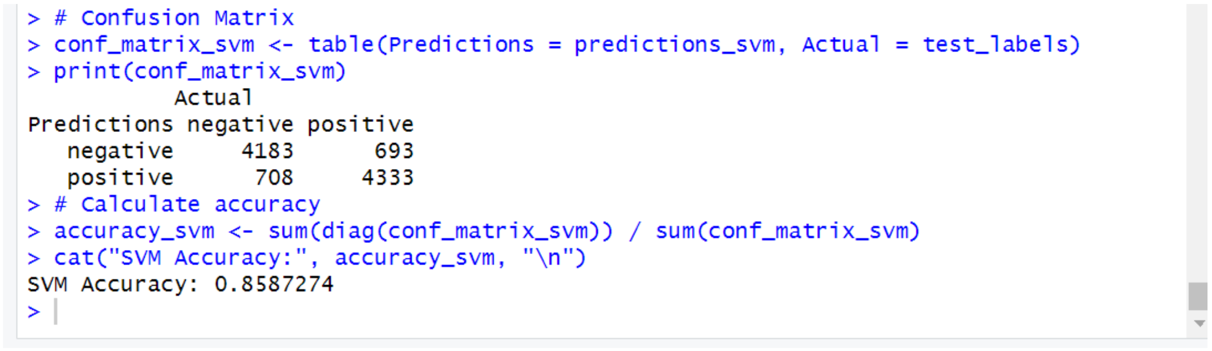
```
> # Confusion Matrix
> conf_matrix <- table(Predictions = predictions, Actual = test_labels)
> print(conf_matrix)
           Actual
Predictions negative positive
   negative     4272      796
   positive      619     4230
> # Calculate accuracy
> accuracy <- sum(diag(conf_matrix)) / sum(conf_matrix)
> cat("Accuracy:", accuracy, "\n")
Accuracy: 0.8573157
> 
```

Accuracy of Logistic Regression model:



```
> conf_matrix_lr <- table(Predictions = predictions_lr, Actual = test_labels)
> print(conf_matrix_lr)
           Actual
Predictions negative positive
   negative     4122      502
   positive      769     4524
> accuracy_lr <- sum(diag(conf_matrix_lr)) / sum(conf_matrix_lr)
> cat("Logistic Regression Accuracy:", accuracy_lr, "\n")
Logistic Regression Accuracy: 0.8718362
> 
```

Accuracy of Support vector machine model:

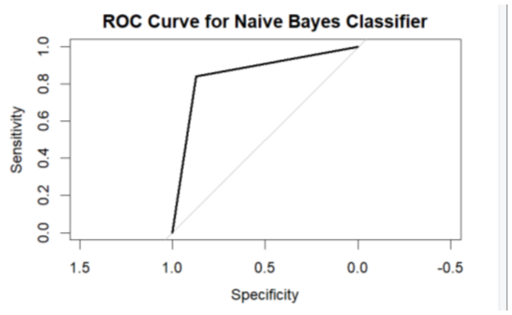


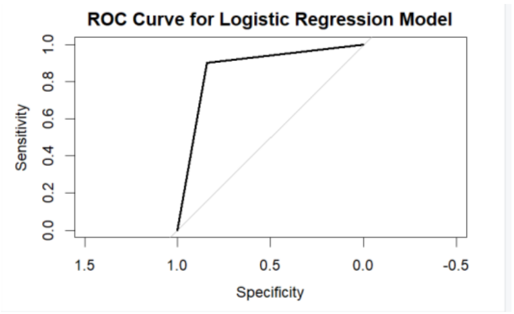
```
> # Confusion Matrix
> conf_matrix_svm <- table(Predictions = predictions_svm, Actual = test_labels)
> print(conf_matrix_svm)
           Actual
Predictions negative positive
   negative     4183      693
   positive      708     4333
> # Calculate accuracy
> accuracy_svm <- sum(diag(conf_matrix_svm)) / sum(conf_matrix_svm)
> cat("SVM Accuracy:", accuracy_svm, "\n")
SVM Accuracy: 0.8587274
> 
```

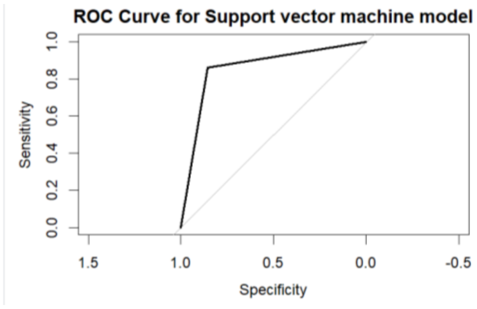
Logistic Regression achieved the highest accuracy of 87.18%.

7.3.3. Receiver Operating Characteristic (ROC) Curve:

ROC curves are plotted for each model, visualizing the trade-off between true positive rate and false positive rate across different classification thresholds.







The Receiver Operating Characteristic (ROC) curves for the Naive Bayes Classifier, Logistic Regression Model, and Support Vector Machine (SVM) model are graphical plots that illustrate the diagnostic ability of binary classifiers as their discrimination threshold is varied.

ROC Curve for Naive Bayes Classifier:

- The plot shows the trade-off between sensitivity (or TPR: True Positive Rate) and specificity (1 - FPR: False Positive Rate) across different thresholds.

- The curve's proximity to the top left corner indicates the model's overall performance; the closer to the top left, the better the model is at distinguishing between the positive and negative classes.

ROC Curve for Logistic Regression Model:

- Similar to the Naive Bayes ROC curve, it also shows sensitivity versus specificity.

- The shape and position of the curve relative to the diagonal line (which represents random chance) show how well the model can separate the classes.

ROC Curve for Support Vector Machine (SVM) Model:

- This curve, too, illustrates the sensitivity vs. specificity for the SVM model.

- The area under the ROC curve (AUC) represents the probability that the classifier will rank a randomly chosen positive instance higher than a randomly chosen negative one.

Comparison:

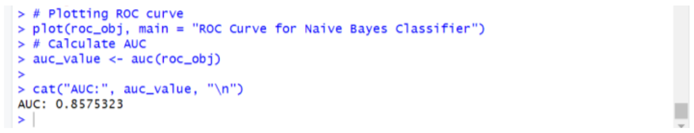
- All three curves are intended to show the effectiveness of the respective models in classifying the sentiment of the reviews.

- A perfect classifier would have a curve that passes through the top left corner (100% sensitivity, 100% specificity). In contrast, a completely ineffective classifier would have a curve that lies along the diagonal line from the bottom left to the top right corner.

7.3.4. Area Under the Curve (AUC):

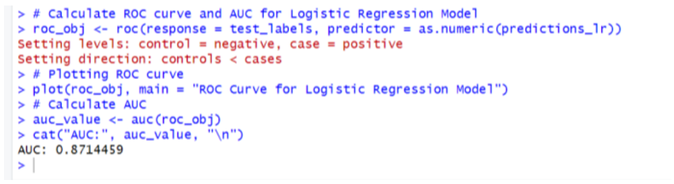
The AUC is calculated for each model's ROC curve, providing a single metric to assess the model's discriminatory power.

AUC for Naïve Bayes model:



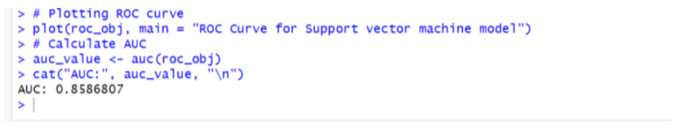
```
> # Plotting ROC curve
> plot(roc_obj, main = "ROC Curve for Naive Bayes Classifier")
> # Calculate AUC
> auc_value <- auc(roc_obj)
>
> cat("AUC:", auc_value, "\n")
AUC: 0.8575323
> |
```

AUC for Logistic Regression model:



```
> # Calculate ROC curve and AUC for Logistic Regression Model
> roc_obj <- roc(response = test_labels, predictor = as.numeric(predictions_lr))
Setting levels: control = negative, case = positive
Setting direction: controls < cases
> # Plotting ROC curve
> plot(roc_obj, main = "ROC Curve for Logistic Regression Model")
> # Calculate AUC
> auc_value <- auc(roc_obj)
> cat("AUC:", auc_value, "\n")
AUC: 0.8714459
> |
```

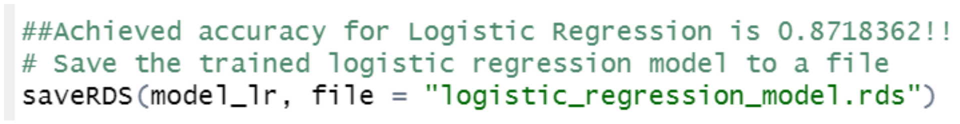
AUC for SVM model:



```
> # Plotting ROC curve
> plot(roc_obj, main = "ROC Curve for Support vector machine model")
> # Calculate AUC
> auc_value <- auc(roc_obj)
> cat("AUC:", auc_value, "\n")
AUC: 0.8586807
> |
```

The Area Under the Curve (AUC) for the Logistic Regression model is higher compared to the Naive Bayes and Support Vector Machine (SVM) models, this indicates that the Logistic Regression model has a better overall performance in terms of distinguishing between the positive and negative classes for this particular sentiment analysis task.

So, saving the Logistic Regression model in Logistic_regression_model.rds



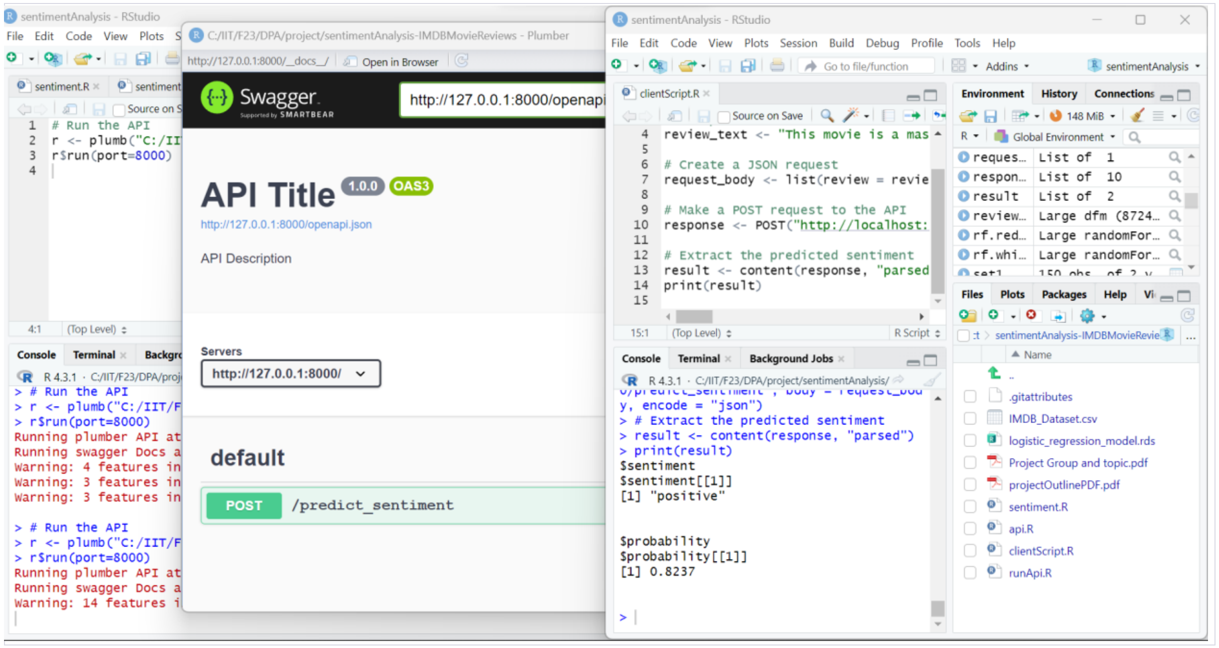
```
##Achieved accuracy for Logistic Regression is 0.8718362!!
# Save the trained logistic regression model to a file
saveRDS(model_lr, file = "logistic_regression_model.rds")
```

8. API Development and Deployment

Developed an API using the plumber package in R. The API is capable of receiving a movie review as input and return the predicted sentiment and its probability.

Followed by testing the API with sample data to ensure proper functionality.

Client script is written using httr to interact with the API. This script will demonstrate how end-users can send a review to the API and receive a sentiment prediction in response.



```
# Run the API
r <- plumb("C:/II
r$run(port=8000)
```

```
> # Run the API
> r <- plumb("C:/IIT/F
> r$run(port=8000)
Running plumber API at
Running swagger Docs a
Warning: 4 features in
Warning: 3 features in
Warning: 3 features in

> # Run the API
> r <- plumb("C:/IIT/F
> r$run(port=8000)
Running plumber API at
Running swagger Docs a
Warning: 14 features i
```

API Title 1.0.0 OAS3

http://127.0.0.1:8000/openapi.json

API Description

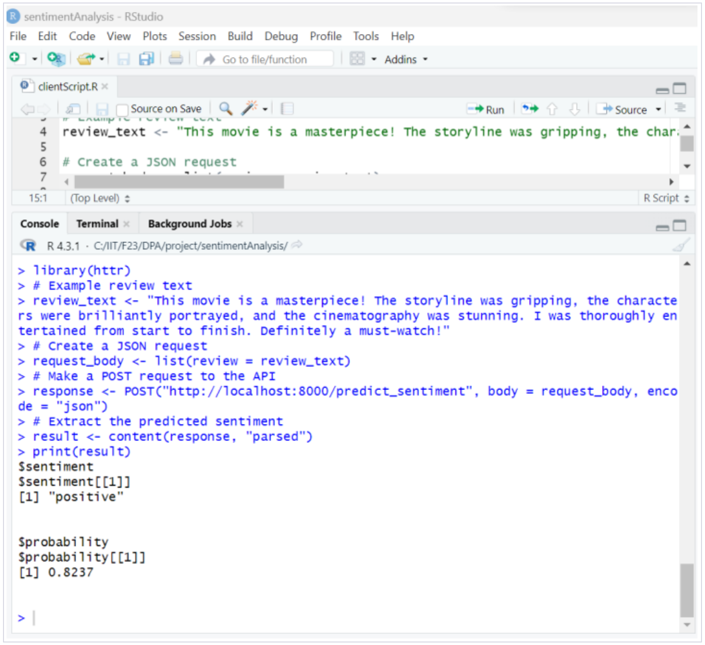
Servers: http://127.0.0.1:8000/

default

POST /predict_sentiment

```
review_text <- "This movie is a mas

# Create a JSON request
request_body <- list(review = revie

# Make a POST request to the API
response <- POST("http://localhost:

# Extract the predicted sentiment
result <- content(response, "parsed
print(result)
```

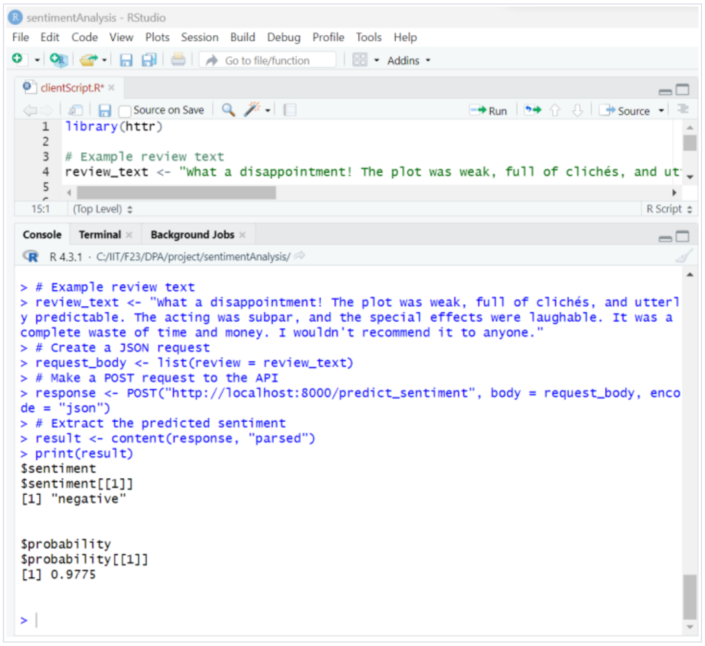
```
y, encode = "json")
> # Extract the predicted sentiment
> result <- content(response, "parsed")
> print(result)
$sentiment
$sentiment[[1]]
[1] "positive"


$probability
$probability[[1]]
[1] 0.8237

>
```



```
review_text <- "This movie is a masterpiece! The storyline was gripping, the char

# Create a JSON request
```

```
> library(httr)
> # Example review text
> review_text <- "This movie is a masterpiece! The storyline was gripping, the characte
rs were brilliantly portrayed, and the cinematography was stunning. I was thoroughly en
tertained from start to finish. Definitely a must-watch!"
> # Create a JSON request
> request_body <- list(review = review_text)
> # Make a POST request to the API
> response <- POST("http://localhost:8000/predict_sentiment", body = request_body, enco
de = "json")
> # Extract the predicted sentiment
> result <- content(response, "parsed")
> print(result)
$sentiment
$sentiment[[1]]
[1] "positive"


$probability
$probability[[1]]
[1] 0.8237

>
```

The review: “This movie is a masterpiece! The storyline was gripping, the characters were brilliantly portrayed, and the cinematography was stunning. I was thoroughly entertained from the start to finish. Definitely a must-watch!”, is classified into “positive” with 82.37% accuracy.



```
library(httr)

# Example review text
review_text <- "What a disappointment! The plot was weak, full of clichés, and ut
```

```
> # Example review text
> review_text <- "What a disappointment! The plot was weak, full of clichés, and utterl
y predictable. The acting was subpar, and the special effects were laughable. It was a
complete waste of time and money. I wouldn't recommend it to anyone."
> # Create a JSON request
> request_body <- list(review = review_text)
> # Make a POST request to the API
> response <- POST("http://localhost:8000/predict_sentiment", body = request_body, enco
de = "json")
> # Extract the predicted sentiment
> result <- content(response, "parsed")
> print(result)
$sentiment
$sentiment[[1]]
[1] "negative"


$probability
$probability[[1]]
[1] 0.9775


> 
```

The review : "What a disappointment! The plot was weak, full of clichés, and utterly predictable. The acting was subpar, and the special effects were laughable. It was a complete waste of time and money. I wouldn't recommend it to anyone.", is classified as "negative" with 97.75% accuracy.

9. Conclusion

The sentiment analysis project, centered on IMDb movie reviews, successfully demonstrates the application of data science and machine learning techniques to the realm of natural language processing. The project achieved its primary objective of developing a robust and efficient tool capable of classifying movie reviews into 'positive' or 'negative' sentiments. This was accomplished through meticulous data preprocessing, which included normalization, cleansing, and feature extraction from the textual data. The use of advanced NLP techniques and machine learning models, specifically Naive Bayes, Logistic Regression, and Support Vector Machine, enabled the accurate interpretation and classification of human sentiments expressed in the reviews.

A significant achievement of this project was the creation of a balanced and unbiased model, capable of handling the subjective and varied nature of movie reviews. The application's effectiveness was further enhanced by its deployment as an API, making it a practical tool for real-world applications. This deployment allows for easy access and integration into various systems, facilitating a deeper understanding of audience sentiments in the film industry.

Through rigorous model evaluation, including accuracy metrics, confusion matrices, ROC curves, and AUC values, the Logistic Regression model emerged as the most effective, striking a balance between accuracy and computational efficiency with 87.18% accuracy. This model was subsequently encapsulated into an API, showcasing the project's full-cycle capability from data analysis to real-world application.

In conclusion, this project not only contributes a valuable tool for sentiment analysis in the film industry but also sets a precedent for similar applications in other domains. It demonstrates the power of machine learning in extracting meaningful insights from complex, unstructured data, and underscores the potential of such technologies in enhancing decision-making processes and understanding public opinions.

REFERENCES

[1] Pang, B., & Lee, L. (2008). Opinion mining and sentiment analysis. *Foundations and Trends® in Information Retrieval, 2*(1–2), 1-135.

[2] Kuhn, M., & Johnson, K. (2013). Applied Predictive Modeling. Springer.

[3] Wickham, H., & Grolemund, G. (2016). R for Data Science: Import, Tidy, Transform, Visualize, and Model Data. O'Reilly Media.

[4] Manning, C. D., Raghavan, P., & Schütze, H. (2008). Introduction to Information Retrieval. Cambridge University Press.

[5] Quinlan, J. R. (1993). C4.5: Programs for Machine Learning. Morgan Kaufmann.

[6] James, G., Witten, D., Hastie, T., & Tibshirani, R. (2013). An Introduction to Statistical Learning: with Applications in R. Springer.

[7] Yu, Hong, et al. "Improved movie review sentiment classification using sentence-level lexicon polari." Expert Systems with Applications 36.2 (2009)

[8] Joshi, Mahesh, et al. "Movie reviews and revenues: An experiment in text regression." Proceedings of the 5th international conference on Weblogs and Social Media. 2011.

[9] Pak, A., & Paroubek, P. (2010). Twitter as a Corpus for Sentiment Analysis and Opinion Mining.