Fake News Detection Using Machine Learning

Introduction

In this report, we evaluate the performance of several machine learning models for the task of fake news detection. The models considered include Naive Bayes (NB), Random Forest (RF), and Support Vector Machine (SVM). The evaluation is based on accuracy rates achieved on a test dataset.

Methodology

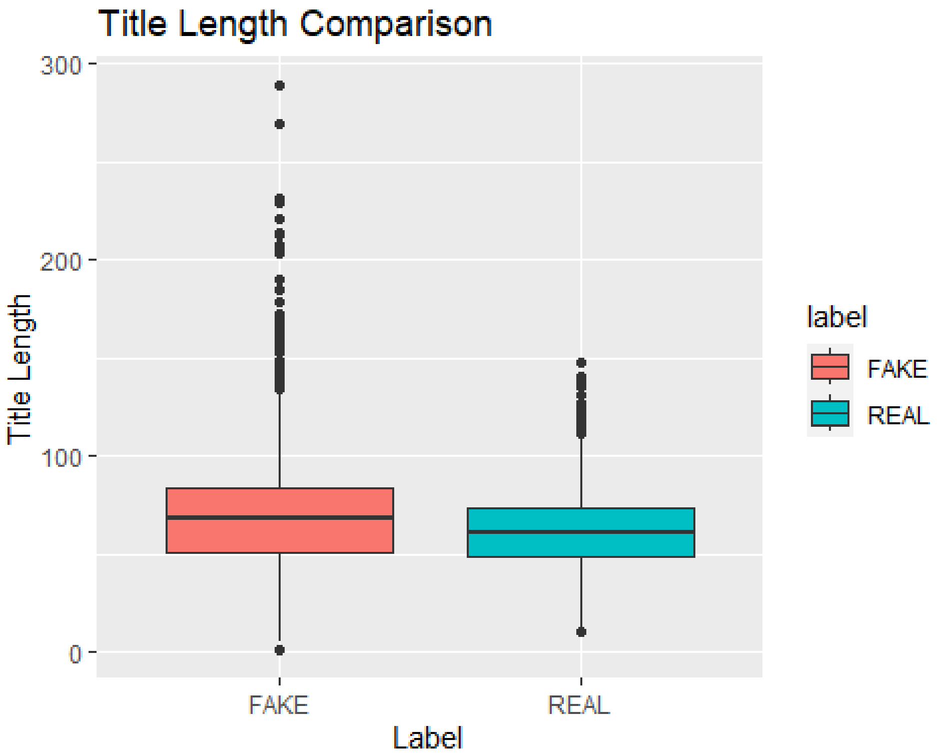
We trained and tested models using features derived from the title, text, and a combination of both. The features include length, word count, presence of numbers, and punctuation count. The dataset was split into 80% training and 20% testing for robust evaluation. We also analyzed the data on the title length and semantic analysis of real and fake texts.

```
library(tidyverse)

library(tidytext)

library(syuzhet)

library(tibble)
library(plyr)

library(tm) #for cleaning

library(caret)

library(reshape2)

library(e1071) #for NB

library(randomForest) #for RF




data = read.csv("fake_or_real_news.csv")
```

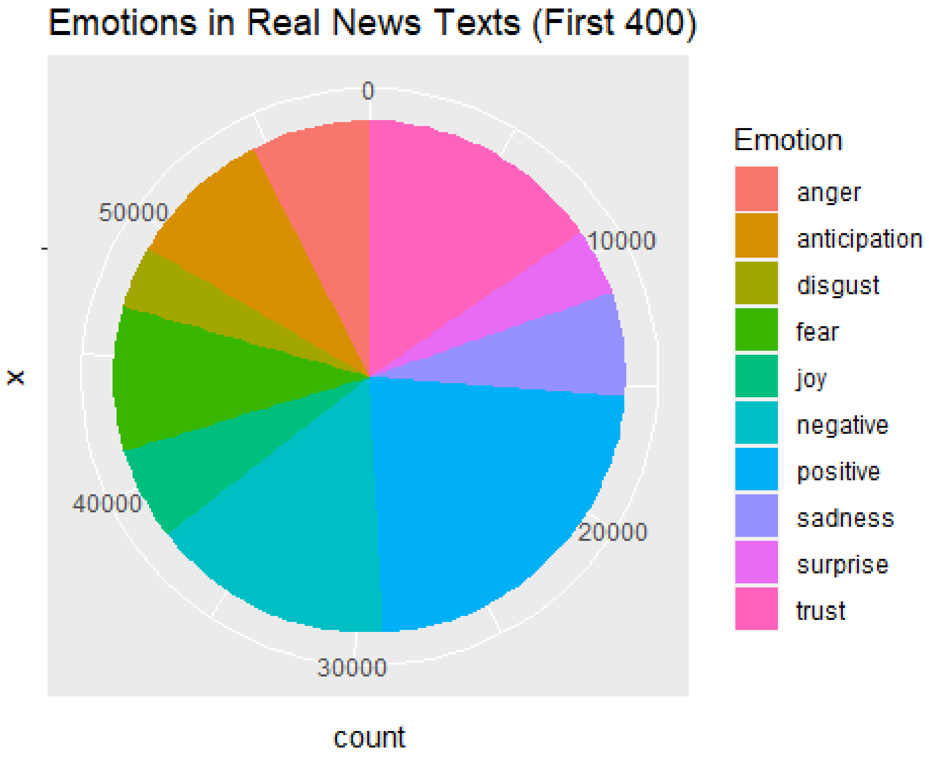
DATA ANALYSIS

```
summary(data)
```

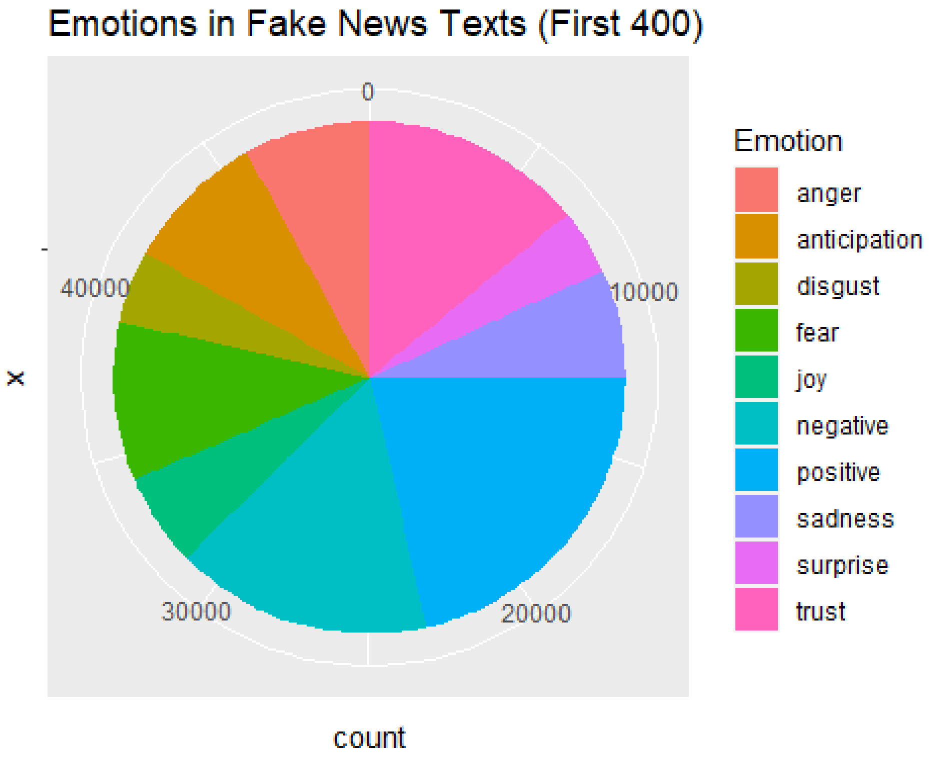
```
##        X              title                text               label          
##  Min.   :    2   Length:6335        Length:6335        Length:6335       
##  1st Qu.: 2674   Class :character   Class :character   Class :character  
##  Median : 5271   Mode  :character   Mode  :character   Mode  :character  
##  Mean   : 5280                                                           
##  3rd Qu.: 7901                                                           
##  Max.   :10557                                                           

summary(data$label)

##    Length     Class      Mode 
##      6335 character character 

 We have 3164 of fake news and 3171 of real news.
This provides a balanced distribution between
the two classes, which is good for training classification models.

#title length comparison
t_test_result = t.test(nchar(data$title[data$label == "FAKE"]),
                        nchar(data$title[data$label == "REAL"]))



print(t_test_result)

## 
##  Welch Two Sample t-test
## 
## data:  nchar(data$title[data$label == "FAKE"]) and
nchar(data$title[data$label == "REAL"])
## t = 13.249, df = 5677.4, p-value < 2.2e-16
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  6.643859 8.951389
## sample estimates:
## mean of x mean of y 
##  69.18078  61.38316 


The extremely small p-value (< 0.05) suggests that there is
a significant difference in the mean title length between fake and real news.
The positive t-value (13.249) and the 95% confidence interval that
does not include 0 (6.643859 to 8.951389) indicate that the mean title length
for fake news
is significantly larger than the mean title length for real news.



ggplot(data, aes(x = label, y = nchar(title), fill = label)) +
  geom_boxplot() +
  labs(title = "Title Length Comparison",
       x = "Label",
       y = "Title Length")
```



```
#sentiment analysis on the first 400 real and fake texts and pie chart
visualization
real_texts = data$text[data$label == "REAL"][1:400]
fake_texts = data$text[data$label == "FAKE"][1:400]


sentiments_real_texts = get_nrc_sentiment(real_texts)
emotions_real_texts = colSums(sentiments_real_texts)
emotions_real_texts_df = data.frame(emotion = names(emotions_real_texts), count
= emotions_real_texts)
head(sentiments_real_texts)

##   anger anticipation disgust fear joy sadness surprise trust negative
positive
## 1     5            5       3    7   5       3        4    16       11
21
## 2     3            7       1    5   4       4        4     6        6
11
## 3     1            1       0    2   0       1        0     1        4
1
## 4    20           20       6   19  10      14       11    31       38
66
## 5     7           13       3    8   4       4        5    15       18
21
## 6     3           13       4    6   8       4        7    21        7
23

ggplot(emotions_real_texts_df, aes(x = "", y = count, fill = emotion)) +
  geom_bar(stat = "identity", width = 1) +
  coord_polar("y") +
  labs(title = "Emotions in Real News Texts (First 400)",
       fill = "Emotion")
```



```
sentiments_fake_texts = get_nrc_sentiment(fake_texts)
emotions_fake_texts = colSums(sentiments_fake_texts)
emotions_fake_texts_df = data.frame(emotion = names(emotions_fake_texts), count
= emotions_fake_texts)
head(sentiments_fake_texts)

##   anger anticipation disgust fear joy sadness surprise trust negative
positive
## 1    38           16      21   43   6      20       14    24       60
40
## 2     5            7       2    1   5       3        7    11       16
16
## 3     4            5       4    3   3       3        3     8        7
14
## 4    36           36      19   48  39      25       23    56       67
93
## 5    10            7       5   16   4      10        5     6       18
13
## 6     1            3       1    1   1       0        0     8        3
13

ggplot(emotions_fake_texts_df, aes(x = "", y = count, fill = emotion)) +
  geom_bar(stat = "identity", width = 1) +
  coord_polar("y") +
  labs(title = "Emotions in Fake News Texts (First 400)",
       fill = "Emotion")
```



```
"Fake news texts tend to have higher counts of anger, disgust.
Real news texts exhibit higher counts of trust and positive sentiments.
The presence of surprise in both fake and real news texts suggests
that both types designed to capture the reader's attention.
Fake news texts seem to evoke a wider range of emotions,
including both positive and negative, which might align with the goal
of capturing attention or generating sensationalism."




NEWS TYPE PREDICTION

#cleaning data
preprocess_corpus = function(text) {

  text = tolower(text)
  text = removePunctuation(text)
  text = removeWords(text, stopwords("en"))
  text = stemDocument(text)
  text = stripWhitespace(text)

  return(text)
}

data$title = sapply(data$title, preprocess_corpus)
data$text = sapply(data$text, preprocess_corpus)
```

```r
data$label = as.factor(data$label)

#train and test data split (80% - training, 20 - testing )
set.seed(123)
splitIndex = createDataPartition(data$label, p = 0.8, list = FALSE)
train_data = data[splitIndex, ]
test_data = data[-splitIndex, ]

#retraining with different features

#length
train_data$title_length = nchar(train_data$title)
test_data$title_length = nchar(test_data$title)

train_data$text_length = nchar(train_data$text)
test_data$text_length = nchar(test_data$text)

#word count
train_data$title_word_count = sapply(strsplit(train_data$title, " "), length)
test_data$title_word_count = sapply(strsplit(test_data$title, " "), length)

train_data$text_word_count = sapply(strsplit(train_data$text, " "), length)
test_data$text_word_count = sapply(strsplit(test_data$text, " "), length)

#presence of numbers
train_data$text_num = grepl("\\d", train_data$text)
test_data$text_num = grepl("\\d", test_data$text)

train_data$title_num = grepl("\\d", train_data$title)
test_data$title_num = grepl("\\d", test_data$title)

#punctuation count
train_data$text_punct_count =sapply(strsplit(train_data$text, "[[:punct:]]"),
length) - 1
test_data$text_punct_count = sapply(strsplit(test_data$text, "[[:punct:]]"),
length) - 1

train_data$title_punct_count = sapply(strsplit(train_data$title, "[[:punct:]]"),
length) - 1
test_data$title_punct_count = sapply(strsplit(test_data$title, "[[:punct:]]"),
length) - 1



#detection from title using Naive Bayes Classifier
title_nb_model = naiveBayes(label ~ title + title_length + title_word_count +
title_num + title_punct_count, data = train_data)
title_nb_pred = predict(title_nb_model, newdata = test_data)
title_nb_accuracy = confusionMatrix(title_nb_pred,
test_data$label)$overall["Accuracy"]
```

```r
print(paste("Accuracy of Naive Bayes Classifier - Title :", title_nb_accuracy))
```

```
## [1] "Accuracy of Naive Bayes Classifier - Title : 0.600315955766193"
```

```r
#detection from title using Random Forest Classifier
title_rf_model = randomForest(label ~ title +  title_length + title_word_count +
title_num + title_punct_count, data = train_data)
title_rf_pred = predict(title_rf_model, newdata = test_data)
title_rf_accuracy = confusionMatrix(title_rf_pred,
test_data$label)$overall["Accuracy"]

print(paste("Accuracy of Random Forest Classifier - Title:", title_rf_accuracy))
```

```
## [1] "Accuracy of Random Forest Classifier - Title: 0.605845181674566"
```

```r
#detection from text using Naive Bayes Classifier
text_nb_model = naiveBayes(label ~ text + text_length + text_word_count +
text_num + text_punct_count, data = train_data)
text_nb_pred = predict(text_nb_model, newdata = test_data)
text_nb_accuracy = confusionMatrix(text_nb_pred,
test_data$label)$overall["Accuracy"]


print(paste("Accuracy of Naive Bayes Classifier - Text :", text_nb_accuracy))
```

```
## [1] "Accuracy of Naive Bayes Classifier - Text : 0.617693522906793"
```

```r
#detection from text using Random Forest Classifier
text_rf_model = randomForest(label ~ text + text_length + text_word_count +
text_num + text_punct_count, data = train_data)
text_rf_pred = predict(text_rf_model, newdata = test_data)
text_rf_accuracy = confusionMatrix(text_rf_pred,
test_data$label)$overall["Accuracy"]


print(paste("Accuracy of Random Forest Classifier - Tex :", text_rf_accuracy))
```

```
## [1] "Accuracy of Random Forest Classifier - Tex : 0.657187993680885"
```

```r
#detection using terms appearing in title or text using Naive Bayes Classifier
combined_nb_model = naiveBayes(label ~ title + text + title_length +
title_word_count + title_num + title_punct_count +
                                  text_length + text_word_count + text_num +
text_punct_count, data = train_data)
combined_nb_pred = predict(combined_nb_model, newdata = test_data)
combined_nb_accuracy = confusionMatrix(combined_nb_pred,
test_data$label)$overall["Accuracy"]


print(paste("Accuracy of Naive Bayes Classifier - Combined (Title + Text):",
combined_nb_accuracy))
```

```
## [1] "Accuracy of Naive Bayes Classifier - Combined (Title + Text):
0.607424960505529"
```

```
#detection using terms appearing in title or text using Random Forest Classifier
combined_rf_model = randomForest(label ~ title + text + title_length +
title_word_count + title_num + title_punct_count +
                                    text_length + text_word_count + text_num +
text_punct_count, data = train_data)
combined_rf_pred = predict(combined_rf_model, newdata = test_data)
combined_rf_accuracy = confusionMatrix(combined_rf_pred,
test_data$label)$overall["Accuracy"]


print(paste("Accuracy of Random Forest Classifier - Combined (Title + Text):",
combined_rf_accuracy))
```

```
## [1] "Accuracy of Random Forest Classifier - Combined (Title + Text):
0.699842022116904"
```

```
#detection from text using SVM Classifier
title_svm_model = svm(label ~ title_length + title_word_count + title_num +
title_punct_count, data = train_data)
title_svm_pred = predict(title_svm_model, newdata = test_data)
title_svm_accuracy = confusionMatrix(title_svm_pred,
test_data$label)$overall["Accuracy"]

print(paste("Accuracy of SVM Classifier - Title:", title_svm_accuracy))
```

```
## [1] "Accuracy of SVM Classifier - Title: 0.60347551342812"
```

```
#detection from text using SVM Classifier
text_svm_model = svm(label ~ text_length + text_word_count + text_num +
text_punct_count, data = train_data)
text_svm_pred = predict(text_svm_model, newdata = test_data)
text_svm_accuracy = confusionMatrix(text_svm_pred,
test_data$label)$overall["Accuracy"]

print(paste("Accuracy of SVM Classifier - Text:", text_svm_accuracy))
```

```
## [1] "Accuracy of SVM Classifier - Text: 0.682464454976303"
```

```
#detection using terms appearing in title or text using SVM Classifier
combined_svm_model = svm(label ~ title_length + title_word_count + title_num +
title_punct_count +
                            text_length + text_word_count + text_num +
text_punct_count, data = train_data)
combined_svm_pred = predict(combined_svm_model, newdata = test_data)
combined_svm_accuracy = confusionMatrix(combined_svm_pred,
test_data$label)$overall["Accuracy"]

print(paste("Accuracy of SVM Classifier - Combined (Title + Text):",
combined_svm_accuracy))
```

```
## [1] "Accuracy of SVM Classifier - Combined (Title + Text): 0.709320695102686"
```

```r
#WRITE THE TITLE HERE
new_title = "Trump takes on Cruz, but lightly
"

#WRITE THE TEXT HERE
new_text = "Killing Obama administration rules, dismantling Obamacare and
pushing through tax reform are on the early to-do list.
"

# to predict from a new title
predict_from_title = function(title, model, features) {

  title = preprocess_corpus(title)


  new_data =data.frame(
    title = title,
    title_length = nchar(title),
    title_word_count = length(unlist(strsplit(title, " "))),
    title_num = grepl("\\d", title),
    title_punct_count = sum(str_count(title, "[[:punct:]]"))
  )


  pred = predict(model, newdata = new_data)

  return(pred)
}



new_title_prediction_nb = predict_from_title(new_title, title_nb_model)
new_title_prediction_rf = predict_from_title(new_title, title_rf_model)
new_title_prediction_svm = predict_from_title(new_title, title_svm_model)


print(paste("Naive Bayes Prediction for the new title:",
new_title_prediction_nb, "Accuracy:", title_nb_accuracy*100, "%"))

## [1] "Naive Bayes Prediction for the new title: FAKE Accuracy:
60.0315955766193 %"

print(paste("Random Forest Prediction for the new title:",
new_title_prediction_rf, "Accuracy:", title_rf_accuracy*100, "%"))

## [1] "Random Forest Prediction for the new title: FAKE Accuracy:
60.5845181674566 %"

print(paste("SVM Prediction for the new title:", new_title_prediction_svm,
"Accuracy:", title_svm_accuracy*100, "%") )

## [1] "SVM Prediction for the new title: REAL Accuracy: 60.347551342812 %"
```

```r
# to predict from a new text
predict_from_text = function(text, model, features) {

  text = preprocess_corpus(text)


  new_data = data.frame(
    text = text,
    text_length = nchar(text),
    text_word_count = length(unlist(strsplit(text, " "))),
    text_num = grepl("\\d", text),
    text_punct_count = sum(str_count(text, "[[:punct:]]"))
  )


  pred = predict(model, newdata = new_data)

  return(pred)
}


new_text_prediction_nb = predict_from_text(new_text, text_nb_model,
                                           c('text_length', 'text_word_count',
'text_num',
                                             'text_punct_count'))


new_text_prediction_rf = predict_from_text(new_text, text_rf_model,
                                           c('text_length', 'text_word_count',
'text_num',
                                             'text_punct_count'))

new_text_prediction_svm = predict_from_text(new_text, text_svm_model,
                                            c('text_length', 'text_word_count',
'text_num',
                                              'text_punct_count'))


print(paste("Naive Bayes Prediction for the new text:", new_text_prediction_nb,
"Accuracy:", text_nb_accuracy*100, "%"))

## [1] "Naive Bayes Prediction for the new text: REAL Accuracy: 61.7693522906793
%"

print(paste("Random Forest Prediction for the new text:",
new_text_prediction_rf, "Accuracy:", text_rf_accuracy*100, "%"))

## [1] "Random Forest Prediction for the new text: REAL Accuracy:
65.7187993680885 %"

print(paste("SVM Prediction for the new text:", new_text_prediction_svm,
"Accuracy:", text_svm_accuracy*100, "%"))
```

```
## [1] "SVM Prediction for the new text: REAL Accuracy: 68.2464454976303 %"
```

```r
# to predict from a new title and text
predict_from_title_and_text = function(title, text, model, features) {

  title = preprocess_corpus(title)
  text = preprocess_corpus(text)


  new_data = data.frame(
    title = title,
    title_length = nchar(title),
    title_word_count = length(unlist(strsplit(title, " "))),
    title_num = grepl("\\d", title),
    title_punct_count = sum(str_count(title, "[[:punct:]]")),

    text = text,
    text_length = nchar(text),
    text_word_count = length(unlist(strsplit(text, " "))),
    text_num = grepl("\\d", text),
    text_punct_count = sum(str_count(text, "[[:punct:]]"))
  )


  pred = predict(model, newdata = new_data)

  return(pred)
}


combined_nb_prediction = predict_from_title_and_text(new_title, new_text,
combined_nb_model,
                                                       c('title_length',
'title_word_count', 'title_num', 'title_punct_count',
                                                         'text_length',
'text_word_count', 'text_num', 'text_punct_count'))

combined_rf_prediction = predict_from_title_and_text(new_title, new_text,
combined_rf_model,
                                                       c('title_length',
'title_word_count', 'title_num', 'title_punct_count',
                                                         'text_length',
'text_word_count', 'text_num', 'text_punct_count'))

combined_svm_prediction = predict_from_title_and_text(new_title, new_text,
combined_svm_model,
                                                        c('title_length',
'title_word_count', 'title_num', 'title_punct_count',
                                                          'text_length',
'text_word_count', 'text_num', 'text_punct_count'))
```

```
print(paste("Naive Bayes Prediction for the new title and text:",
combined_nb_prediction,
            "Accuracy:", combined_nb_accuracy*100, "%"))
```

```
## [1] "Naive Bayes Prediction for the new title and text: REAL Accuracy:
60.7424960505529 %"
```

```
print(paste("Random Forest Prediction for the new title and text:",
combined_rf_prediction,
            "Accuracy:", combined_rf_accuracy*100, "%"))
```

```
## [1] "Random Forest Prediction for the new title and text: REAL Accuracy:
69.9842022116904 %"
```

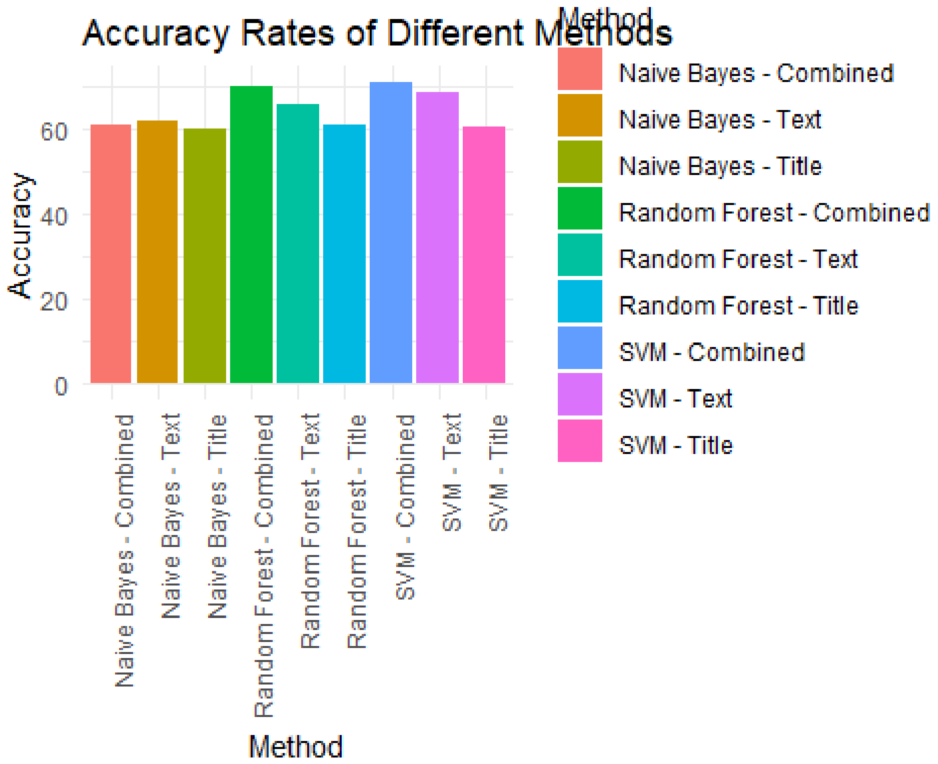
```
print(paste("SVM Prediction for the new title and text:",
combined_svm_prediction,
            "Accuracy:", combined_svm_accuracy*100, "%"))
```

```
## [1] "SVM Prediction for the new title and text: REAL Accuracy:
70.9320695102686 %"
```

```
accuracy_data = data.frame(
  Method = c("Naive Bayes - Title", "Random Forest - Title", "Naive Bayes -
Text", "Random Forest - Text", "Naive Bayes - Combined", "Random Forest -
Combined", "SVM - Title", "SVM - Text", "SVM - Combined"),
  Accuracy = c(title_nb_accuracy*100, title_rf_accuracy*100,
text_nb_accuracy*100, text_rf_accuracy*100, combined_nb_accuracy*100,
combined_rf_accuracy*100, title_svm_accuracy*100, text_svm_accuracy*100,
combined_svm_accuracy*100)
)
accuracy_data
```

```
##                     Method Accuracy
## 1      Naive Bayes - Title 60.03160
## 2    Random Forest - Title 60.58452
## 3       Naive Bayes - Text 61.76935
## 4     Random Forest - Text 65.71880
## 5   Naive Bayes - Combined 60.74250
## 6 Random Forest - Combined 69.98420
## 7              SVM - Title 60.34755
## 8               SVM - Text 68.24645
## 9           SVM - Combined 70.93207
```

```
ggplot(accuracy_data, aes(x = Method, y = Accuracy, fill = Method)) +
  geom_bar(stat = "identity", position = "dodge") +
  theme_minimal() +
  labs(title = "Accuracy Rates of Different Methods", x = "Method", y =
"Accuracy") +
  theme(axis.text.x = element_text(angle = 90, hjust = 1))
```



Conclusion

Naive Bayes and Random Forest

Title vs. Text: Naive Bayes and Random Forest performed comparably on both title and text features, with Random Forest showing a slight advantage in accuracy.

Combined Features: Combining title and text features resulted in improved accuracy for both Naive Bayes and Random Forest. Random Forest, in particular, demonstrated a notable increase in performance when using combined features.

Support Vector Machine (SVM)

Title vs. Text: SVM achieved competitive accuracy rates on both title and text features.

Combined Features: SVM outperformed Naive Bayes and Random Forest when using combined features, showing the highest accuracy among the considered methods (70.93%).