Harvard Capstone Project - Fake News Detection

Karen Antonia Paul

2024-06-05

The Structure:

- Section 1: Introduction
- Section 2: Data Extraction, Data Preparation and Data Exploration
- Section 3: Modeling Analysis
- Section 4: Final Results
- Section 5: Conclusion

Introduction

The Challenge of Fake News in the Digital Age: The rise of social media and online platforms has made spreading fake news alarmingly easy. These fabricated stories can be deceptively believable, leading people to accept them as truth. This, in turn, can have serious negative consequences.

Goal: This project investigates the effectiveness of machine learning models in detecting fake news. We compare the performance of Naive Bayes (NB), Random Forest (RF), and Support Vector Machine (SVM) models on a fake news detection task. The evaluation is based on their accuracy in classifying real and fake news articles within a test dataset.

Methodology: We trained and evaluated the models using features extracted from the news articles. Textual features - Length, word count, presence of numbers, and punctuation count. Combined features - We also explored models trained on a combination of title and text features. To ensure robust evaluation, the dataset was split into 80% for training and 20% for testing.

Data: The dataset contains a balanced number of fake news articles (3164) and real news articles (3171). This balance is ideal for training machine learning models that classify text data.

Dataset contains four columns:

- A serial number
- title (news heading)
- text (news content)
- label **(0 = fake and 1 = real)**

Reference of data: https://www.kaggle.com/datasets/hassanamin/textdb3

Data Extraction, Data Preparation and Data Exploration

Load Required Packages & Libraries

```
if(!require(tidyverse)) install.packages("tidyverse", repos = "http://cran.us.r-project.org")
if(!require(tm)) install.packages("tm", repos = "http://cran.us.r-project.org")
if(!require(caret)) install.packages("caret", repos = "http://cran.us.r-project.org")
if(!require(e1071)) install.packages("e1071", repos = "http://cran.us.r-project.org")
if(!require(randomForest)) install.packages("randomForest", repos = "http://cran.us.r-project.org")
if(!require(knitr)) install.packages("knitr", repos = "http://cran.us.r-project.org")
if(!require(SnowballC)) install.packages("SnowballC", repos = "http://cran.us.r-project.org")

library(tidyverse)
library(tm)
library(caret)
library(e1071)
library(randomForest)
library(knitr)
library(SnowballC)
```

Data Extraction

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

Data Preparation

Our first step is to take a quick look at the data and to ensure clear names for all columns and address missing data (null values) in the the text, title and label if it exists

```
# Sneak peak into the data
head(data)
```

```
##       X
## 1  8476
## 2 10294
## 3  3608
## 4 10142
## 5   875
## 6  6903
##                                                                                title
## 1                                                        You Can Smell Hillary's Fear
## 2 Watch The Exact Moment Paul Ryan Committed Political Suicide At A Trump Rally (VIDEO)
## 3                                              Kerry to go to Paris in gesture of sympathy
## 4  Bernie supporters on Twitter erupt in anger against the DNC: 'We tried to warn you!'
## 5                                    The Battle of New York: Why This Primary Matters
## 6                                                                        Tehran, USA
##
## 1
## 2
```

ever, when push came to shove, Paul Ryan - like many of his colleagues - turned into a sniveling appeaser. After all

```
ent laws that ##nstion as barriers to foreign investment and hopes Modi's government will act to open the huge Indian
 refusal to sup##rt the DNC-anointed candidate, pointing to WikiLeaks' revelations that top officials at the DNC had
e step closer ## avoiding a contested convention.\n\n"We've got to vote and you know Cruz is way, way down in the pol
## 6   \nI'm not an immigrant, but my grandparents are. More than 50 years ago, they arrived in New Yor
##   label
## 1  FAKE
## 2  FAKE
## 3  REAL
## 4  FAKE
## 5  REAL
## 6  FAKE
```

```
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
```

```
# How many rows and columns are there in the dataset?
dimensions <- dim(data)
names(dimensions) <- c("Number of Rows", "Number of Columns")
dimensions
```

```
##    Number of Rows Number of Columns
##              6335                 4
```

```
# Renaming the column names
colnames(data) <- c("serial_number", "title", "text", "label")

# Checking for null values in title, text and label columns
count_of_empty_title <- sum(nchar(data$title) == 0)
print(paste("Count of null values in the title -",count_of_empty_title))
```

```
## [1] "Count of null values in the title - 0"
```

```
count_of_empty_text <- sum(nchar(data$Text) == 0)
print(paste("Count of null values in the text -",count_of_empty_text))
```

```
## [1] "Count of null values in the text - 0"
```

```
count_of_empty_label <- sum(nchar(data$label) == 0)
print(paste("Count of null values in the label -",count_of_empty_label))
```

```
## [1] "Count of null values in the label - 0"
```

Now that we have done our preliminary check, to prepare the data for analysis, lets add some more text based features like number of characters, word count, presence of punctuation and numbers to the data which can be used for modelling.

Title Features:

- title_length: Length of the text in the "title" column.
- title_word_count: Number of words in the text of the "title" column.
- title_has_number: Presence of numbers (TRUE/FALSE) in the text of the "title" column.
- title_punctuation_count: Number of punctuation marks in the text of the "title" column.

Text Features:

- text_length: Length of the text in the "text" column.
- text_word_count: Number of words in the text of the "text" column.
- text_has_number: Presence of numbers (TRUE/FALSE) in the text of the "text" column.
- text_punctuation_count: Number of punctuation marks in the text of the "text" column.

```
# Adding a new columns to give the length, word count, number of punctuation and to see if they have nu
data <- data %>%
  mutate(
    label_digit = ifelse(label == "REAL", "1", "0"),
    title_length = nchar(title),
    text_length = nchar(text),
    title_word_count = sapply(strsplit(title, " "), length),
    text_word_count = sapply(strsplit(text, " "), length),
    title_has_number = grepl("\\d", title),
    text_has_number = grepl("\\d", text),
    title_punctuation_count = sapply(strsplit(title, "[[:punct:]]"), length),
    text_punctuation_count = sapply(strsplit(text, "[[:punct:]]"), length)
  )
```

We define a function named preprocess_corpus which takes the corpus of the text data and performs several cleaning operations to prepare it for further analysis. It will return a new corpus with the cleaned text.

The cleaning steps include:

- **Lowercasing:** All characters are converted to lowercase for consistency.
- **Number removal:** Numbers are removed from the text.
- **Punctuation removal:** Punctuation marks are removed, focusing on the meaning conveyed by the words themselves.
- **Stopword removal:** Common English words with little meaning (e.g., "the", "a", "is") are removed to improve the focus on content-rich words.
- **Stemming:** Words are reduced to their base or root form (e.g., "running" becomes "run").
- **Whitespace normalization:** Extra whitespace characters are removed to ensure consistent formatting within the corpus.

```
# Cleaning the text in the data
preprocess_corpus = function(text) {

  # Convert the text to lower case
  text = tolower(text)
  # Remove numbers
```

```
  text = removeNumbers(text)
  # Remove punctuation
  text = removePunctuation(text)
  # Remove common English stopwords
  text = removeWords(text, stopwords("en"))
  # Stem words to root words
  text = stemDocument(text)
  # Eliminate extra white spaces
  text = stripWhitespace(text)

  return(text)
}

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

Data Exploration

In this stage, we'll delve into the characteristics of the text data by comparing real and fake news articles. We'll focus on four key aspects:

Title Length Distribution: We will explore the distribution of title lengths between real and fake news articles. Are titles of real news articles generally longer or shorter than those of fake news articles? Do we observe a similar level of variation in title length within each category?

Text Word Count Distribution: We will examine the distribution of word counts within the article text for real and fake news. Are real news articles typically written with more or fewer words compared to fake news? Is there a significant difference in the spread of word counts between the two categories?

Number of Digits: We will analyze the prevalence of numbers in real and fake news articles. Does the inclusion of numbers differ between the two categories? Could this be indicative of specific writing styles or the presence of statistics and data in factual news reporting?

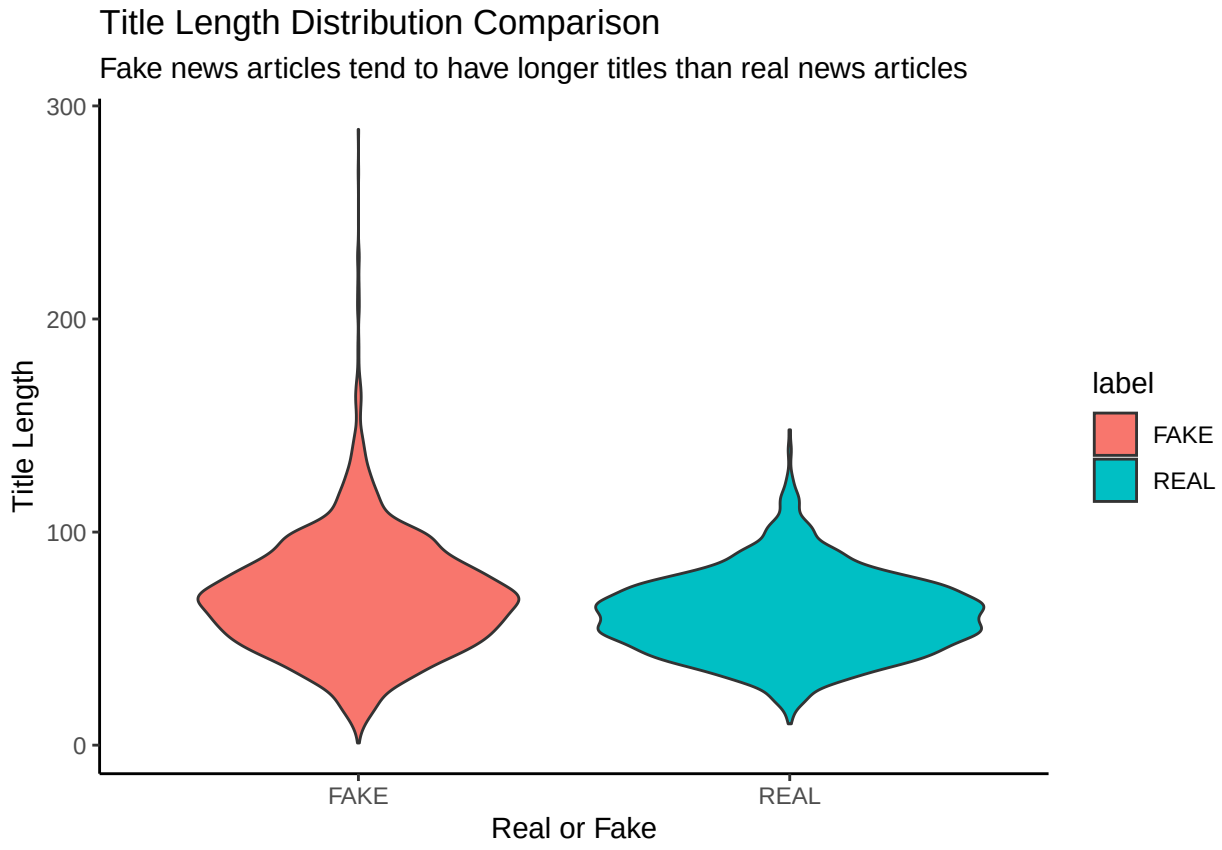
Punctuation Usage: We will explore how punctuation is used in real and fake news articles. Could punctuation usage be related to the sentiment or urgency conveyed in the text?

This can reveal potential patterns related to writing styles and information density in real and fake news.

Deep dive into news titles

Title length distribution

```
# Representation of length of real and fake news titles
ggplot(data, aes(x = label, y = title_length, fill = label)) +
  geom_violin() +
  labs(title = "Title Length Distribution Comparison",
  subtitle = "Fake news articles tend to have longer titles than real news articles",
  x = "Real or Fake",
  y = "Title Length")+
  guides(none)+
  theme_classic()
```



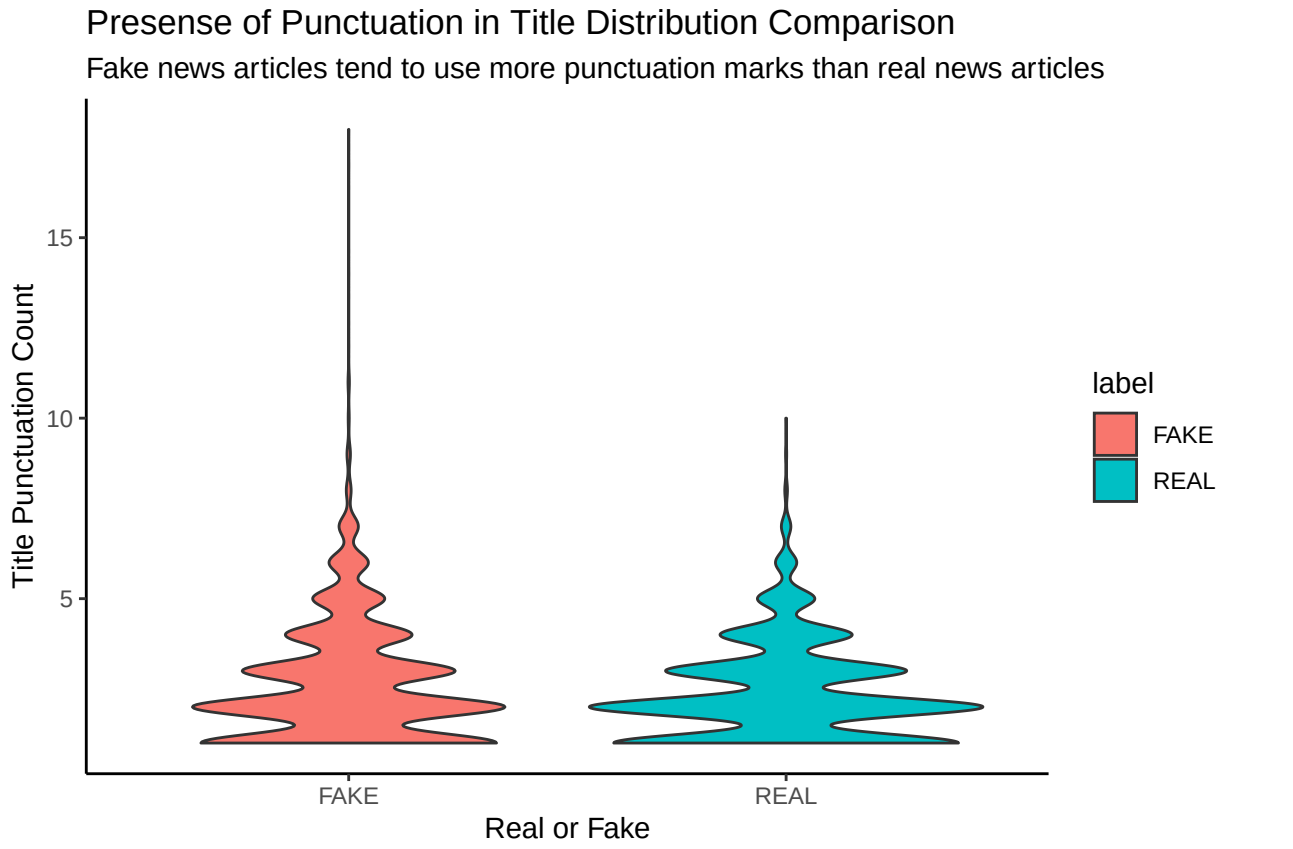
The presence of numbers in the title of real and fake news

```
# Representation of presence of numbers in the title of real and fake news
data_summary <- data %>%
                count(label, title_has_number)

ggplot(data_summary, aes(x = label, y = n, fill = title_has_number)) +
  geom_bar(stat = "identity") +
  labs(title = "Presense of Numbers in Title Comparison",
  subtitle = "Most of the titles in both real and fake news, do not contain numbers",
  x = "Real or Fake",
  y = "Title Has Number Count",
  fill = "Title has Number") +
  theme_classic()
```

Punctuation usage in title

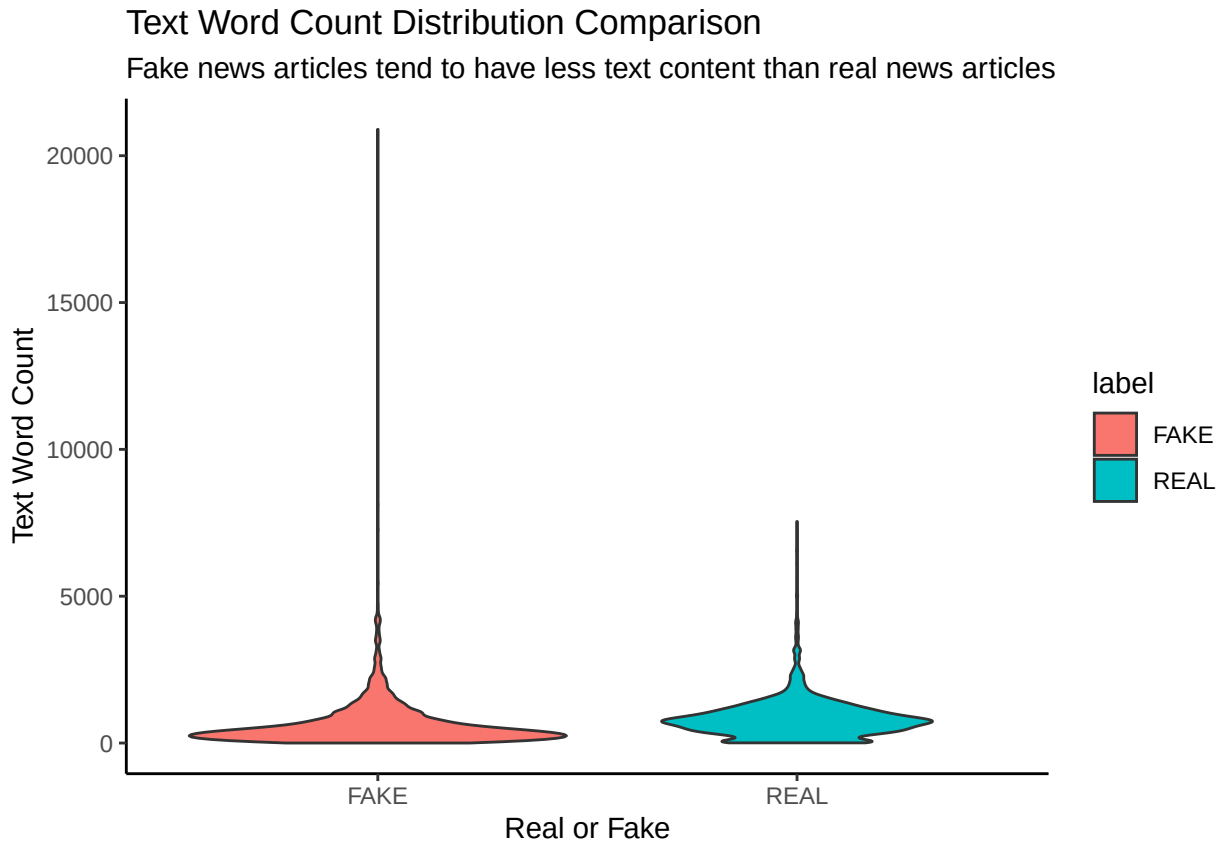
```
# Representation of number of punctuation in title of real and fake news
ggplot(data, aes(x = label, y = title_punctuation_count, fill = label)) +
  geom_violin() +
  labs(title = "Presense of Punctuation in Title Distribution Comparison",
  subtitle = "Fake news articles tend to use more punctuation marks than real news articles",
  x = "Real or Fake",
  y = "Title Punctuation Count")+
  guides(none)+
  theme_classic()
```



Deep dive into news text

Text word count distribution

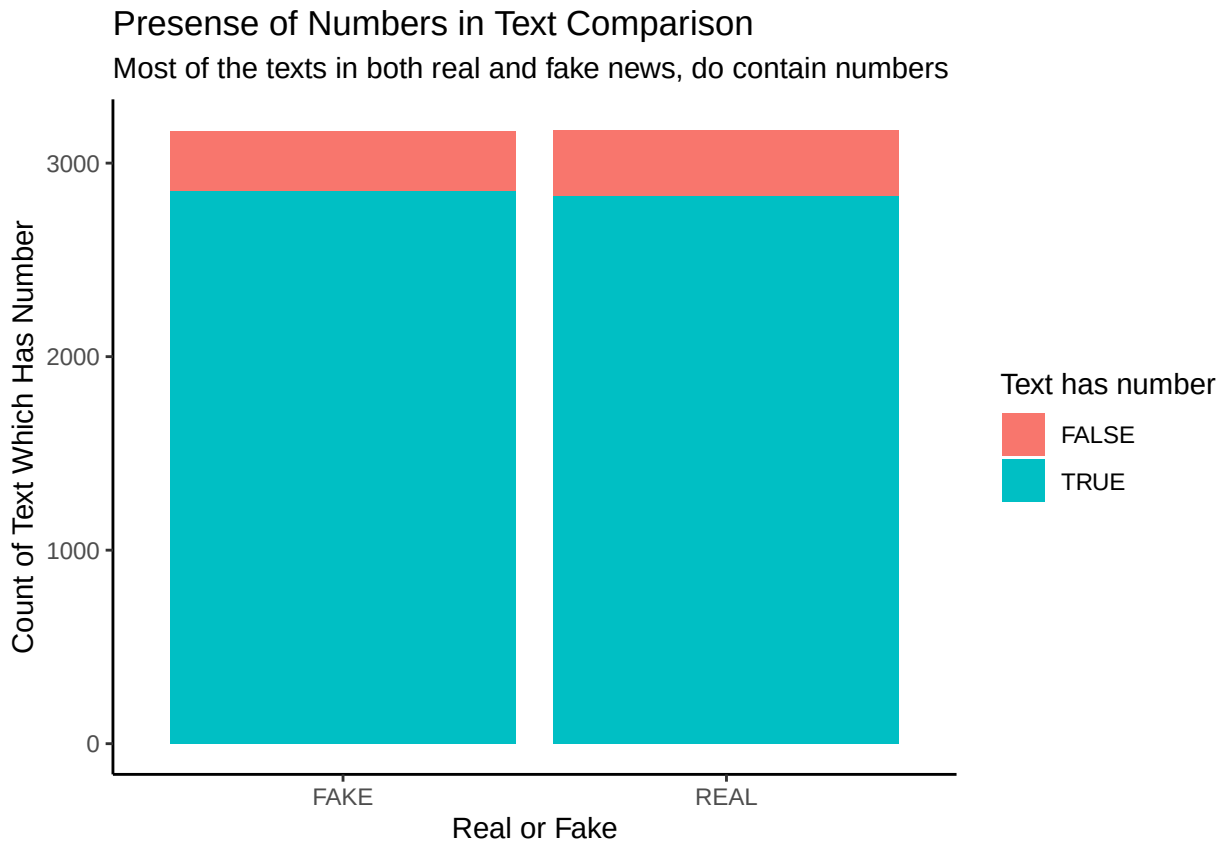
```
# Representation of word count of real and fake news texts
ggplot(data, aes(x = label, y = text_word_count, fill = label)) +
  geom_violin() +
  labs(title = "Text Word Count Distribution Comparison",
  subtitle = "Fake news articles tend to have less text content than real news articles",
  x = "Real or Fake",
  y = "Text Word Count")+
  guides(none)+
  theme_classic()
```



The presence of numbers in the text of real and fake news

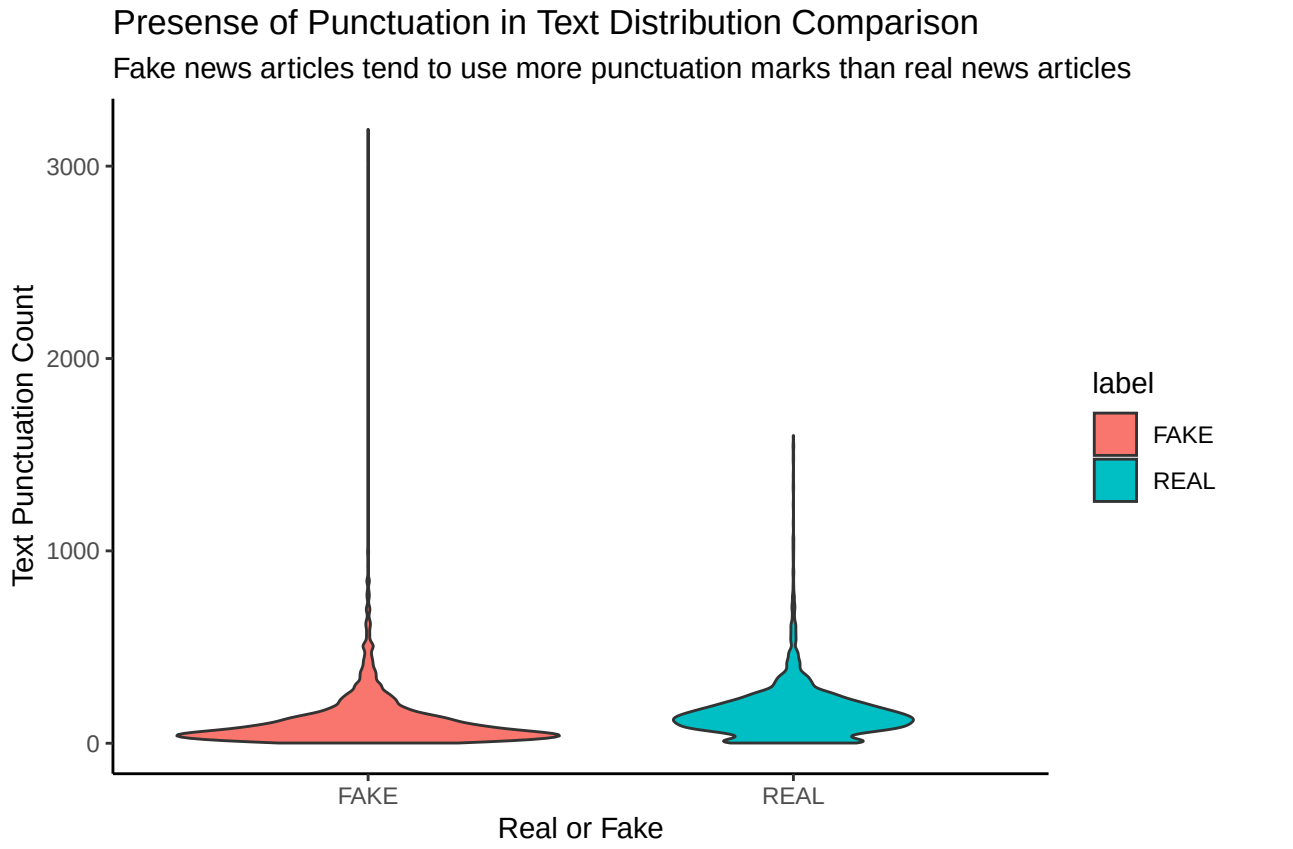
```
# Representation of numbers in the text of real and fake news
data_summary_2 <- data %>%
                  count(label, text_has_number)

ggplot(data_summary_2, aes(x = label, y = n, fill = text_has_number)) +
  geom_bar(stat = "identity") +
  labs(title = "Presense of Numbers in Text Comparison",
  subtitle = "Most of the texts in both real and fake news, do contain numbers",
  x = "Real or Fake",
  y = "Count of Text Which Has Number",
  fill = "Text has number") +
  theme_classic()
```



Punctuation usage in text

```
# Representation of number of punctuation in text of real and fake news
ggplot(data, aes(x = label, y = text_punctuation_count, fill = label)) +
  geom_violin() +
  labs(title = "Presense of Punctuation in Text Distribution Comparison",
  subtitle = "Fake news articles tend to use more punctuation marks than real news articles",
  x = "Real or Fake",
  y = "Text Punctuation Count")+
  guides(none)+
  theme_classic()
```



Listing down our findings from the charts above:

- **Title length:** Fake news articles tend to have longer titles than real news articles. The violin plot for fake news articles seems taller, indicating a potentially larger variation in title lengths. There could be fake news articles with very short titles and some with very long titles. The violin plot for real news articles appears to be shorter, suggesting a tighter distribution of title lengths. Most real news articles might have titles within a closer range of lengths.
- **Text word count:** Majority of the fake news articles tend to be less text-intensive than real news articles. This could be for a number of reasons, like, fake news articles may be designed to be quickly read and shared on social media. They may not be well-researched or well-written and rely more on images and videos than text. Since the violin plot for fake news article seems longer, this indicates a larger variation also.
- **Numbers:** The majority of titles in both real and fake news articles likely don't contain numbers. But a significant portion of the text of both real and fake news articles likely contain numbers.
- **Punctuation in the title:** The violin plot suggests that titles of fake news articles tend to use more punctuation marks compared to real news titles. The taller violin plot for fake news titles indicates a potentially larger variation in punctuation use. There could be fake news titles with very low and very high punctuation usage. The shorter violin plot for real news titles suggests a tighter distribution of punctuation use. Most real news titles might have punctuation counts within a closer range.
- **Punctuation in the text:** Fake news articles might have a higher overall frequency of punctuation compared to real news. This could indicate a few things like:

    1. Exclamation Points and Emotional Appeal used in fake news might rely more on exclamation points (!) to create a sense of urgency, shock, or outrage, manipulating emotions.
    2. Questions and Engagement: Question marks (?) could be used more frequently to spark curiosity or create a false sense of interactivity, making the reader feel involved.
    3. Choppier Sentences: Increased use of commas (,) or semicolons (;) could lead to shorter, punchier sentences, making the fake news easier to read but potentially less nuanced.

Modeling Analysis

In this phase, we'll delve into three popular machine learning methods for classification: Naive Bayes (NB), Random Forest (RF), and Support Vector Machines (SVM). We'll train and evaluate models using these techniques to determine which approach yields the most effective results for fake news detection in our specific dataset.

Naive Bayes: Naive Bayes is a classification technique that utilizes probability theory to predict the likelihood of an item belonging to a specific category. Imagine you have a pile of emails and want to separate them into "important" and "spam." Naive Bayes works by analyzing the email's content and calculating the probability of it being spam based on individual words or phrases.

Support Vector Machine: Support Vector Machines (SVMs) are another popular machine learning technique for classification tasks. Imagine you have a dataset with two distinct categories, like real and fake news. An SVM aims to find a dividing line called a hyperplane in the data space that best separates these categories.

Random Forest: Random Forest is a powerful machine learning technique that builds upon the idea of decision trees. Imagine a series of simple yes-or-no questions, like a flowchart, that lead you to a final decision. This is essentially a decision tree. In Random Forest, we don't rely on just one decision tree, but rather a whole "forest" of them!

Confusion matrix: A confusion matrix is what we use to helps visualize the performance of the models. It summarizes how often the model makes correct and incorrect predictions on a set of data.

Accuracy - Is the proportion of all correctly classified cases among all cases:

$$\text{Accuracy} = \frac{\text{True Positives} + \text{True Negatives}}{(\text{True Positives} + \text{False Positives} + \text{True Negatives} + \text{False Negatives})}$$

To start with the modelling, we are splitting the data into training and testing sets.

```
# Convert variable label into a factor data type
data$label = as.factor(data$label)

# train and test data split (80% - training, 20 - testing )
set.seed(1)
splitIndex = createDataPartition(data$label, p = 0.8, list = FALSE)
train_data = data[splitIndex, ]
test_data = data[-splitIndex, ]
```

Naive Bayes

Naive bayes using the title features

```
# detection from title using Naive Bayes
# Model
title_nb_model = naiveBayes(label ~ title_length
                            + title_word_count
                            + title_punctuation_count
                            , data = train_data)

# Prediction
title_nb_prediction = predict(title_nb_model, newdata = test_data)

# Result
title_nb_accuracy = confusionMatrix(title_nb_prediction, test_data$label)$overall["Accuracy"]

results <- tibble(Model = "Naive Bayes - Using Title Features"
                , Accuracy = round(title_nb_accuracy*100,2))
print(paste("Naive Bayes - Using Title Features: ",title_nb_accuracy*100))
```

```
## [1] "Naive Bayes - Using Title Features:  58.9257503949447"
```

Naive bayes using the text features

```
# detection from text using Naive Bayes
# Model
text_nb_model = naiveBayes(label ~ text_length
                           + text_word_count
                           + text_has_number
                           + text_punctuation_count
                           , data = train_data)

# Prediction
text_nb_prediction = predict(text_nb_model, newdata = test_data)

# Result
text_nb_accuracy = confusionMatrix(text_nb_prediction, test_data$label)$overall["Accuracy"]

results <- bind_rows(results
                , tibble(Model ="Naive Bayes - Using Text Features"
                , Accuracy = round(text_nb_accuracy*100,2)))
print(paste("Naive Bayes - Using Text Features: ",text_nb_accuracy*100))
```

```
## [1] "Naive Bayes - Using Text Features:  59.2417061611374"
```

Naive bayes using the title and text features

```
# detection from title and text using Naive Bayes
# Model
combined_nb_model = naiveBayes(label ~ title_length
                               + title_word_count
                               + title_has_number
                               + title_punctuation_count
                               + text_length
                               + text_word_count
                               + text_has_number
                               + text_punctuation_count
                               , data = train_data)

# Prediction
combined_nb_prediction = predict(combined_nb_model, newdata = test_data)

# Result
combined_nb_accuracy = confusionMatrix(combined_nb_prediction, test_data$label)$overall["Accuracy"]

results <- bind_rows(results
                , tibble(Model ="Naive Bayes - Using Title and Text Features"
                , Accuracy = round(combined_nb_accuracy*100,2)))
print(paste("Naive Bayes - Using Title and Text Features: ",combined_nb_accuracy*100))
```

```
## [1] "Naive Bayes - Using Title and Text Features:  60.347551342812"
```

Support Vector Machine

Support Vector Machine using the title features

```
#detection from title using SVM
# Model
title_svm_model = svm(label ~ title_length
                      + title_word_count
                      + title_has_number
                      + title_punctuation_count
                      , data = train_data)

# Prediction
title_svm_prediction = predict(title_svm_model, newdata = test_data)

# Result
title_svm_accuracy = confusionMatrix(title_svm_prediction, test_data$label)$overall["Accuracy"]

results <- bind_rows(results
                , tibble(Model = "Support Vector Machine - Using Title Features"
                , Accuracy = round(title_svm_accuracy*100,2)))
print(paste("Support Vector Machine - Using Title Features: ",title_svm_accuracy*100))
```

```
## [1] "Support Vector Machine - Using Title Features:  59.8736176935229"
```

Support Vector Machine using the text features

```
#detection from text using SVM
# Model
text_svm_model = svm(label ~ text_length
                     + text_word_count
                     + text_has_number
                     + text_punctuation_count
                     , data = train_data)

# Prediction
text_svm_prediction = predict(text_svm_model, newdata = test_data)

# Result
text_svm_accuracy = confusionMatrix(text_svm_prediction, test_data$label)$overall["Accuracy"]

results <- bind_rows(results
                , tibble(Model = "Support Vector Machine - Using Text Features"
                , Accuracy = round(text_svm_accuracy*100,2)))
print(paste("Support Vector Machine - Using Text Features: ",text_svm_accuracy*100))
```

```
## [1] "Support Vector Machine - Using Text Features:  65.4028436018957"
```

Support Vector Machine using the title and text features

```
#detection from title and text using SVM
# Model
combined_svm_model = svm(label ~ title_length
                         + title_word_count
                         + title_has_number
                         + title_punctuation_count
                         + text_length
                         + text_word_count
                         + text_has_number
                         + text_punctuation_count
                         , data = train_data)

# Prediction
combined_svm_prediction = predict(combined_svm_model, newdata = test_data)

# Result
combined_svm_accuracy = confusionMatrix(combined_svm_prediction, test_data$label)$overall["Accuracy"]

results <- bind_rows(results
                , tibble(Model = "Support Vector Machine - Using Title and Text Features"
                , Accuracy = round(combined_svm_accuracy*100,2)))
print(paste("Support Vector Machine - Using Title and Text Features: ",combined_svm_accuracy*100))
```

```
## [1] "Support Vector Machine - Using Title and Text Features:  68.5624012638231"
```

Random Forest

Random Forest using the title features

```
# detection from title using Random Forest
# Model
title_rf_model = randomForest(label ~ title_length
                              + title_word_count
                              + title_has_number
                              + title_punctuation_count
                              , data = train_data)

# Prediction
title_rf_prediction = predict(title_rf_model, newdata = test_data)

# Result
title_rf_accuracy = confusionMatrix(title_rf_prediction, test_data$label)$overall["Accuracy"]

results <- bind_rows(results
                , tibble(Model = "Random Forest - Using Title Features"
                , Accuracy = round(title_rf_accuracy*100,2)))
print(paste("Random Forest - Using Title Features: ",title_rf_accuracy*100))
```

```
## [1] "Random Forest - Using Title Features:  58.2148499210111"
```

Random Forest using the text features

```
# detection from text using Random Forest
# Model
text_rf_model = randomForest(label ~ text_length
                             + text_word_count
                             + text_has_number
                             + text_punctuation_count
                             , data = train_data)

# Prediction
text_rf_prediction = predict(text_rf_model, newdata = test_data)

# Result
text_rf_accuracy = confusionMatrix(text_rf_prediction, test_data$label)$overall["Accuracy"]

results <- bind_rows(results
                , tibble(Model = "Random Forest - Using Text Features"
                , Accuracy = round(text_rf_accuracy*100,2)))
print(paste("Random Forest - Using Text Features: ",text_rf_accuracy*100))
```

```
## [1] "Random Forest - Using Text Features:  69.2733017377567"
```

Random Forest using the title and text features

```
# detection from title and text using Random Forest
# Model
combined_rf_model = randomForest(label ~ title
                                 + text
                                 + title_length
                                 + title_word_count
                                 + title_has_number
                                 + title_punctuation_count
                                 + text_length
                                 + text_word_count
                                 + text_has_number
                                 + text_punctuation_count
                                 , data = train_data)

# Prediction
combined_rf_prediction = predict(combined_rf_model, newdata = test_data)

# Result
combined_rf_accuracy = confusionMatrix(combined_rf_prediction, test_data$label)$overall["Accuracy"]

results <- bind_rows(results
                , tibble(Model = "Random Forest - Using Title and Text Features"
                , Accuracy = round(combined_rf_accuracy*100,2)))
print(paste("Random Forest - Using Title and Text Features: ",combined_rf_accuracy*100))
```

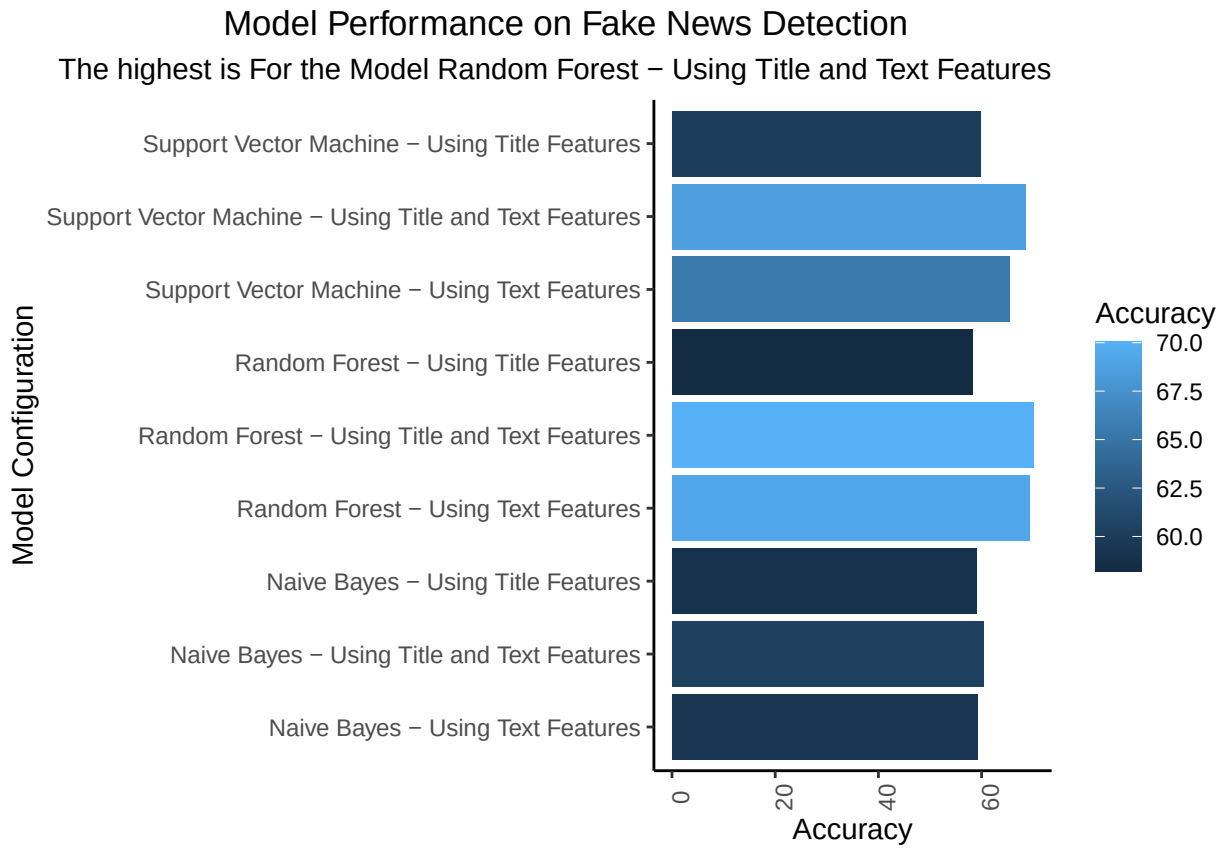
```
## [1] "Random Forest - Using Title and Text Features:  70.0631911532385"
```

Final Result

```
results_final <- results %>% kable()
results_final
```

Model	Accuracy
Naive Bayes - Using Title Features	58.93
Naive Bayes - Using Text Features	59.24
Naive Bayes - Using Title and Text Features	60.35
Support Vector Machine - Using Title Features	59.87
Support Vector Machine - Using Text Features	65.40
Support Vector Machine - Using Title and Text Features	68.56
Random Forest - Using Title Features	58.21
Random Forest - Using Text Features	69.27
Random Forest - Using Title and Text Features	70.06

```
ggplot(results, aes(y = Model, x = Accuracy, fill = Accuracy)) +
  geom_col() +
  labs(title = "Model Performance on Fake News Detection",
  subtitle = "The highest is For the Model Random Forest - Using Title and Text Features",
  y = "Model Configuration", x = "Accuracy") +
  theme_classic() +
  theme(axis.text.x = element_text(angle = 90),
  plot.title = element_text(hjust = 1.5),
  plot.subtitle = element_text(hjust = 1))
```



Conclusion

Our investigation into machine learning models for fake news detection revealed that **Random Forest achieved the best overall performance in our specific dataset with an accuracy of 70.06%**. Here's a breakdown of the key findings:

Model Performance: Random Forest achieved the highest accuracy among the evaluated models. Both Naive Bayes (NB) and Support Vector Machines (SVM) performed comparably on models trained with title or text features alone. When combining title and text features, their accuracy improved, but they did not surpass Random Forest's performance.

Combining Features Most Effective: Using a combination of title and text features led to the best results for all models. This highlights the importance of considering both title and textual content for accurate fake news detection.

Limtitations

It's important to remember that these results are specific to the chosen dataset and the chosen models. Further improvements can be made to the model by incorporating additional factors such a source, author etc., if its available.