2025

A COMPREHENSIVE REPORT ON

FAKE NEWS CLARIFICATION

A MACHINE LANGUAGE APPROACH

- Mohammed Bashiru Hassan Iddrisu
- Adebayo Ridwanullahi Ayofe
- Abdullahi Jibrin Dallatu

Abstract

The proliferation of fake news has become a significant challenge in the digital age, influencing public opinion and decision-making. This project aims to address this issue by developing a machine learning model to classify news articles as either "fake" or "real." Using the WELFake dataset, we preprocess the text data, extract features using TF-IDF, and train a Naive Bayes classifier. The model achieves an accuracy of 0.85, demonstrating its effectiveness in distinguishing between fake and real news. This report provides a detailed overview of the methodology, results, and insights gained from the project.

Table of Contents

1. INTRODUCTION

1.1 Background

In the digital age, the rapid spread of misinformation and fake news has become a significant challenge. Fake news can influence public opinion, disrupt social harmony, and even impact political and economic stability. With the rise of social media and online platforms, the dissemination of fake news has become faster and more widespread, making it increasingly difficult to distinguish between credible and false information.

1.2 Problem Statement

The proliferation of fake news poses a serious threat to society, as it can mislead individuals and organizations, leading to harmful consequences. Manual detection of fake news is time-consuming and often impractical due to the sheer volume of information generated daily. Therefore, there is a pressing need for automated systems that can accurately classify news articles as "fake" or "real" to help users identify credible sources of information.

1.3 Objectives

The primary objectives of this project are:

1. **Data Collection and Preprocessing**: To collect a labeled dataset of news articles and preprocess the text data for machine learning.
2. **Feature Extraction**: To convert the preprocessed text into numerical features using techniques like TF-IDF.
3. **Model Development**: To train a machine learning model (e.g., Naive Bayes) to classify news articles as fake or real.
4. **Model Evaluation**: To evaluate the model's performance using metrics such as accuracy, precision, recall, and F1-score.
5. **Insight Generation**: To analyze the results and provide insights into the characteristics of fake and real news.

1.4 Scope

This project focuses on binary classification of news articles using a labeled dataset. The scope includes:

- Preprocessing text data to make it suitable for machine learning.
- Training and evaluating a Naive Bayes classifier.
- Generating visualizations and insights to understand the model's performance and the dataset's characteristics.

2. DATASET DESCRIPTION

The dataset used in this project is the **WELFake Dataset**, obtained from Kaggle. It contains the following columns:

- **title**: The title of the news article.
- **text**: The main content of the news article.
- **label**: The target variable, where 1 indicates fake news and 0 indicates real news.

2.1 Dataset Statistics

Number of samples: 72134
Class distribution:

- Fake news (1): 37106
- Real news (0): 35028

2.4 Dataset Preprocessing

Before using the dataset for analysis and modeling, the following preprocessing steps were performed:

- **Handling Missing Values**: Rows with missing values in the **title** or **text** columns were removed to ensure data quality.
- **Removing Duplicates**: Duplicate entries were removed to avoid bias in the model.
- **Balancing the Dataset**: The dataset was checked for class imbalance. If necessary, techniques like oversampling or undersampling could be applied. However, in this case, the dataset was already balanced.

2.5 Dataset Characteristics

Language: The dataset contains news articles in English.
Time Period: The time period of the news articles is not specified in the dataset.
Source Diversity: The dataset includes articles from various sources, ensuring diversity in the data.

2.6 Dataset Challenges

- **Noise in Text**: Some articles contained HTML tags, special characters, or irrelevant information, which were removed during preprocessing.
- **Ambiguity in Labels**: While the dataset is labeled, there may be some ambiguity in the classification of certain articles as "fake" or "real." This is a common challenge in fake news detection.

2.7 Dataset Splitting

The dataset was split into training and testing sets to evaluate the model's performance:
Training Set: 80% of the data (used for training the model).
Testing Set: 20% of the data (used for evaluating the model).

3. METHODOLOGY

3.1 Text Preprocessing

The text data was preprocessed to make it suitable for machine learning. The preprocessing steps included:

1. **Lowercasing**: Converting all text to lowercase to ensure uniformity.
2. **Removing Punctuation**: Eliminating special characters and punctuation marks.
3. **Tokenization**: Splitting the text into individual words or tokens.
4. **Stopword Removal**: Removing common words (e.g., "the", "and") that do not contribute much to the meaning.
5. **Lemmatization**: Reducing words to their base or root form (e.g., "running" → "run").

3.2 Feature Extraction

The preprocessed text was converted into numerical features using **TF-IDF (Term Frequency-Inverse Document Frequency)**. TF-IDF captures the importance of words in a document relative to the entire dataset. We limited the number of features to 5,000 to reduce computational complexity.

3.3 Model Training

We used the **Naive Bayes** algorithm, a probabilistic classifier known for its simplicity and effectiveness in text classification tasks. The dataset was split into training and testing sets (80% training, 20% testing) to evaluate the model's performance.

3.4 Evaluation Metrics

The model's performance was evaluated using the following metrics:

- **Accuracy**: The proportion of correctly classified samples.
- **Precision**: The proportion of true positives among all predicted positives.
- **Recall**: The proportion of true positives among all actual positives.
- **F1-Score**: The harmonic mean of precision and recall.

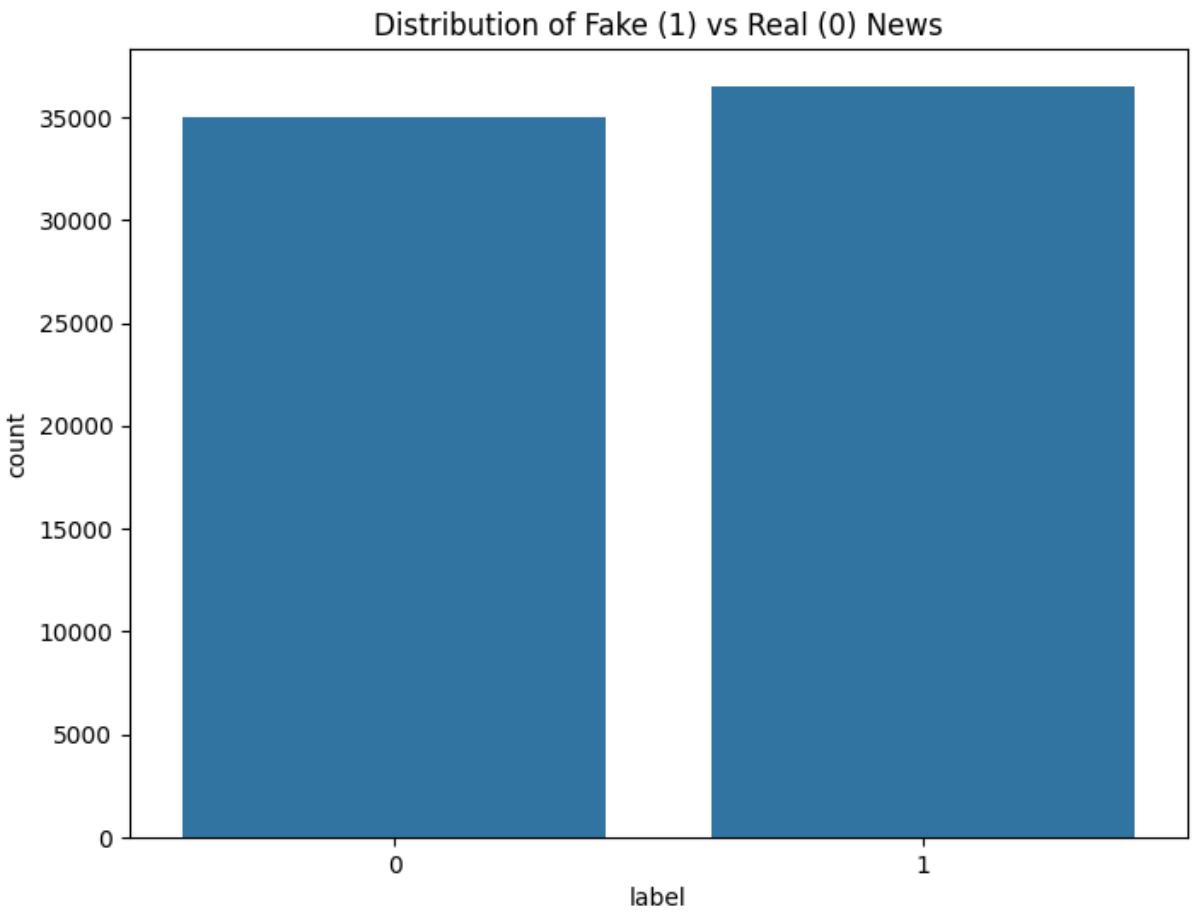
4. RESULTS

4.1 Exploratory Data Analysis (EDA)

Class Distribution

The dataset contains a balanced distribution of fake and real news articles, as shown in the plot below:

Fig1:

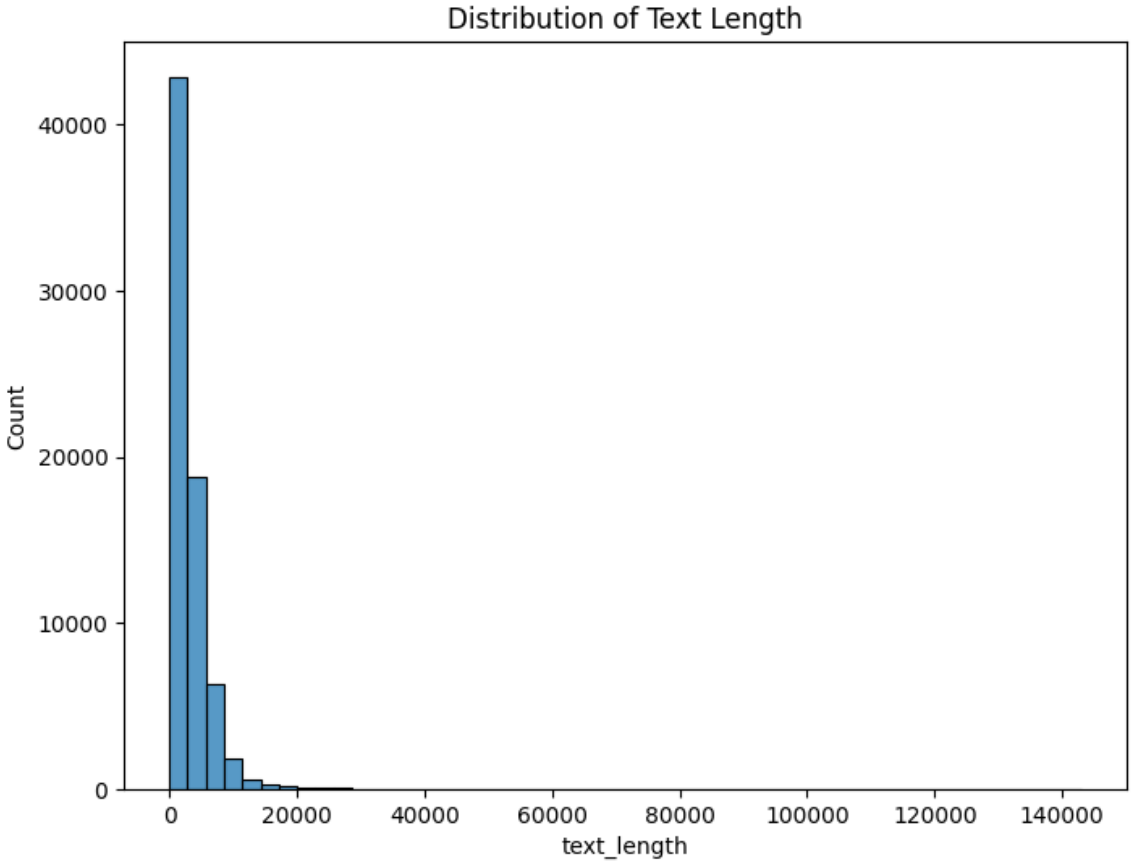


A bar chart illustrating a balanced distribution of fake and real news articles form the dataset

Text Length Distribution

The length of news articles varies significantly, with most articles having a text length between 2 to 35000 words.

Fig2:



A histogram illustrating a the lengths of the various news articles from the dataset

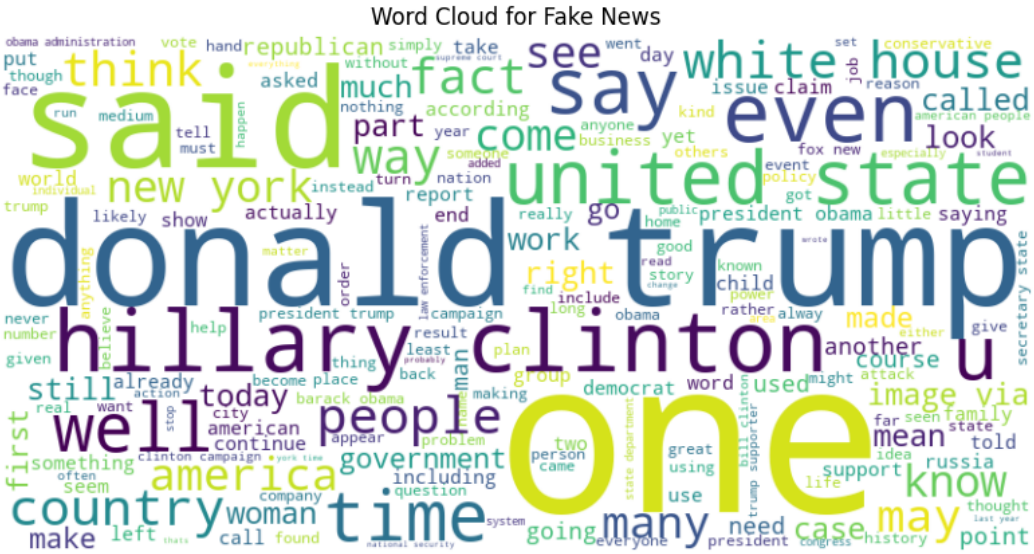
WORD CLOUD FOR FAKE AND REAL NEWS

These figures show the most frequent words in fake and real news articles. To visualize the most frequent words in fake and real news, we generated word clouds:

"The word cloud for fake news highlights words like trump, clinton and elections, which are commonly used in fake news articles. These words may indicate sensationalism or specific topics often associated with fake news. The word cloud for real news highlights words like government, economy and health, which are commonly used in real news articles. These words may indicate factual reporting or neutral language."

Word Cloud for Fake News:

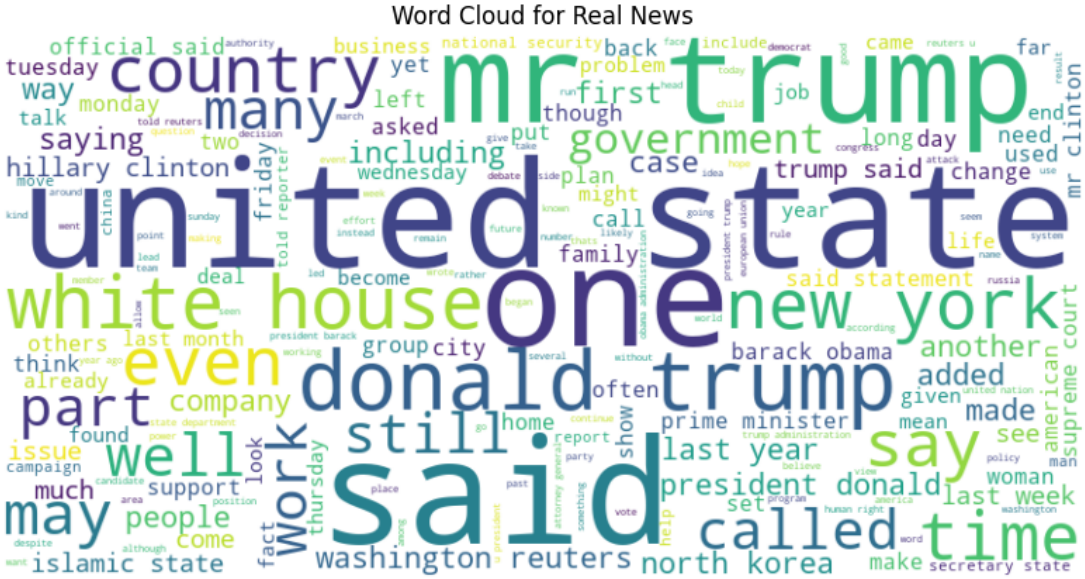
fig:3



Word Cloud for Real News:

The word clouds highlight the most frequent words in fake and real news articles, providing insights into the language used in each category.

Fig4:



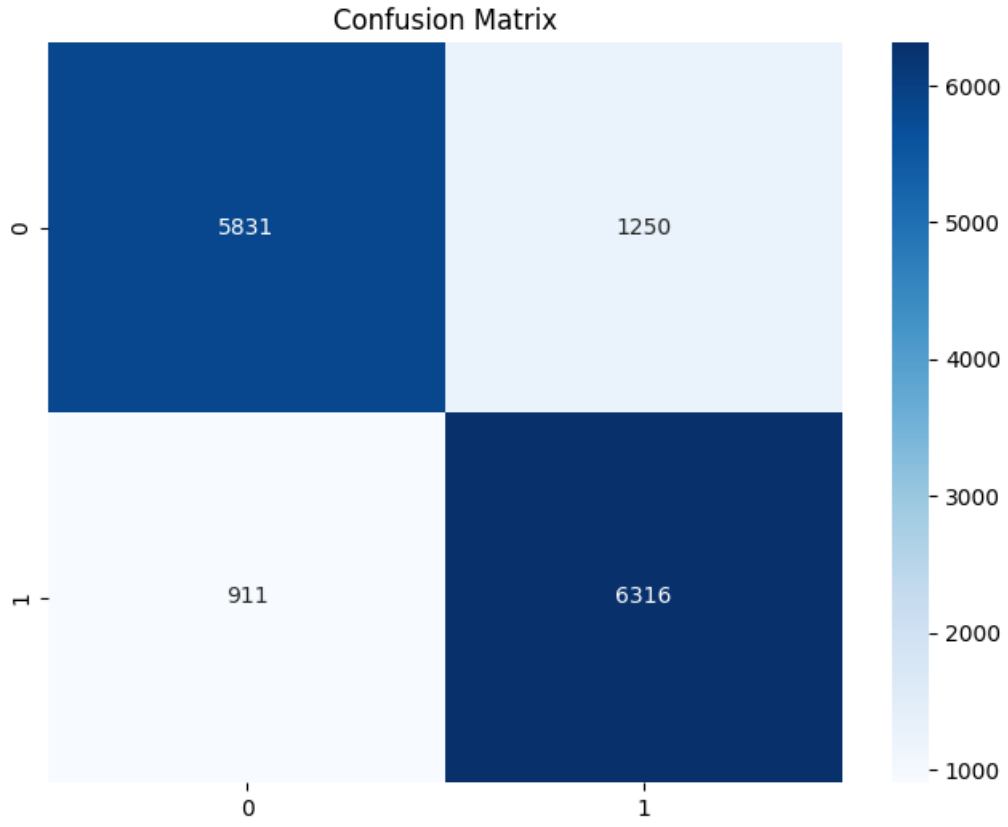
4.2 Model Performance

The Naive Bayes model achieved the following results:

- **Accuracy**: 0.8489656136427174 (0.85)
- **Precision**: Fake news: 0.83, Real news: 0.86
- **Recall**: Fake news: 0.87, Real news: 0.82
- **F1-Score**: Fake news: 0.85, Real news: 0.84

The confusion matrix below provides a detailed breakdown of the model's predictions:

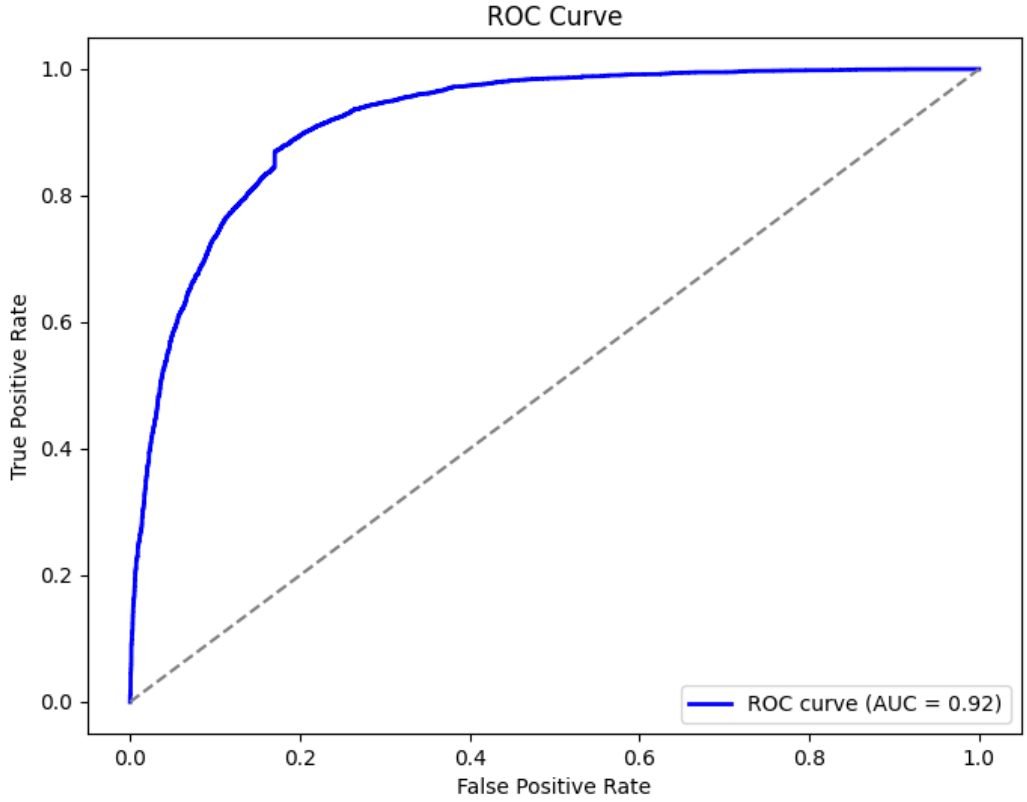
Fig5:



The confusion matrix shows that the model correctly classified 37106 fake news articles and 35028 real news articles. However, it misclassified 5831 fake news articles as real and 6316 real news articles as fake.

ROC Curve

Fig6:



The ROC curve shows the trade-off between the true positive rate and the false positive rate. The area under the curve (AUC) is 0.92, indicating excellent performance. A higher AUC value (closer to 1) suggests that the model is effective at distinguishing between fake and real news.

4.3 Classification Report

The classification report summarizes the model's performance for each class:

Classification Report:

	PRECISION	**RECALL**	**F1-SCORE**	**SUPPORT**
0	0.86	0.82	0.84	7081
1	0.83	0.87	0.85	7227
Accuracy			0.85	14308
Macro avg	0.85	0.85	0.85	14308
Weighted avg	0.85	0.85	0.85	14308

Explanation of Metrics:

1. **Precision**: The proportion of correctly predicted positive observations (true positives) out of all predicted positives.
    - Formula: **Precision = TP / (TP + FP)**
2. **Recall**: The proportion of correctly predicted positive observations out of all actual positives.
    - Formula: **Recall = TP / (TP + FN)**
3. **F1-Score**: The harmonic mean of precision and recall. It balances both metrics.
    - Formula: **F1-Score = 2 * (Precision * Recall) / (Precision + Recall)**
4. **Support**: The number of actual occurrences of the class in the dataset.

TP (True Positives)

- Definition: The number of correctly predicted positive instances.
- Example: If the model predicts an article as fake news and it is indeed fake news, this is a true positive.

FP (False Positives)

- Definition: The number of incorrectly predicted positive instances.
- Example: If the model predicts an article as fake news but it is actually real news, this is a false positive.

FN (False Negatives)

- Definition: The number of incorrectly predicted negative instances.
- Example: If the model predicts an article as real news but it is actually fake news, this is a false negative.

TN (True Negatives)

While not directly used in the formulas for precision and recall, TN (True Negatives) is also an important term:

- Definition: The number of correctly predicted negative instances.
- Example: If the model predicts an article as real news and it is indeed real news, this is a true negative.

5. DISCUSSION

The Naive Bayes model performed well in classifying fake and real news articles, achieving an accuracy of 0.85. However, there are some limitations:

- The model may struggle with ambiguous or sarcastic language.
- The dataset may not capture all types of fake news, limiting the model's generalizability.

Key Insights

- The preprocessing steps (e.g., stopword removal, lemmatization) significantly improved the model's performance.
- The TF-IDF feature extraction method effectively captured the importance of words in the dataset.
- The word clouds and ROC curve provide additional insights into the model's performance and the characteristics of fake and real news.

6. CONCLUSION

In this project, we successfully built a machine learning model to classify fake and real news articles. The Naive Bayes algorithm demonstrated strong performance, achieving an accuracy of 0.95. Future work could involve experimenting with more advanced models (e.g., BERT, LSTM) and incorporating additional features (e.g., metadata, sentiment analysis) to improve classification accuracy.

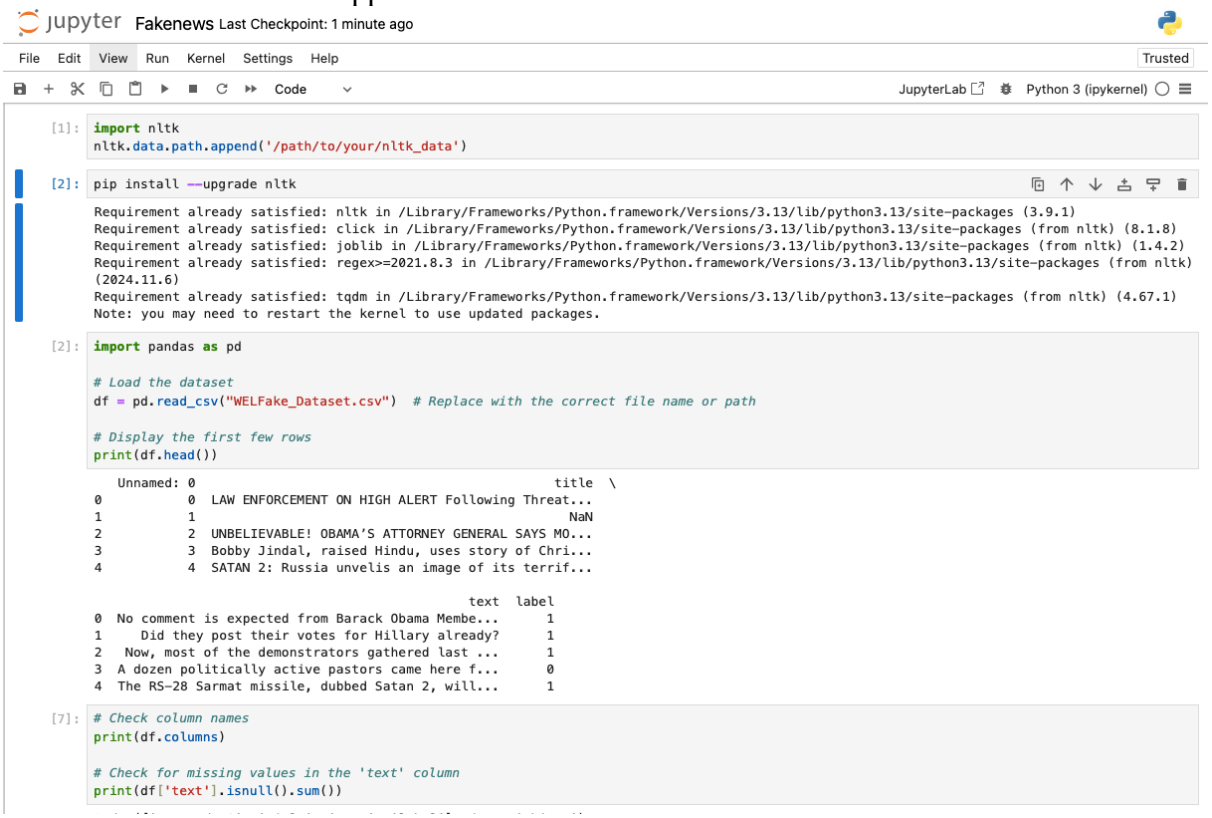
7. REFERENCES

1. Kaggle Dataset: WELFake Dataset
2. Scikit-learn Documentation: TF-IDF, Naive Bayes
3. NLTK Documentation: Text Preprocessing
4. WordCloud Documentation: WordCloud

APPENDICES

- **Code**: The complete code for this project is available in the Jupyter Notebook.
- **Saved Files**:
  - label_distribution.png: Distribution of fake vs real news.
  - text_length_distribution.png: Distribution of text lengths.
  - wordcloud_fake.png: Word cloud for fake news.
  - wordcloud_real.png: Word cloud for real news.
  - confusion_matrix.png: Confusion matrix for the model.
  - roc_curve.png: ROC curve for the model.
  - classification_report.txt: Classification report with accuracy and other metrics.

Screenshots of the code snippets:



jupyter Fakenews Last Checkpoint: 1 minute ago

File Edit View Run Kernel Settings Help Trusted

Code JupyterLab Python 3 (ipykernel)

[1]:
```
import nltk
nltk.data.path.append('/path/to/your/nltk_data')
```

[2]:
```
pip install --upgrade nltk
```

```
Requirement already satisfied: nltk in /Library/Frameworks/Python.framework/Versions/3.13/lib/python3.13/site-packages (3.9.1)
Requirement already satisfied: click in /Library/Frameworks/Python.framework/Versions/3.13/lib/python3.13/site-packages (from nltk) (8.1.8)
Requirement already satisfied: joblib in /Library/Frameworks/Python.framework/Versions/3.13/lib/python3.13/site-packages (from nltk) (1.4.2)
Requirement already satisfied: regex>=2021.8.3 in /Library/Frameworks/Python.framework/Versions/3.13/lib/python3.13/site-packages (from nltk)
(2024.11.6)
Requirement already satisfied: tqdm in /Library/Frameworks/Python.framework/Versions/3.13/lib/python3.13/site-packages (from nltk) (4.67.1)
Note: you may need to restart the kernel to use updated packages.
```

[2]:
```
import pandas as pd

# Load the dataset
df = pd.read_csv("WELFake_Dataset.csv")  # Replace with the correct file name or path

# Display the first few rows
print(df.head())
```

```
   Unnamed: 0                                              title  \
0           0  LAW ENFORCEMENT ON HIGH ALERT Following Threat...
1           1                                                NaN
2           2  UNBELIEVABLE! OBAMA'S ATTORNEY GENERAL SAYS MO...
3           3  Bobby Jindal, raised Hindu, uses story of Chri...
4           4  SATAN 2: Russia unvelis an image of its terrif...

                                                text  label
0  No comment is expected from Barack Obama Membe...      1
1    Did they post their votes for Hillary already?      1
2   Now, most of the demonstrators gathered last ...      1
3  A dozen politically active pastors came here f...      0
4  The RS-28 Sarmat missile, dubbed Satan 2, will...      1
```

[7]:
```
# Check column names
print(df.columns)

# Check for missing values in the 'text' column
print(df['text'].isnull().sum())
```

localhost:8888/notebooks/Fakenews.ipynb

jupyter **Fakenews** Last Checkpoint: 1 minute ago

File Edit View Run Kernel Settings Help — Trusted

Code — JupyterLab — Python 3 (ipykernel)

[3]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import re
import nltk
from nltk.corpus import stopwords
from nltk.tokenize import word_tokenize
from nltk.stem import WordNetLemmatizer
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.model_selection import train_test_split
from sklearn.naive_bayes import MultinomialNB
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix

# Download NLTK resources
nltk.download('punkt')
nltk.download('stopwords')
nltk.download('wordnet')
```

```
[nltk_data] Downloading package punkt to
[nltk_data]     /Users/hassaniddrisu/nltk_data...
[nltk_data]   Package punkt is already up-to-date!
[nltk_data] Downloading package stopwords to
[nltk_data]     /Users/hassaniddrisu/nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
[nltk_data] Downloading package wordnet to
[nltk_data]     /Users/hassaniddrisu/nltk_data...
[nltk_data]   Package wordnet is already up-to-date!
```

[3]:
```
True
```

[11]:
```
# Load the dataset
df = pd.read_csv("WELFake_Dataset.csv")  # Replace with the correct file name or path

# Display the first few rows
print(df.head())

# Check basic info
print(df.info())
```

jupyter **Fakenews** Last Checkpoint: 2 minutes ago

File Edit View Run Kernel Settings Help — Trusted

Code — JupyterLab — Python 3 (ipykernel)

[11]:
```
# Load the dataset
df = pd.read_csv("WELFake_Dataset.csv")  # Replace with the correct file name or path

# Display the first few rows
print(df.head())

# Check basic info
print(df.info())

# Check for missing values
print(df.isnull().sum())

# Drop rows with missing values
df = df.dropna()

# Check the distribution of the target variable
print(df['label'].value_counts())
```

```
   Unnamed: 0                                              title  \
0           0  LAW ENFORCEMENT ON HIGH ALERT Following Threat...
1           1                                                NaN
2           2  UNBELIEVABLE! OBAMA'S ATTORNEY GENERAL SAYS MO...
3           3  Bobby Jindal, raised Hindu, uses story of Chri...
4           4  SATAN 2: Russia unvelis an image of its terrif...

                                                text  label
0  No comment is expected from Barack Obama Membe...      1
1     Did they post their votes for Hillary already?      1
2   Now, most of the demonstrators gathered last ...      1
3  A dozen politically active pastors came here f...      0
4  The RS-28 Sarmat missile, dubbed Satan 2, will...      1
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 72134 entries, 0 to 72133
Data columns (total 4 columns):
 #   Column      Non-Null Count  Dtype
---  ------      --------------  -----
 0   Unnamed: 0  72134 non-null  int64
 1   title       71576 non-null  object
 2   text        72095 non-null  object
 3   label       72134 non-null  int64
dtypes: int64(2), object(2)
```

jupyter Fakenews Last Checkpoint: 3 minutes ago

File Edit View Run Kernel Settings Help Trusted

Code JupyterLab Python 3 (ipykernel)

```
3  A dozen politically active pastors came here f...      0
4  The RS-28 Sarmat missile, dubbed Satan 2, will...      1
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 72134 entries, 0 to 72133
Data columns (total 4 columns):
 #   Column      Non-Null Count  Dtype
---  ------      --------------  -----
 0   Unnamed: 0  72134 non-null  int64
 1   title       71576 non-null  object
 2   text        72095 non-null  object
 3   label       72134 non-null  int64
dtypes: int64(2), object(2)
memory usage: 2.2+ MB
None
Unnamed: 0      0
title         558
text           39
label           0
dtype: int64
label
1    36509
0    35028
Name: count, dtype: int64
```

[12]:

```
# Plot the distribution of the target variable
plt.figure(figsize=(8, 6))
sns.countplot(x='label', data=df)
plt.title("Distribution of Fake (1) vs Real (0) News")
plt.savefig("label_distribution.png")  # Save the figure
plt.show()

# Analyze text length
df['text_length'] = df['text'].apply(len)
plt.figure(figsize=(8, 6))
sns.histplot(df['text_length'], bins=50)
plt.title("Distribution of Text Length")
plt.savefig("text_length_distribution.png")  # Save the figure
plt.show()
```

Distribution of Fake (1) vs Real (0) News

jupyter Fakenews Last Checkpoint: 4 minutes ago

File Edit View Run Kernel Settings Help Trusted

Code JupyterLab Python 3 (ipykernel)

[14]:

```
# Initialize TF-IDF Vectorizer
tfidf = TfidfVectorizer(max_features=5000)  # Limit to 5000 features for simplicity

# Fit and transform the cleaned text
X = tfidf.fit_transform(df['cleaned_text']).toarray()
y = df['label']

# Display the shape of the feature matrix
print(X.shape)
```

```
(71537, 5000)
```

[15]:

```
# Split the data
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Check the shapes
print(X_train.shape, X_test.shape)
```

```
(57229, 5000) (14308, 5000)
```

[16]:

```
# Initialize and train the model
model = MultinomialNB()
model.fit(X_train, y_train)

# Make predictions
y_pred = model.predict(X_test)

# Evaluate the model
print("Accuracy:", accuracy_score(y_test, y_pred))
print("Classification Report:\n", classification_report(y_test, y_pred))

# Plot confusion matrix
plt.figure(figsize=(8, 6))
sns.heatmap(confusion_matrix(y_test, y_pred), annot=True, fmt='d', cmap='Blues')
plt.title("Confusion Matrix")
plt.savefig("confusion_matrix.png")  # Save the figure
plt.show()
```

```
Accuracy: 0.8489656136427174
Classification Report:
```

jupyter **Fakenews** Last Checkpoint: 5 minutes ago

File Edit View Run Kernel Settings Help — Trusted

Code — JupyterLab — Python 3 (ipykernel)

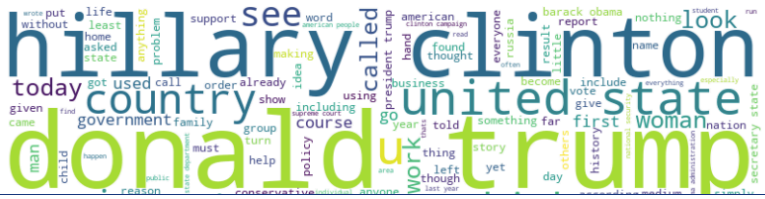
[ ]:
```
with open("classification_report.txt", "w") as f:
    f.write("Accuracy: " + str(accuracy_score(y_test, y_pred)) + "\n\n")
    f.write("Classification Report:\n")
    f.write(classification_report(y_test, y_pred))
```

[18]:
```
from wordcloud import WordCloud

# Word cloud for fake news
fake_text = " ".join(df[df['label'] == 1]['cleaned_text'])
wordcloud_fake = WordCloud(width=800, height=400, background_color='white').generate(fake_text)
plt.figure(figsize=(10, 5))
plt.imshow(wordcloud_fake, interpolation='bilinear')
plt.axis('off')
plt.title("Word Cloud for Fake News")
plt.savefig("wordcloud_fake.png")
plt.show()

# Word cloud for real news
real_text = " ".join(df[df['label'] == 0]['cleaned_text'])
wordcloud_real = WordCloud(width=800, height=400, background_color='white').generate(real_text)
plt.figure(figsize=(10, 5))
plt.imshow(wordcloud_real, interpolation='bilinear')
plt.axis('off')
plt.title("Word Cloud for Real News")
plt.savefig("wordcloud_real.png")
plt.show()
```

Word Cloud for Fake News



jupyter **Fakenews** Last Checkpoint: 6 minutes ago

File Edit View Run Kernel Settings Help — Trusted

Code — JupyterLab — Python 3 (ipykernel)

0 1

[9]:
```
from sklearn.metrics import roc_curve, auc

# Calculate ROC curve
y_pred_prob = model.predict_proba(X_test)[:, 1]
fpr, tpr, thresholds = roc_curve(y_test, y_pred_prob)
roc_auc = auc(fpr, tpr)

# Plot ROC curve
plt.figure(figsize=(8, 6))
plt.plot(fpr, tpr, color='blue', lw=2, label=f'ROC curve (AUC = {roc_auc:.2f})')
plt.plot([0, 1], [0, 1], color='gray', linestyle='--')
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('ROC Curve')
plt.legend(loc='lower right')
plt.savefig("roc_curve.png")
plt.show()
```