Computational Intelligence in Engineering Applications EE-408

Term Project

Department of Electrical Engineering

Additional Details	
Obtained Marks	
Total Marks	
Signature/Comments	

Sr. No	Student Name
1	Laraib Fatima
2	Mian Saad karim

Section : A	**Group:** 03
Project Title: Fake News classification using Gradient Boosting	
Batch : 22-26	**Teacher:** Dr. Sufi Tabassum Gull
Semester : 05	**Date of Submission: December-11-2024**

Department of Electrical Engineering

Contents

Abstract

Fake news is emerging as a serious problem in the digital era and have a great influence on public opinion and cause spread of misinformation on a large scale. This project is based on the development of a robust fake news classification system using Gradient Boosting. This framework uses different techniques for the preprocessing of textual data . These techniques include Vactorization ,Text normalization etc . This GB model is trained to classify real and fake news by recognizing the text patterns and linguistic features in the dataset.

The GB model is evaluated by calculating factors like log loss,precision,accuracy,F1 score, Recall , that demonstrate its effectiveness .Some visualizations like confusion matrices,word frequency plots are used to understand the model's behavior . This process underlines the potential of Gradient Boosting in addressing the challenges caused by fake news, providing strong basis for future advancements in automated misinformation detection systems.

State of the Art in Fake News Classification

Modern fake news classification methods have shown significant advancements through various techniques:

- **Deep Learning Approaches:** The use of advanced neural networks, such as Convolutional Neural Networks (CNNs) and Long Short-Term Memory (LSTM) networks, helps capture semantic and temporal features in textual data.
- **Hybrid Architectures:** Combining models like word embeddings with attention mechanisms has enhanced detection accuracy by focusing on contextually relevant information.
- **Dataset Utilization:** Benchmark datasets like PolitiFact and GossipCop are commonly used to train and evaluate models, enabling robust and comparative analysis of methodologies.
- **Explainability and Interpretability:** There is a growing emphasis on model transparency, improving understanding of how linguistic patterns contribute to fake news identification.

These findings are based on recent developments in fake news detection, as discussed in [1].

1 Problem Background

In today's digital world, fake news is spreading faster than ever across social media and news websites. It's everywhere — fake headlines, misleading stories, and outright lies, all disguised as credible information.It's getting harder to tell real from fake. The consequences are huge: fake news can sway elections, fuel public panic, and mislead millions of people.

Deep learning models like CNNs and Long Short-Term Memory networks are the real heroes here, helping these systems analyze massive amounts of text and spot patterns that humans might miss. But even with all this technology, the battle isn't over. We still face huge challenges in building systems that are not only accurate but also transparent and easy to trust.

This report dives into the latest techniques in fake news detection, examining how these technologies are fighting the war against misinformation and how we can make them even better.

2 Exploratory Data Analysis (EDA)

The dataset consisted of two main datasets: one containing fake news articles and the other with true news articles.. The datasets were preprocessed to remove unnecessary features such as titles, subjects, and dates, and to label the data for training and evaluation.Initially our model was geared towards fake news because the size of the fake.csv file was greater than true.csv. The dataset was acquired form kaggle [3]

2.1 Size of Data

Let's begin our data analysis with size estimation of our datasets.

```
data_fake.shape,data_true.shape
```

```
((23481, 5), (21432, 5))
```

Figure 1: Size of True and Fake Data Sets

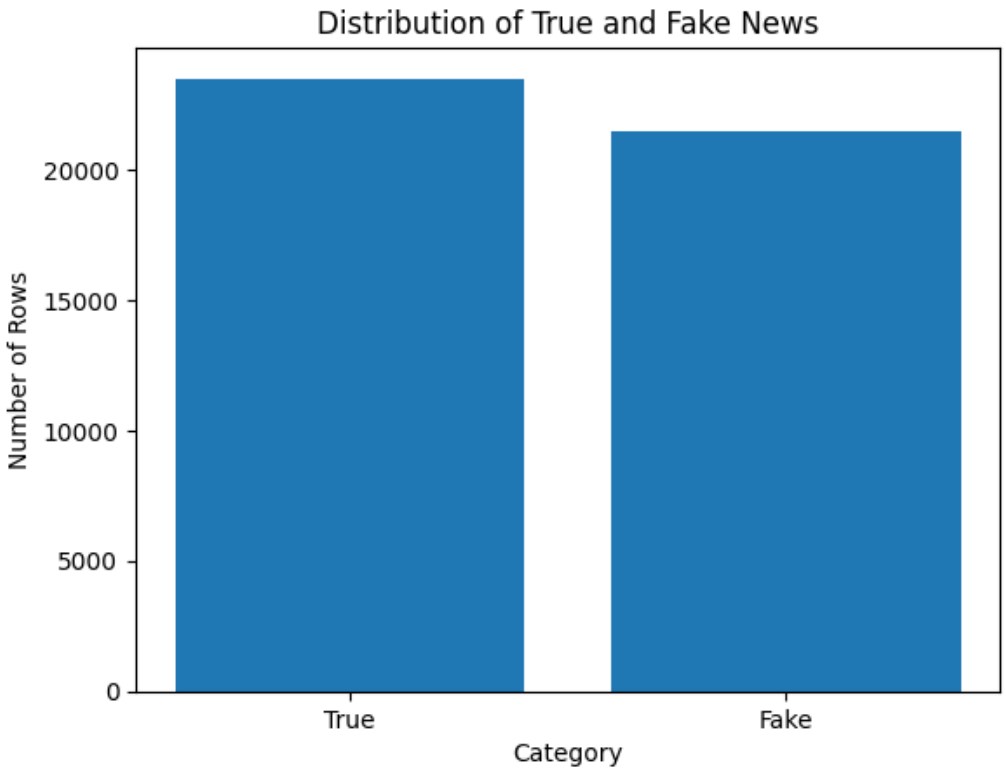

Figure 2: Histogram of True and Fake Data Sets

2.2 Data Description:

We can check data description of using first 1100 rows of True news data set(since ,In our case,these include only politicalnews so there is no unique data here).

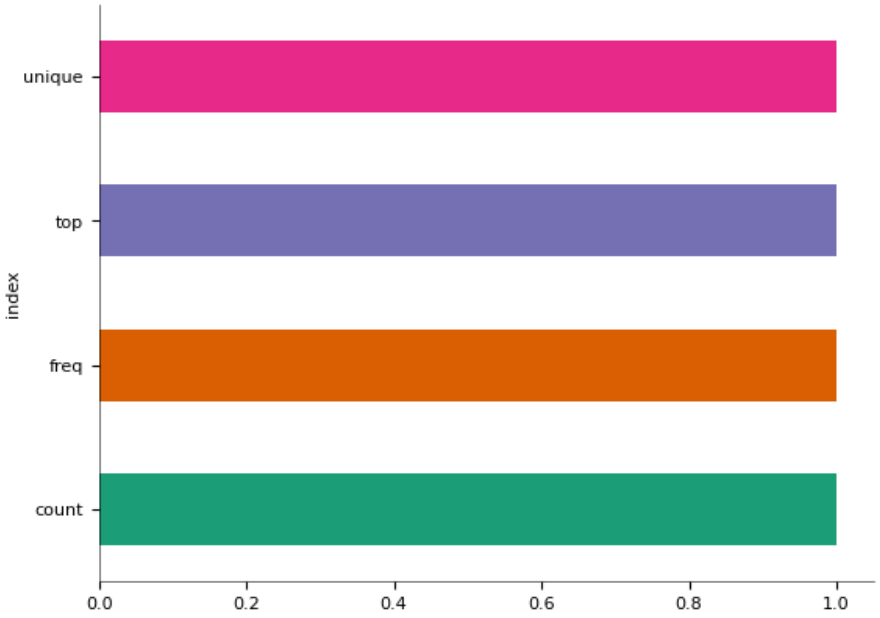

Figure 3: Categorical distributions

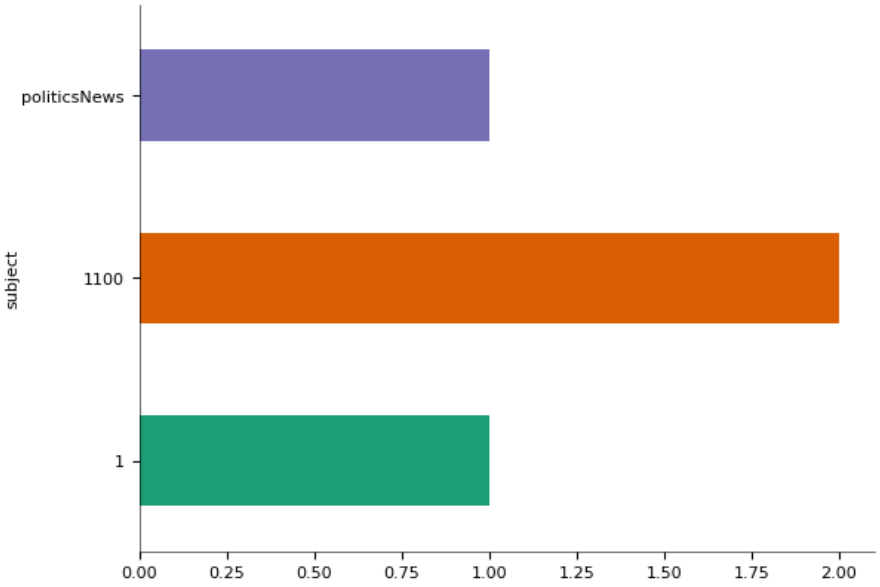

Figure 4: Categorical distributions

2.3 Textual Analysis:

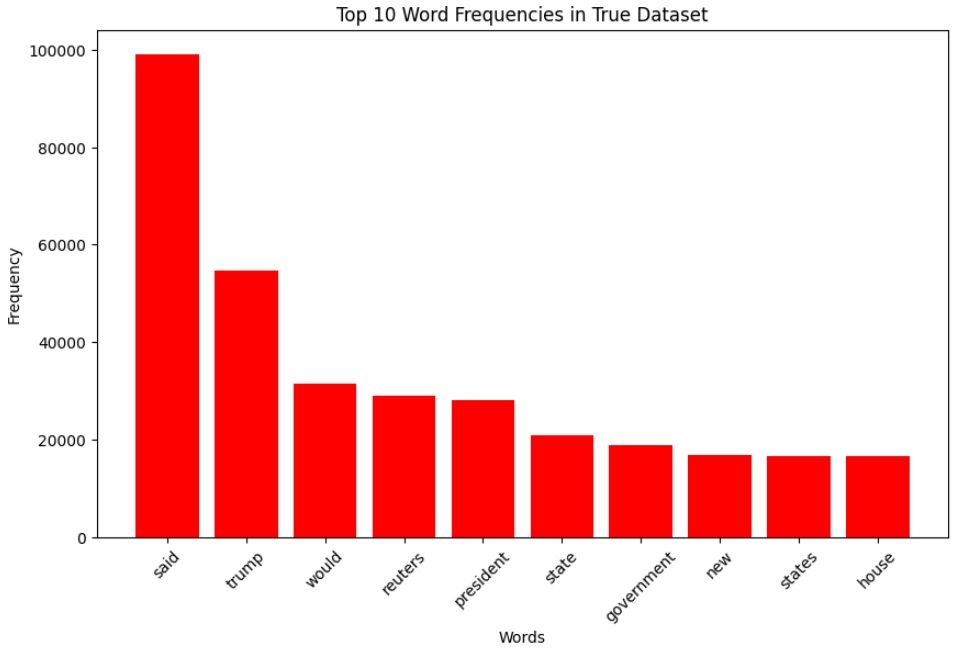

Figure 5: frequency of words in the True dataset

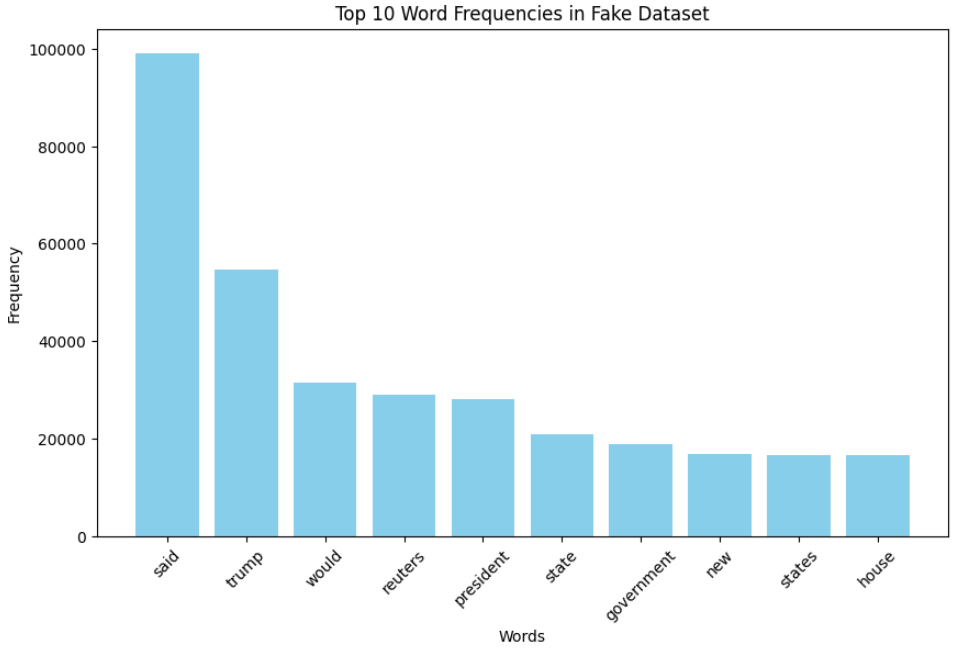

Figure 6: frequency of words in the fake dataset

2.4 Frequency Analysis

This frequency is calculated as:

$$\text{TF}(t, d) = \frac{\text{Frequency of term } t \text{ in document } d}{\text{Total number of terms in document } d}$$

This frequency refered as term frequency is used in calculation of the term frequency-inverse document frequency($TF - IDF$)of words that balances the term frequency and uniqueness of a term. Uniqueness is measured using IDF :

$$\text{IDF}(t) = \log \frac{\text{Total number of documents in the corpus D}}{\text{Number of documents containing t}}$$

TF-IDF:

$$\text{TF-IDF}(t, d) = \text{TF}(t, d) \times \text{IDF}(t)$$

TF-IDF has following significance:

1. Used for Feature Extraction (Convert Textual data into numerical features,as our machine learning ,odels work with numerical data only)
2. Removes noice in data ,As it assigns a low tf-idf value to commmon and unnecessory terms that affect the performance of model in classification.
3. Preparing data to be vectorized

TF-IDF Score of unique words from our datasets:

Table 1: TF-IDF score for frequent words in True Dataset

Word	**TF-IDF Score**
said	0.25
trump	0.15
would	0.10
president	0.10
reuters	0.10
state	0.10
government	0.10
house	0.08
people	0.08
united	0.08

Table 2: TF-IDF score for frequent words in Fake Dataset

Word	**TF-IDF Score**
trump	0.22
said	0.12
people	0.10
president	0.10
one	0.09
obama	0.09
would	0.09
clinton	0.08
like	0.08
donald	0.08

3 Explanation of Selected Classifier

For this project, we chose the Gradient Boosting algorithm because it excels at handling complex data patterns and providing highly accurate predictions. It works by combining multiple weak learners, typically decision trees, into a strong predictive model. Each decision tree attempts to correct the mistakes made by the previous trees, making it a powerful method for improving model accuracy.

To better understand how this process works and how the model improves over time, we have included several visualizations:

3.1 Decision Tree 1

With **gradient boosting**, multiple decision trees are used to improve the model's performance gradually:

- **Weak Learners**: Each decision tree is a weak learner, meaning it doesn't perform well on its own. But by combining many of them, the model becomes stronger.
- **Sequential Improvement**: Trees are added one after the other, with each new tree working on correcting the errors made by the previous tree. This process helps refine the model over time.
- **Gradient Descent**: The term "gradient" refers to using gradient descent to minimize errors step-by-step, making each tree improve the model incrementally.

The first tree in the sequence is just one weak learner that contributes to the overall improvement. See Figure 7 for a visual example.

3.2 Decision Tree 2

Here, we see the second decision tree, which attempts to correct the errors made by the first one. Refer to Figure 8 for details.

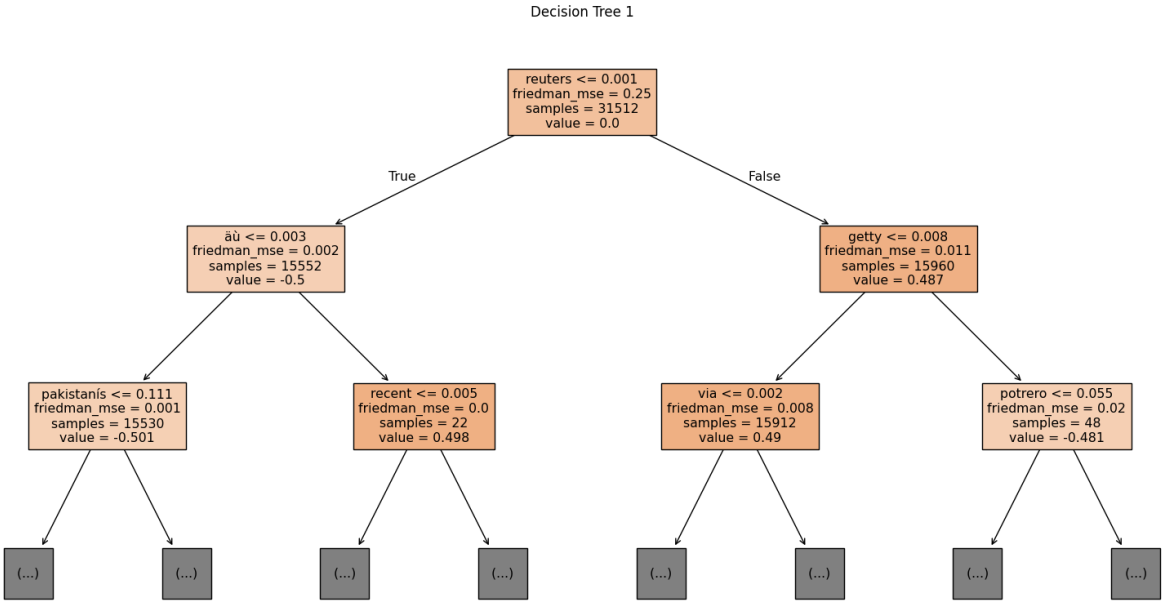

Figure 7: Decision Tree 1

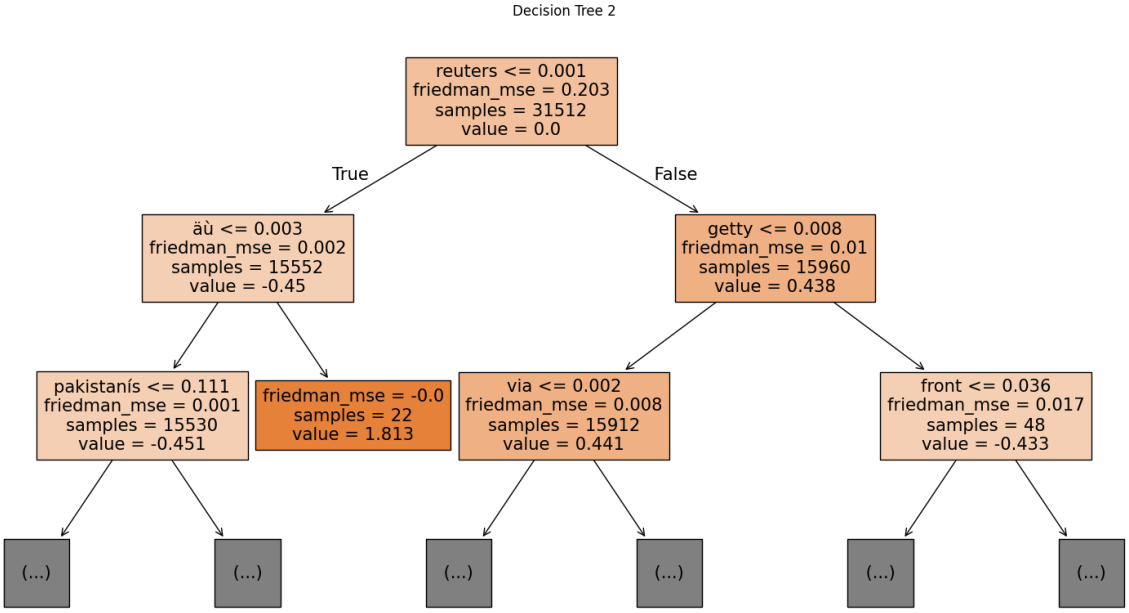

Figure 8: Decision Tree 2

3.3 Log Loss Over Boosting Iterations

This graph shows how the log loss metric decreases as we add more boosting iterations. A lower log loss indicates that the model is getting better at making accurate predictions. Refer to Figure 9 for the graph.

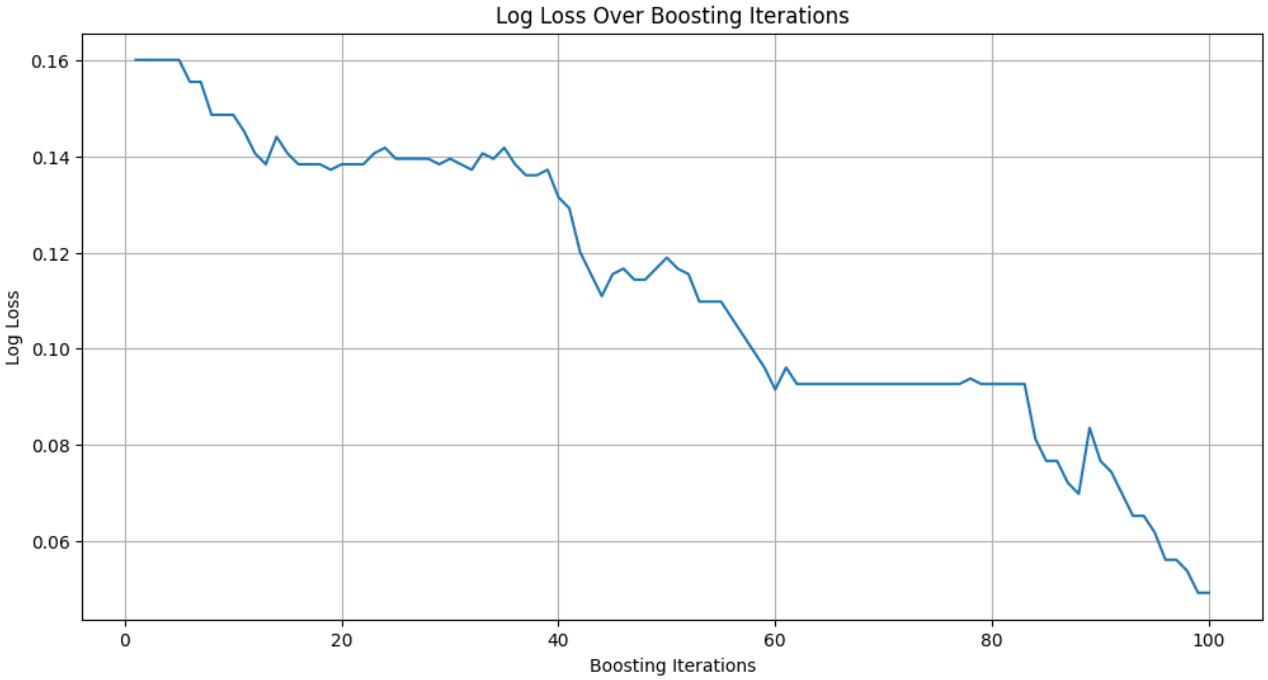

Figure 9: Log Loss Over Boosting Iterations

3.4 Model Performance Over Iterations

In this graph, we track how the model's accuracy, precision, and recall change as boosting iterations progress. Refer to Figure 10 for the visualization.

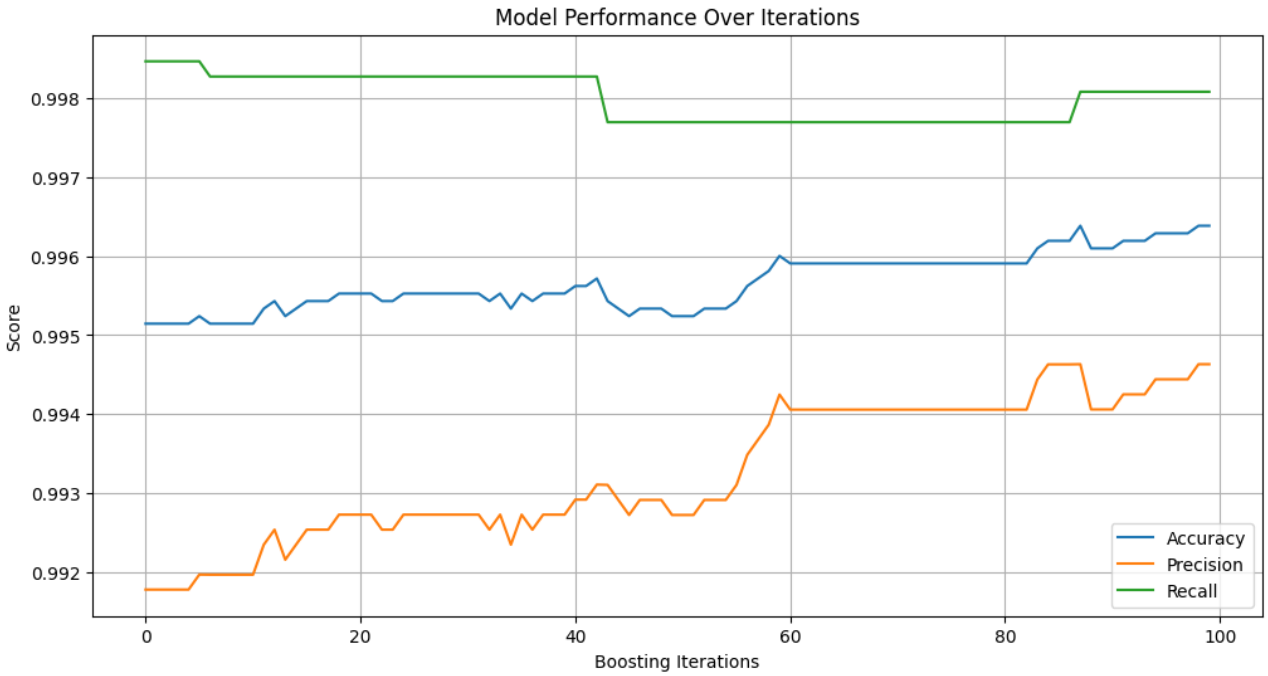

Figure 10: Model Performance Over Iterations

4 Implementation

The following steps were followed in implementing the fake news classification model:

- Data Preprocessing
- Feature Extraction (TF-IDF)
- Model Training (Gradient Boosting)
- Evaluation Metrics

4.0.1 Data Preprocessing

To prepare the dataset for analysis and classification, the following Python code was used which was referenced from [2].

Data Preprocessing Code

```
import pandas as pd, numpy as np
import seaborn as sns, matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score, classification_report
import re, string
# Load datasets
data_fake = pd.read_csv("Fake.csv")
data_true = pd.read_csv("True.csv")
# Assign class labels
data_fake["class"] = 0
data_true["class"] = 1
# Reserve 10 samples for manual testing
data_fake_manual_testing = data_fake.tail(10)
for i in range(23480, 23470, -1):
    data_fake.drop([i], axis=0, inplace=True)
data_true_manual_testing = data_true.tail(10)
for i in range(21416, 21406, -1):
    data_true.drop([i], axis=0, inplace=True)
# Combine datasets
data_merge = pd.concat([data_fake, data_true], axis=0)
data = data_merge.drop(['title', 'subject', 'date'], axis=1)

# Shuffle and reset index
data = data.sample(frac=1)
data.reset_index(inplace=True)
data.drop(["index"], axis=1, inplace=True)
```

4.0.2 Text Cleaning

A custom function was implemented to clean the text by removing unwanted characters, punctuation, and irrelevant data:

	text	class
0	WASHINGTON (Reuters) - The U.S. Congress is st...	1
1	Tune in to the Alternate Current Radio Network...	0
2	Donald Trump s wall is a stupid, money-wasting...	0
3	As we know, the final United Nations General A...	0
4	21st Century Wire says The war between the Whi...	0
5	SAN FRANCISCO (Reuters) - Alphabet Inc's Googl...	1
6	During Wednesday s Presidential Town Hall on C...	0
7	MANILA (Reuters) - The Philippines Congress I...	1

Figure 11: Illustration of the data preprocessing steps for fake news classification.

Text cleaning

```
def wordopt(text):
    text = text.lower()
    text = re.sub('\[.*?\]', '', text)
    text = re.sub("\\W", " ", text)
    text = re.sub('https?://\S+|www\.\S+', '', text)
    text = re.sub('<.*?>+', '', text)
    text = re.sub('[%s]' % re.escape(string.punctuation), '', text)
    text = re.sub('\n', '', text)
    text = re.sub('\w*\d\w*', '', text)
    return text

# Apply cleaning function
data['text'] = data['text'].apply(wordopt)

# Split data into features (X) and labels (y)
x = data['text']
y = data['class']
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.25)
```

4.0.3 Feature Extraction

The TF-IDF vectorizer was used to transform textual data into numerical form:

	text	class
0	washington reuters the u s congress is st...	1
1	tune in to the alternate current radio network...	0
2	donald trump s wall is a stupid money wasting...	0
3	as we know the final united nations general a...	0
4	century wire says the war between the white h...	0
5	san francisco reuters alphabet inc s googl...	1
6	during wednesday s presidential town hall on c...	0
7	manila reuters the philippines congress l...	1

Figure 12: Illustration of the text cleaning process for fake news classification.

Feature Extraction

```
from sklearn.feature_extraction.text import TfidfVectorizer

# TF-IDF Vectorization
vectorization = TfidfVectorizer()
xv_train = vectorization.fit_transform(x_train)
xv_test = vectorization.transform(x_test)
```

4.0.4 Model Training and Evaluation

A Gradient Boosting Classifier was used to train the model and evaluate its performance:

Model Training

```
from sklearn.ensemble import GradientBoostingClassifier
from sklearn.metrics import classification_report
# Train the Gradient Boosting model
GB = GradientBoostingClassifier(random_state=0)
GB.fit(xv_train, y_train)
# Predict and evaluate the model
predict_gb = GB.predict(xv_test)
print("Classification Report:\n", classification_report(y_test, predict_gb))
```

4.0.5 Manual Testing

The model was tested on individual samples using a custom manual testing function:

	precision	recall	f1-score	support
0	1.00	0.99	1.00	5870
1	0.99	1.00	0.99	5352
accuracy			0.99	11222
macro avg	0.99	1.00	0.99	11222
weighted avg	0.99	0.99	0.99	11222

Figure 13: Illustration of the model training and evaluation process for fake news classification.

setting up manual testing

```
def output_lable(n):
    if n == 0:
        return "Fake News"
    elif n == 1:
        return "Not A Fake News"

def manual_testing(news):
    testing_news = {"text": [news]}
    new_def_test = pd.DataFrame(testing_news)
    new_def_test["text"] = new_def_test["text"].apply(wordopt)
    new_x_test = new_def_test["text"]
    new_xv_test = vectorization.transform(new_x_test)
    pred_GB = GB.predict(new_xv_test)
    return print("\n\nGB Prediction: {} ".format(output_lable(pred_GB[0])))

# Example of manual testing
news = str(input("Enter a news headline: "))
manual_testing(news)
```

```
Pieas is one of the top universities of pakistan


GB Prediction: Not A Fake News
```

Figure 14: Illustration of the manual testing process for fake news classification.

```
PMLN is the greatest party of pakistan


GB Prediction: Fake News
```

Figure 15: Illustration of the manual testing process for fake news classification.

5 Results and Discussion

5.1 Model Performance

The Gradient Boosting Classifier was evaluated using metrics such as accuracy, precision, recall, and F1-score. The classification report below provides a detailed overview of the model's performance:

	precision	recall	f1-score	support
0	0.99	0.99	0.99	5246
1	0.99	0.99	0.99	5286
accuracy			0.99	10532
macro avg	0.99	0.99	0.99	10532
weighted avg	0.99	0.99	0.99	10532

The model achieved an overall accuracy of 99%, demonstrating exceptional performance in distinguishing between fake and real news. Precision and recall values for both classes are nearly perfect, which indicates that the model rarely makes incorrect predictions.

5.2 Confusion Matrix

To further illustrate the model's performance, the confusion matrix in Figure 16 highlights the breakdown of predictions.

The confusion matrix provides the following insights:

- **True Positives (TP)**: The model correctly identified 5,256 instances of "Not Fake News."
- **True Negatives (TN)**: It correctly labeled 5,216 instances of "Fake News."
- **False Positives (FP)**: 30 instances of "Fake News" were mistakenly classified as "Not Fake News."
- **False Negatives (FN)**: Only 3 instances of "Not Fake News" were incorrectly labeled as "Fake News."

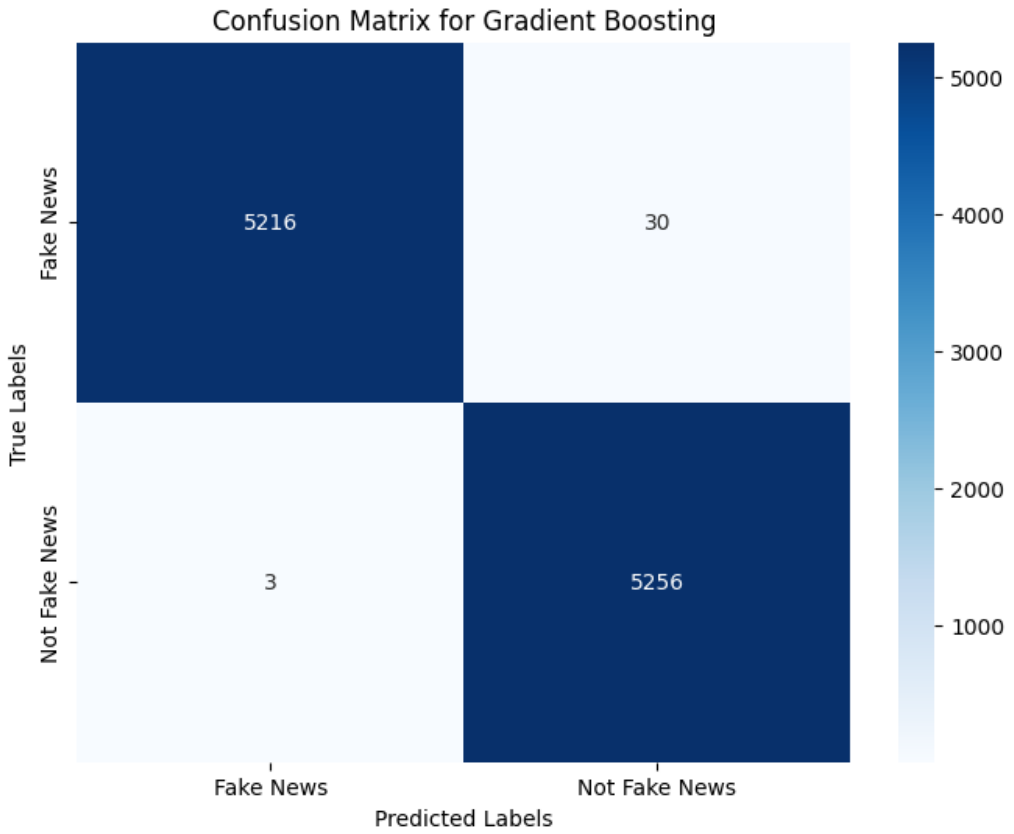

Figure 16: Confusion Matrix for Gradient Boosting Classifier.

5.3 Discussion

The Gradient Boosting Classifier shows impressive accuracy and F1-scores, which demonstrate its ability to effectively balance precision and recall. This makes it a reliable tool for distinguishing between fake and real news. The confusion matrix (Figure 16) shows almost flawless performance, with very few errors in both false positives and false negatives, reflecting its strong classification power.

The log loss graph (Figure 9) illustrates a steady decline across boosting iterations, indicating that the model is gradually learning and reducing errors. As the curve flattens towards the end, it suggests that the model is converging well and has been thoroughly trained.

Furthermore, the performance metrics graph (Figure 10) shows consistently high accuracy, precision, and recall values throughout the process, with minor improvements in later iterations. This highlights the model's stability and robustness, even as it becomes more complex.

6 Conclusions and Comments

This project highlighted the effectiveness of Gradient Boosting for fake news detection. By applying solid data preprocessing techniques and using TF-IDF for feature extraction, the model achieved excellent performance, with high accuracy, precision, recall, and F1-score. Visualizations such as confusion matrices, word frequency plots, and performance graphs provided valuable insights into the model's behavior and its overall accuracy.

The use of Gradient Boosting was particularly beneficial because it improves predictions through the combination of multiple weak learners. The consistent reduction in log loss across iterations further confirmed the model's ability to adapt and identify complex patterns in the data.

Key takeaways from the project include:

- Effective preprocessing and text cleaning are crucial for improving data quality.
- TF-IDF is a powerful tool for converting text into meaningful numerical features for classification.
- Gradient Boosting excels at recognizing patterns that differentiate fake from real news.

While the project was successful, there are areas for future enhancement, such as:

- Expanding the dataset to include a wider range of news sources for better generalization.
- Adding features like metadata or social media interactions to provide richer context for classification.
- Experimenting with other algorithms, such as deep learning models, to compare performance.

In conclusion, this project provides a strong foundation for building automated fake news detection systems. With improvements and further research, the model could play a significant role in combating misinformation in today's digital landscape.

References

[1] Fady Achour Rime Hussein Amir Hajj Hazem. Antoun, Wissam Baly. State of the art models for fake news detection tasks. http://bit.ly/3BmRRkZ. Accessed: 2024-12-11.

[2] GeeksforGeeks. Fake news detection using machine learning. https://www.geeksforgeeks.org/fake-news-detection-using-machine-learning/. Accessed: 2024-12-11.

[3] Kaggle. Fake news detection dataset. https://www.kaggle.com/code/therealsampat/fake-news-detection/input. Accessed: 2024-12-11.

7 Appendix

The code below performs various text-based data analysis and visualization tasks on two datasets, True.csv and Fake.csv the, figures obtained can be observed at (Figure 6) and (Figure 5)

Graph Generation and Data Analysis

```
file_path = "True.csv"  # Replace with your file path
data = pd.read_csv(file_path)
text_data = data['text']
stop_words = list(stopwords.words('english'))
vectorizer = CountVectorizer(stop_words=stop_words)
word_counts = vectorizer.fit_transform(text_data)
word_freq = word_counts.toarray().sum(axis=0)
vocab = vectorizer.get_feature_names_out()

freq_df = pd.DataFrame({'Word': vocab, 'Frequency': word_freq})
freq_df = freq_df.sort_values(by='Frequency', ascending=False)

plt.figure(figsize=(10, 6))
plt.bar(freq_df['Word'][:10], freq_df['Frequency'][:10], color='red')
plt.xlabel('Words')
plt.ylabel('Frequency')
plt.title('Top 10 Word Frequencies in True Dataset')
plt.xticks(rotation=45)
plt.show()

file_path = "Fake.csv"  # Replace with your file path
data = pd.read_csv(file_path)

text_data = data['text']

stop_words = list(stopwords.words('english'))
vectorizer = CountVectorizer(stop_words=stop_words)
word_counts = vectorizer.fit_transform(text_data)
word_freq = word_counts.toarray().sum(axis=0)
vocab = vectorizer.get_feature_names_out()

freq_df = pd.DataFrame({'Word': vocab, 'Frequency': word_freq})
freq_df = freq_df.sort_values(by='Frequency', ascending=False)
plt.figure(figsize=(10, 6))
plt.bar(freq_df['Word'][:10], freq_df['Frequency'][:10], color='skyblue')
plt.xlabel('Words')
plt.ylabel('Frequency')
plt.title('Top 10 Word Frequencies in Fake Dataset')
plt.xticks(rotation=45)
plt.show()
```