FAKE NEWS CLASSIFICATION WITH AI

A machine learning technique for detecting fake news

Kenneth Nyangweso
KENNETH NYANGWESO
DATA SCIENTIST | MACHINE LEARNING ENGINEER

FAKE NEWS CLASSIFICATION WITH NLP

BUSINESS UNDERSTANDING:

Misinformation is spreading faster than ever on social media and news websites, affecting public opinion, politics, and societal trust. Manual verification of news is slow, expensive, and error-prone. This project **builds an automated system to classify news articles as fake or real**, helping users and organizations identify misinformation reliably.

PROJECT OVERVIEW:

This project leverages **Natural Language Processing (NLP)** and **transformer-based models** for fake news detection. The workflow includes:

1. Data Understanding : Understanding the dataset in-terms of column , row, and statistical distribution
2. Data Cleaning and Preprocessing : cleaning, tokenization, handling very large news articles (multi-paragraph).
3. Exploratory Data Analysis (EDA) : understanding text distributions, class balance, and article lengths.
4. Modeling : Training multiple models i.e machine learning models, neural networks and transformer-based models
5. Evaluation : Evaluate model performance using classifications metrics such accuracy and F1-Score
6. Deployment : Using API servers for real-time predictions

PROBLEM STATEMENT:

Manual verification of news is inefficient and error-prone. Users and organizations need an **automated, scalable method** to detect fake news accurately. The challenge is detecting both short and long news articles while providing confidence scores.

OBJECTIVES:

1. Compare multiple models (Naive Bayes, SVM, LSTM, BiLSTM, Distil BERT) for fake news classification.
2. Train models to handle large, multi-paragraph news articles.
3. Build a Fast API backend for serving predictions.
4. Create a simple React frontend for user interaction.
5. Evaluate performance using accuracy, F1-score, and prediction confidence.

METRICS OF SUCCESS:

- Accuracy ≥ 95% for transformer-based models
- F1-score ≥ 95% for high reliability
- Correctly classifying multi-paragraph news articles
- Prediction latency < 1 second per article
- User-friendly API and frontend interface

DATA UNDERSTANDING:

The dataset was sourced from the Kaggle website. It is called WELFake which classifies international news as either real or fake

The dataset consists of 72,134 entries and 4 columns. The column names are as follows:

- Unnamed - ID
- Title – Title of the news article
- Text – The news article itself
- Label – Classification indication in integer form where 0 is fake and 1 is real

DATA CLEANING AND PREPROCESSING:

In this phase I begun cleaning of the dataset by:

- Dropping the Unnamed column as it was not necessary for the analysis
- Handling missing values by dropping all rows with missing values
- Combined the title column and text column as one for easier analysis and more insights
- Text cleaning i.e tokenization, lemmatization, removal of stop words and removal of punctuation signs and special characters

Text cleaning definitions :

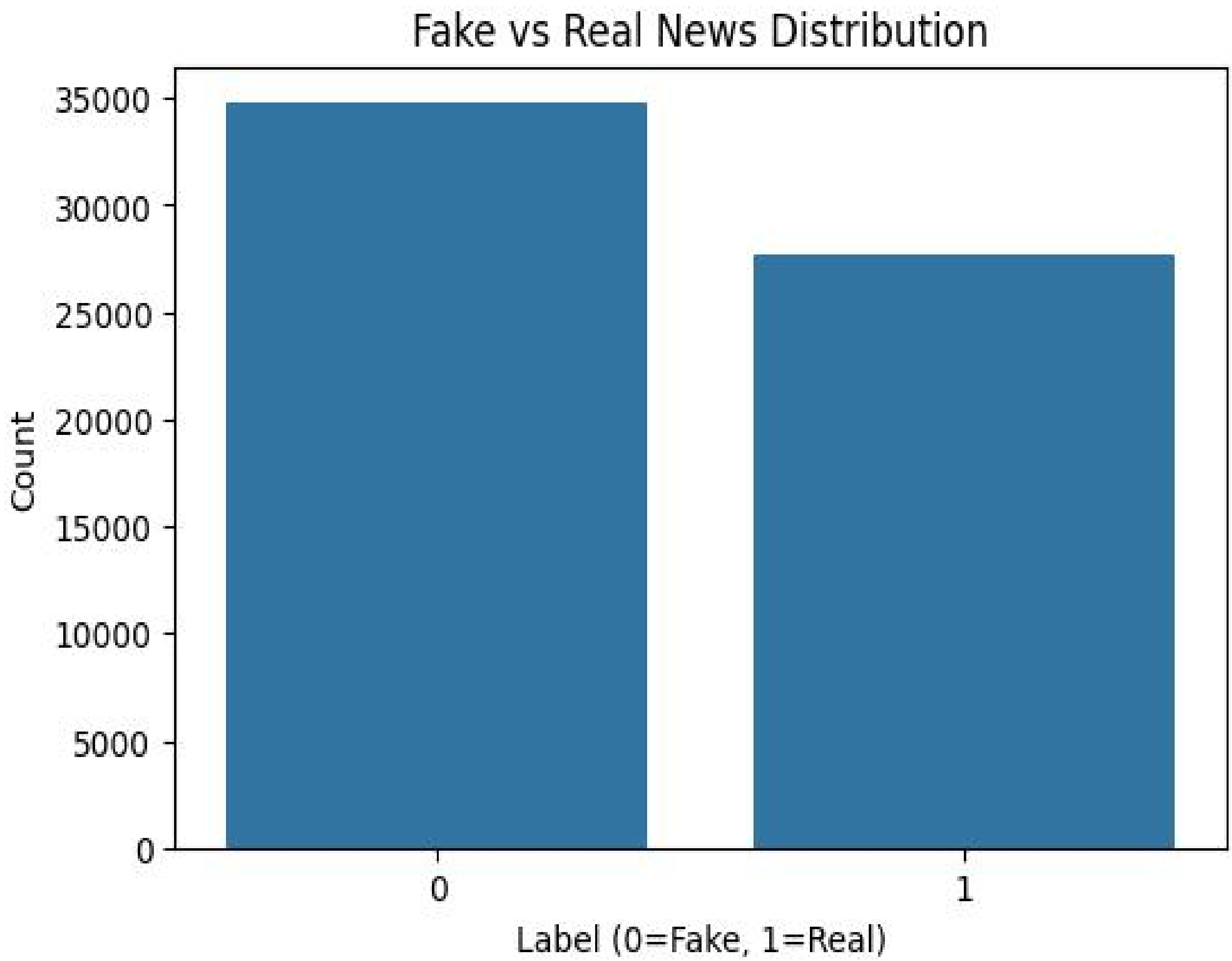
1. Tokenizaton – Process of splitting texts into smaller units called tokens
2. Stop words – Words that occur frequently in a language and do not carry much meaning on their own examples are ‘a’, ‘the’ etc.
3. Lemmatization – It is the processes of reducing a word to its base form for instance studying to study

Preprocessing :

The techniques used were:

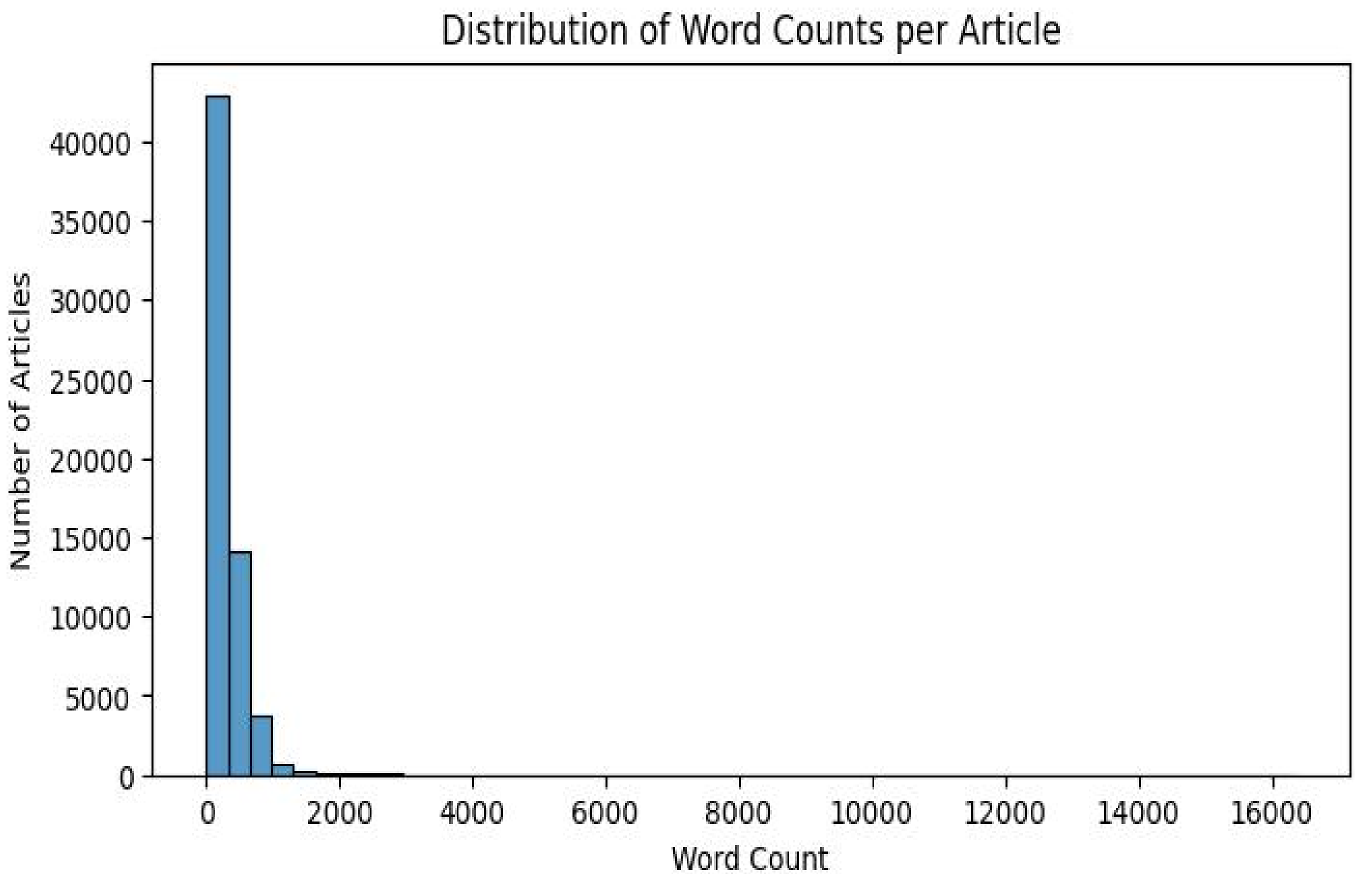
1. Term Frequency – Inverse Document Frequency (TF-IDF) – This a combination of two individual metrics TF and IDF. TF – Is a measure of how frequently a term appears in a document and IDF is a measure of rare a common term occurs in the entire corpus
2. Auto Tokenizer – Is a utility in Hugging Face Transformers Library that automatically loads the correct tokenizer for any pre-trained model without the needing to know the tokenizer class.

EXPLORATORY DATA ANALYSIS:



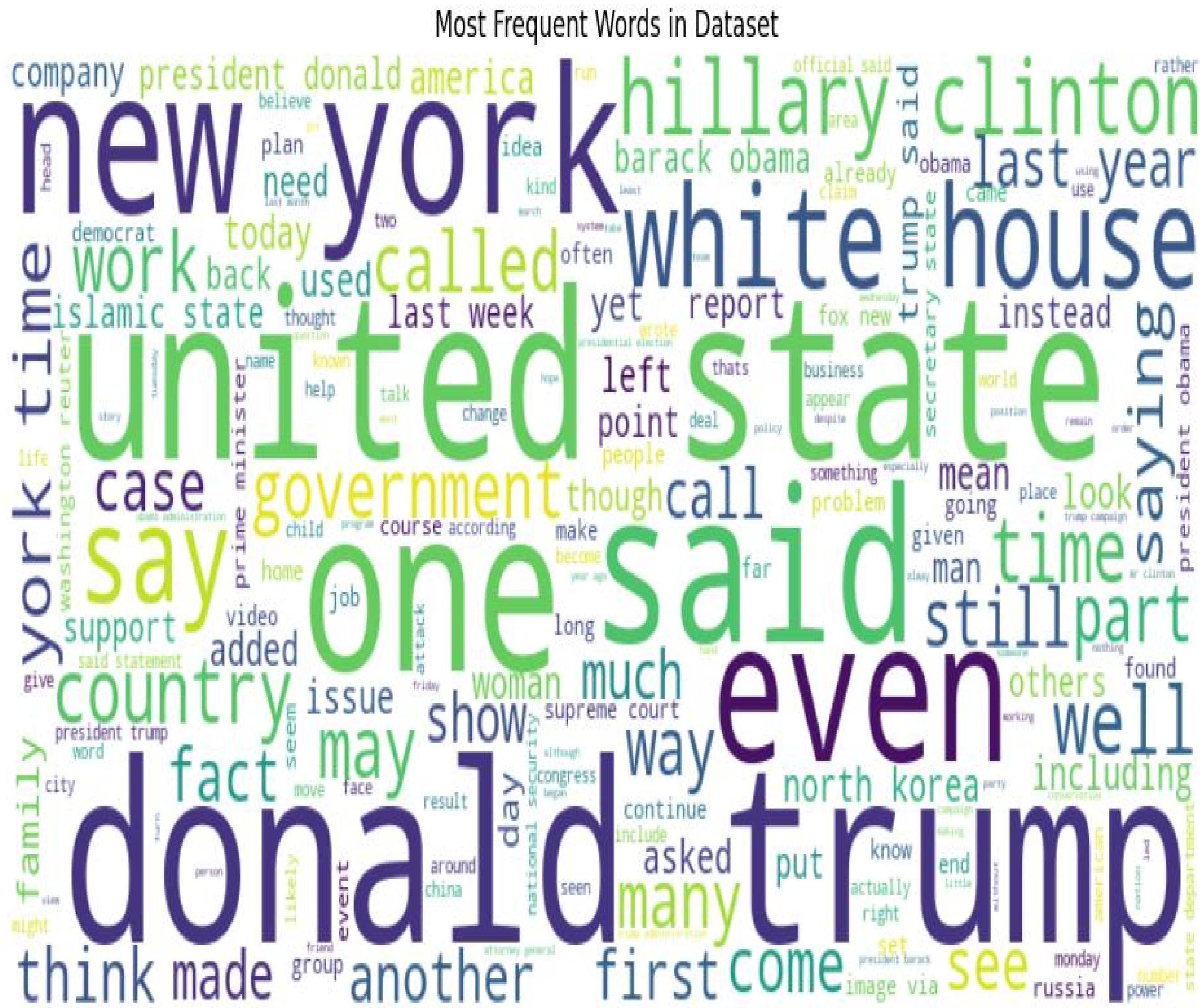
Observations:

- **Imbalanced Classes:** "Fake News" (0) significantly outweighs "Real News" (1).
- **Bias Risk:** Models might naturally lean toward predicting "Fake" due to the higher frequency.
- **Metrics:** Accuracy may be misleading; prioritize **F1-Score** or **Precision-Recall** to account for the gap.



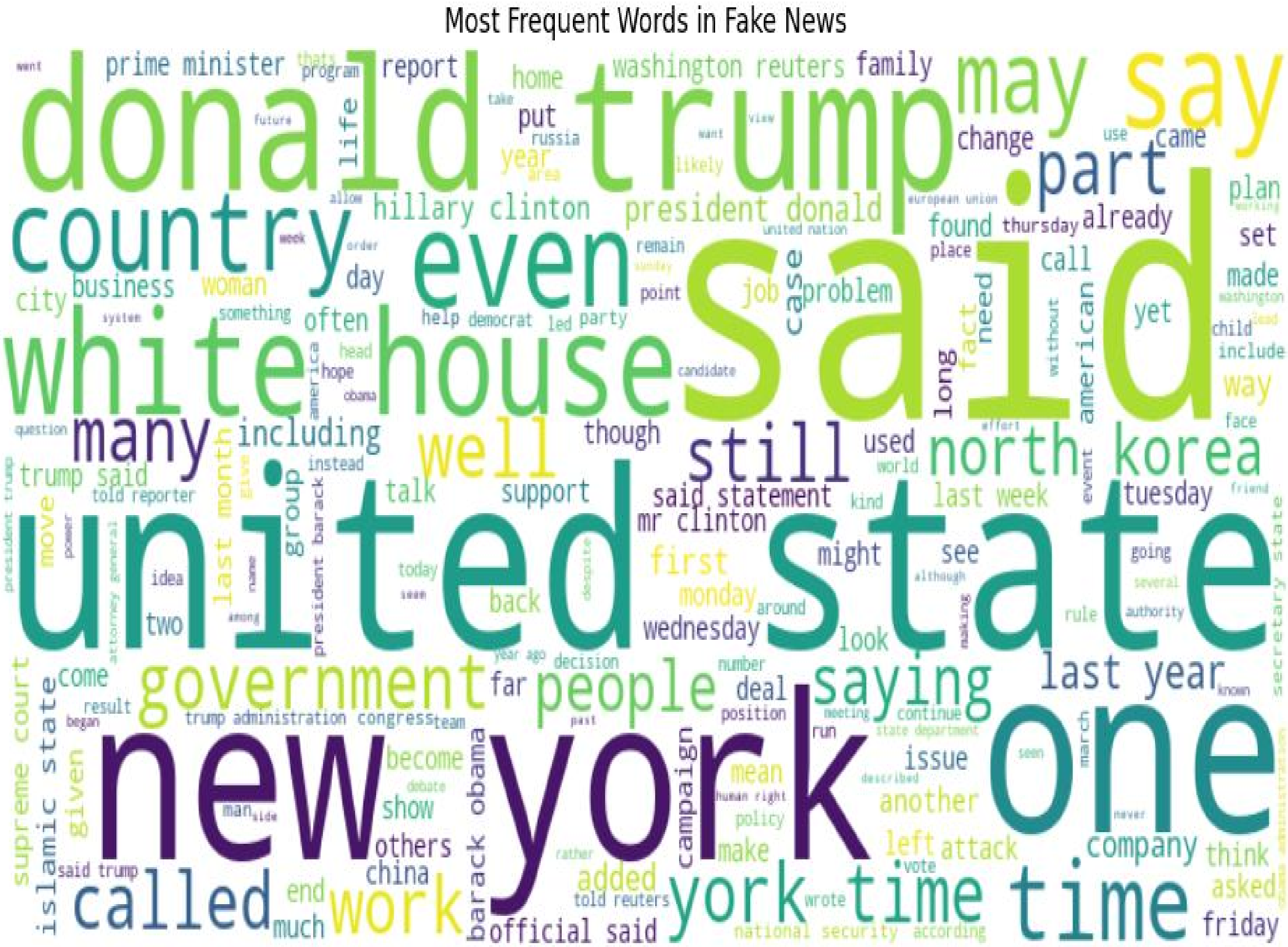
Observations:

- **Skewed Distribution:** Highly right-skewed; the vast majority of articles are short, concentrated under **1,000 words**.
- **Outliers:** A small number of extreme outliers extend beyond **16,000 words**.



Observations:

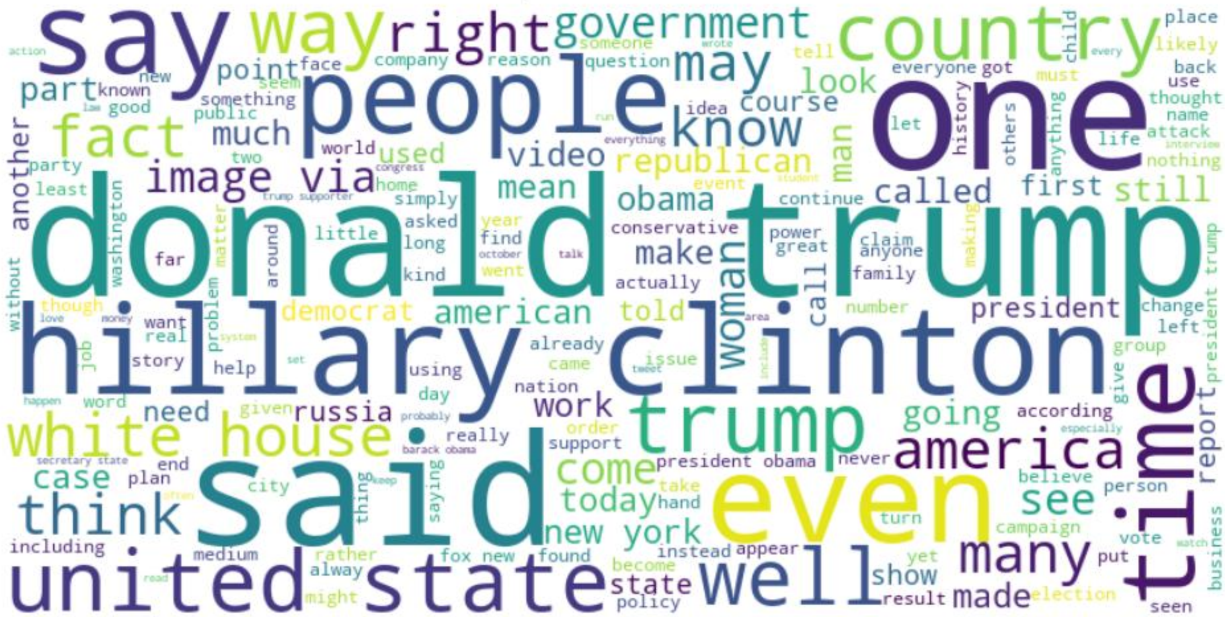
- **Political Focus:** Both categories are dominated by US politics, specifically terms like **"Donald Trump," "Hillary Clinton,"** and **"White House."**
- **Neutral Language:** The most frequent words across both are functional terms like **"said," "one,"** and **"state,"** suggesting similar reporting styles on the surface.



Observations:

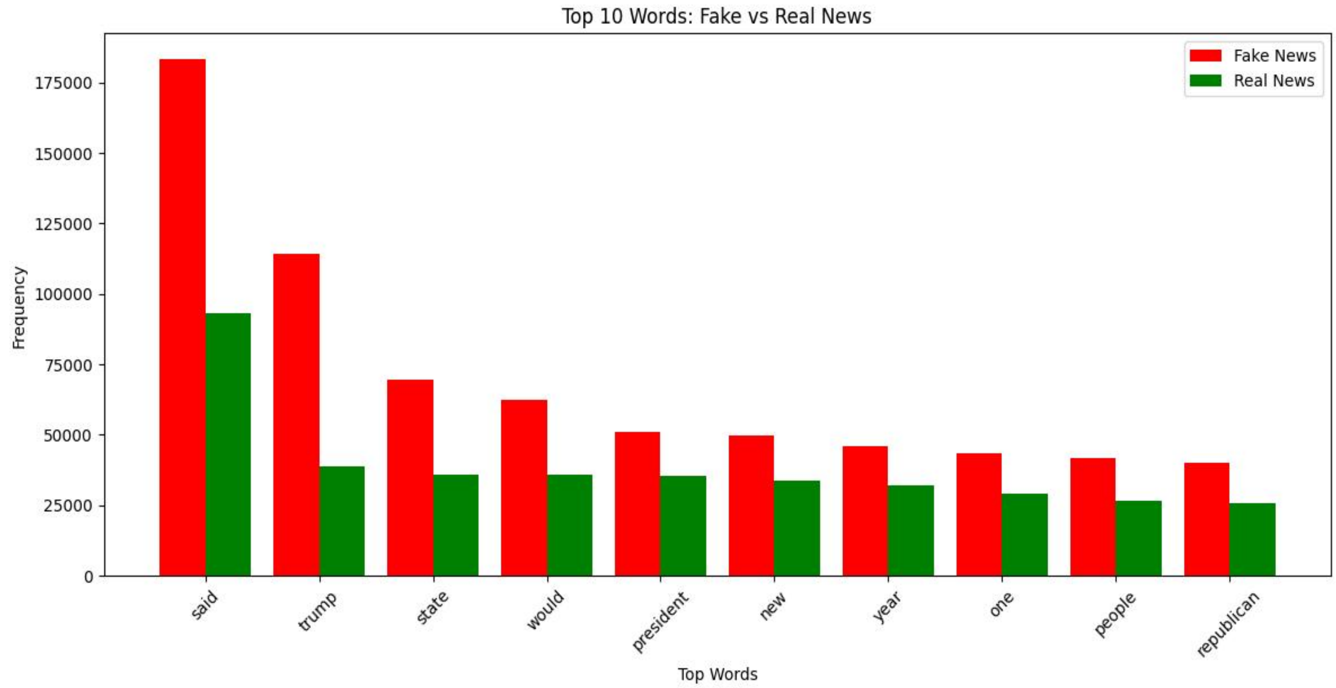
- **Dominant Terms:** The largest words, indicating the highest frequency, include **"donald," "trump," "said," "one," "new,"** and **"york"**.
- **Political Focus:** Many high-frequency terms are related to United States politics, such as **"hillary clinton," "white house," "obama," "government,"** and **"united state"**.

Most Frequent Words in Real News



Observations:

- **Central Figures & Entities:** The most prominent names are "donald trump," "clinton," "hillary," and "obama". Other major entities include the "white house," "republican," and "democrat."
- **Geographic Focus:** Frequent references to "united state," "new york," and "america" indicate that this dataset is primarily focused on U.S. national and local reporting.



Observations:

Higher Volume in Fake News: The word "said" appears nearly twice as often in fake news (~180k) compared to real news (~95k).

Consistent Ranking: The top keywords (Trump, state, president) follow a similar frequency pattern in both classes, though fake news consistently shows higher raw counts across all top 10 words.

MODELING:

For the modeling phase I used numerous models i.e machine learning models, neural networks and HuggingFace transformers

1. Machine Learning Models:
    - Naïve Bayes
    - Support Vector Machine
2. Neural Networks:
    - Long Short-Term Memory (LSTM)
    - Bi-directional Long Short-Term Memory (Bi-LSTM)

3. HuggingFace Transformer:
    - Distil BERT

EVALUATION:

For this project the evaluation metrics used are:

- Accuracy
- F1-score

For the visual I used:

- Confusion Matrix
- ROC-AUC Curve

Results:

MODEL/METRIC	Naïve Bayes	SVM	LSTM	BiLSTM	Distil BERT
Accuracy (%)	85	96	91	96	98
F1-Score(%)	84	96	90	95	98

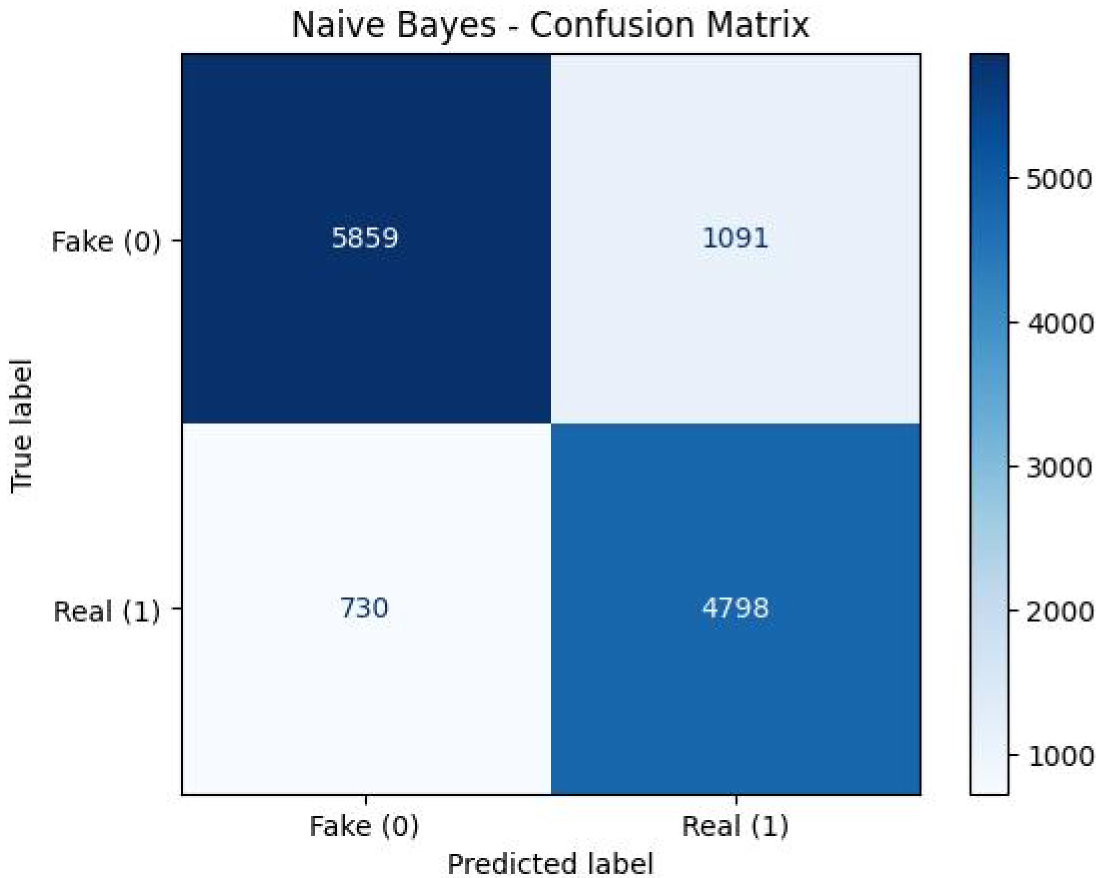
Model Performance Ranking :

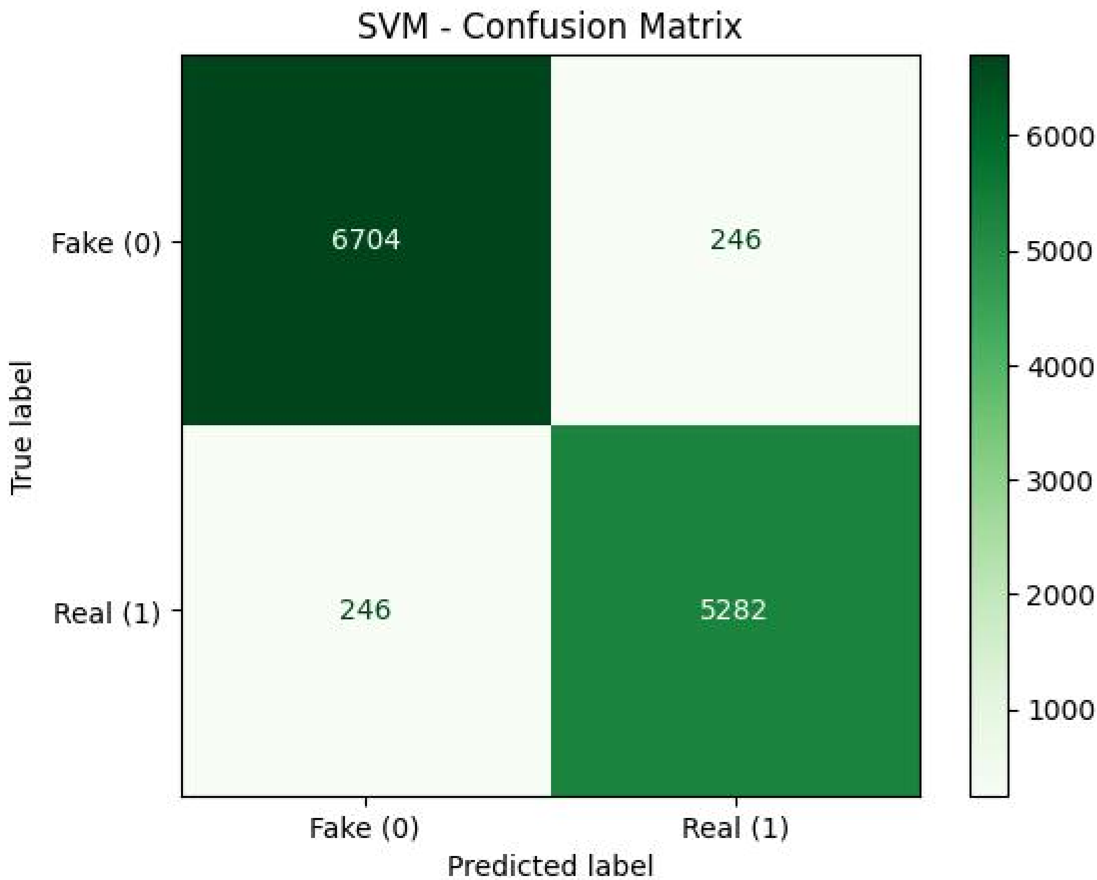
- **Best Overall: DistilBERT** leads with **98%** in both metrics, showing the superior power of transformer-based contextual learning.
- **High Efficiency: SVM** and **BiLSTM** are tied for second at **96%** accuracy, though SVM maintains a slightly higher F1-score than BiLSTM.
- **Weakest Link: Naïve Bayes** is the lowest performer (**85%**), struggling to capture complex patterns compared to deep learning methods.

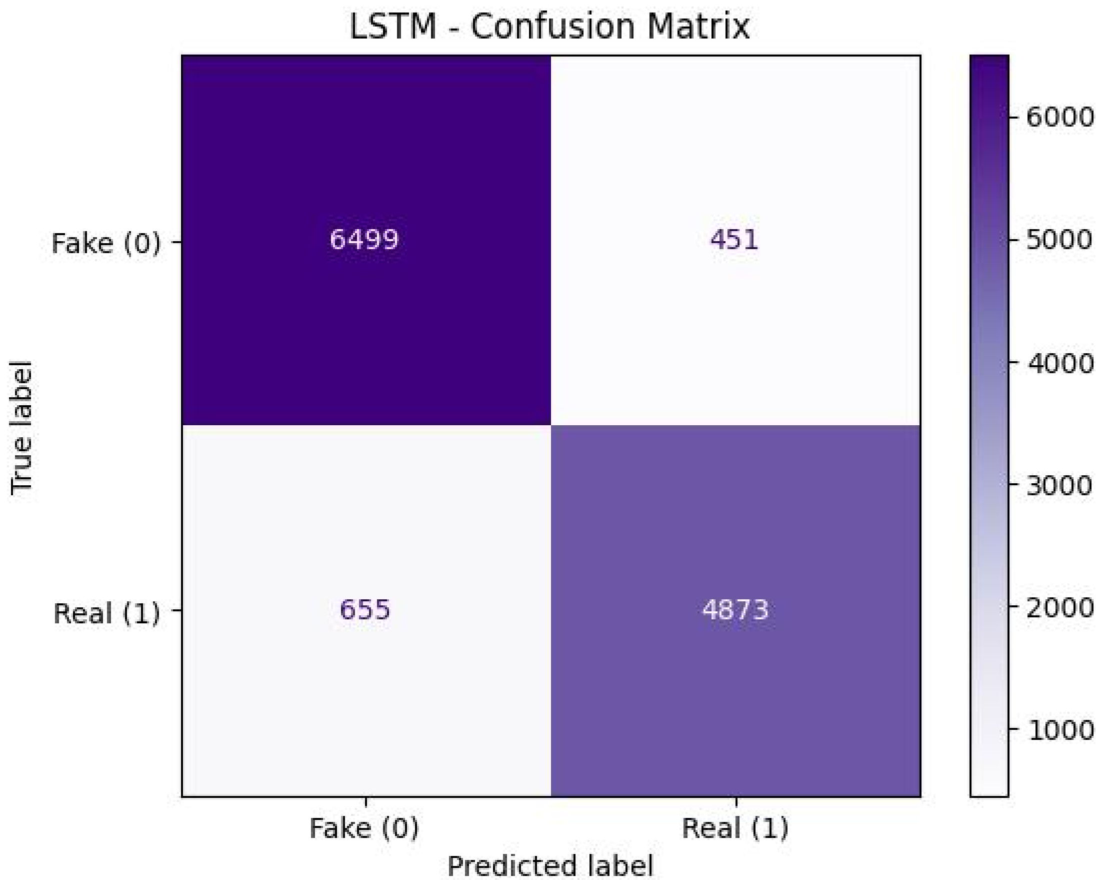
Key Comparisons

- **Architectural Impact:** The **BiLSTM** (96%) significantly outperforms the standard **LSTM** (91%), proving that reading text in both directions is crucial for news classification.
- **Classical vs. Deep Learning:** The **SVM** (96%) surprisingly matches the more complex **BiLSTM**, suggesting it is a highly efficient, lower-resource alternative for this specific dataset.
- **Metric Stability:** Across all models, **Accuracy and F1-Score** are nearly identical (within 1%), indicating that the models are performing consistently across both "Real" and "Fake" classes without heavy bias.

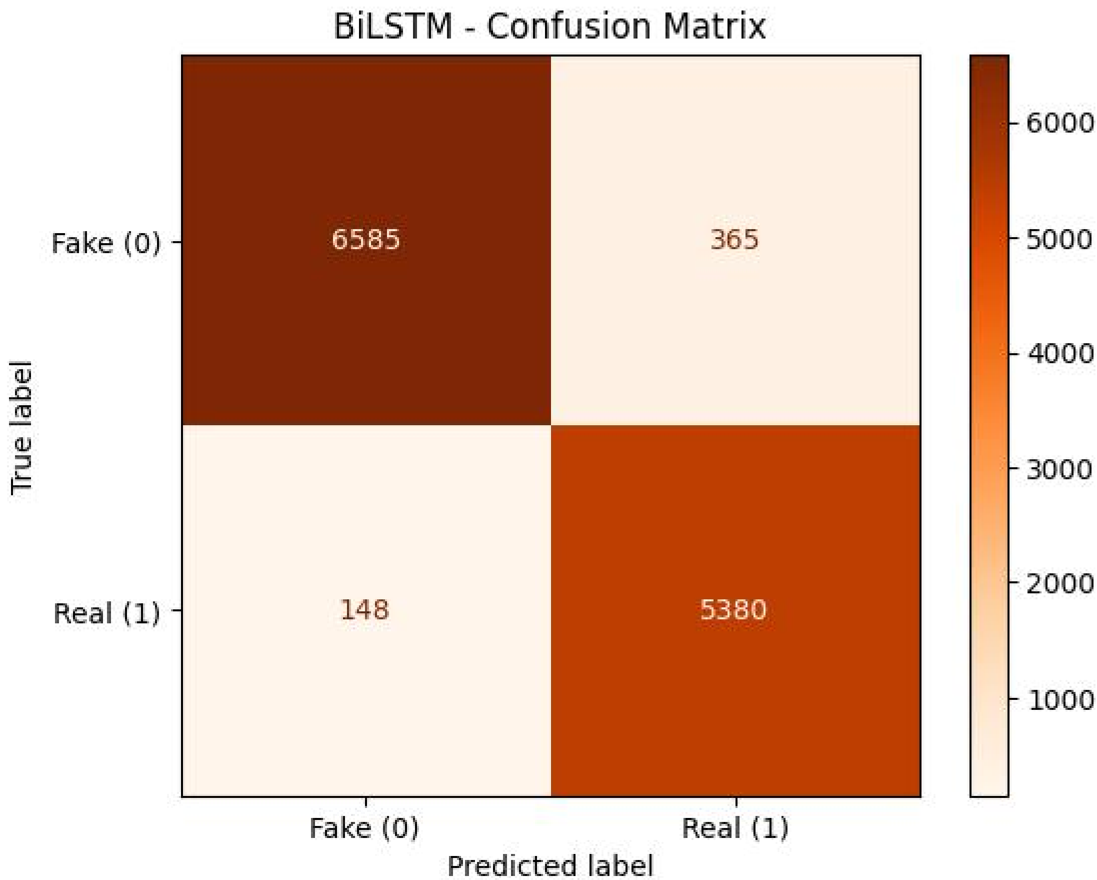
Confusion Matrices:



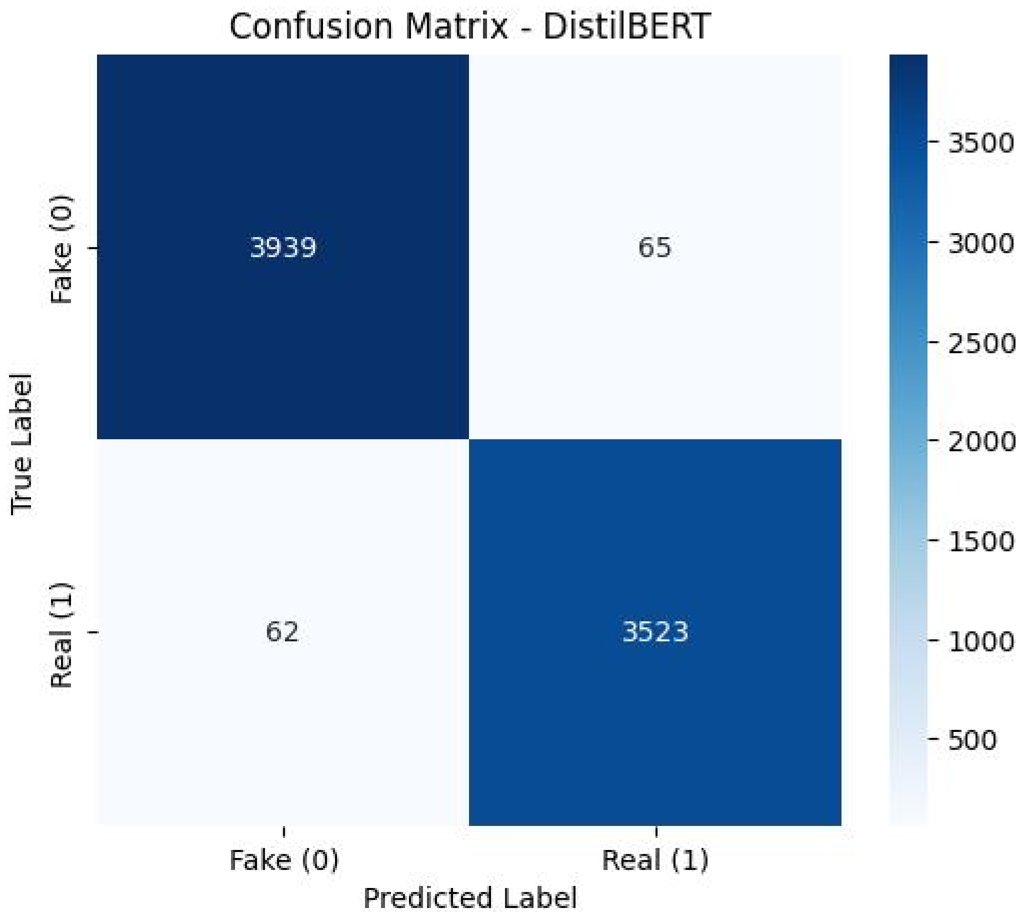

SVM - Confusion Matrix
Fake (0)
Real (1)
True label
6704
246
246
5282
Fake (0)
Real (1)
Predicted label
6000
5000
4000
3000
2000
1000



LSTM - Confusion Matrix
Fake (0)
Real (1)
True label
6499
451
655
4873
Fake (0)
Real (1)
Predicted label
6000
5000
4000
3000
2000
1000



BiLSTM - Confusion Matrix
Fake (0)
Real (1)
True label
6585
365
148
5380
Fake (0)
Real (1)
Predicted label
6000
5000
4000
3000
2000
1000



Confusion Matrix - DistilBERT
3939
65
62
3523
Fake (0)
Real (1)
True Label
Fake (0)
Real (1)
Predicted Label
3500
3000
2500
2000
1500
1000
500

Insights:

- **Naive Bayes:** High number of False Positives (1,091); weakest performer overall with the most misclassifications.
- **SVM:** Significant improvement over Naive Bayes, achieving a balanced error rate with exactly 246 misclassifications for both classes.
- **LSTM:** Shows a slight struggle with False Negatives (655), indicating it occasionally misses real news more than fake news.
- BiLSTM: Highly effective; significantly reduces errors compared to the standard LSTM, with only 148 False Negatives.
- **DistilBERT:** The top performer; achieves near-perfect classification with only 65 False Positives and 62 False Negatives.

DEPLOYMENT:

Overview

This document describes the steps to deploy the **Fake News Classification system** locally. The system consists of:

- **Backend:** FastAPI server with the best performing model DistilBERT.
- **Frontend:** React application that interacts with the backend to classify user-provided news articles.

The deployment focuses on a **local setup**, avoiding cloud services for simplicity.

2. Prerequisites

Before deploying, ensure the following software is installed:

1. **Python 3.11** (or compatible version)
2. **Node.js and npm** (for frontend)
3. **Virtual environment tools** (venv or conda)
4. **Git** (optional, if cloning from repository)

- The **backend** hosts trained models and provides endpoints for classification requests.
- The **frontend** is a user interface that submits news text and displays predictions and confidence scores.
- Users can run the system locally without cloud services, allowing full control over the environment.
- Long, multi-paragraph articles achieve higher accuracy, while short articles can optionally leverage SVM for better predictions.

Architecture Diagram

Description:

- **Input Layer:** User submits news article via frontend.
- **Backend Layer:** Receives the input, passes through preprocessing, and selects the appropriate model.
- **Model Layer:** Performs prediction using trained models (Naive Bayes, SVM, LSTM, BiLSTM, DistilBERT).
- **Output Layer:** Returns prediction label (Fake/Real) and confidence score to frontend.

CONCLUSIONS:

- DistilBERT achieves the highest accuracy and F1-score, particularly on multi-paragraph news articles.
- Traditional ML models (SVM, Naive Bayes) are useful for short articles and quick predictions.

- DistilBERT was trained and evaluated on full-length news articles (some with 10+ paragraphs). Short texts may lead to lower confidence or misclassification because transformer-based models are optimized for richer context.

- Proper preprocessing and article length consideration significantly improve performance.

RECOMMENDATIONS:

- ✓ **Use DistilBERT for production** as primary model; fall back to SVM for very short articles.
- ✓ **Deploy behind a server** if you later want web access. Could use Docker, Railway, or Render once cloud knowledge is comfortable.
- ✓ **Add caching** for repeated predictions on large articles to reduce latency.
- ✓ **Extend dataset** with more varied short and long articles to improve generalization.
- ✓ **Frontend UI improvements**: allow pasting large text, provide scrollable textarea for multi-paragraph input.

PROJECT OUTCOMES AND RESULTS: