REPORT



**SUMMER TRAINING/INTERNSHIP**

**PROJECT REPORT**

(Term June-July 2025)

**Fake News Detection System using Machine Learning**

Submitted by

**Palika Ghai**

**Registration Number: 12310349**

**Course Code: PETV79**

Under the Guidance of

**Manipal Singh Papola**

**Assistant Professor**

**School of Computer Science and Engineering**

CERTIFICATE

This is to certify that this project titled "FAKE NEWS DETECTION SYSTEM USING MACHINE LEARNING " is a project completed by Palika Ghai ( 12310349) during the Summer Internship for the term June-July 2025, in partial fulfilment of the requirements for the course CSE343 / CSE443

The work presented in this report is the original work of the student and carried out with sincerity and is in accordance with academic ethics.

**Signature**  
Manipal Singh Papola

Assistant Professor  
School of Computer Science and  
Engineering

Palika Ghai

12310349

**Signature**

**Signature**

Head of Department

**Acknowledgment**

I want to extend my deepest and most heartfelt gratitude to Assistant Professor. Manipal Singh Papola. His unwavering support with consistent encouragement were truly the bedrock upon which this project was built. From the very initial concept building to the minor details of implementation, his invaluable guidance proved impactfully helpful at every turn. His remarkable expertise in the field of machine learning, coupled with his practical understanding of real-world data processing, not only broadened my perspective but also critically shaped the building of my project. All those countless discussions and queries were patiently handled and explained by him helped me reaching the goal I wanted to.

This journey will have been more challenging, and certainly less joyful, without the incredible support system of my family. Their belief in me, their understanding of me working during long hours, and their constant encouragement also helped me overcoming the obstacles, and discouragement and celebrate successes. Their presence made every step of this demanding process smoother.

Finally, I wish to express my sincere appreciation to the global community of open-source tool developers and contributors. Specifically, my profound thanks go to the creators and maintainers of essential tools such as Python, Scikit-learn, Streamlit, and NLTK. These powerful frameworks and libraries formed the backbone of my project, enabling me to the exploration of complex ideas and the development of sophisticated solutions.

**Content of the Report**

- Cover page

- Certificate

- Acknowledgement

- Table of Contents

Chapter-wise Report

Chapter 1: Introduction

* 1. Introduction
  2. Company profile
  3. Overview of training domain
  4. Objective of the project

Chapter 2: Training Overview

* 1. Tools & technologies used
  2. Areas covered during training
  3. Daily/weekly work summary

Chapter 3: Project Details

* 1. Title of the project
  2. Problem definition
  3. Scope and objectives
  4. System Requirements
  5. Architecture Diagram
  6. Data flow / UML Diagrams

Chapter 4: Implementation

* 1. Tools used
  2. Methodology
  3. Modules / Screenshots

Chapter 5: Results and Discussion

* 1. Output / Report
  2. Challenges faced
  3. Learnings

Chapter 6: Conclusion

* 1. Summary
  2. Key Insights & Reflection
  3. Limitations of the Current System
  4. Working Screenshots

**CHAPTER 1: INTRODUCTION**

1.1 Introduction

The rapid growth of digital platforms has transformed how news is distributed and consumed. While these platforms offer convenience and accessibility, but they also contribute a rapid spread of fake news, which refers to false or misleading information presented as news, often with the intent to mislead or manipulate readers.

Fake news has the potential to create misinformation at a very large scale, influencing public opinion. Various traditional methods for checking the authenticity of news such as human fact-checking are inadequate in the face of the large volume of content generated daily.

So, this project focuses on the development of a machine learning-based Fake News Detection System. The system uses Natural Language Processing (NLP) techniques to analyse the textual data of news headlines and articles and classify them as either real or fake. By doing so, it also offers a scalable, and reliable solution for detecting misinformation before it spreads.

The project integrates essential phases of the machine learning pipeline such as data preprocessing, feature extraction using TF-IDF, sentiment analysis, model building using Random Forest and Logistic Regression, and deployment via Streamlit for real-time use.

1.2 Company Profile

As this is an academic and self-driven project, no specific external company is involved. The work is completed independently using open-source resources, academic guidance, and publicly available datasets.

This project relies on the Python ecosystem, which includes tools like Pandas,Matplotlib,Seaborn, NLTK, TextBlob, Scikit-learn, and Streamlit which are commonly used in industry-standard NLP pipelines.

1.3 Overview of Training Domain

The project falls under the intersection of Machine Learning, Natural Language Processing, and Data Visualization.

Key concepts covered include:

1. Text preprocessing and cleaning
2. Word vectorization using TF-IDF
3. Sentiment polarity scoring
4. Classification models: Random Forest, Logistic Regression
5. Visualization of linguistic and statistical patterns
6. Model deployment using Streamlit

These all steps are used to detect and find patterns from the texts .

1.4 Objective of the Project

1. To build a system capable of automatically classifying news articles or headlines as *Fake* or *Real*.
2. To extract relevant features such as word frequency and sentiment polarity using NLP techniques.
3. To compare multiple machine learning algorithms and identify the best performer based on evaluation metrics.
4. To visualize the data distribution, feature importance, and classification accuracy.
5. To deploy the model using a web interface for real-time predictions.

**CHAPTER 2: TRAINING OVERVIEW**

2.1 Tools & Technologies Used

This project was developed using a robust suite of programming languages, machine learning libraries, and web-based tools. The following tools were used throughout the training and implementation phases:

|  |  |
| --- | --- |
| Tool / Library | Purpose & Usage |
| Python 3.x | Primary programming language |
| Jupyter Notebook | Experimentation, testing, and visualization |
| Pandas | Data wrangling and preprocessing |
| NumPy | Numerical operations and array manipulation |
| Scikit-learn | ML algorithms, model evaluation, TF-IDF, GridSearchCV |
| NLTK | Text cleaning and stopword filtering |
| TextBlob | Sentiment polarity (polarity score extraction) |
| Matplotlib/Seaborn | Visualization: class distribution, confusion matrix, etc. |
| WordCloud | Word frequency visualizations |
| Streamlit | Web-based UI for real-time news analysis |
| Joblib | Model and vectorizer serialization |

2.2 Areas Covered During Training

During the internship/project development, the following major areas and skillsets were covered:

1. Natural Language Processing (NLP):
   * Text normalization, cleaning, tokenization
   * Stopword removal using NLTK
   * Word vectorization using TF-IDF
   * Sentiment polarity extraction using TextBlob
2. Class balancing, outlier removal
   * Creating word clouds, distribution plots, and feature importance graphs
   * Interpreting confusion matrices and classification results
3. Model Building:
   * Training classification models (Logistic Regression, Random Forest)
   * Use of evaluation metrics: Accuracy, Precision, Recall, F1-Score
   * Hyperparameter tuning with GridSearchCV
4. Software Engineering and Deployment:
   * Writing clean, modular Python scripts (app.py, model\_train.py)
   * Serializing models for reuse (.pkl files)
   * Building an interactive frontend using Streamlit

2.3 Daily/Weekly Work Summary

Here’s a breakdown of progress I made week-by-week:

|  |  |
| --- | --- |
| Week | Activities Completed |
| Week I | Finalized project topic; gathered and explored datasets (Fake.csv, True.csv) & Performed data cleaning, preprocessing (stopwords, punctuation, case handling) |
| Week II | Implemented EDA: generated word clouds, class distribution plots and applied TF-IDF vectorization and polarity scoring using TextBlob |
| Week III | Trained and evaluated Logistic Regression and Random Forest models, Conducted GridSearchCV-based hyperparameter tuning and Visualized results: confusion matrix, accuracy, feature importance. |
| Week IV | Developed Streamlit app, integrated .pkl files, tested live predictions and finalized report writing, documentation, code cleanup, and visualization enhancements |

**CHAPTER 3: PROJECT DETAILS**

3.1 Title of the Project

Fake News Detection System Using Machine Learning

3.2 Problem Definition

In today’s digital ecosystem, the spread of unverified, false, or misleading news commonly referred to as *fake news* that has become increasingly huge in number . Such misinformation, often packaged to look legit, and can easily manipulate public perception, mislead users, and potentially destabilize social and political environments.

The problem has become more critical when we consider the scale at which this content is generated and consumed. Manual verification methods are no longer scalable. Hence, there arises a need for an automated solution that can detect fake news using data-driven techniques.

This project aims to tackle the above challenge by using Natural Language Processing (NLP) and Machine Learning (ML) to identify patterns and linguistic features that distinguish fake news from real news.

3.3 Scope and Objectives

Scope:

1. This project is limited to analysing English news articles, specifically using titles/headlines as primary input.
2. It is focused on binary classification: *Fake (0)* or *Real (1)*.
3. The system supports real-time analysis via a deployed web interface built using Streamlit.

Objectives:

1. Preprocess and clean news text data using NLP techniques.
2. Extract features using TF-IDF vectorization and sentiment polarity.
3. Train multiple ML models and evaluate their performance using proper metrics.
4. Fine-tune models using GridSearchCV.
5. Visualize feature importance, classification metrics, and word usage trends.
6. Deploy the best-performing model using a user-friendly Streamlit interface.

3.4 System Requirements

Hardware:

1. Minimum 4 GB RAM (8+ GB preferred)
2. Multi-core CPU
3. Internet connection (for library installation, dataset access)

Software:

1. Operating System: Windows 10
2. IDE: Jupyter Notebook, VS Code
3. Languages Used: Python 3.x
4. Libraries Required: Pandas, Numpy, Scikit-learn, NLTK, TextBlob, Streamlit, Matplotlib, Seaborn, Joblib,Naïve-Bayes

3.5 Architecture Diagram

A black grid with white text

AI-generated content may be incorrect.

Figure I. Architecture Diagram

3.6 Data Flow(UML Diagram)

A diagram of text and text removal

AI-generated content may be incorrect.

Figure II. Data Flow

**CHAPTER 4: IMPLEMENTATION**

4.1 Tools Used

The implementation of the Fake News Detection System was carried out using the Python programming language and associated libraries for data science, machine learning, and deployment.

|  |  |
| --- | --- |
| Tool / Library | Role in Implementation |
| Pandas | Data loading and manipulation |
| NumPy | Array handling and numerical processing |
| NLTK | Stopwords removal and token processing |
| TextBlob | Sentiment polarity calculation for each headline |
| Scikit-learn | TF-IDF vectorization, ML models, model evaluation, GridSearchCV |
| Matplotlib & Seaborn | Data visualization (feature importance, class distribution, confusion matrix) |
| Naïve-Bayes | As its suited for word frequencies |
| WordCloud | Visual representation of frequent words in fake and real articles |
| Streamlit | Web application deployment |
| Joblib | Serialization of models and vectorizers into .pkl format |

4.2 Methodology

The full project was divided into the following major phases:

I.Data Collection and Preprocessing

* Combined two CSV files: Fake.csv and True.csv
* Each dataset had a title column, renamed to text for consistency
* Labels assigned: 0 for Fake, 1 for Real
* Cleaning included:
  + Lowercasing text
  + Removing punctuation, HTML tags, URLs
  + Eliminating stopwords using nltk.corpus
  + Applying TextBlob to extract a polarity score (ranging -1 to +1)

II. Feature Engineering

* Used TF-IDF Vectorizer with a max of 5000 words to transform text into feature vectors
* Saved the fitted vectorizer using joblib.dump() to tfidf\_vectorizer.pkl

III. Model Building & Evaluation

* Two classifiers trained: LogisticRegression() and RandomForestClassifier()
* Used train\_test\_split() with 80% for training and 20% for testing
* Evaluated using:
  + Accuracy
  + Confusion Matrix
  + Precision, Recall, F1-score
  + Feature importance from Random Forest

1V. Hyperparameter Tuning

* Applied GridSearchCV to the Random Forest:
  + n\_estimators: [100, 200]
  + max\_depth: [10, 20, None]
* Selected best parameters and saved the model as fake\_news\_model.pkl

V. Deployment using Streamlit

* Created a separate app.py file
* Streamlit UI includes:
  + Text box for user input
  + Button: “Predict”
  + Output: “Fake” or “Real” with confidence % into it.
* Used joblib.load() to load both model and TF-IDF vectorizer

4.3 Modules / Screenshots

I. Class Distribution Plot

* Shows nearly equal distribution of real and fake news
* Confirms no major class imbalance
* Helps validate fairness of training

A blue rectangular object with text

AI-generated content may be incorrect.

Figure III. Class Distribution

II. WordCloud - Fake News

* Words like VIDEO, TRUMP, BREAKING, WATCH dominate
* Reflects emotional and attention-grabbing language

A close up of words

AI-generated content may be incorrect.

Figure IV. Fake News WordCloud

III. WordCloud - Real News

* Words like HOUSE, PRESIDENT, POLICY, US appear frequently
* Shows structured and policy-based language

A close up of words

AI-generated content may be incorrect.

Figure V. Real News WordCloud

IV. Confusion Matrix

* High diagonal values show accurate classification
* Few misclassifications (False Positives, False Negatives)

A blue squares with white text

AI-generated content may be incorrect.

Figure VI. Confusion Matrix

V. Feature Importance Plot

* Highlights top predictive features like video, says, report
* Also confirms polarity as a useful numerical feature

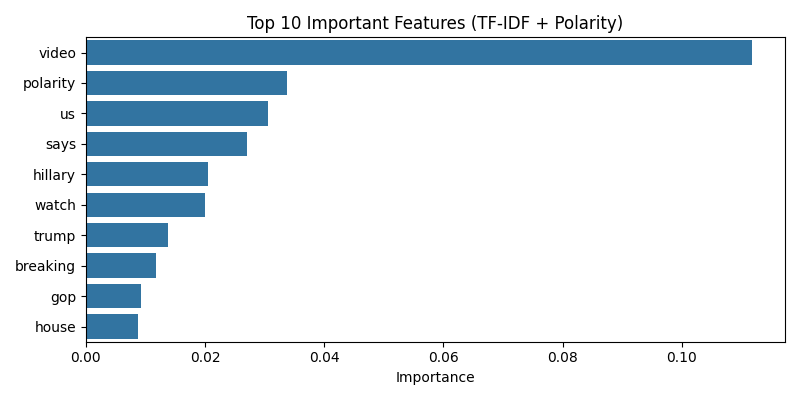


Figure VII. Feature Importance Plot

VI. Streamlit App Screenshot

* Real-time prediction system
* Easy UI: paste headline ->predict -> get result with probability

**A screenshot of a computer

AI-generated content may be incorrect.**

Figure VIII. Streamlit App UI

**CHAPTER 5: RESULTS AND DISCUSSION**

5.1 Output & Evaluation Results

After training and testing both classifiers (Logistic Regression and Random Forest), the system’s performance was evaluated using various metrics.

1. *Logistic Regression* is a statistical method used for binary classification problems, where the goal is to predict the probability of an event occurring or not

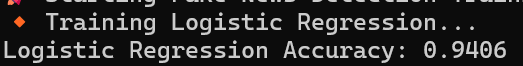


Figure IX. Logistic Regression

1. Random Forest is a machine learning technique that's used to solve regression and classification problems.

A black background with white text

AI-generated content may be incorrect.

Figure X. Random Forest

1. Evaluation of each model’s performance of a **single Random Forest model** after optimizing its hyperparameters using GridSearchCV: Evaluated Linear Regression , Random Forest , Tuning Random Forest, and Naïve-Bayes.

A graph of different colored bars

AI-generated content may be incorrect.

Figure XI. Each Model’s Performance Analysis through BarChart

5.2 Feature Importance Discussion

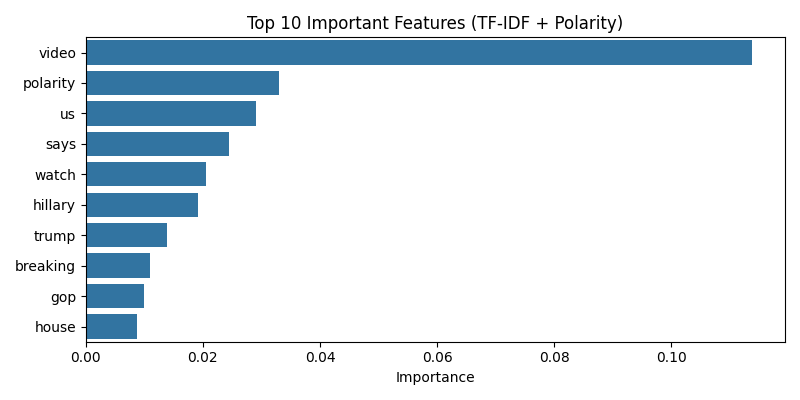


Figure XII. Feature Importance Discussion

Important features include:

1. "video", "report", "says", "hillary" which are frequently associated with clickbait or exaggerated claims in fake news.
2. "us", "president", "white house" are few of the common words in real news.
3. polarity (sentiment) is ranked high which shows emotional tone that contributes meaningfully.

This analysis explains the model’s decisions and supports transparency.

5.3 Challenges Faced

1. Imbalanced Tone & Vocabulary:  
   Fake news uses a highly emotional and sensational tone, making cleaning and feature extraction non-trivial.
2. Overfitting Risk:  
   High-dimensional TF-IDF vectors risk overfitting. Controlled with hyperparameter tuning and limiting max\_features.
3. Deployment Adjustments:  
   Integrating polarity with TF-IDF in Streamlit required converting sparse and dense arrays handled using hstack().
4. Evaluation Complexity:  
   Balancing multiple metrics (Accuracy vs. Recall) required careful model selection based on end-use priorities.

5.4 Learnings

1. Learned how text preprocessing pipelines (cleaning, stopwords, polarity) significantly influence model performance.
2. Understood the role of feature engineering in NLP — adding polarity enhanced prediction power.
3. Gained hands-on experience in model deployment using Streamlit and .pkl model loading via Joblib.
4. Improved my data interpretation skills using visuals: word clouds, confusion matrices, and feature plots.

**CHAPTER 6: CONCLUSION**

6.1 Summary

The primary goal of this project is to develop an intelligent system that’s capable of identifying whether a given news article or headline is *fake* or *real* using machine learning and natural language processing techniques as with the increasing spread of misinformation across digital media platforms. This model is built on the two datasets containing one verified real news and another composed of fabricated or misleading news content. A complete end-to-end pipeline was designed and implemented, beginning with data collection and progressing through stages of text cleaning, feature extraction, model training, tuning, evaluation, and deployment.

The major accomplishments of this project include:

1. Implementing **data preprocessing** techniques such as stopword removal, punctuation stripping, lowercasing, and sentiment polarity extraction.
2. Performing **exploratory data analysis**, including word clouds, class distribution plots, and sentiment visualization to understand the nature of real and fake news.
3. Applying **feature engineering** using the TF-IDF (Term Frequency-Inverse Document Frequency) approach to convert textual data into numerical format.
4. Introducing **polarity score** as a numerical feature based on sentiment, which helped the model grasp the emotional tone of the news.
5. Training and comparing multiple classifiers—**Logistic Regression** and **Random Forest**—to select the best-performing model.
6. Applying **hyperparameter tuning** using GridSearchCV to enhance the model’s generalization ability and performance.
7. Deploying the system using **Streamlit**, enabling real-time predictions with a user-friendly web interface.
8. Saving trained models and vectorizers as .pkl files using Joblib for reproducibility and lightweight deployment.

6.2 Key Insights & Reflections

1. **Language Patterns Matter:**  
   Word clouds and polarity analysis revealed that fake news often uses emotionally charged or exaggerated language, while real news tends to maintain a formal and objective tone.
2. **Feature Engineering Makes a Difference:**  
   Combining TF-IDF with polarity score significantly improved the model's accuracy and interpretability, demonstrating the importance of thoughtful feature design.
3. **Model Interpretability is Crucial:**  
   The feature importance graph helped explain model decisions, making the system more trustworthy and transparent—critical in domains like misinformation detection.
4. **Machine Learning is Iterative:**  
   From preprocessing tweaks to repeated model retraining and tuning, the project underscored the iterative nature of machine learning workflows.
5. **Deployment Bridges Theory and Practice:**  
   Building and testing a live Streamlit app brought the project out of theory and into real-world application. It also enhanced practical skills in frontend integration with ML models.

6.3 Limitations of the Current System

While the project achieved its goals, certain limitations remain:

1. **Dataset Language**: The model was trained only on English news; performance on other languages is unknown.
2. **Input Length**: The model performs best on headlines and short articles; longer content might dilute keyword relevance.
3. **Dynamic Language Use**: Language trends and political topics evolve rapidly. Without retraining on updated datasets, the model may lose effectiveness over time.
4. **No Fact Checking Logic**: The system classifies based on learned patterns, not factual truth, and may misclassify clever misinformation.

6.4 Working screenshots:-

1. When it has a high confidence that entered data is correct and can be regarded as

“*real news”*

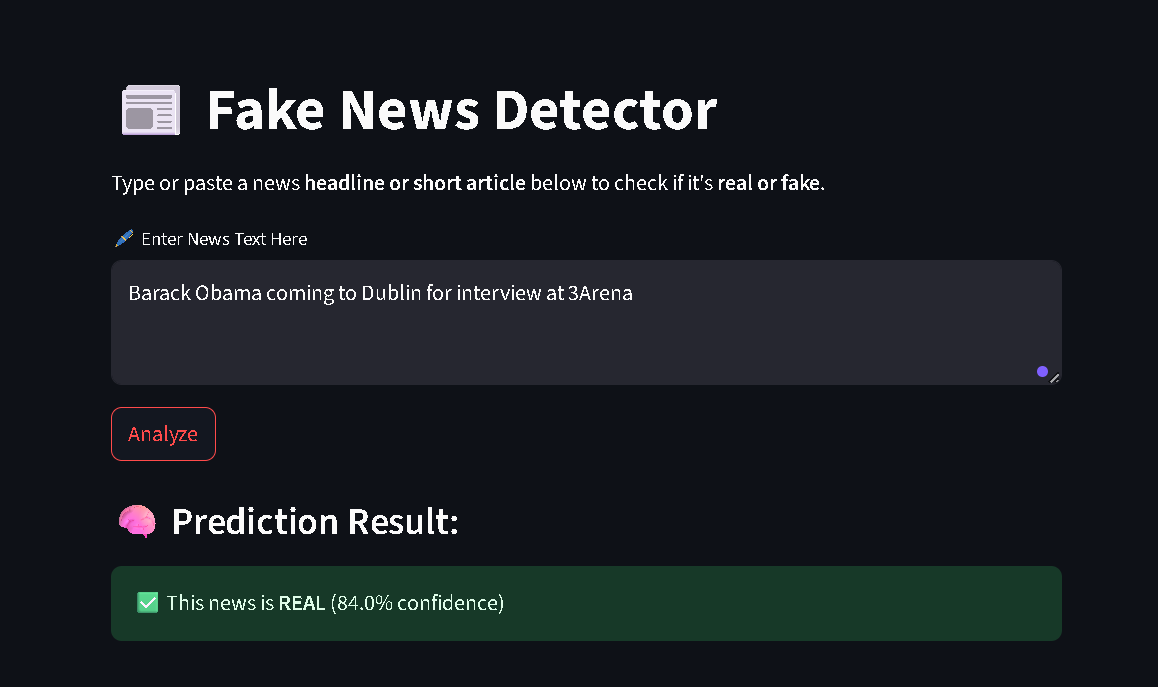


Figure XIII. Result when News is Correct

1. When the model predicts that entered data isn’t correct or it finds the presence of highly repeatitive fake news words.

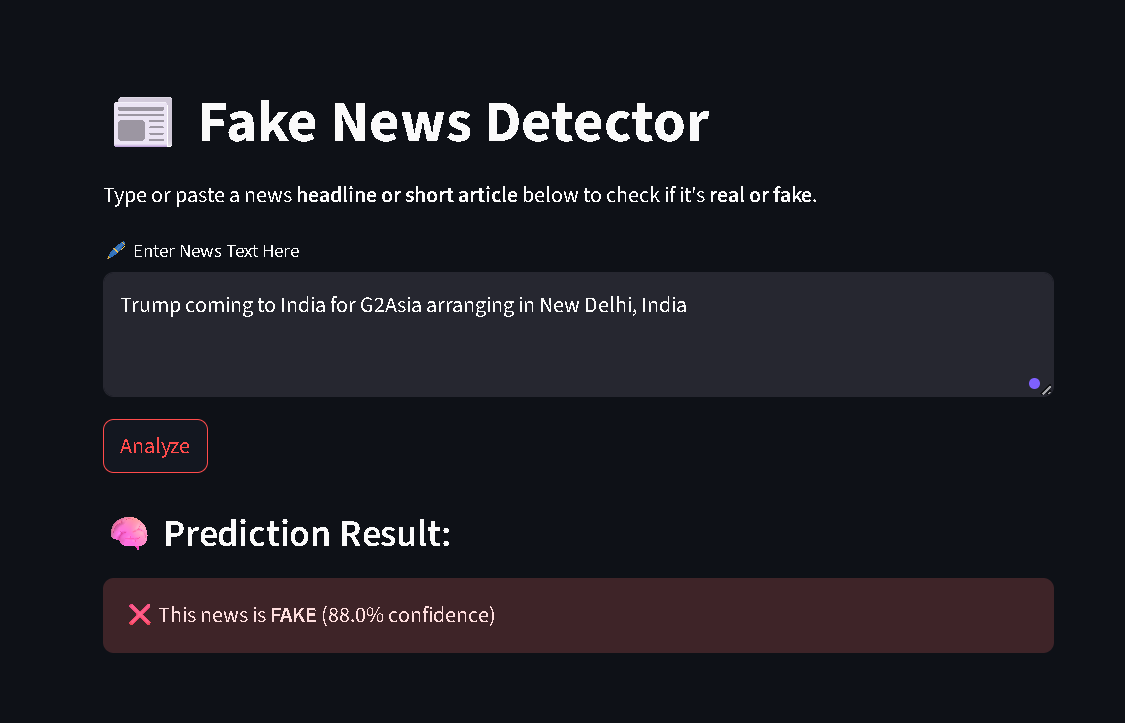


Figure XIV. Result when News is Incorrect

1. Also, if someone intentionally or accidentally left the text box empty

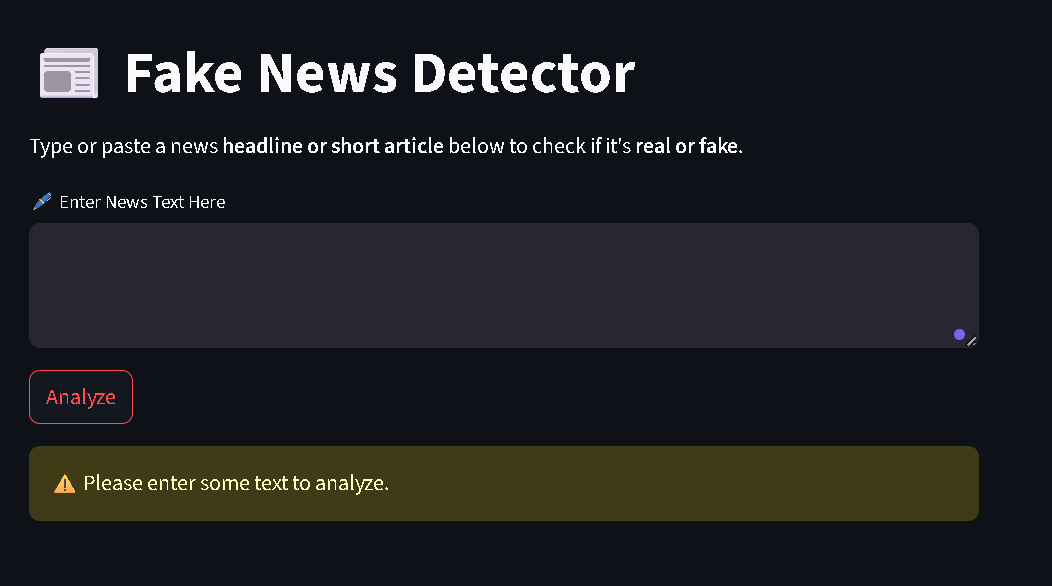


Figure XV. Result when you left the textbox Empty