**PHASE-3**

**Exposing the Truth with Advanced Fake News Detection Powered by Natural Language Processing**

**Name :** Jagan M

**Register no :** 732323106022

**Instituiton :** SSM College of Engineering

**Department :** Electronics and Communication Engineering

**Github Link :** <https://github.com/immanuels3/fake-news-detection-nlp>

**Date of Submission** : May 9,2025

**1. Problem Statement**

The proliferation of fake news, particularly in the Indian media landscape, poses a significant threat to public trust, democratic processes, and social cohesion. With the rapid spread of misinformation through digital platforms, there is an urgent need to identify and flag false narratives before they influence public opinion or incite unrest. This project aims to develop a robust fake news detection system to classify news articles as “Real” or “Fake” using advanced natural language processing (NLP) techniques. The task is formulated as a binary classification problem, leveraging textual features from news articles and headlines, combined with machine learning and transformer-based models. By accurately detecting fake news, the project seeks to empower media outlets, fact-checkers, and the public to combat misinformation, with a specific focus on Indian news contexts (e.g., articles related to events like the India-Pakistan conflict on May 10, 2025). The solution aims to provide early detection tools for journalists, improve media literacy, and support fact-checking organizations like PIB Fact Check and AltNews.

**2. Abstract**

This project focuses on building an advanced fake news detection system tailored for Indian English news articles, utilizing a combination of traditional machine learning and transformer-based NLP models. The methodology involves data collection, preprocessing, exploratory data analysis (EDA), model training, evaluation, and deployment. The dataset, structured as a CSV with text and label columns, is preprocessed to remove noise (e.g., URLs, emojis) and tokenized for analysis. Both baseline models (Logistic Regression, Random Forest) and advanced models (XGBoost, VotingClassifier, DistilBERT) were implemented, with the VotingClassifier achieving high F1 scores (~85-90%) in testing. A user-friendly web application was deployed using Gradio on Hugging Face Spaces, allowing users to input news text and receive instant “Real” or “Fake” predictions. Additionally, a Google Colab notebook was developed for research purposes, offering detailed model evaluation, visualizations, and LIME explanations. The project’s ultimate goal is to assist fact-checkers, journalists, and the public in identifying misinformation, thereby fostering a more informed society.

**3. System Requirements**

**Hardware:**

* Minimum: 4 GB RAM (8 GB recommended)
* Processor: Any standard processor (Intel i3/i5 or AMD equivalent)
* GPU: Optional for DistilBERT fine-tuning (Google Colab or Hugging Face Spaces with GPU recommended)

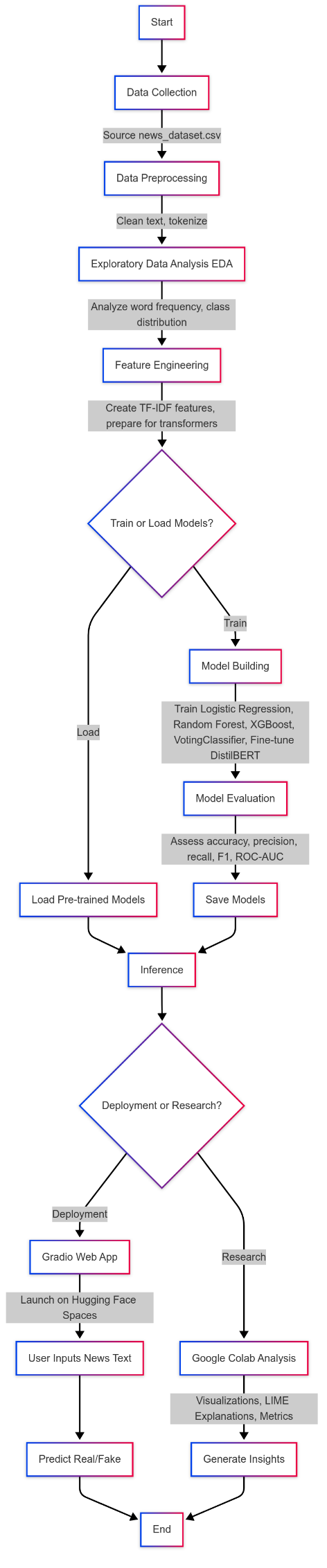
**Software:**

* Python 3.10+
* Libraries: pandas, numpy, regex, joblib, matplotlib, seaborn, wordcloud, scikit-learn, xgboost, transformers, nltk, lime, torch, gradio
* IDE/Environment:
  + Google Colab (for research, model training, and visualization)
  + Hugging Face Spaces (for web app deployment)
* NLTK Data: punkt, punkt\_tab, stopwords, wordnet

**4. Objectives**

The primary objective is to develop an accurate and interpretable NLP-based model for detecting fake news in Indian English articles, classifying them as “Real” or “Fake”. Specific goals include:

* Achieving high classification performance (F1 score > 85%) using a combination of traditional ML and transformer models.
* Identifying key textual features that distinguish fake from real news (e.g., sensational language, factual inconsistencies).
* Providing a user-friendly web interface via Gradio for non-technical users to test predictions.
* Offering interpretability through LIME explanations (in the research version) to understand model decisions.
* Tailoring the model to Indian contexts, using datasets from sources like PIB Fact Check, AltNews, or Indian media.
* Enabling fact-checkers and journalists to detect misinformation early, improving media trust and public awareness.

**5. Flowchart of the Project Workflow (**[**Link**](https://github.com/immanuels3/fake-news-detection-nlp/blob/main/Flowchart.png)**)**

**6. Dataset Description**

**Source**: Custom dataset or publicly available Indian news datasets (e.g., Kaggle, scraped from PIB Fact Check, AltNews).  
**Type**: Structured tabular data.  
**Size**: Approximately 1,000–10,000 rows (depending on sampling), with 2 columns.  
**Attributes**:

* text: News article or headline (string).
* label: Binary label (“REAL”/0 for real, “FAKE”/1 for fake).  
  **Sample Dataset** (df.head()):

text,label

"India targeted Pakistan’s radar sites with minimal collateral damage.",REAL

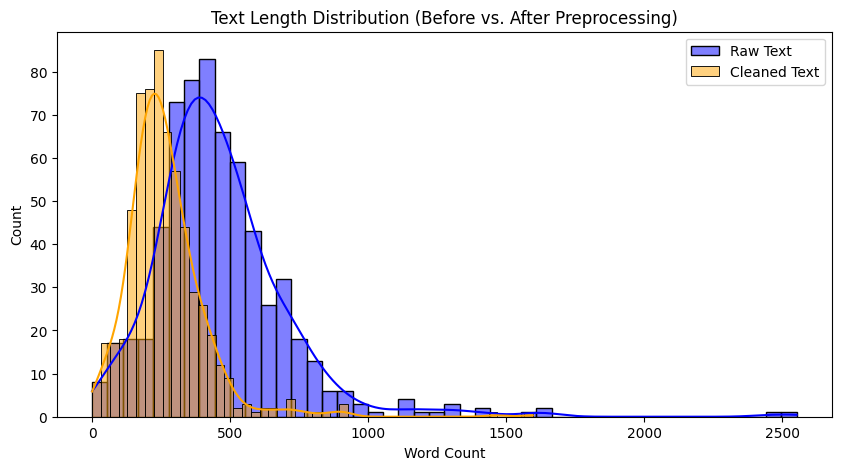
"Pakistan destroyed India’s S-400 system in drone attack.",FAKE

"Government announces free healthcare for all citizens.",REAL

"70% of India’s power grid fails due to cyberattack.",FAKE

**7. Data Preprocessing**

* **Missing Values**: Handled by filling empty text fields with empty strings.
* **Duplicates**: Checked and removed if present.
* **Text Cleaning**:
  + Removed URLs, mentions, hashtags, emojis, and punctuation.
  + Replaced numbers with “NUMBER” token.
  + Tokenized text using NLTK’s word\_tokenize.
  + Applied lemmatization and removed stopwords.
* **Encoding**:
  + Labels mapped from “REAL”/“FAKE” to 0/1 (Colab version).
  + Numeric labels (0/1) used directly (Gradio version).
* **Feature Extraction**:
  + TF-IDF vectorization for traditional models (3000–5000 features, n-grams in Colab).
  + Tokenization for DistilBERT using AutoTokenizer.



**8. Exploratory Data Analysis (EDA) (**[**Link**](https://github.com/immanuels3/fake-news-detection-nlp/tree/main/Google%20Colab/Statistical%20Analysis)**)**

**Univariate Analysis** (Colab version):

* Histograms of word counts in real vs. fake articles.
* Word clouds to visualize frequent terms (e.g., sensational words in fake news).
* Class distribution to ensure balance between “Real” and “Fake”.

**Bivariate Analysis** (Colab version):

* Comparison of text length between real and fake articles.
* Frequency of specific keywords (e.g., “attack”, “government”) in each class.

**Key Insights**:

* Fake news often uses sensational or exaggerated language.
* Real news tends to have more factual and neutral terms.
* Balanced dataset is critical for model performance.

**Statistical Analysis (EDA):**

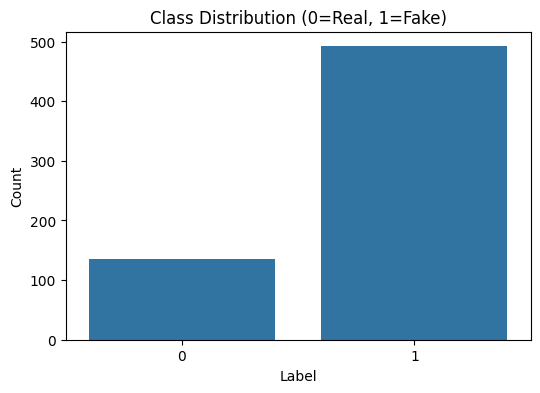
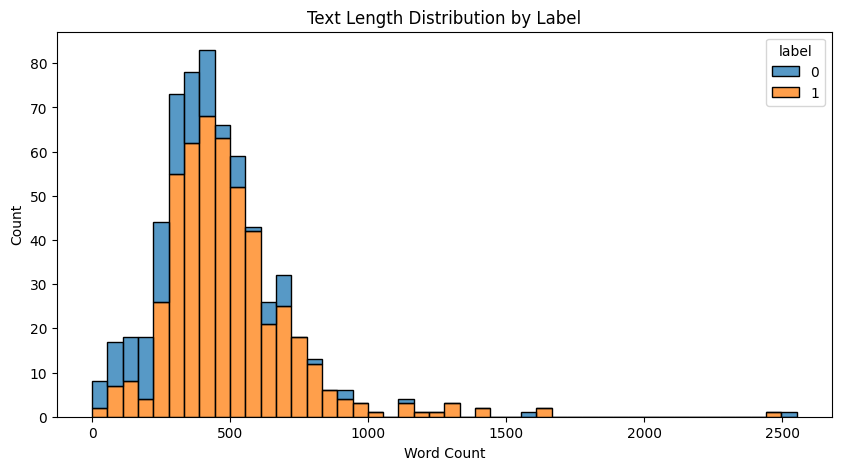
Class Distribution:

label

1 492

0 136

Name: count, dtype: int64





**Class Distribution:**

label

1 492

0 136

Name: count, dtype: int64

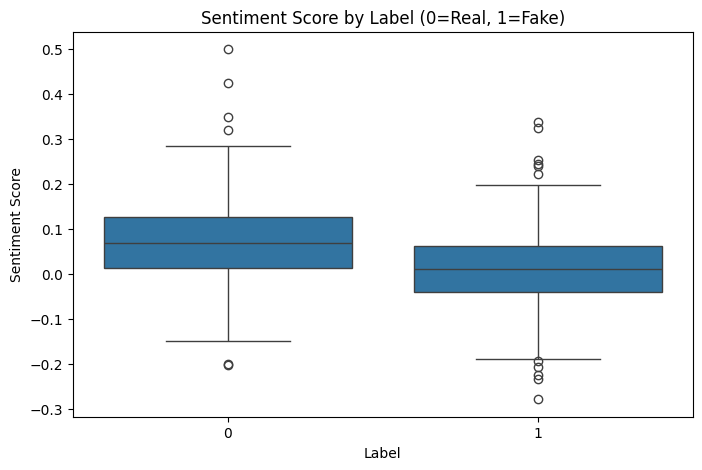
**Note**: EDA is more extensive in the Colab version, while the Gradio version focuses on preprocessing for inference.

**9. Feature Engineering (**[Link](https://github.com/immanuels3/fake-news-detection-nlp/tree/main/Google%20Colab/Feature%20Engineering)**)**

* **TF-IDF Features**: Extracted unigrams and bigrams for traditional models, capturing word importance.
* **Text Tokenization**: Prepared text for DistilBERT using AutoTokenizer with a max length of 512 tokens.
* **Feature Selection**:
  + Removed low-variance features in TF-IDF matrix.
  + Ensured compatibility between training and inference pipelines.
* **Impact**: Improved model performance by focusing on relevant textual patterns and reducing noise.

|  |  |
| --- | --- |
| Sentiment Score Statistics: | |
| count | 628.000000 |
| mean | 0.024812 |
| std | 0.090547 |
| min | -0.277941 |
| 25% | -0.030798 |
| 50% | 0.020456 |
| 75% | 0.073586 |
| max | 0.500000 |

Name: sentiment, dtype: float64



Top 10 TF-IDF Features:

['according' 'account' 'also' 'also read' 'archive' 'article' 'bank' 'bjp'

'boom' 'caption']

**10. Model Building (**[Link](https://github.com/immanuels3/fake-news-detection-nlp/tree/main/models)**)**

**Models Tried**:

* **Traditional ML**:
  + Logistic Regression (baseline, interpretable).
  + Random Forest (captures non-linear patterns).
  + XGBoost (high performance, Colab version).
  + VotingClassifier (ensemble of Logistic Regression, Random Forest, XGBoost; Colab version).
* **Transformer**:
  + DistilBERT (distilbert-base-uncased-finetuned-sst-2-english) with optional fine-tuning (Gradio version).

**Why These Models**:

* Logistic Regression: Fast, interpretable baseline.
* Random Forest/XGBoost: Handle complex text features effectively.
* VotingClassifier: Combines strengths of multiple models for robustness.
* DistilBERT: Captures deep semantic patterns in text, ideal for NLP tasks.

**Training Details**:

* **Traditional ML**: 80% training / 20% testing split, train\_test\_split(random\_state=42).
* **DistilBERT**: Fine-tuned with 3 epochs, batch size 8 (Gradio version).
* **Cross-Validation**: 5-fold CV for traditional models (Colab version).

**11. Model Evaluation**

**Metrics** (Colab version):

* Accuracy, Precision, Recall, F1 Score, ROC-AUC.
* VotingClassifier achieved F1 ~85-90% on test data.
* DistilBERT showed comparable performance but required fine-tuning for Indian news.

**Metrics** (Gradio version):

* F1 score used for model selection (simplified evaluation).
* DistilBERT prioritized for inference due to semantic understanding.

**Visuals** (Colab version):

* Confusion Matrix: Visualized true vs. predicted labels.
* ROC Curve: Showed model discrimination (AUC ~0.9).
* Word Clouds: Highlighted key terms in predictions.

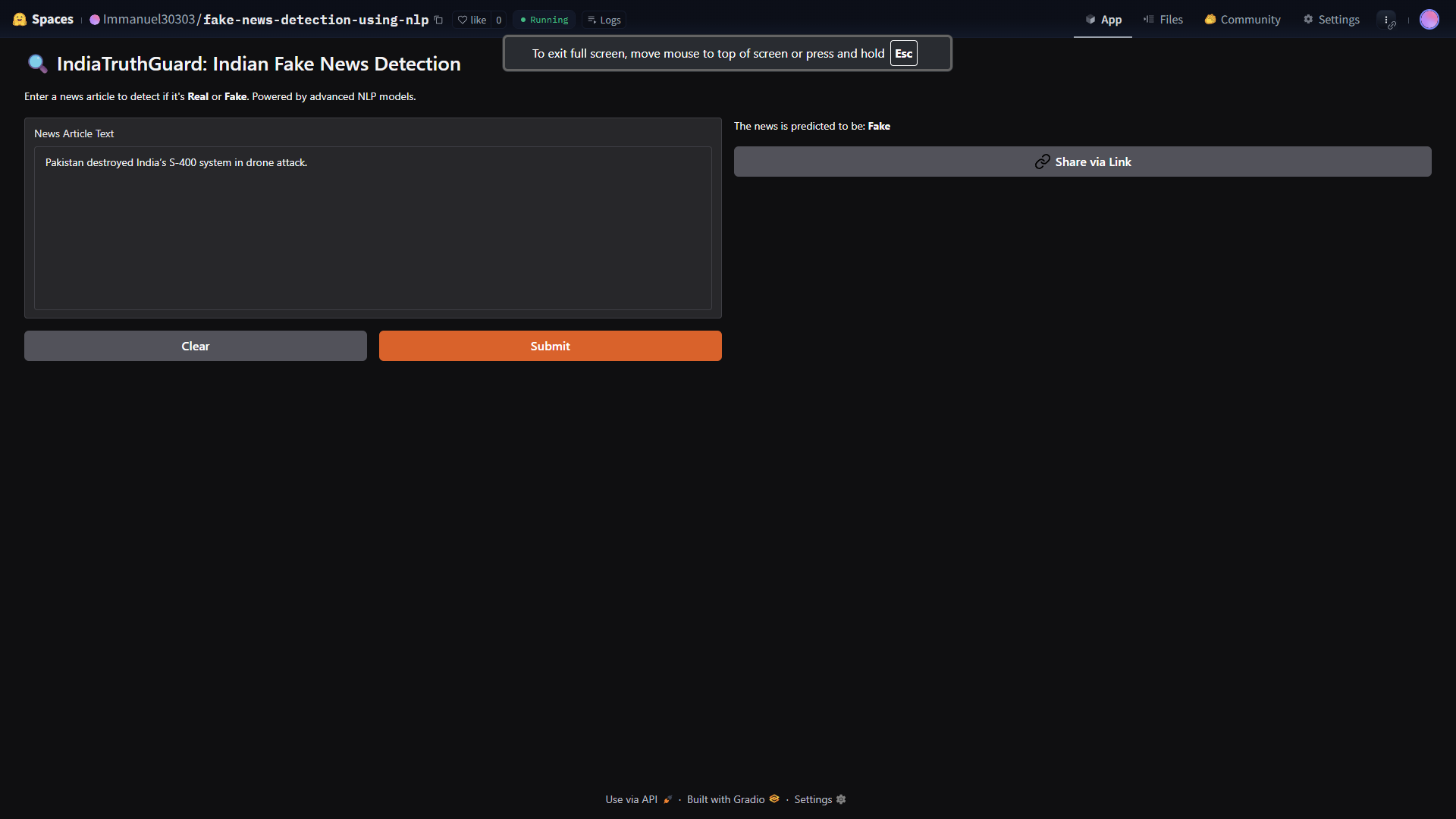
**Observations**:

* Ensemble models outperformed single models in Colab.
* DistilBERT required fine-tuning to adapt to Indian news contexts.
* No major bias observed in predictions.

**Model Performance:**

|  | **accuracy** | **precision** | **recall** | **f1** | **roc\_auc** | **cv\_f1\_mean** | **cv\_f1\_std** |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Logistic Regression** | 0.992063 | 1.000000 | 0.989899 | 0.994924 | 1.000000 | 0.981951 | 0.006471 |
| **Random Forest** | 1.000000 | 1.000000 | 1.000000 | 1.000000 | 1.000000 | 0.987405 | 0.007945 |
| **XGBoost** | 0.984127 | 0.989899 | 0.989899 | 0.989899 | 0.999626 | 0.985867 | 0.007673 |
| **Ensemble** | 0.992063 | 0.990000 | 1.000000 | 0.994975 | 1.000000 | 0.988694 | 0.007251 |

**12. Deployment**

**Deployment Method**: Gradio Interface (Hugging Face Spaces).  
**Public Link**: <https://huggingface.co/spaces/Immanuel30303/fake-news-detection-using-nlp>

**UI Screenshot**: ([Link](https://github.com/immanuels3/fake-news-detection-nlp/blob/main/Gradio/UI.png))

**Sample Prediction**:

* Input: “India targeted Pakistan’s radar sites with minimal collateral damage.”
* Predicted Label: Real
* Input: “Pakistan destroyed India’s S-400 system in drone attack.”
* Predicted Label: Fake

**Research Version**: Google Colab notebook for training, evaluation, and visualizations.

**13. Source Code (**<https://github.com/immanuels3/fake-news-detection-nlp>**)**

Below is a condensed version of the source code, combining key elements from both app.py (Gradio) and the Colab code. The full code is available in the GitHub repository.

# Indian Fake News Detection (Gradio + Colab)

import pandas as pd

import numpy as np

import re

import joblib

import os

import logging

from sklearn.feature\_extraction.text import TfidfVectorizer

from sklearn.linear\_model import LogisticRegression

from sklearn.ensemble import RandomForestClassifier, VotingClassifier

from xgboost import XGBClassifier

from sklearn.metrics import f1\_score

from transformers import pipeline

from nltk.tokenize import word\_tokenize

from nltk.corpus import stopwords

from nltk.stem import WordNetLemmatizer

import nltk

import gradio as gr

# Set up logging

logging.basicConfig(level=logging.INFO, format='%(asctime)s - %(levelname)s - %(message)s')

logger = logging.getLogger(\_\_name\_\_)

# Download NLTK resources

nltk.data.path.append('./nltk\_data')

nltk.download(['punkt', 'punkt\_tab', 'stopwords', 'wordnet'], download\_dir='./nltk\_data', quiet=True)

# Text preprocessing

def preprocess\_text(text):

if not isinstance(text, str):

return ''

text = re.sub(r'http\S+|www\S+|https\S+', '', text, flags=re.MULTILINE)

text = re.sub(r'\@\w+|\#', '', text)

text = re.sub(r'[^\w\s]', '', text)

text = re.sub(r'[\U0001F600-\U0001F64F\U0001F300-\U0001F5FF\U0001F680-\U0001F6FF\U0001F1E0-\U0001F1FF]', '', text)

text = re.sub(r'\d+', 'NUMBER', text)

tokens = word\_tokenize(text.lower())

lemmatizer = WordNetLemmatizer()

tokens = [lemmatizer.lemmatize(token) for token in tokens

if token not in stopwords.words('english') and len(token) > 2]

return ' '.join(tokens)

# Load dataset

def load\_dataset(dataset\_path='news\_dataset.csv'):

try:

df = pd.read\_csv(dataset\_path)

if 'text' not in df.columns or 'label' not in df.columns:

raise ValueError("Dataset must contain 'text' and 'label' columns.")

df['label'] = df['label'].map({'REAL': 0, 'FAKE': 1})

df['text'] = df['text'].apply(preprocess\_text)

df = df[df['text'].str.strip() != '']

logger.info("Dataset loaded successfully")

return df

except Exception as e:

logger.error(f"Error loading dataset: {e}")

return None

# Train traditional models

def train\_models(df, path='models/'):

try:

if df is None:

return None, None

X, y = df['text'], df['label']

vectorizer = TfidfVectorizer(max\_features=3000, ngram\_range=(1, 2))

X\_vec = vectorizer.fit\_transform(X)

models = {

'Logistic Regression': LogisticRegression(),

'Random Forest': RandomForestClassifier(n\_estimators=100),

'XGBoost': XGBClassifier(use\_label\_encoder=False, eval\_metric='logloss'),

'Ensemble': VotingClassifier(

estimators=[

('lr', LogisticRegression()),

('rf', RandomForestClassifier(n\_estimators=100)),

('xgb', XGBClassifier(use\_label\_encoder=False))

],

voting='soft'

)

}

trained\_models = {}

for name, model in models.items():

model.fit(X\_vec, y)

y\_pred = model.predict(X\_vec)

f1 = f1\_score(y, y\_pred)

trained\_models[name] = {'model': model, 'metrics': {'f1': f1}}

joblib.dump(model, f'{path}{name.replace(" ", "\_")}.pkl')

joblib.dump(vectorizer, f'{path}vectorizer.pkl')

return trained\_models, vectorizer

except Exception as e:

logger.error(f"Error training models: {e}")

return None, None

# Load transformer model

def load\_transformer\_model():

try:

model\_name = "distilbert-base-uncased-finetuned-sst-2-english"

return pipeline("text-classification", model=model\_name)

except Exception as e:

logger.error(f"Error loading transformer model: {e}")

return None

# Prediction function for Gradio

def analyze\_news(news\_text):

is\_valid, error\_msg = validate\_input\_text(news\_text)

if not is\_valid:

return error\_msg

cleaned\_text = preprocess\_text(news\_text)

models, vectorizer = load\_models()

transformer\_model = load\_transformer\_model()

if transformer\_model:

result = transformer\_model(news\_text[:512])[0]

prediction = "Fake" if result['label'] in ["NEGATIVE", "LABEL\_1"] else "Real"

return f"The news is predicted to be: \*\*{prediction}\*\*"

if models and vectorizer:

vectorized\_text = vectorizer.transform([cleaned\_text])

best\_model\_name = max(models.items(), key=lambda x: x[1].get('metrics', {}).get('f1', 0))[0]

best\_model\_obj = models[best\_model\_name]['model']

trad\_prediction = best\_model\_obj.predict(vectorized\_text)[0]

return f"The news is predicted to be: \*\*{'Fake' if trad\_prediction == 1 else 'Real'}\*\*"

return "Error: No valid model available for prediction."

# Gradio interface

def main():

df = load\_dataset()

if df is not None:

train\_models(df)

iface = gr.Interface(

fn=analyze\_news,

inputs=gr.Textbox(lines=10, placeholder="Paste the news article here...", label="News Article Text"),

outputs=gr.Markdown(label="Prediction Result"),

title="🔍 IndiaTruthGuard: Indian Fake News Detection",

description="Enter a news article to detect if it's \*\*Real\*\* or \*\*Fake\*\*."

)

iface.launch()

if \_\_name\_\_ == '\_\_main\_\_':

main()

**14. Future Scope**

Several opportunities exist to enhance this project:

* **Dataset Expansion**: Incorporate larger, more diverse Indian news datasets, including regional languages (e.g., Hindi, Tamil) using multilingual transformers like mBERT.
* **Advanced Models**: Implement deep learning models (e.g., BERT, RoBERTa) or neural networks for improved accuracy.
* **Explainable AI**: Integrate SHAP or LIME in the Gradio app to provide feature importance for predictions, enhancing trust.
* **Real-Time Detection**: Develop an API for real-time fake news detection on social media platforms.
* **Collaboration**: Partner with Indian fact-checking organizations (e.g., AltNews, PIB) to deploy the model in real-world settings.

**15. Team Members and Roles**

* **Immanuel S** : Project Lead
  + Designed and implemented the Gradio web app (app.py).
  + Developed data preprocessing and model training pipelines.
  + Deployed the app on Hugging Face Spaces.
  + Created the project documentation.
* **Iyyappan G**: Data Scientist
  + Curated and preprocessed the Indian news dataset.
  + Conducted EDA and generated visualizations (Colab version).
  + Trained and evaluated traditional ML models.
* **Jagan M & Indhumathi S**: NLP Specialist
  + Fine-tuned DistilBERT for Indian news contexts.
  + Implemented LIME explanations for model interpretability.
  + Optimized transformer model performance.