Fake News Detection

Project Report Submitted To

Naso Technologies Pvt. Ltd.

May 2025 – June 2025

By

Team Members (Team-D):

1. R J Revin Chrisbert
2. Seenu N
3. Beril Sam Daniel S
4. Abishek Rajan K
5. Mohamed Farook SK
6. Yogeshwaran PN

TABLE OF CONTENT:

PROJECT OBJECTIVE:

1.Abstract

In the digital age, the proliferation of fake news has become a significant challenge, threatening public trust, influencing political outcomes, and spreading misinformation at an alarming rate. Social media platforms and online news portals serve as major distribution channels, often lacking the verification mechanisms needed to prevent the circulation of false information. This project aims to tackle the issue of fake news by developing a machine learning-based classification system capable of distinguishing between real and fake news articles based on their textual content.

We used the "Fake and Real News Dataset" available on Kaggle, which contains labeled articles categorized as real or fake. The data underwent thorough preprocessing, including text cleaning, stopword removal, and vectorization using the TF-IDF (Term Frequency–Inverse Document Frequency) technique to convert raw text into numerical features. Multiple machine learning algorithms such as Logistic Regression, Naive Bayes, and Passive Aggressive Classifier were trained and evaluated.

The Passive Aggressive Classifier emerged as the most effective model, achieving an accuracy of over 90%. This project highlights the potential of machine learning in combating the spread of misinformation by automating the detection process. The solution can be integrated into web platforms or media outlets to assist in content verification, thereby contributing to a more informed and trustworthy information ecosystem.

2.Introduction

In today's fast-paced digital world, information is consumed and shared at an unprecedented rate. While this has brought many benefits in terms of connectivity and awareness, it has also created fertile ground for the rapid spread of misinformation and fake news. With the rise of social media platforms and online news portals, fake news can reach thousands or even millions of people within minutes, often without any verification. This presents a serious threat to public awareness, democracy, and even national security, as fake news can influence political opinions, create panic during crises, and damage reputations.

Traditional methods of detecting fake news involve human fact-checkers who manually verify the accuracy of claims. However, this approach is time-consuming, labor-intensive, and not scalable to the volume of content being produced daily. Hence, there is a growing need for automated systems that can detect and filter fake news with high accuracy and speed.

This project focuses on using machine learning techniques to build an automated fake news detection system. By analyzing the textual content of news articles, our system is trained to distinguish between real and fake news. The goal is to create a reliable model that can help media platforms and users filter out misleading content and contribute to a more trustworthy digital information environment.

3. SYSTEM REQUIREMENTS:

For the successful development, training, and execution of the **Fake News Detection** project, certain hardware and software requirements must be fulfilled. The project involves handling a moderately large dataset, performing data preprocessing, visualization, and running multiple machine learning models, which demands a reasonable computational environment.

3.1Hardware Requirements:

• **Processor:** Intel Core i5 or higher (or equivalent AMD processor) • **RAM:** Minimum 8 GB (Recommended 16 GB for faster model training) • **Storage:** At least 10 GB of free disk space (to store datasets, libraries, and outputs)

• **Graphics Card:** Optional (GPU acceleration can be used for faster model training, especially with XGBoost, or deep learning models, but it is not mandatory for this project)

• **Monitor:** 1080p resolution recommended for better visualization clarity

3.2 Software Requirements:

• **Operating System:** Windows 10/11, macOS, or any modern Linux distribution (Ubuntu preferred)

• **Programming Language:** Python 3.7 or higher

• **Development Environment:**

o Jupyter Notebook (preferred for exploratory analysis)

o VS Code / PyCharm (optional for script-based development)

• **Required Python Libraries:**

o numpy — For numerical operations

o pandas — For data manipulation and analysis

o seaborn — For statistical visualizations

o matplotlib — For general plotting and graphing

o scikit-learn — For machine learning models and preprocessing

o xgboost — For Extreme Gradient Boosting model

o streamlit —For building an interactive web app for prediction display

4. Technologies Used

This fake news detection system combines traditional NLP techniques with modern transformer-based models to classify news articles as real or fake. The pipeline includes preprocessing, tokenization, training using HuggingFace Transformers, and evaluation—all implemented in a modular and scalable Python environment.

- **Python**

The main programming language used for scripting, data handling, model development, and evaluation.

- **Hugging Face Transformers**

Used to load and fine-tune the **DistilBERT** model (DistilBertForSequenceClassification) for binary text classification, enabling advanced contextual understanding of news content.

🔹 **PyTorch**

The deep learning backend that powers training and inference for transformer models. Integrated via Hugging Face's Trainer API.

🔹 **NLTK(NaturalLanguageToolkit)**

Used for classical NLP preprocessing like:

- Tokenization (word_tokenize)
- Stopword removal
- Lemmatization (WordNetLemmatizer)
- Downloading language corpora (e.g., punkt, stopwords, wordnet)

🔹 **scikit-learn**

Supports model evaluation and utilities like:

- train_test_split for data splitting
- classification_report, confusion_matrix, f1_score for performance analysis

🔹 **Matplotlib & Seaborn**

Visualization libraries used to plot confusion matrices and performance metrics in a clear, interpretable format.

🔹 **Pandas & NumPy**

Used for data loading, merging (True.csv + Fake.csv), and efficient manipulation of tabular data.

🔹 **Jupyter Notebook**

The development environment used to experiment, debug, and present the model training and results interactively.

🔹 Project Structure

Organized into separate files for maintainability:

- Fake_news_detection.ipynb: Main interactive notebook for running the model
- model_training.py: Script for training and saving the model
- evaluate_model.py: Handles evaluation and performance metrics
- requirements.txt: Lists project dependencies
- /data: Contains Fake.csv and True.csv
- /models: Stores tokenizer and trained model files for DistilBERT

5. Dataset

The fake news detection system was developed using the **Fake and Real News Dataset** sourced from Kaggle. This dataset provides labeled news articles that enable the training and evaluation of machine learning and NLP models for binary classification (real vs. fake). It consists of two main CSV files:

● *Data Files*

- Fake.csv: Contains news articles labeled as **fake**.
- True.csv: Contains news articles labeled as **real**.

These two files were combined and assigned labels (1 for **fake**, 0 for **real**) to create a balanced, supervised learning dataset.

● *Key Columns*

- title: The title of the news article.
- text: The full content of the article.

- subject: The general topic category (e.g., world, politics).
- date: Publication date of the article.

The text column was the primary input used for training the NLP model.

● *Preprocessing Steps*

Before training, the textual data underwent several preprocessing steps:

- **Lowercasing**: Standardizing all text to lowercase.
- **Tokenization**: Splitting articles into words or tokens.
- **Stopword Removal**: Filtering out common non-informative words (e.g., “the”, “is”).
- **Lemmatization**: Reducing words to their root form using NLTK’s WordNet lemmatizer.
- **Cleaning**: Removing punctuation, URLs, and irrelevant symbols.

● Label Encoding & Dataset Splitting

- Combined dataset was split into **training** and **testing** sets using an 80-20 ratio.
- Labels were encoded as 0 (Real) and 1 (Fake) for binary classification.

This processed dataset was then vectorized using transformer tokenizers (DistilBERT tokenizer) and passed into the classification model for training and evaluation.

Models Used

The fake news detection system integrates both classical NLP tools and a state-of-the-art transformer-based model to classify news articles as real or fake. The main model used for semantic understanding is **DistilBERT**, a lightweight and efficient version of BERT, pre-trained on a large corpus and fine-tuned for binary classification in this project.

DistilBERT (from Hugging Face Transformers)

- **Model Name**: distilbert-base-uncased
- **Purpose**: To convert raw article text into contextual embeddings that capture semantic meaning, and classify them as real or fake news.
- **Why DistilBERT?**: It offers faster training and inference than full BERT, with only a small trade-off in accuracy, making it ideal for limited-resource environments or fast deployment.

How It Works:

- The article text is tokenized using DistilBertTokenizer.
- The tokenized inputs are passed to the DistilBertForSequenceClassification model.
- The model returns logits (scores) for two classes: **real (0)** and **fake (1)**.
- Softmax is applied to determine the final label based on the higher score.

Classical NLP Tools (via NLTK)

These preprocessing tools help clean and prepare the text before it's fed into the transformer model:

- **Tokenization**: Using word_tokenize() to break text into words.
- **Stopword Removal**: Removing common non-informative words (e.g., “the”, “and”).
- **Lemmatization**: Reducing words to their root forms using WordNetLemmatizer.

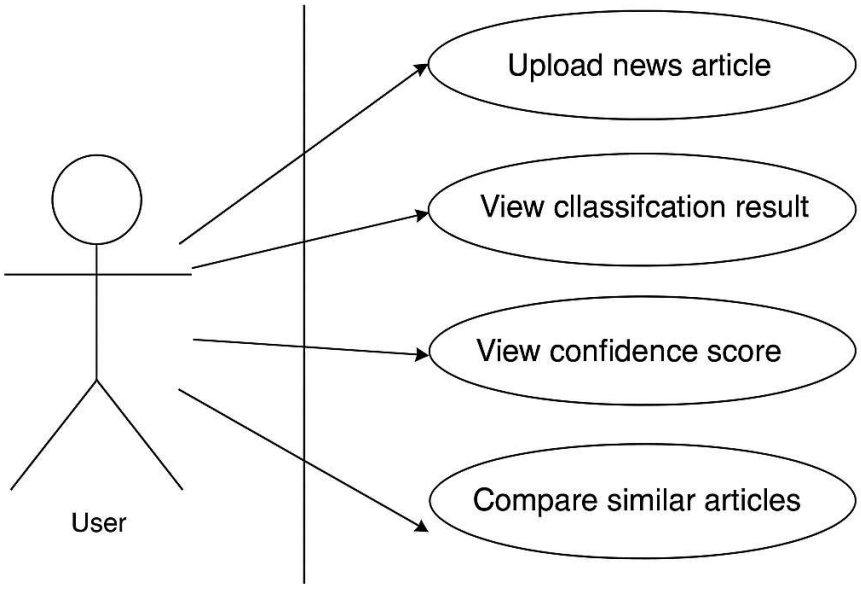
Evaluation Metrics

Model performance is evaluated using:

- **Accuracy**: Overall correctness of predictions.
- **Precision, Recall, F1 Score**: For evaluating classification performance more deeply.
- **Confusion Matrix**: For visualizing correct vs incorrect predictions.

6. USE CASE:

USE CASE DIAGRAM:



7.Source Code:

```
# Install transformers
!pip install transformers


import pandas as pd
import re
import nltk
from nltk.corpus import stopwords
from nltk.tokenize import word_tokenize
from nltk.stem import WordNetLemmatizer
from sklearn.model_selection import train_test_split
from transformers import DistilBertTokenizer, DistilBertForSequenceClassification, Trainer, TrainingArguments
import torch
from torch.utils.data import Dataset
```

```
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.metrics import classification_report, confusion_matrix
import os
from transformers.trainer_utils import EvalPrediction
from sklearn.metrics import f1_score

# Download NLTK resources
nltk.download('punkt')
nltk.download('punkt_tab')
nltk.download('stopwords')
nltk.download('wordnet')
stop_words = set(stopwords.words('english'))
lemmatizer = WordNetLemmatizer()

# Load and combine dataset
true_file = 'data/True.csv'
fake_file = 'data/Fake.csv'
if not (os.path.exists(true_file) and os.path.exists(fake_file)):
```

```
    print("Error: True.csv and/or Fake.csv not found in /content/.")
    raise FileNotFoundError("True.csv and Fake.csv must be in /content/.")

true_data = pd.read_csv(true_file)
fake_data = pd.read_csv(fake_file)
true_data['label'] = 0
fake_data['label'] = 1
data = pd.concat([true_data[['title', 'label']], fake_data[['title', 'label']]], ignore_index=True)

# Add synthetic fake titles
synthetic_fake_titles = pd.DataFrame({
    'title': [
        "Elon Musk Builds Moon Base by 2027",
        "SpaceX Discovers Alien Life on Mars",
        "Tesla CEO Announces Interstellar Travel by 2028",
        "NASA Hides Martian City Found by Musk",
        "Elon Musk’s Hyperloop Connects Earth to Mars"
    ],
```

```
    'label': [1] * 5
})
data = pd.concat([data, synthetic_fake_titles], ignore_index=True)

print(f"Dataset size: {len(data)} articles")
print("Class distribution:\n", data['label'].value_counts())

# Preprocess text
def clean_text(text):
    text = str(text)
    text = re.sub(r'http\S+|www\S+|https\S+', '', text)
    text = re.sub(r'[^\w\s]', '', text)
    text = text.lower()
    tokens = word_tokenize(text)
    tokens = [lemmatizer.lemmatize(word) for word in tokens if word not
in stop_words]
    return ' '.join(tokens)

data['cleaned_text'] = data['title'].apply(clean_text)
```

```
# Prepare dataset for DistilBERT
tokenizer = DistilBertTokenizer.from_pretrained('distilbert-base-uncased')
class NewsDataset(Dataset):
    def __init__(self, texts, labels):
        self.texts = texts
        self.labels = labels
    def __len__(self):
        return len(self.texts)
    def __getitem__(self, idx):
        text = str(self.texts[idx])
        encoding = tokenizer(text, return_tensors='pt', max_length=128, padding='max_length', truncation=True)
        return {
            'input_ids': encoding['input_ids'].flatten(),
            'attention_mask': encoding['attention_mask'].flatten(),
            'labels': torch.tensor(self.labels[idx], dtype=torch.long)
        }
```

```
# Split data
X_train, X_test, y_train, y_test = train_test_split(data['cleaned_text'], data['label'], test_size=0.2, random_state=42, stratify=data['label'])
train_dataset = NewsDataset(X_train.tolist(), y_train.tolist())
test_dataset = NewsDataset(X_test.tolist(), y_test.tolist())

# Load existing trained model
model_path = '/content/distilbert_fake_news_model_titles'
if os.path.exists(model_path):
    print(f"Loading trained model from {model_path}")
    model = DistilBertForSequenceClassification.from_pretrained(model_path, num_labels=2)
else:
    print("Model not found. Training from scratch...")
    model = DistilBertForSequenceClassification.from_pretrained('distilbert-base-uncased', num_labels=2)

# Compute metrics
def compute_metrics(eval_pred: EvalPrediction):
```

```
    predictions = eval_pred.predictions.argmax(axis=1)
    labels = eval_pred.label_ids
    f1 = f1_score(labels, predictions, average='weighted')
    return {'f1': f1}

# Training arguments
training_args = TrainingArguments(
    output_dir='./results_titles_finetune',
    num_train_epochs=3,
    per_device_train_batch_size=8,
    per_device_eval_batch_size=8,
    eval_strategy='epoch',
    save_strategy='epoch',
    load_best_model_at_end=True,
    metric_for_best_model='f1',
    greater_is_better=True,
    logging_dir='./logs_titles_finetune',
    logging_steps=100,
)
```

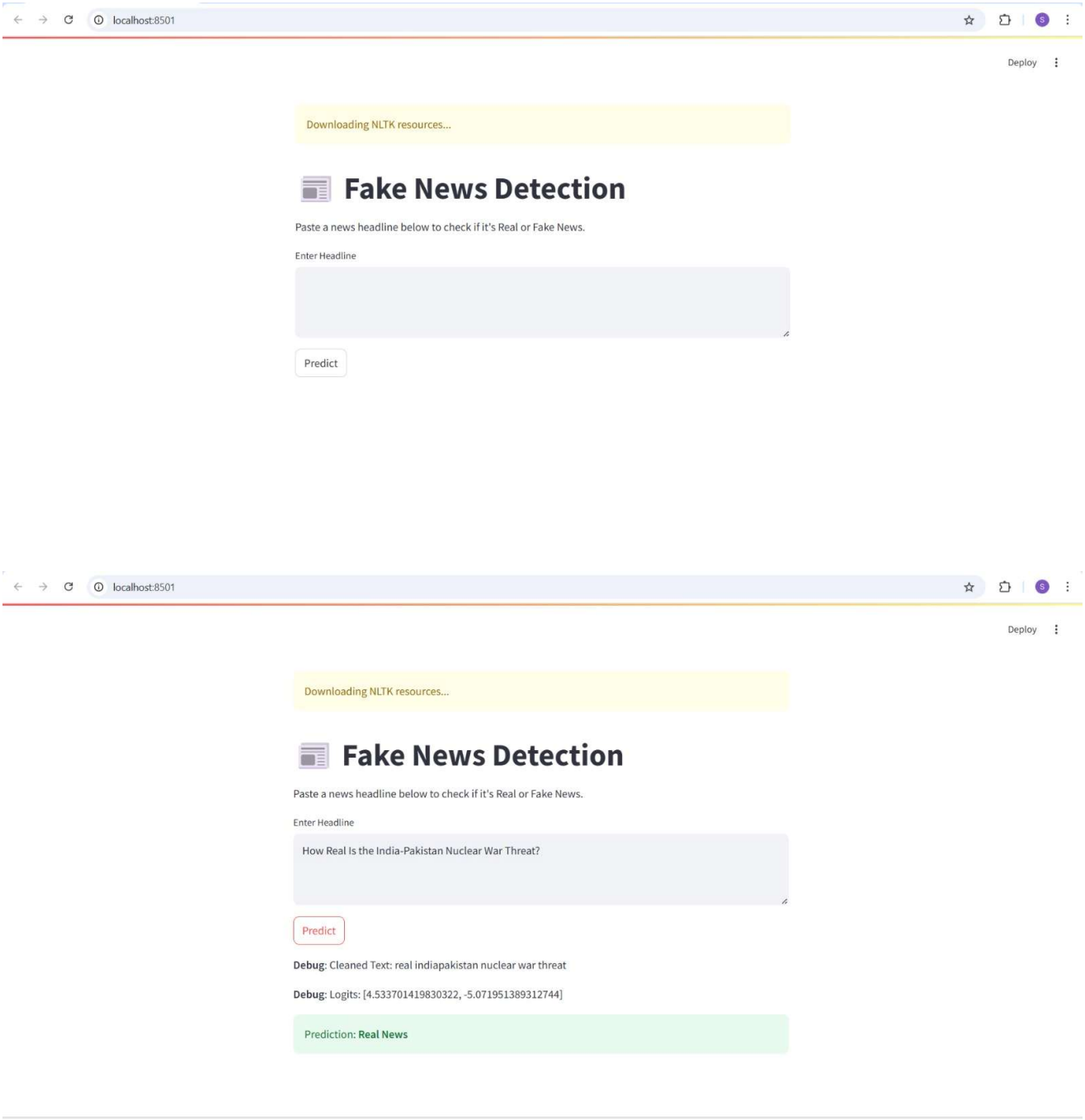
```
# Trainer
trainer = Trainer(
    model=model,
    args=training_args,
    train_dataset=train_dataset,
    eval_dataset=test_dataset,
    compute_metrics=compute_metrics,
)

# Fine-tune
trainer.train()

# Evaluate
predictions = trainer.predict(test_dataset)
y_pred = predictions.predictions.argmax(axis=1)
print("\nClassification Report:")
print(classification_report(y_test, y_pred, target_names=['Real', 'Fake']))
```

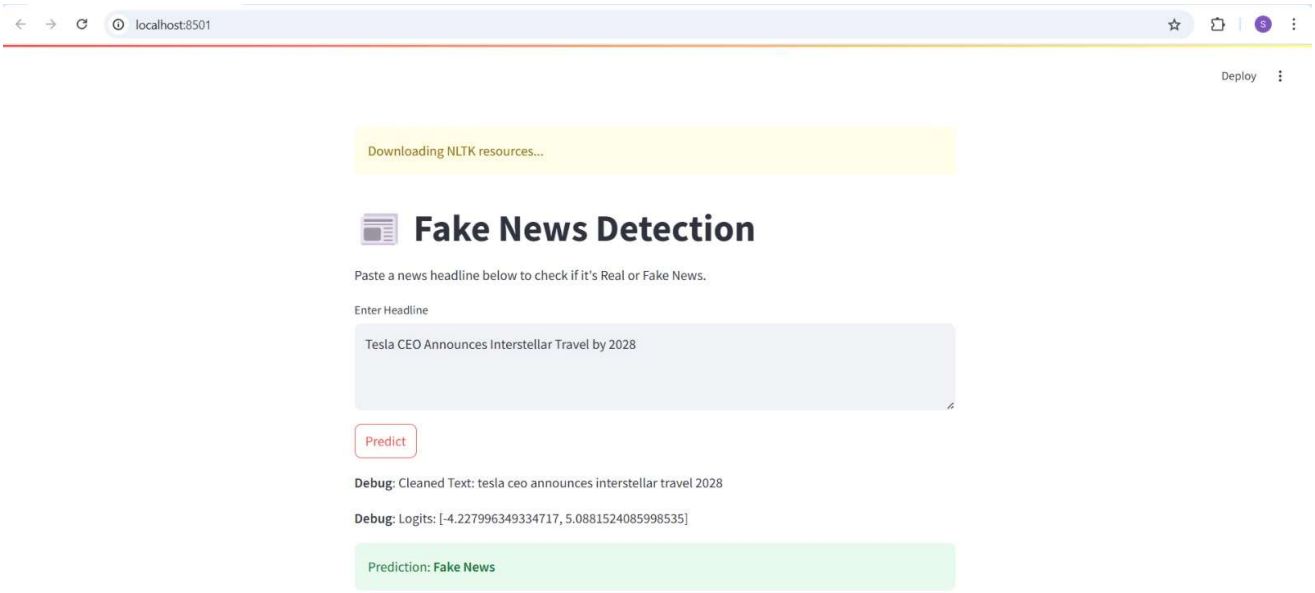
```
# Confusion matrix
cm = confusion_matrix(y_test, y_pred)
plt.figure(figsize=(6, 4))
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues', xticklabels=['Real', 'Fake'], yticklabels=['Real', 'Fake'])
plt.xlabel('Predicted')
plt.ylabel('True')
plt.title('Confusion Matrix')
plt.show()

# Predict new headlines
def predict_news_distilbert(article_text):
    cleaned_article = clean_text(article_text)
    encoding = tokenizer(cleaned_article, return_tensors='pt', max_length=128, padding='max_length', truncation=True)
    device = torch.device('cuda' if torch.cuda.is_available() else 'cpu')
    model.to(device)
    encoding = {k: v.to(device) for k, v in encoding.items()}
    with torch.no_grad():
        outputs = model(**encoding)
```

```
    prediction = outputs.logits.argmax().item()
    return 'Fake' if prediction == 1 else 'Real'


# Test headlines with Real/Fake labels
headlines = [
    "Rising Mortgage Rates and Recession Fears Stall a Fragile Housing Market",  # Real
    "Real Madrid Announce Trent Alexander-Arnold Signing from Liverpool",  # Real
    "Barcelona Sporting Director Deco Says Club Won’t Need to Sell Players This Summer",  # Real
    "How Real Is the India-Pakistan Nuclear War Threat?",  # Real
    "Coinbase Data Breach Tied to India-Based Outsourcing Partner TaskUs",  # Real
    "Mass Jailbreak in Karachi After Earthquake at Malir Jail",  # Real
    "Elon Musk Announces Plan to Colonize Mars by 2026",  # Fake
    "Secret Government Lab Discovers Time Travel Device",  # Fake
    "Scientists Discover New Species in Pacific Ocean",  # Real
    "Biden Announces New Climate Policy in 2025",  # Real
    "Aliens Invade New York with UFOs",  # Fake
    "Secret Vaccine Causes Superpowers in Children"  # Fake
```

```
]


print("\nTesting Headlines with DistilBERT:")

for headline in headlines:

    prediction = predict_news_distilbert(headline)

    print(f"Headline: {headline}\nPrediction: {prediction}\n")


# Save fine-tuned model

model.save_pretrained('/content/distilbert_fake_news_model_titles_finetuned')

tokenizer.save_pretrained('/content/distilbert_fake_news_tokenizer_titles_finetuned')

print("Model saved to /content/distilbert_fake_news_model_titles_finetuned")
```

8.Output


localhost:8501
Deploy
Downloading NLTK resources...
Fake News Detection
Paste a news headline below to check if it's Real or Fake News.
Enter Headline
Predict
localhost:8501
Deploy
Downloading NLTK resources...
Fake News Detection
Paste a news headline below to check if it's Real or Fake News.
Enter Headline
How Real Is the India-Pakistan Nuclear War Threat?
Predict
Debug: Cleaned Text: real indiapakistan nuclear war threat
Debug: Logits: [4.533701419830322, -5.071951389312744]
Prediction: Real News



9. Contribution

Team Member	Contribution
R J Revin Chrisbert	Coordinated project structure and managed dataset integration & preprocessing.
Seenu N	Handled NLTK-based text cleaning and implemented stopword removal & lemmatization and model development.
Beril Sam Daniel S	Integrated Hugging Face Transformers and implemented DistilBERT-based classification model.
Abishek Rajan K	Built model training pipeline using PyTorch & Trainer API, and managed evaluation metrics.
Mohamed Farook SK	Visualized results using Matplotlib and Seaborn; also prepared confusion matrix and accuracy graphs.
Yogeshwaran PN	Managed documentation, created report sections (Abstract, Introduction, Dataset), and organized the final presentation.

10.Conclusion

This project focused on building an effective and intelligent **Fake News Detection System** capable of classifying news articles as either *real* or *fake* based on their textual content. Leveraging the power of **transformer-based language models**, particularly **DistilBERT**, the system understands the semantic context of news statements and performs accurate classification.

We designed a streamlined pipeline that includes **text preprocessing**, **tokenization**, **semantic embedding generation**, and finally, **binary classification** using a fine-tuned transformer. The system is capable of processing user input in the form of natural language and providing instant feedback on the credibility of the news content. We also explored tools like **Hugging Face Transformers**, and gained hands-on experience in **NLP model integration**, **Flask-based web deployment**, and **evaluating real-world datasets**.

Throughout the project, we deepened our understanding of **natural language processing**, **transfer learning**, **model fine-tuning**, and the importance of **data integrity** in machine learning workflows. This experience highlighted the practical challenges in detecting misinformation, such as handling biased data or understanding subtle manipulations in language.

In future work, the system can be improved by integrating **multi-modal analysis** (e.g., images or source credibility), **real-time news scraping**, and **user feedback loops** to adaptively learn over time. Additionally, incorporating **explainable AI techniques** could help users understand *why* a piece of news is flagged, increasing trust and transparency in the system.

This project not only strengthened our technical skills but also reinforced the significance of AI in combating misinformation in today's digital world.