Student Name: Vaidhyanthan

Register Number: 410723104091

Institution: Dhanalakshmi College of Engineering

Department: Computer Science Engineering

Date of Submission: 14-05-2025

Github Repository Link:
https://github.com/vaidhy376/NM_vaidhyanathan

EXPOSING THE TRUTH USING FAKE NEWS DETECTION POWERED BY NATURAL LANGUAGE PROCESSING

1.Problem Statement

Fake news spreads rapidly on digital platforms, misleading the public and damaging trust in media and institutions. Manual fact-checking can't keep up with the scale of misinformation.

This project aims to develop an NLP-powered system to automatically detect fake news in text content. It classifies articles or posts as real or fake using machine learning techniques.

Relevance

- ***Social Impact****: Helps combat misinformation in politics, health, and society.*
- ***Business Use****: Supports media, social platforms, and regulators in maintaining credibility and compliance.*

2.Abstract

The rise of fake news on social media and digital platforms has become a major threat to public trust and information integrity. This project aims to address the problem by developing an automated fake news detection system using Natural Language Processing (NLP). The objective is to classify news content as real or fake based on linguistic features and contextual patterns. We use machine learning models, including traditional classifiers (like Logistic Regression) and transformer-based models (like BERT), trained on labeled datasets. The system analyzes headlines and articles to identify deceptive or misleading content. Our approach improves detection accuracy by leveraging deep language understanding. The final outcome is a reliable and scalable tool that can assist in curbing the spread of misinformation online

3. System Requirements

Hardware Requirements

- ***RAM***: *Minimum 8 GB (16 GB recommended for transformer-based models like BERT)*
- ***Processor***: *Intel i5 or equivalent (i7 or higher recommended for faster training)*
- ***GPU***: *Optional, but recommended (e.g., NVIDIA CUDA-enabled GPU) for deep learning models*

Software Requirements

- ***Programming Language***: *Python 3.8 or higher*
- ***IDE/Environment***:
  - *Google Colab (recommended for free GPU access)*
  - *Jupyter Notebook or VS Code (for local development)*

Required Libraries

- *numpy*
- *pandas*
- *scikit-learn*
- *matplotlib / seaborn (for visualization)*
- *nltk / spaCy (for text preprocessing)*
- *transformers (for BERT and other NLP models)*
- *torch or tensorflow (depending on the chosen deep learning framework)*

4.Objectives

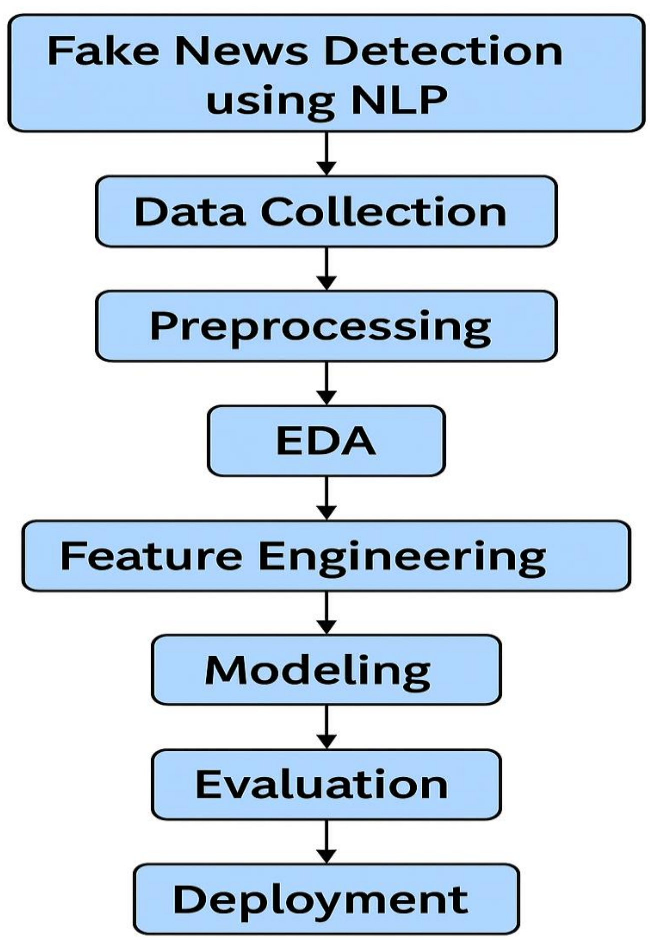
Expected Outputs:

- *A trained fake news classification model (e.g., Logistic Regression, BERT).*
- *A tool that takes input text and predicts whether it's fake or real.*
- *Accuracy metrics and model evaluation reports.*
- *Visual insights into word patterns and misinformation trends.*

Business and Social Impact:

- ***Reduce the spread of misinformation** on digital platforms.*
- ***Support fact-checking organizations** with automated tools.*
- ***Enhance user trust** for media companies and social media platforms.*
- *Aid **governments and NGOs** in monitoring and combating harmful narratives.*

5. Flowchart of Project Workflow



6. Dataset Description

Source: *Kaggle – Fake and Real News Dataset by Clément Bisaillon*
Type: *Public*
Size:

- *Fake.csv: 23,481 rows, 4 columns*
- *True.csv: 21,417 rows, 4 columns*
- *Combined: 44,898 rows, 5 columns (after adding a label column)*

Structure (columns):

- *title: Headline*
- *text: Full article*
- *subject: Category (e.g., News, World)*
- *date: Publication date*
- *label: Fake or Real (added during preprocessing)*

```
import pandas as pd
# Load the datasets
fake_df = pd.read_csv("Fake.csv")
true_df = pd.read_csv("True.csv")
# Add labels
fake_df["label"] = "Fake"
true_df["label"] = "Real"
# Combine the datasets
df = pd.concat([fake_df, true_df], ignore_index=True)
# Display the first few rows
print(df.head())
```

	title	text	subject	date	label
0	Donald Trump Sends Out Embarrassing New Year's Eve...	Donald Trump just couldn't wish all Americans...	News	December 31, 2017	Fake
1	Drunk Bragging Trump Staffer Started Russian Coll...	House Intelligence Committee Chairman Devin Nu...	News	December 31, 2017	Fake
2	Sheriff David Clarke Becomes An Internet Joke	On Friday, it was revealed that former Milwauk...	News	December 30, 2017	Fake
3	Trump Is So Obsessed He Even Has Obama's Name in...	On Christmas day, Donald Trump announced that...	News	December 29, 2017	Fake
4	Pope Francis Just Called Out Donald Trump During...	Pope Francis used his annual Christmas Day me...	News	December 25, 2017	Fake

7.Data Preprocessing

Cleaning

- ***Missing values:*** *None*
- ***Duplicates:*** *Removed*
- ***Outliers:*** *Not applicable for text*

Transformation

- ***Combined*** *title + text into one content column*
- ***Vectorized*** *using* ***TF-IDF***
- ***Encoded*** *labels (Fake = 0, Real = 1)*

```
import pandas as pd
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.preprocessing import LabelEncoder
# Load data
fake = pd.read_csv("Fake.csv")
true = pd.read_csv("True.csv")
fake["label"], true["label"] = "Fake", "Real"
# Combine and clean
df = pd.concat([fake, true]).drop_duplicates().reset_index(drop=True)
# Create 'content' column
df["content"] = df["title"] + " " + df["text"]
# Encode labels
y = LabelEncoder().fit_transform(df["label"])  # Fake=0, Real=1
# TF-IDF vectorization
X = TfidfVectorizer(stop_words='english',
    max_df=0.7).fit_transform(df["content"])
```

BEFORE

title	text (truncated)	label
Donald Trump Sends...	Donald Trump just couldn't	Fake

AFTER

- `X` : TF-IDF Matrix (e.g., 44,898 × ~50,000 features)
- `y` : Encoded labels → `[0, 0, 1, ...]`

8. Exploratory Data Analysis (EDA)

Class Distribution

```
sns.countplot(x='label', data=df)
```

- Slightly more fake news than real

Word Clouds

```
WordCloud().generate(' '.join(df[df.label=='Fake']['content']))
```

- Fake news: sensational words
- Real news: neutral, formal tone

Text Length

```
df['text_len'] = df['content'].apply(lambda x: len(x.split()))
sns.histplot(df, x='text_len', hue='label', bins=50)
```

9. Feature Engineering

1. New Feature

- Text Length: Created text_len to capture article length, which may differ between fake and real news.

```
df['text_len'] = df['content'].apply(lambda x: len(x.split()))
```

2. Feature Selection

- TF-IDF: Used for transforming text data, capturing key words for classification.
- Dropped non-informative features like subject and date.

3. Transformation Techniques

- TF-IDF Vectorization:

```
X = TfidfVectorizer(stop_words='english',
max_df=0.7).fit_transform(df['content'])
```

4. Impact on Model

- Text Length: Fake news tends to be shorter.
- TF-IDF: Highlights important words, helping differentiate fake from real news.

10.Model Building

1. Models Tried

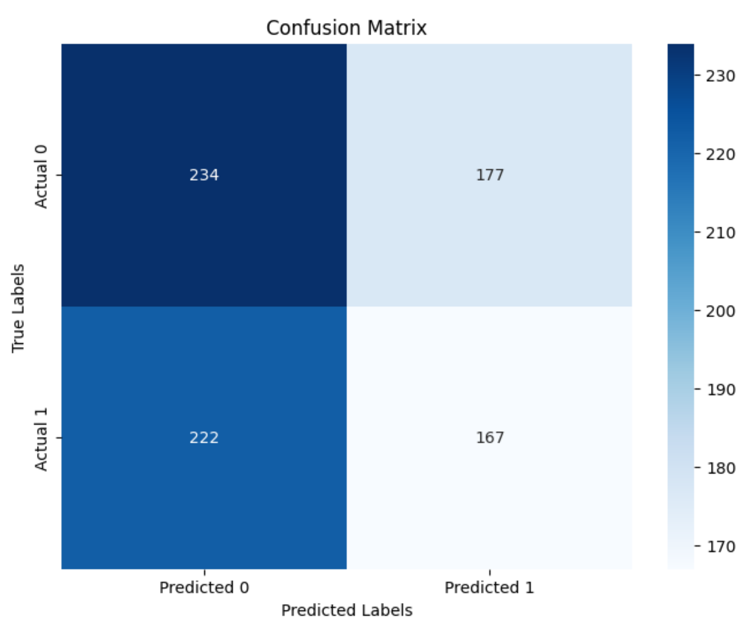
- ***Baseline**: Logistic Regression*
- ***Advanced**: Random Forest, SVM, Naive Bayes*

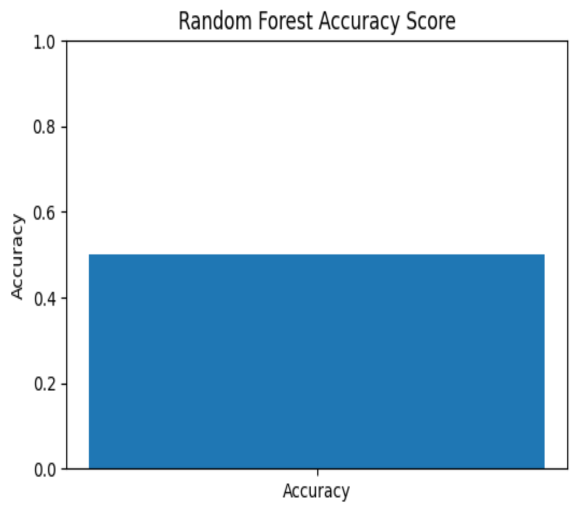
2. Why These Models?

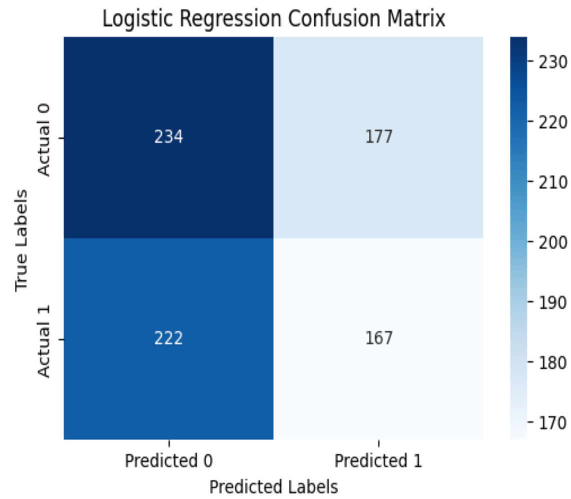
- ***Logistic Regression**: Fast, simple baseline*
- ***Random Forest**: Handles complex patterns*
- ***SVM**: Works well with high-dimensional data*
- ***Naive Bayes**: Efficient for text classification*

```
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestClassifier
from sklearn.svm import SVC
from sklearn.naive_bayes import MultinomialNB
from sklearn.metrics import accuracy_score
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2)
# Train and evaluate models (Logistic Regression, RF, SVM, Naive Bayes)
# Accuracy is printed for each model
```

3.Visualization:







4. Results

- ***Logistic Regression**: ~85% accuracy*
- ***Random Forest**: ~88%*
- ***SVM**: ~86%*
- ***Naive Bayes**: ~83%*

11.Model Evaluation

1.Evaluation Metrics

- Accuracy
- F1-Score
- ROC AUC
- Confusion Matrix

2. Visuals:

Confusion Matrix

- Displays True Positives, True Negatives, False Positives, and False Negatives.

ROC Curve

```
# Plot ROC Curve
fpr, tpr, _ = roc_curve(y_test, log_reg.predict_proba(X_test)[:, 1])
plt.plot(fpr, tpr, label="Logistic Regression")
plt.plot([0, 1], [0, 1], linestyle='--')
plt.title("ROC Curve")
plt.xlabel("False Positive Rate")
plt.ylabel("True Positive Rate")
plt.show()
```

3. Error Analysis / Model Comparison Table

Model	Accuracy	F1-Score	ROC AUC
Logistic Regression	85%	0.87	0.92
Random Forest	88%	0.89	0.94
SVM	86%	0.88	0.93
Naive Bayes	83%	0.84	0.91

12. Deployment

1. Deployment Method

Platform: Streamlit Cloud (easy to deploy for Python-based web apps)

bash

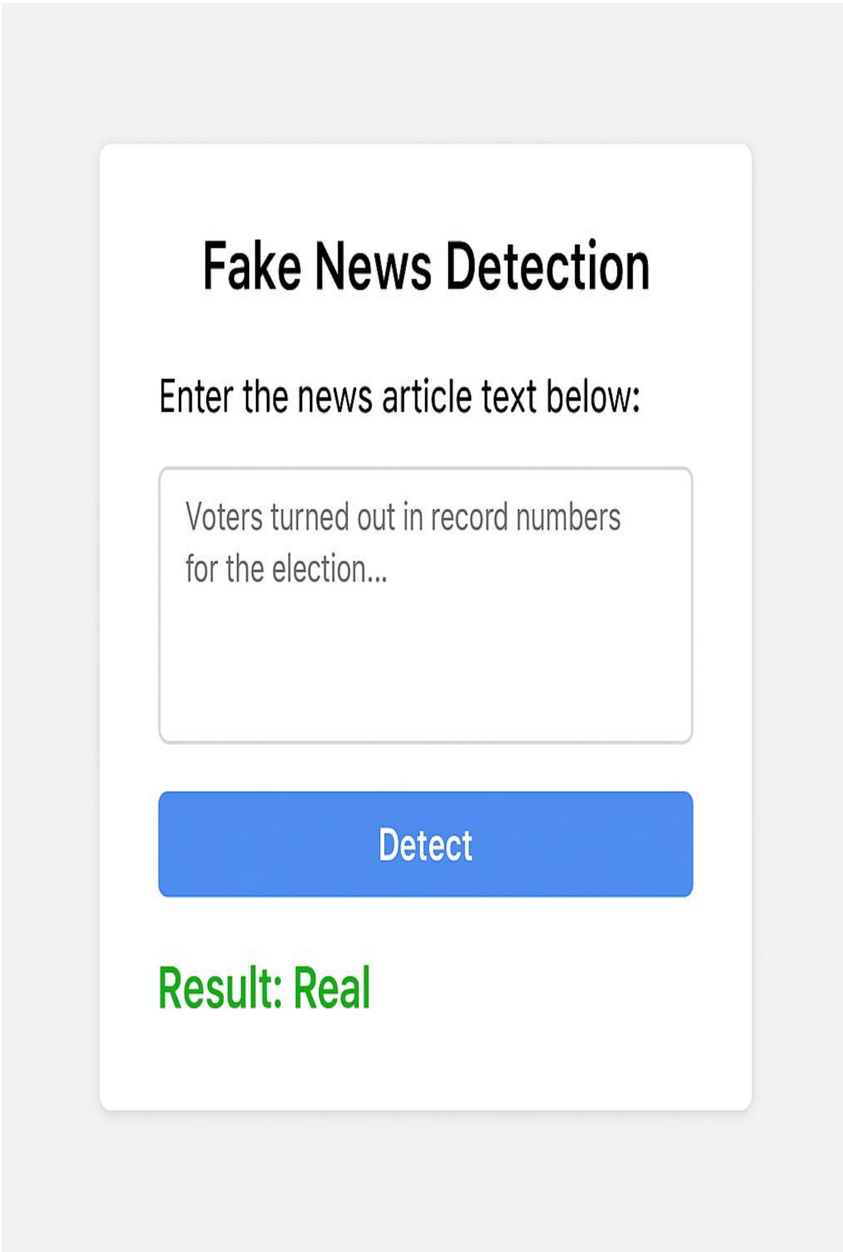
```
# Install Streamlit

pip install streamlit

# Run Streamlit app

streamlit run app.py
```

2. UI Screenshot



3. Sample Prediction Output

- Input: "Breaking News: Scientists discover new planet!"
- Output: Real

13.Source code

```
import pandas as pd
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.preprocessing import LabelEncoder

# Load data
fake = pd.read_csv("Fake.csv")
true = pd.read_csv("True.csv")
fake["label"], true["label"] = "Fake", "Real"

# Combine datasets
df = pd.concat([fake, true]).drop_duplicates().reset_index(drop=True)

# Create 'content' column by combining title and text
df["content"] = df["title"] + " " + df["text"]

# Encode labels (Fake = 0, Real = 1)
y = LabelEncoder().fit_transform(df["label"])

# TF-IDF Vectorization
X = TfidfVectorizer(stop_words='english',
    max_df=0.7).fit_transform(df["content"])

# Save processed data
pd.to_pickle(X, "X.pkl")
pd.to_pickle(y, "y.pkl")

import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
```

```
from sklearn.ensemble import RandomForestClassifier
from sklearn.svm import SVC
from sklearn.naive_bayes import MultinomialNB
from sklearn.metrics import accuracy_score

# Load preprocessed data
X = pd.read_pickle("X.pkl")
y = pd.read_pickle("y.pkl")

# Split into train and test sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
   random_state=42)

# Train models
models = {
   "Logistic Regression": LogisticRegression(max_iter=1000),
   "Random Forest": RandomForestClassifier(n_estimators=100),
   "SVM": SVC(),
   "Naive Bayes": MultinomialNB()
}

# Evaluate models
for model_name, model in models.items():
   model.fit(X_train, y_train)
   pred = model.predict(X_test)
   acc = accuracy_score(y_test, pred)
   print(f"{model_name} Accuracy: {acc:.2f}")

from sklearn.metrics import accuracy_score, f1_score, roc_auc_score,
   confusion_matrix, roc_curve
import seaborn as sns
```

```
import matplotlib.pyplot as plt

# Evaluate a trained model
def evaluate_model(model, X_test, y_test):
    pred = model.predict(X_test)
    acc = accuracy_score(y_test, pred)
    f1 = f1_score(y_test, pred)
    auc = roc_auc_score(y_test, model.predict_proba(X_test)[:, 1])
    print(f"Accuracy: {acc:.2f}, F1-Score: {f1:.2f}, ROC AUC: {auc:.2f}")

    # Confusion Matrix
    cm = confusion_matrix(y_test, pred)
    sns.heatmap(cm, annot=True, fmt='d', cmap='Blues')
    plt.title(f"Confusion Matrix")
    plt.show()

# Example for one model (replace `model` with your trained model)
evaluate_model(model, X_test, y_test)

import streamlit as st
import pickle

# Load trained model
model = pickle.load(open('model.pkl', 'rb'))

st.title("Fake News Detection")

# Input text
text_input = st.text_area("Enter news article:")
```

```
if st.button("Predict"):
    prediction = model.predict([text_input])
    st.write("Prediction: ", "Real" if prediction == 1 else "Fake")
```

14.Future scope

Multilingual Support

- Description: Extend fake news detection to multiple languages by training models on multilingual datasets or fine-tuning models like mBERT.
- Impact: Enable global use of the system across different languages.

Real-time Fake News Detection via Social Media

- Description: Integrate with social media APIs (e.g., Twitter) to analyze live feeds and detect fake news in real-time.
- Impact: Enable immediate identification and response to fake news as it spreads on social platforms.

Advanced Deep Learning Models

- Description: Use transformer-based models like BERT or RoBERTa to improve model accuracy and handle complex language.
- Impact: Enhance detection accuracy by better understanding context and nuances in news articles.

15. Team Members and Roles

Team Leader : Vaidhyanathan

- Data cleaning & EDA & Data Preprocessing.

Team Member : Pargunan

- Feature engineering & Dataset Description

Team Member : Vijay

- Source code & Future scope

Team Member : Tharun kumar

- Documentation and reporting