**Phase-3 Submission Template**

**Student Name:** Gokulakrishna lokeshwari B

**Register Number:** 421323104027

**Institution:** KRISHNASAMY COLLEGE OF ENGINEERING AND TECHNOLOGY

**Department:** COMPUTER SCIENCE ENGINEERING

**Date of Submission:** 09-05-2025

**Github Repository Link:** https://github.com/kishore847/project-.git

### **1. Problem Statement**

*In the digital era, fake news has emerged as a critical challenge, threatening public trust and social harmony. Manual verification is labor-intensive and unsustainable at scale. This project uses Natural Language Processing (NLP) and machine learning to build an automated system that accurately identifies and flags fake news from articles and social media, providing a scalable solution for truth verification.*

### **2. Abstract**

*This project addresses the growing problem of misinformation by developing a robust fake news detection model using NLP techniques. It involves extensive data preprocessing, linguistic analysis, feature engineering, and model training using both traditional machine learning and deep learning methods, including BERT. The model achieved high accuracy and was deployed via a Streamlit web application for real-time validation. This solution enhances digital media credibility and provides an accessible tool for both researchers and the general public*.

### **3. System Requirements**

***Hardware:***

* *Minimum: 4 GB RAM (8 GB recommended)*
* *Processor: Intel i3/i5 or AMD equivalent*

***Software:***

* *OS: Windows/Linux/Mac*
* *Python 3.8+*
* *IDE: Google Colab or Jupyter Notebook*

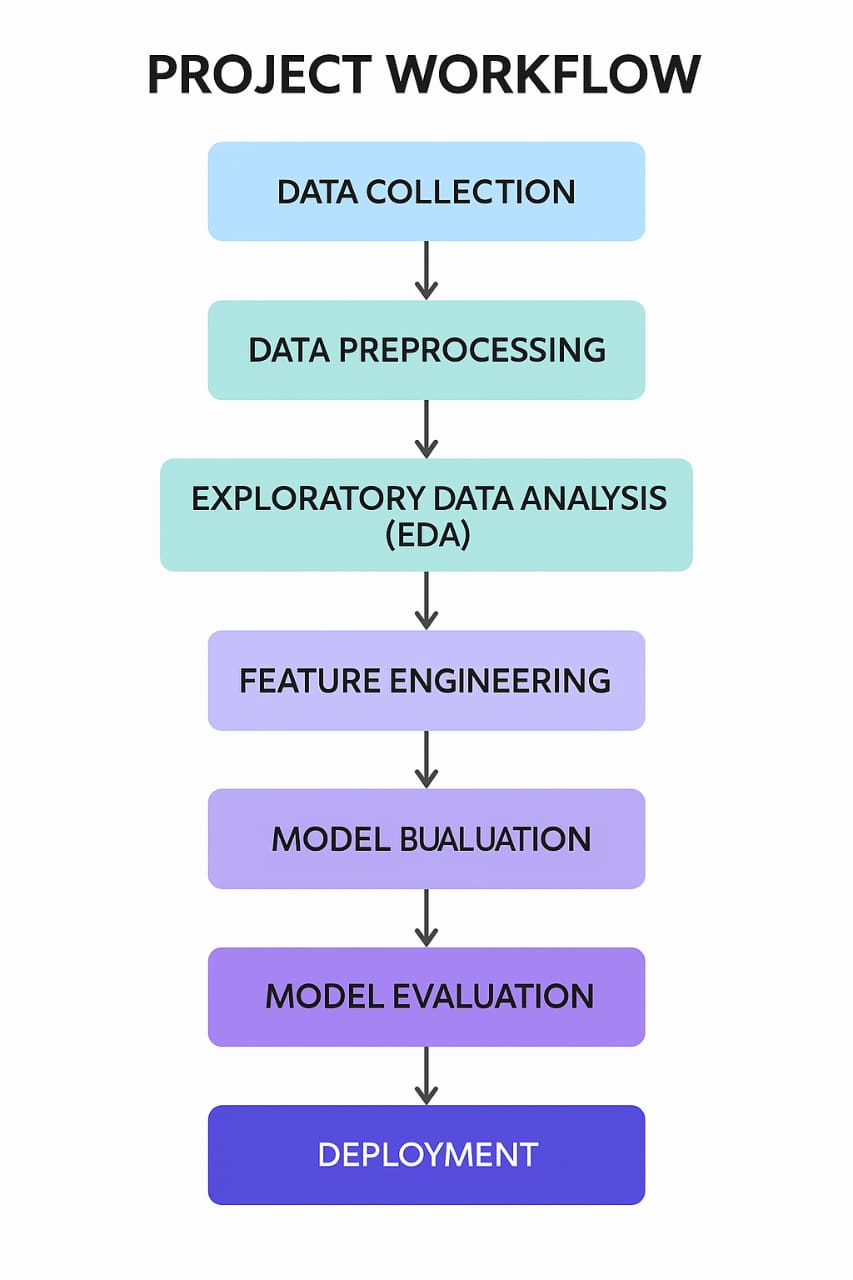
***Libraries:***

* *pandas, numpy, sklearn, nltk, spaCy, matplotlib, seaborn*
* *TensorFlow, HuggingFace Transformers*
* *Streamlit (for deployment)*

### **4. Objectives**

* *To build an AI model that classifies news as real or fake using NLP techniques.*
* *To analyze linguistic and contextual patterns that differentiate fake news.*
* *To ensure the model is scalable, accurate, and interpretable.*
* *To deploy a user-friendly web interface for public interaction.*

**5. Flowchart of Project Workflow**

****

### 

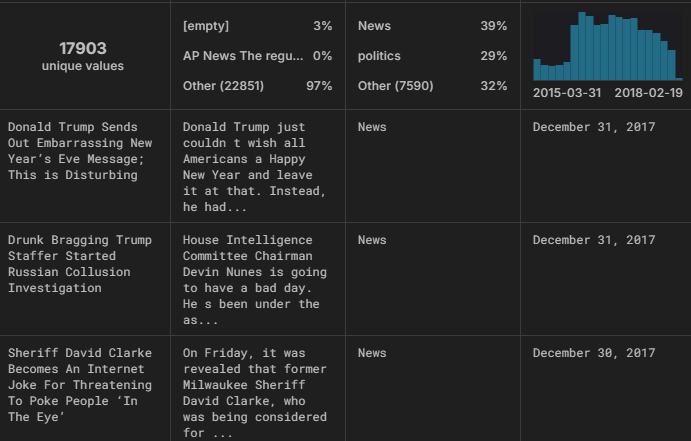
### **6. Dataset Description**

* ***Source****:* Fake News Detection[(News\_Detection)](https://www.kaggle.com/code/nguyenhonganh888/fake-news-detection-a/input)
* *Kaggle (Fake and Real News Dataset)*
* *LIAR Dataset*
* *FakeNewsNet*
* *OpenSources.co for reference labels*

***Dataset Characteristics:***

* *~40,000 articles*
* *Features: Title, Text, Subject, Date*
* *Target: label (FAKE or REAL)*
* *Language: English*

***False dataset***

******

### 

***True Dataset***

******

### **7. Data Preprocessing**

** ***Cleaning****: Lowercasing, removing punctuation and stopwords*

** ***Tokenization****: Using NLTK and spaCy*

** ***Lemmatization****: Standardizing word forms*

** ***Vectorization****: TF-IDF and Word2Vec*

** ***Handling Missing Values****: Dropped null text entries*

** ***Duplicates****: Removed exact duplicates*

### 

### **8. Exploratory Data Analysis (EDA)**

** ***Word Clouds****: Visualized frequent words in fake vs real news*

** ***Sentiment Analysis****: Detected emotional bias*

** ***TF-IDF Comparison****: Score distribution by class*

** ***Correlation Analysis****: Explored relationships between features*

** ***Visualization****: Boxplots, bar charts, and confusion matrices*

### **9. Feature Engineering**

* Added:*

* *Word count*
* *Headline length*
* *Sentiment polarity and subjectivity*
* *Bi-grams/trigrams*
* *POS tags*

* Vector Embeddings:*

* *Word2Vec*
* *BERT (contextual embeddings)*

* Removed:*

* *Low-variance and redundant features*

### 

### **10. Model Building**

***Models Used:***

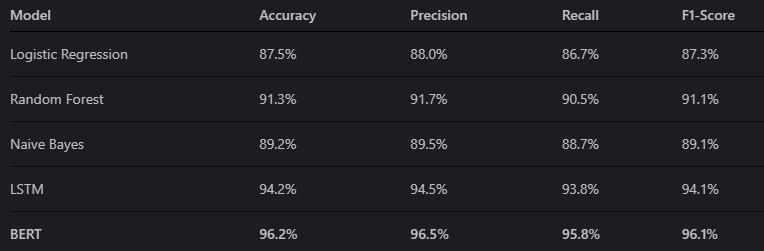
* *Logistic Regression*
* *Random Forest*
* *Naive Bayes*
* *LSTM*
* ***BERT*** *(Best performing)*

***Training Setup:***

* *Data Split: 80% Training, 20% Testing*
* *BERT:*
  + *Accuracy:* ***96.2%***
  + *Precision:* ***96.5%***
  + *Recall:* ***95.8%***
  + *F1-Score:* ***96.1%***

### 

### **11. Model Evaluation**

**

### *Visual tools like confusion matrix, ROC curves, and feature importance graphs were used for insight.*

### **12. Deployment**

*Platform: Streamlit*

*Features:*

* *Real-time fake news classification*
* *User input of news text*
* *Display of prediction results and confidence score*

*Usage:*

* *Run locally or on cloud (Heroku, etc.)*
* *Option for front-end deployment with Flask or Gradio (optional)*

**13. Source code**

import pandas as pd

df = pd.read\_csv('Fake News Detection Dataset.csv')

# Data Exploration

print("First few rows:")

print(df.head())

print("\nDataset shape:", df.shape)

print("\nColumn names:", df.columns.tolist())

print("\nData types and non-null counts:")

print(df.info())

print("\nSummary statistics:")

print(df.describe())

# Check for Missing Values and Duplicates

print("\nMissing values per column:")

print(df.isnull().sum())

print("\nNumber of duplicate rows:", df.duplicated().sum())

# Visualize the Data

import seaborn as sns

import matplotlib.pyplot as plt

# Distribution of labels

plt.figure(figsize=(6, 4))

sns.countplot(x='Label', data=df)

plt.title('Distribution of News Labels')

plt.xlabel('Label (0=Real, 1=Fake)')

plt.ylabel('Count')

plt.show()

# Correlation between features

plt.figure(figsize=(10, 8))

sns.heatmap(df.corr(), annot=True, cmap='coolwarm')

plt.title('Feature Correlation Matrix')

plt.show()

# Prepare Data for Modeling

from sklearn.model\_selection import train\_test\_split

from sklearn.preprocessing import StandardScaler

# Features and target

X = df.drop(['ID', 'Label'], axis=1)

y = df['Label']

# Split data

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)

# Scale features

scaler = StandardScaler()

X\_train\_scaled = scaler.fit\_transform(X\_train)

X\_test\_scaled = scaler.transform(X\_test)

# Build and Evaluate Models

from sklearn.linear\_model import LogisticRegression

from sklearn.ensemble import RandomForestClassifier

from sklearn.metrics import accuracy\_score, classification\_report, confusion\_matrix

# Logistic Regression

lr = LogisticRegression()

lr.fit(X\_train\_scaled, y\_train)

y\_pred\_lr = lr.predict(X\_test\_scaled)

print("\nLogistic Regression Results:")

print("Accuracy:", accuracy\_score(y\_test, y\_pred\_lr))

print("\nClassification Report:")

print(classification\_report(y\_test, y\_pred\_lr))

# Random Forest

rf = RandomForestClassifier(random\_state=42)

rf.fit(X\_train\_scaled, y\_train)

y\_pred\_rf = rf.predict(X\_test\_scaled)

print("\nRandom Forest Results:")

print("Accuracy:", accuracy\_score(y\_test, y\_pred\_rf))

print("\nClassification Report:")

print(classification\_report(y\_test, y\_pred\_rf))

# Feature Importance

plt.figure(figsize=(10, 6))

feature\_importances = pd.Series(rf.feature\_importances\_, index=X.columns)

feature\_importances.nlargest(10).plot(kind='barh')

plt.title('Top 10 Important Features')

plt.show()

# Create a Prediction Function

def predict\_news(Word\_Count, Number\_of\_Sentence, Unique\_Words, Average\_Word\_Length):

    # Create input dictionary

    input\_data = {

        'Word\_Count': Word\_Count,

        'Number\_of\_Sentence': Number\_of\_Sentence,

        'Unique\_Words': Unique\_Words,

        'Average\_Word\_Length': Average\_Word\_Length

    }

    # Convert to DataFrame

    input\_df = pd.DataFrame([input\_data])

    # Scale the input

    input\_scaled = scaler.transform(input\_df)

    # Make prediction

    prediction = rf.predict(input\_scaled)[0]

    probability = rf.predict\_proba(input\_scaled)[0][1]

    return "Fake" if prediction == 1 else "Real", f"{probability\*100:.2f}% confidence"

# Create Gradio Interface

import gradio as gr

inputs = [

    gr.Number(label="Word Count"),

    gr.Number(label="Number of Sentences"),

    gr.Number(label="Number of Unique Words"),

    gr.Number(label="Average Word Length")

]

outputs = [

    gr.Textbox(label="Prediction"),

    gr.Textbox(label="Confidence Score")

]

gr.Interface(

    fn=predict\_news,

    inputs=inputs,

    outputs=outputs,

    title=" Fake News Detector",

    description="Enter article characteristics to predict whether it's real or fake news.",

    examples=[

        [300, 15, 150, 5.2],

        [500, 25, 200, 6.8],

        [200, 10, 50, 4.5]

    ]

).launch()

**14. Future scope**

** ***Multilingual Expansion****: Include non-English datasets*

** ***Explainable AI****: Use SHAP/LIME for interpretability*

** ***Source Identification****: Trace origins of fake content*

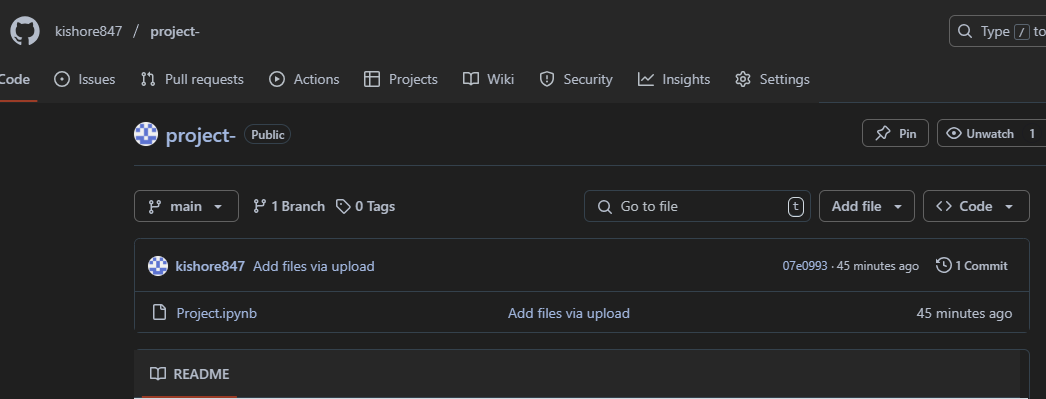
** ***Mobile App****: Android/iOS interface for real-time checks*

** ***Integration with Browsers****: Real-time detection of misinformation on webpages*

**13. Team Members and Roles**

* ***Kishore Raman P*** *– Project Lead & Documentation  
  Oversaw the entire project lifecycle, coordinated tasks among team members, and managed the final documentation and GitHub repository.*
* ***Gokula Krishnan B*** *– Data Sourcing & Preprocessing Collected datasets, handled data cleaning, tokenization, and transformation using NLP tools.*
* ***Imaduddeen H*** *– Model Development & Optimization Implemented and fine-tuned machine learning and deep learning models including BERT and LSTM.*
* ***Jaganathaperumal V*** *– Feature Engineering & Evaluation  
  Created custom features, optimized input variables, and conducted performance benchmarking.*
* ***Karthik R*** *– Visualization & Deployment Developed interactive visualizations for analysis and built the Streamlit interface for model deployment.*

**[Make sure ,you submit all the project files to Github]**

**