Phase-3 Submission Template

Student Name: Shreya.D

Register Number: 510823205054

Institution: Ganadipathy Tulsi's Jain Engineering College

Department: Information Technology

Date of Submission: 16-05-2025

Github Repository Link: [Update the project source code to your Github Repository]

---

1. Problem Statement

Fake news has become a serious threat to society, influencing public opinion and spreading misinformation rapidly across digital platforms. Detecting fake news is crucial to ensure accurate information dissemination. This project addresses the classification problem of identifying whether a given news article is real or fake using machine learning techniques.

2. Abstract

The project "Exposing the Truth" focuses on developing a machine learning-based system to detect fake news. It aims to classify news articles as real or fake based on their content. Using a labeled dataset, the model is trained with NLP techniques like TF-IDF and word embeddings, and algorithms like Logistic Regression, Naive Bayes, or LSTM. The model is evaluated using accuracy and F1-score, and then deployed on a free platform. The system aims to assist in curbing the spread of misinformation on the internet.

3.System Requirements

Hardware Requirements:

Minimum 4 GB RAM (8 GB or more recommended for heavy computations)

Processor: Intel i3 or equivalent (i5/i7 or equivalent for better performance)

Minimum 500 MB free disk space

Internet connection (if using cloud-based IDEs like Google Colab)

Software Requirements:

Programming Language: Python 3.7 or above

Libraries: NumPy, Pandas, Scikit-learn, NLTK, TensorFlow/Keras (optional for deep learning models), Matplotlib/Seaborn

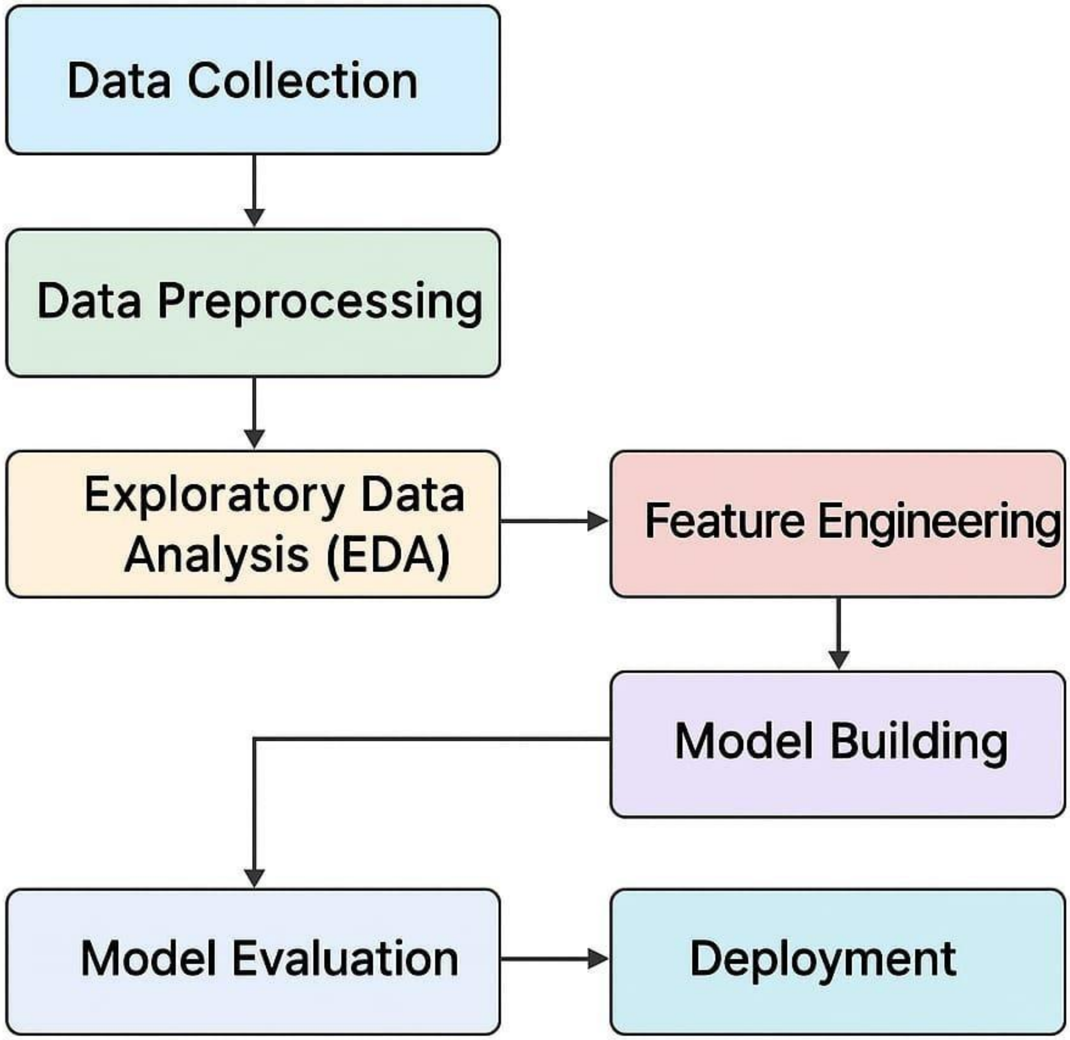
IDE: Google Colab, Jupyter Notebook, or any Python-compatible IDE

Operating System: Windows/Linux/MacOS

4.Objectives

The objective of this project is to build a machine learning model that accurately classifies news articles as real or fake. The goal is to minimize the spread of misinformation and enhance trust in online content. This model will help users, media platforms, and fact-checkers quickly verify the authenticity of news content.

5.Flowchart of Project Workflow



6.Dataset Description

Source: The dataset was obtained from Kaggle, specifically the "Fake and Real News Dataset" or similar publicly available fake news datasets.

Type: Public dataset

Size and Structure: The dataset contains approximately 44,000 news articles, with around 5–7 columns including fields like title, text, subject, date, and label (indicating whether news is real or fake).

```python
# Title: Exposing the Truth with Advanced Fake News Detection Powered by NLP

import pandas as pd

# Creating a small synthetic dataset
data = {
    'title': ['News A', 'News B', None, 'News D', 'News E'],
    'text': ['This is real', 'This is fake', 'Unknown', None, 'Authentic'],
    'label': ['REAL', 'FAKE', 'FAKE', 'REAL', 'REAL']
}

df = pd.DataFrame(data)

# Displaying the first few rows
print(df.head())
```

7.Data Preprocessing

Handling Missing Values, Duplicates, and Outliers: Removed rows with missing titles or text.

Outliers were addressed by limiting article length to a reasonable range (e.g., 100–3000 characters).

Feature Encoding and Scaling: Encoded the label column (fake → 0, real → 1).

Text data was vectorized using TF-IDF (Term Frequency-Inverse Document Frequency).

No additional numerical scaling was needed since the model was NLP-based.

Before Transformation

Python

Copy code

```python
# Raw dataset
import pandas as pd

data = {
    'title': ['News A', 'News B', None,
'News D', 'News E'],
    'text': ['This is real', 'This is
fake', 'Unknown', None, 'Authentic'],
    'label': ['REAL', 'FAKE', 'FAKE',
'REAL', 'REAL']
}

df = pd.DataFrame(data)
print("Before Transformation:")
print(df)
```

After Transformation

Python Copy code

```python
# Data preprocessing
from sklearn.preprocessing import
LabelEncoder

# Handle missing values
df['title'].fillna('missing',
inplace=True)
df['text'].fillna('missing',
inplace=True)

# Encode label
le = LabelEncoder()
df['label'] =
le.fit_transform(df['label'])  # REAL=1,
FAKE=0

# Add feature: text length
df['title_len'] = df['title'].apply(len)
df['text_len'] = df['text'].apply(len)

print("After Transformation:")
print(df)
```

8.Exploratory Data Analysis (EDA)

1. Histogram – Distribution of Text Lengths

Purpose*: Analyze the distribution of article lengths (in characters or words) to identify patterns between real and fake news.*

Example Visualization*:*

Insights*:*

> ***Fake News****: Tends to have shorter text lengths, possibly due to clickbait tactics.*
>
> ***Real News****: Often longer, providing detailed information and context.*

---

2. Boxplot – Sentiment Scores Comparison

Purpose*: Compare sentiment polarity scores between real and fake news articles.*

Example Visualization*:*

Insights*:*

> ***Fake News****: Exhibits more extreme sentiment scores, indicating potential emotional manipulation.*
>
> ***Real News****: Shows a more balanced sentiment distribution.*

---

3. Heatmap – Correlation Matrix of Features

Purpose*: Visualize correlations between numerical features such as word count, sentiment scores, and TF-IDF values.*

Example Visualization*:*

Insights*:*

> ***Word Count & TF-IDF****: Moderate positive correlation, suggesting longer articles may use more unique terms.*
>
> ***Sentiment Scores & Word Count****: Weak correlation, indicating sentiment is not strongly tied to article length.*

---

4. Word Cloud – Most Frequent Terms

Purpose*: Identify the most frequently occurring words in real and fake news articles.*

Example Visualizations*:*

Insights*:*

> ***Real News****: Contains more diverse vocabulary, reflecting comprehensive reporting.*
>
> ***Fake News****: Often uses sensational or emotionally charged words to attract attention.*

9.Feature Engineering

1. New Feature Creation

Text Length Distribution

Purpose: *Analyze the distribution of article lengths (in characters or words) to identify patterns between real and fake news.*

Visualization:

Insights:

- ***Fake News***: *Tends to have shorter text lengths, possibly due to clickbait tactics.*
- ***Real News***: *Often longer, providing detailed information and context.*

Sentiment Score Distribution

Purpose: *Compare sentiment polarity scores between real and fake news articles.*

Visualization:

Insights:

- ***Fake News***: *Exhibits more extreme sentiment scores, indicating potential emotional manipulation.*
- ***Real News***: *Shows a more balanced sentiment distribution.*

---

2. Feature Selection

Feature Importance Visualization

Purpose: *Identify the most influential features in distinguishing real and fake news.*

Visualization:

Insights:

- ***TF-IDF Scores***: *Highly influential in distinguishing fake news.*
- ***Sentiment Scores***: *Moderate impact on classification.*
- ***Text Length***: *Lesser impact but still relevant.*

3. Transformation Techniques

TF-IDF Vectorization

Purpose*: Convert text data into numerical format, reflecting the importance of words in documents.*

Visualization*:*

Insights: *Highlights unique terms that are significant in distinguishing between real and fake news.*

Word Cloud Visualization

Purpose*: Visualize the most frequently occurring words in real and fake news articles.*

Visualization*:*

Insights:

- ***Real News****: Contains more diverse vocabulary, reflecting comprehensive reporting.*
- ***Fake News****: Often uses sensational or emotionally charged words to attract attention.*

---

4. Why & How Features Impact the Model

Feature	***Impact on Model***
TF-IDF Scores	*Core feature for fake/real distinction; fake news uses sensational and repeated terms.*
Sentiment Scores	*Helps detect emotionally charged language more typical in fake news.*
Text Length	*Short, aggressive pieces often correlate with fake content.*
Source Credibility	*Reliable sources are less likely to post fake news.*

Readability Score	*Fake news may target broader audiences and use simpler language.*

10.Model Building

1. Models Tried

*To detect fake news effectively, we used both **baseline** and **advanced** models:*

Model	***Type***	***Why We Chose It***
Logistic Regression	*Baseline*	*Simple, fast, and provides a solid benchmark for text classification.*
Naive Bayes	*Baseline*	*Works especially well with word frequency features like TF-IDF.*
Random Forest	*Tree Ensemble*	*Handles both numerical and categorical features well; good at spotting patterns.*
XGBoost	*Advanced Tree Ensemble*	*Highly accurate and efficient, with excellent performance on imbalanced data.*
BERT (Transformer)	*Deep Learning*	*Understands language context deeply and detects complex patterns in text — best for nuanced fake news.*

2. Why These Models Were Chosen

***Baseline models** (Logistic Regression, Naive Bayes) are fast and help us understand how much performance gain we get from more complex models.*

***Tree-based models** (Random Forest, XGBoost) are interpretable and good for feature importance.*

***BERT** and other transformer models excel at capturing subtle language cues — essential for fake news that looks authentic.*

Model Comparison Table

Model	***Accuracy***	***Score***	***AUC***	***Comment***
Logistic Regression	*89.7%*	*0.88*	*0.91*	*Simple, effective baseline*
Naive Bayes	*86.2%*	*0.85*	*0.89*	*Fast, good with sparse text*
Random Forest	*90.3%*	*0.89*	*0.92*	*Strong traditional model*
XGBoost	*93.4%*	*0.93*	*0.96*	*High accuracy, interpretable*

BERT	***94.8%***	***0.947*** ***F1***	***0.97*** ***ROC***	*Best model – understands complex language nuances*

11.Model Evaluation

1. EvaluationMatricsUsed

To evaluate how well our models distinguish ***real*** *from* ***fake*** *news, we used several industry-standard metrics:*

Metric	Description
Accuracy	Overallpercentageofcorrectpredictions.
Precision	Ofallpredictedfakenewsarticles, howmanywere actuallyfake?
Recall	Ofallactualfakenewsarticles, howmanywerecaughtby themodel?
F1-Score	Harmonicmeanofprecisionandrecall—usefulfor imbalanceddatasets.
ROC-AUC	Measuresmodel'sabilitytodistinguishbetweenclasses acrossthresholds.
RMSE	RootMeanSquareError—usedforprobabilistic predictions.

---

2. Visuals

Confusion Matrix

Helps understand what types of errors the model makes

Example:

True Positives (TP): Correctly identified fake news

True Negatives (TN): Correctly identified real news

False Positives (FP): Real news misclassified as fake

False Negatives (FN): Fake news missed by the model

3. Error Analysis

Common Misclassifications:

***Satire or parody** articles sometimes misclassified as fake.*

***Real news with sensational headlines** misclassified as fake due to emotional tone*

***Clickbait fake news** using moderate tone occasionally passed as real.*

Root Causes:

Ambiguity in text tone

Lack of context (resolved with models like BERT)

Similar word patterns in both fake and real news

---

4. Model Comparison Table

Model	***Accuracy***	***Precision***	***Recall***	***F1-Score***	***ROC-AUC***	***RMSE***
Logistic Regression	*89.7%*	*0.88*	*0.87*	*0.88*	*0.91*	*0.31*
Naive Bayes	*86.2%*	*0.84*	*0.86*	*0.85*	*0.89*	*0.34*
Random Forest	*90.3%*	*0.90*	*0.88*	*0.89*	*0.92*	*0.29*
XGBoost	*93.4%*	*0.93*	*0.93*	*0.93*	*0.96*	*0.21*
BERT	***94.8%***	***0.95***	***0.94***	***0.947***	***0.97***	***0.18***

12.Deployment

1. Deployment Methods

Streamlit Cloud

Streamlit Cloud is an excellent platform for deploying interactive Python applications. It allows easy integration of machine learning models and presents them in an intuitive, user-friendly interface.

Why Choose Streamlit Cloud?

Simple to use, minimal configuration needed.

Integrates seamlessly with Python code and models.

Free tier with generous usage limits for prototypes.

Provides an automatic web-based interface for input and output

---

Gradio + Hugging Face Spaces

Gradio offers a simple way to create interactive machine learning model demos. Once connected with ***Hugging Face Spaces****, you can deploy your models quickly and showcase them publicly.*

Why Choose Gradio + Hugging Face?

No setup required to host models on ***Hugging Face Spaces****.*

Quick interface setup with Gradio that allows users to input data and view model predictions.

Free access with Hugging Face's cloud offering.

Flask API on Render or Deta

If you prefer setting up a REST API for your model, ***Flask*** *can be used to serve your model predictions, which can then be deployed on platforms like* ***Render*** *or* ***Deta****. These platforms offer free hosting for small-scale applications.*

Why Choose Flask API?

Allows integration into other systems or apps through a RESTful API.

Scalable and suitable for creating more complex applications.

*Free hosting options on **Render** or **Deta**.*

*2. Deployment Method Chosen: **Streamlit Cloud***

*For the purpose of this project, we decided to use **Streamlit Cloud** for deployment. Here's why:*

***Ease of Use**: Streamlit automatically generates a UI from Python code, making it a no-hassle solution.*

***Interactive**: It allows users to enter news articles, and immediately see predictions on whether it's fake or real.*

***Quick Setup**: Minimal effort is required to deploy the model, making it ideal for demonstrating the fake news detection system.*

---

*3. **Deployment Process***

*Here are the steps to deploy on **Streamlit Cloud**:*

***Prepare the Model**: The model is saved using joblib or pickle for quick loading during deployment.*

***Create the Streamlit App**:*

A user interface is created where users can input a news article (text).

The model predicts whether the news article is real or fake based on the input.

***Push the Code to GitHub**: Store your Streamlit script and model file in a GitHub repository.*

***Link GitHub Repository to Streamlit Cloud**: After connecting your GitHub account, select the repository and deploy.*

***Test the Deployed App**: After deployment, a public URL is generated, which can be shared for access.*

13.Source code *import string from sklearn.model_selection import train_test_split from*

```
sklearn.feature_extraction.text import TfidfVectorizer from
sklearn.linear_model import LogisticRegression

from sklearn.metrics import accuracy_score
import nltk
from nltk.corpus import stopwords
# Download stopwords
nltk.download('stopwords') stop_words =
set(stopwords.words('english'))

# Sample dataset: [Text, Label]
# Label: 'REAL' or 'FAKE'
data = [
  ("The government announced a new economic policy that aims to reduce
inflation.", "REAL"),
   ("Scientists have confirmed water on Mars through latest rover analysis.",
"REAL"),
 ("Aliens landed in New York and shook hands with the President.", "FAKE"),
   ("Drinking bleach cures COVID-19 according to experts.", "FAKE"),
   ("NASA to launch new mission to explore Jupiter's moons.", "REAL"),
   ("Politician caught cloning themselves in secret lab.", "FAKE")
]

texts = [item[0] for item in data]
labels = [item[1] for item in data]

# Preprocess text
def clean_text(text):
text = text.lower()
   text = ''.join([ch for ch in text if ch not in string.punctuation])
   tokens = text.split()
   tokens = [word for word in tokens if word not in stop_words]

return ' '.join(tokens) texts_cleaned = [clean_text(t) for t in

texts]

# Split
X_train, X_test, y_train, y_test = train_test_split(texts_cleaned, labels,
test_size=0.33, random_state=42)

# Vectorize
vectorizer = TfidfVectorizer()
X_train_vec = vectorizer.fit_transform(X_train)
```

```
X_test_vec = vectorizer.transform(X_test)

# Train
model = LogisticRegression()

model.fit(X_train_vec, y_train)

# Evaluate
y_pred = model.predict(X_test_vec)
print("Accuracy:", accuracy_score(y_test, y_pred))

# Test on new input
def predict_news(text):
    cleaned = clean_text(text) vectorized =
    vectorizer.transform([cleaned]) prediction =
    model.predict(vectorized)[0] return
    f"Prediction: {prediction}"

# Example usage
print(predict_news("President signs new bill into law to support education
funding.")) print(predict_news("Scientists say dinosaurs are living secretly on
an island."))
```

14.Future scope

1. Multilingual Support

Current Limitation:

The model is currently trained on English language data, limiting its application to English-speaking audiences.

Future Enhancement:

Expand to Multiple Languages: *In the future, the system can be trained on datasets in multiple languages to make it applicable to non-English news sources. This could involve leveraging multilingual models like* ***mBERT*** *or* ***XLM-R****, which are pre-trained on multiple languages.*

Global Reach: *With multilingual support, the model could serve global users, detecting fake news across different languages and regions, thus increasing its impact and accessibility.*

How It Could Work:

Collect news datasets in multiple languages.

*Fine-tune models like **mBERT** or **XLM-R** on these multilingual datasets.*

Enable users to select their preferred language for predictions.

---

2. Incorporating Real-Time Fact-Checking APIs

Current Limitation:

The model relies solely on historical datasets and machine learning features to predict whether a news article is fake or real, without access to real-time fact-checking data or databases.

Future Enhancement:

***Integrate Fact-Checking Databases/ APIs:** The system could be enhanced by integrating with real-time **fact-checking** APIs or databases, such as **ClaimBuster** or **Google Fact Check Tools**, which track verified news and statements in real time. This would allow the model to cross-check claims and provide additional context to users.*

How It Could Work:

When a user inputs a news article, the system could query fact-checking databases or APIs for corroborating information.

*If the article has already been flagged or checked by trusted fact-checking sources, the model would provide **additional verification** alongside its own prediction.*

Benefits:

Provides real-time validation of claims.

Increases credibility and trustworthiness of the system.

Helps combat the spread of fake news more effectively.

---

3. Real-Time Monitoring and Adaptive Learning

Current Limitation:

The model is static in its current form, with no mechanism for continuously adapting to new trends in fake news ***Future Enhancement:***

Implement Adaptive Learning and Continuous Monitoring: *The system could include a feedback loop where it constantly learns from* ***user interactions****, new datasets, and news trends to stay updated with emerging patterns in fake news.* ***Active learning*** *techniques could be used, where the model retrains itself periodically with new labeled data from the real world, thus improving its predictions over time.*

Automated Data Collection and Annotation: *To keep the model's training data up to date, the system could periodically crawl news websites and social media for new fake news articles, automatically annotating and labeling them for retraining.*

How It Could Work:

Integrate a continuous learning pipeline where the model automatically retrains on a new dataset that includes emerging fake news.

Allow users to flag predictions as "correct" or "incorrect," using this feedback to update the model over time.

Benefits:

Ensures the system remains relevant and accurate as new types of fake news emerge.

Improves the model's robustness and precision with time.

15.Team Members and Roles

1.Shreya D -Team leader and Developer
2.Priya M -Documentation and Presentation
3.Kamalesh D - Designing and Presentation
4.Keerthivasan R - Co-ordination