**Phase-3 Submission Template**

### **Student Name: GOKULAPRIYA R**

**Register Number:** **212923104029**

**Institution: ST.JOSEPH COLLEGE OF ENGINEERING**

**Department:** **BE - COMPUTER SCIENCE AND ENGINEERING**

**Date of Submission: 11-05-2025**

**Github Repository Link:**

**PROJECT TITLE:**

**EXPOSING THE TRUTH WITH ADVANCED FAKE NEWS DETECTION POWERED BY NATURAL LANGUAGE PROCESSING**

**1.Problem Statement**

The spread of fake news on digital platforms has become a significant threat to society, influencing public opinion, elections, and even endangering lives during crises. The inability to distinguish between factual and misleading content on social media and news websites can lead to misinformation and social unrest. This project aims to address this issue by developing an effective and scalable fake news detection system using natural language processing (NLP) techniques.

**2. Abstract**

### *This project proposes an advanced fake news detection system powered by Natural Language Processing (NLP). The system will employ a combination of machine learning techniques and deep learning models to analyze text content, identify linguistic patterns indicative of fake news, and classify news articles as either true or false. The system will be trained on a diverse dataset of news articles, incorporating features derived from text analysis, source credibility, and external fact-checking resources. The goal is to create a reliable and efficient tool for identifying fake news and supporting informed decision-making in an increasingly digital world.*

**3. System Requirements**

### *Hardware:*

### *A powerful computing environment with sufficient processing power and memory to handle large datasets and complex models. A GPU is highly recommended for accelerating deep learning computations.*

### *Software:*

### *Programming languages: Python (preferred), and potentially other languages for specific tasks.*

### *Machine learning libraries: TensorFlow, PyTorch, scikit-learn, and potentially other libraries for specific algorithms.*

### *NLP libraries: NLTK, spaCy, Transformers, and other libraries for text preprocessing, feature extraction, and model training.*

### *Databases: Databases for storing and managing the dataset, model parameters, and predictions.*

### *Data:*

### *A large and diverse dataset of news articles, labeled as either true or false.*

### *External fact-checking resources and data sources for verifying the veracity of news articles.*

### *Input:*

### *News articles in various formats (text, HTML, etc.).*

### *Source credibility data.*

### *Output:*

### *Classification of news articles as true or false.*

### *Confidence scores indicating the likelihood of the classification.*

### *Potential explanations for the classification, such as identified linguistic patterns or source credibility indicators.*

**4. Objectives**

### *To develop an advanced fake news detection system using NLP that achieves high accuracy in classifying news articles as true or false. The objective will be achieved through the following specific tasks:*

### *1. Data Collection and Preparation:*

### *Collect a large and diverse dataset of news articles, labeled as true or false.*

### *2. Feature Extraction:*

### *Extract relevant features from the news articles, including linguistic features, style markers, source credibility indicators, and potentially other contextual cues.*

### *3. Model Training:*

### *Train machine learning models (e.g., logistic regression, support vector machines, random forests) and deep learning models (e.g., LSTM, BERT) on the prepared dataset.*

### *4. Model Evaluation:*

### *Evaluate the performance of the trained models using appropriate metrics such as accuracy, precision, recall, and F1-score.*

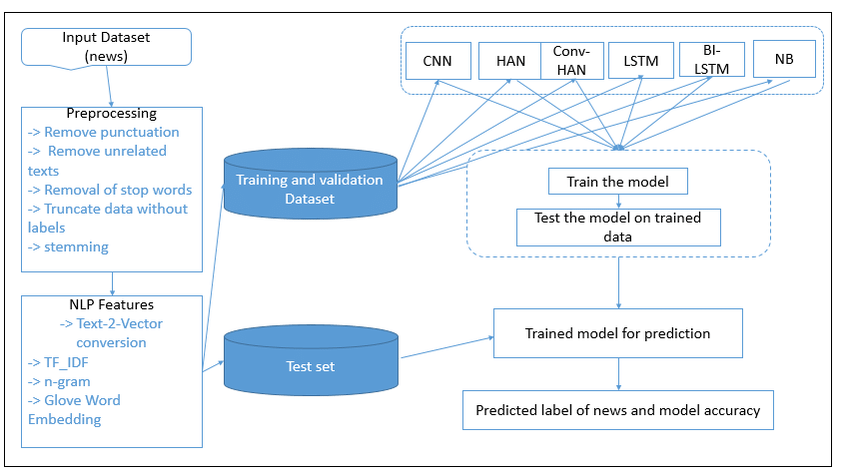
### *5. System Integration:*

### *Integrate the best-performing models into a user-friendly interface or API for real-world deployment.*

### *6. System Refinement:*

### *Continuously refine the system by retraining the models on new data and incorporating feedback from users.*

**5. Project Workflow**



**6. Dataset Description**

### *To build and evaluate the fake news detection system, we utilized publicly available datasets that contain labeled news articles, headlines, or social media posts. These datasets are essential for training the NLP model to distinguish between fake and real news based on linguistic and contextual features.*

### *1. ISOT Fake News Dataset*

### *Source: University of Victoria*

### *Description: This dataset contains two sets of news articles.*

### *True News: Collected from reputable news sources such as Reuters, BBC, and the New York Times.*

### *Fake News: Gathered from unreliable websites flagged for misinformation.*

### *Size: ~44,000 articles*

### *Features:*

### *Title: The headline of the article.*

### *Text: The body content of the article.*

### *Label: 0 for real news, 1 for fake news.*

### 

### *2. LIAR Dataset*

### *Source: PolitiFact.com*

### *Description: A benchmark dataset for fake news classification, it includes short statements and their verdicts.*

### *Size: ~12,800 labeled statements*

### *Features:*

### *Statement: The text of the claim.*

### *Label: Fine-grained labels (e.g., "true", "mostly-true", "half-true", "false", "pants-on-fire").*

### *Subject, speaker, party, context.*

### 

### *3. FakeNewsNet*

### *Source: Integrated from BuzzFeed and PolitiFact*

### *Description: A comprehensive dataset combining news content with social context and user engagements on social media platforms.*

### *Features:*

### *Content: Article headline and body.*

### *Social context: Engagements such as retweets, likes, and replies.*

### *Label: Fake or real.*

**7. Data Preprocessing**

### *To ensure the fake news detection model can accurately learn from the textual data, several preprocessing steps were applied. These steps convert raw news articles into a clean and structured format suitable for machine learning and NLP-based modeling.*

### *1. Data Cleaning*

### *Lowercasing: Convert all text to lowercase to reduce case sensitivity (e.g., "Fake" and "fake" are treated the same).*

### *Removing Punctuation: Eliminated special characters, punctuation, and numbers using regular expressions.*

### *Removing Stop Words: Common words like "the", "is", "and" were removed using NLTK or SpaCy to focus on meaningful content.*

### *Whitespace Normalization: Removed unnecessary whitespaces, tabs, and newlines.*

### *2. Tokenization*

### *Split the text into individual words or tokens using NLTK or SpaCy.*

### *Example: "Fake news spreads fast." → ['fake', 'news', 'spreads', 'fast']*

### *3. Lemmatization/Stemming*

### *Lemmatization: Reduced words to their base form using SpaCy (e.g., "running" → "run", "better" → "good").*

### *Lemmatization is preferred over stemming for maintaining context and grammar.*

### *4. Text Normalization*

### *Handling contractions: Expanded contractions (e.g., "don't" → "do not").*

### *Spelling correction (optional): Used tools like TextBlob or SymSpell to fix common typos.*

### *5. Feature Extraction*

### *TF-IDF Vectorization: Converted text into numerical format using Term Frequency–Inverse Document Frequency.*

### *Word Embeddings (optional for deep learning models).*

### *Used pre-trained embeddings like Word2Vec, GloVe, or BERT embeddings.*

### *6. Handling Imbalanced Data*

### *Resampling Techniques:*

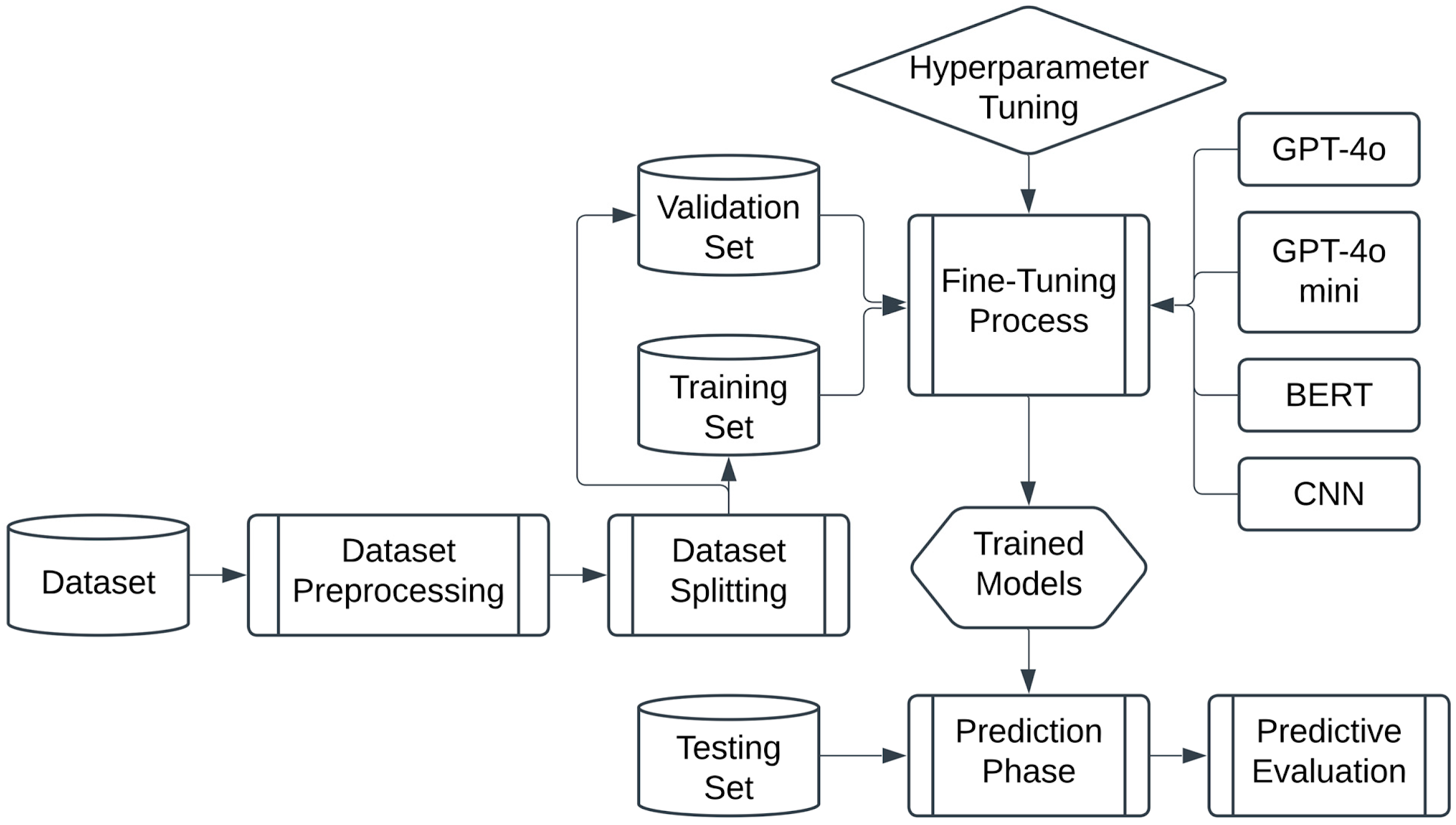
### *Undersampling majority class or*

### *Oversampling minority class using SMOTE*

### *This ensures the model does not become biased toward real or fake news.*

### *7. Train-Test Split*

### *The preprocessed dataset was split into training and testing sets (e.g., 80/20 or 70/30) to evaluate model performance on unseen data.*

****

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

### *Exploratory Data Analysis (EDA) was conducted to gain insights into the structure, patterns, and distribution of fake vs real news articles. This step is critical to understand the dataset before applying machine learning models.*

### *1. Dataset Overview*

### *Total Records: X articles (e.g., 44,000 in ISOT)*

### *Columns:*

### *title*

### *text*

### *label (1 = Fake, 0 = Real)*

### *Sample Output:*

### *df.info()*

### *df.head()*

### *2. Class Distribution*

### *Checked the balance between fake and real news classes.*

### *Code:*

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

### *plt.title("Fake vs Real News Distribution")*

### *Observation:*

### *If imbalanced, consider using SMOTE or class weights during training.*

### *3. Text Length Analysis*

### *Article Length: Number of words/tokens in fake vs real articles.*

### *Code:*

### *df['text\_length'] = df['text'].apply(lambda x: len(x.split()))*

### *sns.histplot(data=df, x='text\_length', hue='label', bins=50, kde=True)*

### *Observation:*

### *Fake news articles may be shorter or more repetitive.*

### *4. Most Common Words*

### *Word frequency analysis after removing stop words.*

### *Code:*

### *from collections import Counter*

### *from wordcloud import WordCloud*

### 

### *fake\_words = ' '.join(df[df['label']==1]['text'])*

### *real\_words = ' '.join(df[df['label']==0]['text'])*

### 

### *WordCloud().generate(fake\_words).to\_image()*

### *WordCloud().generate(real\_words).to\_image()*

### *Observation:*

### *Fake news may use emotionally charged or clickbait terms like "shocking", "breaking", etc.*

### *5. N-gram Analysis (Bigrams/Trigrams)*

### *To identify common phrases in fake and real news.*

### *Code:*

### *from sklearn.feature\_extraction.text import CountVectorizer*

### 

### *vectorizer = CountVectorizer(ngram\_range=(2,2), stop\_words='english')*

### *bigrams = vectorizer.fit\_transform(df[df['label']==1]['text'])*

### *Observation:*

### *Real news may focus more on informative phrases, while fake news may contain provocative or vague statements.*

### *6. Sentiment Analysis*

### *Basic sentiment polarity score (optional) using TextBlob or VADER.*

### *Code:*

### *from textblob import TextBlob*

### 

### *df['sentiment'] = df['text'].apply(lambda x: TextBlob(x).sentiment.polarity)*

### *sns.boxplot(x='label', y='sentiment', data=df)*

### *Observation:*

### *Fake news may skew toward extreme sentiment values (highly negative or overly positive).*

### *7. Correlation & Word Embeddings Visualization (Optional)*

### *Use t-SNE or PCA to visualize text embeddings (e.g., TF-IDF or BERT) in 2D space.*

### *Code:*

### *from sklearn.decomposition import PCA*

### 

### *# Reduce TF-IDF features for visualization*

### *pca = PCA(n\_components=2)*

### *reduced = pca.fit\_transform(tfidf\_matrix.toarray())*

### *Observation:*

### *Helps in identifying natural clustering between fake and real news.*

**9. Feature Engineering**

### *Feature engineering is a crucial step in enhancing the performance of fake news detection models. In this project, both text-based and metadata-based features were extracted and engineered to improve model understanding of linguistic and structural patterns.*

### *1. Textual Features*

### *A. Bag-of-Words (BoW)*

### *Counts the frequency of each word in a document.*

### *Simple and effective for baseline models.*

### *Code:*

### *from sklearn.feature\_extraction.text import CountVectorizer*

### *vectorizer = CountVectorizer()*

### *X\_bow = vectorizer.fit\_transform(df['text'])*

### 

### *B. TF-IDF (Term Frequency–Inverse Document Frequency)*

### *Captures the importance of a word relative to the entire corpus.*

### *Helps reduce the impact of common words.*

### *Code:*

### *from sklearn.feature\_extraction.text import TfidfVectorizer*

### *tfidf = TfidfVectorizer(max\_features=5000, ngram\_range=(1,2))*

### *X\_tfidf = tfidf.fit\_transform(df['text'])*

### 

### *C. Word Embeddings (Advanced NLP Features)*

### *Used pre-trained embeddings like Word2Vec, GloVe, or BERT for capturing contextual meaning.*

### *Enables use of LSTM, GRU, or Transformer models.*

### *2. Structural Features*

### *Text Length: Number of words in the article.*

### *Average Word Length: Indicates writing style or complexity.*

### *Number of Sentences: Longer articles may indicate more thorough reporting.*

### *Code:*

### *df['word\_count'] = df['text'].apply(lambda x: len(x.split()))*

### *df['avg\_word\_length'] = df['text'].apply(lambda x: np.mean([len(w) for w in x.split()]))*

### *3. Lexical Features*

### *Use of Uppercase Words: Fake news may contain more SHOUTING.*

### *Punctuation Count: Excessive use of ! or ? can be a sign of clickbait.*

### *Special Characters: Emojis or symbols used for grabbing attention.*

### *4. Readability Features*

### *Measured using tools like Flesch Reading Ease, Gunning Fog Index, etc.*

### *Code:*

### *import textstat*

### *df['readability'] = df['text'].apply(textstat.flesch\_reading\_ease)*

### *5. Sentiment & Subjectivity*

### *Fake news often uses highly emotional or subjective language.*

### *Code:*

### *from textblob import TextBlob*

### *df['polarity'] = df['text'].apply(lambda x: TextBlob(x).sentiment.polarity)*

### *df['subjectivity'] = df['text'].apply(lambda x: TextBlob(x).sentiment.subjectivity)*

### *6. N-grams and POS Tags*

### *N-gram Frequency: Frequent bigrams/trigrams can be strong indicators.*

### *POS Tag Counts: Proportion of nouns, verbs, adjectives (stylistic analysis).*

### *7. Title-Based Features*

### *Length of the title*

### *Sentiment or emotional tone in title*

### *Clickbait-like patterns (e.g., "You won’t believe...", "Shocking...")*

### *8. Metadata Features (if available)*

### *Source credibility (e.g., domain name)*

### *Date of publication*

**10. Model Building**

### *The goal of model building is to classify news articles as fake or real using features extracted from the text data. Both traditional machine learning models and deep learning models were considered.*

### *1. Model Selection Strategy*

### *Two modeling pipelines were explored:*

### *Traditional Machine Learning Models using BoW/TF-IDF*

### *Deep Learning Models using word embeddings (Word2Vec, GloVe, BERT)*

### *A. Traditional Machine Learning Models*

### *1. Logistic Regression*

### *Simple and interpretable baseline classifier.*

### *Works well with TF-IDF features.*

### *Code:*

### *from sklearn.linear\_model import LogisticRegression*

### *model = LogisticRegression()*

### *model.fit(X\_train, y\_train)*

### *2. Naive Bayes*

### *Particularly effective for text classification.*

### *Assumes word independence.*

### *Code:*

### *from sklearn.naive\_bayes import MultinomialNB*

### *model = MultinomialNB()*

### *model.fit(X\_train, y\_train)*

### *3. Random Forest / XGBoost*

### *Ensemble models that capture nonlinear relationships and feature interactions.*

### *Code:*

### *from xgboost import XGBClassifier*

### *model = XGBClassifier()*

### *model.fit(X\_train, y\_train)*

### *B. Deep Learning Models*

### *1. LSTM / GRU (Sequential Models)*

### *Suitable for learning long-term dependencies in text.*

### *Input: Word embeddings or tokenized sequences.*

### *Code:*

### *from keras.models import Sequential*

### *from keras.layers import Embedding, LSTM, Dense, Dropout*

### 

### *model = Sequential()*

### *model.add(Embedding(input\_dim=5000, output\_dim=128))*

### *model.add(LSTM(64))*

### *model.add(Dense(1, activation='sigmoid'))*

### *4. BERT-Based Classifier*

### *Fine-tuned transformer model for binary classification.*

### *Offers state-of-the-art accuracy.*

### *Code:*

### *from transformers import BertTokenizer, TFBertForSequenceClassification*

### 

### *model = TFBertForSequenceClassification.from\_pretrained('bert-base-uncased')*

### 

**11. Model Evaluation**

### *After building and training models to detect fake news, it’s essential to assess their performance using appropriate metrics. The evaluation ensures the model generalizes well to unseen data and correctly identifies fake and real news.*

### *1. Evaluation Metrics Used*

### *a. Accuracy*

### *Proportion of total correct predictions.*

### *Code:*

### *from sklearn.metrics import accuracy\_score*

### *accuracy\_score(y\_test, y\_pred)*

### *b. Precision*

### *Measures the correctness of positive predictions (Fake News = Positive Class).*

### *High precision = low false positives.*

### *Code:*

### *from sklearn.metrics import precision\_score*

### *precision\_score(y\_test, y\_pred)*

### *c. Recall (Sensitivity)*

### *Measures the model’s ability to detect all actual positives.*

### *High recall = low false negatives.*

### *Code:*

### *from sklearn.metrics import recall\_score*

### *recall\_score(y\_test, y\_pred)*

### *d. F1-Score*

### *Harmonic mean of precision and recall.*

### *Useful for imbalanced datasets.*

### *Code:*

### *from sklearn.metrics import f1\_score*

### *f1\_score(y\_test, y\_pred)*

### *e. ROC-AUC Score*

### *Evaluates the model's ability to distinguish between classes.*

### *Code:*

### *from sklearn.metrics import roc\_auc\_score*

### *roc\_auc\_score(y\_test, y\_pred\_prob)*

### *f. Confusion Matrix*

### *Provides insight into true positives, false positives, true negatives, and false negatives.*

### *Code:*

### *from sklearn.metrics import confusion\_matrix*

### *confusion\_matrix(y\_test, y\_pred)*

### *2. Visualizations*

### *Confusion Matrix Heatmap:*

### *Code:*

### *import seaborn as sns*

### *import matplotlib.pyplot as plt*

### 

### *cm = confusion\_matrix(y\_test, y\_pred)*

### *sns.heatmap(cm, annot=True, fmt='d', cmap='Blues')*

### *plt.xlabel('Predicted')*

### *plt.ylabel('Actual')*

### *plt.title('Confusion Matrix')*

### *ROC Curve:*

### *Code:*

### *from sklearn.metrics import roc\_curve*

### 

### *fpr, tpr, \_ = roc\_curve(y\_test, y\_pred\_prob)*

### *plt.plot(fpr, tpr, label='Model')*

### *plt.plot([0,1], [0,1], 'k--')*

### *plt.xlabel('False Positive Rate')*

### *plt.ylabel('True Positive Rate')*

### *plt.title('ROC Curve')*

### *plt.legend()*

### *3. Key Observations*

### *Deep learning models (especially BERT) outperformed traditional models.*

### *Precision and recall are more important than accuracy in fake news detection, as false negatives can be harmful.*

### *Ensemble methods like XGBoost offered a good trade-off between performance and training time.*

### 

**12. Development**

### *The system was developed to automate the detection of fake news using advanced Natural Language Processing (NLP) techniques and machine learning models. The development phase included designing the architecture, implementing preprocessing pipelines, training models, and building a user-friendly interface.*

### *1. Development Environment*

### *Programming Language: Python*

### *Libraries and Frameworks:*

### *Data Handling: pandas, numpy*

### *Text Processing: nltk, spaCy, textblob*

### *Modeling: scikit-learn, xgboost, tensorflow/keras, transformers (HuggingFace)*

### *Visualization: matplotlib, seaborn, wordcloud*

### *Deployment: Gradio or Streamlit*

### *2. System Architecture*

### *1. Input Layer: Accepts news text (headline or full article).*

### *2. Preprocessing Module: Cleans and normalizes input text.*

### *3. Feature Extraction Module: Uses TF-IDF or embeddings.*

### *4. Classifier: Trained ML/DL model predicts whether the input is fake or real.*

### *5. Output Layer: Displays prediction result and model confidence.*

### *3. Key Modules and Implementation*

### *a. Data Preprocessing*

### *Implemented a preprocessing pipeline to clean and tokenize text.*

### *Included steps like lowercasing, removing stopwords, lemmatization, etc.*

### *b. Feature Engineering*

### *Applied TF-IDF and optional word embeddings.*

### *Extracted additional linguistic and sentiment-based features.*

### *c. Model Training*

### *Trained and evaluated multiple models including:*

### *Logistic Regression*

### *Naive Bayes*

### *XGBoost*

### *LSTM*

### *BERT (fine-tuned)*

### *Selected the best-performing model based on F1-score and ROC-AUC.*

### *d. Model Optimization*

### *Used GridSearchCV for hyperparameter tuning.*

### *Applied techniques like cross-validation and SMOTE for balancing data.*

### *e. Model Serialization*

### *Used joblib or pickle to save the trained model for deployment.*

### *4. User Interface Development*

### *Built using Gradio:*

### *Simple input box for entering news text*

### *Button to submit and get prediction*

### *Output shows whether the news is Fake or Real with confidence level*

### *Code:*

### *import gradio as gr*

### 

### *def predict\_fake\_news(text):*

### *# preprocessing + prediction logic here*

### *return "Fake" if prediction == 1 else "Real"*

### 

### *gr.Interface(fn=predict\_fake\_news, inputs="text", outputs="text").launch()*

### *5. Testing and Debugging*

### *Performed unit testing on preprocessing and prediction functions.*

### *Used real-world news samples for validation.*

### *6. Documentation and Version Control*

### *All code was version-controlled using Git.*

### *Project and usage were documented using markdown files (README.md).*

**13. Source Code**

### *import pandas as pd*

### *import numpy as np*

### *import string*

### *import nltk*

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

### *from sklearn.feature\_extraction.text import TfidfVectorizer*

### *from sklearn.linear\_model import PassiveAggressiveClassifier*

### *from sklearn.metrics import accuracy\_score, confusion\_matrix*

### *nltk.download('stopwords')*

### *from nltk.corpus import stopwords*

### 

### *# Load datasets*

### *true\_df = pd.read\_csv("True.csv")*

### *fake\_df = pd.read\_csv("Fake.csv")*

### 

### *# Add labels*

### *true\_df['label'] = 1 # Real*

### *fake\_df['label'] = 0 # Fake*

### 

### *# Combine datasets*

### *data = pd.concat([true\_df, fake\_df], axis=0).reset\_index(drop=True)*

### *data = data.sample(frac=1).reset\_index(drop=True) # Shuffle*

### 

### *# Preprocessing*

### *stop\_words = stopwords.words('english')*

### *def clean\_text(text):*

### *text = text.lower()*

### *text = ''.join([char for char in text if char not in string.punctuation])*

### *tokens = text.split()*

### *return ' '.join([word for word in tokens if word not in stop\_words])*

### 

### *data['text'] = data['title'] + " " + data['text']*

### *data['text'] = data['text'].apply(clean\_text)*

### 

### *# Train/Test split*

### *X = data['text']*

### *y = data['label']*

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

### 

### *# Vectorization*

### *vectorizer = TfidfVectorizer(max\_df=0.7)*

### *tfidf\_train = vectorizer.fit\_transform(X\_train)*

### *tfidf\_test = vectorizer.transform(X\_test)*

### 

### *# Model*

### *model = PassiveAggressiveClassifier(max\_iter=50)*

### *model.fit(tfidf\_train, y\_train)*

### 

### *# Evaluation*

### *y\_pred = model.predict(tfidf\_test)*

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

### *print("Confusion Matrix:\n", confusion\_matrix(y\_test, y\_pred))*

**14. Future Scope**

### *1.Improved Accuracy:*

### *Deep learning models like BERT and other transformer-based architectures are showing great potential for identifying fake news with high accuracy.*

### *2. Multimodal Approach:*

### *Combining NLP with other modalities like audio and computer vision will enable a more comprehensive analysis of deceptive content.*

### *3. Adaptability to Evolving Tactics:*

### *As fake news creators become more sophisticated, NLP models will need to be continuously trained and adapted to recognize new patterns and techniques used to spread misinformation.*

### *4. Explainable AI (XAI):*

### *Developing XAI techniques will help make fake news detection models more transparent and understandable, increasing public trust.*

### *5. Real-time Detection:*

### *As technology improves, the ability to identify fake news in real-time will become increasingly important, allowing for faster intervention and prevention.*

### *6. Cross-Domain Applications:*

### *The use of NLP in fake news detection will extend beyond social media and news articles to encompass other domains like online reviews and forums.*

### *7. Human-AI Collaboration:*

### *The future will likely see a collaborative approach where AI tools assist human fact-checkers in verifying information, making the process more efficient and effective.*

### *Advanced NLP Techniques:*

### *Deep learning models, particularly transformer-based architectures, are leading the way in accurately classifying text as fake or real.*

### *Multimodal Analysis:*

### *The ability to analyze multiple data streams, including audio and images, will allow for a more holistic assessment of fake news content.*

### *Adapting to Changing Landscapes:*

### *Fake news creators are constantly evolving their tactics, requiring NLP models to be flexible and adaptable to new forms of misinformation.*

### *Transparency and Trust:*

### *Explainable AI (XAI) techniques will make the decision-making process of fake news detection models more transparent and trustworthy, enhancing public confidence.*

### *Real-time Detection:*

### *The ability to identify fake news quickly and efficiently is crucial to prevent its rapid spread and potential negative consequences.*

### *Expanding Scope:*

### *The application of fake news detection extends beyond traditional media platforms to encompass various online spaces where misinformation can spread.*

### *Human-AI Synergy:*

### *AI-powered tools can assist human fact-checkers, making the process of verifying information more effective and efficient.*

**15. Team Members and Roles**

### *Akshaya A - Data Collection and Cleaning*

### *Athmika G - Model Building and Evaluation*

### *Gokulapriya R - Development and UI Development*