## **Fake News Detection using ML and Flask Frontend**

## **1. Introduction**

* **1.1 Project Overview**
  + This project develops a machine learning model for accurate fake news detection, important due to the spread of misinformation.
* **1.2 Goals and Objectives**
  + Goals: data handling, model training/evaluation, feature identification, and user-friendly tool development.
  + Objectives: target accuracy, scalability/efficiency, and clear, interpretable results.

### **2. Literature Review**

* **2.1 Existing Approaches**
  + Current methods: manual fact-checking, rule-based systems, machine/deep learning, and social network analysis.
  + Each has advantages and disadvantages.
* **2.2 Related Work**
  + This project will review relevant research, summarizing methodologies, findings, contributions, and identifying gaps.

### **3. Methodology**

* **3.1 Data Collection**
  + Data sources: public datasets, news APIs, web scraping, and fact-checking websites.
  + Data is selected for quality, relevance, and diversity, with a specified size and distribution.
* **3.2 Data Preprocessing**
  + Preprocessing: text cleaning, tokenization, stop word removal, stemming/lemmatization, and handling imbalanced data.
* **3.3 Feature Extraction**
  + Techniques: Bag-of-Words, TF-IDF, word embeddings, linguistic features, and metadata.
* **3.4 Model Selection and Training**
  + Models considered: Naive Bayes, Logistic Regression, SVM, Decision Trees, Random Forests, Gradient Boosting, and deep learning.
  + The final model(s) will be chosen based on performance, interpretability, and computational cost.
  + The training process: data splitting, hyperparameter tuning, and model optimization/regularization.
* **3.5 Model Evaluation**
  + Evaluation metrics: accuracy, precision, recall, F1-score, AUC-ROC, and confusion matrix.
  + Evaluation will discuss performance on training/validation/test sets, comparison with other approaches, and model strengths/weaknesses.

### **4. Implementation**

* **4.1 Software and Hardware Requirements**
  + Software: Python, machine learning libraries (scikit-learn, TensorFlow, PyTorch), data processing (Pandas, NumPy), visualization (Matplotlib, Seaborn), and development environments (Jupyter Notebook, IDEs). Flask is used for the frontend.
  + Hardware: processing power, memory, and storage.
* **4.2 System Architecture**
  + The system architecture includes data flow, processing pipeline, software components, and UI design.
  + The application consists of a backend (Python/Flask) and a frontend (HTML/JavaScript) for user interaction.
  + Diagrams or flowcharts will illustrate system structure and functionality.
* **4.3 Code Implementation**
  + The project includes:
    - Backend (Flask): Handles model loading, prediction, and API endpoints.
    - Frontend (HTML/JavaScript): Provides a UI for submitting text and displaying results.
  + Code examples will show data handling, feature extraction, model work, and UI development.
  + Links to the complete source code repository will be included, if available.

### **5. Flask Frontend**

A simple web interface is built using Flask for user interaction.

* **HTML (index.html)**  
  <!DOCTYPE html>  
  <html lang="en">  
  <head>  
   <meta charset="UTF-8">  
   <meta name="viewport" content="width=device-width, initial-scale=1.0">  
   <title>Fake News Detector</title>  
   <script src="https://unpkg.com/@tailwindcss/browser@4"></script>  
   <link href="https://fonts.googleapis.com/css2?family=Inter:wght@400;600;700&display=swap" rel="stylesheet">  
   <style>  
   body {  
   font-family: 'Inter', sans-serif;  
   }  
   </style>  
  </head>  
  <body class="bg-gray-100 flex items-center justify-center min-h-screen">  
   <div class="bg-white rounded-lg shadow-md p-8 w-full max-w-md">  
   <h1 class="text-2xl font-semibold text-blue-600 mb-4 text-center">Fake News Detector</h1>  
   <p class="text-gray-700 mb-4 text-center">Enter the news article text below to check if it's fake or real.</p>  
    
   <form id="news-form" class="space-y-4">  
   <div>  
   <label for="news-text" class="block text-gray-700 text-sm font-bold mb-2">News Article Text:</label>  
   <textarea id="news-text" name="news-text" placeholder="Enter the news article text here..." required class="shadow appearance-none border rounded w-full py-2 px-3 text-gray-700 leading-tight focus:outline-none focus:shadow-outline h-32 resize-y"></textarea>  
   </div>  
   <button type="submit" class="bg-blue-500 hover:bg-blue-700 text-white font-bold py-2 px-4 rounded focus:outline-none focus:shadow-outline w-full">  
   Check News  
   </button>  
   </form>  
    
   <div id="result-container" class="mt-6" style="display: none;">  
   <div id="result-box" class="border rounded p-4 text-center font-medium">  
   <p id="result-text" class="text-lg"></p>  
   </div>  
   </div>  
   </div>  
    
   <script>  
   document.getElementById('news-form').addEventListener('submit', function(event) {  
   event.preventDefault();  
    
   const newsText = document.getElementById('news-text').value;  
    
   fetch('/predict', { // Corrected route to '/predict'  
   method: 'POST',  
   headers: {  
   'Content-Type': 'application/json'  
   },  
   body: JSON.stringify({ news\_text: newsText })  
   })  
   .then(response => response.json())  
   .then(data => {  
   const resultContainer = document.getElementById('result-container');  
   const resultText = document.getElementById('result-text');  
   const resultBox = document.getElementById('result-box'); // Get the result box  
    
   resultText.textContent = data.prediction;  
   resultContainer.style.display = 'block';  
    
   if (data.prediction === 'Fake') {  
   resultBox.classList.remove('border-green-500', 'bg-green-100', 'text-green-800');  
   resultBox.classList.add('border-red-500', 'bg-red-100', 'text-red-800');  
   } else {  
   resultBox.classList.remove('border-red-500', 'bg-green-100', 'text-green-800');  
   resultBox.classList.add('border-green-500', 'bg-green-100', 'text-green-800');  
   }  
   })  
   .catch(error => {  
   console.error('Error:', error);  
   const resultContainer = document.getElementById('result-container');  
   const resultText = document.getElementById('result-text');  
   resultText.textContent = 'An error occurred. Please try again.';  
   resultContainer.style.display = 'block';  
   resultBox.classList.remove('border-green-500', 'bg-green-100', 'text-green-800');  
   resultBox.classList.add('border-red-500', 'bg-red-100', 'text-red-800');  
   });  
   });  
   </script>  
  </body>  
  </html>
* **Python (Flask app.py)**  
  from flask import Flask, render\_template, request, jsonify  
  import pickle  
  import numpy as np  
  from sklearn.feature\_extraction.text import TfidfVectorizer  
  from sklearn.linear\_model import PassiveAggressiveClassifier  
    
  app = Flask(\_\_name\_\_)  
    
  # Load the trained model and vectorizer  
  with open('model.pkl', 'rb') as model\_file:  
   model = pickle.load(model\_file)  
  with open('vectorizer.pkl', 'rb') as vectorizer\_file:  
   vectorizer = pickle.load(vectorizer\_file)  
    
  @app.route('/')  
  def index():  
   return render\_template('index.html')  
    
  @app.route('/predict', methods=['POST'])  
  def predict():  
   try:  
   data = request.get\_json() # Use get\_json() to get JSON data  
   news\_text = data['news\_text']  
   print(f"Received news text: {news\_text}") # Debugging line  
    
   # Vectorize the input text  
   vectorized\_text = vectorizer.transform([news\_text])  
    
   # Make the prediction  
   prediction = model.predict(vectorized\_text)[0]  
    
   # Return the result as JSON  
   result = 'Fake' if prediction == 0 else 'Real'  
   return jsonify({'prediction': result})  
   except Exception as e:  
   print(f"Error: {e}")  
   return jsonify({'error': str(e)}), 500  
    
  if \_\_name\_\_ == '\_\_main\_\_':  
   app.run(debug=True)

### **6. Results and Discussion**

* **5.1 Experimental Results**
  + Detailed results will be presented in tables, graphs, and charts.
  + The model's performance will be analyzed across scenarios and datasets.
  + The impact of feature choice, model selection, hyperparameters, and data size/quality will be discussed.
* **5.2 Discussion**
  + The results will be interpreted, and conclusions drawn, explaining the findings' significance in relation to the project's goals.
  + Limitations, potential improvements, and ethical implications will be discussed.

### **7. Conclusion**

* **6.1 Summary of Findings**
  + The project's main findings, achievements, and contributions will be summarized.
  + The project's impact and potential applications will be reiterated.