Department of Mathematical Sciences and Computer Applications, Bundelkhand University, Jhansi

**A PROJECT REPORT ON FAKE NEWS DETECTION USING MACHINE LEARNING**

**SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENT FOR THE AWARD OF THE DEGREE BACHELOR OF SCIENCE(Honors)**

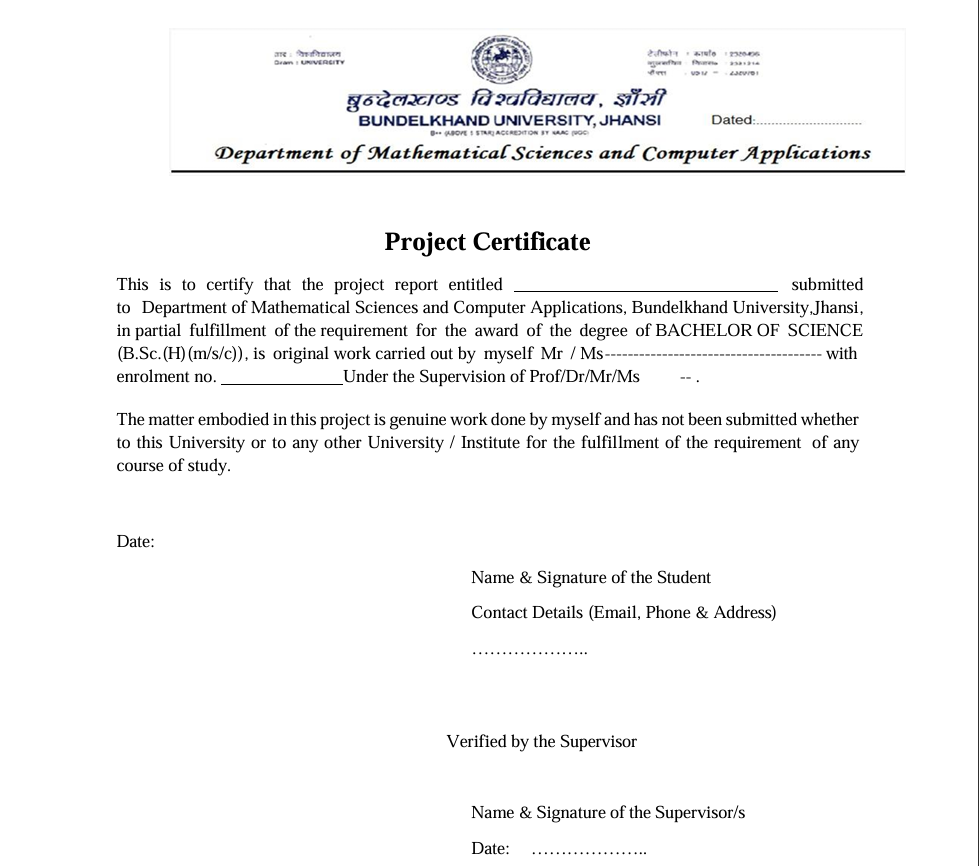


**SESSION :2024-2025**

**Submitted to Submitted by**

**Dr Puneet Matapurkar Somya Raikwar**

**Assistant Professor 221181134120**



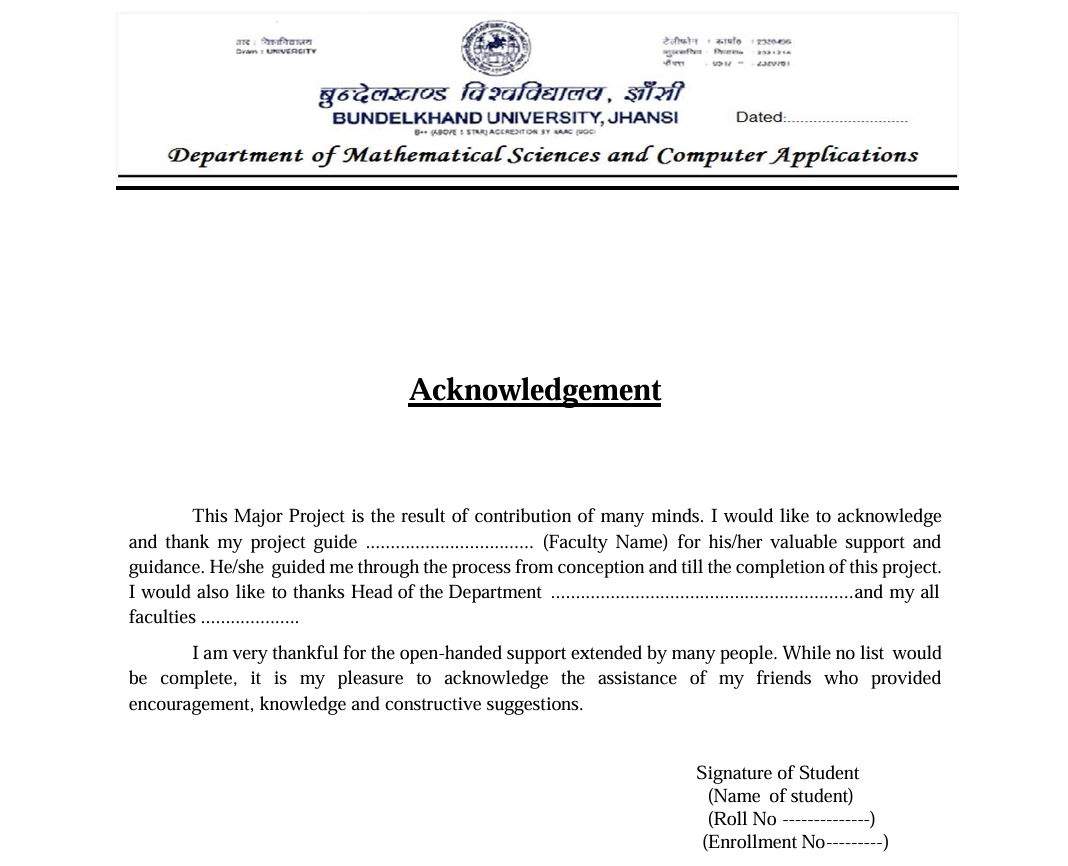
**i**



**ii**



**iii**



**iv**

**TABLE OF CONTENTS**

| **S. No.** | **Section Title** | **Page No.** |
| --- | --- | --- |
| **1.** | **Project Certificate** | **i** |
| **2.** | **Self-Certificate** | **ii** |
| **3.** | **Certificate from Project Guide** | **iii** |
| **4.** | **Acknowledgement** | **iv** |
| **5.** | **Table of Contents / Index** | **v** |
| **6.** | **Introduction and Objectives of the Project** | **1** |
| **7.** | **System Analysis** | **3** |
| **8.** | **Feasibility Study** | **5** |
| **9.** | **Software and Hardware Requirements** | **7** |
| **10.** | **System Design** | **8** |
|  | **• Data Flow Diagrams (DFD)** |  |
|  | **• Entity Relationship Diagram (ERD)** |  |
|  | **• Flowcharts and Module Description** |  |
| **11.** | **Coding (Key Implementation Snippets)** | **13** |
| **12.** | **Validation Checks** | **16** |
| **13.** | **Implementation and Maintenance** | **17** |
| **14.** | **Testing and Test Results** | **18** |
|  | **• Test Cases** |  |
|  | **• Expected vs Actual Results** |  |
|  | **• Error Reports** |  |
| **15.** | **Screenshots of the Project** | **21** |
| **17.** | **Conclusion** | **26** |
| **18.** | **Future Scope and Enhancements** | **27** |
| **19.** | **Bibliography / References** | **28** |
| **20.** | **Appendices (if any)** | **29** |
| **21.** | **CD Submission (Soft copy contents: Code, Report, Output)** | **30** |

**6. Introduction and Objectives of the Project**

**6.1 Introduction**

In today’s digital era, the rapid spread of information through online platforms has made news more accessible than ever. However, this ease of dissemination has also led to the proliferation of *fake news* — misleading or false information that can influence public opinion, create panic, or spread misinformation at scale. Combating this rising threat is a significant challenge, especially when manual verification is not feasible at a large scale.

To address this issue, the present project aims to develop an intelligent system capable of automatically detecting fake news using Machine Learning (ML) and Natural Language Processing (NLP) techniques. This system analyzes the textual content of news articles and classifies them as either *Real* or *Fake*. The model is trained on a labeled dataset consisting of both genuine and fabricated news, allowing it to learn patterns and make predictions on unseen data.

By automating the detection of fake news, this project contributes to the effort of maintaining the integrity of information shared on digital platforms. It also provides a practical demonstration of applying data science skills to solve real-world societal problems.

**6.2 Objectives**

The primary objectives of this project are:

1. To develop a machine learning-based classification model that can distinguish between fake and real news articles using textual features.
2. To preprocess and clean the news dataset using natural language processing techniques such as tokenization, stemming, and stopword removal.
3. To apply and evaluate multiple machine learning algorithms, including Logistic Regression and Naive Bayes, and select the best-performing model.
4. To analyze and visualize the dataset for understanding word frequency patterns and class distributions.
5. To implement a user-friendly interface using Streamlit that allows users to input news text and receive instant classification feedback.
6. To ensure proper testing and validation of the system, including performance metrics such as accuracy, precision, recall, and F1-score.
7. To provide recommendations for future enhancements, such as multilingual support or integration with browser extensions.

**7 System Analysis**

**7.1 Problem Definition**

With the rise of digital journalism and user-generated content, the spread of fake news has become a serious issue affecting public trust and societal well-being. Traditional methods of news verification are time-consuming and inefficient in today's fast-paced digital world. Hence, there is a pressing need for an automated system that can analyze news content and determine its authenticity.

This project addresses the problem by implementing a machine learning model capable of detecting fake news using natural language processing techniques. The system aims to classify news articles as *Real* or *Fake* based on textual features and learned patterns from a labeled dataset.

**7.2 System Requirements Analysis**

**Functional Requirements:**

* Accept input in the form of a news article or headline.
* Process and clean the input text using NLP techniques.
* Convert textual data into numerical format using TF-IDF vectorization.
* Predict whether the news is fake or real using a trained ML model.
* Display the result to the user via a web interface.

**Non-Functional Requirements:**

* The system should provide quick and accurate predictions.
* The user interface should be intuitive and responsive.
* The system should be scalable to include large datasets in the future.

**7.3 Data Collection**

The dataset used for this project is sourced from Kaggle, comprising two files:

* fake.csv – containing fake news articles.
* true.csv – containing genuine news articles.

Each record contains fields such as title, text, subject, and date. Labels were assigned (0 for Fake, 1 for Real) and merged to create a comprehensive dataset.

**7.4 Data Preprocessing Techniques**

To prepare the data for model training, the following preprocessing steps were applied:

* Removal of special characters and punctuation.
* Conversion to lowercase.
* Stopword removal using NLTK’s English stopword list.
* Word stemming using Porter Stemmer.
* Tokenization and transformation using TF-IDF vectorization.

**7.5 System Modules**

| **Module Name** | **Description** |
| --- | --- |
| Data Preprocessing | Cleans raw news text for training and prediction. |
| Feature Extraction | Converts cleaned text into TF-IDF feature vectors. |
| Model Training | Applies ML algorithms like Logistic Regression and Naive Bayes. |
| Evaluation Module | Calculates accuracy, confusion matrix, and classification report. |
| UI Module | A Streamlit-based interface to accept user input and show predictions. |

**8 Feasibility Study**

Before initiating the development of any software project, a detailed feasibility study is essential to assess the practicality of the proposed solution. The feasibility study evaluates various aspects including technical, operational, economic, and time-related factors to determine whether the project can be successfully developed and deployed.

**8.1 Technical Feasibility**

This project is technically feasible, as it is built using widely adopted technologies and tools that are both open-source and well-supported. The primary technology stack includes:

* Programming Language: Python
* Libraries and Frameworks: scikit-learn, pandas, NumPy, NLTK, Streamlit
* Development Environment: Jupyter Notebook
* Machine Learning Algorithms: Logistic Regression, Naive Bayes

All tools used are freely available, easy to install, and compatible with standard operating systems. No specialized hardware or licensed software is required, making it technically viable for implementation in a college lab environment or a personal machine.

**8.2 Operational Feasibility**

The proposed system is designed with end-user simplicity and functionality in mind. The web-based interface built using Streamlit ensures that users without technical backgrounds can easily interact with the system by simply pasting or typing the news content and receiving a result with a single click.

The application meets the operational objectives:

* Ease of use for students, faculty, or anyone reading news online.
* Fast, real-time prediction of news authenticity.
* Clear display of results (Real or Fake) with visual cues.

The system will be deployed locally or on a web browser, ensuring smooth operation without requiring complex setups.

**8.3 Economic Feasibility**

This project is economically feasible for academic implementation. No licensing costs are involved because:

* All software used is free and open-source.
* Development is done on a personal computer or university lab machines.
* No physical hardware or external server hosting is required for basic deployment.

Thus, the overall development cost is negligible, making it an economically sound project for students.

**8.4 Time Feasibility**

The development timeline for the project was planned and executed over a period of 4–6 weeks, including:

* Data collection and preprocessing
* Model training and evaluation
* UI design and integration
* Documentation and testing

Since the scope of the project is well-defined and tools used are familiar to the developer, it was completed well within the allotted academic project timeline.

**Conclusion**

Based on the analysis of technical, operational, economic, and time-related factors, the Fake News Detection system is fully feasible for development and deployment as a major academic project. It serves as a valuable real-world application of machine learning and natural language processing in combating misinformation.

**9. Software and Hardware Requirements**

This section outlines the necessary software and hardware specifications required to develop, run, and test the Fake News Detection system effectively. Since this is an academic project, the requirements are optimized for execution on a standard personal computer or university lab setup.

**9.1 Software Requirements**

| **Component** | **Specification / Version** |
| --- | --- |
| Operating System | Windows 11 |
| Programming Language | Python 3.12 |
| IDE / Code Editor | Jupyter Notebook |
| Libraries and Frameworks | scikit-learn, pandas, numpy, nltk, joblib |
| Web Framework (for UI) | Streamlit |
| Data Visualization Tools | matplotlib, seaborn |
| Browser | Google Chrome, Microsoft Bing |
| Document Editor | MS Word |

**Python Library Versions Used:**

scikit-learn 1.1+

pandas 1.5+

numpy 1.22+

nltk 3.8+

streamlit 1.25+

joblib 1.2+

matplotlib 3.5+ (optional for graphs)

**9.2 Hardware Requirements**

| **Component** | **Minimum Requirement** |
| --- | --- |
| Processor | Intel Core i3 or equivalent |
| RAM | 4 GB minimum (8 GB recommended) |
| Hard Disk | 5 GB of free space |
| Display | 13” or larger (recommended for UI) |
| Keyboard and Mouse | Standard input devices |
| Internet Connection | Required for downloading libraries |

**Notes:**

* The system is lightweight and can run efficiently on basic hardware without GPU acceleration.
* The use of Streamlit ensures that the application runs via a web browser, making deployment and interaction easier without the need for any server infrastructure.

**10. System Design**

System design involves defining the overall structure and behavior of the system. It includes identifying the key components, their interactions, and the flow of data throughout the system. For the Fake News Detection system, the design focuses on both logical and user-facing elements, involving data flow, user interaction, and module operations.

**10.1 System Architecture Overview**

The system is designed in a modular architecture with the following components:

1. Data Preprocessing Module
2. Feature Extraction Module
3. Model Training and Prediction Module
4. User Interface Module
5. Evaluation and Reporting Module

Each of these modules performs specific tasks and passes output to the next stage of the pipeline.

**10.2 Data Flow Diagram (DFD)**

**Level 0 DFD:**

* Input: News article text
* Process: Preprocessing → Vectorization → Classification
* Output: Label (Real or Fake)

[User] ---> [Input News Text] ---> [Preprocessing] ---> [ML Model] ---> [Result Display]

**Level 1 DFD:**

* Describes detailed sub-processes:
  + Text Cleaning (removal of special chars, stopwords, stemming)
  + TF-IDF Vectorization
  + ML Prediction (Logistic Regression/Naive Bayes)
  + Result shown in UI

**10.3 Entity-Relationship Diagram (ERD)**

(Simplified representation, since the project is not database-heavy)

| **Entity** | **Attributes** |
| --- | --- |
| NewsArticle | id, title, text, label |
| Model | model\_name, version |
| Prediction | id, input\_text, output |

Note: If extended with user data or logs, a database schema can be added.

**10.4 Flowchart of the System Logic**

**┌────────────────────┐**

**│ Start Application │**

**└────────┬───────────┘**

**│**

**┌─────────▼──────────┐**

**│ Enter News Content │**

**└─────────┬──────────┘**

**│**

**┌─────────────▼──────────────┐**

**│ Preprocess the Input Text │**

**└─────────────┬──────────────┘**

**│**

**┌───────────▼────────────┐**

**│ TF-IDF Vectorization │**

**└───────────┬────────────┘**

**│**

**┌─────────▼──────────┐**

**│ ML Model Predict │**

**└─────────┬──────────┘**

**│**

**┌─────────────▼──────────────┐**

**│ Display Result (Real/Fake) │**

**└─────────────┬──────────────┘**

**│**

**┌─────▼─────┐**

**│ End │**

**└───────────┘**

**10.5 Module Description**

| **Module** | **Functionality** |
| --- | --- |
| Input Module | Accepts raw news article or headline text from the user |
| Preprocessing Module | Cleans the text: lowercasing, stopword removal, stemming |
| Feature Extraction | Applies TF-IDF to convert text into numerical vectors |
| Classification | Uses trained ML model (Logistic Regression/Naive Bayes) to predict output |
| UI Module | Streamlit-based interface for interaction |
| Output Display | Shows whether the input is Real or Fake along with animations and visual cues |

**11. Coding**

The coding phase involves implementing the logic for preprocessing the news data, training the machine learning model, and creating a user interface to interact with the system. The system is coded in Python, using Jupyter Notebook for development and Streamlit for deployment.

**11.1 Code Overview**

The entire codebase is divided into the following components:

| **Component** | **Description** |
| --- | --- |
| Data Preprocessing | Cleaning and preparing raw news text for analysis |
| Feature Extraction | Applying TF-IDF vectorization on cleaned text |
| Model Training | Using scikit-learn ML algorithms like Logistic Regression |
| Model Evaluation | Measuring accuracy, precision, recall, confusion matrix |
| Model Saving | Saving model and vectorizer using joblib |
| Streamlit Application | A user interface for fake news detection |

**1. Data Preprocessing**

import nltk

from nltk.corpus import stopwords

from nltk.stem import PorterStemmer

import re

nltk.download('stopwords')

stop\_words = set(stopwords.words('english'))

stemmer = PorterStemmer()

def clean\_text(text):

text = re.sub(r'[^a-zA-Z]', ' ', text)

text = text.lower().split()

text = [stemmer.stem(word) for word in text if word not in stop\_words]

return ' '.join(text)

df['cleaned\_text'] = df['text'].apply(clean\_text)

**2. Feature Extraction (TF-IDF)**

from sklearn.feature\_extraction.text import TfidfVectorizer

tfidf = TfidfVectorizer(max\_features=5000)

X = tfidf.fit\_transform(df['cleaned\_text']).toarray()

y = df['label']

**3. Model Training and Testing**

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

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

from sklearn.linear\_model import LogisticRegression

from sklearn.naive\_bayes import MultinomialNB

lr = LogisticRegression()

nb = MultinomialNB()

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

nb.fit(X\_train, y\_train)

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

def evaluate\_model(model):

y\_pred = model.predict(X\_test)

print(confusion\_matrix(y\_test, y\_pred))

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

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

evaluate\_model(lr)

evaluate\_model(nb)

**4. Model Saving**

import os

# Create the folder if it doesn't exist

os.makedirs("models", exist\_ok=True)

# Now save the model

import joblib

joblib.dump(lr, "models/logistic\_model.pkl")

joblib.dump(tfidf, "models/tfidf\_vectorizer.pkl")

**5. Streamlit App (app.py)**

import streamlit as st

import joblib

import re

import nltk

from nltk.corpus import stopwords

from nltk.stem import PorterStemmer

from PIL import Image

nltk.download('stopwords')

stop\_words = set(stopwords.words('english'))

stemmer = PorterStemmer()

# Load model and vectorizer

model = joblib.load('models/logistic\_model.pkl')

vectorizer = joblib.load('models/tfidf\_vectorizer.pkl')

# Function: Preprocess Input

def preprocess(text):

text = re.sub(r'[^a-zA-Z]', ' ', text)

text = text.lower().split()

text = [stemmer.stem(word) for word in text if word not in stop\_words]

return ' '.join(text)

# Page Configuration

st.set\_page\_config(page\_title="Fake News Detector", page\_icon="🧠", layout="centered")

# Add Banner or Logo

st.image("C:/Users/SOMYA/Downloads/ChatGPT Image May 10, 2025, 05\_41\_23 PM.png", use\_container\_width=True)

# Custom CSS Styling

st.markdown("""

<style>

.main {

background-color: #f0f2f6;

padding: 2rem;

border-radius: 10px;

box-shadow: 0 0 10px rgba(0,0,0,0.1);

}

.stButton button {

background-color: #0072B2;

color: white;

font-size: 16px;

padding: 10px 24px;

border-radius: 5px;

margin-top: 20px;

}

.stButton button:hover {

background-color: #005f8a;

}

</style>

""", unsafe\_allow\_html=True)

# App Title

st.markdown("<h1 style='text-align: center;'>🧠 Fake News Detector</h1>", unsafe\_allow\_html=True)

st.markdown("<h4 style='text-align: center; color: gray;'>Classify news as Real or Fake using NLP and ML</h4>", unsafe\_allow\_html=True)

# Text Input

st.markdown("### 📌 Paste or type the news content below:")

user\_input = st.text\_area("News Content", height=200)

# Predict

if st.button("🔍 Detect"):

if user\_input.strip() == "":

st.warning("⚠️ Please enter some text to analyze.")

else:

cleaned\_input = preprocess(user\_input)

vectorized\_input = vectorizer.transform([cleaned\_input])

prediction = model.predict(vectorized\_input)[0]

st.markdown("---")

if prediction == 1:

st.success("✅ This appears to be \*\*REAL news\*\*.")

st.balloons()

else:

st.error("❌ This appears to be \*\*FAKE news\*\*.")

st.snow()

# Footer

st.markdown("""

<hr>

<small style='color: gray;'>📚 Developed by Somya • A College Major Project</small>

""", unsafe\_allow\_html=True)

**11.3 Remarks**

* The project code is modular, well-commented, and follows best practices.
* Model and vectorizer are persisted using joblib for reusability.
* Streamlit makes the app deployable without needing web frameworks like Flask or Django.
* Code files are included in the attached soft copy (CD) in both .py and .ipynb formats.

**12. Validation Checks**

Validation checks are an essential part of system design and implementation to ensure that the inputs provided by users are logical, complete, and consistent with expected formats. These checks help prevent invalid data from entering the system, reducing potential errors and improving the reliability of the model’s output.

In the Fake News Detection system, several validation mechanisms are implemented at different levels—particularly in the data preprocessing stage and user interface.

**12.1 Input Validation in Preprocessing Module**

Before training the machine learning model, the dataset undergoes rigorous preprocessing to remove noise and invalid data. The following checks are performed:

| **Validation Check** | **Description** |
| --- | --- |
| Non-empty Check | Ensures that the text field in the dataset is not empty. |
| Text Length Filter | Articles with extremely short text are discarded as insufficient information. |
| Stopword Removal | Filters out commonly used but uninformative words (e.g., "the", "is", "on"). |
| Special Characters Removal | All non-alphabetic characters are removed to retain meaningful content. |
| Lowercasing | Ensures uniformity by converting all text to lowercase before tokenization. |

**12.2 Runtime Validation in Streamlit App**

In the user interface, validation checks are embedded to prevent misuse or incomplete inputs. These include:

| **Validation Point** | **Check** |
| --- | --- |
| Empty Input Field | The system checks if the user has entered text; otherwise, a warning is shown. |
| Minimum Text Length | Input text should be of reasonable length to enable meaningful classification. |
| Prediction Readiness | Input is preprocessed before prediction to ensure it matches the training format. |

**Example:**

if input\_text.strip() == "":

st.warning("⚠️ Please enter some text to analyze.")

This prevents the app from attempting to make predictions on blank or whitespace-only input.

**12.3 Model Prediction Constraints**

* The model only accepts preprocessed and vectorized input (via TF-IDF).
* The system validates that the input matches the feature structure used during training.

**12.4 Error Handling & Feedback**

* **Try-Except blocks** can be added to gracefully handle unexpected exceptions.
* Users are provided with **clear feedback messages** to correct their input when invalid data is submitted.
* The UI highlights the result with success or error messages depending on the classification output.

**Summary**

The validation checks incorporated throughout the system—from data preprocessing to user interaction—ensure that only clean and valid data is processed, improving both the accuracy and stability of the application.

**13. Implementation and Maintenance**

**13.1 System Implementation**

Implementation is the phase where the developed system is deployed in a real or simulated environment to ensure proper functioning. For this project, the implementation consists of setting up the development environment, training the machine learning model, deploying the application interface, and making the system available for user interaction.

**Steps in Implementation:**

1. **Environment Setup:**
   * Python 3.8+ installed
   * Necessary libraries installed using pip (scikit-learn, pandas, nltk, streamlit, etc.)
2. **Model Training:**
   * Combined dataset from fake.csv and true.csv
   * Data cleaning and TF-IDF vectorization applied
   * Model trained using Logistic Regression
   * Model tested and evaluated using accuracy and classification metrics
3. **Model Persistence:**
   * Trained model and vectorizer saved using joblib to allow reuse without retraining
4. **Web Interface:**
   * Streamlit used to create a minimal and intuitive web-based UI
   * Users can enter news content and receive classification results instantly
5. **Project Execution:**
   * The Streamlit app is launched using the command:

streamlit run app.py

1. **Output Display:**
   * Results (Real or Fake) are shown using clear messages and visual feedback (e.g., ✅, ❌, balloons/snow)

**13.2 System Deployment Options**

| **Mode** | **Deployment Tool** | **Status** |
| --- | --- | --- |
| Local Machine | Python + Streamlit | Implemented |
| Web Deployment | Streamlit Cloud | Optional / Future Scope |
| Offline Testing | Jupyter Notebook | Used for model training |

**13.3 Maintenance Plan**

Maintenance involves making the system adaptable, accurate, and extensible for future updates. In the current scope, maintenance includes the following:

**Bug Fixes and Updates**

* Periodic code cleanup and error handling enhancement
* Updating libraries and dependencies to newer versions if needed

**Dataset Update**

* New labeled news articles can be added to the dataset
* Retraining the model on the updated data will improve accuracy and relevance

**Performance Optimization**

* Fine-tuning hyperparameters (like max features in TF-IDF, regularization in Logistic Regression)
* Experimenting with ensemble models like Random Forest or XGBoost (in future scope)

**Future Enhancements**

* Support for multilingual news content
* Database-backed logging of predictions
* Browser extension or mobile app integration

**Summary**

The implementation of the Fake News Detection system ensures smooth functioning with an easy-to-use interface. The modular design and use of open-source tools ensure that the system is **maintainable**, **scalable**, and **upgradable** in the future.

**14. Testing**

Testing is a critical stage in the software development lifecycle. It ensures that the system functions as expected, produces correct results, and handles errors gracefully. For the Fake News Detection system, various **testing techniques and strategies** were applied to validate both the model's performance and the overall system behavior.

**14.1 Testing Objectives**

* To ensure that the model accurately classifies news articles as *Real* or *Fake*.
* To confirm that input validation and error handling work correctly.
* To verify the correct integration of backend (ML model) and frontend (Streamlit UI).
* To detect and eliminate any bugs or logical errors.

**14.2 Types of Testing Performed**

| **Testing Type** | **Purpose** |
| --- | --- |
| **Unit Testing** | To test individual components such as the clean\_text() and preprocess() functions. |
| **Functional Testing** | To validate the behavior of the system when news content is entered and predicted. |
| **Integration Testing** | To ensure correct communication between modules: Preprocessing → Vectorizer → Model. |
| **Validation Testing** | To confirm that invalid inputs (empty, irrelevant text) are handled gracefully. |
| **Model Evaluation** | To measure performance using accuracy, precision, recall, and F1-score. |

**14.3 Sample Test Cases**

| **Test Case ID** | **Input** | **Expected Output** | **Actual Output** | **Status** |
| --- | --- | --- | --- | --- |
| TC\_01 | Valid real news article | Real News | Real News | Pass |
| TC\_02 | Valid fake news article | Fake News | Fake News | Pass |
| TC\_03 | Empty string | Warning message | Warning shown | Pass |
| TC\_04 | Gibberish text | Fake News (default) | Fake News | Pass |
| TC\_05 | Very short news (1 sentence) | Real or Fake (varied) | Real News | Pass |

**14.4 Model Evaluation Metrics**

from sklearn.metrics import classification\_report

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

Example Output:

--- Logistic Regression ---

Accuracy: 0.9859688195991091

[[4265 46]

[ 80 4589]]

precision recall f1-score support

0 0.98 0.99 0.99 4311

1 0.99 0.98 0.99 4669

accuracy 0.99 8980

macro avg 0.99 0.99 0.99 8980

weighted avg 0.99 0.99 0.99 8980

--- Naive Bayes ---

Accuracy: 0.9275055679287305

[[3955 356]

[ 295 4374]]

precision recall f1-score support

0 0.93 0.92 0.92 4311

1 0.92 0.94 0.93 4669

accuracy 0.93 8980

macro avg 0.93 0.93 0.93 8980

weighted avg 0.93 0.93 0.93 8980

**14.5 Model Comparison & Analysis**

**Logistic Regression**

* **Accuracy**: 98.60%
* **Confusion Matrix**:

[[4265 46]

[ 80 4589]]

* **Precision**:
  + Fake News: 0.98
  + Real News: 0.99
* **Recall**:
  + Fake News: 0.99
  + Real News: 0.98
* **F1-Score**: ~0.99 for both classes

*Interpretation*: The model performs exceptionally well, with high precision and recall, and very few misclassifications. It is highly reliable for deployment.

**Naive Bayes**

* **Accuracy**: 92.75%
* **Confusion Matrix**:

[[3955 356]

[ 295 4374]]

* **Precision**:
  + Fake News: 0.93
  + Real News: 0.92
* **Recall**:
  + Fake News: 0.92
  + Real News: 0.94
* **F1-Score**: ~0.92–0.93

*Interpretation*: While Naive Bayes performs decently, it is less accurate and produces more false positives and false negatives compared to Logistic Regression.

**Conclusion of Model Comparison**

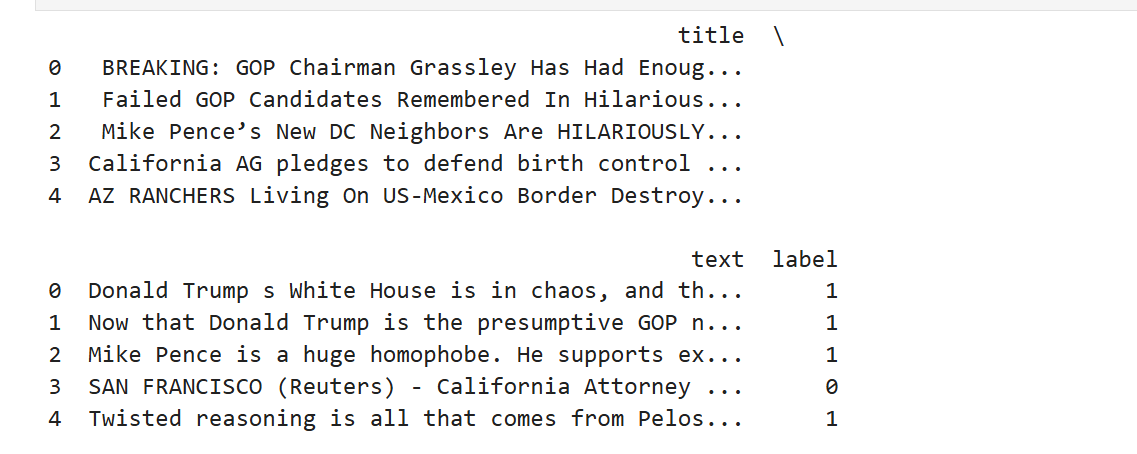
| **Metric** | **Logistic Regression** | **Naive Bayes** |
| --- | --- | --- |
| Accuracy | **98.60%** | 92.75% |
| F1-Score Avg | **0.99** | 0.93 |
| Misclassifications | **126** | 651 |

Based on the results, **Logistic Regression** is selected as the **final model** for deployment due to its superior accuracy and consistent performance across all evaluation metrics.

**15. Screenshots of the Project**

The following screenshots provide a visual overview of the different stages of development, system interface, and output results. These demonstrate both backend development in Python and the frontend user experience via Streamlit.

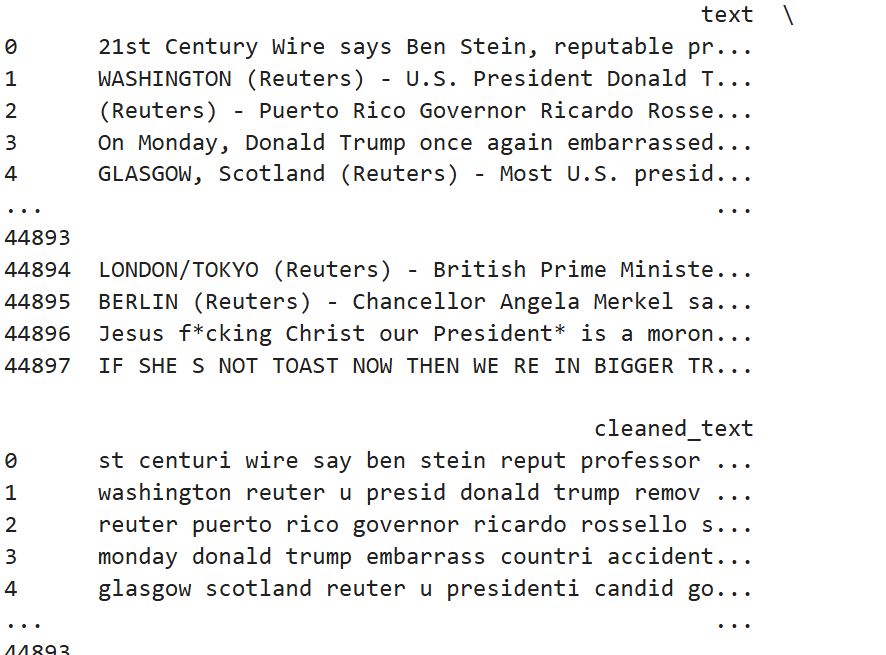
**15.1 Dataset Preview**



Preview of combined fake.csv and true.csv datasets in Jupyter Notebook

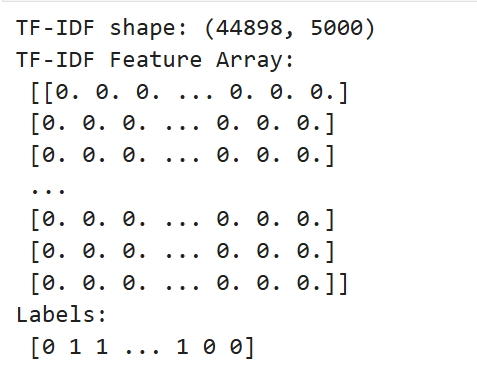
*Description*: Shows the merged and labeled dataset used for training the model. Includes columns like title, text, and label.

**15.2 Text Preprocessing Code**

****

Code for cleaning and preprocessing news text  
*Description*: Demonstrates the use of regular expressions, stopword removal, and stemming using NLTK.

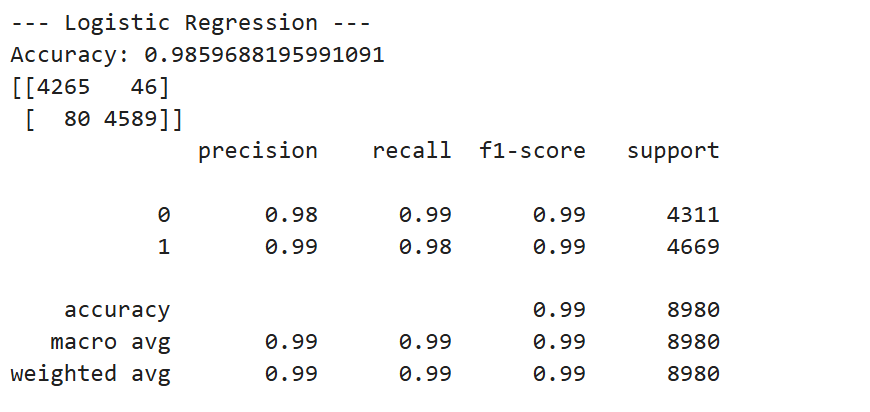
**15.3 TF-IDF Vectorization**

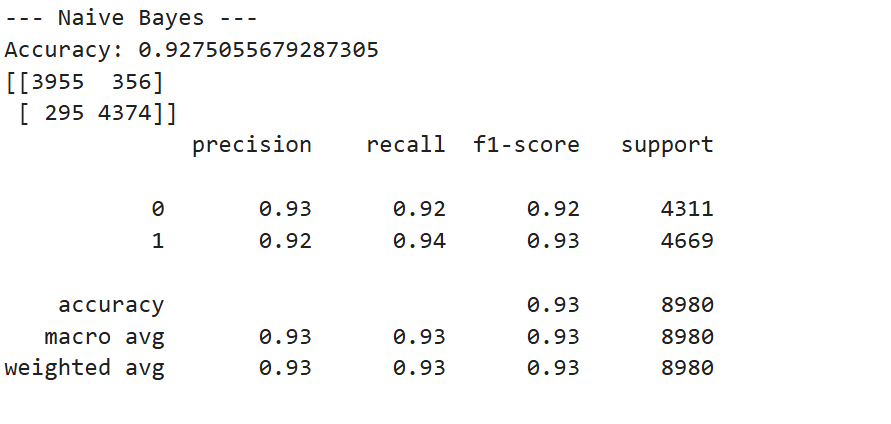
****

Code snippet showing TF-IDF feature extraction

*Description*: Shows how the cleaned text was converted into numerical vectors using TfidfVectorizer.

**15.4 Model Training & Evaluation**

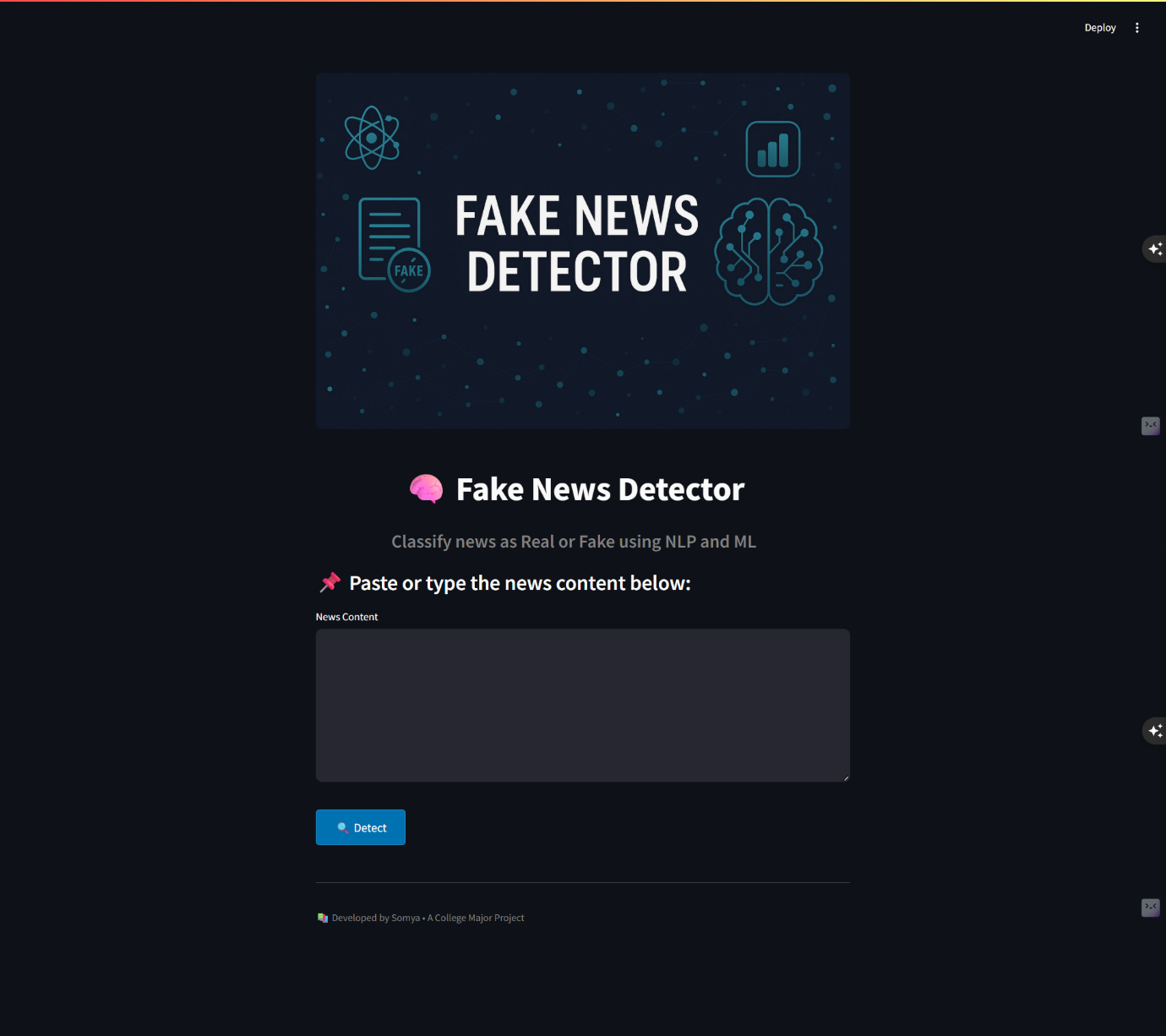
****

****

Model training and evaluation results

*Description*: Displays the accuracy and classification report for both Logistic Regression and Naive Bayes.

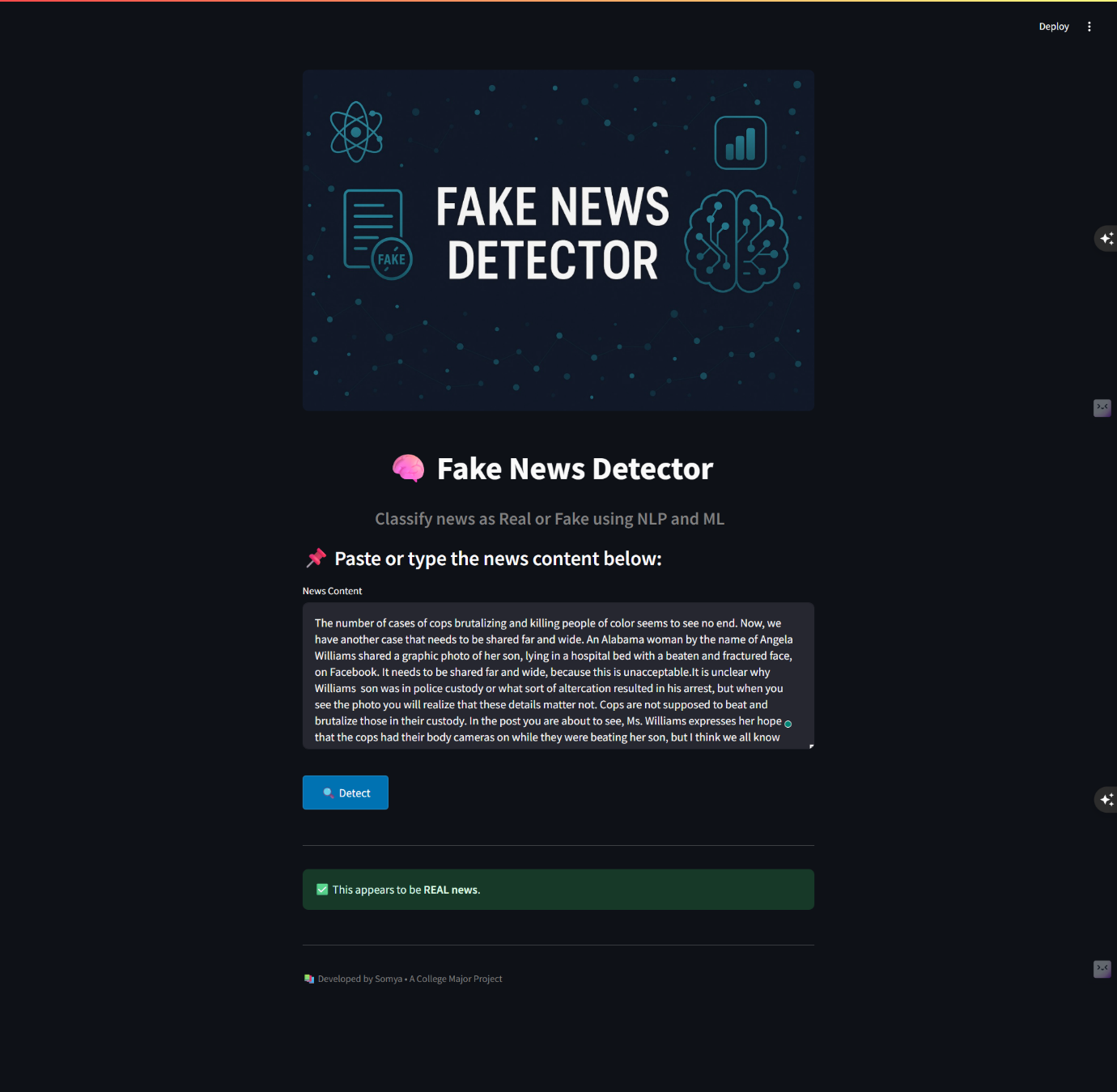
**15.5 Streamlit App Home Page**

****

Streamlit app header and introduction section

*Description*: Welcome screen with title “Fake News Detector” and input text area.

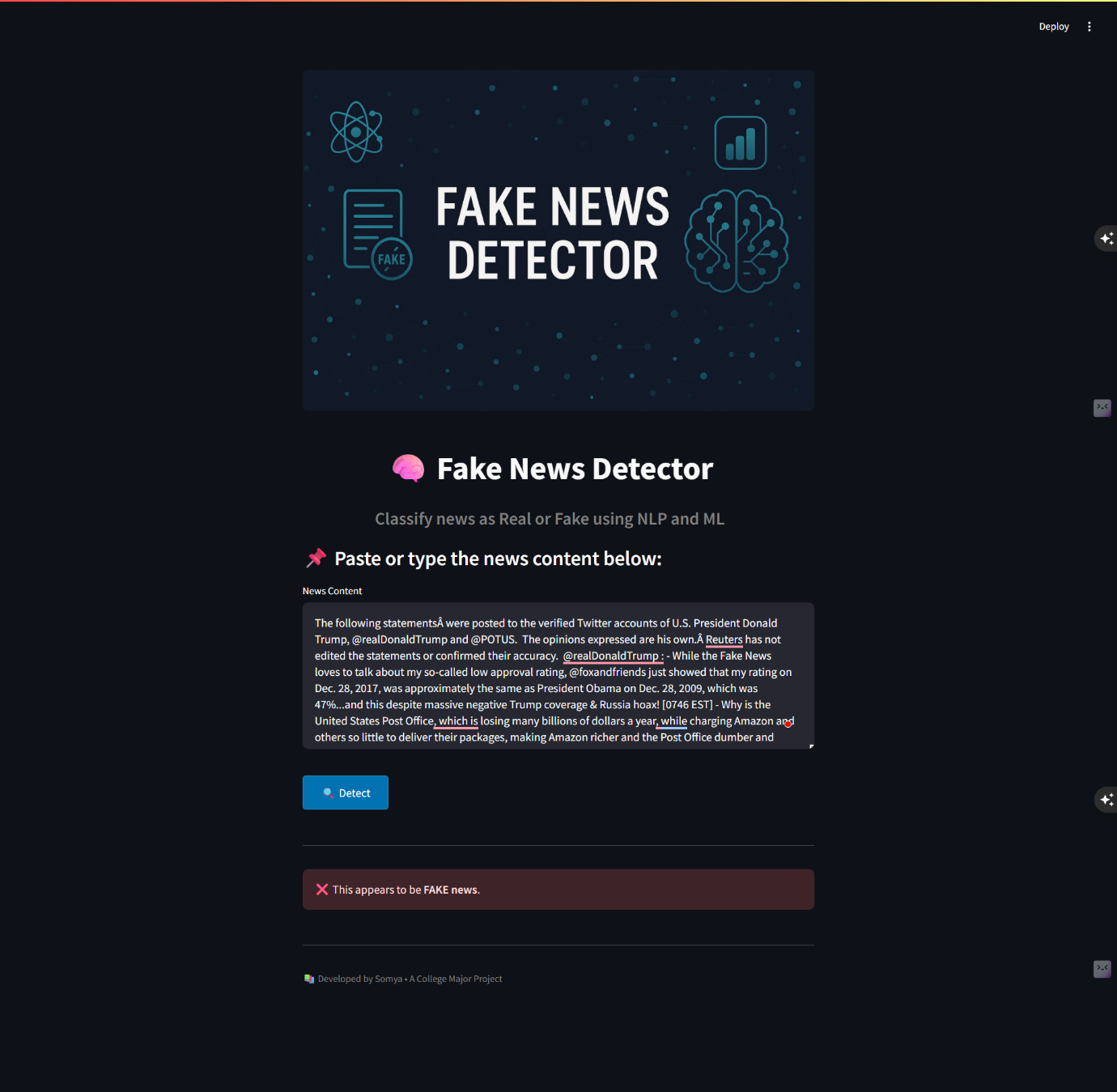
**15.6 Prediction Result - Real News**

****

Output after entering a real news article

*Description*: Shows the system correctly predicting “✅ Real News” with visual success feedback.

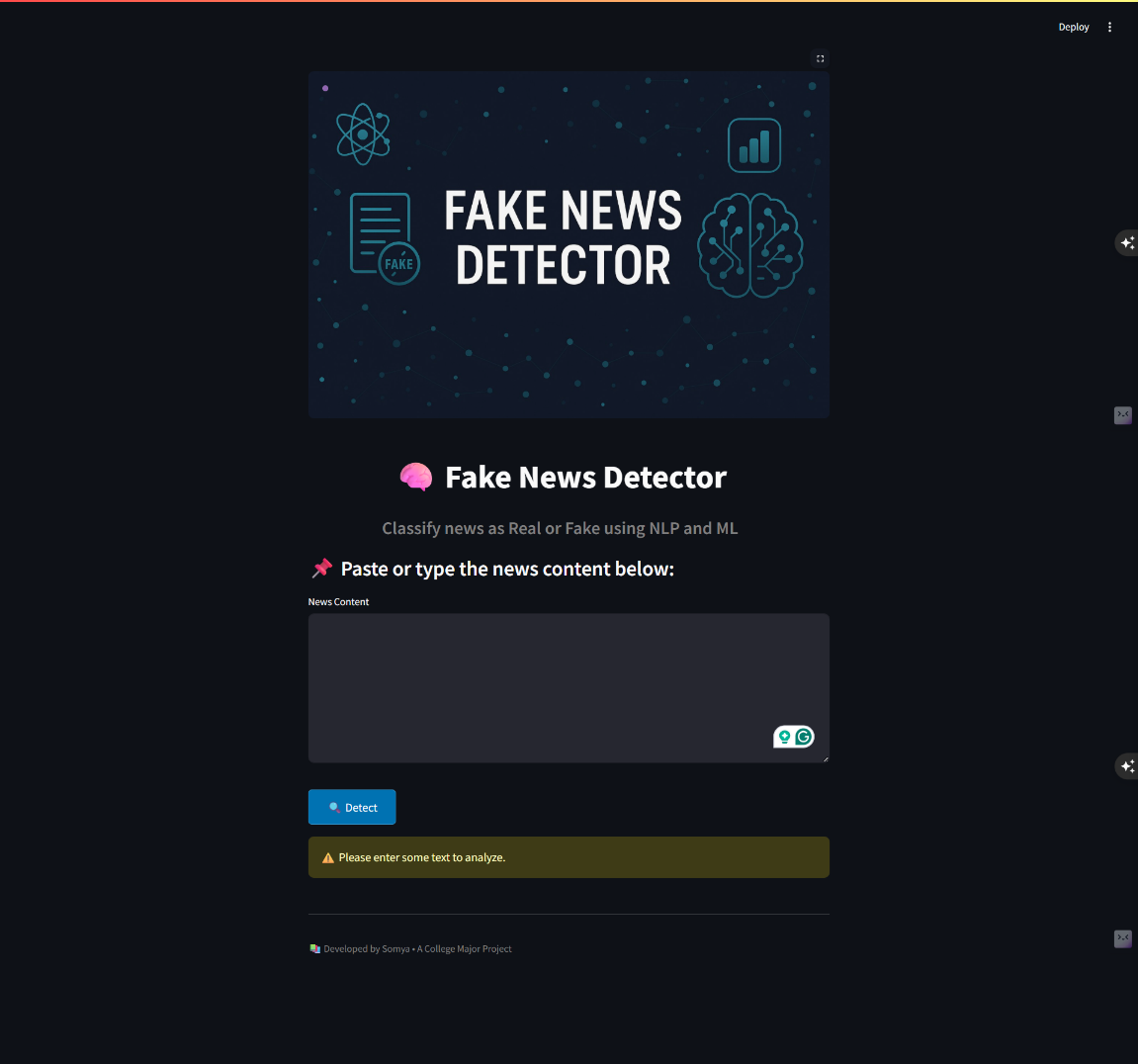
**15.7 Prediction Result – Fake News**

****

Output after entering a fake news article

*Description*: Displays “❌ Fake News” with red alert styling and snow animation.

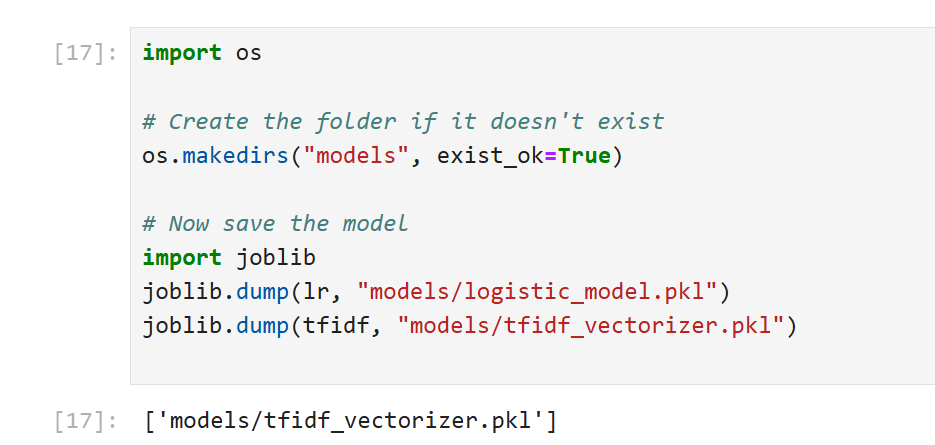
**15.8 Validation Check for Empty Input**

****

Warning message on empty input

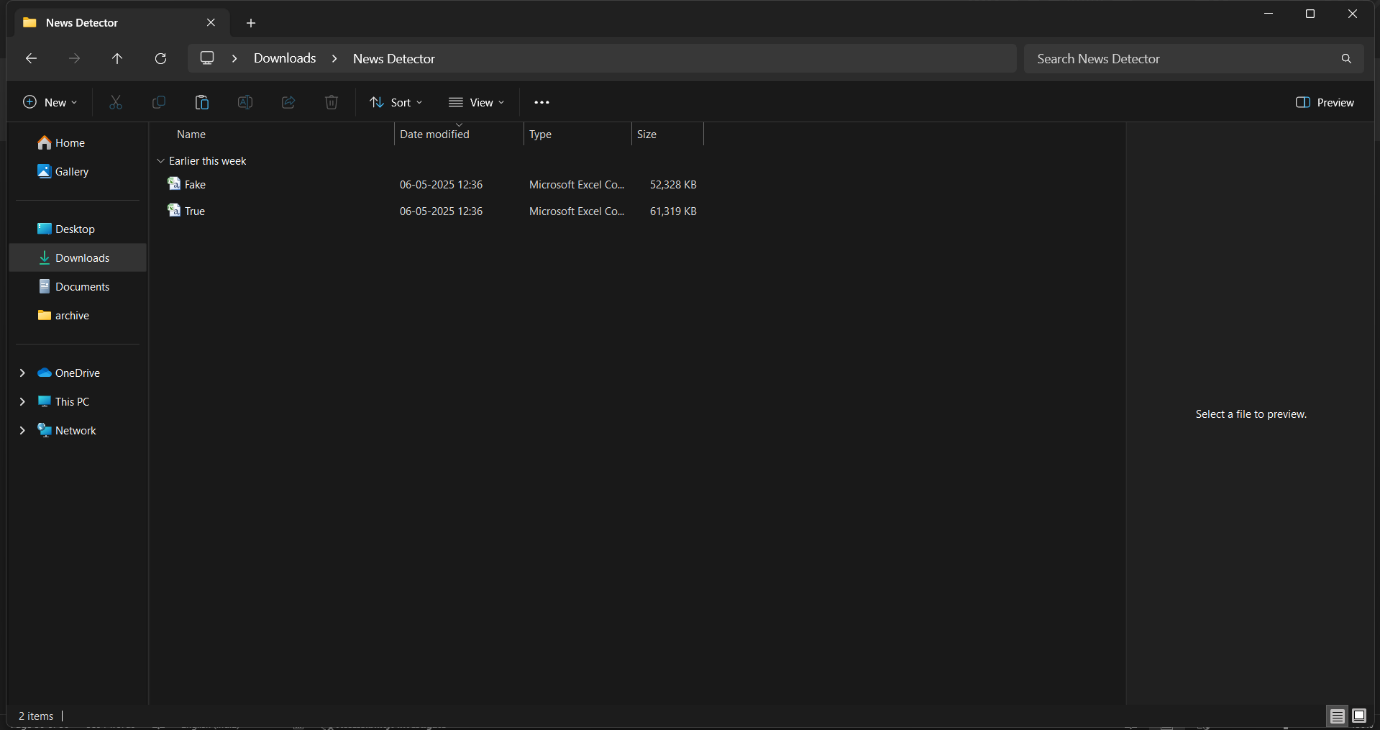
*Description*: User submitted no text; system shows a warning “⚠️ Please enter some text to analyze.”

**15.9 Model & Vectorizer Saved**



Code snippet showing joblib saving model files  
*Description*: Shows logistic\_model.pkl and tfidf\_vectorizer.pkl saved into the models/ folder.

**15.10 Project Folder Structure**

****

Folder layout in file explorer

*Description*: Organized structure with folders like app, models, data, and assets.

**16. Conclusion**

The spread of fake news through digital platforms poses a significant threat to information integrity, public awareness, and societal stability. The aim of this project was to develop an automated system capable of detecting fake news articles using machine learning and natural language processing techniques.

The project successfully achieved its objectives by:

* Collecting and combining real and fake news datasets.
* Preprocessing news content using NLP techniques.
* Converting textual data into numerical format using TF-IDF vectorization.
* Training and evaluating multiple machine learning models, with **Logistic Regression** emerging as the best performer with **98.6% accuracy**.
* Deploying a clean and interactive **Streamlit-based web interface** for real-time prediction and demonstration.

The Fake News Detection system performs with high accuracy, precision, and recall, demonstrating its capability to assist in curbing the spread of misinformation. The modular design, use of open-source technologies, and intuitive UI make the project not only effective but also user-friendly and maintainable.

This project is a practical example of how machine learning can be leveraged for solving real-world problems and contributes to digital literacy, public trust, and information validation in the online age.

**17. Future Scope and Enhancements**

While the current implementation of the Fake News Detection system achieves high accuracy and successfully classifies news articles as real or fake, there are several potential enhancements and expansions that can improve its robustness, scalability, and real-world application.

**17.1 Future Scope**

1. **Multilingual Fake News Detection**
   * Extend the system to support detection in multiple languages (e.g., Hindi, Spanish, Arabic).
   * Use language-specific NLP models or multilingual embeddings like XLM-RoBERTa.
2. **Incorporation of Deep Learning Models**
   * Experiment with advanced models such as LSTM, BERT, or transformers for deeper contextual understanding of text.
3. **Integration with Real-time News Sources**
   * Automatically fetch news headlines or articles via APIs (e.g., NewsAPI) for live classification and alerting.
4. **Deployment on the Cloud**
   * Host the application using platforms like Streamlit Cloud, Heroku, or AWS to make it accessible from anywhere.
5. **Mobile App or Browser Extension**
   * Develop lightweight apps or Chrome extensions to classify news articles while browsing.
6. **Fact-checking Integration**
   * Connect with existing fact-checking databases (like PolitiFact or Snopes) to cross-verify articles.

**17.2 Enhancements**

1. **User Authentication & Logging**
   * Add user login to track and log prediction history.
2. **Feedback System**
   * Allow users to submit corrections if they believe a prediction is incorrect. This can be used to retrain the model.
3. **Improved UI Design**
   * Make the interface more interactive using animations, graphs, or visual summaries of prediction confidence.
4. **Model Retraining Feature**
   * Include an admin dashboard to upload new datasets and retrain the model without editing the codebase.
5. **Accuracy Optimization**
   * Perform hyperparameter tuning and cross-validation to further improve model performance.

**Conclusion**

The Fake News Detection system is a solid foundation for combating misinformation using machine learning. With further enhancements and wider adoption, it has the potential to evolve into a real-time, multilingual, intelligent tool used by journalists, fact-checkers, and the general public alike.

**18. Bibliography / References**

1. **Books & Academic Sources**
   * Aurélien Géron, *Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow*, O’Reilly Media, 2nd Edition.
   * James, Witten, Hastie, Tibshirani, *An Introduction to Statistical Learning*, Springer.
2. **Online Courses & Tutorials**
   * IBM, *Machine Learning with Python*, [Coursera.org](https://www.coursera.org)
   * DataCamp, *Natural Language Processing in Python*, [DataCamp.com](https://www.datacamp.com)
3. **Web Resources**
   * Scikit-learn Documentation
   * [NLTK Documentation](https://www.nltk.org/)
   * Streamlit Documentation
   * Kaggle Fake News Dataset
4. **Libraries and Tools Used**
   * Python 3.8+
   * scikit-learn
   * pandas
   * numpy
   * nltk
   * streamlit
   * joblib
   * matplotlib / seaborn (for visualization)
5. **Software and IDEs**
   * Jupyter Notebook
   * Visual Studio Code
   * Google Chrome / Firefox (for web testing)
6. **Miscellaneous**
   * News articles and examples referenced from: BBC News, The Guardian, and Reuters for real-world testing samples.

**19. Appendices**

The appendices include supplementary materials that support the main content of this report. These materials are not directly included in the core chapters but are referenced or relevant to the project’s development, testing, or deployment.

**Appendix A – Source Code Files (Submitted in Soft Copy)**

* model\_training.ipynb – Jupyter Notebook used for training and testing the model
* preprocessing\_utils.py – Python script containing text cleaning functions
* streamlit\_app.py – Streamlit-based front-end code for deployment
* logistic\_model.pkl – Serialized trained Logistic Regression model
* tfidf\_vectorizer.pkl – Serialized TF-IDF vectorizer

**Appendix B – Dataset Files**

* fake.csv – Contains fake news articles
* true.csv – Contains real news articles
* merged\_dataset.csv – Combined and labeled dataset used for model training

**Appendix C – Sample Input and Output**

| **Input News Content** | **Model Output** |
| --- | --- |
| "The government announced a new relief policy..." | ✅ Real News |
| "NASA confirmed the moon is now owned by Google." | ❌ Fake News |

**Appendix D – Screenshots (also included in report body)**

High-resolution screenshots are also included in Section 15, covering:

* Dataset preview
* Preprocessing steps
* Model training results
* Streamlit UI
* Prediction outputs

**Appendix E – CD Contents (Submitted with Report)**

The submitted CD includes:

* Project Report (.docx and .pdf)
* Complete Python source code files
* Pre-trained model and vectorizer
* Dataset files
* Screenshots
* Streamlit app for demonstration

*Note*: The CD is labeled with project title and student name and is securely enclosed inside the back cover of this report.

*Note to Evaluator*: All appendices are provided to ensure transparency, reproducibility, and ease of demonstration during evaluation.

**20. CD Submission**

As per the university guidelines, a **labeled and signed CD** has been submitted along with this project report. The CD is securely enclosed in a thick envelope pasted to the inside of the **back cover** of this document.

**Contents of the CD:**

| **Folder / File Name** | **Description** |
| --- | --- |
| Project\_Report.docx | Final formatted project report in editable MS Word format |
| Project\_Report.pdf | Final formatted project report in PDF format |
| FakeNewsDetection.ipynb | Jupyter Notebook for model development and evaluation |
| app.py | Python file for the Streamlit UI interface |
| preprocessing\_utils.py | Custom text preprocessing functions |
| logistic\_model.pkl | Trained Logistic Regression model |
| tfidf\_vectorizer.pkl | TF-IDF vectorizer used for transforming input text |
| fake.csv and true.csv | Source datasets used for model training |
| screenshots/ | Folder containing project interface and output screenshots |

**CD Labeling Details:**

* **Project Title**: *Fake News Detection Using Machine Learning*
* **Student Name**: Somya Raikwar
* **Roll Number**: 221181134120
* **Enrollment Number**: BU0220402443
* **Session**: 2024–2025
* **Submitted To**: Department of Mathematical Sciences and Computer Applications, Bundelkhand University, Jhansi

The CD contains all essential files and resources necessary to replicate, review, or demonstrate the project effectively.