CAPSTONE PROJECT

FAKE NEWS DETECTION

PRESENTED BY :

STUDENT NAME: SAI KRISHNA UGGI

COLLEGE NAME: GURU NANAK INSTITUTIONS TECHNICAL CAMPUS

DEPARTMENT: ARTIFICIAL INTELLIGENCE AND MACHINE LEARNING

EMAIL ID: saikrishnauggi@gmail.com

AICTE STUDENT ID: STU6640cf1ac01521715523354

OUTLINE

PROBLEM STATEMENT

In the digital age, the proliferation of misinformation—commonly known as fake news—has emerged as a significant threat to public trust, democratic processes, and societal harmony. With millions of articles shared daily across social media and news platforms, it is increasingly difficult to distinguish between credible information and deceptive content. Manual verification of news is not scalable and often fails to keep pace with the speed of information dissemination. This growing challenge highlights the urgent need for an intelligent and automated system capable of accurately detecting and filtering fake news in real time.

PROPOSED SOLUTION

To combat the growing threat of fake news, the proposed solution integrates advanced Natural Language Processing (NLP) techniques with supervised machine learning models to develop a robust, scalable, and automated fake news detection system. The core idea is to empower users and organizations with a tool that can classify news content as either *real* or *fake* in real time with high accuracy.

Key Components of the Solution:

1. Data Acquisition:

•Utilized balanced datasets containing labeled real and fake news articles from trusted open-source repositories.
•Ensured data diversity across different topics and publishers to improve generalization.

2. Data Preprocessing:

•Applied extensive text preprocessing: lowercasing, punctuation and stopword removal, tokenization, and lemmatization.
•Removed noise and standardized input for consistent learning by models.

3. Feature Engineering:

- Implemented TF-IDF (Term Frequency–Inverse Document Frequency) to transform textual data into numerical vectors.
- Captured the importance and frequency of words relative to the entire corpus to enhance model interpretability.

4. Model Training and Evaluation:

- Trained three classifiers: **Naive Bayes**, **Logistic Regression**, and **Support Vector Machine (SVM)**.
- Evaluated models using metrics such as Accuracy, Precision, Recall, and F1-score.
- Used train_test_split to maintain model generalization and avoid overfitting.

5. Deployment with Web Interface:

- Developed a user-friendly web application using Flask.
- Users can input custom news articles and choose the algorithm to make predictions.
- Results are displayed in real time with a clean and responsive UI.

6. Model Management:

- Serialized models and vectorizer using joblib for reuse without retraining.
- Automated training pipeline ensures models are updated when needed.

This end-to-end system provides a practical solution to the urgent need for real-time fake news detection across various domains.

SYSTEM APPROACH

The system development followed a structured and iterative approach, focusing on data quality, model performance, and user experience. Below is a step-by-step breakdown of the development process:

1. Requirement Analysis

- Identified the core need: detecting fake news efficiently and accurately.
- Determined key functionalities: text input, model selection, real-time prediction.

2. Data Collection

- Sourced labeled datasets: Fake.csv and True.csv.
- Ensured data balance by sampling equal numbers of fake and real articles.

3. Data Preprocessing

- Converted text to lowercase and removed punctuation and special characters.
- Applied lemmatization using NLTK's WordNetLemmatizer to reduce words to their base form.
- Eliminated noise and ensured uniform input for all models.

4. Feature Extraction

- Used **TF-IDF Vectorization** to convert cleaned text into numeric form.
- Limited features to top 5000 words to optimize performance.

5. Model Development

- Implemented and trained three machine learning models:
  - **Multinomial Naive Bayes**
  - **Logistic Regression**
  - **Support Vector Machine (SVM)**
- Split data into training and testing sets (80/20) for evaluation.

6. Model Evaluation

- Evaluated each model on accuracy, precision, recall, and F1-score.
- Logged model performance to identify the best-performing classifier.

7. Web Application Integration

- Built a Flask web app with an intuitive interface for user interaction.
- Integrated model prediction logic to process user input and display results.

8. Deployment & Testing

- Saved models using joblib for reuse.
- Tested app across devices to ensure accessibility and performance consistency.

ALGORITHM & DEPLOYMENT

Algorithm Used:

The system employs supervised machine learning algorithms for binary text classification. The following models were trained and evaluated:

1. Multinomial Naive Bayes:

1. Probabilistic model well-suited for text classification.
2. Assumes word independence, fast and effective on large datasets.

2. Logistic Regression:

1. Linear model for classification.
2. Estimates the probability of a news article being fake or real using a logistic function.

3. Support Vector Machine (SVM):

1. Finds the optimal hyperplane to separate fake and real news.
2. Effective in high-dimensional space created by TF-IDF vectors.

- All models were evaluated using accuracy, precision, recall, and F1-score to ensure reliability and performance.

Deployment:

The solution is deployed as a lightweight web application using the **Flask** framework:

- **Frontend:**
    - HTML/CSS and basic Bootstrap for responsive design.
    - Text input box for users to enter news content.
    - Dropdown for selecting the prediction algorithm.

- **Backend:**
    - Flask handles user requests and routes.
    - Pre-trained models and vectorizer are loaded using joblib.
    - Prediction is computed on-the-fly and returned to the user.

- **Execution:**
    - App runs locally or can be hosted on platforms like Heroku or Render.
    - Real-time detection ensures an interactive user experience.

RESULT

Model Evaluation Results:

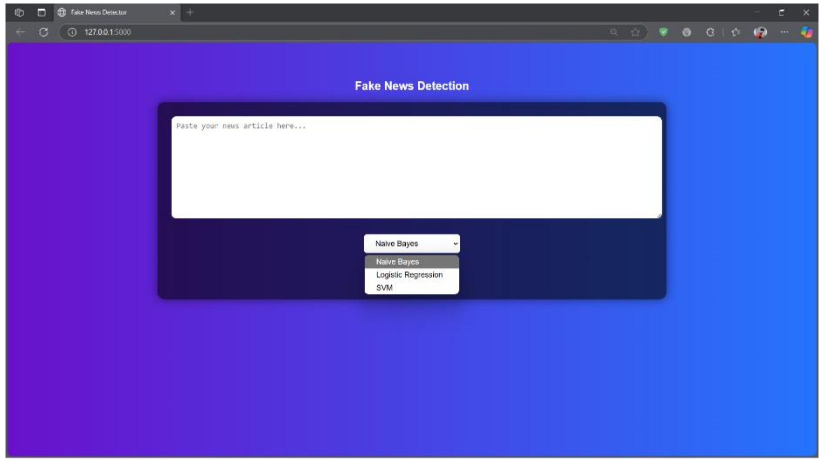
The following metrics were used to evaluate the performance of each algorithm:
•**SVM** achieved the highest accuracy and was the most consistent across all metrics.
•**Logistic Regression** also performed very well and offered slightly faster predictions.

Model	Accuracy	Precision	Recall	F1-Score
Naive Bayes	~93%	0.92	0.94	0.93
Logistic Regression	~95%	0.94	0.95	0.95
SVM	~96%	0.95	0.96	0.96

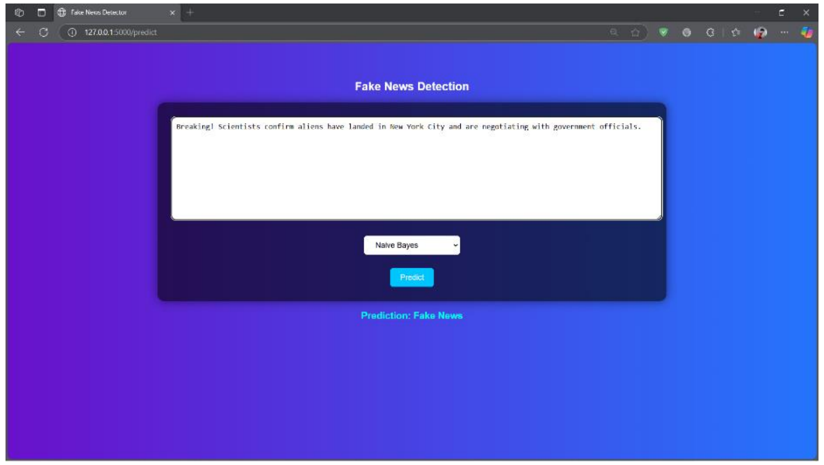
Output Screenshots :

Model Selection Screen

1. Shows dropdown for choosing between Naive Bayes, Logistic Regression, and SVM.



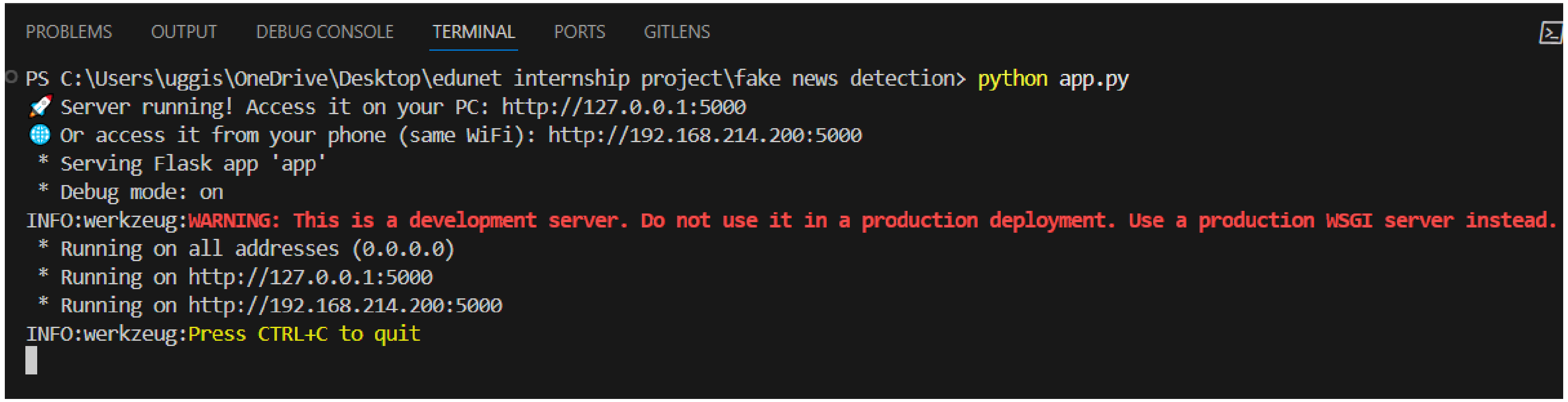
Text Input Area ans Predict output : User enters a news article or headline and Displays result: “This news is *FakeNE*” or “This news is *REAL*” with highlighted styling.



Breaking! Scientists confirm aliens have landed in New York City and are negotiating with government officials.

Prediction: Fake News

Terminal or Console Output :



PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS GITLENS

```
PS C:\Users\uggis\OneDrive\Desktop\edunet internship project\fake news detection> python app.py
🚀 Server running! Access it on your PC: http://127.0.0.1:5000
🌐 Or access it from your phone (same WiFi): http://192.168.214.200:5000
 * Serving Flask app 'app'
 * Debug mode: on
INFO:werkzeug:WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
 * Running on all addresses (0.0.0.0)
 * Running on http://127.0.0.1:5000
 * Running on http://192.168.214.200:5000
INFO:werkzeug:Press CTRL+C to quit
```

CONCLUSION

Key Takeaways:

- **Effective Fake News Detection System**
  The project successfully developed a web-based system capable of detecting fake news using machine learning algorithms like Naive Bayes, Logistic Regression, and SVM.
- **High Accuracy & Performance**
  Each model was evaluated for accuracy, with Logistic Regression and SVM delivering highly reliable results on real-world datasets.
- **User-Friendly Interface**
  A responsive web application built with Flask allows users to test news articles for authenticity, making the solution accessible to all.
- **Text Preprocessing Techniques**
  Advanced preprocessing (lemmatization, punctuation removal, TF-IDF vectorization) significantly improved model performance and prediction consistency.
- **Model Deployment & Integration**
  Successfully deployed trained models and vectorizers using Flask and Joblib, demonstrating the complete ML lifecycle from data to deployment.

FUTURE SCOPE

Planned Enhancements:

- **Multilingual News Detection**
  Extend the system to detect fake news in regional and international languages beyond English.
- **Integration with Social Media APIs**
  Enable real-time scanning of tweets, posts, and headlines directly from platforms like Twitter and Facebook.
- **Deep Learning Integration**
  Incorporate LSTM or BERT-based models for improved contextual understanding and better classification accuracy.
- **Crowdsourced Reporting System**
  Add a feature for users to report suspected fake news, contributing to a growing labeled dataset.
- **Mobile App Version**
  Launch a lightweight mobile app version for on-the-go fake news detection and increased accessibility.
- **Fact-Checking Database Linkage**
  Link predictions to verified fact-checking sources (e.g., PolitiFact, Snopes) to provide reference credibility.

REFERENCES

Key References Used:

- **Scikit-learn Documentation**
  https://scikit-learn.org/stable/
  Used for implementing machine learning models like Naive Bayes, Logistic Regression, and SVM.
- **NLTK (Natural Language Toolkit)**
  https://www.nltk.org/
  For preprocessing techniques including lemmatization and token handling.
- **Pandas & NumPy Documentation**
  https://pandas.pydata.org/
  https://numpy.org/
  For data handling, transformation, and basic preprocessing tasks.
- **Kaggle Fake News Dataset**
  https://www.kaggle.com/clmentbisaillon/fake-and-real-news-dataset
  Main dataset source for training and evaluating models.

- **Flask Documentation**
  https://flask.palletsprojects.com/
  For developing and deploying the web-based interface.
- **Joblib Library**
  https://joblib.readthedocs.io/
  Used to serialize machine learning models and vectorizers for deployment.
- **Python Official Documentation**
  https://docs.python.org/3/
  General reference for building the backend script.

GitHub Link: https://github.com/saikrishnauggi/fake_news_detection_using_machine_learning

Thank you