Fake News Detection Using Machine Learning and Deep Learning

Problem Statement:

The rapid proliferation of fake news across digital platforms undermines the credibility of information, influences public opinion, and can have serious societal consequences. Detecting fake news automatically is crucial for maintaining the integrity of online content and supporting informed decision-making. This project focuses on developing a robust system that can distinguish between fake and real news articles using both traditional machine learning and deep learning techniques.

Approach:

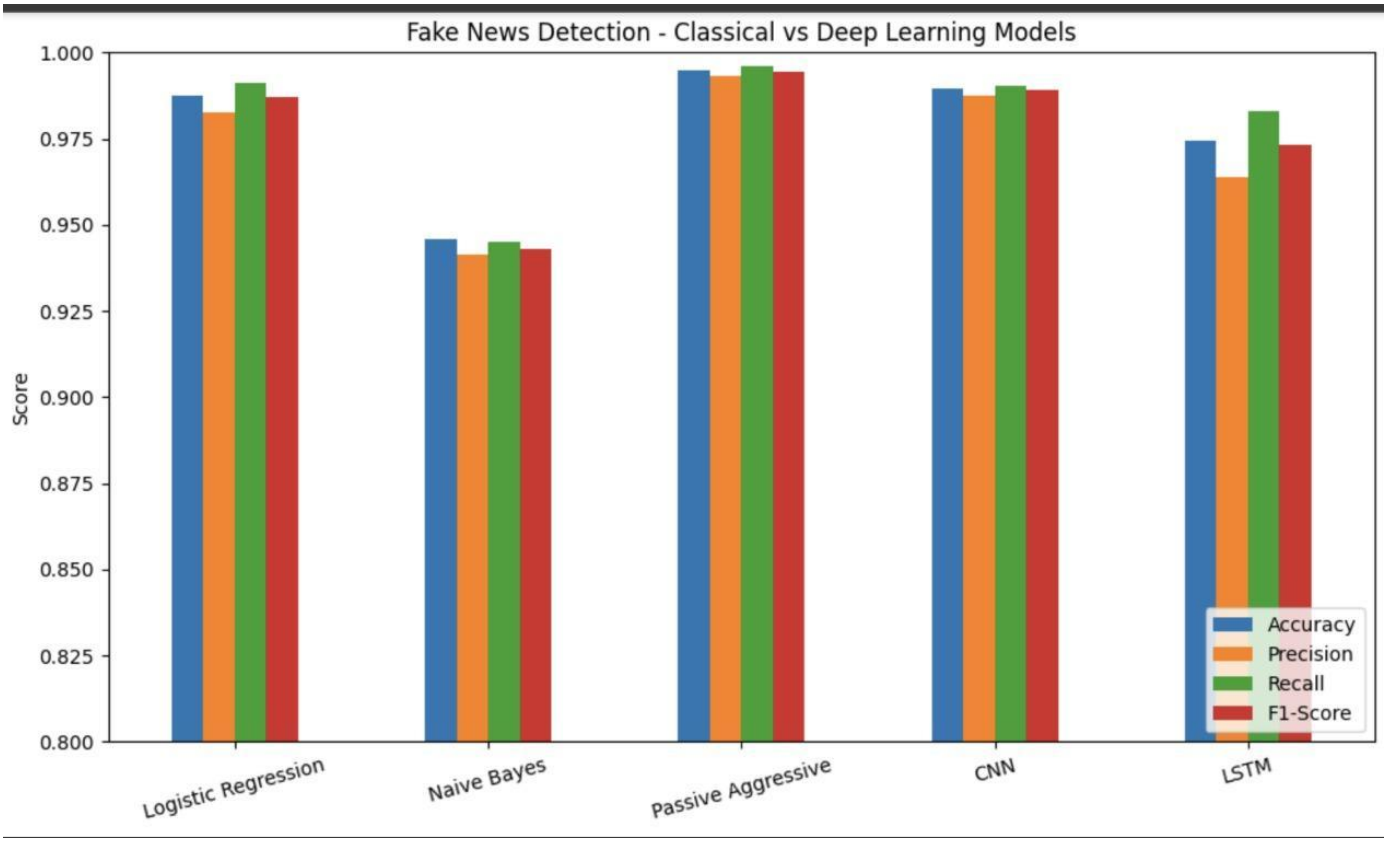
• Data Collection Combined two datasets — True.csv (21,417 real) and Fake.csv (23,481 fake). Total of 44,898 news articles, each labeled as 0 (Fake) or 1 (Real)
• Preprocessing: Cleaned text by removing URLs, HTML tags, non-alphabetic characters, and stopwords; performed lemmatization to reduce words to their base forms.
• Feature Extraction: Converted cleaned text into numerical representations using TF-IDF vectorization for machine learning models, and tokenized sequences with embeddings for CNN and LSTM deep learning models.
• Model Training: Trained several models for comparison – Logistic Regression, Naive Bayes, Passive Aggressive Classifier (ML), and CNN, LSTM (DL). Hyperparameters were optimized for performance.
• Model Evaluation: Evaluated all models using standard metrics (Accuracy, Precision, Recall, F1-Score). Conducted error analysis to identify misclassifications and important contributing features or words.
• Visualization: Generated plots to compare model performances and identify trends across different algorithms.

Implementation Overview:

• **data_loader.py:** Handles loading and merging datasets from ZIP files, shuffling, and labeling.
• **preprocessing.py:** Implements text cleaning, tokenization, stopword removal, and lemmatization using NLTK.
• **train_models.py:** Trains traditional ML models (Logistic Regression, Naive Bayes, Passive Aggressive) and deep learning models (CNN, LSTM) with TensorFlow/Keras; performs vectorization and tokenization.
• CNN Architecture: Embedding layer → Conv1D → GlobalMaxPooling → Dense → Dropout → Output layer (sigmoid).
• LSTM Architecture: Embedding layer → LSTM (128 units) → Dense → Dropout → Output layer (sigmoid).
• **evaluate.py:** Generates bar plots for performance comparison of all models.
• **main.py:** Integrates the full pipeline: data loading → preprocessing → model training → evaluation → saving results.

Model Performance Comparison:

```
 Model Performance Comparison:
                     Accuracy  Precision  Recall  F1-Score
Logistic Regression    0.9875     0.9827  0.9913    0.9870
Naive Bayes            0.9457     0.9414  0.9449    0.9432
Passive Aggressive     0.9948     0.9932  0.9960    0.9946
CNN                    0.9895     0.9876  0.9904    0.9890
LSTM                   0.9743     0.9640  0.9829    0.9733
```



- Logistic Regression: Accuracy 0.9875, Precision 0.9827, Recall 0.9913, F1-Score 0.9870.
- Naive Bayes: Accuracy 0.9457, Precision 0.9414, Recall 0.9449, F1-Score 0.9432.
- Passive Aggressive: Accuracy 0.9948, Precision 0.9932, Recall 0.9960, F1-Score 0.9946.
- CNN: Accuracy 0.9895, Precision 0.9876, Recall 0.9904, F1-Score 0.9890.
- LSTM: Accuracy 0.9743, Precision 0.9640, Recall 0.9829, F1-Score 0.9733

on.

Conclusions and Challenges:

• LSTM and Passive Aggressive models provided the highest accuracy and F1-Score, effectively distinguishing fake from real news.
• Preprocessing and feature engineering were critical for model performance.
• Challenges: Managing class imbalance, avoiding overfitting in deep learning models, and ensuring the pipeline generalizes to unseen news sources.
• Future Work: Explore transformer-based models like BERT, RoBERTa for contextual understanding, integrate multi-source news datasets, and develop real-time detection systems.