Phase-2 Submission

Exposing the truth with advanced fake news detection powered by Natural language processing

Student Name: S.Divya

Register Number :513523104016

Institution: Annai Mira College Of Engineering And Technology

Department: B.E Computer Science Engineering

Date of Submission: 02.05.2025

Github Repository link:

1.Problem Statement

The rapid spread of fake news through digital platforms poses a serious threat to public awareness, democratic discourse, and social trust. Social media allows unverified or deceptive news to reach millions within minutes, influencing opinions and behaviors. Traditional methods of manual fact-checking are time-consuming, subjective, and insufficient for the scale of misinformation we face today. Therefore, an automated, accurate, and scalable solution is essential for identifying and mitigating fake news using artificial intelligence.

2.Project Objectives

- Automate the detection of fake news using text classification models.
- Utilize natural language processing (NLP) to extract and analyze textual features.
- Train and compare multiple machine learning algorithms for best performance.

- Develop a pipeline that can be extended to other domains like social media monitoring.
- Present visual insights into how the models make predictions and what linguistic patterns differentiate fake from real news.

3.Flowchart of the Project Workflow

Data Collection

↓

Data Preprocessing

↓

Exploratory Data Analysis (EDA)

↓

Feature Engineering

↓

Model Building and Training

↓

Model Evaluation

↓

Visualization and Interpretation

↓

Final Report & Deployment (Optional)

4.Data Description

Source: Fake and Real News Dataset from Kaggle.

Total Records: ~20,000 news articles.

Key Features:

Title – Headline of the article.

Text – Main body content of the article.

Subject – Topic category (e.g., politics, world news).

Date – Publication date.

Label – Ground truth (FAKE or REAL).

The dataset provides a diverse collection of news articles, enabling the model to generalize across different topics and writing styles.

5.Data Preprocessing

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

Text Normalization: Lowercasing, removing HTML tags, punctuation, and digits.

Stopword Removal: Eliminated common but non-informative words like “the”, “and”, etc.

Tokenization: Split text into individual words or tokens.

Lemmatization: Reduced words to their root form (e.g., “running” → “run”).

Vectorization: Used TF-IDF (Term Frequency–Inverse Document Frequency) to convert text into numerical feature vectors.

Handling Imbalance: Checked for class imbalance and used stratified sampling if needed.

6.Exploratory Data Analysis (EDA)

The EDA phase involved statistical and visual examination of the dataset to uncover insights:

Label Distribution: Ensured balance between fake and real articles.

Word Clouds: Created for both fake and real news to visualize frequent terms.

Top Keywords: Identified high-frequency terms in each category.

Article Length: Analyzed text length distribution to identify potential correlations.

Subject Trends: Examined if certain subjects were more prone to fake news.

Findings showed that fake news often used more sensational language and emotional tone, while real news leaned towards factual reporting.

7.Feature Engineering

Advanced techniques were used to enrich the model with meaningful inputs:

TF-IDF Vectors: Captured term relevance by reducing the effect of commonly occurring words.

N-grams: Included bigrams and trigrams to capture short phrases and context.

P**art-of-Speech Tags (optional)**: Included grammatical structure.

Sentiment Scores (optional): Derived sentiment using libraries like TextBlob.

Named Entity Recognition (optional): Extracted people, places, and organizations to assess entity-based bias.

8.Model Building

Multiple classification models were trained and evaluated:

Baseline Models:

Logistic Regression

Multinomial Naïve Bayes

Support Vector Machine (SVM)

Ensemble Models:

Random Forest

Gradient Boosting (e.g., XGBoost)

Neural Networks (optional):

Simple LSTM or BERT (if deep learning was used)

Training Approach:

80/20 Train-Test Split

5-Fold Cross-Validation

Hyperparameter tuning with GridSearchCV

9.Visualization of Results & Model Insights

To evaluate and explain the models, several visualizations were created:

Confusion Matrix: To compare true vs. predicted labels.

ROC-AUC Curve: To measure classification threshold performance.

Precision-Recall Curve: To assess performance in imbalanced scenarios.

Feature Importance: Displayed top words or tokens influencing model decisions.

Word-Level Heatmaps (for advanced NLP models): Showed attention on specific words (if using transformer-based models).

10.Tools and Technologies Used

Programming Language: Python

Libraries:NLP: NLTK, spaCy, TextBlob

Modeling: scikit-learn, XGBoost

Visualization: matplotlib, seaborn, WordCloud

Data Handling: pandas, NumPy

Platforms: Google Colab, Jupyter Notebook

Version Control: Git/GitHub

11.Team Members and Contributions

S.Divya– NLP pipeline, EDA, and TF-IDF model

S.Divya– Data preprocessing, visualizations, and Naïve Bayes model

S.Divyajothi–Deep learning models (LSTM, BERT) and evaluation

D.Divyashri– Report writing, presentation design, and deployment setup