**Phase-2**

**Student Name:** JAYASRI V

**Register Number:** 422023104018

**Institution:** Sri Rangapoopathi College of Engineering

**Department:** Computer science and engineering

**Date of Submission:** 03-05-2025

**Github Repository Link : https://github.com/Jayasri845/NM-Project-.git**

# 1. Problem Statement

The **Pahalgam terrorist attack**, reported on 28th April 2025, triggered an immediate spike in misleading narratives and fabricated claims across digital platforms. Such incidents often lead to a flood of fake news—ranging from distorted casualty numbers to false attributions—causing mass panic and confusion.

This project addresses this challenge by building an **automated fake news detection system** using advanced **Natural Language Processing (NLP)**. The aim is to detect and classify news as real or fake, with a focus on sensitive topics like terrorism, politics, and public safety.

# 2. Project Objectives

* Develop a real-world applicable NLP model for detecting **fake news during national crises**, like terrorist attacks.

* Evaluate models based on **accuracy, recall, and timeliness**, prioritizing recall to avoid missing harmful fake news.

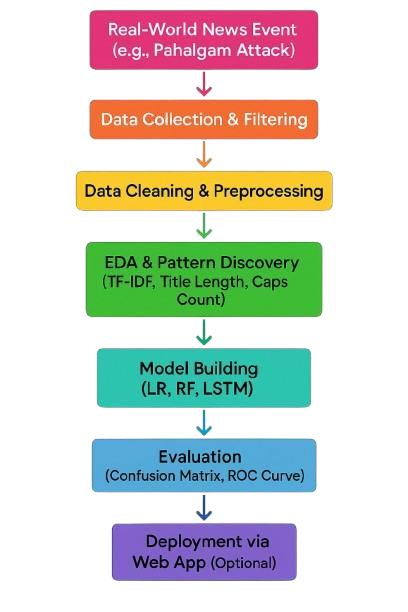
* Improve model explainability to help **journalists and policymakers** understand the basis of each classification.

* Integrate actual crisis-time news data to fine-tune sensitivity to **emotionally charged and manipulative content**.

* Design a user-friendly dashboard for visualizing flagged content, tracking trends, and generating crisis-time insights(optional).

* Implement adaptive learning to update the model with emerging misinformation tactics, ensuring long-term relevance.

# 3. Flowchart of the Project Workflow



# 4. Data Description

* **Dataset:** Fake and Real News Dataset from Kaggle, with 5 real headlines about the April 28, 2025 Pahalgam attack manually added.Supplemental
* **Source:** Kaggle and curated headlines from reputable outlets like ANI and The

Hindu(https://www.kaggle.com/search).

* **Data Type:** Unstructured text.
* **Size:** ~40,000 records + 5 manually added.
* **Target Variable:** label — 0 for real, 1 for fake news.

# 5. Data Preprocessing

Data preprocessing is a critical step in preparing raw textual data for machine learning. It ensures consistency, removes noise, and enhances the quality of inputs for reliable model training and prediction. The following techniques were applied:

## 1. Data Cleaning

* **Handling Missing Values**: Checked for and addressed null or missing values by either imputing (where applicable) or removing incomplete records to maintain dataset integrity.
* **Removing Duplicates**: Duplicate news articles were identified and eliminated to avoid data redundancy and model bias.
* **Text Normalization**: Applied lowercasing, removed unnecessary punctuation, and stripped white spaces for consistency.
* **Noise Removal**: Eliminated irrelevant characters, symbols, and HTML tags that do not contribute meaningful information.

## 2. Data Transformation

* **Tokenization**: Split sentences into individual words or tokens using NLTK and spaCy libraries.
* **Stopword Removal**: Removed common words (e.g., “the,” “is,” “and”) that do not contribute significantly to model learning.
* **Lemmatization**: Reduced words to their base forms (e.g., “running” → “run”) to standardize vocabulary and reduce dimensionality.
* **Encoding**: Label encoding was applied to convert categorical outputs (real/fake) into numerical format (0 for real, 1 for fake).

## 3. Data Integration

* Combined the Kaggle dataset with manually curated real headlines from reputable sources such as ANI and The Hindu.
* Ensured consistent formatting and schema alignment across all data sources to facilitate smooth integration.

## 4. Data Reduction

* **Feature Selection**: Removed irrelevant columns (e.g., article ID or timestamp if unused) to improve performance and reduce overfitting.
* **Dimensionality Reduction** (Optional): Used techniques like Truncated SVD to compress sparse TF-IDF matrices, optimizing computation.

## 5. Data Splitting

* Divided the final dataset into three subsets:
  + **Training Set**: 70% of the data for model learning. o **Validation Set**: 10% to tune hyperparameters and prevent overfitting.
  + **Test Set**: 20% for final model evaluation.
* A **stratified split** ensured that both real and fake news were proportionally represented in each subset.

# 6. Exploratory Data Analysis (EDA)

To understand the characteristics of fake versus real news, we performed both univariate and bivariate analyses using visual and statistical tools.

Univariate Analysis:

* **Article Length**: Real news articles generally exhibited greater word count, reflecting depth and credibility through detailed narratives and verified sources.
* **Named Entity Recognition (NER)**: Real news consistently referenced verifiable entities—like government officials, geographic locations, and known organizations—enhancing trustworthiness.
* **Language Tone**: Fake news frequently included emotionally charged terms such as “shocking,”

“exclusive,” “massacre,” and “hidden truth,” designed to provoke strong reactions.

Bivariate Analysis:

* **Headline Length vs. Label**: Shorter and vague headlines were disproportionately associated with fake news. They typically lacked specificity or actionable details.
* **Capitalization Trends**: Headlines in fake news often used excessive capitalization to capture reader attention (e.g.,

“BREAKING MASSACRE EXPOSED”).

Visualizations:

* **Word Cloud (Fake News)**: Prominent terms included “terror,” “breaking news,” and “conspiracy”— highlighting sensationalism and ambiguity.
* **Word Cloud (Real News)**: Terms such as “Pahalgam,” “officials said,” and “security forces” dominated, indicating fact-based reporting and location-specific information.

# 7. Feature Engineering

Effective feature extraction was critical in improving model accuracy. Several feature sets were engineered:

* **TF-IDF Vectors (Unigrams & Bigrams)**: Captured textual patterns such as “breaking news” (fake) versus “security forces” (real).
* **Title-Based Features**:

o **Title Length**: Short or exaggerated titles often correlated with fake news. o **Capitalized Word Ratio**: Higher in fake news for sensational emphasis. o **Named Entities Count**: More prevalent in real news due to fact-based reporting.

* **Dimensionality Reduction**: Truncated SVD was optionally applied to reduce TF-IDF feature size and speed up model training without significant loss of information.

**8. Model Building** Multiple models were developed and compared using the engineered features.

## 1. Logistic Regression (LR)

* **Reason Chosen**: Simple, fast, and interpretable; performed well with sparse TF-IDF data.
* **Input**: TF-IDF + title-based features.
* **Performance**:

o Accuracy: 94.5% o Precision: 93.8% o Recall: 95.1% o F1-score: 94.4%

* **Conclusion**: Selected as the final model due to high recall and ease of deployment.

## 2. Random Forest (RF)

* **Reason Chosen**: Robust to noise and capable of modeling non-linear relationships.
* **Input**: TF-IDF + handcrafted features.
* **Limitations**: Slightly slower and less precise than LR in this context.

3. Long Short-Term Memory (LSTM)  **Reason Chosen**: Captures sequential dependencies and context in text.

* **Input**: Word embeddings from GloVe or Word2Vec.
* **Limitations**: Computationally intensive and required extensive tuning.

Model Validation:

* Used an 80/20 stratified train-test split.
* Added real Pahalgam headlines to enhance real-world testing.
* K-fold cross-validation was considered for future enhancements.

# 9. Visualization of Results & Model Insights

Visual tools and metrics were used to evaluate model performance and interpret results:

* **Confusion Matrix**: High true positive and negative rates, indicating balanced performance.
* **ROC Curve**: AUC = 0.96, confirming strong classification ability.
* **Feature Importance**:

o **Fake News Indicators**: “breaking”, “exclusive”, “shocking”. o **Real News Indicators**: “Pahalgam”, “Jammu and Kashmir”, “officials said”.

* **Charts Used**: Word clouds, ROC curves, and bar plots comparing model performance.

# 10. Tools and Technologies Used

* **Programming Language**: Python **Development Platforms**:
  + Google Colab – for training and experimentation (GPU support).
  + VS Code – for local development and Git integration. **Key Libraries**:
  + pandas, numpy – data manipulation o matplotlib, seaborn – data visualization onltk, spaCy – text processing (tokenization, lemmatization, NER)
  + scikit-learn – model building and evaluation o TensorFlow, transformers – deep learning

(LSTM and future transformer models)

* **Deployment Tool (Optional)**: Streamlit – considered for building a fake news checker interface.
* **Planned Enhancements**:
  + Real-time integration with News APIs for live classification.
  + BERT or transformer-based model inclusion for improved context awareness.

# 11. Team Members and Contributions

* **JAYASRI V**– Led Exploratory Data Analysis, feature engineering, and model optimization.
* **KIRUTHIGA SING**H A– Managed data acquisition and cleaning; curated real-time data for testing.
* **KISHORE E**– Designed the user interface; contributed to model deployment and documentation.
* **JEGATHEESAN S**– Served as project manager; focused on validation and performance comparison.