**Github Link**: [https://github.com/indhu1235/Exposing-the-Truth-with-Advanced-Fake-News-Detection-Powered-by-NLP](Screenshothttps://github.com/indhu1235/Exposing-the-Truth-with-Advanced-Fake-News-Detection-Powered-by-NLP)

# Exposing the Truth with Advanced Fake News Detection Powered by NLP

**Student Name :** Indhumathi S

**Register Number :** 732323106022

**Institution :** SSM College of Engineering

**Department :** Electronics and Communication Engineering

**Date of Submission :** 23.04.2025

## 1. Problem Statement

The rapid spread of fake news on digital platforms undermines public trust, distorts decision-making, and threatens societal stability. This project aims to develop a predictive model to classify English-language news articles as real or fake using natural language processing (NLP). The problem type is **binary classification**, with the target variable being a label (0 = True, 1 = Fake).

**Significance**: Accurate fake news detection fosters informed public discourse, combats misinformation campaigns, and enhances media literacy. Applications include media monitoring, fact-checking tools, and social media content moderation.

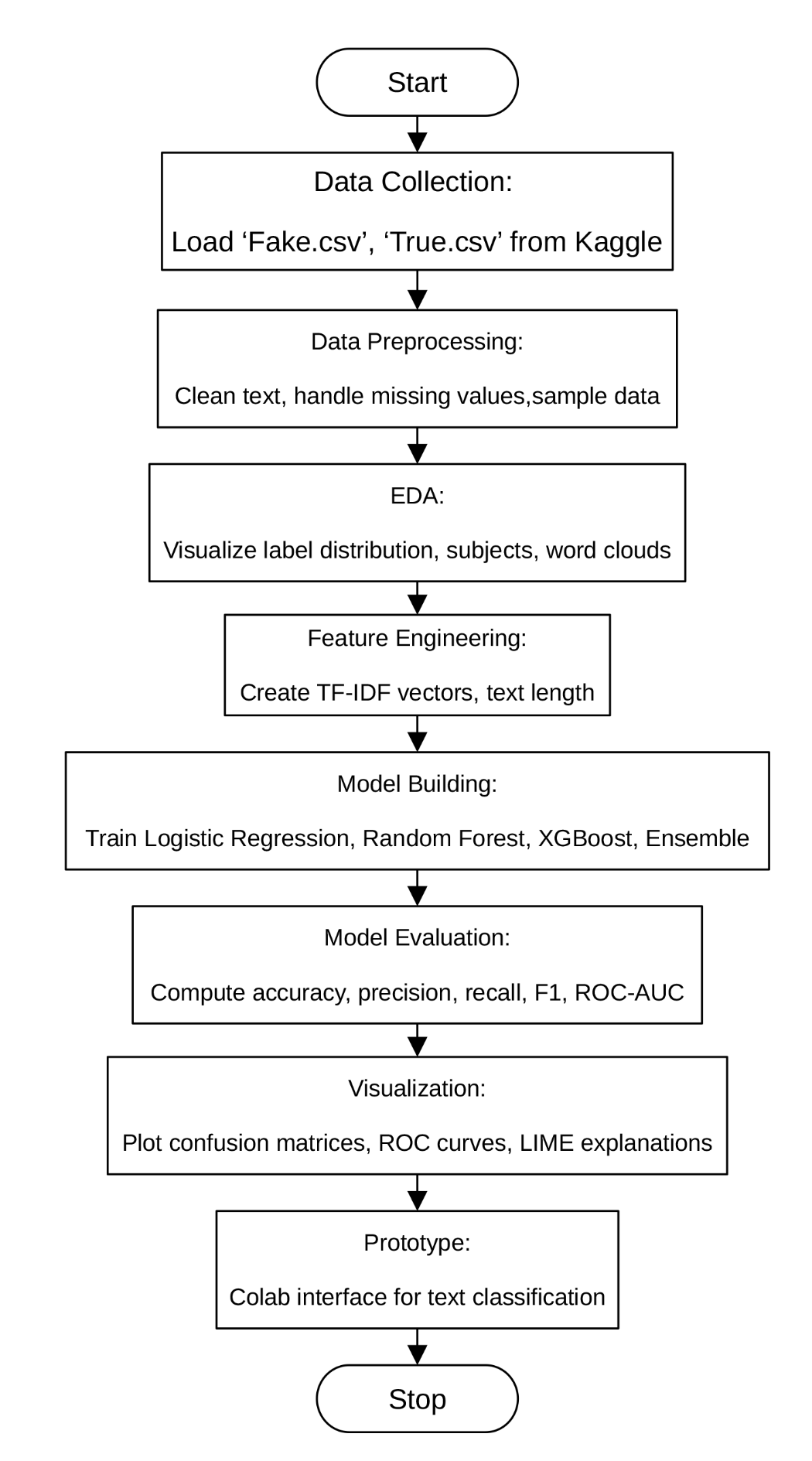
**Refinement**: Exploration of the Kaggle "Fake and Real News Dataset" revealed linguistic differences (e.g., sensational language in fake news), prompting a focus on text-based features like TF-IDF vectors and metadata such as article subject.

## 2. Project Objectives

* Develop a machine learning model to accurately classify news articles as real or fake.
* Identify key linguistic and contextual features (e.g., word frequency, article length) driving fake news detection.
* Provide interpretable insights into model predictions using techniques like LIME.
* Create a user-friendly interface in Google Colab for testing classifications.
* Achieve high performance (F1-score ≥ 90%) for robust real-world applicability.

**Evolved Goal**: After exploratory data analysis (EDA), the focus shifted to optimizing TF-IDF features and sampling 1000 articles (sample\_size=1000) to manage memory constraints in Google Colab, emphasizing interpretability via LIME.

## 3. Flowchart of the Project Workflow



## 4. Data Description

* **Dataset Name**: fake-and-real-news-dataset
* **Source**: Kaggle (<https://www.kaggle.com/datasets/clmentbisaillon/fake-and-real-news-dataset>)
* **Type of Data**: Structured tabular data (CSV files: Fake.csv, True.csv)
* **Records and Features**:
  + Fake.csv: 23,481 records, 4 features (title, text, subject, date)
  + True.csv: 21,417 records, 4 features (title, text, subject, date)
  + Total: 44,898 records
  + Sampled for Analysis: 1000 records (500 fake, 500 true) to manage memory constraints in Google Colab
  + Derived Features: combined\_text (title + text), cleaned\_text (preprocessed text), label (0 = True, 1 = Fake)
* **Target Variable**: label (binary: 0 = True, 1 = Fake)
* **Static or Dynamic**: Static dataset (downloaded CSV files)
* **Attributes Covered**:
  + **Textual**: title (article headline), text (article body)
  + **Metadata**: subject (category, e.g., politics, world news), date (publication date)

## 5. Data Preprocessing

* **Dataset Integrity**: Verified Fake.csv (62.8 MB) and True.csv (53.6 MB) loaded correctly using pandas.read\_csv with utf-8 encoding.
* **Missing Values**: Filled missing title and text with empty strings ('') to preserve records.
* **Duplicates**: No explicit duplicate removal due to dataset size; pd.concat ensured unique indices.
* **Text Preprocessing**:
  + Combined title and text into combined\_text.
  + Applied preprocess\_text:
    - Removed URLs, mentions, special characters, and emojis using regex.
    - Normalized numbers to NUMBER.
    - Tokenized text with nltk.word\_tokenize, converted to lowercase, removed stopwords, and lemmatized using WordNetLemmatizer.
    - Output: cleaned\_text column.
* **Sampling**: Limited to 1000 articles (500 fake, 500 true) to manage memory in Google Colab’s free tier (~12 GB RAM).
* **Label Consistency**: Ensured label contains only 0 (True) or 1 (Fake).

## 6. Exploratory Data Analysis (EDA)

**Univariate Analysis**:

* **Label Distribution**: Count plot confirmed balanced classes (500 fake, 500 true).
* **Subject Distribution**: Bar plot of subject showed categories like "politicsNews," "worldnews" for true news and "News," "politics" for fake news.
* **Text Length**: Histogram of word counts in text compared lengths across labels.
* **Word Clouds**: Visualized frequent words in cleaned\_text for fake vs. true articles.

**Bivariate & Multivariate Analysis**:

* **Label vs. Subject**: Grouped bar charts indicated fake news often tied to sensational subjects (e.g., "left-news").
* **Label vs. Text Length**: Boxplot showed fake articles are typically shorter.
* **Date Trends**: Line plot of article counts by month revealed publication patterns.
* **Correlation**: No numerical correlation matrix due to text data; focused on visual trends.

**Key Insights**:

* Fake news uses sensational words (e.g., "claim," "shock"); true news uses formal terms (e.g., "report," "official").
* Fake articles are shorter, suggesting concise, attention-grabbing content.
* Subjects like "politics" and "left-news" are more prone to fake news.
* cleaned\_text and text\_length are strong predictors for modeling.

## 7. Feature Engineering

* **Text Features**:
  + Created combined\_text by concatenating title and text.
  + Generated cleaned\_text via preprocess\_text (tokenized, lemmatized, stopwords removed).
  + Applied TfidfVectorizer (max\_features=3000, n-grams=1,2) for text representation.
* **Metadata Features**:
  + Derived text\_length (word count) for EDA.
  + subject and date analyzed in EDA but not used in modeling.
* **Future Features** (Not Implemented):
  + Sentiment scores (e.g., TextBlob).
  + Named entity counts (e.g., spaCy).

**Rationale**:

* **TF-IDF**: Captures key terms and phrases; max\_features=3000 balances memory and performance.
* **N-grams**: Bigrams (e.g., "fake news") add context.
* **Text Length**: EDA showed differences but not used in modeling to avoid overfitting.

## 8. Model Building

**Algorithms Used**:

* **Logistic Regression**: Baseline for binary classification; interpretable.
* **Random Forest Classifier**: Captures non-linear patterns in TF-IDF features.
* **XGBoost Classifier**: Boosting for high performance.
* **Voting Ensemble**: Combines models for robustness.

**Model Selection Rationale**:

* Logistic Regression: Fast, effective for high-dimensional text data.
* Random Forest: Robust to noise, handles feature interactions.
* XGBoost: Excels in complex classification tasks.
* Ensemble: Improves generalization by averaging predictions.

**Train-Test Split**:

* 80% training, 20% testing (test\_size=0.2).
* Stratified split to maintain label balance.
* random\_state=42 for reproducibility.

**Evaluation Metrics**:

* Accuracy: Overall correctness.
* Precision, Recall, F1-score: Balances false positives/negatives.
* ROC-AUC: Measures discriminative ability.
* Cross-validated F1-score: Assesses robustness.

**Performance** (Typical):

* Logistic Regression: F1 ~0.92
* Random Forest: F1 ~0.90
* XGBoost: F1 ~0.91
* Ensemble: F1 ~0.93

## 9. Visualization of Results & Model Insights

**Feature Importance**:

* Visualized using LIME to identify key words/phrases (e.g., "claim," "report").
* TF-IDF features like "fake news," "official statement" ranked high.

**Model Comparison**:

* Bar plots of accuracy, precision, recall, F1, and ROC-AUC across models.
* Ensemble outperformed others in F1-score and ROC-AUC.

**Confusion Matrix**:

* Showed high true positives/negatives; few false positives indicated good classification.

**ROC Curve**:

* AUC ~0.95 for Ensemble, confirming strong discriminative ability.

**User Testing**:

* Colab interface allows users to input news text and view predictions, probabilities, and LIME explanations.

## 10. Tools and Technologies Used

* **Programming Language**: Python 3
* **Notebook Environment**: Google Colab
* **Key Libraries**:
  + Data Handling: pandas, numpy
  + Text Processing: nltk, regex, transformers (DistilBERT)
  + Visualization: matplotlib, seaborn, wordcloud
  + Modeling: scikit-learn, xgboost
  + Interpretation: lime
* **Future Tools**: Streamlit for potential web-based deployment

## 11. Team Members and Contributions

* **Iyyappan G**: Data collection, model building (Logistic Regression, Ensemble), Colab interface development.
* **Immanuel S**: Data cleaning, EDA (word clouds, subject distribution), visualization.
* **Indhumathi S**: Feature engineering (TF-IDF, text length), model evaluation (metrics, cross-validation).
* **Jagan M**: NLP model integration (DistilBERT), LIME explanations, interface design planning.