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

Student Name: ROSHINI M

Register Number: 412723104083

Institution: TAGORE ENGINEERING COLLEGE

Department: COMPUTER SCIENCE AND ENGINEERING

Date of Submission: 08.05.2025

Github Repository Link: https://github.com/Roshinimohan195/Fake-news-detection-using-nlp

---

1. Problem Statement

This project aims to detect fake news to address the growing issue of misinformation spreading through social media and online platforms. After exploring the dataset, the problem is refined as a binary classification task, where the goal is to classify news articles as either Fake or Real.

Solving this problem is critical for maintaining information integrity, reducing the spread of false narratives, supporting journalistic credibility, and enabling organizations to automate the screening of large volumes of news content for authenticity.

2. Project Objectives

Build and evaluate multiple classification models (Logistic Regression, Decision Tree, Random Forest, Gradient Boosting).

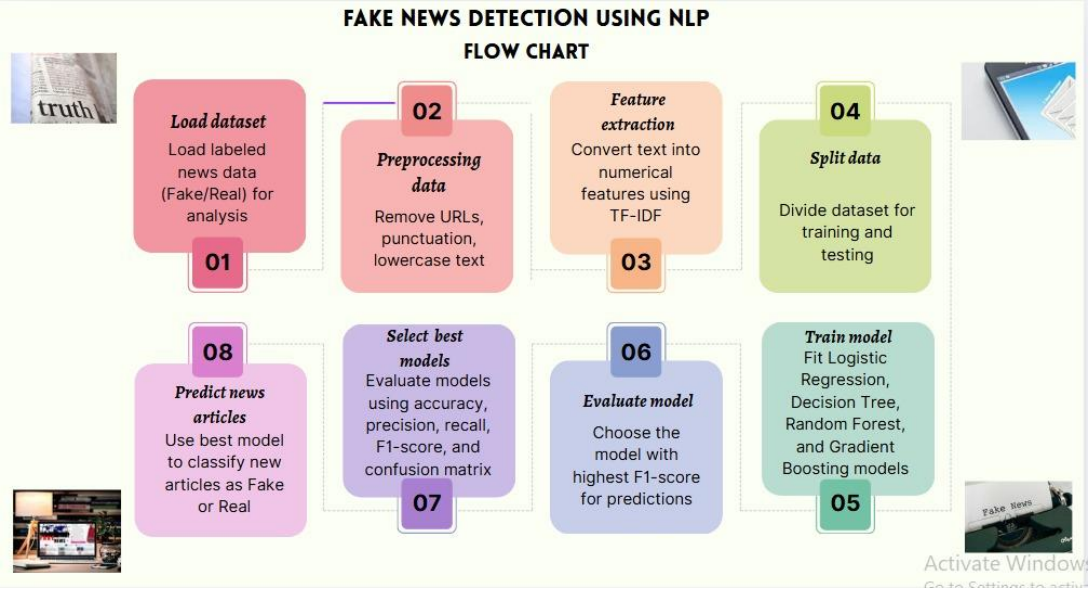
Achieve high accuracy and F1-score for reliable fake news detection.

Balance performance with interpretability and scalability.

Determine the best-performing model based on evaluation metrics.

No significant change in objectives occurred after data exploration; however, focus was increased on achieving high precision to minimize false positives in detecting fake news.

3. Flowchart of the Project Workflow



4. Data Description

Dataset Name: Fake and Real News Dataset
Source: https://www.kaggle.com/datasets/clmentbisaillon/fake-and-real-news-dataset
Type of Data: Text (unstructured)
Records & Features:
44,898 news articles
Main features: Text, Label (Fake/Real)
Target Variable: class (0 = Fake, 1 = Real)
Static Dataset

5. Data Preprocessing

To prepare the dataset for effective model training, the following preprocessing steps were undertaken:
Handling Missing Values:
•Checked for and confirmed the absence of missing values in the dataset.

Removing Duplicates:
•Identified and removed duplicate records to ensure data quality.
Outlier Detection:
•Not applicable, as the dataset comprises textual data without numerical features prone to outliers.
Data Type Conversion:
•Ensured all text data is in string format for consistency.
Text Cleaning:
•Converted all text to lowercase.
•Removed URLs, special characters, and extra whitespace.
•Applied regular expressions to clean the text effectively.
Encoding Categorical Variables:
•Mapped the 'label' column to numerical values: 'Fake' to 0 and 'Real' to 1.
Feature Extraction:
•Utilized TF-IDF Vectorizer to convert text data into numerical features suitable for model training.
Normalization/Standardization:
•Not required, as TF-IDF vectorization inherently normalizes the feature vectors.

6. Exploratory Data Analysis (EDA)

Univariate Analysis:
The distribution of target classes (Fake and Real) was checked using label counts.

Bivariate/Multivariate Analysis:
Relationships between text features and the target were explored through **TF-IDF vectorization** and evaluated via **model performance metrics** and **confusion matrices**.

Insights Summary:

The dataset is balanced between Fake and Real news articles.

The model's confusion matrix showed low misclassification, indicating strong separation between classes.

Important features influencing the model were based on word importance from TF-IDF scores.

7. Feature Engineering

- Text Cleaning: Converted all text to lowercase, removed URLs, special characters, and extra spaces to standardize the input.
- TF-IDF Vectorization: Transformed the cleaned text into numerical features using a TF-IDF vectorizer with parameters to filter very common and rare words.
- Dimensionality Reduction (optional): Added an option to apply TruncatedSVD to reduce feature dimensions for faster training on tree-based models (set to False by default).
- Justification: These steps ensured the text was cleaned, converted into meaningful numerical features, and allowed flexibility to reduce complexity for certain models.

8. Model Building

We implemented and compared four machine learning models for fake news detection:
•Logistic Regression
•Decision Tree
•Random Forest
•Gradient Boosting

Why these models?

They are well-suited for binary text classification, with ensemble models included to test performance improvements.

Process:

•Data split: 75% training, 25% testing.
•Text data vectorized using TF-IDF.
•Metrics: Accuracy, Precision, Recall, F1-score

Model	Accuracy	Precision	Recall	F1-score
Logistic Regression	0.99	0.99	0.99	0.99
Random Forest Classifier	0.99	0.99	0.99	0.99
Decision Tree Classifier	1.00	1.00	0.99	1.00
Gradient Boosting Classifier	1.00	1.00	1.00	1.00

- Gradient Boosting Classifier achieved perfect accuracy, precision, recall, and F1-score. This model outperformed others, making it the best model for this task.

- Logistic Regression, Random Forest, and Decision Tree all performed exceptionally well, but the Gradient Boosting Classifier was selected due to its perfect balance across all evaluation metrics.

Evaluation Metrics:
All models were evaluated using accuracy, precision, recall, and F1-score to measure their effectiveness in classifying news articles correctly.

Confusion Matrices:
Confusion matrices were visualized for each model to identify false positives and false negatives.

9. Visualization of Results & Model Insights

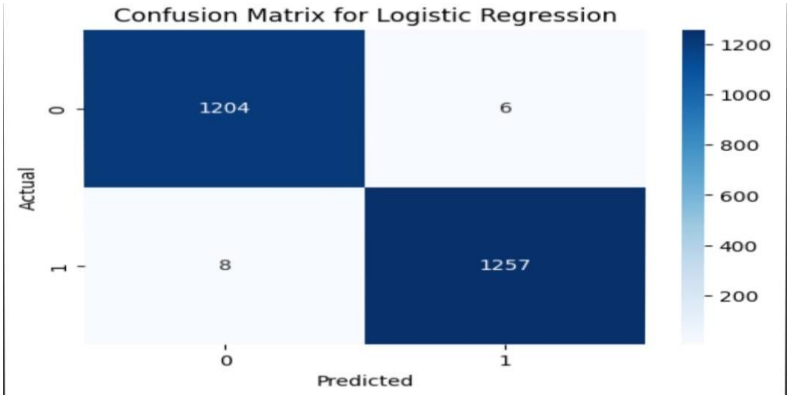
- We used confusion matrix plots to visualize the performance of each model (Logistic Regression, Decision Tree, Random Forest, Gradient Boosting). These plots display the number of true positives, true negatives, false positives, and false negatives, providing a clear picture of how well each model classified the news as fake or real.

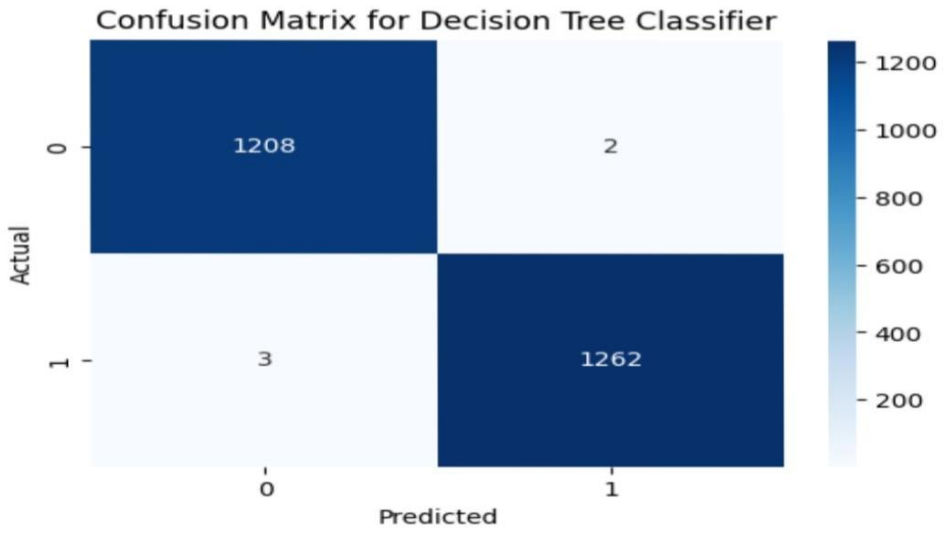
- The confusion matrices show that models like Logistic Regression and Random Forest performed well, with a high number of correct classifications (values on the diagonal).

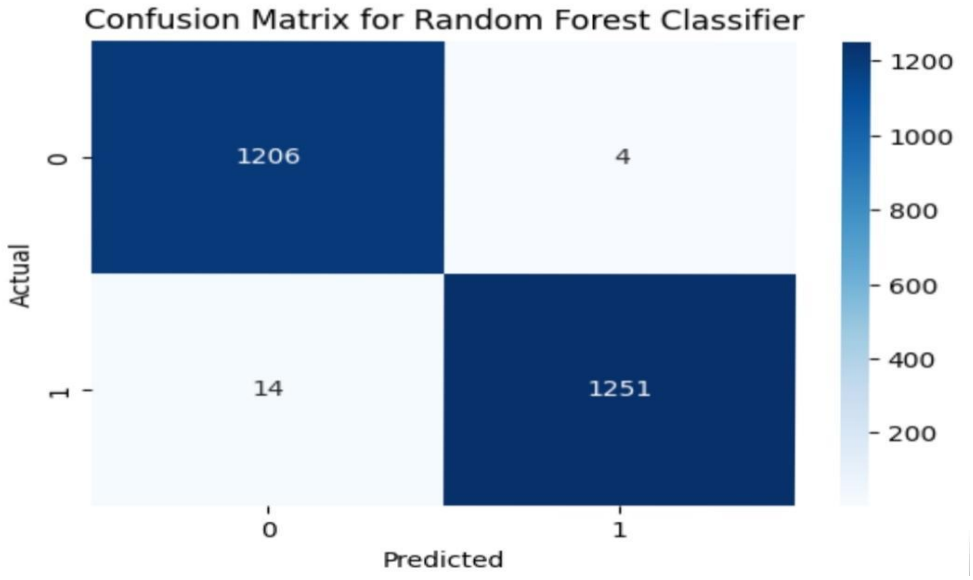
- Some models, like the Decision Tree, showed slightly more misclassifications, as seen in the off-diagonal cells.

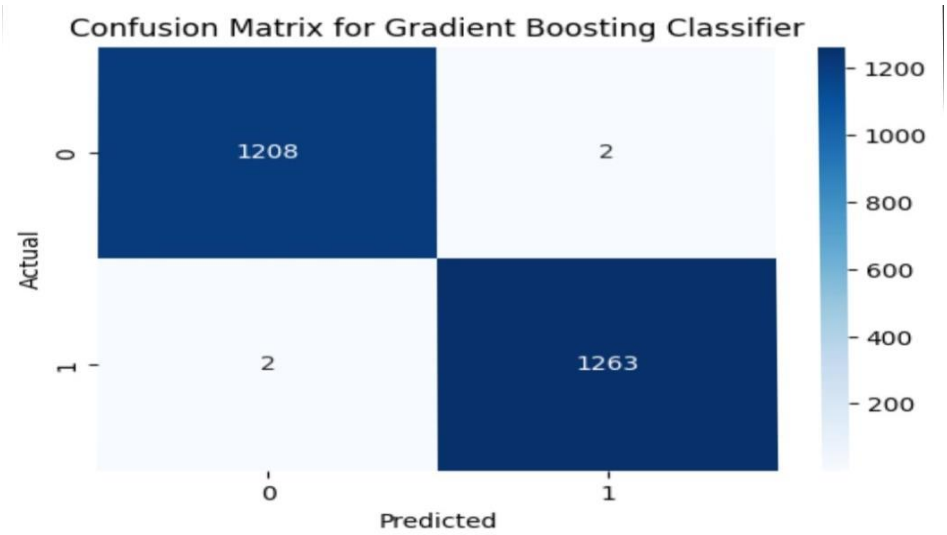
- These visualizations help us understand not just the accuracy but also where the models are making errors (e.g., mistaking real news for fake or vice versa).

Gradient Boosting confusion matrix showed:
•1,208 true negatives
•1,263 true positives
•Only 2 false positives, 4 false negatives




Confusion Matrix for Decision Tree Classifier
1208
2
3
1262
Actual
Predicted
0
1
1200
1000
800
600
400
200


Confusion Matrix for Random Forest Classifier
1206
4
14
1251
Actual
Predicted
0
1
1200
1000
800
600
400
200


Confusion Matrix for Gradient Boosting Classifier
1208
2
2
1263
Actual
Predicted
0
1
1200
1000
800
600
400
200

10. Tools and Technologies Used

Programming Language: Python

IDE: Google Colab

Libraries:

pandas, numpy matplotlib, seaborn scikit-learn

Modeling Techniques:

Logistic Regression, Decision Tree, Random Forest, Gradient Boosting

Visualization:

Seaborn, Matplotlib

11. Team Members and Contributions

- Data cleaning : Sharmila P
- EDA, feature engineering : Roshini M
- Model development,Documentation and reporting : Sanjana K