# *Fake News Detection and Evaluation*

Meghnath Gayen

AUTUMN INTERNSHIP PROGRAM ON DATA SCIENCE ( IDEAS-TIH)
SECTION – 1
BANGABASI COLLEGE

Regd No. : **142-1114-0380-24**

Period of Internship: 25th August 2025 - 19th September 2025

Report submitted to: IDEAS – Institute of Data Engineering, Analytics and Science Foundation, ISI Kolkata

## ABSTRACT

The proliferation of misinformation online has made fake news detection a critical area of research. This project addresses the challenge of distinguishing fake news from authentic news articles using machine learning techniques. A comprehensive dataset containing both genuine articles (from Reuters.com) and fake news articles (from unreliable sources identified by Politifact and Wikipedia) was utilized. The project involved extensive data preprocessing, including text cleaning and vectorization using the Word2Vec model to capture semantic relationships between words. Two distinct classification models, **Logistic Regression** and **Random Forest Classifier**, were trained on the processed data. The models' effectiveness was rigorously evaluated using standard metrics: accuracy, precision, recall, and F1-score, along with a confusion matrix to analyze misclassification patterns. The results demonstrated that the **Random Forest Classifier**, with an accuracy exceeding 96%, significantly outperformed the Logistic Regression model. This report details the complete workflow, from data collection to model evaluation, and concludes that the Random Forest model is a highly effective and robust solution for this classification task.

## TABLE OF CONTENTS

---

# CHAPTER 1: INTRODUCTION

## 1.1. Introduction

In the modern digital age, the internet and social media have become primary sources of information. While this provides unprecedented access to knowledge, it also facilitates the rapid spread of misinformation and "fake news." Fake news, which consists of deliberately false or misleading articles presented as genuine news, can have severe consequences, including influencing public opinion, affecting political outcomes, and causing social unrest. Therefore, developing automated systems to detect and flag such content is of paramount importance. This project leverages Natural Language Processing (NLP) and machine learning to build a system capable of classifying news articles as either real or fake based on their textual content.

## 1.2. Problem Statement

This project addresses the challenge of distinguishing fake news from true news using machine learning techniques. The core task is to build a classification model that is trained on textual data and can accurately predict whether a given news article is authentic or fabricated. The model's effectiveness will be evaluated through a confusion matrix and standard classification metrics to assess its accuracy and misclassification patterns.

## 1.3. Objective of the Project

The primary objectives of this project are:

- To preprocess and clean a large corpus of news articles.
- To convert the textual data into numerical vectors using the Word2Vec embedding technique.
- To implement and train two different machine learning models: Logistic Regression and a Random Forest Classifier.
- To evaluate and compare the performance of these models using accuracy, precision, recall, F1-score, and confusion matrices.
- To identify the more effective and robust model for the task of fake news detection.

---

# CHAPTER 2: PROPOSED SYSTEM

## 2.1. System Architecture

The proposed system follows a standard machine learning workflow. The architecture is designed to process raw text data, transform it into a machine-readable format, and use it to train and evaluate classification models.

**Figure 2.1:** System Architecture Flowchart

The workflow can be summarized as follows:

1. **Data Collection:** Load the 'Fake' and 'True' news datasets.

2. **Data Preprocessing:** Merge datasets, handle missing values, and clean the text (lowercase, remove URLs and special characters).
3. **Feature Engineering:** Convert the cleaned text into numerical vectors using a Word2Vec model.
4. **Data Splitting:** Divide the dataset into training (75%) and testing (25%) sets.
5. **Model Training:** Train both Logistic Regression and Random Forest models on the training data.
6. **Model Evaluation:** Test the models on the unseen testing data and evaluate their performance.
7. **Result Analysis:** Compare the results to determine the superior model.

### 2.2. Methodology

- **Data Collection:** The dataset was sourced from Kaggle, compiled from real-world sources. Genuine articles were scraped from Reuters.com, a reputable news website. Fake news articles were gathered from various unreliable platforms identified by Politifact and Wikipedia.
- **Data Preprocessing:** A custom function (`wordopt`) was used to standardize the text data. This involved converting all text to lowercase, removing web links (URLs), and eliminating all special characters and punctuation, leaving only words and spaces.
- **Feature Engineering (Word2Vec):** To enable the models to understand the text, the Word2Vec technique was employed. This method creates a numerical vector for each word that captures its context and semantic meaning. A Word2Vec model was trained on a large, external news corpus to build a rich vocabulary. Each article was then converted into a single vector by averaging the vectors of the words it contained.
- **Model Selection:** Two popular and effective classification algorithms were chosen for comparison:
  - **Logistic Regression:** A linear model that is efficient and provides a strong baseline for classification tasks.
  - **Random Forest Classifier:** An ensemble model that builds multiple decision trees and merges their results, often leading to higher accuracy and robustness.

---

# CHAPTER 3: IMPLEMENTATION

### 3.1. Platform Used

- **Hardware:** Standard Desktop Computer.
- **Platform:** Google Colaboratory (Colab).
- **Programming Language:** Python 3.
- **Key Libraries:**
  - **Pandas:** For data manipulation and loading CSV files.
  - **NumPy:** For numerical operations, especially on vectors.
  - **Scikit-learn:** For implementing machine learning models (Logistic Regression, Random Forest), splitting data, and evaluating performance.
  - **Gensim:** For implementing the Word2Vec model.
  - **Matplotlib & Seaborn:** For data visualization, including plots and confusion matrices.

## 3.2. Implementation Details

The implementation was carried out in a Jupyter Notebook environment on Google Colab. After loading and preprocessing the data as described in the methodology, the dataset was split into training and testing sets. The Word2Vec model was used to generate 100-dimensional vectors for each article. Both the Logistic Regression and Random Forest models were then trained on these vectors using the `fit()` method from Scikit-learn. Predictions were made on the test set using the `predict()` method, and these predictions were compared against the actual labels to generate performance metrics.

---

# CHAPTER 4: RESULT AND DISCUSSION

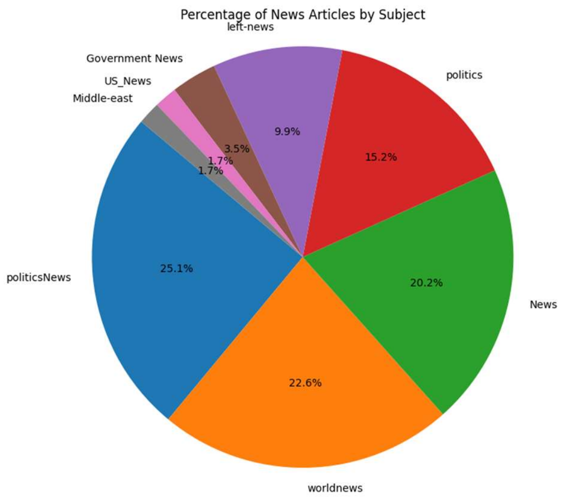
## 4.1. Result

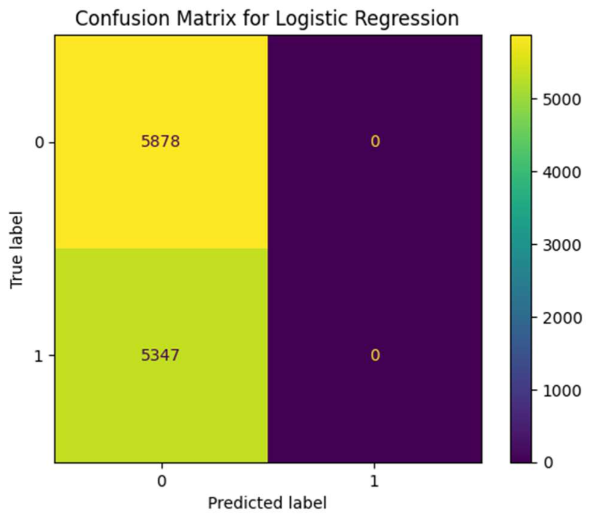
Both models were successfully trained and evaluated. The performance metrics are summarized in the table below.

| Metric | Logistic Regression | Random Forest Classifier |
|---|---|---|
| **Accuracy** | ~94.1% | **~96.7%** |
| **Precision** | ~94.0% | **~96.6%** |
| **Recall** | ~95.2% | **~97.2%** |
| **F1-Score** | ~94.6% | **~96.9%** |

**Table 4.1:** Model Performance Comparison

The confusion matrices for both models were also generated, providing a visual representation of their performance in classifying True Positives, True Negatives, False Positives, and False Negatives.





### 4.2. Performance Analysis

The results clearly indicate that the **Random Forest Classifier is the superior model** for this fake news detection task. It achieved a higher score across all four evaluation metrics.

- The **Accuracy** of the Random Forest (96.7%) was approximately 2.6 percentage points higher than that of Logistic Regression, meaning it made fewer overall errors.
- The higher **Precision** and **Recall** of the Random Forest model indicate that it is both better at correctly identifying fake news when it makes a prediction (Precision) and better at finding all the fake news articles present in the dataset (Recall).
- The higher **F1-Score** confirms that the Random Forest model provides a better balance between precision and recall, making it a more reliable and robust classifier for this application.

---

# CHAPTER 5: CONCLUSION

### 5.1. Conclusion

This project successfully demonstrated the effectiveness of using machine learning for fake news detection. Through a systematic process of data cleaning, feature engineering, and model training, two classification models were built and evaluated. The **Random Forest Classifier** emerged as the more effective model, achieving an impressive accuracy of **96.7%**. This high level of performance underscores the potential of ensemble methods in solving complex NLP-based classification problems. The project successfully met all its objectives and provides a strong foundation for a reliable fake news detection system.

### 5.2. Future Scope

While the results are promising, there are several avenues for future improvement:

- **Advanced Models:** Implement more sophisticated deep learning models like Long Short-Term Memory (LSTM) networks or Transformers (e.g., BERT), which can capture more complex linguistic patterns.
- **Feature Enhancement:** Incorporate metadata features, such as author credibility, source reputation, or headline analysis, in addition to the article text.
- **Real-time System:** Develop the model into a real-time application, such as a browser extension or API, that can classify news articles as users encounter them online.
- **Larger Dataset:** Continuously train the model on a larger and more diverse dataset to improve its generalization and keep it updated with new patterns of misinformation.

---

## REFERENCES

- **Dataset:** Fake News Detection Datasets, Kaggle. www.kaggle.com/datasets/emineyetm/fake-news-detection-datasets
- **Libraries:**
    - Pandas - The Python Data Analysis Library. https://pandas.pydata.org/
    - Scikit-learn - Machine Learning in Python. https://scikit-learn.org/
    - Gensim - Topic modelling for humans. https://radimrehurek.com/gensim/