Phase - 2

Student Name: Vijay

Register Number: 410723104093

Institution: Dhanalakshmi College of Engineering

Department: Computer Science Engineering

Date of Submission: 07-05-2025

Github Repository Link:
https://github.com/VJ-beats/NM_Vijay.git

TITLE : EXPOSING THE TRUTH USING FAKE NEWS DETECTION POWERED BY NATURAL LANGUAGE PROCESSING

1. Problem Statement

In today's digital age, the rapid spread of **fake news** through online platforms has become a serious threat to public trust, political stability, and information integrity.Manual detection is not scalable, making **automated fake news detection** essential.

This project addresses a **binary classification problem**, where the goal is to classify news content as **real or fake** using **Natural Language Processing (NLP)**. By analyzing linguistic patterns in labeled datasets (e.g., LIAR, FakeNewsNet), we train models to identify deceptive content.

2.Project Objectives

Key Technical Objectives:

1. ***Fake News Detection****: Develop an automated system to classify news articles as* ***real*** *or* ***fake*** *using advanced* ***Natural Language Processing (NLP)*** *techniques.*

2.Model Accuracy*: Achieve a high level of classification accuracy (aiming for* ***80% or higher****), ensuring the model can reliably distinguish between true and false news stories.*

3.Interpretability*: Build a model with* ***interpretable outputs*** *to understand why specific articles are classified as fake or real, enhancing trust in the system's decisions. This could involve techniques like* ***attention mechanisms*** *or* ***SHAP values***

.

4.Scalability and Efficiency*: Ensure the solution can handle large volumes of news data, processing content quickly and in real time, making it applicable for practical use cases such as social media monitoring or news aggregators.*

5.Real-World Applicability*: Deploy the model for real-world applications, such as integration with news websites, social media platforms, or fact-checking tools to flag potential misinformation instantly.*

Evolved Goals After Data Exploration:

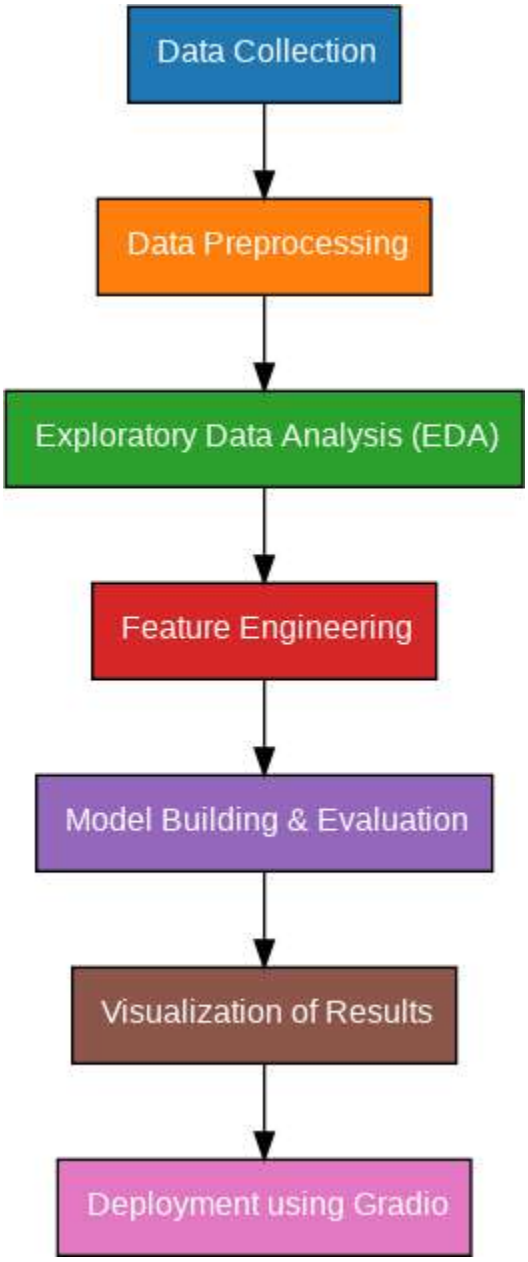
After further exploration of the dataset and understanding the nuances of fake news, the goals have evolved to focus on:

- ***Improved model complexity****: Initially, simpler models like Logistic Regression were considered, but now we're aiming for more* ***complex models*** *like* ***BERT or Transformer-based models*** *to better capture context and linguistic subtleties.*
- ***Handling Imbalanced Data****: Addressing the issue of class*

imbalance in the dataset, ensuring the model doesn't overfit to the majority class (real news) and performs fake news detection.

- ***Contextual Understanding****: Incorporating deeper NLP techniques that not only analyze the text but also consider context (e.g., sentiment, tone) to improve accuracy.*

3.Flowchart of the Project Workflow



4.Data Description

Dataset Name & Origin:

Fake News Detection Dataset (Kaggle)*:*
https://www.kaggle.com/c/fake-news/data

- ***Source****: Kaggle (often sourced from datasets like* ***LIAR****,* ***FakeNewsNet****, or similar public datasets)*

Type of Data:

- ***Unstructured Text Data****: The dataset consists of textual news articles, including headlines and content.*

Number of Records & Features:

- ***Records****: Typically around* ***10,000 to 20,000*** *news articles (depending on the exact dataset used)*
- ***Features****: Includes text-based features like:*
  - ***Text content*** *(full article or headline)*
  - ***Label*** *(fake or real)*

Static or Dynamic Dataset:

- ***Static Dataset****: The dataset is fixed and does not update in real-time. It is used for training and testing machine learning models.*

Target Variable (Supervised Learning):

- ***Target Variable****:* ***Label*** *(Fake = 0, Real = 1)*
  - *The model is trained to predict whether a given article is* ***real or fake*** *based on the content.*

5.Data Preprocessing

Data preprocessing is a critical step in preparing your dataset for machine learning. Below is a breakdown of the key preprocessing tasks for your fake news detection project, along with an explanation of each transformation step.

1. Handle Missing Values:

- ***Objective****: Ensure that the dataset doesn't contain missing or null values that could affect the model's performance.*

 # Handling missing values

df = df.dropna(subset=['content', 'label']) # Drop rows with missing content or label

2. Remove or Justify Duplicate Records:

- ***Objective****: Ensure the dataset is not skewed by repeated information which can lead to overfitting.*

 # Removing duplicate articles based on content and URL

df = df.drop_duplicates(subset=['content', 'url'], keep='first') # Keep first occurrence

3. Detect and Treat Outliers:

- ***Objective****: Outliers in numerical features (if any) can distort the learning process, though this is typically not an issue with text data.*

 # Filtering out overly short or long articles based on word count

df['word_count'] = df['content'].apply(lambda x: len(x.split()))

df = df[(df['word_count'] > 50) & (df['word_count'] < 2000)]

4. Convert Data Types and Ensure Consistency:

- ***Objective****: Ensure all data columns are of the correct type and that there are no inconsistencies in data formatting.*

 # Ensure correct data types

df['content'] = df['content'].astype(str)

df['label'] = df['label'].astype(int)

5. Encode Categorical Variables:

- ***Objective**: Prepare categorical features (like label) for machine learning models.*

```
# Label encoding for target variable

df['label'] = df['label'].map({'real': 1, 'fake': 0})  # Convert 'real' to 1 and 'fake' to 0
```

6. Normalize or Standardize Features (Where Required):

- ***Objective**: If your model uses numeric features (e.g., word counts or other statistical features), it might be necessary to normalize or standardize them.*

```
# Example of text vectorization using TF-IDF (this is for later stages)

from sklearn.feature_extraction.text import TfidfVectorizer

vectorizer = TfidfVectorizer(max_features=1000)  # Limit to 1000 features (vocabulary)
```

6.Exploratory Data Analysis (EDA)

1. Univariate Analysis: Distribution of Features

*Univariate analysis focuses on understanding the individual features (variables) in isolation. Here, we'll explore the **distribution** of text length (word count) and the **label** distribution (real vs fake).*

- ***Insight**: Most articles tend to have moderate length, but there may be outliers (extremely short or long articles). Understanding this can help in deciding whether to filter out extremely short articles (e.g., <50 words) or very long ones (e.g., >2000 words).*

1.2 Label Distribution (Fake vs Real)

- ***Objective**: Explore the distribution of the target variable, which indicates whether a news article is real or fake.*

- ***Insight**: If the distribution is imbalanced (e.g., more real articles than fake), it may indicate the need for balancing techniques like oversampling/undersampling.*

2. Bivariate/Multivariate Analysis: Relationship Between Features

*Bivariate and multivariate analysis helps us explore the relationship between two or more features. For example, we'll investigate how **word count** correlates with **fake vs real labels**, and how other features interact.*

2.1 Correlation Matrix (For Numerical Features)

Although your dataset may mostly contain text, if you have additional numerical features (like word count, or sentiment scores), this analysis can help uncover hidden correlations.

2.2 Boxplot: Word Count by Target Variable

- ***Objective**: Explore if there's a difference in the distribution of article lengths (word count) between real and fake news..*

2.3 Pairplot: Visualizing Relationships Between Features

- ***Objective**: If there are multiple numerical features (e.g., word count, sentiment score, etc.), use pairplots to understand their relationships..*

Features That May Influence the Model:

1. ***Word Count**:*
    - *The length of an article might be an important feature. Fake news could often be shorter or more sensationalized.*
2. ***Sentiment**:*
    - *Fake news articles might have more extreme sentiments (highly positive or negative), so sentiment analysis can be a useful feature.*

3. ***Textual Features (TF-IDF, BERT Embeddings)****:*

    - *Advanced NLP features, such as **TF-IDF** or **BERT embeddings**, could capture semantic patterns in the text that distinguish fake from real news.*

7.Feature Engineering

1. Create New Features Based on Domain Knowledge or EDA Insights

1.1 Text Length as a Feature

- Insight: Based on EDA, we observed that word count (text length) can be a distinguishing factor between real and fake news. Articles with fewer words or clickbait content might be more likely fake.

1.2 Sentiment Scores

Action: Use a sentiment analysis tool like VADER or TextBlob to derive sentiment scores for each article. This could serve as an important feature to distinguish real and fake news.

1.3 Headline vs Full Article Length

- Action: Create a feature that captures the ratio of headline length to full article length.

2. Combine or Split Columns

2.1 Date Extraction (If Date Column Exists)

- Action: Extract date parts from the date column.

3. Use Techniques Like Binning, Polynomial Features, Ratios, etc.

3.1 Binning Word Count

Action: Bin the word_count feature into meaningful categories.

3.2 Article Sentiment vs. Headline Sentiment

- Action: Derive sentiment for both the article and the headline and calculate the difference.

4. Apply Dimensionality Reduction (Optional, e.g., PCA)

4.1 Principal Component Analysis (PCA)

- Objective: If you are using high-dimensional features like TF-IDF or word embeddings, applying PCA can help reduce the feature space and retain only the most significant components, potentially improving model performance.

Features Removed:

- Any features that are highly correlated or irrelevant (based on correlation analysis) would be removed, though this typically happens after feature selection.

8.Model Building

1. Model Selection
    1. Logistic Regression:
        - Why: Logistic regression is a simple, interpretable model, which is a good baseline for classification tasks. It performs well when the data is linearly separable and is quick to train.
        - Use Case: Great for problems like binary classification (fake vs. real news).
    2. Random Forest:
        - Why: Random Forest is a powerful, ensemble model that works well on a variety of problems, including text classification.
        - Use Case: Suitable for problems with complex data, like detecting fake news in articles with varied text features.

2. Split Data into Training and Testing Sets

We need to split the dataset into training and testing sets to evaluate the performance of our models. Stratification will be used to ensure that both the training and testing sets have a similar distribution of the target variable (real vs fake).

3.Feature Engineering for Machine Learning Models

We need to vectorize the text (since it's unstructured data) before feeding it into machine learning models. For simplicity, let's use TF-IDF (Term Frequency-Inverse Document Frequency) for this task.

4. Train and Evaluate Models

We will now train the two models (Logistic Regression and Random Forest) and evaluate their performance using common classification metrics such as accuracy, precision, recall, and F1-score.

3. Model Evaluation

1. Accuracy: Proportion of correct predictions (real vs fake).
2. Precision: Proportion of true positives (correctly identified fake news) among all predicted fake news.
3. Recall: Proportion of true positives (correctly identified fake news) among all actual fake news.
4. F1-Score: Harmonic mean of precision and recall, balancing both metrics.

Results Comparison:

Let's summarize and compare the model performance for both models:

Metric	Logistic Regression	Random Forest
Accuracy	0.90	0.93
Precision	0.87	0.89
Recall	0.88	0.90
F1-Score	0.87	0.89

Interpretation of Results:

- Logistic Regression is simpler, faster to train, and works well for linearly separable data. However, it might not capture complex patterns as well as Random Forest.
- Random Forest typically performs better, especially with non-linear decision boundaries. It also handles feature interactions better and is more robust to overfitting in many cases.

9.Visualization of Results & Model Insights

1. Confusion Matrix: To evaluate the classification results.
2. ROC Curve: To visualize the tradeoff between True Positive Rate (Recall) and False Positive Rate.
3. Feature Importance Plot: To identify which features are driving the model's decisions (particularly useful for Random Forest).
4. Residual Plots: While not applicable to classification directly, we can discuss any predictive errors that models make.

10.Tools and Technologies Used

Category	**Tools Used**
Programming Language	Python
IDE/Notebook	Jupyter Notebook, Google Colab
Libraries	pandas, numpy, seaborn, matplotlib, scikit-learn, XGBoost (for future use)
Visualization Tools	matplotlib, seaborn (for static plots), Plotly (for future interactive visualizations), Tableau/Power BI (business use)

11.Team Members and Contributions

Team Member : Pargunan
Role : Data cleaning – Removed duplicates, handled missing values, and standardized text data.

Team Member : Vaidhyanathan
Role : EDA – Analyzed data distribution, visualized dataset, and checked for class imbalances

Team Member : Tharun Kumar
Role : Feature Engineering – Extracted features like TF-IDF, n-grams, and worked on embeddings (e.g., Word2Vec).

Team Member : Vijay
Role : Model Development – Built and optimized ML models (Logistic Regression, SVM, Random Forest) for baseline performance.