### **Fake News Detection System using Natural Processing Language**

## **Phase-2**

**Student Name:Niruba.V**  
**Register Number:** 512223149026  
**Institution:**SKP Engineering college  
**Department:**  BE CSE(Cyber security)  
**Date of Submission:** 10-5-2025  
**GitHub Repository Link:** <https://github.com/Nirubav16/Niruba-NM.git>

1.Problem Statement:

In today's digital age, the rapid dissemination of information through social media and online platforms has made it easier for fake news to spread widely and quickly. Fake news, which consists of misleading or entirely false information presented as legitimate news, poses significant threats to society by influencing public opinion, manipulating political outcomes, and causing social unrest. The manual detection of fake news is inefficient, time-consuming, and often unreliable due to the vast amount of content generated daily. Therefore, there is an urgent need for an automated and accurate system that can identify and filter fake news in real-time.

2. Project Objectives

1. To develop an automated fake news detection system using Natural Language Processing (NLP) techniques that can accurately classify news articles and social media content as real or fake.

2. To design a robust NLP pipeline capable of preprocessing raw textual data, including tokenization, stopword removal, lemmatization, and feature extraction using techniques such as TF-IDF, word embeddings, and sentiment analysis.

3. To build and evaluate machine learning and deep learning models such as Logistic Regression, Random Forest, Support Vector Machine (SVM), LSTM, and transformer-based models (e.g., BERT, IndicBERT) for fake news classification.

4. To curate or utilize an appropriate dataset consisting of labeled real and fake news, including and regional Indian languages, with special attention to code-mixed content (e.g., Hinglish).

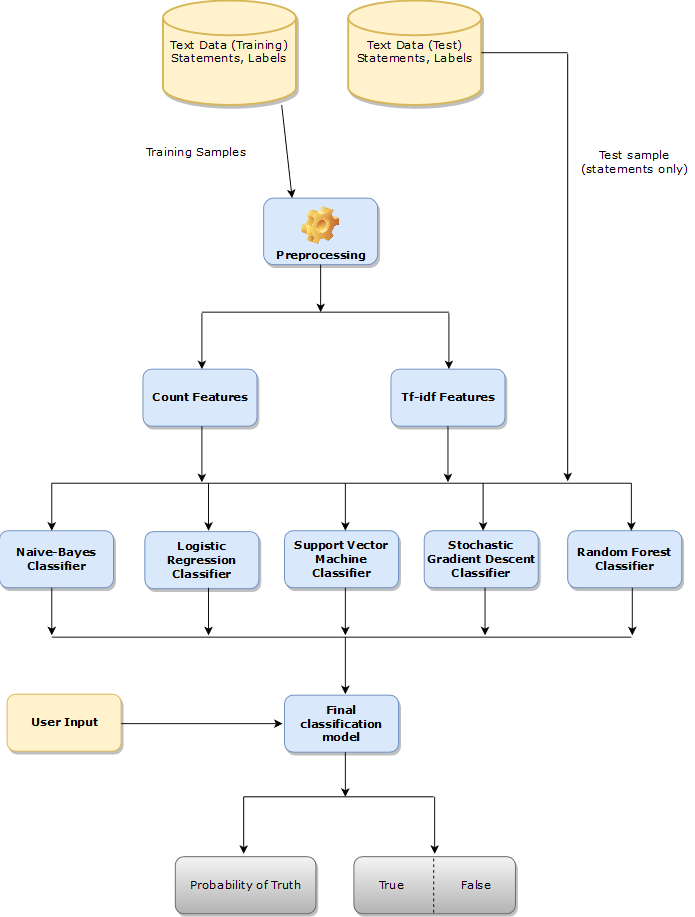
5. To handle multilingual and code-mixed text, leveraging language detection, transliteration tools, and pretrained multilingual models suitable for India’s diverse linguistic landscape.

6. To assess model performance using standard metrics such as accuracy, precision, recall, F1-score, and ROC-AUC, ensuring reliability and generalization across varied data.

7. To develop a user-friendly interface or application where users can input text or URLs to verify the credibility of news content in real-time.

8. To contribute to misinformation mitigation efforts by providing a scalable and practical solution that supports fact-checking and promotes information integrity in India's digital ecosystem.

3.Flow chart of the project work flow



4.Data Description

The dataset used for the Fake News Detection project plays a crucial role in training and evaluating the performance of the classification models. It consists of a collection of news articles labeled as either “real” or “fake.” The data may be sourced from fact-checking websites, online news portals, and public repositories.

1. Data Sources

Fake News: Sourced from known misinformation repositories, fake news detection datasets (e.g., FakeNewsNet, LIAR dataset), and Indian fact-checking platforms like Alt News, BoomLive, and Factly.

Real News: Collected from reputable Indian news agencies and online portals such as The Hindu, Times of India, NDTV, etc.

2. Data Format

Each news article or entry typically includes:

ID: Unique identifier for the article/post

Title: The headline or title of the news item

Text: Full news content or main body

Label: Target variable (1 = Fake, 0 = Real)

Source (optional): The origin of the news (e.g., website or platform)

Language (optional): Language of the content (e.g., English, Hindi)

Date (optional): Publication or posting date

3. Data Characteristics

Multilingual Content: The dataset may contain English, Hindi, or code-mixed content (e.g., Hinglish).

Balanced Classes: Effort is made to balance the number of fake and real news items to avoid bias in model training.

Text Length: Varies from short headlines (for social media posts) to full-length articles.

4. Data Size

Typical datasets used contain between 5,000 to 50,000 entries, depending on the sources and scope.

For multilingual models, additional regional datasets may be collected and annotated manually or semi-automatically.

5.Data Processing

This phase prepares raw textual data for effective modeling by cleaning, normalizing, and converting it into numerical features.

1. Text Cleaning

Lowercasing: Standardizes text (e.g., "Fake" → "fake").

Remove punctuation: Eliminates symbols like !, ?, ., etc.

Remove URLs, HTML tags, emojis: Strips out web links and formatting noise.

Remove numbers: Unless numerics are contextually important.

2. Tokenization

Splits text into individual words (tokens) using:

nltk.word\_tokenize()

spaCy tokenizer

Example: "Fake news spreads quickly" → ["fake", "news", "spreads", "quickly"]

3. Stopword Removal

Removes common words like "is", "the", "and" that don’t add value.

Use language-specific stopword lists (e.g., for Hindi or Hinglish).

4. Lemmatization or Stemming

Lemmatization reduces words to their base form: “running” → “run”

Stemming cuts suffixes: “nationality” → “nation”

Libraries: nltk, spaCy, or TextBlob

5. Handling Multilingual & Code-Mixed Text

Language Detection: Use langdetect to identify the text language.

Transliteration/Translation: Convert Hindi or code-mixed Hinglish to English (optional).

Use multilingual NLP models (e.g., IndicBERT) to avoid manual translation.

6. Text Normalization

Corrects spelling errors and slang.

Expands contractions: "don’t" → "do not"

Handles abbreviations: "u" → "you"

7. Vectorization (Feature Extraction)

Converts text into numeric format:

Bag of Words (BoW) – frequency of each word.

TF-IDF (Term Frequency-Inverse Document Frequency) – word importance.

Word Embeddings – Word2Vec, GloVe.

Transformers (BERT, IndicBERT) – contextual representations.

8. Final Output

Cleaned and vectorized dataset, ready for training machine learning models.

6.Exploratory Data Analysis(EDA)

EDA helps you understand the structure, patterns, and key insights from your dataset before applying machine learning models. In fake news detection, EDA can reveal word usage trends, data imbalance, text length distributions, and more.

1. Data Overview

Check dataset shape: Number of rows and columns.

Preview samples: View the first few entries of fake and real news.

Class distribution: Count how many samples are labeled as “fake” vs. “real.”

Use value\_counts() to detect class imbalance.

df['label'].value\_counts().plot(kind='bar')

2. Text Length Analysis

Measure number of words or characters per article.

Helps understand if fake news tends to be shorter or longer than real news.

df['text\_length'] = df['text'].apply(lambda x: len(x.split()))

df['text\_length'].hist(by=df['label'])

3. Word Frequency Analysis

Most common words in fake vs. real news.

Use CountVectorizer or collections.Counter.

from collections import Counter

fake\_words = " ".join(df[df.label==1]['text']).split()

real\_words = " ".join(df[df.label==0]['text']).split()

Counter(fake\_words).most\_common(10)

Word clouds can visually show prominent words.

from wordcloud import WordCloud

WordCloud().generate(" ".join(df[df.label==1]['text']))

4. N-gram Analysis

Bigrams or trigrams (e.g., "breaking news", "government plans") may distinguish fake from real news.

Helps capture collocations and phrases.

5. Sentiment Analysis

Analyze polarity and subjectivity using tools like TextBlob or VADER.

Fake news may use more emotional or exaggerated language.

from textblob import TextBlob

df['polarity'] = df['text'].apply(lambda x: TextBob(x).sentiment.polarity)

7.Tools And Technologies:

1. Programming Language

Python: Widely used in NLP and machine learning due to its powerful libraries and community support.

2. Data Collection & Handling

Pandas: For data manipulation, cleaning, and analysis.

NumPy: For numerical operations and array handling.

BeautifulSoup / Scrapy: For web scraping if collecting your own news data.

Google Sheets / CSV / JSON: For storing and importing structured datasets.

3. Natural Language Processing (NLP)

NLTK: For basic NLP tasks like tokenization, stemming, and stopword removal.

spaCy: For fast NLP operations including POS tagging, NER, and dependency parsing.

TextBlob: For quick sentiment analysis and text preprocessing.

langdetect: For identifying language in multilingual/code-mixed content.

Indic NLP Library: For handling Indian languages (tokenization, transliteration, etc.)

4. Feature Extraction

Scikit-learn:

CountVectorizer, TfidfVectorizer for converting text to numeric features.

Gensim: For Word2Vec and topic modeling.

Hugging Face Transformers:

For using pre-trained models like BERT, DistilBERT, and IndicBERT (for Indian languages).

5. Machine Learning & Deep Learning

Scikit-learn: For traditional ML algorithms (Logistic Regression, SVM, Random Forest).

TensorFlow / Keras: For building de learning models like LSTM, CNN.

PyTorch: Alternative deep learning framework, especially useful with Hugging Face models.

6. Model Evaluation & Visualization

Matplotlib / Seaborn: For data visualization and EDA (bar charts, histograms, confusion matrix).

Plotly: For interactive visualizations.

Scikit-learn metrics: For evaluating model performance (accuracy, precision, recall, F1-score).

7. Deployment (Optional)

Flask / Django: For developing a web-based interface.

Streamlit / Gradio: For quickly creating interactive model demos.

Heroku / Render / AWS / Google Cloud: For hosting your application online.

8. Version Control

Git / GitHub: For code versioning, collaboration, and project documentation.

8.Team Memebers and Contributions

Name:C.Hema Sri

Role:🔧 1. Front-End Developer (HTML/CSS/JavaScript)

Responsible for designing and building the user interface (UI) that users interact with.

Responsibilities:

Create clean, responsive web pages using HTML and CSS

Design input forms for users to submit news text or URLs

Display prediction results and feedback messages

Ensure mobile and cross-browser compatibility

Tools:

HTML5, CSS3

Name:P.Kavitha

Role:🐍 2. Back-End Developer (Python/Flask/Django)

Handles the server-side logic, integrates the machine learning model, and connects the front-end with back-end logic.

Responsibilities:

Build the web server using Flask or Django

Connect the HTML form to the Python model

Handle user input, process predictions, and return results

Ensure API endpoints (if needed) work correctly

Tools:

Python

Flask or django

Name:V.Niruba

Role: 📊 .Data Analyst( Develops and trains the fake news detection model.)

Responsibilities:

Collect and clean the dataset (e.g., LIAR, FakeNewsNet)

Preprocess text using NLP (tokenization, stopword removal, etc.)

Train and evaluate machine learning models (Logistic Regression, Random Forest, etc.)

Save the trained model (using Pickle or Joblib) for integration with the backend

Tools:

Python (NumPy, Pandas)

Scikit-learn, NLTK, spaCy