**Phase 3**

**Student Name:** DINESH.M

**Register Number:** 731323104006

**Institution:** J.K.K.NATARAJA COLLEGE OF ENGINEERING & TECHNOLOGY

**Department:**B.E COMPUTER SCIENCE AND ENGINEERING

**Date of Submission:** 16.05.2025

**Github Repository Link:** <https://github.com/Dineshdk-17/Batch_04>

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

**1. Problem Statement:**

threat to public opinion, democracy, and societal trust. Fake news, often crafted to mislead or manipulate readers, spreads faster than factual content. This project aims to develop an automated system using Natural Language Processing (NLP) to classify news articles as fake or real.

This is a binary classification problem, and solving it can help in mitigaThe proliferation of fake news across digital platforms poses a serious ting misinformation, enabling platforms to take corrective action, and assisting readers in verifying the authenticity of information**.**

**2. Abstract:**

In today's hyper-connected digital world, the proliferation of fake news poses a significant threat to public discourse, democratic processes, and social stability. The speed and scale at which misinformation spreads across social media and online platforms demand effective and scalable solutions for its detection. This project leverages the power of Natural Language Processing (NLP) and machine learning to develop an advanced fake news detection system capable of analyzing and classifying news articles with high accuracy.

**3. System Requirements:**

**Hardware:**

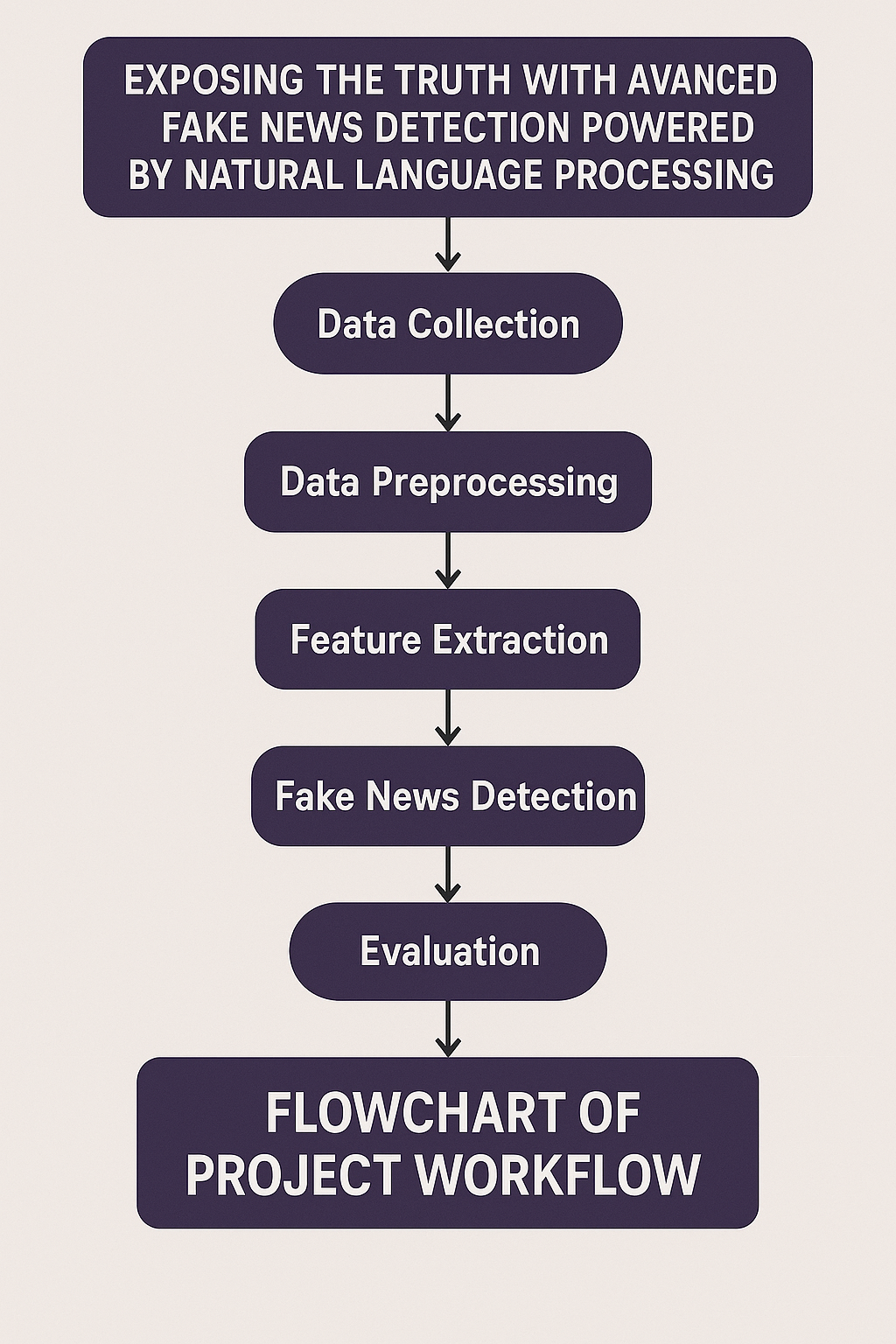
* Intel Core i5 or AMD Ryzen 5
* Intel Core i7 or AMD Ryzen 7 and above

**Software:**

* Windows 10/11
* Linux (Ubuntu preferred)
* macOS
* Libraries/Packages:NumPy, Pandas, Scikit-learn, NLTK or spaCy, Matplotlib

**4. Objectives:**

* Develop a Robust Fake News Detection Model
* Leverage NLP for Text Classification
* Enhance Content Source Validation
* Identify Manipulative Linguistic Patterns
* Develop Real-Time Detection Tools
* Improve Public Awareness and Trust

**5. Flowchart of Project Workflow:**

**6. Dataset Description:**

1. Content: The dataset contains news articles with fields like title, text, and a label indicating whether the news is fake or real.
2. Labels: Each entry is classified as either "Fake" or "Real", enabling binary classification for model training.
3. Source: Data is collected from reputable online platforms such as Kaggle or fact-checking websites like PolitiFact and GossipCop.
4. Format & Size: Typically provided in CSV format, the dataset includes thousands of entries and is primarily in English.

**7. Data Preprocessing:**

* Convert text to lowercase, and remove punctuation, special characters, and stopwords to simplify the input.
* Break text into words (tokens) and reduce them to their root form to standardize word usage (e.g., “running” → “run”).
* Transform cleaned text into numerical format using methods like **TF-IDF** or **Word Embeddings** (e.g., Word2Vec, BERT) for model input.

**8. Exploratory Data Analysis (EDA):**

* Analyze the number of fake vs. real news articles to identify any class imbalance.
* Examine the distribution of word and character counts in articles and titles to detect outliers or inconsistencies.
* Identify and visualize the most common words in fake and real news separately using bar charts or word clouds.
* Analyze frequently occurring bigrams/trigrams (e.g., “breaking news”, “according to”) to find patterns in fake vs. real content.

**9. Feature Engineering:**

* TF-IDF: Captures word importance across documents.
* N-grams: Includes word pairs/triples for context.
* Text Length: Uses article or title length as a feature.
* Embeddings: Converts text to dense vectors (e.g., Word2Vec).

**10. Model Building:**

* Select Model: Choose algorithms like Logistic Regression or BERT.
* Train Model: Fit on vectorized training data.
* Validate Model: Use cross-validation to assess performance.
* Tune Parameters: Optimize using Grid or Random Search**.**

**11. Model Evaluation:**

* Accuracy: Overall prediction correctness.
* Precision: Correctly identified fake news ratio.
* Recall: Ability to find all fake news.
* F1-Score: Balance between precision and recall.

**12. Deployment:**

* Save Model: Export trained model for use.
* Build API: Create prediction service endpoint.
* Integrate: Connect API to apps or websites.
* Monitor: Track and update model performance.

**13. Source code:**

import pandas as pd

from sklearn.model\_selection import train\_test\_split

from sklearn.feature\_extraction.text import TfidfVectorizer

from sklearn.linear\_model import LogisticRegression

from sklearn.metrics import accuracy\_score, classification\_report

# Load dataset (replace with your dataset path)

data = pd.read\_csv('fake\_news\_dataset.csv')

# Fill missing text values

data['text'].fillna('', inplace=True)

# Features and labels

X = data['text']

y = data['label'] # 0 = Fake, 1 = Real

# Split data into train and test sets

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)

# Vectorize text with TF-IDF

vectorizer = TfidfVectorizer(stop\_words='english', max\_df=0.7)

X\_train\_vec = vectorizer.fit\_transform(X\_train)

X\_test\_vec = vectorizer.transform(X\_test)

# Train Logistic Regression model

model = LogisticRegression()

model.fit(X\_train\_vec, y\_train)

# Predict and evaluate

y\_pred = model.predict(X\_test\_vec)

print(f"Accuracy: {accuracy\_score(y\_test, y\_pred):.4f}")

print(classification\_report(y\_test, y\_pred))

**14. Future scope:**

* Expand to multilingual fake news detection.
* Enable real-time fake news monitoring.
* Integrate transformer-based deep learning models.
* Use user feedback to improve accuracy.

**15. Team Members and Roles:**

**MEMBERS**

**1. MANIKANDAN A**

**Role:** Model Developer

**Responsibilities:**

Build and train the CNN model for digit recognition

Tune hyperparameters for improved accuracy

Evaluate model using accuracy, F1-score, confusion matrix

Integrate and test final project flow

**2.DINESH M**– Data Specialist

**Role:** Data Preprocessing & EDA

**Responsibilities:**

Download and clean the MNIST dataset

Normalize and reshape image data

Perform EDA with visualizations (histograms, heatmaps)

Document insights from data analysis

**3.SANTHOSH KUMAR S**– Deployment Engineer

**Role:** UI & Deployment

**Responsibilities:**

Build user interface using Streamlit or Gradio

Link trained model to the frontend for predictions

Deploy on cloud platform (Streamlit Cloud or Hugging Face Spaces)

Capture UI screenshots and provide deployment link

**4.DHANUSH K**– Documentation Expert

**Role:** Report & Presentation

**Responsibilities:**

Draft report sections like problem statement, abstract, objectives, etc.

Create the flowchart and future scope

Prepare submission-ready presentation slides

Ensure final formatting and completeness of documentation

