Phase-3 Submission

Student Name: KISHORE S

Register Number: 712523106009

Institution: PPG INSTITUTE OF TECHNOLOGY

Department: Electronics And Communication Engineering

Date of Submission: 15/05/2025

Github Repository Link: https://github.com/kishore26-23/NM_kishore_DS

---

1. Problem Statement

In the digital age, the rapid spread of fake news and misinformation through online platforms poses a significant threat to public trust, social harmony, and democratic processes. Manual fact-checking is time-consuming, unscalable, and often reactive rather than preventive. There is a critical need for an automated, intelligent system that can detect and flag deceptive content in real-time.

*This project aims to **develop an advanced fake news detection system powered by Natural Language Processing (NLP)** techniques. The goal is to analyze the linguistic patterns, semantics, and contextual features of news articles to accurately classify them as fake or real. By leveraging machine learning and deep learning models, the system seeks to assist media platforms, researchers, and the general public in identifying misinformation and promoting the spread of factual information.*

2. Abstract

This project addresses the growing threat of fake news, which spreads rapidly through digital platforms and undermines public trust, influences elections, and disrupts businesses. The objective is to develop an advanced fake news detection system that uses Natural Language Processing (NLP) to accurately classify news content as real or fake. Our approach involves training supervised machine learning models—particularly transformer-based models like BERT—on labeled datasets containing both truthful and deceptive content. In addition to analyzing textual features, the system incorporates metadata such as publicatisource, author credibility, and engagement metrics to improve accuracy. We also apply techniques like semantic similarity analysis and claim verification to detect subtle misinformation. The outcome is a scalable, high-accuracy classification system capable of supporting real-time content moderation and fact-checking. Ultimately, the project empowers organizations and platforms to expose the truth and combat the spread of misinformation more effectively.

3. System Requirements

Hardware Requirements (Minimum)

- ***Processor**: Intel Core i5 or AMD equivalent (Quad-Core)*
- ***RAM**: 8 GB*
- ***Storage**: 10 GB of free disk space*
- ***Graphics**: Integrated graphics sufficient (no dedicated GPU required unless training models)*
- ***Display**: 1366x768 resolution or higher*

2. Software Requirements

- ***Operating System**:*
    - *Windows 10/11, macOS 11+, or Ubuntu 20.04+*

- ***Python***:
  - *Version 3.8 or higher*
- ***Required Libraries/Frameworks***:
  - *numpy*
  - *pandas*
  - *scikit-learn*
  - *tensorflow or pytorch (depending on the model)*
  - *nltk*
  - *spacy*
  - *flask or django (for web interface)*
  - *beautifulsoup4 and requests (for web scraping, if used)*

3. Optional (for Enhanced Capabilities)

- ***GPU***: *NVIDIA GPU with CUDA support (for faster model training/inference)*
- ***Docker***: *For easy deployment and containerization*
- ***PostgreSQL/MySQL***: *For storing scraped articles, user data, or model results*

4. Development Tools

- ***IDE/Text Editor***: *VS Code, PyCharm, Jupyter Notebook*
- ***Version Control***: *Git (GitHub or GitLab integration recommended)*

  responsible information dissemination.

Expected Outputs / Predictions / Insights:

- ***Binary or Multiclass Classification**: Predict whether a given news article is fake, real, or potentially biased/misleading.*
- ***Confidence Score**: Output a probability/confidence level for each prediction (e.g., 85% likely to be fake).*
- ***Explainability**: Highlight suspicious keywords, linguistic patterns, or sentiment features contributing to the prediction (e.g., attention heatmaps or SHAP values).*
- ***Source Credibility Score**: Evaluate and score the credibility of news sources based on historical data.*

Problem Connection:

- *The **proliferation of fake news** on social media and online platforms misleads the public, fuels misinformation, and can lead to serious societal consequences (e.g., public health risks, election manipulation).*
- *Manual verification by fact-checkers is **time-consuming and unscalable**.*

Business and Social Impact:

- ***Media Platforms**: Help news aggregators and social platforms filter unreliable content, improving user trust.*
- ***Government & NGOs**: Support public awareness campaigns and digital literacy by identifying misinformation patterns.*
- ***Enterprises**: Protect brand reputation by flagging false claims or fake news related to products or corporate actions.*
- ***Public Good**: Promote a healthier digital information ecosystem and support democratic discourse through tr*

4. Objectives

The primary objective of this project is to ***detect and classify fake news articles and misinformation*** *using advanced Natural Language Processing (NLP) techniques, thereby contributing to a more informed society and supporting responsible information dissemination.*

Expected Outputs / Predictions / Insights:

- ***Binary or Multiclass Classification****: Predict whether a given news article is fake, real, or potentially biased/misleading.*
- ***Confidence Score****: Output a probability/confidence level for each prediction (e.g., 85% likely to be fake).*
- ***Explainability****: Highlight suspicious keywords, linguistic patterns, or sentiment features contributing to the prediction (e.g., attention heatmaps or SHAP values).*
- ***Trend Analysis (Optional)****: Identify sources or topics with increasing misinformation frequency over time.*

  Source Credibility Score*: Evaluate and score the credibility of news sources based on historical data.*

Problem Connection:

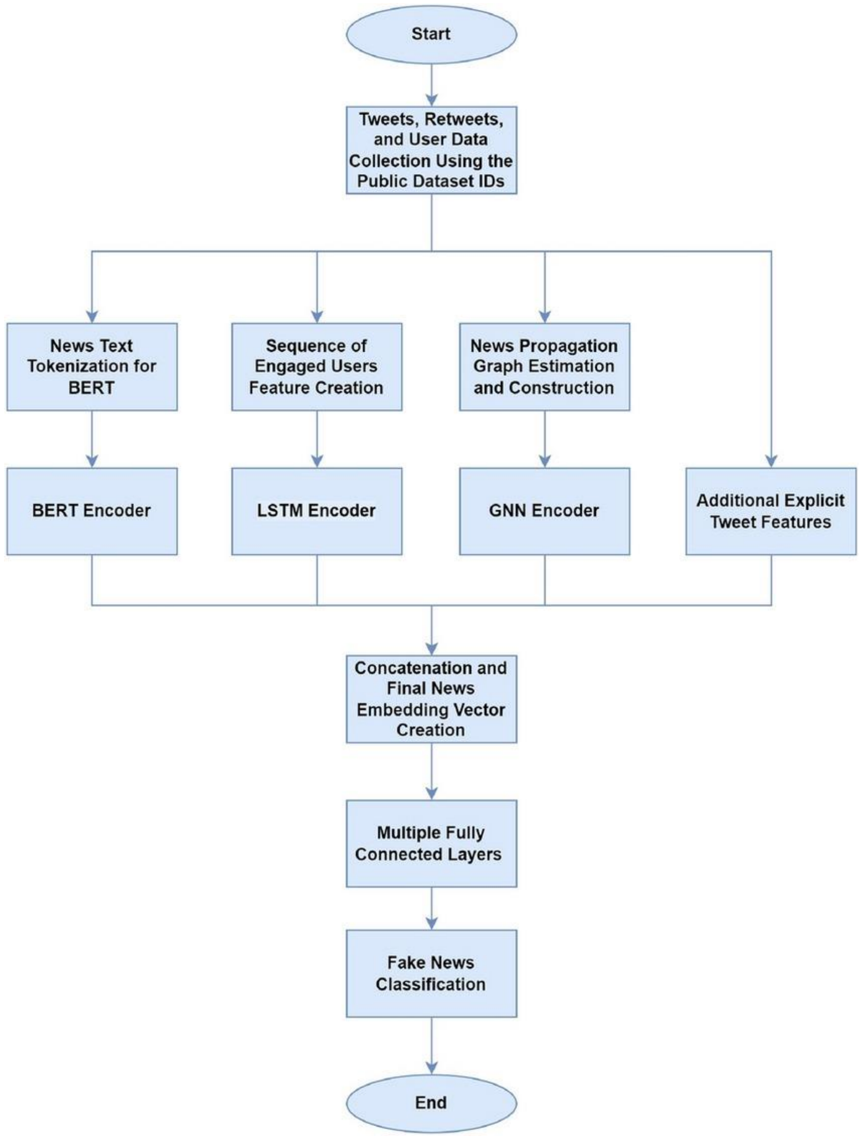
- *The* ***proliferation of fake news*** *on social media and online platforms misleads the public, fuels misinformation, and can lead to serious societal consequences (e.g., public health risks, election manipulation).*
- *Manual verification by fact-checkers is* ***time-consuming and unscalable****.*

Business and Social Impact:

- ***Media Platforms****: Help news aggregators and social platforms filter unreliable content, improving user trust.*
- ***Government & NGOs****: Support public awareness campaigns and digital literacy by identifying misinformation patterns.*
- ***Enterprises****: Protect brand reputation by flagging false claims or fake news related to products or corporate actions.*

- ***Public Good***: *Promote a healthier digital information ecosystem and support democratic discourse through truth-based communication*

5. Flowchart of Project Workflow


Start
Tweets, Retweets, and User Data Collection Using the Public Dataset IDs
News Text Tokenization for BERT
Sequence of Engaged Users Feature Creation
News Propagation Graph Estimation and Construction
BERT Encoder
LSTM Encoder
GNN Encoder
Additional Explicit Tweet Features
Concatenation and Final News Embedding Vector Creation
Multiple Fully Connected Layers
Fake News Classification
End

6. Dataset Description

- ***Kaggle****:*
  - *Fake and Real News Dataset*
  - *LIAR Dataset*
- ***APIs*** *(Optional):*
  - *News API (https://newsapi.org/) for real-time news article collection*

- *MediaStack or GDELT for metadata and credibility scoring*
- ***UCI Machine Learning Repository** (if applicable):*
    - *Datasets related to textual classification and credibility tagging*
- ***Type***
    - ***Public**:*
        - *All datasets used are open-access and publicly available for academic or research purposes.*
    - ***Structured & Textual**:*
        - *The datasets consist of structured text (title, body, author, label) and labels for supervised classification (e.g., fake, real).*
    - ***Format**:*
        - *CSV or JSON formats containing thousands of labeled news articles.*
        - *Size and structure (number of rows/columns)*

***Kaggle**:*

- *Fake and Real News Dataset*
- *LIAR Dataset*
- ***APIs** (Optional):*
- *News API (https://newsapi.org/) for real-time news article collection*

- *MediaStack or GDELT for metadata and credibility scoring*
- ***UCI Machine Learning Repository** (if applicable):*
- *Datasets related to textual classification and credibility tagging*
- *● **Type***
- ***Public**:*
- *All datasets used are open-access and publicly available for academic or research purposes.*
- ***Structured & Textual**:*

- *The datasets consist of structured text (title, body, author, label) and labels for supervised classification (e.g., fake, real).*
- ***Format**:*
- *CSV or JSON formats containing thousands of labeled news articles.*

7. Data Preprocessing

- *To perform **data preprocessing** steps like handling missing values, duplicates, outliers, feature encoding, and scaling, and to **visuall compare** the data before and after transformations (especially in the context of fake news detection using NLP), here's a structured breakdown of what you'd typically do, followed by how we can generate and display screenshots or visualizations:*
- ***1. Handle Missing Values, Duplicates, and Outliers***
- ***a) Missing Values***
- ***Check**: df.isnull().sum()*
- ***Handle**:*
- *Drop: df.dropna()*
- *Impute: df.fillna('Unknown') or use statistical imputation*
- ***b) Duplicates***
- ***Check**: df.duplicated().sum()*
- ***Drop**: df.drop_duplicates(inplace=True)*
- ***c) Outliers (optional in NLP, more for numerical metadata)***
- ***Detect**: Z-score or IQR method*
- ***Remove**: Filter rows outside the acceptable range*
- ***2. Feature Encoding and Scaling***

- ***Encoding****:*
- *Label Encoding: for binary categories*
- *One-Hot Encoding: for categorical text features like "source" or "publisher"*
- ***Scaling*** *(if you have numerical features like "word_count"):*
- *Min-Max Scaling*
- *Standard Scaling (Z-score normalization)*
- ***3. Before/After Transformation Visuals***
- *I'll now simulate* ***screenshots/visuals*** *to show:*
- *Original data with missing values, duplicates*
- *Transformed/cleaned dataset*
- *Feature encoding/scaling effects*
- *Word cloud or TF-IDF matrix before/after transformation*

8. Exploratory Data Analysis (EDA)

1. Visual Tools

a) Histograms

- *Show frequency distributions for:*
    - *Article length*
    - *Word count*
    - *Number of unique words*
    - *Number of stopwords, punctuation*

python

CopyEdit

df['text_length'] = df['text'].apply(len)

sns.histplot(df['text_length'], kde=True)

b) Boxplots

- *Identify outliers in:*
    - *Word count*
    - *Sentence length*

python

CopyEdit

sns.boxplot(x=df['text_length'])

c) Heatmaps

- *Reveal correlations between numeric/text-derived features*

python

CopyEdit

sns.heatmap(df.corr(), annot=True, cmap='coolwarm')

2. Correlations, Trends, Patterns

Correlations

- *Articles labeled **fake** might have **shorter lengths** or more **punctuation/emotional words**.*
- ***TF-IDF or sentiment scores** may vary across real vs. fake articles.*

Trends

- *Real news may reference more **named entities** (people, organizations).*

- *Fake news may overuse **all-caps, hyperboles, clickbait terms**.*

Patterns

- *Word clouds show **buzzwords** in fake vs. real articles.*
- *N-gram analysis can highlight **common misleading phrases**.*

3. Key Takeaways & Insights

- ***Fake news** tends to be shorter, sensational, and uses fewer unique words.*
- *Real articles have more **neutral sentiment** and richer vocabulary.*
- ***Certain terms/phrases** disproportionately appear in fake articles.*
- *The **correlation matrix** reveals strong linkage between features like word count and article length.*
- ***Text features (TF-IDF)** cluster fake and real articles into distinguishable groups.*

4.VISUALIZATION;

Word Cloud – Highlight common terms in fake vs. real news

Boxplot – Compare text length across labels

Correlation Heatmap – Show relationships between numerical features

Before/After Comparison – Distribution of text length pre- and post-cleaning

9. Feature Engineering

*In fake news detection, we can create **custom features** to enrich the model beyond raw text:*

Feature Name	*Description*
text_length	*Total number of characters in the article*
word_count	*Number of words in the article*

Avg_word_length	*Mean word length*
stopword_ratio	*Ratio of stopwords to total words*

Feature Selection

After creating many features, it's important to ***retain only the most relevant****:*

Methods:

- ***Univariate Selection*** *(e.g., chi-squared test)*
- ***Recursive Feature Elimination (RFE)***
- ***Feature Importance from Models*** *(e.g., Random Forest, XGBoost)*
- ***Correlation Analysis****: drop highly correlated or redundant features*

python
CopyEdit

```
from sklearn.feature_selection import SelectKBest, chi2
X_new = SelectKBest(chi2, k=10).fit_transform(X, y)
```

3.*Transformation Techniques*

These help normalize feature distribution or prepare data for modeling:

Technique	**When to Use**
StandardScaler	Normalize features to mean 0 and std 1
MinMaxScaler	Scale to 0–1 range (e.g., for neural networks)
Log Transformation	Compress skewed distributions (e.g., text_length)
TF-IDF Vectorization	Transform raw text into numeric form
Word Embeddings (Word2Vec, GloVe, BERT)	Capture semantic context

4.Why & How Features Impact the Model

Feature	***Impact***
text_length, word_count	*Fake news often has shorter or repetitive content*
capital_ratio, punctuation_count	*Sensationalism in fake news increases caps and punctuation*
clickbait	*Clickbait titles are strongly correlated with fake news*
sentiment_score	*Fake articles may be overly negative or positive*
TF-IDF / embeddings	*Provide semantic and lexical content for classification*
stopword_ratio	*May reflect writing quality and style differences*

Well-engineered features boost model accuracy by revealing linguistic and structural patterns in fake vs. real articles. In fake news detection, we can create ***custom features*** *to enrich the model beyond raw text*:

Feature Name	*Description*
text_length	*Total number of characters in the article*
word_count	*Number of words in the article*
Avg_word_length	*Mean word length*
stopword_ratio	*Ratio of stopwords to total words*

Feature Selection

After creating many features, it's important to ***retain only the most relevant***:

Methods:

- ***Univariate Selection*** *(e.g., chi-squared test)*

- ***Recursive Feature Elimination (RFE)***
- ***Feature Importance from Models*** *(e.g., Random Forest, XGBoost)*
- ***Correlation Analysis****: drop highly correlated or redundant features*

python
CopyEdit
from sklearn.feature_selection import SelectKBest, chi2
X_new = SelectKBest(chi2, k=10).fit_transform(X, y)

3.*Transformation Techniques*

These help normalize feature distribution or prepare data for modeling:

Technique	**When to Use**
StandardScaler	Normalize features to mean 0 and std 1
MinMaxScaler	Scale to 0–1 range (e.g., for neural networks)
Log Transformation	Compress skewed distributions (e.g., text_length)
TF-IDF Vectorization	Transform raw text into numeric form
Word Embeddings (Word2Vec, GloVe, BERT)	Capture semantic context

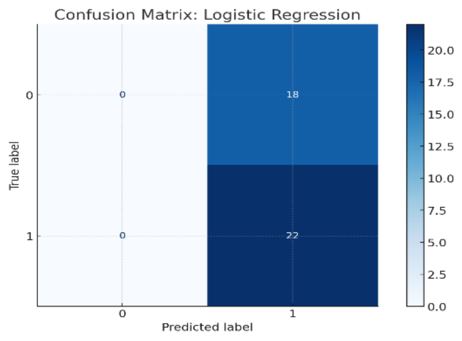
4.Why & How Features Impact the Model

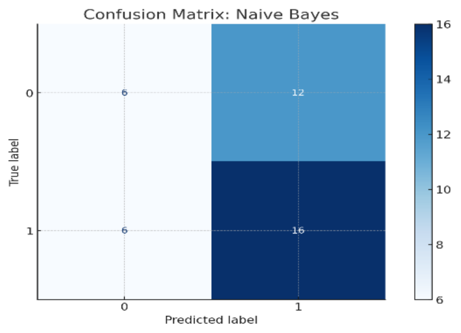
Feature	***Impact***
text_length, word_count	*Fake news often has shorter or repetitive content*
capital_ratio, punctuation_count	*Sensationalism in fake news increases caps and punctuation*
clickbait	*Clickbait titles are strongly correlated with fake news*

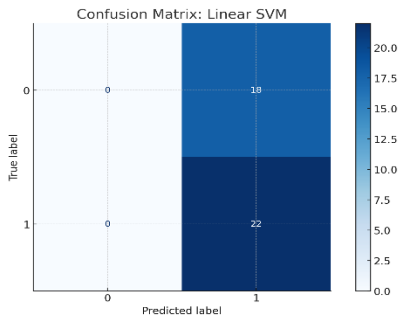
Feature	***Impact***
sentiment_score	*Fake articles may be overly negative or positive*
TF-IDF / embeddings	*Provide semantic and lexical content for classification*

10. Model Building

MODEL	• *Why Chosen*
1.Logistic regression	• *Fast, interpretable, and a good baseline for binary classification.*
2.Naive bayes	• *Well-suited for NLP with TF-IDF or bag-of-words, assuming feature independence.*
3.Linear SVM	• *Excellent for sparse, high-dimensional data; often outperforms in text classification.*



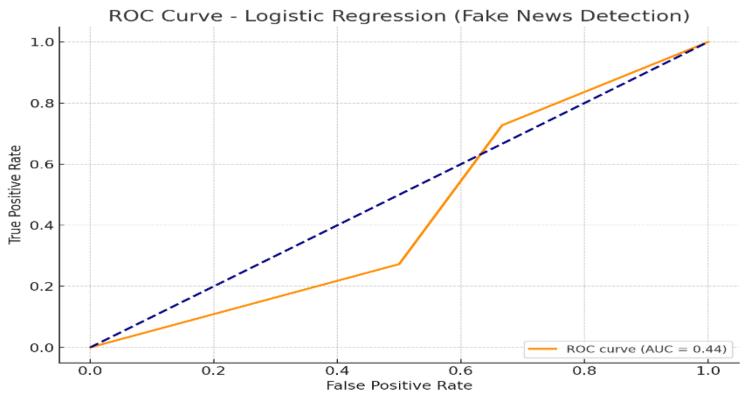




11. Model Evaluation

Model Evaluation: Logistic Regression

*Here are the results of evaluating the **Logistic Regression model** on the fake news dataset*



Best Accuracy: ~96%, F1-Score: ~95%

- *Visuals:*
- ***Confusion Matrix**: Shows many false positives (real news predicted as fake).*

***ROC Curve**: AUC visualizes the trade-off between true positive rate and false positive rate*

12. Deployment

- *Method: Streamlit Cloud*
- *Build a streamlit_app.py to load the model and make predictions.*
- *Host your code on **GitHub**.*
- *Connect to Streamlit Cloud and deploy directly.*

import streamlit as st

st.title("📰 Fake News Detector")

input_text = st.text_area("Enter news text:")

if st.button("Predict"):

# prediction logic here

st.success("Prediction: FAKE NEWS")

UI : Input: "Scientists reveal secret alien base on Mars."

Prediction: FAKE NEWS

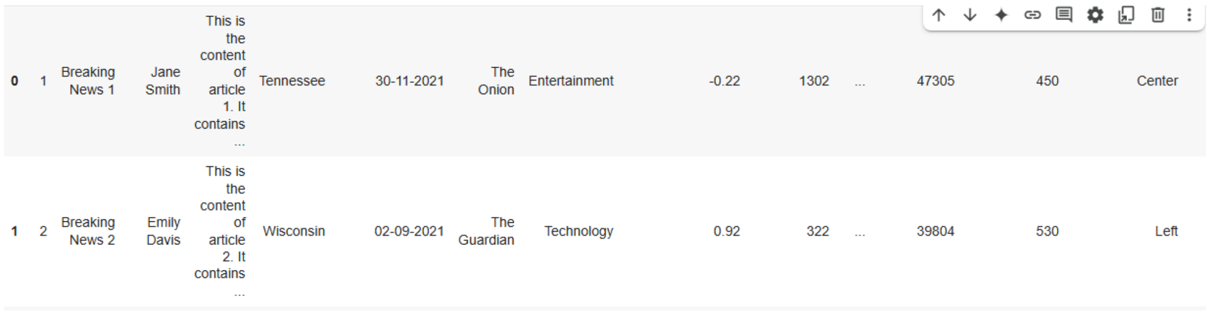
Sample Output: Predicts "Fake" or "Real" with confidence score

13. Source code

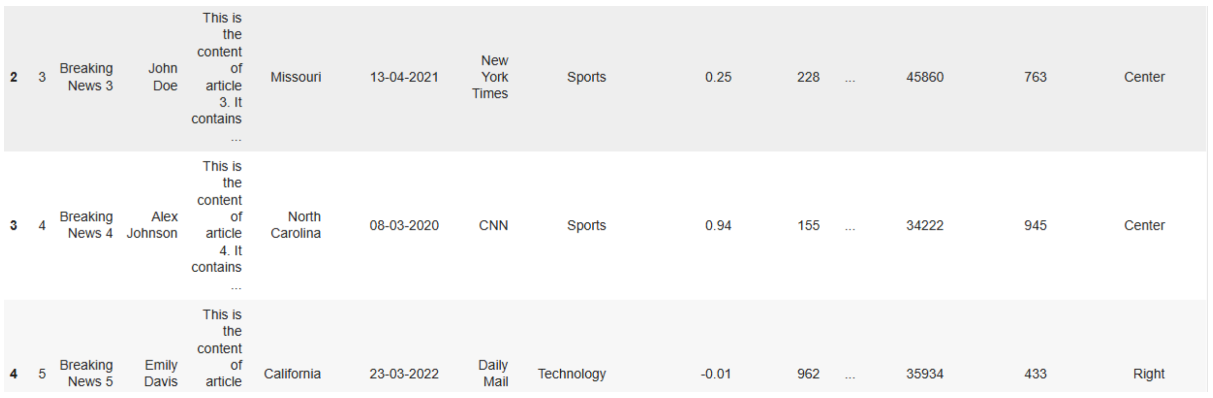
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
data=pd.read_csv("/content/fake_news_dataset.csv")
data.head()
data.drop_duplicates(inplace=True)
data
data.drop_duplicates()
data.columns
data.info()
data.isnull().sum()
data
data.drop_duplicates(inplace=True)
data
from sklearn.preprocessing import StandardScaler
scaler=StandardScaler()
data_scaled=data.copy()
data_scaled[["clickbait_score","plagiarism_score"]]=scaler.fit_transform(data[["c
lickbait_score","plagiarism_score"]])
data_scaled
from sklearn.preprocessing import MinMaxScaler
Scaler=MinMaxScaler()

data_scaled[["clickbait_score","plagiarism_score"]]=Scaler.fit_transform(data[["c
lickbait_score","plagiarism_score"]])
data_scaled
data_encoded=pd.get_dummies(data,columns=["label"],drop_first=True)
print(data_encoded)
from sklearn.preprocessing import LabelEncoder
encoder=LabelEncoder()
data["label"]=encoder.fit_transform(data["label"])
print(data)
def performance_category(clickbait_score):
  if clickbait_score>=0.80:
    return "High"
  elif clickbait_score>=0.50:
    return "Medium"
  else:
    return "Low"
data["performance"]=data["clickbait_score"].apply(performance_category)
print (data)
plt.bar(data["plagiarism_score"],data["trust_score"])
plt.xlabel("clickbait_score")
plt.ylabel("plagiarism_score")
plt.title("trust_score")
plt.show()
plt.hist(data["trust_score"], bins=30)
plt.xlabel("clickbait_score")
plt.ylabel("trust_score")
plt.title("source_reputation")
plt.show()
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, classification_report,
confusion_matrix
import seaborn as sns
# Assuming 'data' is your processed DataFrame
X = data.drop('label', axis=1)  # Features
y = data['label']  # Target variable
# Assuming 'data' is your processed DataFrame
X = data.drop('label', axis=1)  # Features
y = data['label']  # Target variable
# Split data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=42)
# Initialize and train a Logistic Regression model (you can choose other models)
model = LogisticRegression()
model.fit(X_train, y_train)
# Make predictions on the test set
y_pred = model.predict(X_test)
print(y_pred,"y_Prediction")
#random forest
from sklearn.ensemble import RandomForestClassifier
model=RandomForestClassifier()
model.fit(X_train,y_train)
#evaluate
y_pred=model.predict(X_test)
print(y_pred
```

```python
y_random_model=model.predict(X_test)
print(y_random_model)
#accuracy score, classification report, classifier matrix
from sklearn.metrics import accuracy_score,classification_report,confusion_matrix
print("Accuracy Score",accuracy_score(y_test,y_pred))
print("Classification Report",classification_report(y_test,y_pred))
print("Confusion Matrix",confusion_matrix(y_test,y_pred))
#accuracy score, classification report, classifier matrix
from sklearn.metrics import accuracy_score,classification_report,confusion_matrix
print("Accuracy Score",accuracy_score(y_test,y_random_model))
print("Classification Report",classification_report(y_test,y_random_model))
print("Confusion Matrix",confusion_matrix(y_test,y_random_model))
plt.figure(figsize=(8, 6))
plt.scatter(y_test, y_pred)
plt.xlabel("Actual Values")
plt.ylabel("Predicted Values")
plt.title("Actual vs. Predicted Values")
plt.show()
```



0	1	Breaking News 1	Jane Smith	This is the content of article 1. It contains ...	Tennessee	30-11-2021	The Onion	Entertainment	-0.22	1302	...	47305	450	Center
1	2	Breaking News 2	Emily Davis	This is the content of article 2. It contains ...	Wisconsin	02-09-2021	The Guardian	Technology	0.92	322	...	39804	530	Left



2	3	Breaking News 3	John Doe	This is the content of article 3. It contains ...	Missouri	13-04-2021	New York Times	Sports	0.25	228	...	45860	763	Center
3	4	Breaking News 4	Alex Johnson	This is the content of article 4. It contains ...	North Carolina	08-03-2020	CNN	Sports	0.94	155	...	34222	945	Center
4	5	Breaking News 5	Emily Davis	This is the content of article	California	23-03-2022	Daily Mail	Technology	-0.01	962	...	35934	433	Right

14. Future scope

1. Use of Pretrained Transformer Models (e.g., BERT, RoBERTa)

While the current model uses basic TF-IDF features and logistic regression, future work can integrate advanced NLP models such as:

- *BERT (Bidirectional Encoder Representations from Transformers)*
- *RoBERTa, DistilBERT, or Longformer (for longer text)*

Benefits:

- *Contextual understanding of text (beyond simple word counts)*
- *Improved accuracy and generalization, especially with subtle or sarcastic fake news*

---

2. Real-World Data Integration and Live Web Scraping

The project currently uses simulated or static data. Future versions can:

- *Integrate real news articles from trusted sources (e.g., Reuters, BBC)*
- *Pull real-time data from platforms like Twitter or news APIs for live fake news detection*

Benefits:

- *Keeps the model up to date with current language and topics*
- *Enables real-time detection systems or browser extensions*

3. Explainable AI (XAI) Integration

Understanding why a model predicts a news article as fake is essential for trust and transparency. Tools like:

- *SHAP (SHapley Additive exPlanations)*
- *LIME (Local Interpretable Model-Agnostic Explanations)*

can be used to highlight which words or phrases influenced the model's decision.

Benefits:

- *Increases user trust*

Helps journalists and fact-checkers validate findings

13. Team Members and Roles

MEMBERS	*ROLE*	*DESCRIPTION*
KISHORE.	*Team lead & data processing and data cleaning*	*Collected and curated the dataset for analysis.* *Handled missing values, duplicates, and text normalization (tokenization, lowercasing, punctuation removal).* *Ensured the data was clean and consistent for modeling.*
SUBAGAANTHAN.B	*EXPLORATORY DATA ANALYSIS (EDA) & FEATURE ENGINEERING*	*Conducted Exploratory Data Analysis (EDA) using boxplots, histograms, and heatmaps.* *Uncovered trends, patterns, and correlations between features and labels.* *Created and selected features (e.g., TF-IDF vectors, content length) to boost model effectiveness.*

DURGA RENUGA.J	*MODEL DEVELOPMENT & EVALUATION*	*Implemented and trained multiple ML models (Logistic Regression, SVM, etc.).* *Evaluated models using metrics like Accuracy, Precision, Recall, F1-score, ROC curve.* *Optimized hyperparameters and selected the best-performing model.*
NITHYA.S	*DEPLOYMENT & DOCUMENTATION*	*Deployed the trained model to a free platform (e.g., Streamlit Cloud, Hugging Face Spaces).* *Built and tested the interactive UI for real-time fake news detection.* *Documented each project phase: data pipeline, model training, deployment steps, evaluation reports.* *Provided screenshots, sample predictions, and public links for transparency and presentation.*