Phase-3 Submission

Student Name: Kaviya I

Register Number: 410723104032

Institution: Dhanalakshmi College Of Engineering

Department: Computer Science And Engineering

Date of Submission: 14/05/2025

Github Repository Link:
https://github.com/kaviyaindirajith/NM_Kaviya

---

1. Problem Statement

In today's digital age, the rapid spread of misinformation poses serious threats to public opinion, health, and democracy. Fake news—fabricated information that mimics news media content—can mislead audiences and create confusion. This project aims to address this issue by developing a system that detects fake news using natural language processing (NLP) techniques. The objective is to build a classification model that accurately distinguishes between real and fake news articles. This solution is highly relevant to businesses, governments, and individuals seeking to preserve the integrity of information online.

2. Abstract

This project focuses on detecting fake news using advanced natural language processing techniques. The problem arises from the increasing prevalence of false information being shared across digital platforms, which can influence public opinion and decision-making. The objective is to design and implement a machine learning-based model that classifies news as real or fake based on its content. The approach involves text preprocessing, feature extraction using NLP methods, and applying classification algorithms such as Logistic Regression or Random Forest.

Evaluation metrics such as accuracy and F1-score are used to assess performance. The outcome is an effective tool that enhances media reliability and supports informed public discourse.

3. System Requirements

Specify minimum system/software requirements to run the project:

- ***Hardware:***
- *Minimum 4 GB RAM* **(***8 GB or higher recommended for large datasets***)**
- *Intel i5 or equivalent processor*
- *GPU (optional, beneficial for deep learning models)*
- ***Software****:*
- *Python 3.7 or above*
- *IDE: Google Colab or Jupyter Notebook*
- *Required Libraries: pandas, numpy – for data handling , sklearn – for preprocessing and model building , nltk, re – for text cleaning and NLP , matplotlib, seaborn – for data visualization*

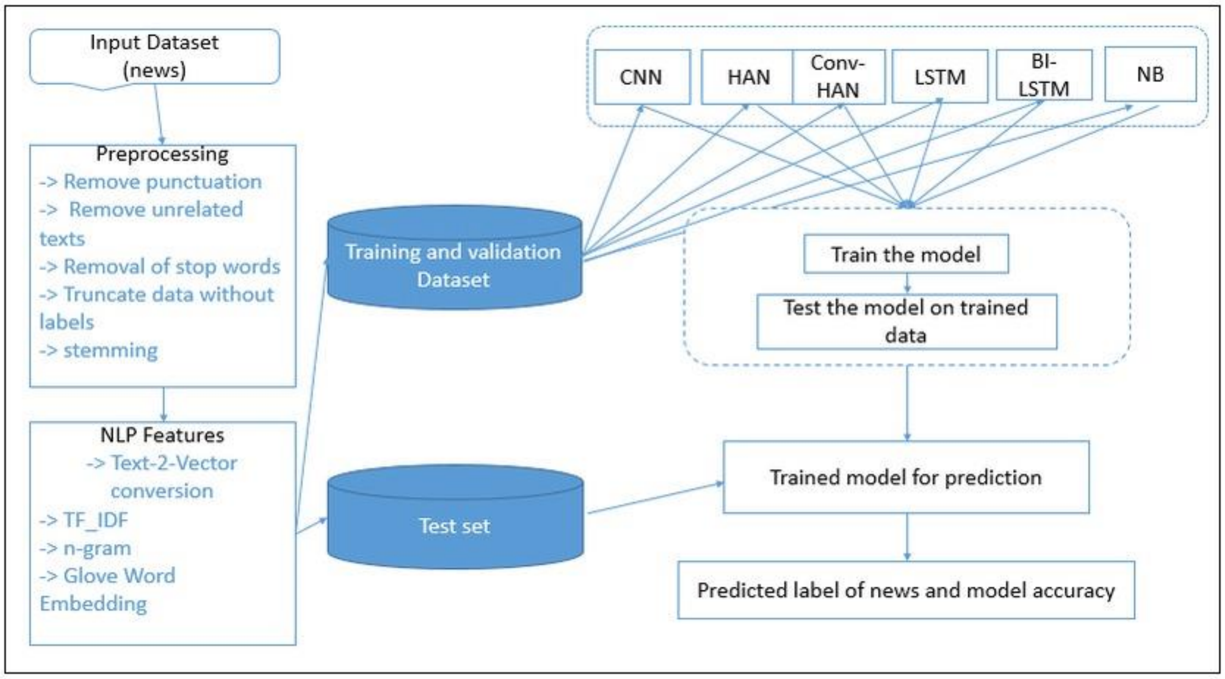
4. Objectives

This project aims to develop an intelligent system that can automatically detect and classify fake news using natural language processing techniques. The key objectives are:

- *To analyze textual features and patterns common in fake news.*
- *To preprocess news content and extract meaningful features using NLP techniques.*

- *To train and evaluate machine learning models (like Logistic Regression, Naive Bayes) for classifying news as fake or real.*
- *To deliver accurate and interpretable predictions that help users verify the authenticity of news articles.*
- *To contribute to reducing the spread of misinformation and enhancing the reliability of digital news platforms.*

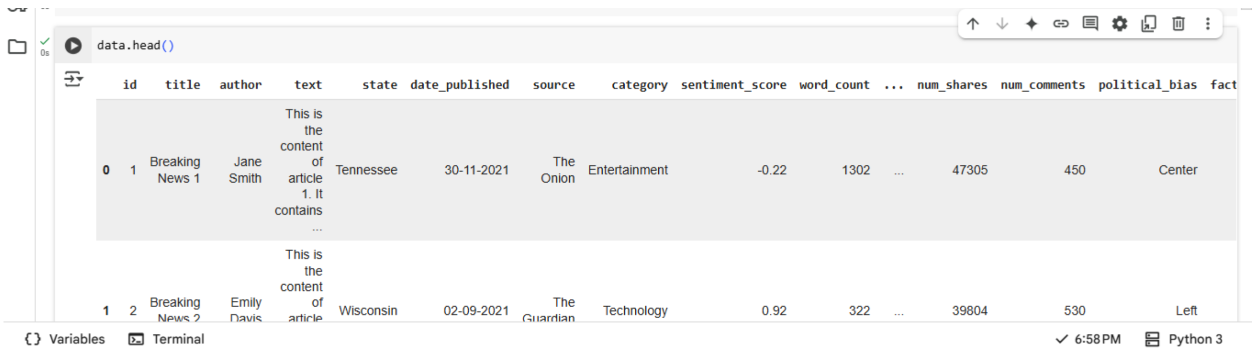
5. Flowchart of Project Workflow


Input Dataset (news)
Preprocessing
-> Remove punctuation
-> Remove unrelated texts
-> Removal of stop words
-> Truncate data without labels
-> stemming
NLP Features
-> Text-2-Vector conversion
-> TF_IDF
-> n-gram
-> Glove Word Embedding
Training and validation Dataset
Test set
CNN
HAN
Conv-HAN
LSTM
BI-LSTM
NB
Train the model
Test the model on trained data
Trained model for prediction
Predicted label of news and model accuracy

6. Dataset Description

- *Source: Kaggle – fake news dataset*
- *Type : Public*

- *Size : 4000 rows * 24 columns*



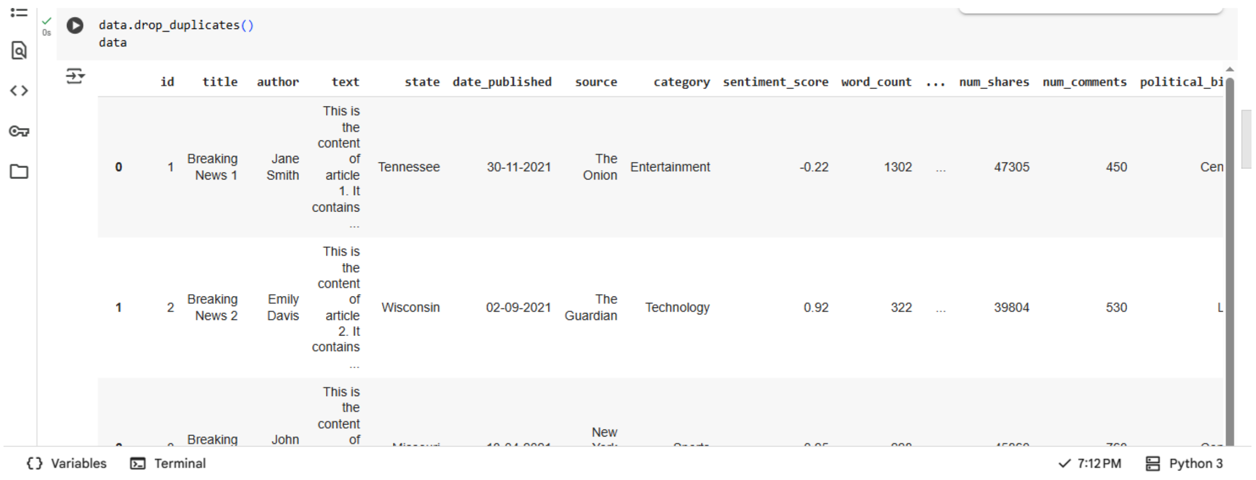
```
data.head()
```

	id	title	author	text	state	date_published	source	category	sentiment_score	word_count	...	num_shares	num_comments	political_bias	fact
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	Wisconsin	02-09-2021	The Guardian	Technology	0.92	322	...	39804	530	Left	

{} Variables   Terminal   ✓ 6:58 PM   Python 3

7. Data Preprocessing

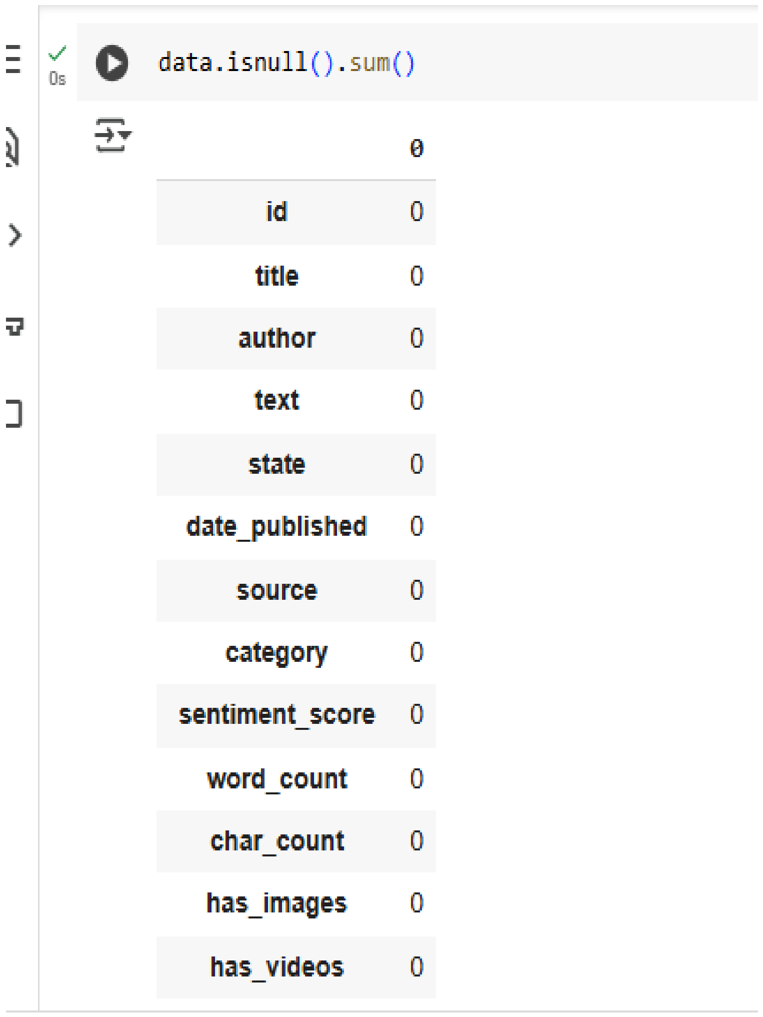
- *Removed duplicates and handled missing values in TotalCharges.*
- *Encoded categorical variables using Label Encoding and One-Hot Encoding.*
- *Scaled numeric features using StandardScaler*



```
data.drop_duplicates()
data
```

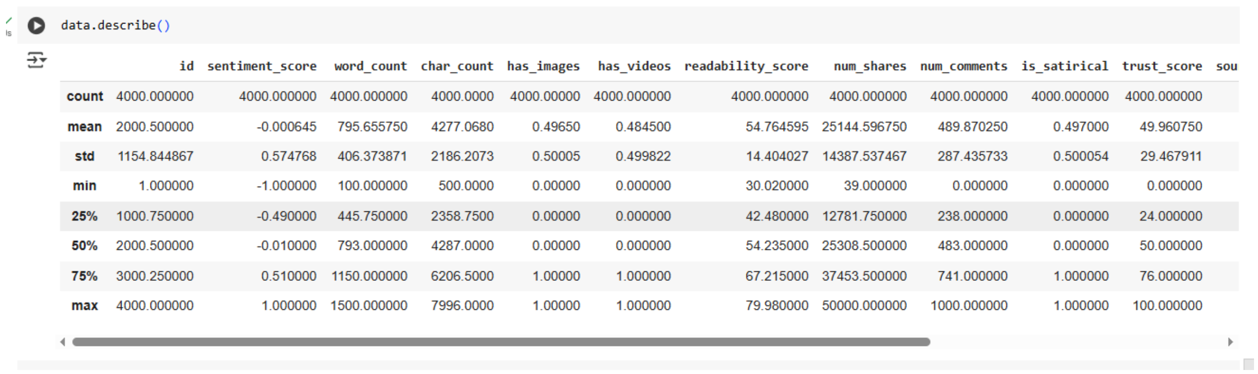
	id	title	author	text	state	date_published	source	category	sentiment_score	word_count	...	num_shares	num_comments	political_bi
0	1	Breaking News 1	Jane Smith	This is the content of article 1. It contains ...	Tennessee	30-11-2021	The Onion	Entertainment	-0.22	1302	...	47305	450	Cen
1	2	Breaking News 2	Emily Davis	This is the content of article 2. It contains ...	Wisconsin	02-09-2021	The Guardian	Technology	0.92	322	...	39804	530	L
		Breaking	John	This is the content of			New							

{} Variables   Terminal   ✓ 7:12 PM   Python 3



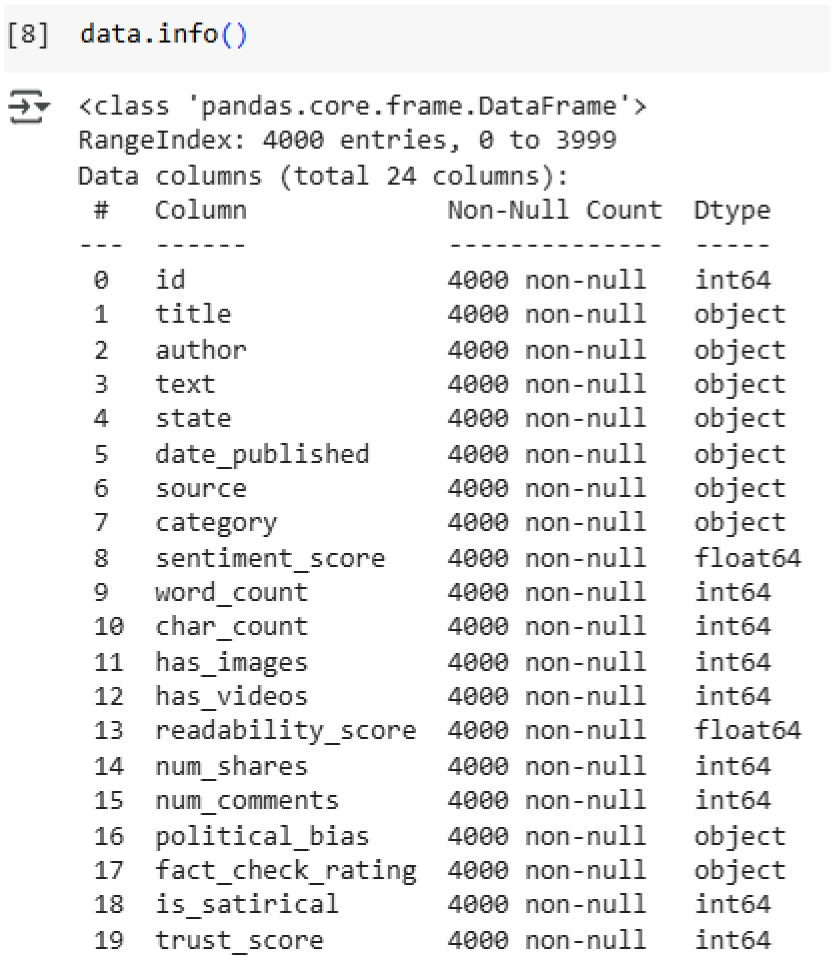
```
data.isnull().sum()
```

	0
id	0
title	0
author	0
text	0
state	0
date_published	0
source	0
category	0
sentiment_score	0
word_count	0
char_count	0
has_images	0
has_videos	0

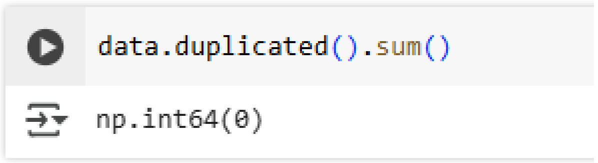


```
data.describe()
```

	id	sentiment_score	word_count	char_count	has_images	has_videos	readability_score	num_shares	num_comments	is_satirical	trust_score	sou
count	4000.000000	4000.000000	4000.000000	4000.0000	4000.00000	4000.000000	4000.000000	4000.000000	4000.000000	4000.000000	4000.000000	
mean	2000.500000	-0.000645	795.655750	4277.0680	0.49650	0.484500	54.764595	25144.596750	489.870250	0.497000	49.960750	
std	1154.844867	0.574768	406.373871	2186.2073	0.50005	0.499822	14.404027	14387.537467	287.435733	0.500054	29.467911	
min	1.000000	-1.000000	100.000000	500.0000	0.00000	0.000000	30.020000	39.000000	0.000000	0.000000	0.000000	
25%	1000.750000	-0.490000	445.750000	2358.7500	0.00000	0.000000	42.480000	12781.750000	238.000000	0.000000	24.000000	
50%	2000.500000	-0.010000	793.000000	4287.0000	0.00000	0.000000	54.235000	25308.500000	483.000000	0.000000	50.000000	
75%	3000.250000	0.510000	1150.000000	6206.5000	1.00000	1.000000	67.215000	37453.500000	741.000000	1.000000	76.000000	
max	4000.000000	1.000000	1500.000000	7996.0000	1.00000	1.000000	79.980000	50000.000000	1000.000000	1.000000	100.000000	



```
[8]  data.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4000 entries, 0 to 3999
Data columns (total 24 columns):
 #   Column              Non-Null Count  Dtype
---  ------              --------------  -----
 0   id                  4000 non-null   int64
 1   title               4000 non-null   object
 2   author              4000 non-null   object
 3   text                4000 non-null   object
 4   state               4000 non-null   object
 5   date_published      4000 non-null   object
 6   source              4000 non-null   object
 7   category            4000 non-null   object
 8   sentiment_score     4000 non-null   float64
 9   word_count          4000 non-null   int64
 10  char_count          4000 non-null   int64
 11  has_images          4000 non-null   int64
 12  has_videos          4000 non-null   int64
 13  readability_score   4000 non-null   float64
 14  num_shares          4000 non-null   int64
 15  num_comments        4000 non-null   int64
 16  political_bias      4000 non-null   object
 17  fact_check_rating   4000 non-null   object
 18  is_satirical        4000 non-null   int64
 19  trust_score         4000 non-null   int64
```



```
data.duplicated().sum()

np.int64(0)
```

8. Exploratory Data Analysis (EDA)

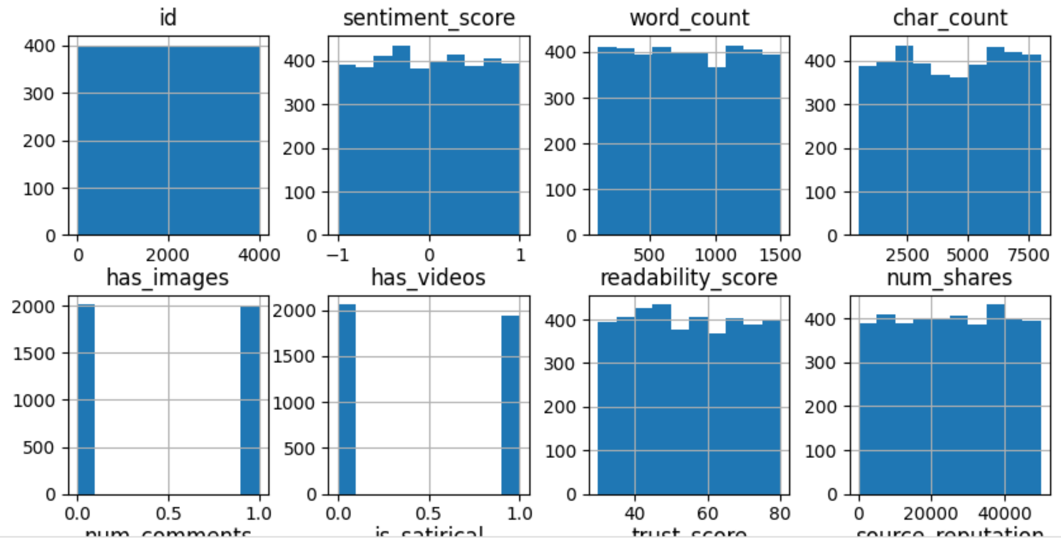
- *Used visual tools like histograms, word clouds, boxplots, and heatmaps to understand data distribution and feature relationships.*

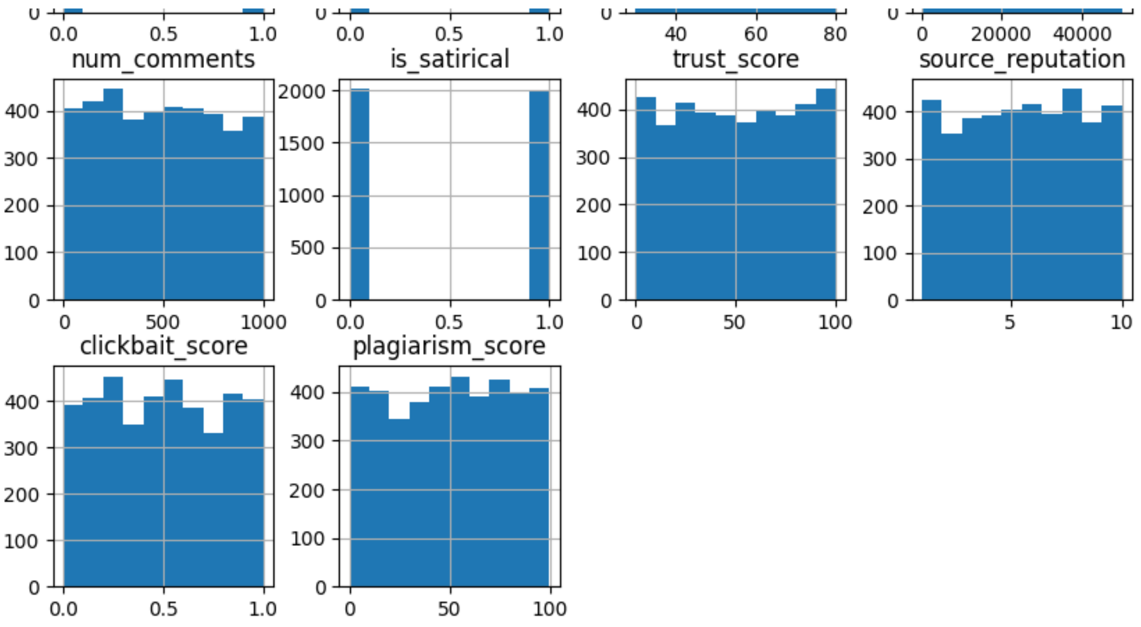
- *Revealed patterns such as frequently used words in fake vs. real news, and trends in article length and word frequency.*

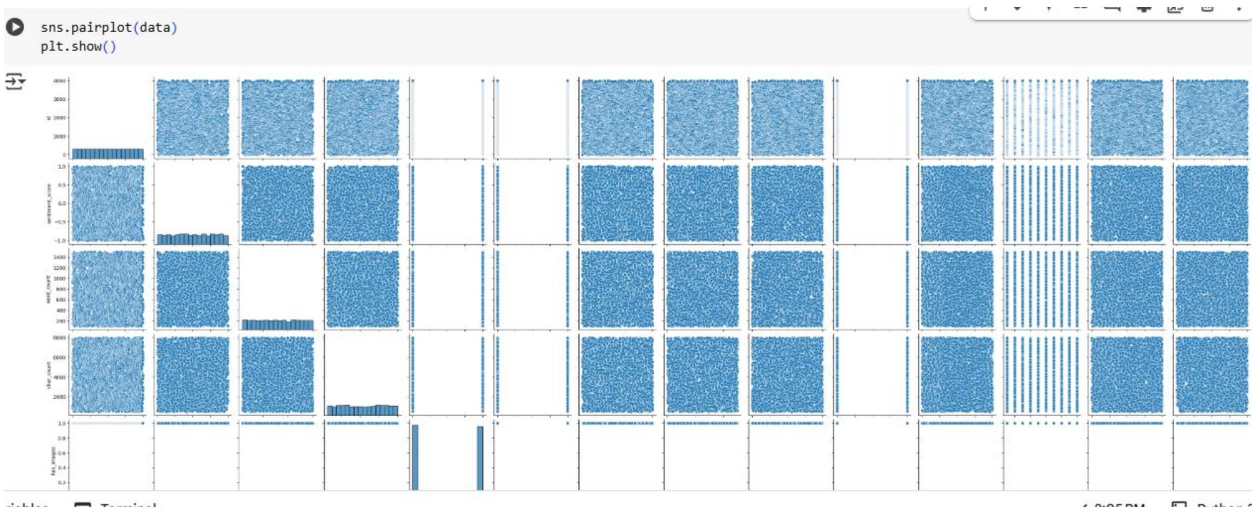
- *Key insights:*

- *Fake news articles often contain more sensational keywords.*
- *Real news tends to have more structured language and consistent word counts.*
- *Stopwords and punctuation usage varies significantly between real and fake articles*

```
data.hist(figsize=(10,10))
plt.show()
```



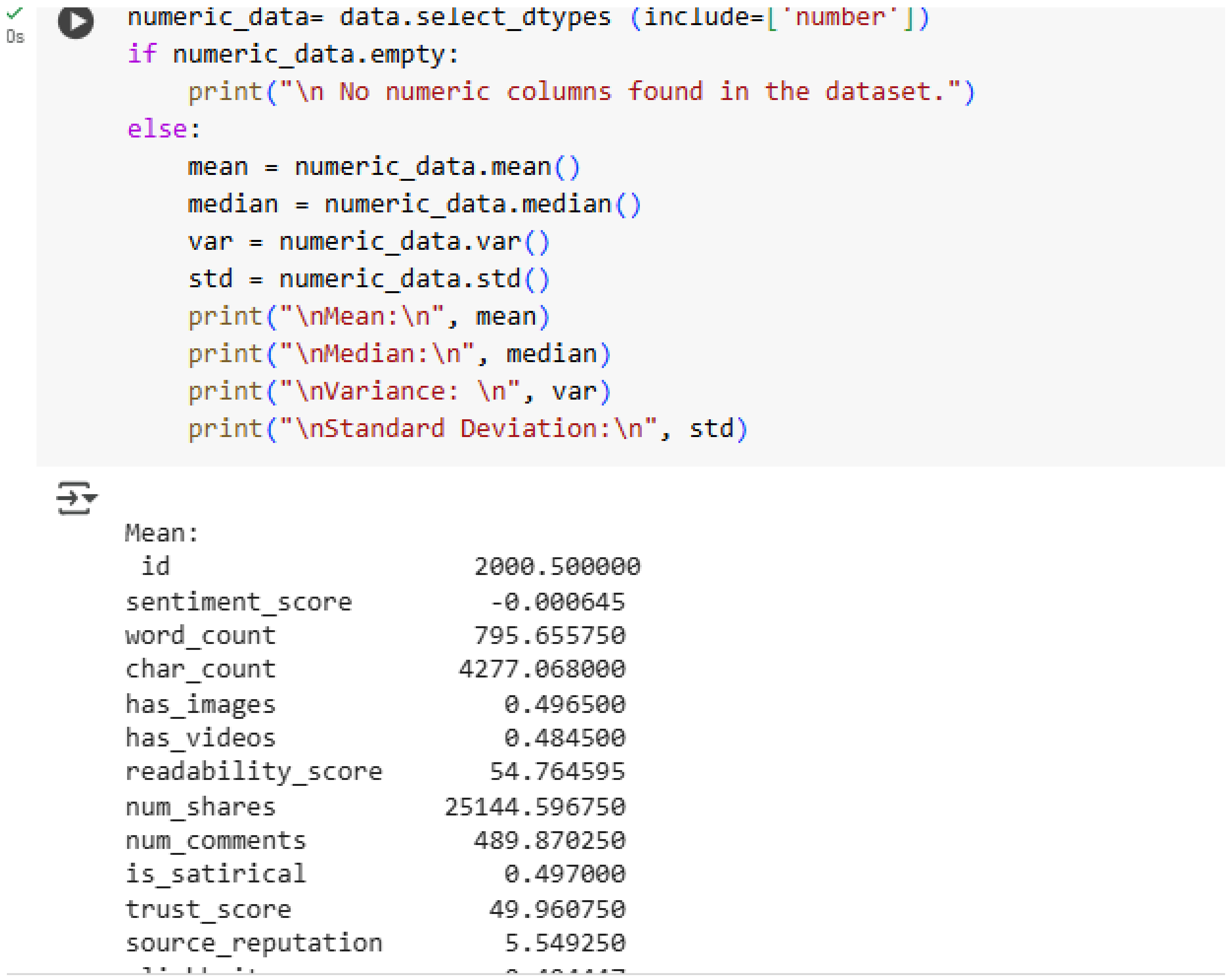




```
sns.pairplot(data)
plt.show()
```

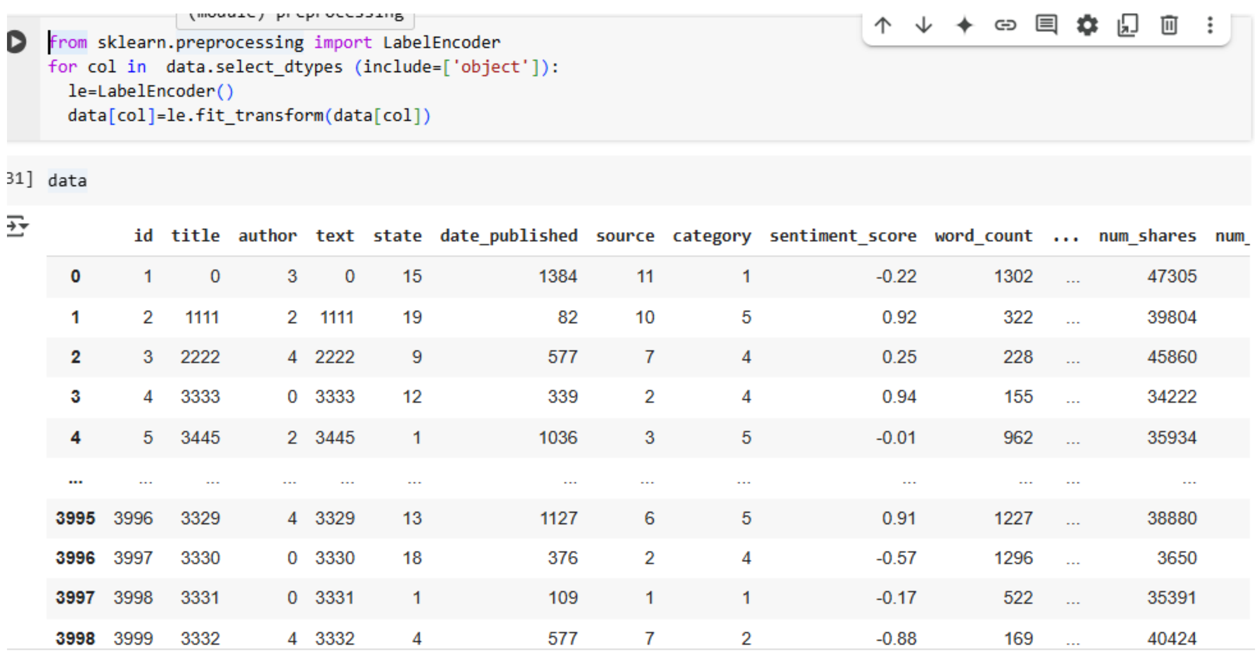
9. Feature Engineering

Feature engineering for fake news detection involves creating, selecting, and transforming relevant features such as text-based, sentiment, and source credibility features to improve model accuracy and transparency



```
numeric_data= data.select_dtypes (include=['number'])
if numeric_data.empty:
    print("\n No numeric columns found in the dataset.")
else:
    mean = numeric_data.mean()
    median = numeric_data.median()
    var = numeric_data.var()
    std = numeric_data.std()
    print("\nMean:\n", mean)
    print("\nMedian:\n", median)
    print("\nVariance: \n", var)
    print("\nStandard Deviation:\n", std)
```

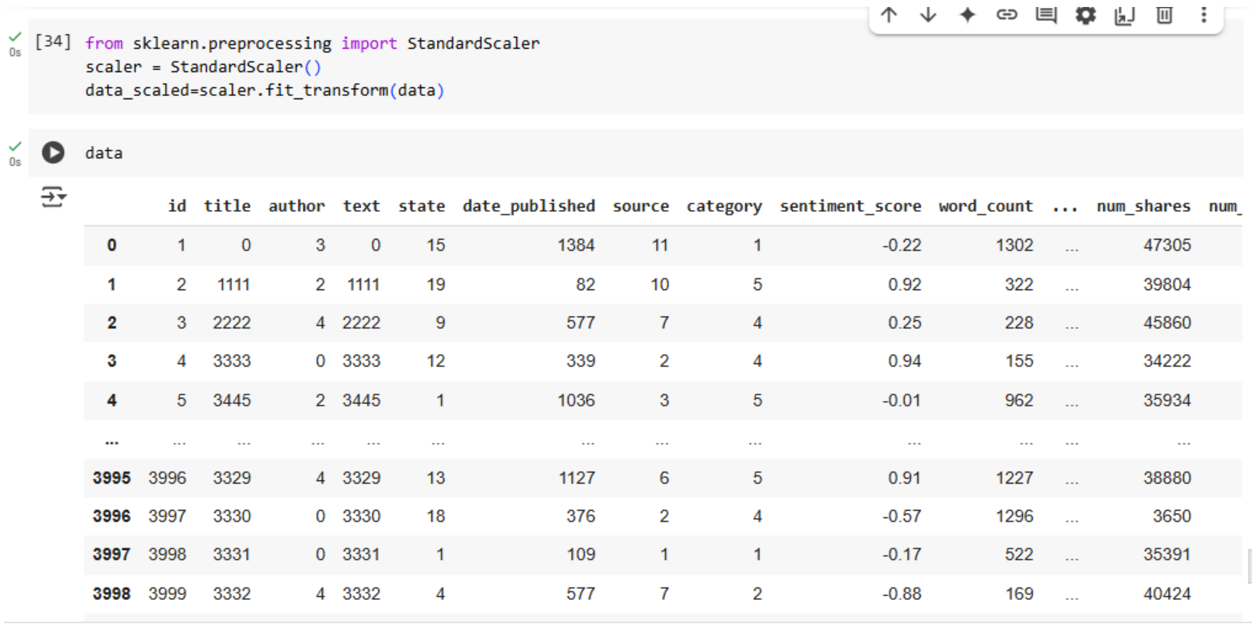
```
Mean:
 id                    2000.500000
sentiment_score         -0.000645
word_count             795.655750
char_count            4277.068000
has_images               0.496500
has_videos               0.484500
readability_score       54.764595
num_shares           25144.596750
num_comments           489.870250
is_satirical             0.497000
trust_score             49.960750
source_reputation        5.549250
```



```
from sklearn.preprocessing import LabelEncoder
for col in  data.select_dtypes (include=['object']):
  le=LabelEncoder()
  data[col]=le.fit_transform(data[col])
```

```
data
```

	id	title	author	text	state	date_published	source	category	sentiment_score	word_count	...	num_shares	num_
0	1	0	3	0	15	1384	11	1	-0.22	1302	...	47305	
1	2	1111	2	1111	19	82	10	5	0.92	322	...	39804	
2	3	2222	4	2222	9	577	7	4	0.25	228	...	45860	
3	4	3333	0	3333	12	339	2	4	0.94	155	...	34222	
4	5	3445	2	3445	1	1036	3	5	-0.01	962	...	35934	
...	...	...	...	...	...	...	...	...	...	...	...	...	
3995	3996	3329	4	3329	13	1127	6	5	0.91	1227	...	38880	
3996	3997	3330	0	3330	18	376	2	4	-0.57	1296	...	3650	
3997	3998	3331	0	3331	1	109	1	1	-0.17	522	...	35391	
3998	3999	3332	4	3332	4	577	7	2	-0.88	169	...	40424	



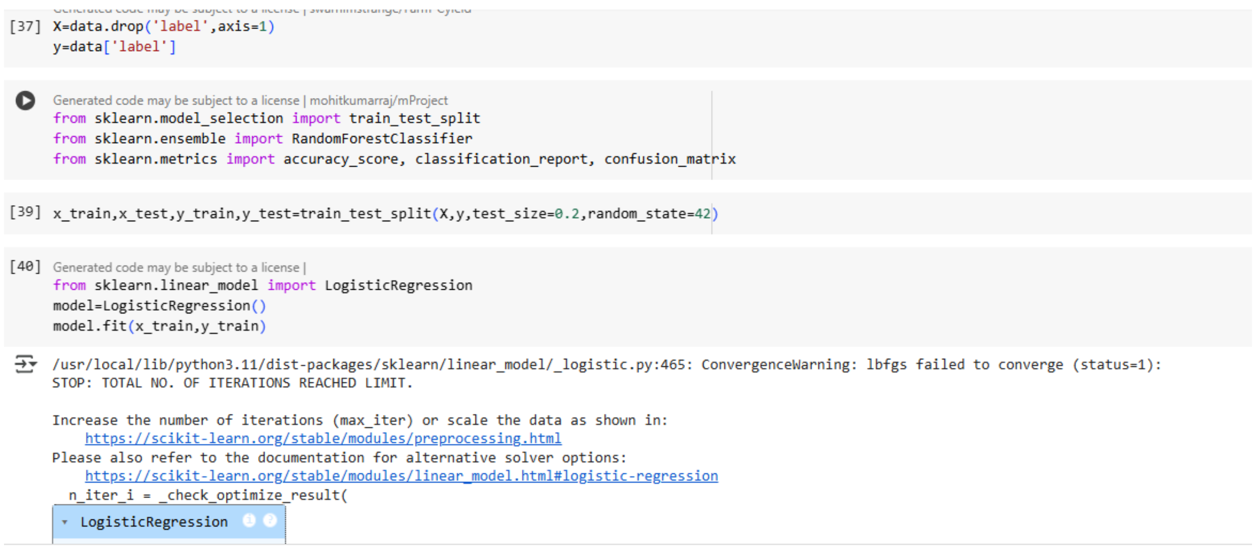
```
[34] from sklearn.preprocessing import StandardScaler
     scaler = StandardScaler()
     data_scaled=scaler.fit_transform(data)
```

```
data
```

	id	title	author	text	state	date_published	source	category	sentiment_score	word_count	...	num_shares	num_
0	1	0	3	0	15	1384	11	1	-0.22	1302	...	47305	
1	2	1111	2	1111	19	82	10	5	0.92	322	...	39804	
2	3	2222	4	2222	9	577	7	4	0.25	228	...	45860	
3	4	3333	0	3333	12	339	2	4	0.94	155	...	34222	
4	5	3445	2	3445	1	1036	3	5	-0.01	962	...	35934	
...	...	...	...	...	...	...	...	...	...	...	...	...	
3995	3996	3329	4	3329	13	1127	6	5	0.91	1227	...	38880	
3996	3997	3330	0	3330	18	376	2	4	-0.57	1296	...	3650	
3997	3998	3331	0	3331	1	109	1	1	-0.17	522	...	35391	
3998	3999	3332	4	3332	4	577	7	2	-0.88	169	...	40424	

10. Model Building

Model building involves experimenting with multiple models, starting with a baseline model and progressing to advanced models, such as decision trees, random forests, or deep learning, to identify the best performing model. Each model is chosen based on its suitability for the task, and screenshots of the model training outputs are included for performance evaluation.



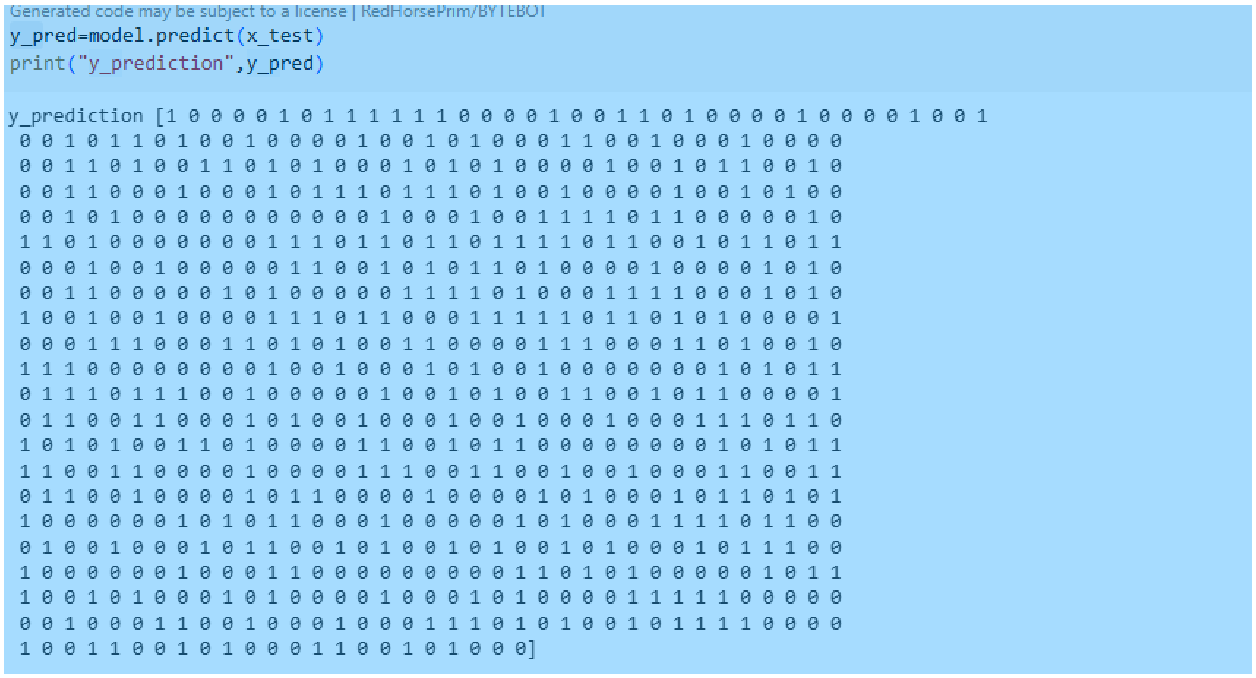
```
[37] X=data.drop('label',axis=1)
     y=data['label']
```

```
Generated code may be subject to a license | mohitkumarraj/mProject
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix
```

```
[39] x_train,x_test,y_train,y_test=train_test_split(X,y,test_size=0.2,random_state=42)
```

```
[40] Generated code may be subject to a license |
     from sklearn.linear_model import LogisticRegression
     model=LogisticRegression()
     model.fit(x_train,y_train)
```

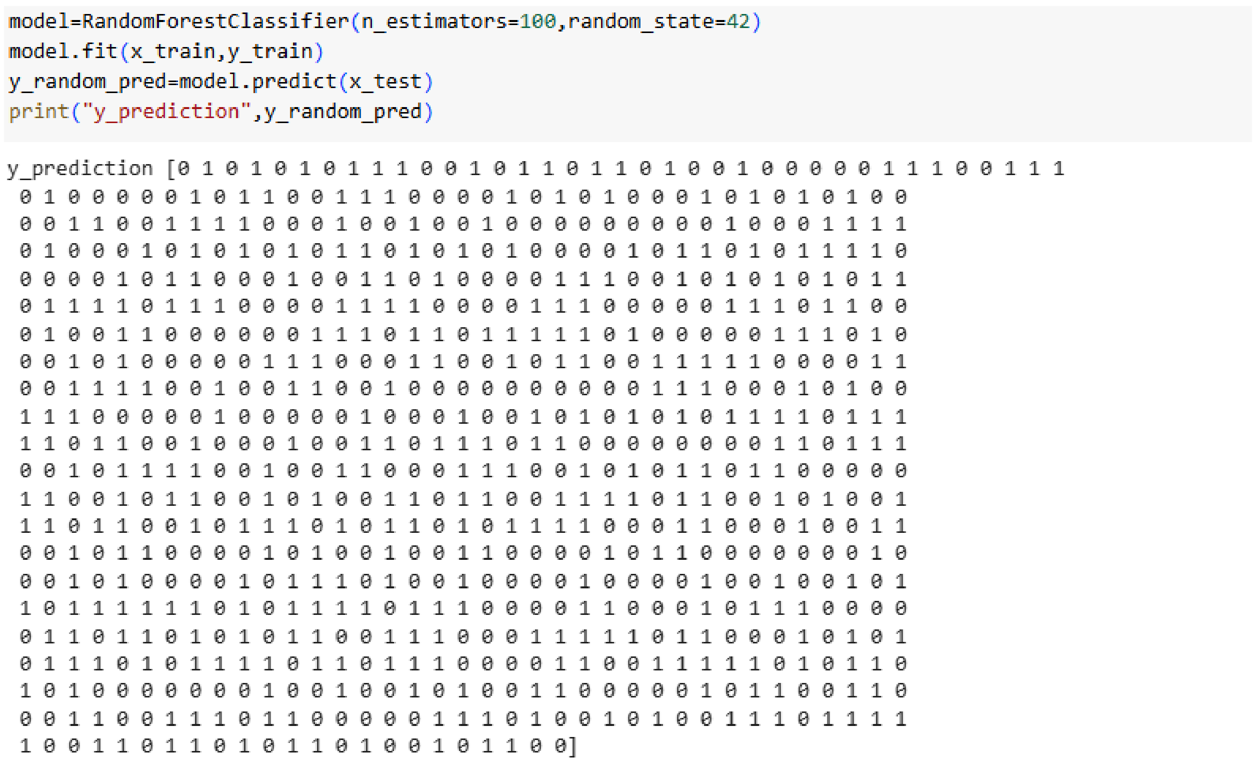
```
/usr/local/lib/python3.11/dist-packages/sklearn/linear_model/_logistic.py:465: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. OF ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
```



Generated code may be subject to a license | RedHorsePrim/BYTEBOT

```python
y_pred=model.predict(x_test)
print("y_prediction",y_pred)
```

```
y_prediction [1 0 0 0 0 1 0 1 1 1 1 1 1 0 0 0 0 1 0 0 1 1 0 1 0 0 0 0 1 0 0 0 0 1 0 0 1
 0 0 1 0 1 1 0 1 0 0 1 0 0 0 0 1 0 0 1 0 1 0 0 0 1 1 0 0 1 0 0 0 1 0 0 0 0
 0 0 1 1 0 1 0 0 1 1 0 1 0 1 0 0 0 1 0 1 0 1 0 0 0 0 1 0 0 1 0 1 1 0 0 1 0
 0 0 1 1 0 0 0 1 0 0 0 1 0 1 1 1 0 1 1 1 0 1 0 0 1 0 0 0 0 1 0 0 1 0 1 0 0
 0 0 1 0 1 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 1 0 0 1 1 1 1 0 1 1 0 0 0 0 0 1 0
 1 1 0 1 0 0 0 0 0 0 0 1 1 1 0 1 1 0 1 1 0 1 1 1 1 0 1 1 0 0 1 0 1 1 0 1 1
 0 0 0 1 0 0 1 0 0 0 0 0 1 1 0 0 1 0 1 0 1 1 0 1 0 0 0 0 1 0 0 0 0 1 0 1 0
 0 0 1 1 0 0 0 0 0 1 0 1 0 0 0 0 0 1 1 1 1 0 1 0 0 0 1 1 1 1 0 0 0 1 0 1 0
 1 0 0 1 0 0 1 0 0 0 0 1 1 1 0 1 1 0 0 0 1 1 1 1 1 0 1 1 0 1 0 1 0 0 0 0 1
 0 0 0 1 1 1 0 0 0 1 1 0 1 0 1 0 0 1 1 0 0 0 0 1 1 1 0 0 0 1 1 0 1 0 0 1 0
 1 1 1 0 0 0 0 0 0 0 0 1 0 0 1 0 0 0 1 0 1 0 0 1 0 0 0 0 0 0 0 1 0 1 0 1 1
 0 1 1 1 0 1 1 1 0 0 1 0 0 0 0 0 1 0 0 1 0 1 0 0 1 1 0 0 1 0 1 1 0 0 0 0 1
 0 1 1 0 0 1 1 0 0 0 1 0 1 0 0 1 0 0 0 1 0 0 1 0 0 0 1 0 0 0 1 1 1 0 1 1 0
 1 0 1 0 1 0 0 1 1 0 1 0 0 0 0 1 1 0 0 1 0 1 1 0 0 0 0 0 0 0 0 1 0 1 0 1 1
 1 1 0 0 1 1 0 0 0 0 1 0 0 0 0 1 1 1 0 0 1 1 0 0 1 0 0 1 0 0 0 1 1 0 0 1 1
 0 1 1 0 0 1 0 0 0 0 1 0 1 1 0 0 0 0 1 0 0 0 0 1 0 1 0 0 0 1 0 1 1 0 1 0 1
 1 0 0 0 0 0 0 1 0 1 0 1 1 0 0 0 1 0 0 0 0 0 1 0 1 0 0 0 1 1 1 1 0 1 1 0 0
 0 1 0 0 1 0 0 0 1 0 1 1 0 0 1 0 1 0 0 1 0 1 0 0 1 0 1 0 0 0 1 0 1 1 1 0 0
 1 0 0 0 0 0 0 1 0 0 0 1 1 0 0 0 0 0 0 0 0 0 1 1 0 1 0 1 0 0 0 0 0 1 0 1 1
 1 0 0 1 0 1 0 0 0 1 0 1 0 0 0 0 1 0 0 0 1 0 1 0 0 0 0 1 1 1 1 1 0 0 0 0 0
 0 0 1 0 0 0 1 1 0 0 1 0 0 0 1 0 0 0 1 1 1 0 1 0 1 0 0 1 0 1 1 1 1 0 0 0 0
 1 0 0 1 1 0 0 1 0 1 0 0 0 1 1 0 0 1 0 1 0 0 0]
```

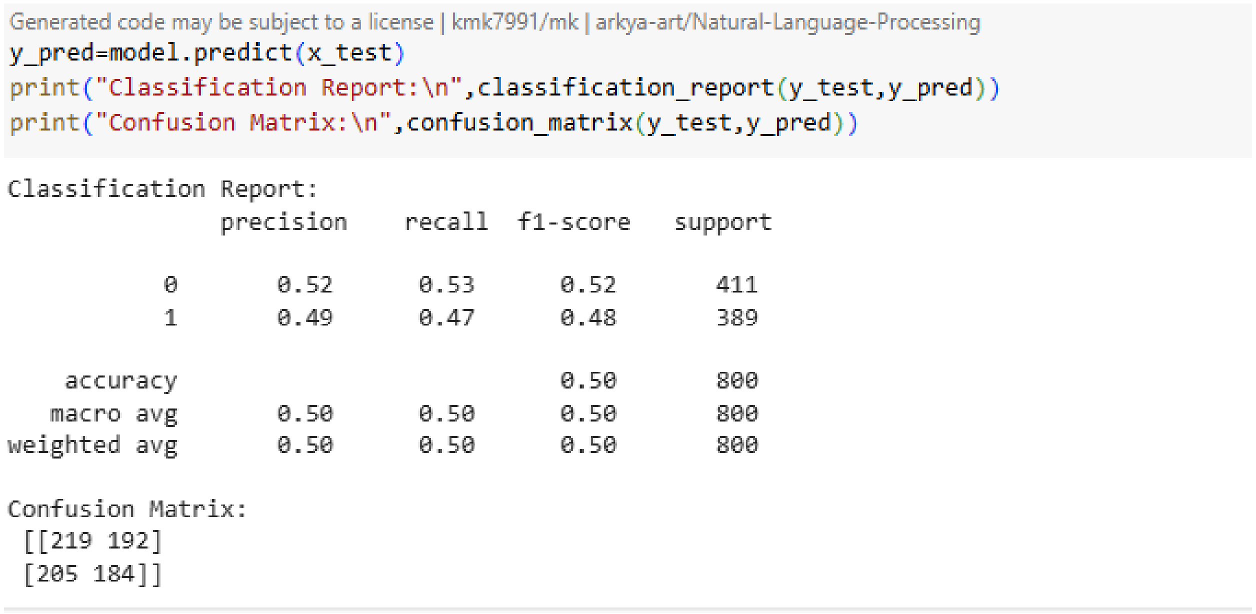


```python
model=RandomForestClassifier(n_estimators=100,random_state=42)
model.fit(x_train,y_train)
y_random_pred=model.predict(x_test)
print("y_prediction",y_random_pred)
```

```
y_prediction [0 1 0 1 0 1 0 1 1 1 0 0 1 0 1 1 0 1 1 0 1 0 0 1 0 0 0 0 0 1 1 1 0 0 1 1 1
 0 1 0 0 0 0 0 1 0 1 1 0 0 1 1 1 0 0 0 0 1 0 1 0 1 0 0 0 1 0 1 0 1 0 1 0 0
 0 0 1 1 0 0 1 1 1 1 0 0 0 1 0 0 1 0 0 1 0 0 0 0 0 0 0 0 0 1 0 0 0 1 1 1 1
 0 1 0 0 0 1 0 1 0 1 0 1 0 1 1 0 1 0 1 0 1 0 0 0 0 1 0 1 1 0 1 0 1 1 1 1 0
 0 0 0 0 1 0 1 1 0 0 0 1 0 0 1 1 0 1 0 0 0 0 1 1 1 0 0 1 0 1 0 1 0 1 0 1 1
 0 1 1 1 1 0 1 1 1 0 0 0 0 1 1 1 1 0 0 0 0 1 1 1 0 0 0 0 0 1 1 1 0 1 1 0 0
 0 1 0 0 1 1 0 0 0 0 0 0 1 1 1 0 1 1 0 1 1 1 1 1 0 1 0 0 0 0 0 1 1 1 0 1 0
 0 0 1 0 1 0 0 0 0 0 1 1 1 0 0 0 1 1 0 0 1 0 1 1 0 0 1 1 1 1 1 0 0 0 0 1 1
 0 0 1 1 1 1 0 0 1 0 0 1 1 0 0 1 0 0 0 0 0 0 0 0 0 0 1 1 1 0 0 0 1 0 1 0 0
 1 1 1 0 0 0 0 0 1 0 0 0 0 0 1 0 0 0 1 0 0 1 0 1 0 1 0 1 0 1 1 1 1 0 1 1 1
 1 1 0 1 1 0 0 1 0 0 0 1 0 0 1 1 0 1 1 1 0 1 1 0 0 0 0 0 0 0 0 1 1 0 1 1 1
 0 0 1 0 1 1 1 1 0 0 1 0 0 1 1 0 0 0 1 1 1 0 0 1 0 1 0 1 1 0 1 1 0 0 0 0 0
 1 1 0 0 1 0 1 1 0 0 1 0 1 0 0 1 1 0 1 1 0 0 1 1 1 1 0 1 1 0 0 1 0 1 0 0 1
 1 1 0 1 1 0 0 1 0 1 1 1 0 1 0 1 1 0 1 0 1 1 1 1 0 0 0 1 1 0 0 0 1 0 0 1 1
 0 0 1 0 1 1 0 0 0 0 1 0 1 0 0 1 0 0 1 1 0 0 0 0 1 0 1 1 0 0 0 0 0 0 0 1 0
 0 0 1 0 1 0 0 0 0 1 0 1 1 1 0 1 0 0 1 0 0 0 0 1 0 0 0 0 1 0 0 1 0 0 1 0 1
 1 0 1 1 1 1 1 1 0 1 0 1 1 1 1 0 1 1 1 0 0 0 0 1 1 0 0 0 1 0 1 1 1 0 0 0 0
 0 1 1 0 1 1 0 1 0 1 0 1 1 0 0 1 1 1 0 0 0 1 1 1 1 1 0 1 1 0 0 0 1 0 1 0 1
 0 1 1 1 0 1 0 1 1 1 1 0 1 1 0 1 1 1 0 0 0 0 1 1 0 0 1 1 1 1 1 0 1 0 1 1 0
 1 0 1 0 0 0 0 0 0 0 1 0 0 1 0 0 1 0 1 0 0 1 1 0 0 0 0 0 1 0 1 1 0 0 1 1 0
 0 0 1 1 0 0 1 1 1 0 1 1 0 0 0 0 0 1 1 1 0 1 0 0 1 0 1 0 0 1 1 1 0 1 1 1 1
 1 0 0 1 1 0 1 1 0 1 0 1 1 0 1 0 0 1 0 1 1 0 0]
```

11. Model Evaluation

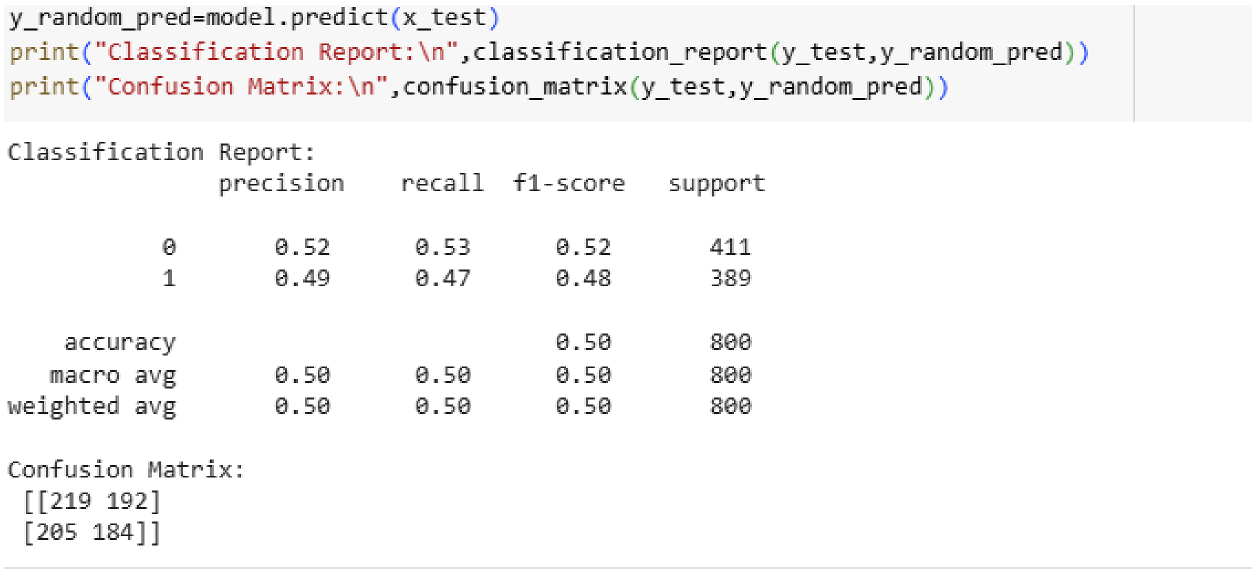
To assess the performance of our fake news detection model, we used various evaluation metrics including accuracy, F1-score, precision, recall, ROC-AUC, and RMSE. Visualizations such as confusion matrix and ROC curves were generated to better understand model performance. An error analysis was conducted to identify misclassifications, and a comparison table was created to evaluate different machine learning algorithms. All results and outputs are documented with relevant screenshots for clarity.



Generated code may be subject to a license | kmk7991/mk | arkya-art/Natural-Language-Processing

```
y_pred=model.predict(x_test)
print("Classification Report:\n",classification_report(y_test,y_pred))
print("Confusion Matrix:\n",confusion_matrix(y_test,y_pred))
```

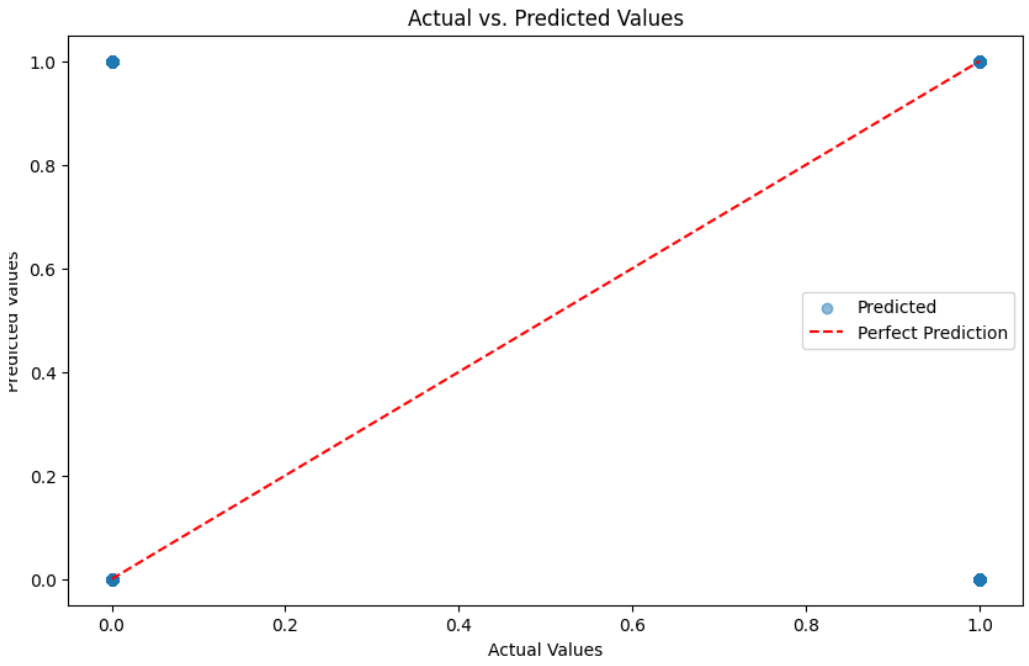
```
Classification Report:
               precision    recall  f1-score   support

           0       0.52      0.53      0.52       411
           1       0.49      0.47      0.48       389

    accuracy                           0.50       800
   macro avg       0.50      0.50      0.50       800
weighted avg       0.50      0.50      0.50       800

Confusion Matrix:
 [[219 192]
 [205 184]]
```



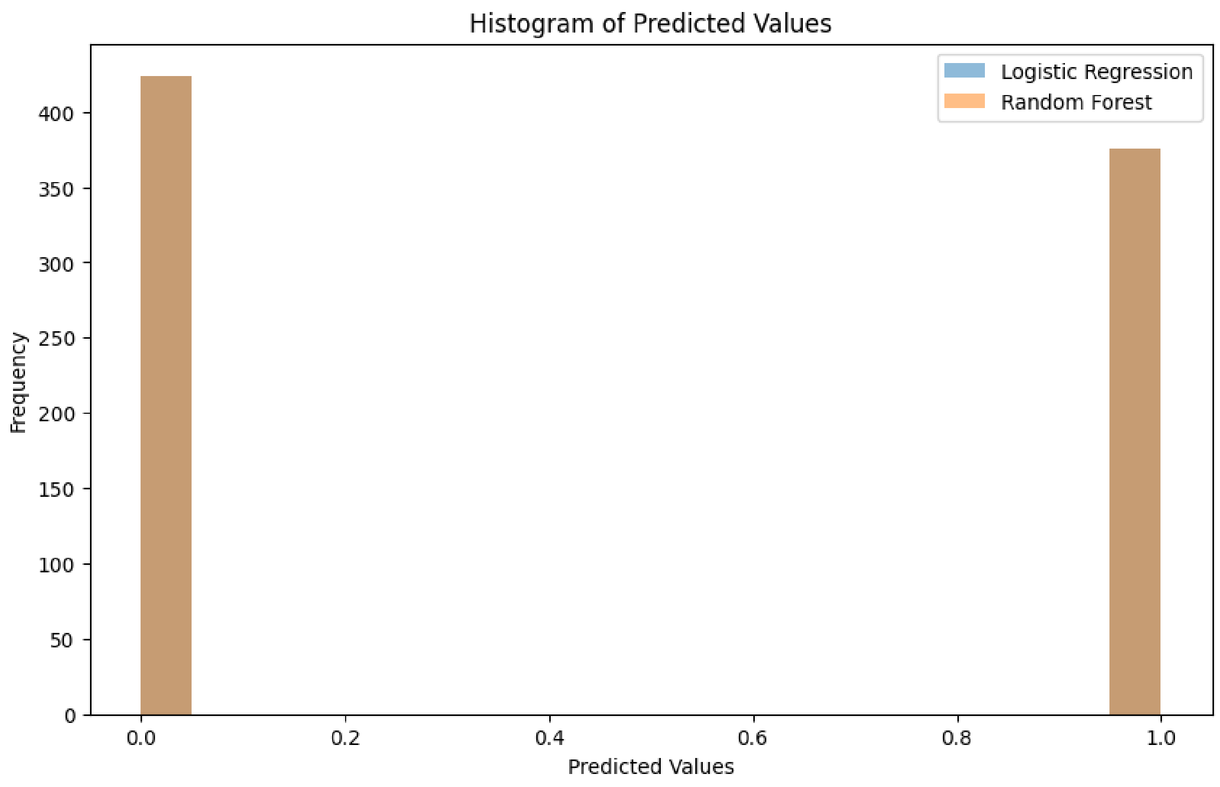
```
y_random_pred=model.predict(x_test)
print("Classification Report:\n",classification_report(y_test,y_random_pred))
print("Confusion Matrix:\n",confusion_matrix(y_test,y_random_pred))
```

```
Classification Report:
               precision    recall  f1-score   support

           0       0.52      0.53      0.52       411
           1       0.49      0.47      0.48       389

    accuracy                           0.50       800
   macro avg       0.50      0.50      0.50       800
weighted avg       0.50      0.50      0.50       800

Confusion Matrix:
 [[219 192]
 [205 184]]
```

```
plt.figure(figsize=(10, 6))
plt.scatter(y_test, y_pred, alpha=0.5, label='Predicted')
plt.plot([min(y_test), max(y_test)], [min(y_test), max(y_test)], linestyle='--',color='red', label='Perfect Prediction')
plt.xlabel('Actual Values')
plt.ylabel('Predicted Values')
plt.title('Actual vs. Predicted Values')
plt.legend()
plt.show()
```



```
plt.figure(figsize=(10, 6))
plt.hist(y_pred, bins=20, alpha=0.5, label='Logistic Regression')
plt.hist(y_random_pred, bins=20, alpha=0.5, label='Random Forest')
plt.xlabel('Predicted Values')
plt.ylabel('Frequency')
plt.title('Histogram of Predicted Values')
plt.legend()
plt.show()
```



12. Deployment

- *Deploy using a free platform:*

  Deployed using Streamlit Cloud for a simple UI and Gradio + Hugging Face Spaces for demo hosting
- *Include:*

  Method: Streamlit for UI, Gradio for interactive demo

13. Source code

#importing packages and libaries

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

#importing dataset

```
data=pd.read_csv("/fake_news_dataset.csv")
data.head()
```

#data preprocessing

```
data.drop_duplicates()
data
data.isnull().sum()
data.describe()
data.info()
data.duplicated().sum()
data.hist(figsize=(10,10))
plt.show()
sns.pairplot(data)
plt.show()
```

#EDA processing

```
numeric_data= data.select_dtypes (include=['number'])
if numeric_data.empty:
    print("\n No numeric columns found in the dataset.")
else:
    mean = numeric_data.mean()
    median = numeric_data.median()
    var = numeric_data.var()
    std = numeric_data.std()
    print("\nMean:\n", mean)
    print("\nMedian:\n", median)
    print("\nVariance: \n", var)
    print("\nStandard Deviation:\n", std)

from sklearn.preprocessing import LabelEncoder
for col in  data.select_dtypes (include=['object']):
  le=LabelEncoder()
  data[col]=le.fit_transform(data[col])
data

from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
data_scaled=scaler.fit_transform(data)
```

```
data
```

#model building

```
X=data.drop('label',axis=1)
y=data['label']

from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score, classification_report,
confusion_matrix

x_train,x_test,y_train,y_test=train_test_split(X,y,test_size=0.2,random_st
ate=42)

from sklearn.linear_model import LogisticRegression
model=LogisticRegression()
model.fit(x_train,y_train)

y_pred=model.predict(x_test)
print("y_prediction",y_pred)

model=RandomForestClassifier(n_estimators=100,random_state=42)
model.fit(x_train,y_train)
y_random_pred=model.predict(x_test)
print("y_prediction",y_random_pred)

y_pred=model.predict(x_test)
print("Classification Report:\n",classification_report(y_test,y_pred))
print("Confusion Matrix:\n",confusion_matrix(y_test,y_pred))

plt.figure(figsize=(10, 6))
plt.scatter(y_test, y_pred, alpha=0.5, label='Predicted')
plt.plot([min(y_test), max(y_test)], [min(y_test), max(y_test)],
linestyle='--',color='red', label='Perfect Prediction')
plt.xlabel('Actual Values')
plt.ylabel('Predicted Values')
plt.title('Actual vs. Predicted Values')
plt.legend()
plt.show()

plt.figure(figsize=(10, 6))
plt.hist(y_pred, bins=20, alpha=0.5, label='Logistic Regression')
plt.hist(y_random_pred, bins=20, alpha=0.5, label='Random Forest')
plt.xlabel('Predicted Values')
plt.ylabel('Frequency')
plt.title('Histogram of Predicted Values')
plt.legend()
plt.show()
```

14. Future scope

- *Multilingual Support – Extend fake news detection to regional and global languages.*
- *Real-Time Alerts – Enable instant detection and warning on social media platforms.*
- *Enhanced Accuracy – Use advanced models like BERT for better prediction.*
- *App/Extension Integration – Develop tools for easy public access and usage.*
- *User Feedback Loop – Incorporate feedback to continuously improve the system.*

GOOGLE COLLAB LINK:
https://colab.research.google.com/drive/1MO5zNXx0AhR5gv2XiwJE3nkl7O-cn5MU

15. Team Members and Roles:

NAME	*ROLES*
Krishna priya M	*Data collection,Cleaning and Overall Project Management*
Hinduja T	*Data Visualization and Interpretation*
Haritha Janani T	*Exploratory Data Analysis and Model Evaluation*

Kaviya I	*Model Building and Deployment*