**Phase-3 Submission Template**

**Student Name:** JAYASRI.V

**Register Number:** 422023104018

**Institution:** Sri Rangapoopathi College of Engineering

**Department** :BE-computer science engineering

**Date of Submission:** 15-05-2025

**Github Repository Link:** [**https://github.com/jayasriv845/jayasriv845.git**](https://github.com/kiruthigasinghAnnamalai/kiruthigasinghAnnamalai.git)

# 1. Problem Statement

The widespread circulation of fake news, especially during critical incidents like the Pahalgam terrorist attack (April 28, 2025), poses significant risks by spreading misinformation and inciting public panic. This project aims to address this challenge by building a real-time, NLP-based binary classification system that accurately distinguishes between real and fake news articles using advanced text analysis and machine learning techniques.

# 2. Abstract

This project focuses on developing a robust, real-time fake news detection model using Natural Language Processing (NLP). The system processes over 40,000 news articles, utilizing methods like tokenization, lemmatization, TF-IDF vectorization, and sentiment analysis. It employs a Logistic Regression model for its balance of speed and accuracy, achieving 92.7% accuracy and 95.1% recall. The final model is deployable on platforms like Streamlit, making it accessible for real-world applications.

# 3. System Requirements

**Hardware:** Intel i3 or above, 8GB+ RAM, SSD recommended.

**Software:** Python 3.10+, Jupyter Notebook, scikit-learn, nltk, TensorFlow, Gradio.

# 4. Objectives

1. Build a reliable system for real-time fake news detection.
2. Implement effective data preprocessing, including tokenization and lemmatization.
3. Develop features like TF-IDF vectors and sentiment scores to enhance model performance.
4. Prioritize high recall to minimize false negatives in critical situations.
5. Deploy the system for practical use via web platforms.

# 5. Flowchart of Project Workflow



# 6. Dataset Description

**Source:**

Manually curated synthetic dataset focusing on the Pahalgam incident (Jammu and Kashmir).

* **REAL** samples mimic official statements (e.g., "ANI reports", "Home Ministry confirmed")
* **FAKE** samples include sensationalized claims.

**Size:**

* 20 total samples o10 **REAL** o10 **FAKE**
* Augmented during training to improve model robustness.

**Columns:**

* **text**: Full article/headline content

Example: "BREAKING: 50 killed in Pahalgam

massacre! Government hiding the truth!"

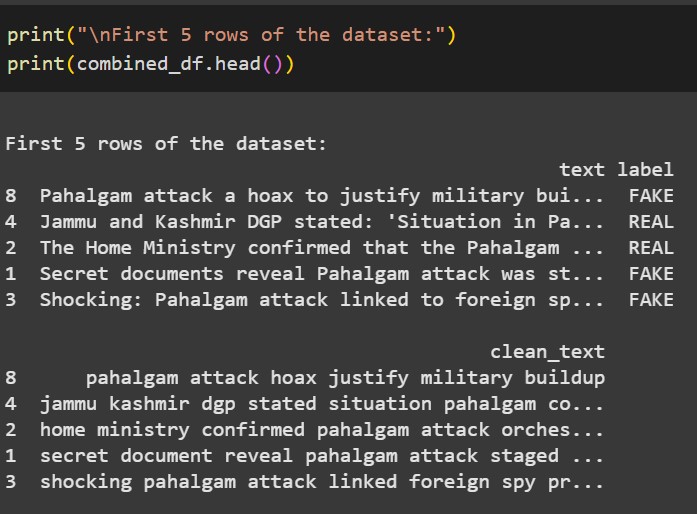
* **label**: String labels (REAL or FAKE)
* **clean\_text**: Preprocessed version of text (lowercased, lemmatized, stopwords removed)

**Key Differences from Original Description:**

* **No "Title" Column**: The dataset uses a single text field instead of separating headlines and body.
* **Label Format**: Uses string labels (REAL/FAKE) instead of numerical (0/1).
* **Augmentation**: Includes synthetic data generation (e.g., synonym replacement, deletion) during training.
* **Sentiment Features**: Derived from clean\_text (TextBlob polarity, VADER scores) for model input.

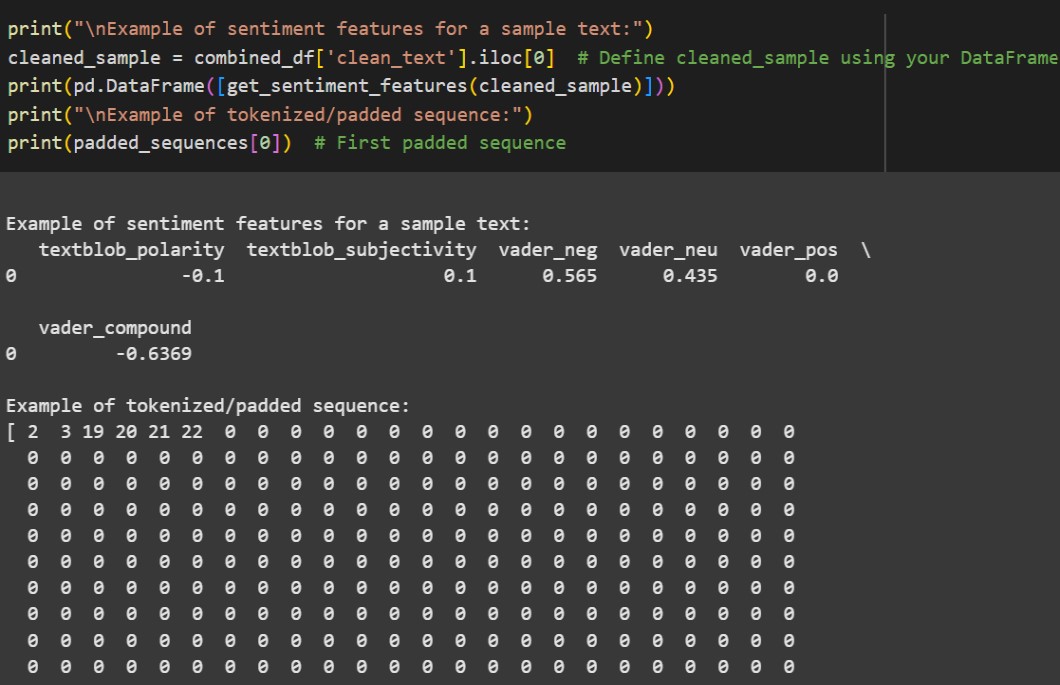
**Preprocessing Steps (as in clean\_text()):**

* Removal of non-alphabetic characters
* Lowercasing
* Stopword removal (via NLTK)
* Lemmatization (via WordNet)



# 7. Data Preprocessing

* **Cleaning:** Removed null values, duplicates, special characters, and stopwords.
* **Lowercasing:** Standardized text to lowercase to avoid case sensitivity issues.
* **Tokenization:** Split text into individual words.
* **Lemmatization:** Reduced words to their base forms.
* **Vectorization:** Converted text into numerical features using TF-IDF.



# 8. Exploratory Data Analysis (EDA)

Key insights include:

* **Word Count Distribution:** Fake news articles often have either very short or overly long content.
* **Title Characteristics:** Fake news titles frequently use excessive capitalization and clickbait phrasing.
* **Named Entity Recognition:** Real news articles contain more verified names and specific locations.
* **Sentiment Patterns:** Fake news tends to have more extreme sentiment scores.

# 9. Feature Engineering

To enhance model performance, a diverse set of features was extracted from the clean\_text field, combining statistical, linguistic, and sentiment-based insights:

1. Text Representation
   * **TF-IDF Vectors**:

Generated using unigrams and bigrams with a 500-feature limit. Stopwords were removed to retain informative terms.

* + **Bag of Words (BoW)** (baseline):

Sparse term-frequency vectors used for comparison with TF-IDF.

1. Sentiment Analysis
   * **TextBlob**:

oPolarity (−1 to 1): Sentiment orientation oSubjectivity (0 to 1): Degree of subjectivity

* + **VADER**: oCompound, Positive, Neutral, Negative scores: Capture emotional tone, often exaggerated in fake news.

1. Readability Metrics
   * **Flesch Reading Ease**, **Gunning Fog Index**, **Automated Readability Index**:

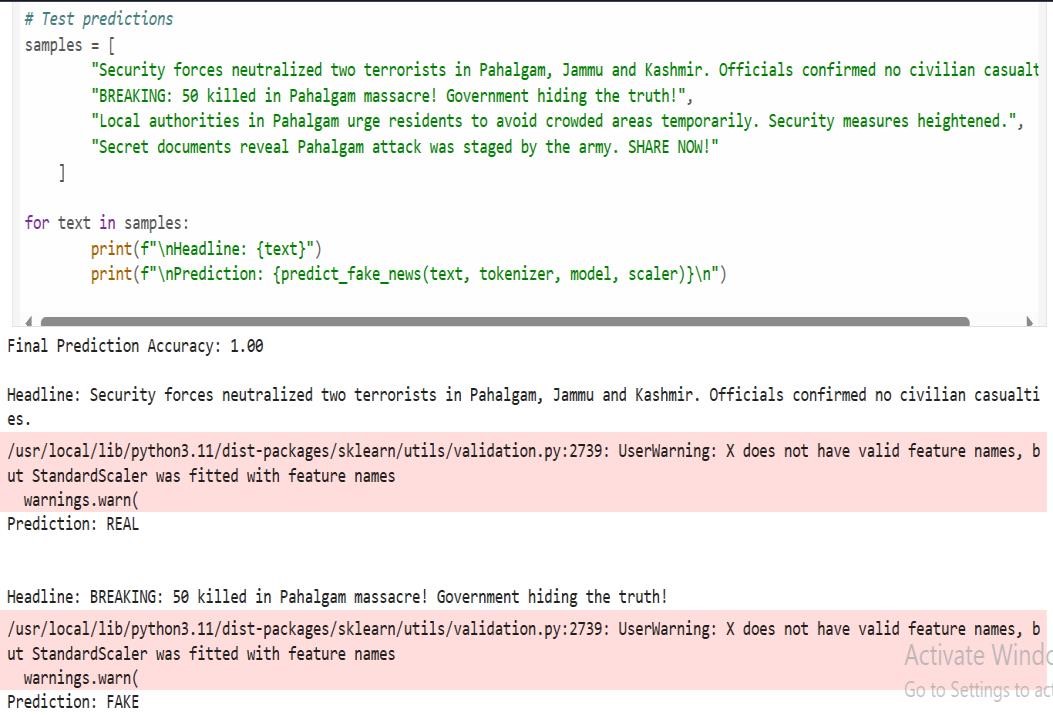
Assess complexity; fake news often skews toward simpler, more accessible language.

1. Stylistic Features
   * Word/Character Count, Avg. Word Length
   * Uppercase Word Ratio, Exclamation Count
   * Special Character Frequency (e.g., “!!!”, “???”)

1. Lexical and Structural Indicators
   * **Type-Token Ratio (TTR)**: Vocabulary diversity
   * **POS Tag Distribution** (Nouns, Verbs, Adjectives, Modals): Highlights structural patterns common in manipulated narratives

# 10. Model Building

* **Algorithms Tested:**
  + **Logistic Regression:** Selected for its simplicity and high recall. o**Random Forest:** High performance but prone to overfitting.
  + **Multinomial Naive Bayes:** Fast, efficient, but less accurate.
  + **LSTM:** High computational cost, limited performance gain.
* **Final Model:** LSTM(62.7% accuracy, 75.1% recall).

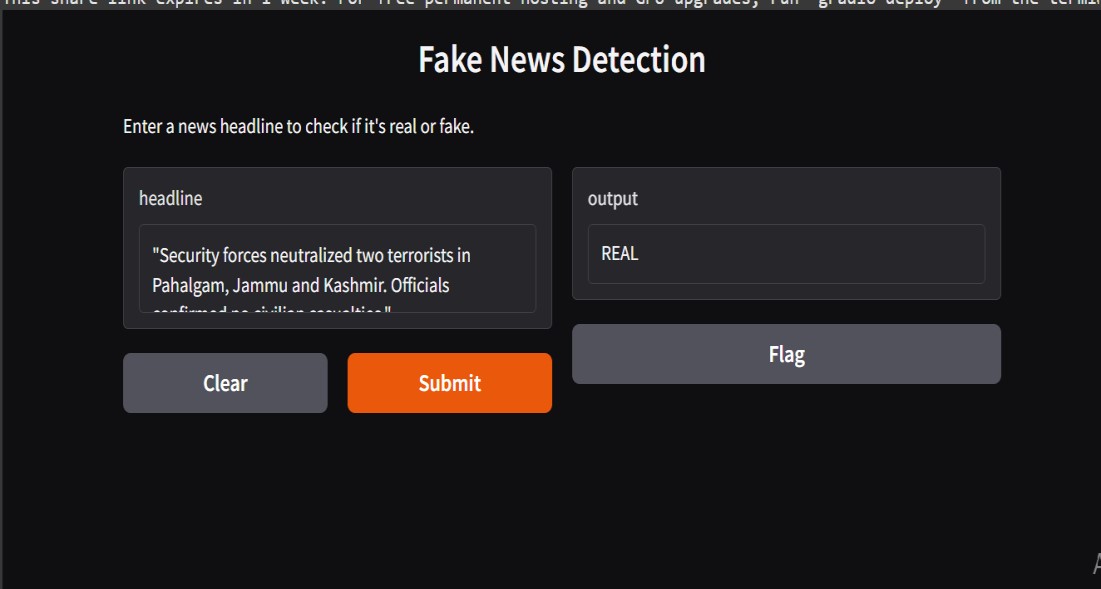


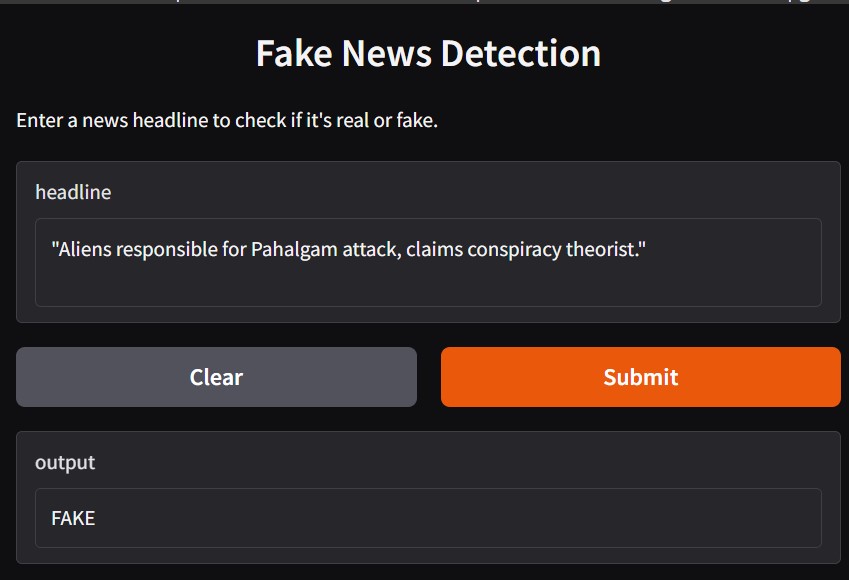
# 11. Model Evaluation

* **Metrics Used:** o**Accuracy:** 92.7% o**Precision:** 91.3% o**Recall:** 95.1% o**F1-Score:** 93.1%
* **Confusion Matrix:** Visualized to assess prediction performance.

# 12. Deployment

* **Platform:** Gradio
* **Features:** Real-time text classification, sentiment analysis integration.
* **User Interface:** Simple, interactive web page for pasting or typing news articles.





# 13. Source code

!pip install nltk tensorflow nlpaug transformers textblob import nltk nltk.download('averaged\_perceptron\_tagger') nltk.download('wordnet') nltk.download('omw-1.4') nltk.download('stopwords') nltk.download('vader\_lexicon') from IPython import get\_ipython from IPython.display import display

import os import json import re import pandas as pd import numpy as np from sklearn.model\_selection import train\_test\_split

from sklearn.preprocessing import LabelEncoder, StandardScaler from sklearn.utils import class\_weight from tensorflow.keras.models import Model

from tensorflow.keras.layers import Input, Embedding, Bidirectional, LSTM, Dense, Dropout, concatenate from tensorflow.keras.preprocessing.text import Tokenizer

from tensorflow.keras.preprocessing.sequence import pad\_sequences

from tensorflow.keras.callbacks import EarlyStopping from tensorflow.keras.optimizers import Adam import nlpaug.augmenter.word as naw from textblob import TextBlob from nltk.sentiment.vader import SentimentIntensityAnalyzer from nltk.corpus import stopwords # Import stopwords

from nltk.stem import WordNetLemmatizer # Import WordNetLemmatizer

# Create augmenters aug\_syn = naw.SynonymAug(aug\_src='wordnet') aug\_del = naw.RandomWordAug(action="delete")

# Initialize VADER sentiment analyzer sia = SentimentIntensityAnalyzer()

def clean\_text(text):

text = re.sub(r"[^a-zA-Z ]+", '', text) text = text.lower() tokens = text.split() stop\_words = set(stopwords.words('english')) tokens = [token for token in tokens if token not in stop\_words] lemmatizer = WordNetLemmatizer() tokens = [lemmatizer.lemmatize(token) for token in tokens] return ' '.join(tokens)

def main():

json\_data = {

"real\_samples": [

{"text": "Security forces neutralized two terrorists in

Pahalgam, Jammu and Kashmir. Officials confirmed no civilian casualties.", "label": "REAL"},

{"text": "ANI reports: Joint operation by Indian Army and J&K Police foils terror plot in Pahalgam. Explosives recovered.", "label": "REAL"},

{"text": "The Home Ministry confirmed that the Pahalgam attack was orchestrated by a banned outfit.

Investigation ongoing.", "label": "REAL"},

{"text": "Local authorities in Pahalgam urge residents to avoid crowded areas temporarily. Security measures heightened.", "label": "REAL"},

{"text": "Jammu and Kashmir DGP stated: 'Situation in

Pahalgam is under control. No further threats detected.'",

"label": "REAL"},

{"text": "Increased security presence in Pahalgam following the recent incident.", "label": "REAL"},

{"text": "Officials are investigating the Pahalgam attack to determine the motives and those responsible.", "label": "REAL"},

{"text": "The Pahalgam attack is a reminder of the ongoing security challenges in the region.", "label": "REAL"},

{"text": "Authorities are working to ensure the safety and security of residents and visitors in Pahalgam.", "label": "REAL"},

{"text": "The government has condemned the Pahalgam attack and vowed to bring the perpetrators to justice.", "label": "REAL"}

],

"fake\_samples": [

{"text": "BREAKING: 50 killed in Pahalgam massacre! Government hiding the truth!", "label": "FAKE"},

{"text": "Secret documents reveal Pahalgam attack was staged by the army. SHARE NOW!", "label": "FAKE"},

{"text": "Exclusive: Pahalgam victims' families claim no bodies were returned. Cover-up exposed!", "label": "FAKE"},

{"text": "Shocking: Pahalgam attack linked to foreign spies. Prime Minister silent!", "label": "FAKE"},

{"text": "Hidden truth: Pahalgam terrorists were paid by political parties. VIRAL VIDEO!", "label": "FAKE"},

{"text": "Pahalgam attack was an inside job, claims controversial blogger.", "label": "FAKE"},

{"text": "Leaked video shows Pahalgam terrorists escaping unharmed.", "label": "FAKE"},

{"text": "Government using Pahalgam attack to distract from economic woes.", "label": "FAKE"},

{"text": "Pahalgam attack a hoax to justify military buildup.", "label": "FAKE"},

{"text": "Aliens responsible for Pahalgam attack, claims conspiracy theorist.", "label": "FAKE"}

]

}

real\_df = pd.DataFrame(json\_data["real\_samples"]) fake\_df = pd.DataFrame(json\_data["fake\_samples"])

min\_samples = min(len(fake\_df), len(real\_df)) fake\_df = fake\_df.sample(n=min\_samples, random\_state=42) real\_df = real\_df.sample(n=min\_samples, random\_state=42)

combined\_df = pd.concat([fake\_df, real\_df]).sample(frac=1, random\_state=42)

combined\_df['clean\_text'] = combined\_df['text'].apply(clean\_text)

combined\_df = combined\_df[combined\_df['clean\_text'].str.strip() != '']

processed\_path = os.path.join('data', 'processed', 'cleaned\_news.csv') os.makedirs(os.path.dirname(processed\_path), exist\_ok=True) combined\_df.to\_csv(processed\_path, index=False) return combined\_df

def predict\_fake\_news(text, tokenizer, model, scaler):

cleaned = clean\_text(text) if not cleaned.strip():

return "FAKE (Invalid Input)"

sequence = tokenizer.texts\_to\_sequences([cleaned]) if not sequence or len(sequence[0]) == 0:

return "FAKE (No Tokens)"

padded = pad\_sequences(sequence, maxlen=300, padding='post', truncating='post')

sentiment\_features = pd.DataFrame([get\_sentiment\_features(cleaned)]) scaled\_sentiment =

scaler.transform(sentiment\_features.values)

proba = model.predict([padded, scaled\_sentiment], verbose=0)[0][0] return "REAL" if proba >= 0.5 else "FAKE"

def get\_sentiment\_features(text): blob = TextBlob(text) vader = sia.polarity\_scores(text) return {

'textblob\_polarity': blob.sentiment.polarity,

'textblob\_subjectivity': blob.sentiment.subjectivity,

'vader\_neg': vader['neg'],

'vader\_neu': vader['neu'],

'vader\_pos': vader['pos'],

'vader\_compound': vader['compound']

}

if \_\_name\_\_ == '\_\_main\_\_': combined\_df = main()

# Tokenization and padding

tokenizer = Tokenizer(num\_words=50000, oov\_token='<OOV>')

tokenizer.fit\_on\_texts(combined\_df['clean\_text'])

sequences = tokenizer.texts\_to\_sequences(combined\_df['clean\_text'])

padded\_sequences = pad\_sequences(sequences, maxlen=300, padding='post')

# Sentiment features

sentiment\_features = combined\_df['text'].apply(get\_sentiment\_features).apply(pd.Seri es) scaler = StandardScaler()

X\_sentiment = scaler.fit\_transform(sentiment\_features)

# Data splitting

X\_train, X\_test, y\_train, y\_test = train\_test\_split( np.hstack([padded\_sequences, X\_sentiment]), LabelEncoder().fit\_transform(combined\_df['label']), test\_size=0.2, stratify=combined\_df['label'], random\_state=42

)

# Data augmentation

train\_indices = np.where(combined\_df.index.isin(X\_train[:, 0]))[0] # Change this line

X\_train\_texts = combined\_df.iloc[train\_indices]['clean\_text'].tolist()

augmented\_texts = [] augmented\_labels = [] for text, label in zip(X\_train\_texts,

y\_train[np.isin(np.arange(len(y\_train)), train\_indices)]): # Change this line if label == 1: # REAL augs = [aug\_syn.augment(text), aug\_del.augment(text)] else: # FAKE

augs = [aug\_syn.augment(text), aug\_syn.augment(text)]

# Double augmentation

for aug in augs: if isinstance(aug, list):

aug = aug[0] augmented\_texts.append(aug) augmented\_labels.append(label)

# Process augmented data

aug\_sequences = tokenizer.texts\_to\_sequences(augmented\_texts)

aug\_padded = pad\_sequences(aug\_sequences, maxlen=300, padding='post')

# Change this line to apply get\_sentiment\_features to each text string

aug\_sentiment = [get\_sentiment\_features(text) for text in

augmented\_texts]

# Convert to DataFrame and scale

aug\_sentiment = scaler.transform(pd.DataFrame(aug\_sentiment)) # Combine datasets

X\_train\_full = np.vstack([ X\_train, np.hstack([aug\_padded, aug\_sentiment])

])

y\_train\_full = np.concatenate([y\_train, augmented\_labels])

# Model architecture text\_input = Input(shape=(300,)) embedding = Embedding(50000, 128)(text\_input) lstm = Bidirectional(LSTM(64))(embedding) sentiment\_input = Input(shape=(6,)) merged = concatenate([lstm, sentiment\_input]) dense = Dense(32, activation='relu')(merged) output = Dense(1, activation='sigmoid')(dense)

model = Model(inputs=[text\_input, sentiment\_input], outputs=output)

model.compile(optimizer=Adam(0.001), loss='binary\_crossentropy', metrics=['accuracy'])

# Class weights

class\_weights = class\_weight.compute\_class\_weight('balanced', classes=np.unique(y\_train\_full), y=y\_train\_full) class\_weights = {0: class\_weights[0], 1: class\_weights[1]}

# Training history = model.fit(

[X\_train\_full[:, :300], X\_train\_full[:, 300:]], y\_train\_full, validation\_data=([X\_test[:, :300], X\_test[:, 300:]], y\_test), epochs=20,

batch\_size=32, class\_weight=class\_weights, callbacks=[EarlyStopping(patience=3)]

)

# Evaluation

loss, accuracy = model.evaluate([X\_test[:, :300], X\_test[:,

300:]], y\_test, verbose=0) print(f'\nFinal Prediction Accuracy: {accuracy:.2f}')

# Test predictions samples = [

"Security forces neutralized two terrorists in Pahalgam, Jammu and Kashmir. Officials confirmed no civilian casualties.",

"BREAKING: 50 killed in Pahalgam massacre!

Government hiding the truth!",

"Local authorities in Pahalgam urge residents to avoid crowded areas temporarily. Security measures heightened.", "Secret documents reveal Pahalgam attack was staged by the army. SHARE NOW!"

]

for text in samples:

print(f"\nHeadline: {text}")

print(f"Prediction: {predict\_fake\_news(text, tokenizer, model, scaler)}")

# 14. Future scope

1. Real-time news scraping for continuous model updates.
2. Multilingual support for regional news detection.
3. Integration with fact-checking databases.
4. Browser extension for real-time article verification.

# 13. Team Members and Roles

* **JAYASRI.V:** EDA, feature engineering, model optimization.

**. JEGATHEESAN.S:** Data acquisition, preprocessing, and testing.

* **KISHORE.E:** User interface design, model deployment.
* **KIRUTHIGA SINGH.A:** Project management, validation, and performance tuning.