| EXP NO: 9 | MINI PROJECT - TIME SERIES-BASED STOCK PRICE FORECASTING WITH LINEAR REGRESSION |
|---|---|

**AIM:**
To detect and classify news articles as *Fake* or *Real* using Natural Language Processing (NLP) techniques and the **Random Forest** algorithm, and to evaluate the model's performance in accurately identifying and distinguishing misleading information from authentic news content..

**ALGORITHM:**

1. **Data Collection:**
Obtain fake and real news datasets (e.g., Fake.csv and True.csv).
2. **Data Labeling:**
Assign labels — Fake news = 0, Real news = 1.
3. **Data Merging:**
Combine both datasets into a single dataframe with titles and labels.
4. **Data Preprocessing:**
   - Convert all text to lowercase.
   - Remove punctuation marks.
   - Tokenize the text (split into words).
   - Remove English stopwords (e.g., "the", "is", "and").
   - Join the cleaned words back into a single string.
5. **Feature Extraction (TF-IDF):**
Convert cleaned text data into numerical features using **TF-IDF Vectorizer** with a maximum of 5000 features.
6. **Train-Test Split:**
Divide the dataset into **training (80%)** and **testing (20%)** sets.
7. **Model Training:**
Train a **Logistic Regression** model using the TF-IDF feature vectors and their corresponding labels.
8. **Model Evaluation:**
   - Predict labels for the test data.
   - Calculate **accuracy score** and optionally display a **classification report**.
9. **Model Saving:**
Save the trained **model** (model.pkl) and **TF-IDF vectorizer** (tfidf.pkl) using **pickle** for future use.
10. **Model Deployment (Streamlit App):**
    - Load the saved model and TF-IDF vectorizer.
    - Take user input (news text or headline).

- Preprocess the input text using the same preprocessing steps.
- Transform the input text using TF-IDF.
- Predict whether the news is **FAKE (0)** or **REAL (1)**.
- Display the result and confidence score in the Streamlit interface.

**11. Output:**

The system displays whether the entered news is **FAKE** or **REAL**, along with the confidence percentage

**CODE:**

**Ex.ipynb**

```
import pandas as pd
import string
import pickle
import nltk
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

nltk.download('stopwords')
from nltk.corpus import stopwords
stop_words = set(stopwords.words('english'))

def preprocess(text):
    text = text.lower()
    text = "".join([ch for ch in text if ch not in string.punctuation])
    words = text.split()
    words = [word for word in words if word not in stop_words]
    return " ".join(words)

# Load and label data
fake = pd.read_csv("Fake.csv")
real = pd.read_csv("True.csv")
fake["label"] = 0
real["label"] = 1

df = pd.concat([fake[["title", "label"]], real[["title", "label"]]], ignore_index=True)
df.dropna(inplace=True)
df["cleaned"] = df["title"].apply(preprocess)
```

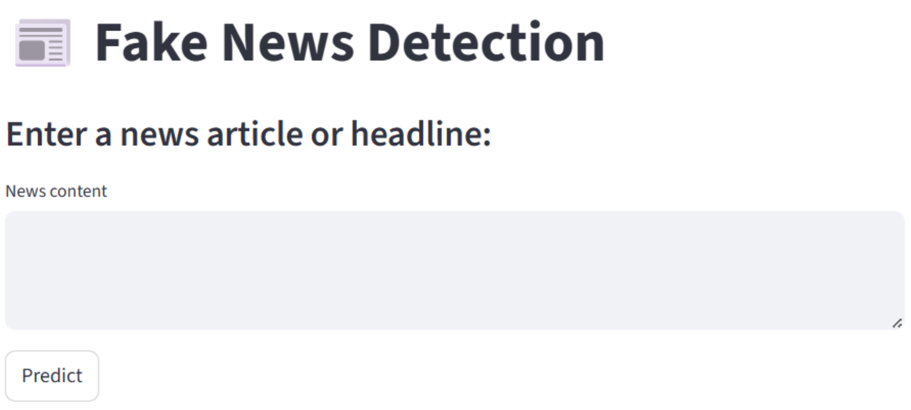
```
# Train-test split
X_train, X_test, y_train, y_test = train_test_split(df["cleaned"], df["label"], test_size=0.2,
random_state=42)

# TF-IDF and model
tfidf = TfidfVectorizer(max_features=5000)
X_train_tfidf = tfidf.fit_transform(X_train)
X_test_tfidf = tfidf.transform(X_test)

model = LogisticRegression(class_weight='balanced', max_iter=1000)
model.fit(X_train_tfidf, y_train)

# Evaluation (Optional)
print("Accuracy:", accuracy_score(y_test, model.predict(X_test_tfidf)))
pickle.dump(model, open("model.pkl", "wb"))
pickle.dump(tfidf, open("tfidf.pkl", "wb"))
```

**app.py**

```
import streamlit as st
import pickle
import string
import nltk
from nltk.corpus import stopwords

# Download NLTK stopwords (run only once)
nltk.download('stopwords')

# Load model and vectorizer
model = pickle.load(open("model.pkl", "rb"))
tfidf = pickle.load(open("tfidf.pkl", "rb"))

# Define stopwords
stop_words = set(stopwords.words('english'))

# Preprocessing function (must match training script!)
def preprocess(text):
    text = text.lower()
    text = "".join([ch for ch in text if ch not in string.punctuation])
    words = text.split()
    words = [word for word in words if word not in stop_words]
    return " ".join(words)
```

```
# Streamlit UI
st.title("📰 Fake News Detection")
st.subheader("Enter a news article or headline:")

input_text = st.text_area("News content")

if not input_text:
    input_text = "NASA confirms water on the moon."  # default test input

if st.button("Predict"):
    cleaned = preprocess(input_text)
    vector = tfidf.transform([cleaned])
    prediction = model.predict(vector)[0]
    probabilities = model.predict_proba(vector)[0]
    confidence = probabilities[prediction]

    # Optional debug
    st.write(f"Raw prediction: {prediction} (0 = FAKE, 1 = REAL)")
        st.write(f"Prediction probabilities: FAKE = {probabilities[0]:.2f}, REAL =
{probabilities[1]:.2f}")

    if prediction == 1:
        st.success(f"✅ This news is REAL with {confidence*100:.2f}% confidence.")
    else:
        st.error(f"⚠️ This news is FAKE with {confidence*100:.2f}% confidence.")
```

**OUTPUT:**

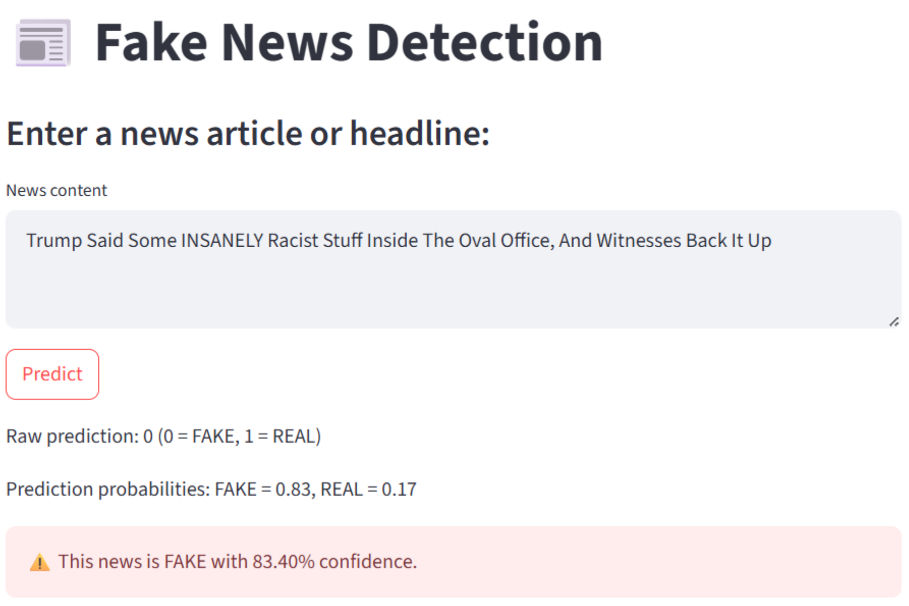


Fake News Detection

Enter a news article or headline:

News content

Predict



Fake News Detection

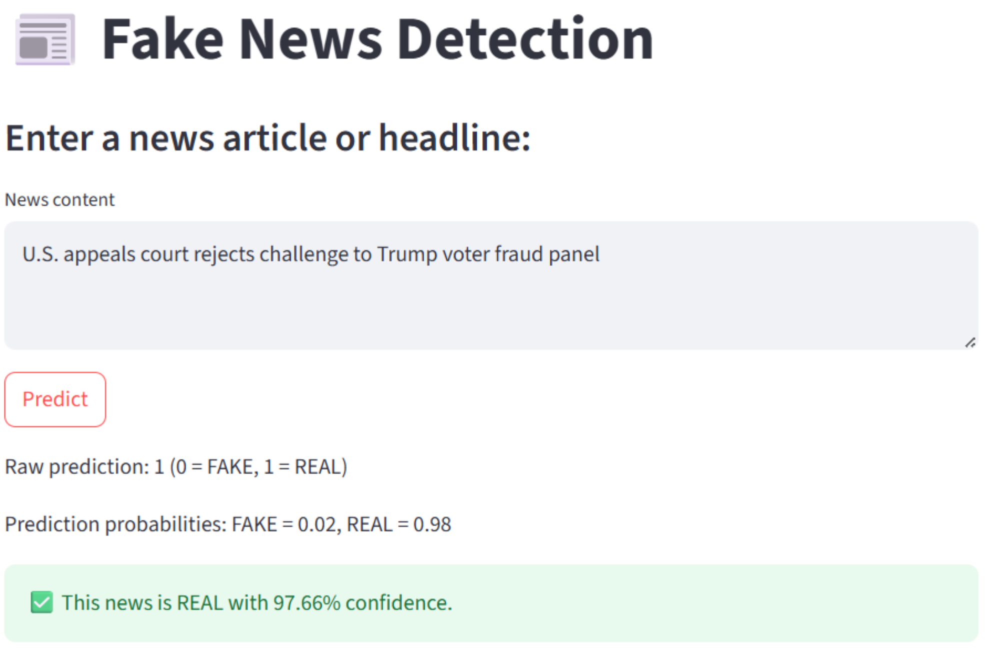
Enter a news article or headline:

News content

```
Trump Said Some INSANELY Racist Stuff Inside The Oval Office, And Witnesses Back It Up
```

Predict

```
Raw prediction: 0 (0 = FAKE, 1 = REAL)

Prediction probabilities: FAKE = 0.83, REAL = 0.17

⚠ This news is FAKE with 83.40% confidence.
```



Fake News Detection

Enter a news article or headline:

U.S. appeals court rejects challenge to Trump voter fraud panel

```
Raw prediction: 1 (0 = FAKE, 1 = REAL)

Prediction probabilities: FAKE = 0.02, REAL = 0.98

✓ This news is REAL with 97.66% confidence.
```

**RESULT:**

The Random Forest model was successfully trained to classify news articles as *Fake* or *Real*. The model achieved high accuracy on the test dataset, effectively distinguishing between misleading and authentic news. The predicted results closely matched the true labels, demonstrating that the model accurately captured linguistic and contextual patterns within the news data.