AI PHASE_3 SUBMISSION

FAKE NEWS DETECTION USING NLP

Loading and Preprocessing the Dataset

```
from google.colab import files

# Use the files.upload() method to upload your dataset
uploaded = files.upload()
# @title Default title text
# Install required libraries
!pip install scikit-learn nltk

# Import necessary libraries
import pandas as pd
import numpy as np
import re
import nltk
nltk.download('stopwords')
from nltk.corpus import stopwords
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix

# Load your dataset (assuming you have a CSV file with 'text' and 'label' columns)
# You can upload your dataset to Google Colab or provide a link to your data file.
data = pd.read_csv('/content/drive/MyDrive/Fake[1].csv')
```

```
# Text Preprocessing
stop_words = set(stopwords.words('english'))

def preprocess_text(text):
    text = text.lower()  # Convert to lowercase
    text = re.sub(r'https?://\S+|www\.\S+', '', text)  # Remove URLs
    text = re.sub(r'<.*?>', '', text)  # Remove HTML tags
    text = re.sub(r'[^\w\s]', '', text)  # Remove punctuation
    text = ' '.join(word for word in text.split() if word not in stop_words)  # Remove stopwords
    return text

data['text'] = data['text'].apply(preprocess_text)

# Feature Extraction (TF-IDF)
tfidf_vectorizer = TfidfVectorizer(max_features=5000)
X = tfidf_vectorizer.fit_transform(data['text']).toarray()
y = data['label']

# Split the 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)

# Model Training (Random Forest Classifier)
classifier = RandomForestClassifier(n_estimators=100, random_state=42)
classifier.fit(X_train, y_train)
```

AI PHASE_3 SUBMISSION

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Model Training (Random Forest Classifier)
classifier = RandomForestClassifier(n_estimators=100, random_state=42)
classifier.fit(X_train, y_train)

# Model Evaluation
y_pred = classifier.predict(X_test)

accuracy = accuracy_score(y_test, y_pred)
confusion = confusion_matrix(y_test, y_pred)
report = classification_report(y_test, y_pred)

print("Accuracy: {:.2f}%".format(accuracy * 100))
print("Confusion Matrix:\n", confusion)
print("Classification Report:\n", report)
```

OUTPUT

```
Accuracy: 92.30%
Confusion Matrix:
 [[902  61]
 [ 94 950]]
Classification Report:
              precision    recall  f1-score   support

           0       0.91      0.94      0.93       963
           1       0.94      0.91      0.93      1044

    accuracy                           0.93      2007
   macro avg       0.93      0.93      0.93      2007
weighted avg       0.93      0.93      0.93      2007
```