FAKE NEWS DETECTION

PROGRAM/SOURCE OF CODE

import numpy as np

import pandas as pd

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, confusion\_matrix

# Step 1: Load the data

df = pd.read\_csv('news.csv') # Make sure your CSV contains columns: 'title', 'text', 'label'

labels = df.label

# Step 2: Split the data

x\_train, x\_test, y\_train, y\_test = train\_test\_split(df['text'], labels, test\_size=0.2, random\_state=7)

# Step 3: Feature extraction

vectorizer = TfidfVectorizer(stop\_words='english', max\_df=0.7)

tfidf\_train = vectorizer.fit\_transform(x\_train)

tfidf\_test = vectorizer.transform(x\_test)

# Step 4: Train the model

model = LogisticRegression()

model.fit(tfidf\_train, y\_train)

# Step 5: Predictions and accuracy

y\_pred = model.predict(tfidf\_test)

score = accuracy\_score(y\_test, y\_pred)

print(f'Accuracy: {score\*100:.2f}%')

# Step 6: Confusion Matrix

print(confusion\_matrix(y\_test, y\_pred))

# Optional: Test with your own news text

def predict\_news(news\_text):

vect = vectorizer.transform([news\_text])

prediction = model.predict(vect)

return prediction[0]

# Example usage

example\_text = "Sample news article text here."

print("Prediction:", predict\_news(example\_text))

EXPLANATION OF THE CODE

1. **Import Libraries**:
   * numpy and pandas: Handle numerical computations and data manipulation (loading CSV).
   * sklearn.model\_selection.train\_test\_split: Splits data into training and testing sets.
   * sklearn.feature\_extraction.text.TfidfVectorizer: Converts text to TF-IDF features.
   * sklearn.linear\_model.LogisticRegression: Performs binary classification.
   * sklearn.metrics.accuracy\_score, confusion\_matrix: Evaluate model performance.
2. **Load the Data**:
   * Loads news.csv into a pandas DataFrame, expecting columns: title, text, label (real or fake).
   * Extracts label column as the target variable.
3. **Split the Data**:
   * Uses train\_test\_split to divide text (features) and labels (target) into 80% training and 20% testing sets.
   * random\_state=7 ensures reproducibility.
4. **Feature Extraction**:
   * TfidfVectorizer transforms text into TF-IDF features:
     + stop\_words='english': Removes common words (e.g., "the").
     + max\_df=0.7: Ignores terms in >70% of documents.
   * Fits and transforms training text (fit\_transform) and transforms test text (transform) into sparse matrices.
5. **Train the Model**:
   * Initializes a LogisticRegression model.
   * Trains it on TF-IDF features (tfidf\_train) and training labels (y\_train).
6. **Predictions and Accuracy**:
   * Predicts labels for test data (tfidf\_test) using the trained model.
   * Computes accuracy by comparing predictions (y\_pred) to true labels (y\_test).
   * Prints accuracy as a percentage.
7. **Confusion Matrix**:
   * Generates a 2x2 confusion matrix showing true negatives, false positives, false negatives, and true positives.
   * Helps analyze misclassifications.
8. **Custom Prediction Function**:
   * Defines predict\_news(news\_text) to classify new text.
   * Transforms input text to TF-IDF features and predicts the label (real or fake).
9. **Example Usage**:
   * Tests predict\_news with sample text, printing the predicted label.