Explanation for Fake News Detection Using NLP

Introduction:

This document explains the code provided for Phase 1 of the Fake News Detection project. In this phase, we perform data preprocessing, feature extraction using TF-IDF, build a Logistic Regression model, and evaluate its performance. Below is a step-by-step breakdown of the code.

Steps :

Import Necessary Libraries:

The code begins by importing essential Python libraries for data manipulation, natural language processing, and machine learning. These include Pandas, scikit-learn (for machine learning), and NLTK (Natural Language Toolkit) for text preprocessing.

Download NLTK Data:

NLTK requires additional data resources, such as stopwords and punkt, for text preprocessing. The code downloads these resources to facilitate text cleaning and tokenization.

Load Datasets:

Two datasets, 'fake.csv' and 'true.csv', are loaded into Pandas DataFrames. These datasets should contain fake and true news articles, respectively. The code assigns labels '0' for fake news and '1' for true news.

Concatenate Datasets:

The code concatenates the two DataFrames into a single 'data' DataFrame, ensuring that the labels are appropriately assigned.

Data Preprocessing:

Combine Title and Text: The title and text of each news article are concatenated into a single 'content' column, simplifying text processing.

Clean and Preprocess Text: The 'preprocess_text' function is defined to clean and preprocess the text data. It converts text to lowercase, tokenizes it, removes stopwords, and retains only alphabetic words. The cleaned text is stored in the 'cleaned_content' column.

Split Data:

The 'data' DataFrame is split into training and testing sets using scikit-learn's 'train_test_split' function. 80% of the data is used for training, and 20% for testing, with a fixed random seed for reproducibility.

TF-IDF Vectorization:

TF-IDF Vectorizer: A TF-IDF vectorizer is initialized with a maximum of 5000 features. This step converts the cleaned text data into numerical vectors, where each feature represents a unique word or phrase.

Transform Data:

The training data ('X_train') is transformed into TF-IDF vectors using the 'fit_transform' method, while the testing data ('X_test') is transformed using 'transform' method. This ensures that the vectorizer uses the same vocabulary for both sets.

Build and Train a Classification Model:

A Logistic Regression classifier is chosen for this project. The classifier is initialized, and then it's trained on the TF-IDF transformed training data ('X_train_tfidf') with corresponding labels ('y_train').

Make Predictions:

The trained model is used to make predictions on the TF-IDF transformed testing data ('X_test_tfidf'), resulting in predicted labels ('y_pred').

Evaluate the Model:

Accuracy: The code calculates the accuracy score of the model by comparing the predicted labels ('y_pred') with the actual labels from the testing set ('y_test'). The accuracy score measures the proportion of correctly predicted instances.

Classification Report:

The code generates a classification report that includes precision, recall, and F1-score for both the 'fake' and 'true' classes. The classification report provides a more detailed performance evaluation.

Conclusion:

This code represents Phase 1 of the Fake News Detection project, where data is preprocessed, features are extracted using TF-IDF, a Logistic Regression model is built, and its performance is evaluated. The accuracy and classification report are key indicators of how well the model distinguishes between fake and true news articles.