The code provided is divided into two sections: data preprocessing and text categorization using the Naive Bayes technique. Here is a code breakdown with related documentation:

1. Data Preparation:

a. Cleaning Text: The function 'clean\_text' is defined to preprocess text data.

- It uses the NLTK library to transform the text to lowercase, remove special letters and symbols, tokenize the text into individual words, and remove stopwords.

- The cleaned text is returned in the form of a single string.

b. Dataset Loading and Cleaning: - The pandas library is imported as 'pd'.

- The NLTK module'stopwords' is downloaded using 'nltk.download('stopwords')'.

- The True dataset is loaded into a DataFrame named 'df' from the file "True.csv".

- The `clean\_text` function is applied to the 'text' column of the DataFrame, and the cleaned text is stored in a new column named 'cleaned\_text'.

- The 'clean\_text' function is applied to the DataFrame's 'text' column, and the cleaned text is saved in a new column called 'cleaned\_text'.

- The cleaned text has been printed.

2. Text categorization:

a. Dataset Loading and Labelling: - The following libraries are imported: numpy as 'np,' pandas as 'pd,' GridSearchCV and train\_test\_split from'sklearn.model\_selection,' MultinomialNB from'sklearn.naive\_bayes,' CountVectorizer from'sklearn.svm,' TfidfV

- The Fake and True datasets are loaded into separate DataFrames from the files "Fake.csv" and "True.csv", respectively.

- A 'label' column is added to both DataFrames, with 0 representing Fake and 1 representing True.

- The two DataFrames are combined into a single DataFrame called 'data'.

b. Splitting the Dataset: 'X' is assigned to the 'text' column, while 'y' is assigned to the 'label' column.

- Using the 'train\_test\_split' function, the dataset is divided into training and testing sets, with 80% for training and 20% for testing.

- The features of the training set ('X\_train') and the testing set ('X\_test') are produced.

- The labels for the training and testing sets ('y\_train' and 'y\_test') are generated.

c. Feature Extraction: A 'CountVectorizer' object called'vectorizer' is created.

- The 'fit\_transform' method on 'X\_train' is invoked to learn the vocabulary and transform the training data into feature vectors.

- The 'transform' method on 'X\_test' is used to convert the testing set into feature vectors.

d. Evaluation and Training: - The Multinomial Naive Bayes classifier ('MultinomialNB') is instantiated as 'classifier'.

- Using the training data and labels, the classifier is subjected to the 'fit' procedure.

- The classifier's 'predict' method is invoked to produce predictions based on the testing data.

- The accuracy of the predictions is determined and stored in 'accuracy' using the 'accuracy\_score' function.

e. Making New Predictions: 'new\_texts' contains a list of new texts.

- Using the same'vectorizer' object, the 'transform' method is performed on 'new\_texts' to convert them into feature vectors.

- The classifier's 'predict' method is invoked, and predictions are made using the feature vectors of the new texts.

- The forecasts are printed beside the relevant text.

Documentation References:

- NLTK: Natural Language Toolkit - https://www.nltk.org/

- Pandas: Python Data Analysis Library - https://pandas.pydata.org/

- scikit-learn: Machine Learning in Python - https://scikit-learn.org/stable/