The provided code consists of two main parts: data preprocessing and text classification using the Naive Bayes algorithm. Here's a breakdown of the code with relevant documentation:

1. Data Preprocessing:

a. Cleaning Text:

- The `clean\_text` function is defined to preprocess the text data.

- It converts the text to lowercase, removes special characters and symbols, tokenizes the text into individual words, and removes stopwords using the NLTK library.

- The cleaned text is returned as a single string.

b. Loading and Cleaning Dataset:

- The pandas library is imported as `pd`.

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

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

- 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 cleaned text is printed.

2. Text Classification:

a. Dataset Loading and Labeling:

- The required 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.feature\_extraction.text`, LinearSVC from `sklearn.svm`, TfidfVectorizer from `sklearn.feature\_extraction.text`, accuracy\_score from `sklearn.metrics`, DecisionTreeClassifier from `sklearn.tree`, shuffle from `sklearn.utils`, and hstack from `scipy.sparse`.

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

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

- The two DataFrames are concatenated into a single DataFrame named `data`.

b. Splitting the Dataset:

- The 'text' column is assigned to `X`, and the 'label' column is assigned to `y`.

- The dataset is split into training and testing sets using the `train\_test\_split` function, with 80% for training and 20% for testing.

- The training set features (`X\_train`) and testing set features (`X\_test`) are created.

- The training set labels (`y\_train`) and testing set labels (`y\_test`) are created.

c. Feature Extraction:

- A `CountVectorizer` object named `vectorizer` is instantiated.

- The `fit\_transform` method is called on `X\_train` to learn the vocabulary and transform the training set into feature vectors.

- The `transform` method is called on `X\_test` to transform the testing set into feature vectors.

d. Training and Evaluation:

- The Multinomial Naive Bayes classifier (`MultinomialNB`) is instantiated as `classifier`.

- The `fit` method is called on the classifier using the training data and labels.

- The `predict` method is called on the classifier using the testing data to make predictions.

- The accuracy of the predictions is calculated using the `accuracy\_score` function and stored in `accuracy`.

e. Making New Predictions:

- A list of new texts is provided as `new\_texts`.

- The `transform` method is called on `new\_texts` to convert them into feature vectors using the same `vectorizer` object.

- The `predict` method is called on the classifier using the feature vectors of the new texts to make predictions.

- The predictions are printed along with the corresponding 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/