========================================================================

Email Spam Classification System

========================================================================

This Python script implements a simple email spam classification system using the XGBoost classifier. It consists of several functions for loading, preprocessing, training, and predicting email data.

Data Sources:

For the execution purpose of the assignment, I have considered the input as follows:

* The training data is sourced from two folders: 'spam' and 'ham', which are renamed versions of the '20050311\_spam\_2.tar' and '20030228\_easy\_ham\_2.tar' datasets respectively. The 'spam' folder contains spam emails, while the 'ham' folder contains non-spam (ham) emails.
* The 'Prediction\_data' folder contains emails from '20021010\_easy\_ham.tar' and '20021010\_spam.tar' datasets, but in a jumbled order.

Functions:

1. load\_training\_email\_data(folder):

- Loads email data from the specified folder.

- Assigns labels: 1 for spam, 0 for ham.

- Returns a list of emails and their corresponding labels.

2. preprocess\_text(text):

- Preprocesses text data by removing non-alphanumeric characters, HTML tags, and converting to lowercase.

- Tokenizes the text, lemmatizes tokens, and joins them back into text.

- Returns preprocessed text.

3. build\_XGBClassifier(X, y):

- Builds an XGBoost classifier.

- Fits the classifier to the training data.

- Returns the trained classifier.

4. train\_model(emails, labels):

- Vectorizes the text data using CountVectorizer.

- Splits the data into training and testing sets.

- Trains an XGBoost classifier on the training data.

- Evaluates the classifier's accuracy on the testing data.

- Returns the trained vectorizer and classifier.

5. load\_prediction\_email\_data(folder):

- Loads email data from the specified prediction folder.

- Returns a list of emails.

Main Execution:

---------------

- Loads training email data from 'spam' and 'ham' folders.

- Preprocesses the email data.

- Trains the XGBoost classifier using the preprocessed data.

- Loads prediction email data from the 'Prediction\_data' folder.

- Predicts labels for the new emails and prints the results.

- Prints evaluation metrics such as precision, recall, confusion matrix, accuracy, and F1 score.

Data and Resource Dependencies:

-------------------------------

- NLTK resources: punkt, wordnet, omw, omw-1.4

- XGBoost library

- NumPy

- Regular Expressions (re)

- OS module

Note: Ensure NLTK resources are downloaded before running the script.

Pipeline Output:

I have parsed each email from Prediction\_data folder which is a mixture of spam and not spam emails. The cleaning and preprocessing is done and the prediction is done on each email. The prediction\_data folder can be loaded with any number off emails and the prediction is made and reported based on the file position in the folder. Example:

The email at position 2254 is Not Spam

The email at position 2255 is a Spam

The email at position 2256 is a Spam

The email at position 2257 is Not Spam

The email at position 2258 is Not Spam

The email at position 2259 is Not Spam

The email at position 2260 is Not Spam

The email at position 2261 is Not Spam

The email at position 2262 is Not Spam

The email at position 2263 is a Spam

The email at position 2264 is a Spam

The email at position 2265 is Not Spam

The email at position 2266 is Not Spam

The email at position 2267 is Not Spam

The email at position 2268 is Not Spam

The email at position 2269 is Not Spam

The email at position 2270 is a Spam

The email at position 2271 is Not Spam

The email at position 2272 is Not Spam

The email at position 2273 is Not Spam

The email at position 2274 is Not Spam

The email at position 2275 is Not Spam

The email at position 2276 is a Spam

The email at position 2277 is Not Spam

The email at position 2278 is Not Spam

The email at position 2279 is Not Spam