### **Step 4: Text Preprocessing**

The goal of text preprocessing is to transform the raw email text into a format that can be effectively used by our machine learning model. This involves several sub-steps:

1. **Removing punctuation**.
2. **Converting text to lowercase**.
3. **Removing stopwords**.

Here is the breakdown of each step in the preprocess\_text function:

#### **Step 4.1: Downloading Stopwords**

Stopwords are common words that don't carry much meaning and can be removed. NLTK provides a list of stopwords.

Run this code once at the beginning to download the stopwords:

python

Copy code

import nltk

nltk.download('stopwords')

You can run this code in any Python environment, such as:

* Jupyter Notebook
* VS Code with a Python environment set up
* Python script executed from the command line or an IDE

#### **Step 4.2: Define the Preprocessing Function**

Now, let's define the preprocess\_text function which will be applied to each email message.

python

Copy code

import string

from nltk.corpus import stopwords

# Load stopwords from NLTK

stop\_words = set(stopwords.words('english'))

def preprocess\_text(text):

# Remove punctuation

text = ''.join([char for char in text if char not in string.punctuation])

# Convert to lowercase

text = text.lower()

# Remove stopwords

text = ' '.join([word for word in text.split() if word not in stop\_words])

return text

You can define this function in a Python script, Jupyter Notebook cell, or any Python environment where you are writing and executing your code.

#### **Step 4.3: Applying the Preprocessing Function to the Dataset**

Next, we apply this preprocessing function to each email message in our dataset. Assuming data is a DataFrame with a column message containing the email texts, we use the apply method to process each message.

python

Copy code

data['message'] = data['message'].apply(preprocess\_text)

This should be done after you have loaded your dataset into the data DataFrame.

### **Where to Run the Code**

1. **Jupyter Notebook**:
   * Run each of these code blocks in separate cells in a Jupyter Notebook.
   * Ensure you have the dataset loaded in a previous cell.
2. **VS Code**:
   * Open a new Python file (e.g., spam\_preprocessing.py).
   * Write all the code snippets in the file sequentially.
   * Ensure you have a Python interpreter set up in VS Code.
   * Run the file by pressing F5 or using the terminal: python spam\_preprocessing.py.
3. **Command Line/Terminal**:
   * Create a Python script (e.g., spam\_preprocessing.py) and include all the code snippets.
   * Execute the script using the command: python spam\_preprocessing.py.
4. **Any Python IDE**:
   * Similar to VS Code, you can use any Python IDE (e.g., PyCharm, Anaconda) to write and run the code.

### **Complete Example Code**

Here is a complete example of how you might structure your code in a script:

python

Copy code

import numpy as np

import pandas as pd

import string

import nltk

from nltk.corpus import stopwords

from sklearn.model\_selection import train\_test\_split

# Download stopwords

nltk.download('stopwords')

# Load stopwords from NLTK

stop\_words = set(stopwords.words('english'))

# Define the preprocessing function

def preprocess\_text(text):

# Remove punctuation

text = ''.join([char for char in text if char not in string.punctuation])

# Convert to lowercase

text = text.lower()

# Remove stopwords

text = ' '.join([word for word in text.split() if word not in stop\_words])

return text

# Load the dataset

data = pd.read\_csv('spam.csv', encoding='latin-1')

data = data[['v1', 'v2']]

data.columns = ['label', 'message']

# Encode labels (spam=1, ham=0)

data['label'] = data['label'].map({'ham': 0, 'spam': 1})

# Preprocess messages

data['message'] = data['message'].apply(preprocess\_text)

# Train-test split

X\_train, X\_test, y\_train, y\_test = train\_test\_split(data['message'], data['label'], test\_size=0.2, random\_state=42)

This script covers the downloading of stopwords, defining the preprocessing function, loading the dataset, and applying the preprocessing function to the dataset. You can then proceed with the remaining steps in your script.