SPAM or HAM?

Description of data:

In this assignement I am implementing **Naïve Bayes** algorithm for classification with bernoulli distribution. The dataset I have used for this purpose is found from the follwing link:

https://www.kaggle.com/datasets/wanderfj/enron-spam

The description about the above mentioned data can be found here:

http://nlp.cs.aueb.gr/software_and_datasets/Enron-Spam/readme.txt

This data constitutes of two separate folders out of which one is for spam emails and the other is for non-spam/ham emails. It comprises 5975 emails in total out of which 3672 are non spam and 1500 are spam emails.

Steps followed in the code:

a. In the first step we **read** the spam data and ham data in two separate lists as it is present in the text files.
b. In the second step we have **cleaned** the data as part of our data-preprocessing. In this step we do three major operations
   i. We remove the punctuations.
   ii. We romove the stopwords.
   iii. We remove the non alpha-numeric words.
c. Now we proceed to create the **vectorized representation** of the processed data. To begin with, we first create a collection of all spam and non spam emails to pass it to count-vectorizer. Therefore from count vectorizer we get the frequencies of each word of each mail from our corpus.
d. Next, I find the **probabilities** of each words in either spam or non-spam, using the formula:

Probability of word (i) in a category:

(number of mails containing the word i/total number of mails of the category)

We calculate it as follows:

e. Seperating out the vectors into spam and non-spam categories and summing them up. Then dividing by the corresponding total number of mails of that category. Now the result of this does not give us a value between 0 and 1 since I added the number of occurences instead of the presence or not indicator, therefore I normalise the obtained result and that preserves the weightage of each word. The more number of times it is present in a mail adds more contribution.
f. After this I prepare the linear seperator and bias as we find in the logarithmic formula of the decision function.

Therefore, I obtain the predicted labels of train data by performing transpose(W) * the normalised frequency vector. And comparing it with the actual labels, I can find the **train accuracy** as

92.11 %

At the last step I create a function called **classifier()** that takes input from a folder called **test** in current directory and gives the output as:
“email#.txt is predicted as spam” or “email#.txt is predicted as non-spam”

Note: To read the input for training and testing, the console working directory as the current directory so that the relative indirection works properly for tarin and test data.