PRML Assignment 3 Report

Roll Number : CS21M022
Name : Hanumantappa Budihal

Email Spam Classifier

Spam filtering is a classification task which involves classifying an email as spam or ham. Spam box in our gmail account is the best example of this. Spam messages are messages sent to a large group of recipients without their knowledge, typically advertising for goods or services and often business opportunities.

In the recent period , the percentage of scam messages has increased sharply. Scam messages typically trick people into giving away money or personal details by offering an attractive or false deal. There are various traditional methods to prevent spam messages such as blacklisting certain senders or IP addresses but none of them are as impactful as recent techniques which use aspects of machine learning and artificial intelligence.

In this assignment I have created a spam classifier that reads the mails and classify them as Ham (Label 0) or Spam (Label 1). This classifier is an example of a binary classifier and is implemented using Naive Bayes Algorithm.

Dataset :

The dataset has a collection of 11425 mails labeled as either 0 (ham) or 1 (spam). The dataset contains around 81.82 % ham and around 19.18 % spam mails . The dataset is present in the assignment folder by the name ”training_dataset.csv”. The dataset looks like as,

-

Email	label
came to look at the flat, seems ok, in his 50s? * Is away alot wiv work.	0
said kiss, kiss, i can't do the sound effects! He is a gorgeous man isn't	0
says that he's quitting at least5times a day so i wudn't take much notic	0
-PLS STOP bootydelious (32/F) is inviting you to be her friend. Reply Y	1
-PLS STOP bootydelious (32/F) is inviting you to be her friend. Reply Y	1
, , and picking them up from various points | going 2 yeovil | and they	0
, how's things? Just a quick question.	0
, im .. On the snowboarding trip. I was wondering if your planning to ge	0
, ow u dey.i paid 60,400thousad.i told u would call .	0
;-(oh well, c u later	0
;-) ok. I feel like john lennon.	0

Approach

1. Data Pre-Processing
2. Model Training
3. Model Evaluation
4. Testing
5. Conclusion

1.Data Pre-Processing :

To prepare dataset for training we need to preprocess the raw dataset and convert it to a format that is suitable for training model.

Below steps involved

I. Tokenization
II. Removing Mail ID's
III. Removing Format Words
IV. Removing Numbers and Punctuations
V. Remove Stop Words
VI. Lemmatizing
VII. Remove the Duplicate Words

1. **Tokenization :** In this step every mail dataset will convert the whole paragraphic content of that main into a list of the words(token). It is done because these tokens are the features on which we will train our model.

   I have used the NLTK library to do the token generation.

2. **Removing mail ID's :** In this step for every token in the list of tokens generated in the above step we will remove that token if it is an email ID. It is done because email Id' does not give any indication whether a mail is spam or not, therefore email ID's are redundant and only add increases the size of the token list.

   This can be achieved by removing the token that has both @ and .character present.

3. **Remove Format Words :** In this step for every token in the list of tokens generated in the tokenization step we will remove that token if it is some sort of a format word such as \pard, \red255 etc. It is done because these format words do not add any meaning to the content and are just for beautification purposes.

4. **Removing Numbers and Punctutations:** In this step for every token in list of tokens generated in tokenization step we will remove that token if it is some sort of a number or a punctuation word such as #, +, $ etc.

   It is done because numbers and punctuations alone do not give any hint about the mail being spam or not. Numbers along with some text may give hint e.g. Lucky draw of 5 million dollars might suggest email is spam but since we tokenized the content and Naive Bayes works on word by word basis, the single token 5 is redundant and does not help us in classification, therefore we removed numbers and punctuations. For this purpose we simply checked each word to see if it is either a number or punctuation or not and removed those that fell into any one of these two categories.

5. **Removing Stop Words:** Stop words refers to the most common words in a language. For e.g. In English language stop words are a, an, the, them etc. In this step for every token in list of tokens generated in the tokenization step we will remove that token if it is a stop word. It is done because stop words are common and present in both spam as well as non spam messages and thus does not help in distinguishing them, therefore we removed such words.

   For this purpose we have used the list of stop words in english language provided by NLTK as well as added some by ourselves such as could, might, would, may etc and for each word (token) in the list of tokens we removed those words that fell in the list of stop words.

6. **Lemmatizing:** Lemmatizing is the process of grouping together the inflected forms of a word so they can be analysed as a single item, identified by the word's lemma, or dictionary form.

   For e.g. reader becomes read. In this step for every token in the list of tokens generated in the tokenization step we will lemmatize that word. It is done because it might be the case that our training spam mail has word jackpots and test mail might have word jackpot. Now despite same convenying same meaning our model will say that it has no idea about the test mail as our traning data does not contain the exact word jackpots therefore it will reduce the accuracy, therefore we have used lemmatization. However lemmatization is harder to implement and slower compared to stemming but since our training dataset is not so large we implemented it.

   For lemmatizing I have used **WordNetLemmatizer** from NLTK.

7. **Removing Duplicate Words:** In this step for every token in list of tokens generated in tokenization step if that token appears more than once then only one instance will be kept and remaining instances will be removed.

Removing duplicates is done because multiplying the probability of duplicate word **P(DuplicateWord/Spam)** and **P(DuplicateWord/Ham))** with **P(Words/Spam)** and **P(Words/Ham)** respectively will only make the latter smaller which creates problems due to truncation. Removing duplicate words will also make the list of tokens smaller and thus faster to compute.

For this purpose we convert the list of tokens into a set (so all duplicates are removed) and then reconvert the set into the list. After removing duplicate words.

After pre-processing the tokens in the list are the pre-processed features on which our model will be trained. After pre-processing the above raw mail is converted into data points with features.

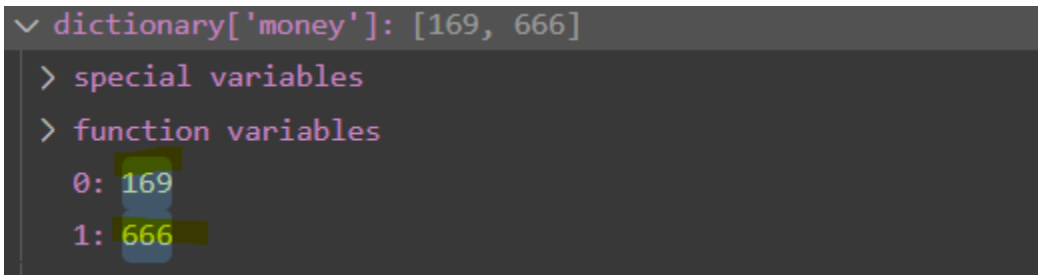
2.Model Training:

We will use Naive Bayes algorithm to train our model. After pre-processing the dataset we now have a list of datapoints (mails) and each datapoint itself is a list of features (words). One thing to note is that each datapoint has a different number of words therefore the number of features/dimensions for each datapoint is different.

Now we construct a dictionary data structure, named dictionary whose key is a word and value is a list containing two elements. The first element (index 0 element) indicates the frequency or the number of ham mails which contains the given word and the second element (index 1 element) indicates the frequency or the number of spam mails which contains the given word.

dictionary[‘money’][0]=169 dictionary[‘money’][1]=666

There are 169 ham mails and 666 spam mails respectively which contain the word ‘money’. The dictionary generated is as



After the frequency dictionary is generated we perform Laplacian Smoothing by incrementing the count of every word in the dictionary for both ham and spam mails.

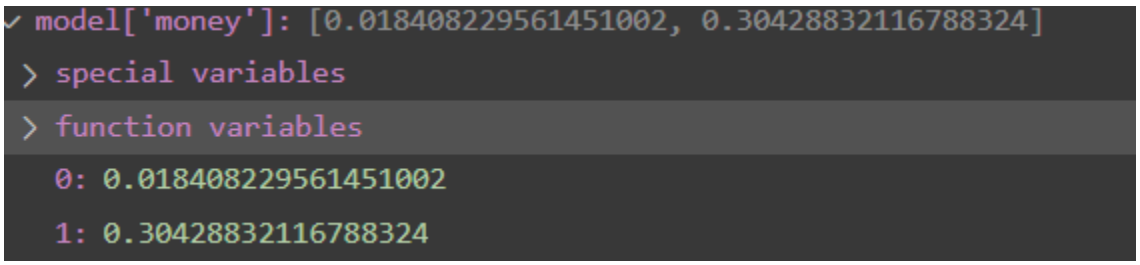
This is analogous to adding one dummy ham and one dummy spam mail to the dataset. We aslo increase the count of total ham and spam mails by one. Smoothing guarantees that there is at least one mail in ham and spam mails which contain the given in turn guarantees probability of mail being spam or ham is non zero and thus decidable.

Now at the last stage of training we generate a probability table named probability table which is similar to a dictionary but unlike a dictionary it tells the probability of a word given ham or spam. For e.g.

$$Probability\_table['money'][0] = dictionary[word][0]/number\ of\ Ham\ mails$$

$$Probability\_table['money'][1] = dictionary[word][1]/number\ of\ Spam\ mails$$

The probability table generated is as ,



3.Model Evaluation :

It uses Naive Bayes algorithm to calculate probability of test mail being spam and ham and assign the label that has higher probability. Here the testing mails are pre-processed in the same way as training mails.

The formula to calculate probability of spam and ham given mail (bunch of words) are as,

$$P(Spam/Words) = \frac{P(Spam)*P(Words/Spam)}{P(Words)}$$

$$P(Ham/Words) = \frac{P(Ham)*P(Words/Ham)}{P(Words)}$$

Since any mail is equally likely to be spam or ham therefore we took

$$P(Ham) = P(Spam) = 0.5$$

In Naive Bayes algorithm words occurring in mails are assumed to be independent therefore $P(Words/Spam)$ can be written as ,

$$P(Words/Spam) = P(Word_1/Spam) * P(Word_2/Spam). . . . . P(Word_k/Spam)$$

and as we saw earlier that,

$$P(Words/Spam) = Probability\_table[word_k][1]$$

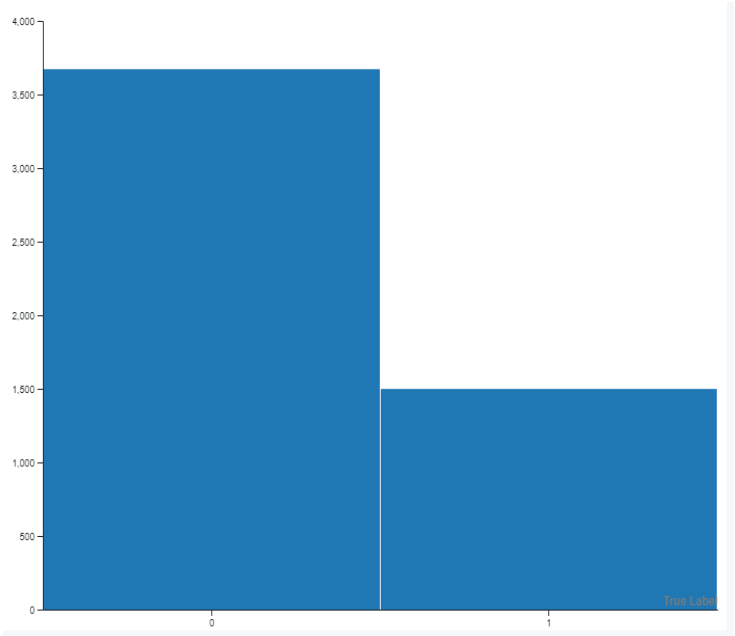
4.Testing :

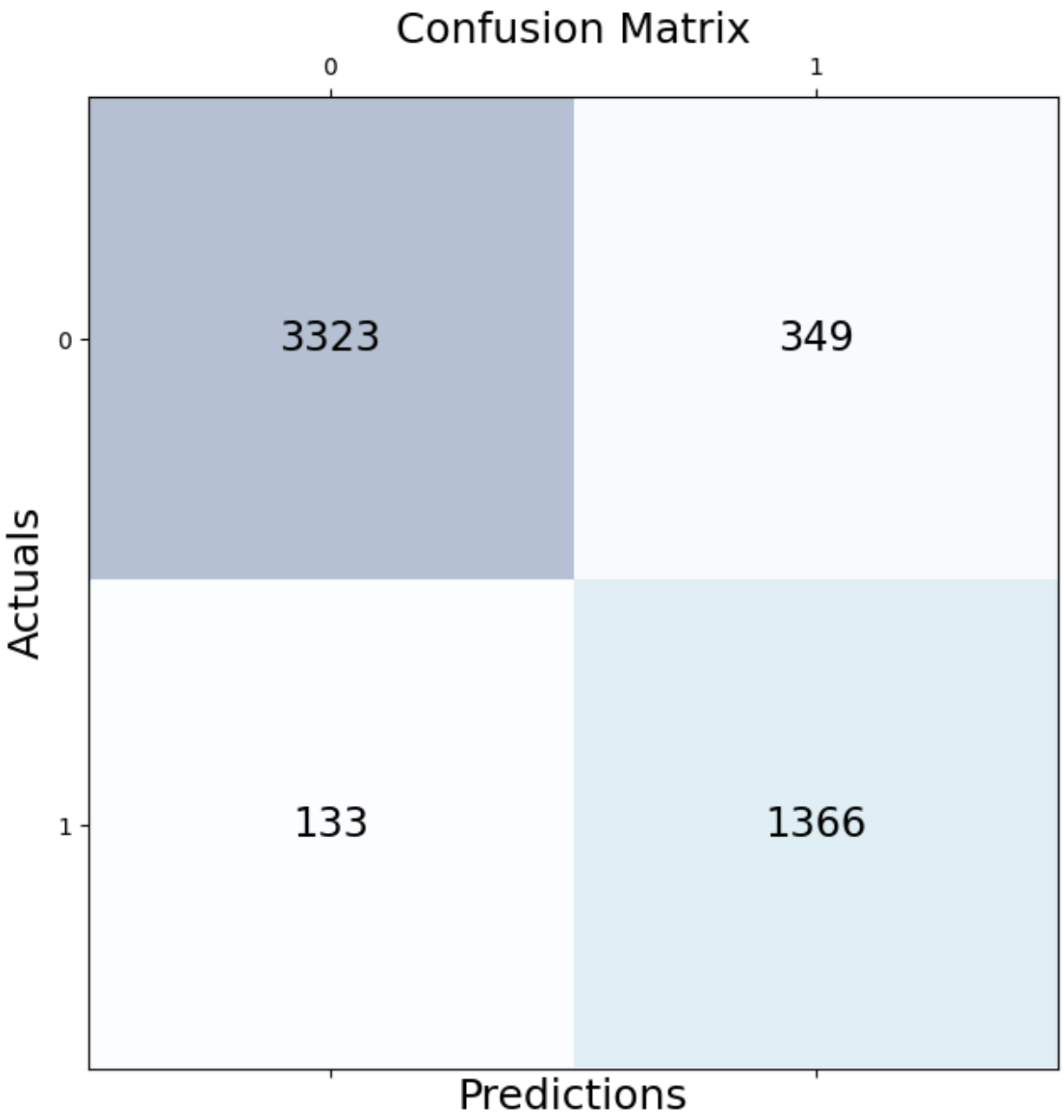
1.Dataset1 : (..\dataset\testdata1.csv)

Test dataset : ../dataset/testdata1.csv
Output : ../ouput/testdata1_output.csv
Confusion matrix : ../plots/Testdata1_confusion_matrix.png

This dataset has 5171 email samples which contains 3672 (71%) Ham emails and 1499 (29%) Spam emails.

Email classifiers are able to achieve the 90.67% of accuracy on this dataset.





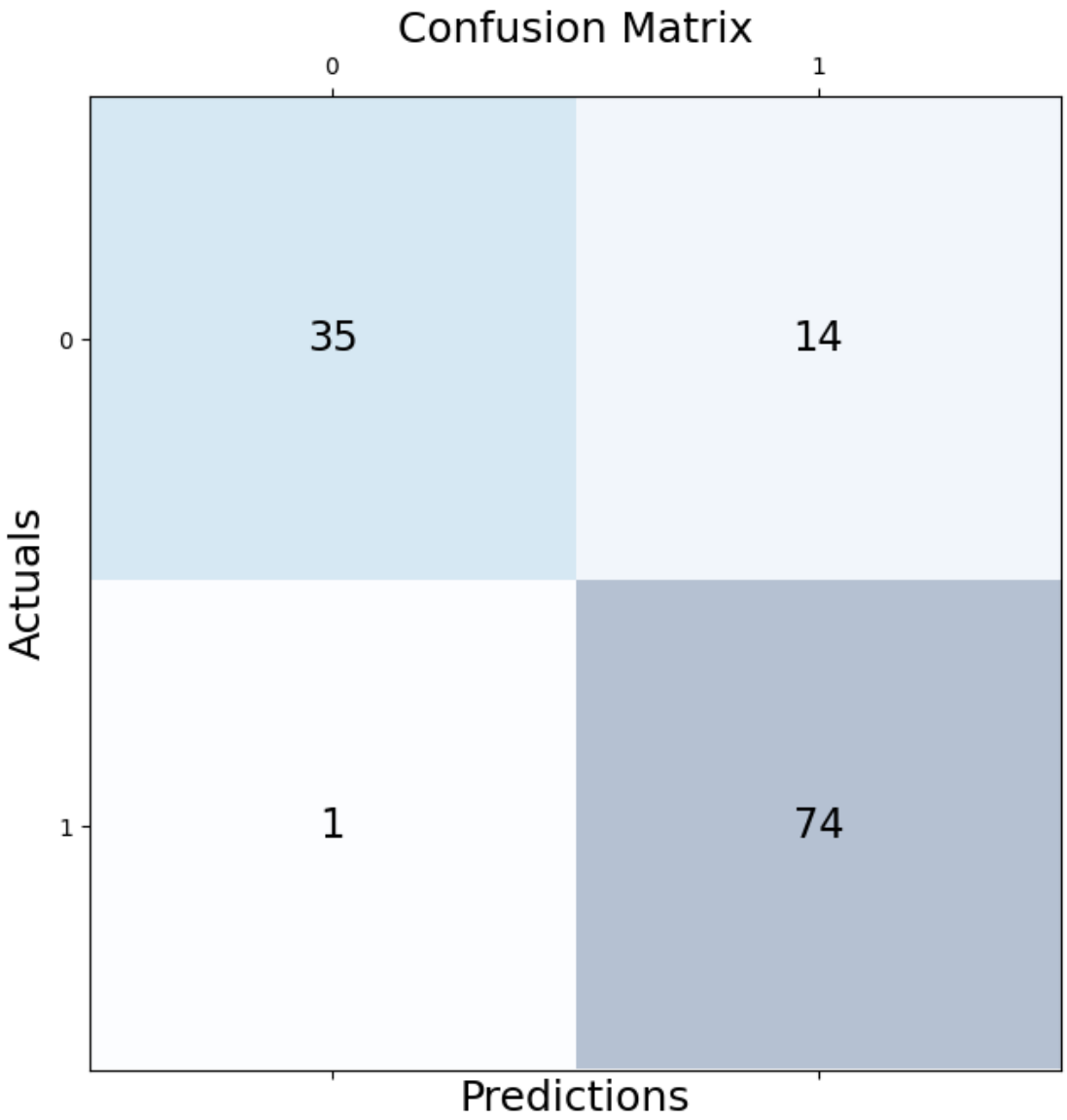
2.Dataset2 :

Test dataset : ../dataset/testdata2.csv

Output : ../ouput/testdata2_output.csv
Confusion matrix : ../plots/Testdata2_confusion_matrix.png

This dataset has 124 email samples which contains 49 (71%) Ham emails and 75 (29%) Spam emails.

Email classifiers are able to achieve 89.7% of accuracy on this dataset.



5.Conclusion :

We tried different datasets for training the model. Pre-processing has a noticeable effect on the model and significantly improves the speed as well as precision of the model. Along with training data we will pre-process test data so that it can be converted into a format suitable for assigning labels.

Overall Naive Bayes is a very simple and fundamental algorithm with very good practical accuracy.

6.Code Execution:

Install the required dependency

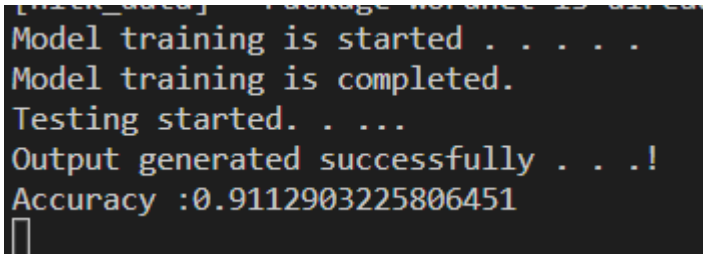
```
$ pip install -r requirements.txt
```

Runing a main.py or test.py

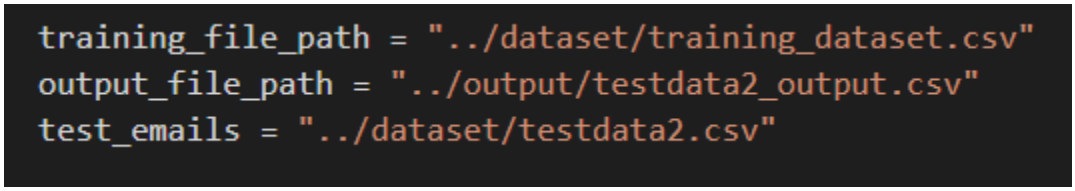
```
$ python main.py or $ python test.py
```

Training a model takes around 3-4 min at max.

Test.py - used to test the dataset and print the accuracy and plot the confusion matrix. We can see the output of the classifier under the folder **../output/**



We need to change the below variable in the Test.py to perform the test against the different dataset.



Main.py : This file get the all email from **../test** folder(as mentioned in the assignment) and produce the output to the file **../output/output.csv**

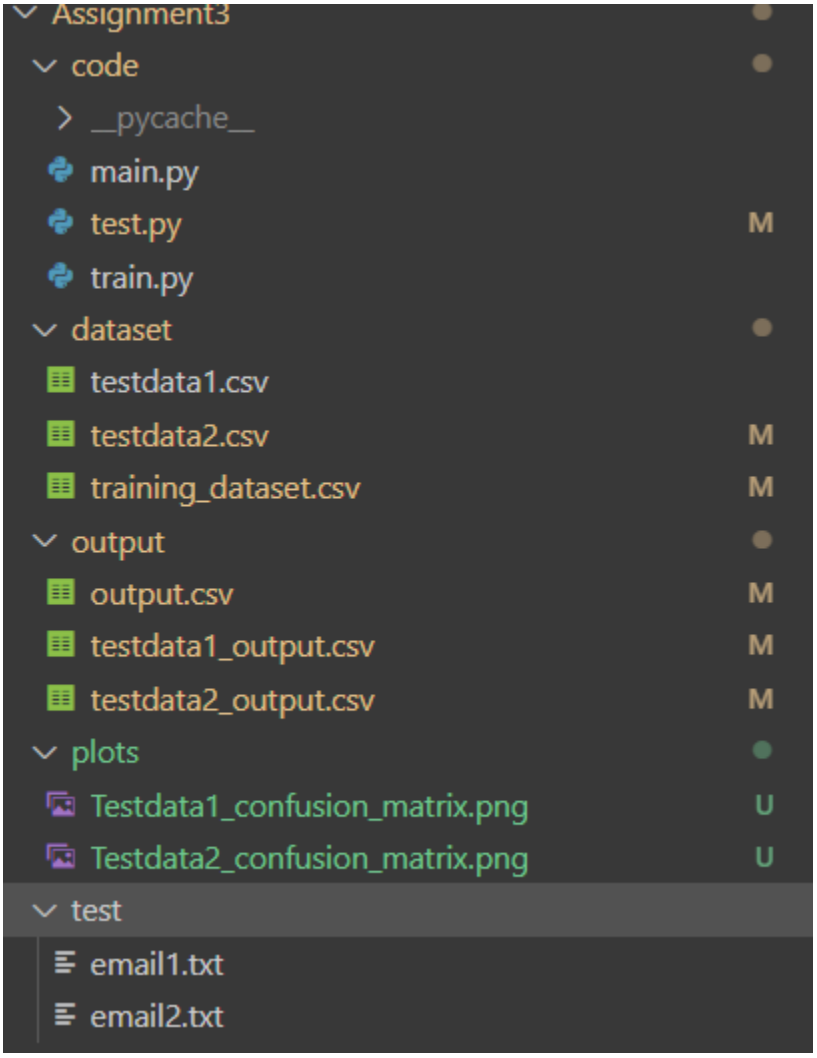
7.Code Structure :

code : This Contains the code implementation for the assignment .

main.py : this file has to code to get the email from the ../test folder and execute the classifier and produce the output to the ../output/output.csv.

test.py : This file has to code the executed classifier code to test the test dataset and produce the output and plots.

train.py : This file has the code to train the model for the email spam classifiers.



dataset : Contains dataset for the training the model and testing datasets
output : Contains output of the classifiers.

plots : Contains confusion matrix plots
test : Contains test emails