Hanoi University of Science and Technology

Machine Learning and Data Mining Report

<u>Project title</u>: Email spam filtering(Naïve Bayes classification)

Class code: 131081

Group code: 6

Group members: Nguyễn Minh Tuấn – StudentID :20194876

Nguyễn Thụ Hiếu – StudentID: 20194761

Submitted to : Mr. Nguyen Nhat Quang,PhD

I.Introduction:

Spam email (or junk email) is one of the biggest problems in the world of Web 2.0. According to the statistic from statista.com[1],from October 2020 to September 2021,spam emails accounted for over 280 billions of total 336 billion emails sent over the Internet.

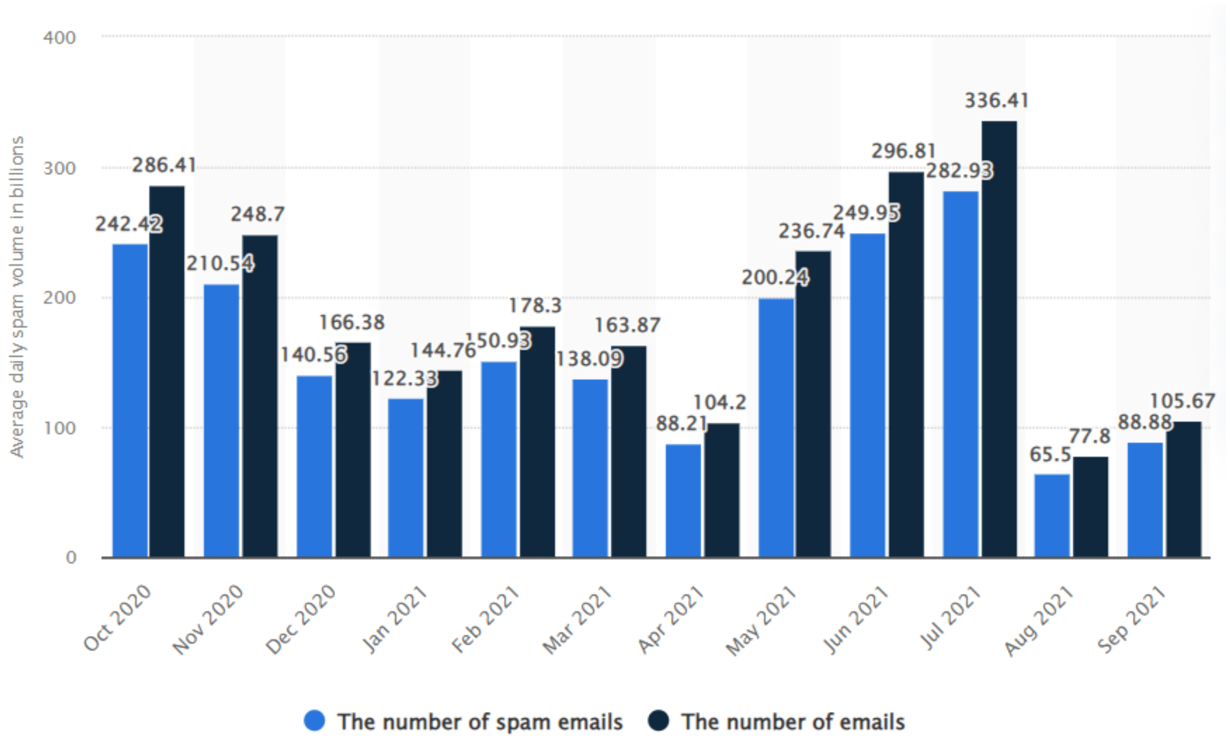


Figure 1:
Spam email statistic-Source: statista.com

Spam emails are uninvited email sent by the people you don't know (aka spamers).Often, spam email's contain fraudulent information or promoting information for a product of unknown origin that can endanger and annoy the recipient.

Understanding the threats that spam emails can bring, our team goal is to build a machine learning model to identify an email is spam or not. The input is the content of email while output is a label 'SPAM' or 'HAM'(not spam).For example with the input:

"Subject: Looking for medication ? We`re the best source. It is difficult to make our material condition better by the best laws, but it is easy enough to ruin it by bad laws. Excuse me . . . :) You just found the best and simplest site for medication on the net. No prescription, easy delivery. Private, secure and easy . Better see rightly on a pound a week than squint on a million. We 've got anything that you will ever want. Erection treatment pills, anti - depressant pills , weight loss , and more ! http://splicings.bombahakcx.com/3/ Knowledge and human power are synonymous. Only high - quality stuff for low rates! 100 % money back

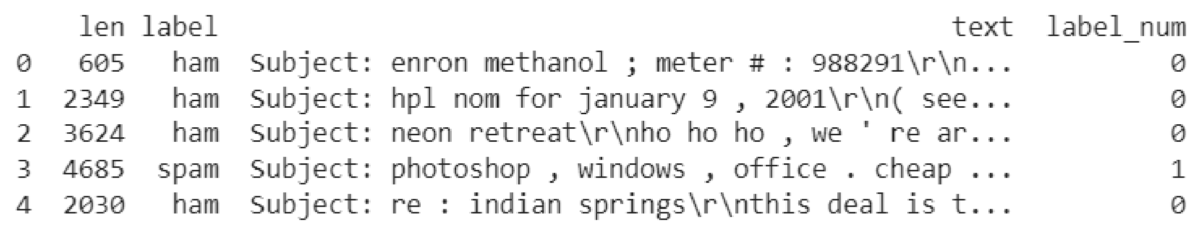
guarantee ! There is no god, nature sufficeth unto herself in no wise hath she needs an author."

The output is label “spam” because the content of the email is promotion for a suspicious medication.

In addition, the machine learning approach we apply is Naïve Bayes classification (include Multinomial Naïve Bayes and Bernoulli Naïve Bayes algorithm)

II.Dataset

The data set we use is Spam Mails Dataset from Kaggle[2].It is a csv file name “spam_ham_dataset.csv”:



```
    len label                                              text  label_num
0   605   ham  Subject: enron methanol ; meter # : 988291\r\n...          0
1  2349   ham  Subject: hpl nom for january 9 , 2001\r\n( see...          0
2  3624   ham  Subject: neon retreat\r\nho ho ho , we ' re ar...          0
3  4685  spam  Subject: photoshop , windows , office . cheap ...          1
4  2030   ham  Subject: re : indian springs\r\nthis deal is t...          0
```

Figure 2: First 5 record of spam_ham_dataset

As can be seen from first 5 records ,the file has 4 columns :

-len:the length of the content of the email

-lable: indicate whether the email is spam or ham

-text:the content of the email

-label_num:0 is ham and 1 means spam

The datasets has total 5171 records ,including 3672 ham emails (71.01% of total records) and 1499 spam emails (28.99% of total records)

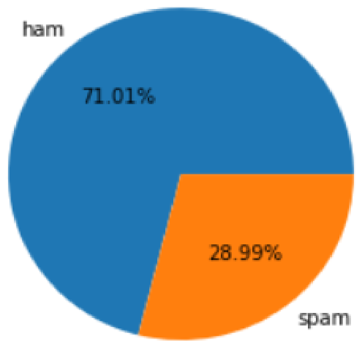


Figure 3:Ratio between spam and ham emails in the dataset

III. Algorithm

1.Overview about MAP hypothesis and Naïve Bayes Classifier:

1.1. MAP hypothesis:

Given the set H of possible hypotheses, the learner find the most probable hypothesis h ∈ H given the observed data D

Such a maximally probable hypothesis is called a maximum a posteriori (MAP) hypothesis:

$$h_{MAP} = \underset{h \in D}{arg\,max}\, P(h|D) \quad (1.1)$$

By Bayes theorem the above expression can convert to:

$$h_{MAP} = \underset{h \in H}{arg\,max} \frac{P(D|h).P(h)}{P(D)} \quad (1.2)$$

Since P(D) is the same for all classes, we can have the final formula:

$$h_{MAP} = \underset{h \in H}{arg\,max}\, P(D|h).P(h) \quad (1.3)$$

1.2. Naïve Bayes classifier

Given a training set D, where each training instance x represented as an n-dimensional attribute vector : (x_1,x_2,...x_n).A pre-defined set of classes {c_1,c_2,...,c_m}.Given a new instance z,where should we classify z into:

From MAP hypothesis, we have:

$$c_{MAP} = \underset{c_i \in C}{arg\,max}\, P(c_i|z) = \underset{c_i \in C}{arg\,max}\, P(c_i|z_1, z_2, \ldots, z_n) \quad (1.4)$$

Apply Bayes theorem :

$$c_{MAP} = \underset{c_i \in C}{arg\,max} \frac{P(z_1, z_2, \ldots, z_n|c_i).P(c_i)}{P(z_1, z_2, \ldots, z_n)} \quad (1.5)$$

Since $P(z_1, z_2, \ldots, z_n)$ is the same for all classes, the most probable class for z is identified by:

$$c_{MAP} = \underset{c_i \in C}{arg\,max}\, P(z_1, z_2, \ldots, z_n|c_i).P(c_i) \quad (1.6)$$

Apply conditional probability rule:

$$c_{MAP} = \underset{c_i \in C}{arg\,max}\, P(z_1, z_2, \ldots, z_n, c_i) \quad (1.7)$$

Apply Chain rule, $P(z_1, z_2, \ldots, z_n, c_i)$ can decompose to:

$$P(z_1, z_2, \dots, z_n, c_i) = P(z_1|z_2, \dots, z_n, c_i).P(z_2, \dots, z_n, c_i) = \cdots$$
$$= P(z_1|z_2, \dots, z_n, c_i).\dots.P(z_{n-1}|z_n, c_i).P(z_n|c_i).P(c_i) \quad (1.8)$$

However, the above formula is not easy to compute In this case, we use assumption in Naïve Bayes classifier: the attributes are conditionally independent given classification

$$P(z_1|z_2, \dots, z_n, c_i) \dots.P(z_{n-1}|z_n, c_i).P(z_n|c_i).P(c_i) = P(z_1|c_i) \dots.P(z_{n-1}|c_i).P(z_n|c_i).P(c_i)$$
$$= P(c_i).\prod_{j=1}^{n} P(z_j|c_i) \quad (1.9)$$

Finally, we have Naïve Bayes classifier to finds the most probable class for z:

$$c_{MAP} = \underset{c_i \in C}{arg\,max}\, P(c_i).\prod_{j=1}^{n} P(z_j|c_i) \quad (1.10)$$

2.Apply Naïve Bayes classifier in the spam email filtering

2.1. Training

*Problem definition:

Given a training set D, where each training instance is a document(email content) associated with a class label :D ={(d$_k$,c$_i$)}.

A Pre-defined set of class labels: C={spam,ham}

*The training algorithm

From the documents collection contained in the training set D, extract the vocabulary of distinct keywords: T={t$_1$,t$_2$,...t$_j$}

Denote D_c the set of documents in D with label c

For each class c :

-Compute the priori probability of class c$_i$:

$$P(c) = \frac{|D\_c|}{|D|} \quad (2.1)$$

-For each term t$_j$, compute the probability of term t$_j$ given class c$_i$:P(t$_j$|c)

There are two formula we will apply to calculate the P(t$_j$|c): Multinomial Naïve Bayes formula and Bernoulli Naïve formula (explain soon in the **section 2.3**)

2.2.Classifying

Given a new email with content d ,assign the class label for this.

*From d ,extract a set T_d of all terms appear in this document

Apply Naïve Bayes assumption method: the words in the email are conditionally independent of each other and also independent of its position in the documents

For each class c compute the likelihood of document d given c:

$$P(c).\prod_{t\in T\_d} P(t|c) \quad (2.2)$$

The formula to calculate P(t|c) in case t is not known by the vocabulary T will be explained soon in section **2.3**

Classify document d in class c*

$$c *= \underset{c\in C}{arg\,max}\ P(c).\prod_{t\in T\_d} P(t|c) \qquad (2.3)$$

However, P(t|c)<1 for every t and class c. So in case the number of words in T_d is very large : $\lim_{n\to\infty}(\prod_{t\in T\_d} P(t|c)) = 0$

To solve this problem we will use logarithmic function of probability

$$c *= \underset{c\in C}{arg\,max} \left[\log\left(P(c).\prod_{t\in T\_d} P(t|c)\right)\right] \quad (2.3)$$

$$= \underset{c\in C}{arg\,max} \left[\log(P(c) + \sum_{t\in T\_d} \log P(t|c)\right] \quad (2.4)$$

2.3: Formula to calculate P(t_j|c)

a. Multinomial Naïve Bayes

Denote:

-N_{cj} is the number of occurrences of word t_j in the set of documents in D with label c

-N_c is the total length of documents with label c (sum of the length of all document whose label is c) including the duplicate words

We have :

$$P(t_j|c) = \frac{N_{cj}}{N_c} \quad (2.5)$$

However, there is a problems with that formula. If a new word (never appear in documents with label c in the training set) appears, its probability given c is 0 ,that makes the right hand side of equation (2.3) become zero

To solve this problem, Laplace smoothing method should be use:

$$P(t_j|c) = \frac{N_{cj} + 1}{N_c + |T|} \quad (2.6)$$

with |T| is the number of words in vocabulary T

So if a new word t appear ,the its probability given c is:

$$P(t|c) = \frac{1}{N_c + |T|} \quad (2.7)$$

b. Bernoulli Naïve Bayes :

Denote:

-N_{cj_d} is the number of documents with label c that contain the word t_j

-N_{c_d} is the total number of documents with label c

We have :

$$P(t_j|c) = \frac{N_{cj\_d}}{N_{c\_d}} \quad (2.8)$$

Similar to the problem that we mention in Multinomial Naïve Bayes ,Laplace smoothing method should be use to avoid $P(t_j|c)=0$

$$P(t_j|c) = \frac{N_{cj\_d} + 1}{N_{c\_d} + 2} \quad (2.9)$$

So if a new word t appear , its probability given c is:

$$P(t|c) = \frac{1}{N_{c\_d} + 2} \quad (2.10)$$

III. Implementation

1.Dataset pre-processing:

Pre-processing content of the email is a very important stage because the emails can contain a lot of unnecessary characters and words that can have a huge impact on the accuracy of our machine learning model.

The text pre-processing include:

-Remove all url such as https://facebook.com,.....

-Remove numbers

-Remove unnecessary characters such as punctuation, spaces (\n,\t,...),symbols and some special characters.

-Tokenizing to make an array which each element is a word

-Remove stop words(commonly used words) such as the, is , a , an ,..... using nltk[3] list of stop words in python.

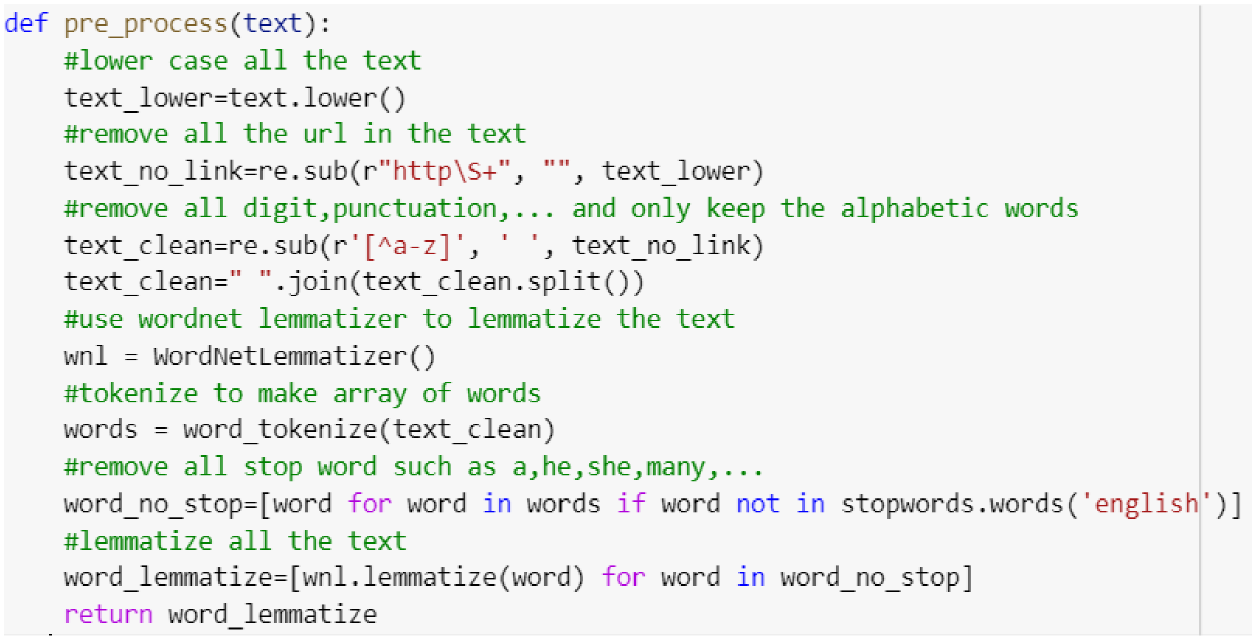
-Lemmatization (convert words to its based for) using nltk in python

-For Bernoulli Naïve Bayes classifier ,we also need to remove duplicate word .

Example:

Raw text	The dogs goes tonight.Today is too hot.Tonight is very cold {}.Please Contacts 09323893\n or http\google.com.vn .He finds the dog
After lowering	the dogs goes tonight.today is too hot.tonight is very cold {}.please contacts 09323893 or http\google.com.vn .he finds the dog
After remove URL, number, unwanted character	the dogs goes tonight today is too hot tonight is very cold please contacts or he finds the dog
After tokenizing	['the', 'dogs', 'goes', 'tonight', 'today', 'is', 'too', 'hot', 'tonight', 'is', 'very', 'cold', 'please', 'contacts', 'or', 'he', 'finds', 'the', 'dog']
After remove stop words:	['dogs', 'goes', 'tonight', 'today', 'hot', 'tonight', 'cold', 'please', 'contacts', 'finds', 'dog']
After lemmatizing	['dog', 'go', 'tonight', 'today', 'hot', 'tonight', 'cold', 'please', 'contact', 'find', 'dog']
After removing duplicate words(for Bernoulli Naïve Bayes)	['dog', 'go', 'tonight', 'today', 'hot', 'cold', 'please', 'contact', 'find']

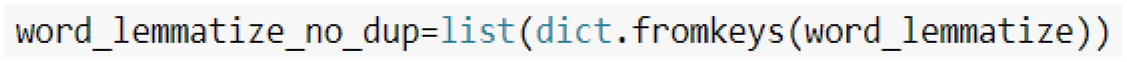
Here is our python code function for data pre-processing ,follow the instruction from https://towardsdatascience.com/ [4]



```python
def pre_process(text):
    #lower case all the text
    text_lower=text.lower()
    #remove all the url in the text
    text_no_link=re.sub(r"http\S+", "", text_lower)
    #remove all digit,punctuation,... and only keep the alphabetic words
    text_clean=re.sub(r'[^a-z]', ' ', text_no_link)
    text_clean=" ".join(text_clean.split())
    #use wordnet lemmatizer to lemmatize the text
    wnl = WordNetLemmatizer()
    #tokenize to make array of words
    words = word_tokenize(text_clean)
    #remove all stop word such as a,he,she,many,...
    word_no_stop=[word for word in words if word not in stopwords.words('english')]
    #lemmatize all the text
    word_lemmatize=[wnl.lemmatize(word) for word in word_no_stop]
    return word_lemmatize
```

Figure 4: Python code for data preprocessing

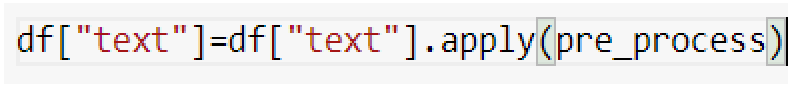
For Bernoulli Naïve Bayes ,after lemmatizing ,we remove some duplicate word to ensure each word appear once in the array.



```python
word_lemmatize_no_dup=list(dict.fromkeys(word_lemmatize))
```

Figure 5: Python code for removing duplicate key

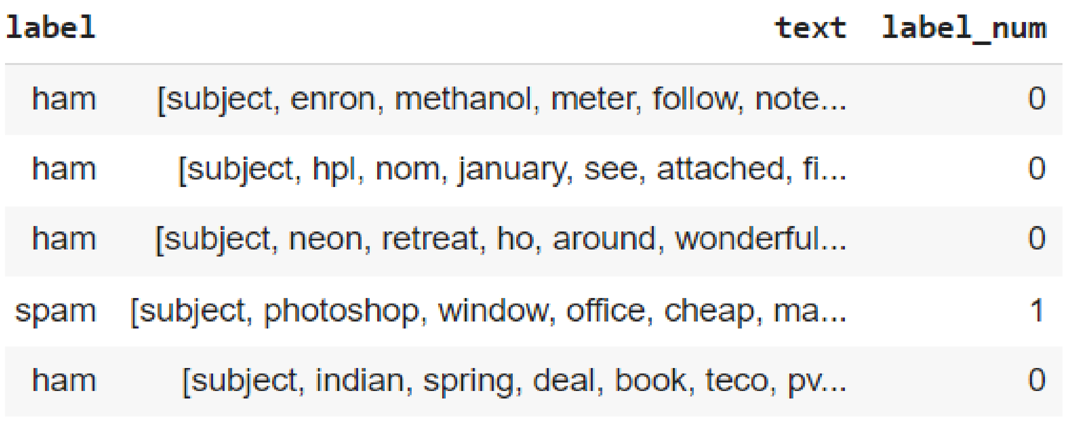
Then apply the above function to preprocess all the email from the dataset:



```python
df["text"]=df["text"].apply(pre_process)
```

Figure 6: Python code for applying pre_process for the whole dataset

Finally, the result after apply function pre_process:



label	text	label_num
ham	[subject, enron, methanol, meter, follow, note...	0
ham	[subject, hpl, nom, january, see, attached, fi...	0
ham	[subject, neon, retreat, ho, around, wonderful...	0
spam	[subject, photoshop, window, office, cheap, ma...	1
ham	[subject, indian, spring, deal, book, teco, pv...	0

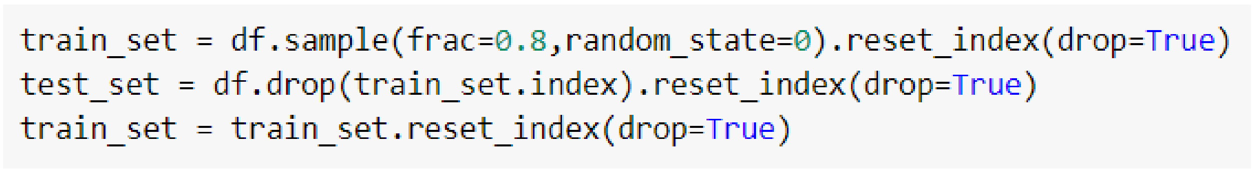
Figure 7: Result after apply pre_processing(first 5 records)

2.Training and Classifying:

2.1.Split dataset into train set and dataset

We split the dataset into training set and test set (with ratio 8:2) and also keep the ratio between ham records and spam record in both training set and test set similar to the original ratio between them .

Here is the code to split the dataset, we follow the code from: iq.opengenus[5]



```
train_set = df.sample(frac=0.8,random_state=0).reset_index(drop=True)
test_set = df.drop(train_set.index).reset_index(drop=True)
train_set = train_set.reset_index(drop=True)
```

Figure 8: Python code for splitting dataset

After splitting, we have the train set with 4137 records (including 2936 records with label ‘ham’ and 1174 records with label ‘spam’

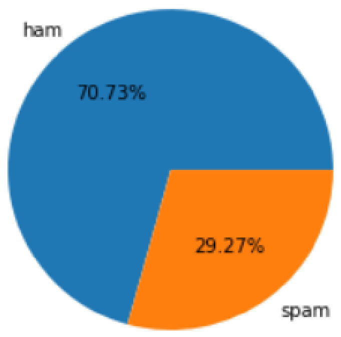


Figure 9: Ratio between ‘spam’ and ‘ham’ records in the train set

Also, the test set has 1034 records (including 714 records with label ‘ham’ and 320 records with label ‘spam’)

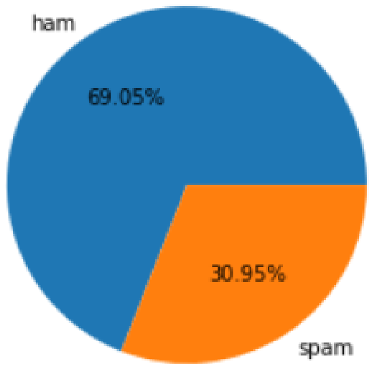
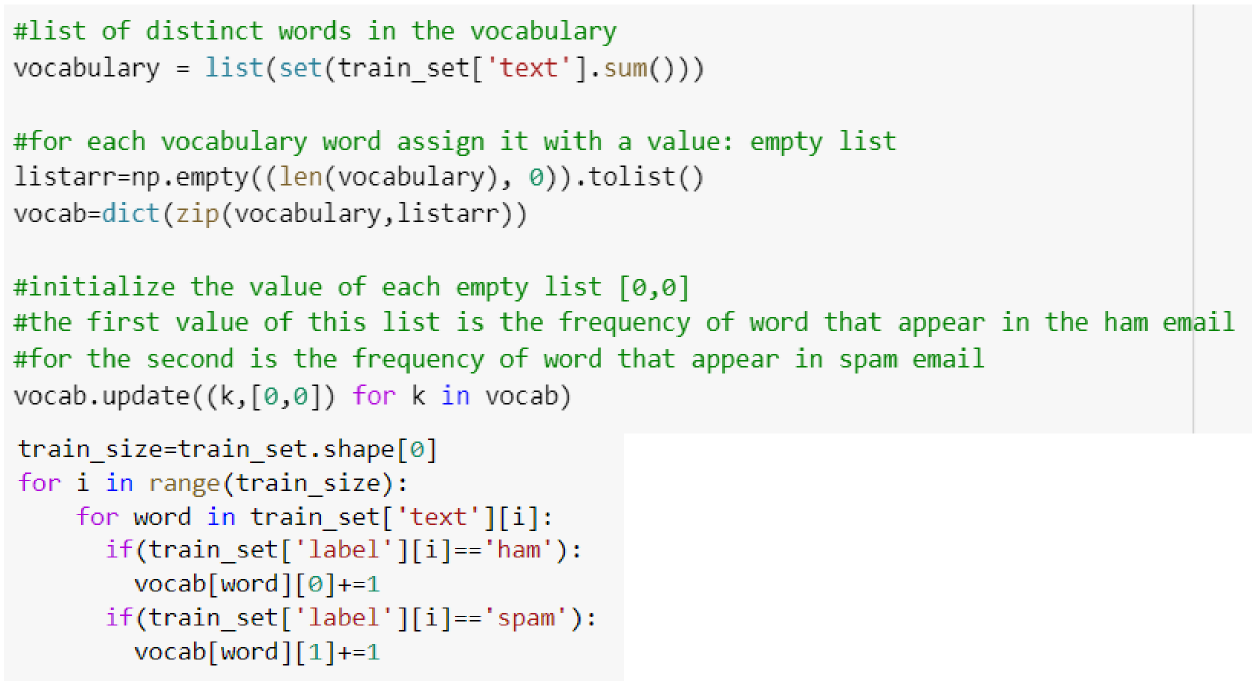


Figure 10: Ratio between ‘spam’ and ‘ham’ records in the test set

2.2. Create vocabulary

From the training set, we extract the dictionary of distinct key words:



```
#list of distinct words in the vocabulary
vocabulary = list(set(train_set['text'].sum()))

#for each vocabulary word assign it with a value: empty list
listarr=np.empty((len(vocabulary), 0)).tolist()
vocab=dict(zip(vocabulary,listarr))

#initialize the value of each empty list [0,0]
#the first value of this list is the frequency of word that appear in the ham email
#for the second is the frequency of word that appear in spam email
vocab.update((k,[0,0]) for k in vocab)
```

```
train_size=train_set.shape[0]
for i in range(train_size):
    for word in train_set['text'][i]:
      if(train_set['label'][i]=='ham'):
        vocab[word][0]+=1
      if(train_set['label'][i]=='spam'):
        vocab[word][1]+=1
```

Figure 11: Python code for creating vocabulary and calculating necessary data

We map each keyword with an array of size 2:

For Multinomial Naïve Bayes classifier:

-First element of the array is the number of occurrences of this word in all ham record of training set

-First element of the array is the number of occurrences of this word in all spam record of training set

For Bernoulli Naïve Bayes classifier:

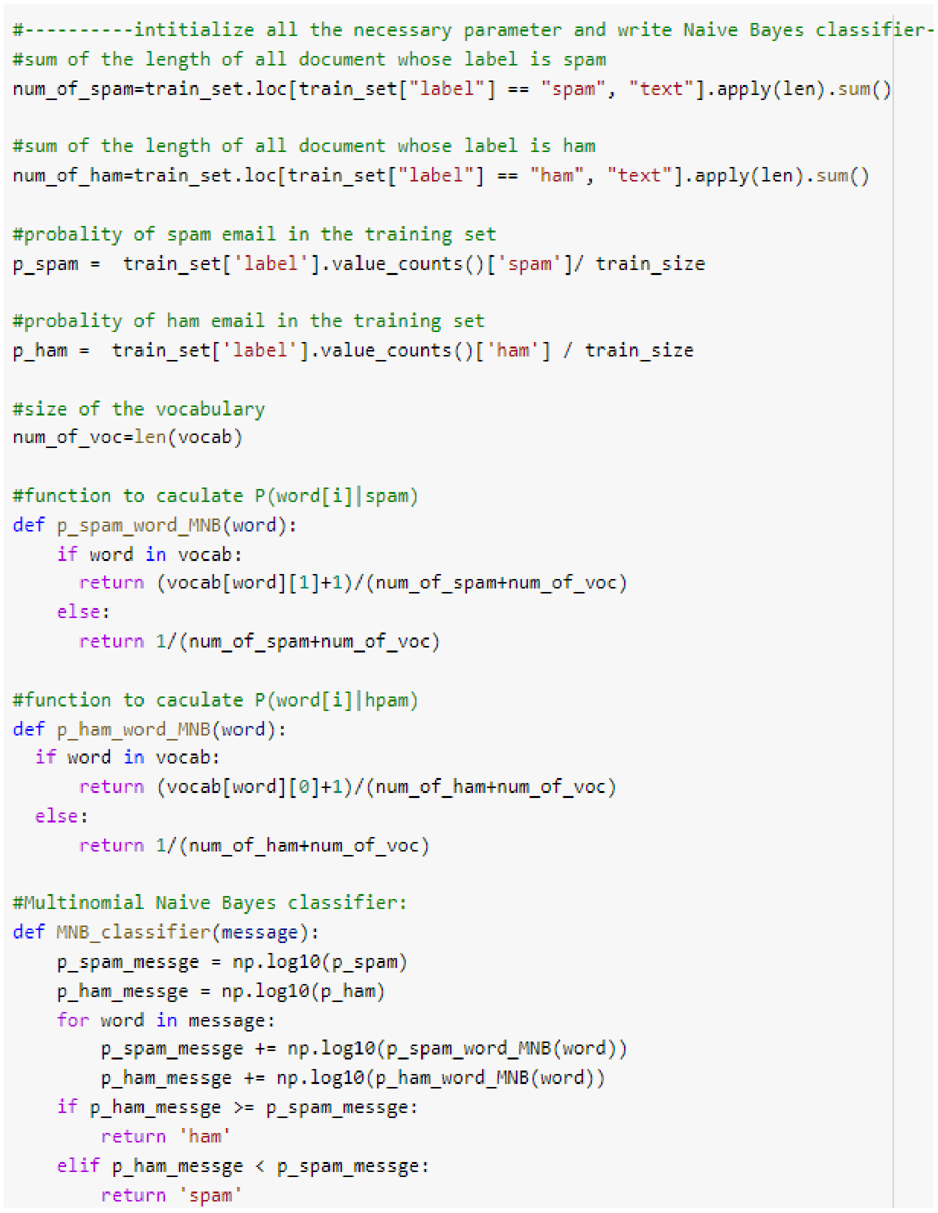
-First element of the array is the number of record with label “ham” containing that word

-Second element of the array is the number of record with label “spam” containing that word

2.3.Write the classifier

2.3.1.Multinomial Naïve Bayes Classifier

From equation (2.4),(2.6),(2.7) ,we implement our own Multinomial Naïve Bayes Classifier ,insprired by the code from iq.opengenus[5]

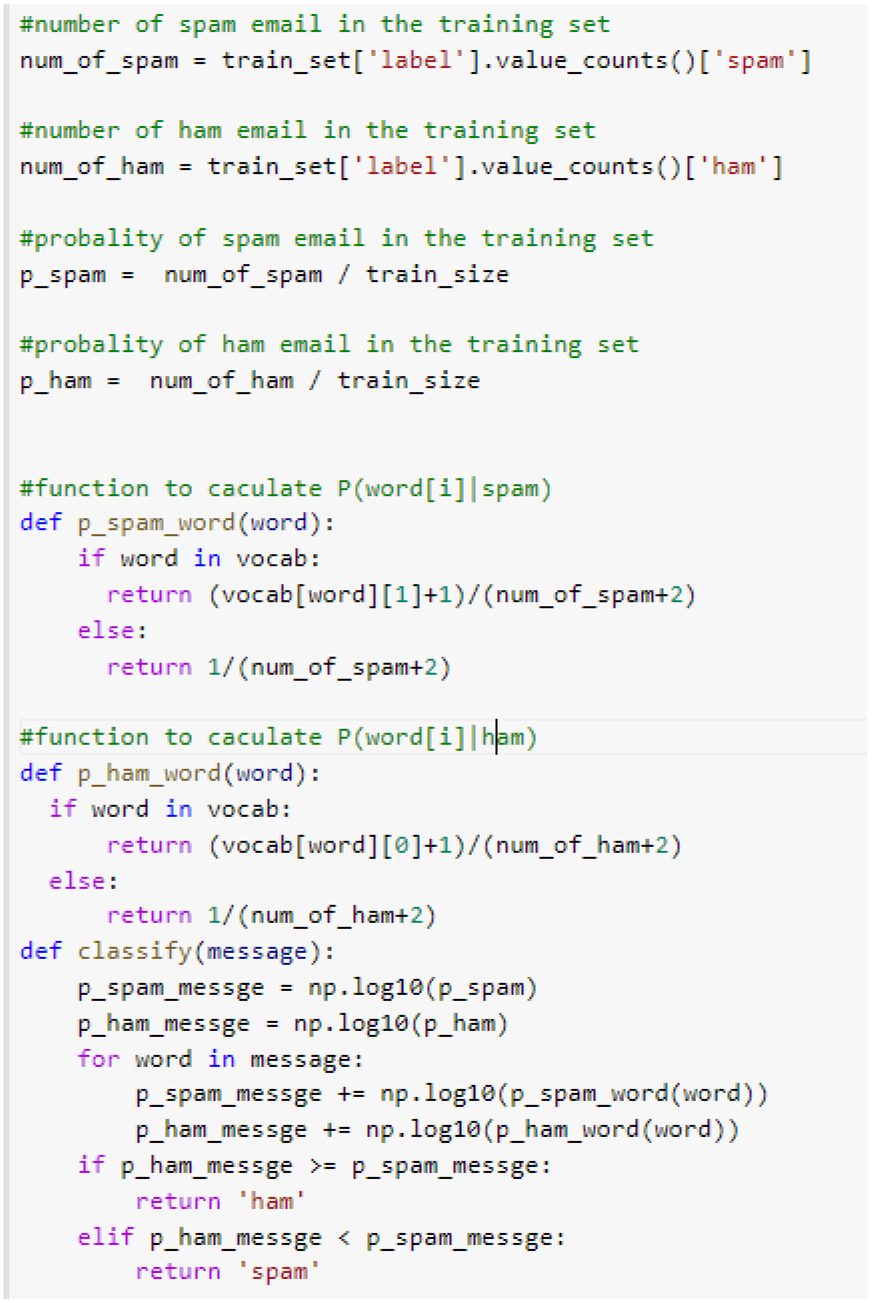


```
#----------intitialize all the necessary parameter and write Naive Bayes classifier-
#sum of the length of all document whose label is spam
num_of_spam=train_set.loc[train_set["label"] == "spam", "text"].apply(len).sum()

#sum of the length of all document whose label is ham
num_of_ham=train_set.loc[train_set["label"] == "ham", "text"].apply(len).sum()

#probality of spam email in the training set
p_spam =  train_set['label'].value_counts()['spam']/ train_size

#probality of ham email in the training set
p_ham =  train_set['label'].value_counts()['ham'] / train_size

#size of the vocabulary
num_of_voc=len(vocab)

#function to caculate P(word[i]|spam)
def p_spam_word_MNB(word):
    if word in vocab:
      return (vocab[word][1]+1)/(num_of_spam+num_of_voc)
    else:
      return 1/(num_of_spam+num_of_voc)

#function to caculate P(word[i]|hpam)
def p_ham_word_MNB(word):
  if word in vocab:
      return (vocab[word][0]+1)/(num_of_ham+num_of_voc)
  else:
      return 1/(num_of_ham+num_of_voc)

#Multinomial Naive Bayes classifier:
def MNB_classifier(message):
    p_spam_messge = np.log10(p_spam)
    p_ham_messge = np.log10(p_ham)
    for word in message:
        p_spam_messge += np.log10(p_spam_word_MNB(word))
        p_ham_messge += np.log10(p_ham_word_MNB(word))
    if p_ham_messge >= p_spam_messge:
        return 'ham'
    elif p_ham_messge < p_spam_messge:
        return 'spam'
```

Figure 12:Multinomial Naïve Bayes Classifier in Python code

2.3.2. Bernoulli Naïve Bayes Classifier

From equation (2.4),(2.9),(2.10) ,we implement our own Bernoulli Naïve Bayes Classifier



```
#number of spam email in the training set
num_of_spam = train_set['label'].value_counts()['spam']

#number of ham email in the training set
num_of_ham = train_set['label'].value_counts()['ham']

#probality of spam email in the training set
p_spam =  num_of_spam / train_size

#probality of ham email in the training set
p_ham =  num_of_ham / train_size


#function to caculate P(word[i]|spam)
def p_spam_word(word):
    if word in vocab:
      return (vocab[word][1]+1)/(num_of_spam+2)
    else:
      return 1/(num_of_spam+2)

#function to caculate P(word[i]|ham)
def p_ham_word(word):
  if word in vocab:
      return (vocab[word][0]+1)/(num_of_ham+2)
  else:
      return 1/(num_of_ham+2)
def classify(message):
    p_spam_messge = np.log10(p_spam)
    p_ham_messge = np.log10(p_ham)
    for word in message:
        p_spam_messge += np.log10(p_spam_word(word))
        p_ham_messge += np.log10(p_ham_word(word))
    if p_ham_messge >= p_spam_messge:
        return 'ham'
    elif p_ham_messge < p_spam_messge:
        return 'spam'
```

Figure 12: Bernoulli Naïve Classifier in Python code

IV. Experimental Result:

1.Multinomial Naïve Bayes:

For the test set:

*Confusion Matrix:

		Predict label	
		Ham	Spam
Actual Label	Ham	710	4
	Spam	10	310

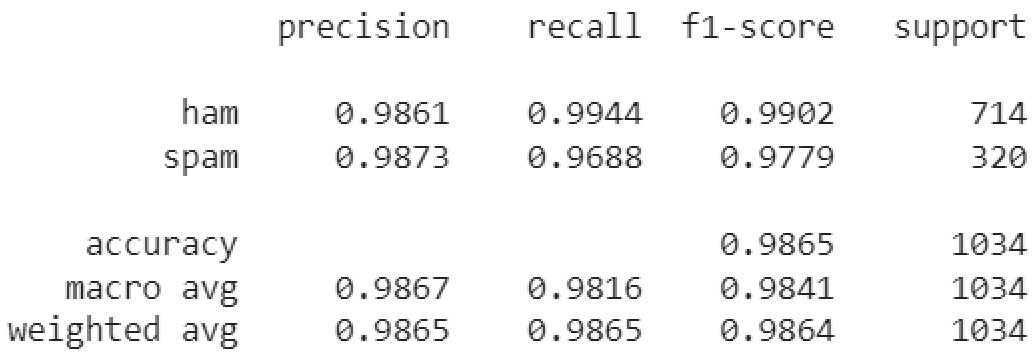
As we can see from the confusion matrix:

-710 ham emails and 310 spam emails are recognized correctly

-4 ham emails are recognized incorrectly as spam emails

-10 spam emails are recognized incorrectly as ham emails

*Classification report from sklearn.metrics[6]



```
              precision    recall  f1-score   support

         ham     0.9861    0.9944    0.9902       714
        spam     0.9873    0.9688    0.9779       320

    accuracy                         0.9865      1034
   macro avg     0.9867    0.9816    0.9841      1034
weighted avg     0.9865    0.9865    0.9864      1034
```

Figure 13: Classification report on test set using Multinomial Naïve Bayes

*Accuracy score: (710+310)/1034=98.64%

2.Bernoulli Naïve Bayes:

For the test set

*Confusion Matrix:

		Predict label	
		Ham	Spam
Actual Label	Ham	657	57
	Spam	3	317

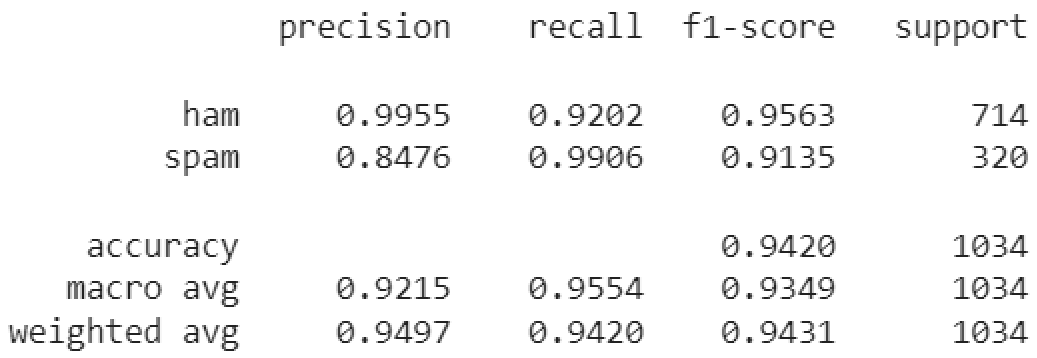
As we can see from the confusion matrix:

-657 ham emails and 317 spam emails are recognized correctly

-57 ham emails are recognized incorrectly as spam emails

-3 spam emails are recognized incorrectly as ham emails

*Classification report from sklearn.metrics[6]



```
              precision    recall  f1-score   support

         ham     0.9955    0.9202    0.9563       714
        spam     0.8476    0.9906    0.9135       320

    accuracy                         0.9420      1034
   macro avg     0.9215    0.9554    0.9349      1034
weighted avg     0.9497    0.9420    0.9431      1034
```

Figure 14: Classification report on test set using Bernoulli Naïve Bayes

*Accuracy score: (657+317)/1034=94.20%

3.Compare two classifier:

Denote:

TP(True Positive) : the number of ham emails that are recognized correctly

FP(False Positive): the number of spam emails that are recognized incorrectly as ham email

TN(True Negative):the number of spam emails that are recognized correctly

FN(False Negative):the number of ham emails that are recognize incorrectly as spam email

Evaluation Measure	Multinomial Naïve Bayes	Bernoulli Naïve Bayes
TP	**710**	**657**
FP	**10**	**3**
TN	**310**	**317**
FN	**4**	**57**
Precision(macro avg)	**98.67%**	**92.15%**
Recall(macro avg)	**98.16%**	**95.54%**
F1-score(macro avg)	**98.41%**	**93.49%**
Accuracy	**98.64%**	**95.20%**

From the comparation , we can see that ,on the used dataset:

-Multinomial Naïve Bayes classifier has better accuracy than Bernoulli Naïve Bayes

-Multinomial Naïve Bayes only recognized incorrectly 4 ham email as spam email while this number in Bernoulli Naïve Bayes is 57. Therefore using Bernoulli Naïve Bayes classifier will be not “safe” .Because too many ham email is recognized incorrectly, which is very bad in the real life.

V. Our difficulties in the course project implementation:

1.At the start of the project

1.1: Difficulties

At the start of the project, we had difficulty finding materials and instructions. Finally, we decided to follow the instructions from iq.opengenus[5].Although ,on their used data set ,we could achieve an accuracy of 99% ,on our chosen Kaggle dataset ,we can only get 30% accuracy

1.2: Overcome

We tried to read the slides again ,found more materials to know more about our project and finally achieved the better accuracy

2. Try to implement the classifier

1.1: Difficulties

Although ,we knew we could use MultinomialNB and BernoulliNB classifier from sklearn library of Python to save time, we wanted to write our own classifier (base on the Multinomial and Bernoulli Naïve Bayes algorithm) to understand the project more deeper. However ,no matter how many times we tried, we always got very poor accuracy

1.2: Overcome

Finally ,we realized we forgot to remove stop word and lemmatize all the text before training (because the from the instruction we follow in iq.opengenus[5], they only remove number ,hyperlink and digits).Then we try to use nltk of python to remove all noises and unnecessary words.

Next ,we found that we didn’t consider if a new word (not known by the vocabulary) appear that make many our calculations always become zero.Finally, resolved it by apply Laplace smoothing (equation (2.7),(2.10))

VI.Conclusion.

Naïve Bayes classifier is easy-to-implement algorithm and is a good starting point for any text classification problem. Despite its conditional independence assumption that rarely happening in real life, Naïve Bayes shows a good performance in email spam filtering problem.

Our project use two algorithm Multinomial and Bernoulli Naïve Bayes , and both faces the "zero probability" when a new word that was not available in the train set appear. That's one of the disadvantages of Naïve Bayes algorithm ,and we had to use Laplace smoothing method to resolve this issue. Also, as can be seen from the experimental result, on the used dataset, Multinomial Naïve Bayes has better accuracy than Bernoulli Naïve Bayes

In the future, we hope we can build a machine learning model to classify email that write in Vietnamese. It can be very difficult and will take a lot of effort since Vietnamese email spam dataset is very hard to find.

References:

[1] Email spam statistics: https://www.statista.com/statistics/1270424/daily-spam-volume-global/

[2] Spam ham dataset : Spam Mails Dataset | Kaggle

[3] Natural Language toolkit: https://www.nltk.org/

[4] How to design a spam filtering system : https://towardsdatascience.com/email-spam-detection-1-2-b0e06a5c0472

[5] Filtering spam using Naïve Bayes: https://iq.opengenus.org/filtering-spam-using-naive-bayes/

[6] Scikit learn metrics and scoring: https://scikitlearn.org/stable/modules/model_evaluation.html

[7] Slides from PhD Nguyen Nhat Quang, Hanoi University of Sciene and Technology