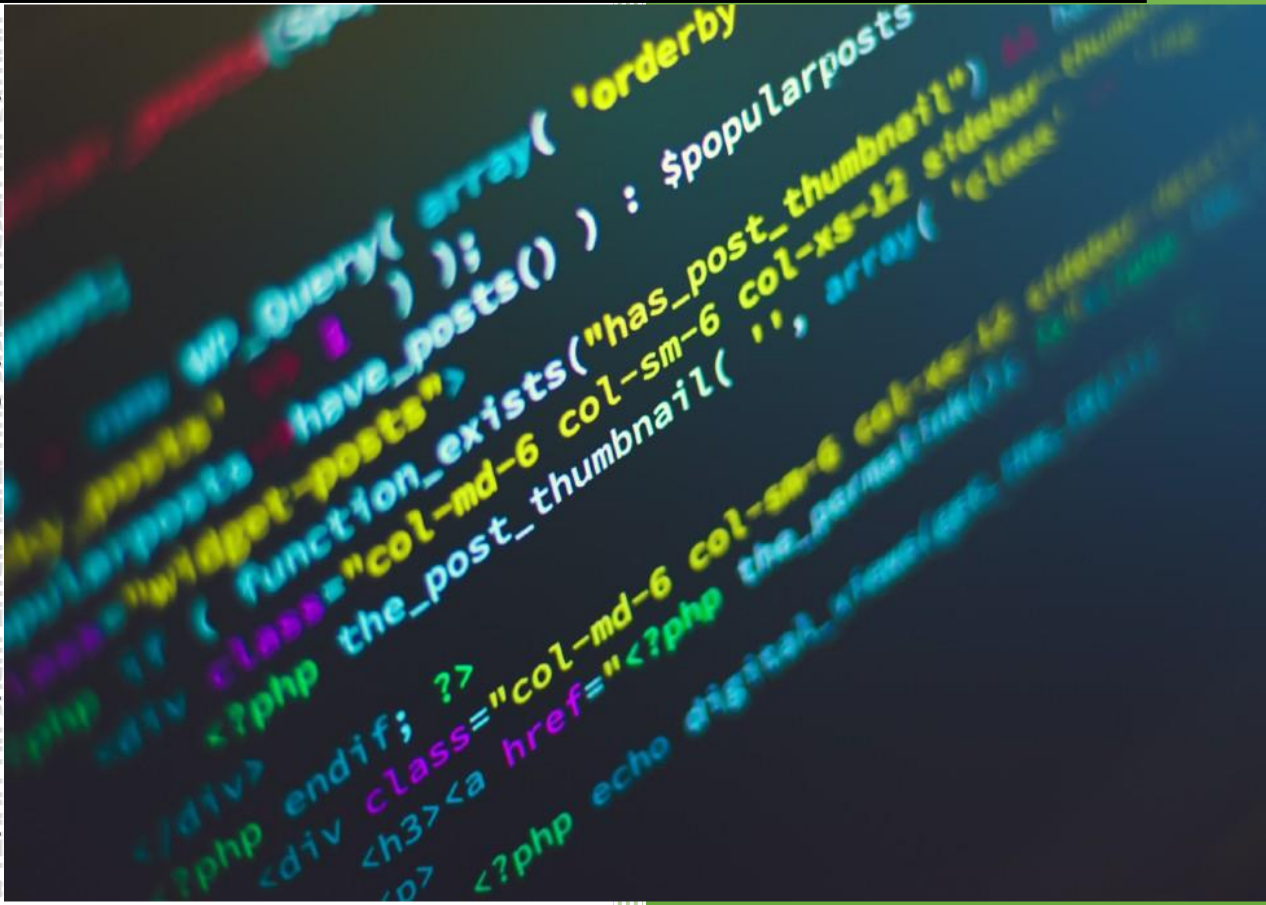
2023

CheckMail: Spam Email Detection

A report on spam email detecting system based on Python.



Udaya Singh Rai

Student No.: 4129328

School of Engineering,

London South Bank University

Abstract: Natural Language Processing (NLP) is a sector in Artificial Intelligence (AI) that deals with helping computers understand how humans communicate. There is a wide range of application of NLP such as smart assistance, predictive texts, data analysis, etc. One of such applications of NLP is called spam email detection. Spam emails are the unsolicited and unwanted junk emails sent out in bulk to an indiscriminate recipient list. Typically, spam is sent for commercial purposes. Spam email is one of the important issues of today's internet, conveying money related harm to organizations and irritating individual clients. It carries many risks like risk of identity theft, financial damage or corrupting your computers with malwares. There are many methods use to detect and filter such emails. One of the most popular among these methods is binary classification (a supervised learning algorithm that categorizes new observations into one of two classes). One of the most applied binary classifiers is Naïve Bayesian Classifier which works based on Bayes' Theorem assuming that the presence of a particular feature in a class is not related to the presence of any other features. So, this paper reports on the development of a binary classifier (Naïve Bayesian Classifier) to fulfil the purpose of spam email detection along with the modules and methodologies used, implementation and performance of the classifier its troubleshooting and its future scopes.

Table of Contents

1. INTRODUCTION

The growth and the development in AI have gone above and beyond what a human could possibly imagine. The tools and techniques have been demonstrating wonders in every sector of human advancement. Along with all other aspects of AI, NLP has contributed a very important role in bringing intelligent systems to where they stand today. The history of NLP can be traced back to the 1950s where it began as the common field of research for AI and linguistics [1]. In the beginning, NLP was different from text information retrieval (IR). IR is an aspect that employs highly scalable techniques based on statistics for indexing and searching large volumes of text accurately and efficiently [2]. However, NLP and IR have intersected and converged to some extent over time. And in present days, NLP borrows from IR in many very diverse fields, requiring NLP researchers and developers in current time to broaden their mental base of knowledge significantly.

NLP has wide varieties of usages in different AI aspects such as machine translation, speech recognition and text mining [3]. And comparatively, the contribution of NLP in text mining is more significant. Text mining, also known as text data mining, can be understood as the procedure of converting unstructured text into a structured format to take out meaningful patterns and new insights as a result. By applying advanced analytical techniques of NLP, such as Naïve Bayes, Support Vector Machines (SVM), and other deep learning algorithms, one can explore and discover hidden relationships within their unstructured text data. These unstructured text data can be anything that is written or spoken in human language such as news articles, tweets, comments in online social sites, emails, articles, etc. In this paper, we will focus majorly on email or spam emails to be precise.

The act of distribution unsolicited messages, optionally sent in bulk, using email is known as spam email. On the other hand, emails of the opposite nature and are useful are known as ham [4]. The origination of the word "spam" came from "Shoulder Pork HAM", which is a canned meat that was precooked and marketed in 1937, and eventually, the name was allocated to the digital mailing junks [5]. Spammers send out such spam emails generally for marketing purposes and to unfold more malicious operations such as financial disruption, reputational damage, and identity theft, both at institutional and personal levels [6]. Along with spam email, the practice of spamming is spreading speedily in other online communication sites as well. One of the main reasons for spammers operating such illegal and unethical syndicate is the financial motivation and an estimation of around USD 3.5 million annually has been made as the earning of spammers from such spamming [7].

As mentioned previously, it is possible for us to filter and be safe from such kinds of spam emails with the aid of suitable advanced NLP analytical techniques like Naïve Bayes. Because such analytical techniques have algorithms and functions that we can use to structure the spam emails and analyze their true intention and hence categorize them into spam or a ham. So, in this paper, we are going to look at the implementation of Naïve Bayes using Python programming language to detect the spam emails using a Graphical User Interface (GUI) that we will refer to as CheckMail. This paper has been divided into different sections for better understanding. In section 2 of this paper, we will discuss spam emails and their causes and effects in detail. Section 3 discusses Naïve Bayes and its different functions and techniques that can be (is) used to detect spam emails. Section 4 consists of the explanation of implementation, fine tuning and developing of the detecting system along with the challenges faced during the process. A brief analysis of the performance of the system and its flowchart is included in section 5 of this report. In Section 6 we will discuss the challenges faced and how we overcame those challenges. Section includes the individual contribution of our team members to this project and Section 8 will conclude the report.

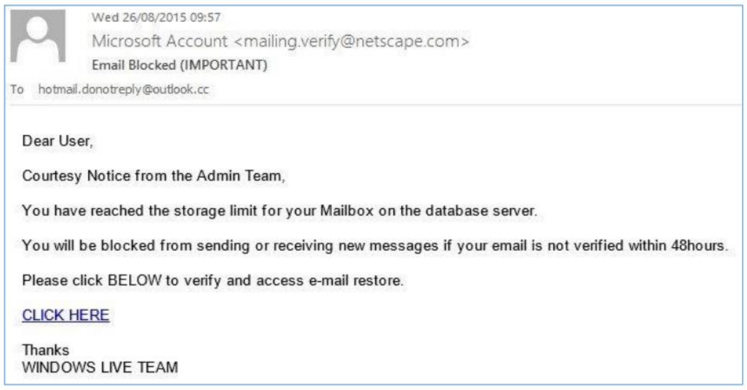
2. SPAM EMAIL

Over the years, the means of communication has changed quite a lot of times. Amongst today's generation, email is one of the most popular ways of communication. The statistics regarding this generation adopting email as a way of communication are quite astonishing. The studies show that there were about 5.5 billion email accounts as of 2017 which are active and are being used till these days and the number is expected to grow over the years [8]. And the exchange of spam emails among these numbers is also huge. As per the studies, about 236 billion emails were exchanged daily as of 2018 [9] and among these emails, around 53.5% are only spam emails [8]. In fact, it was observed that in 2018, approximately 14.5 billion spam emails exchanged in every 24 hours and the number never stopped going up [7].

Many businesses have lost a lot of finances due to such spamming emails. According to research, the financial loss faced by different companies and businesses because of this kind of spam emailing may go exponentially up in time of just few years. And this loss was projected to accumulate to about USD 257 billion from 2012 till the end of 2020 with an annual damage of around USD 20.5 billion [7]. Even though the United States had been the largest source of such kinds of spam emailing, in the present days, spam emails occur from every part of the world.

Because of the above-mentioned facts, there are numerous attacks that are being constantly targeted on users worldwide. Examples of such attacks are email phishing [10] - [11], email spoofing [12] - [13], various types of phishing attacks such as spear phishing, clone phishing, whaling, covert redirect etc. Along with spoofing and phishing attacks of such kinds, other variations like clickjacking have also emerged within spam. Hackers have tried hard to battle the anti-spam projects by hiding the text behind images [14]. According to the demonstration given by Chung-Man [15], it is possible that phishing alerts lead to a significant negative return on stocks or global firms' market value.

Spam email does possess a threat to financial and social damage at personal and organizational stages. With billions of spam emails being exchanged daily, it is hard for all the recipients to recognize it and be safe from it. This results in financial loss on a local and global scale. And to eradicate such spam, we can use the techniques offered by NLP such as Naïve Bayes and develop a system that can detect and filter spam emails and help the users stay safe from being scammed. The use of such techniques will be discussed in the following section of this paper.



Wed 26/08/2015 09:57
Microsoft Account <mailing.verify@netscape.com>
Email Blocked (IMPORTANT)
To hotmail.donotreply@outlook.cc

Dear User,

Courtesy Notice from the Admin Team,

You have reached the storage limit for your Mailbox on the database server.

You will be blocked from sending or receiving new messages if your email is not verified within 48hours.

Please click BELOW to verify and access e-mail restore.

CLICK HERE

Thanks
WINDOWS LIVE TEAM

Fig.1: Example of spam email.

3. Naïve Bayes Classifier

Naïve Bayes Classifiers (NB) are the Bayesian classifiers that assign the most likely class to an example as described by its feature vector assuming unrealistically that the features are independent given class. Mathematically, the assumption made by the NB is $P(A|B) = \prod_{i=1}^{n} P(A_i|B)$, where A = ($A_1$, A_2…….A_n) is a feature vector and B is a class. Even though the NB classifier works under such an unrealistic assumption, it is used quite often and has proven to be effective in several practical usages like text classification, managing performance of system, and medical diagnosis [16] - [17].

In other words, Naïve Bayes is a classification technique based on Bayes' Theorem with an independent assumption among predictors. Generally, a Naïve Bayes classifier assumes that the presence of a particular feature in a class has no relation to the presence of any other feature. For instance, some fruit might be an apple if it is round, red, and about 1.5 inches in radius. Even if these features depend on the existence of other features or on each other, all these properties independently contribute to the probability that this fruit is an apple and that is why it is known as 'naïve' [18].

3.1. Bayes' Theorem

Let us assume a sample space X which is considered as "evidence" in Bayesian terms and X = {x_1, x_2…., x_n} where the components x_1, x_2…., x_n represent values made on a set of n number of attributes. If we make a hypothesis (say H) such that the data X belongs to a specific class C. Our main aim is to determine probability that our hypothesis (H) holds true given the evidence (X) i.e., P(H|X) in mathematical language for the problems raised for classification. In general, we are seeking for the probability that the sample X belongs to class C under the circumstance that we are aware of the attribute of X. P(H|X) is also known as posteriori probability of X under the condition H provided.

According to Bayes' Theorem, this probability can be calculated as follows:

$$P(H|X) = \frac{P(X|H)\,P(H)}{P(X)}$$

Where P(H|X) is a probability that our hypothesis H holds given the attribute of sample X.
P(X|H) is a probability that the sample holds given our hypothesis is true.
P(H) is a probability that the hypothesis is true for any sample.
P(X) is a probability that the attribute of the sample is met.

4. Implementation

The spam email detection system that we developed was fully based on Python programming language. And to implement this programming language, we used Anaconda as the platform. Following sections will illustrate how we developed CheckMail (spam email detecting program) in a stepwise manner.

4.1. Data Collection and Preprocessing

For any classifier to work with high accuracy and efficiently, training the classifier correctly and with the right dataset is very important. So, on the search for an appropriate dataset we started exploring Kaggle (largest data science community with powerful tools and resources to help anyone achieve their data science goals). We were finally able to get our hands to a comma-separated values' dataset named "completeSpamAssassin.csv" in Nitisha Bharathi's portal in Kaggle. For the dataset, the like is 'https://www.kaggle.com/datasets/nitishabharathi/email-spam-dataset'.

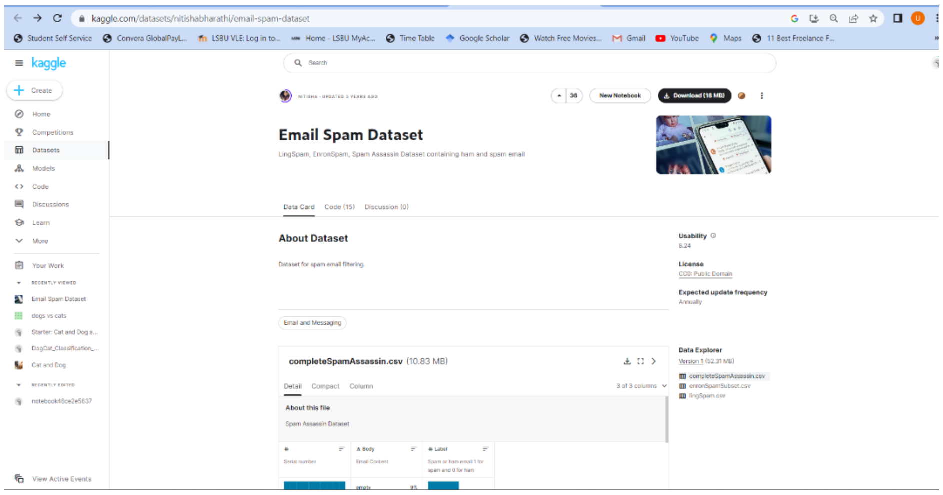
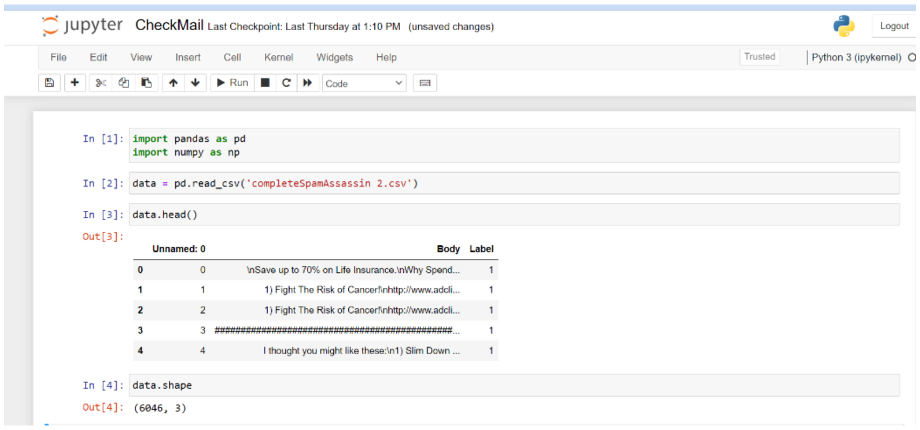

kaggle.com/datasets/nitishabharathi/email-spam-dataset
Email Spam Dataset
LingSpam, EnronSpam, Spam Assassin Dataset containing ham and spam email
Data Card
Code (15)
Discussion (0)
About Dataset
Dataset for spam email filtering
Usability
License
CC0: Public Domain
Expected update frequency
Annually
completeSpamAssassin.csv (10.83 MB)
Data Explorer
completeSpamAssassin.csv
enronSpamSubset.csv
lingSpam.csv

Fig. 2: Dataset that we used in the portal from Kaggle.

The dataset that we used for our model was already labeled as 0 for decent messages and 1 for spam messages. This made our preprocessing easier and faster as we could skip this part which I will be explaining in the following sections.

4.2. Understanding The Data

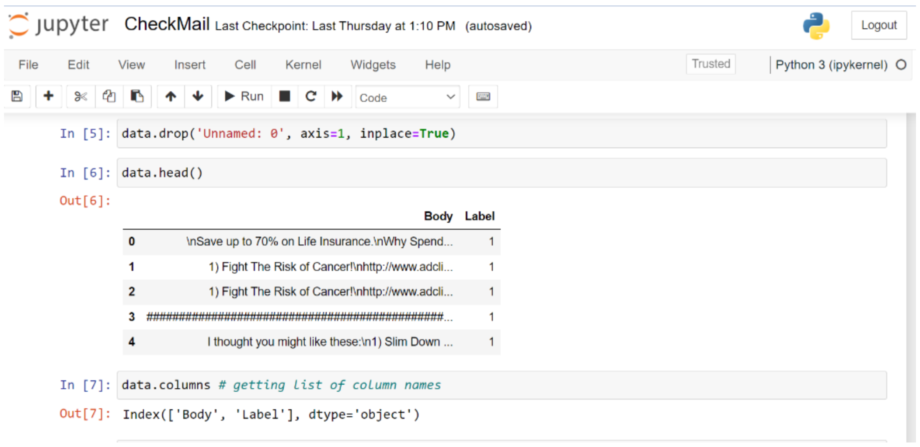
The next important thing to do before preprocessing the data would be to understand the data properly.



```
In [1]: import pandas as pd
        import numpy as np

In [2]: data = pd.read_csv('completeSpamAssassin 2.csv')

In [3]: data.head()

Out[3]:
     Unnamed: 0                                                Body  Label
  0           0      \nSave up to 70% on Life Insurance.\nWhy Spend...      1
  1           1           1) Fight The Risk of Cancer!\nhttp://www.adcli...      1
  2           2           1) Fight The Risk of Cancer!\nhttp://www.adcli...      1
  3           3  ##################################################...      1
  4           4          I thought you might like these:\n1) Slim Down ...      1

In [4]: data.shape

Out[4]: (6046, 3)
```

Fig.3: Data importing and reading the data.

And to do so, we have used Pandas [19] library form python. We used pandas to import and manipulate data as per our need. There were basically columns 'Unnamed: 0', 'Body' and 'Label' and 6046 rows as shown in Fig. 3 below. There were 6046 different emails (messages) that we were going to preprocess and feed into the model for training and testing purposes for our model. The column 'Body' includes all the messages in the emails and the column 'Label' includes the label of these email bodies as 0 if the email is not a spam and 1 if the email is a spam email indeed. Out of these three columns, the column 'Unnamed: 0' was the one we did not need. So, we removed the column first by using the '.drop()' function of pandas library [18]. After which the data looked like the one in Fig. 4.



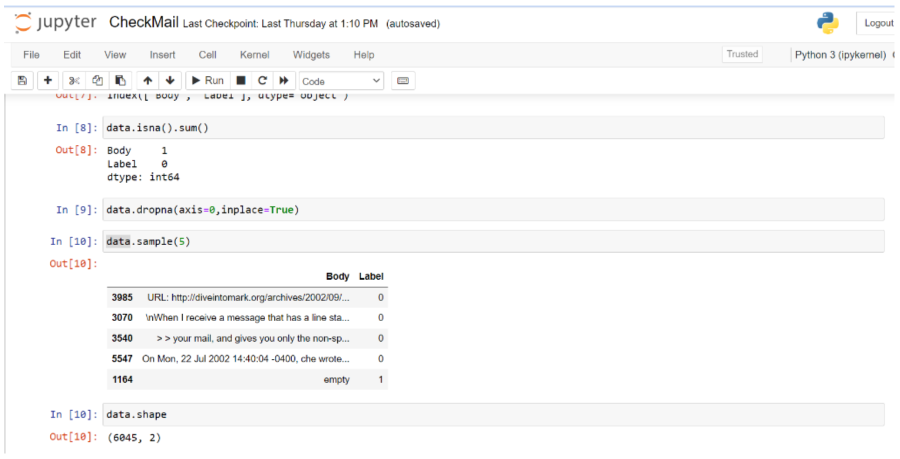
```
In [5]: data.drop('Unnamed: 0', axis=1, inplace=True)

In [6]: data.head()

Out[6]:
```

	Body	Label
0	\nSave up to 70% on Life Insurance.\nWhy Spend...	1
1	1) Fight The Risk of Cancer!\nhttp://www.adcli...	1
2	1) Fight The Risk of Cancer!\nhttp://www.adcli...	1
3	##...	1
4	I thought you might like these:\n1) Slim Down ...	1

```
In [7]: data.columns # getting list of column names

Out[7]: Index(['Body', 'Label'], dtype='object')
```

Fig. 4: Removing the unwanted column from the data.

The next thing we did was to findout if there were any NaN values in the data. NaN values are those strings (words) that pandas considers as Not a Number. We did this process by '.isna()' function and found out that there was 1 NaN value in the Body so we removed it using the '.dropna()' fucntion. So, now there were 6054 messages left in our data.



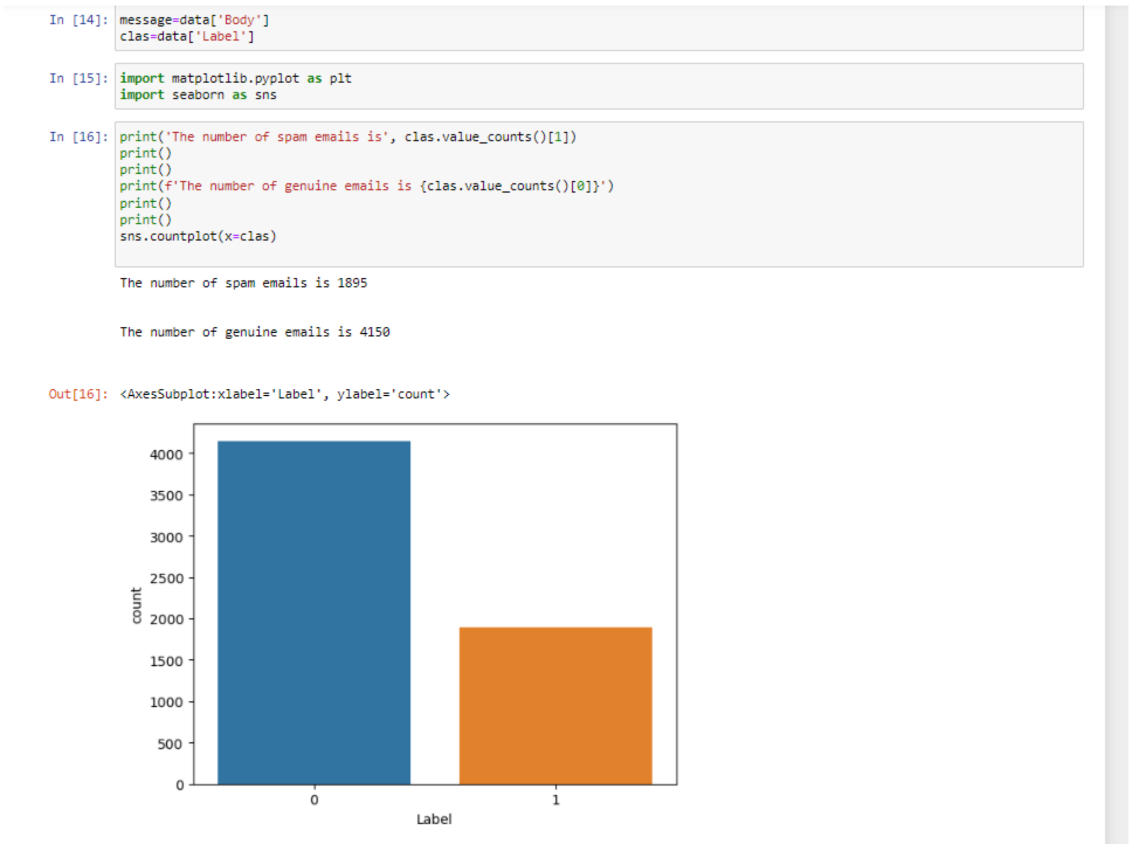
```
In [8]: data.isna().sum()

Out[8]: Body     1
        Label    0
        dtype: int64

In [9]: data.dropna(axis=0,inplace=True)

In [10]: data.sample(5)

Out[10]:
```

	Body	Label
3985	URL: http://diveintomark.org/archives/2002/09/...	0
3070	\nWhen I receive a message that has a line sta...	0
3540	> > your mail, and gives you only the non-sp...	0
5547	On Mon, 22 Jul 2002 14:40:04 -0400, che wrote...	0
1164	empty	1

```
In [10]: data.shape

Out[10]: (6045, 2)
```

Fig. 5: Finding and removing NaN values in the data.

And to visualize the data we used Seabron [20] library form the Python programming language. And we found out using seabron that, there were 1895 spam messages and 4150 messages which were not spams. We alson renamed the Body of our data to 'message' and Label to 'clas' using pandas so that it will be easy for us to understand proceed further with our data.



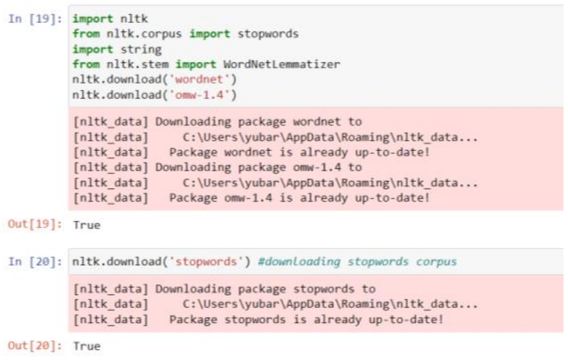
```
In [14]: message=data['Body']
         clas=data['Label']

In [15]: import matplotlib.pyplot as plt
         import seaborn as sns

In [16]: print('The number of spam emails is', clas.value_counts()[1])
         print()
         print()
         print(f'The number of genuine emails is {clas.value_counts()[0]}')
         print()
         print()
         sns.countplot(x=clas)

         The number of spam emails is 1895


         The number of genuine emails is 4150


Out[16]: <AxesSubplot:xlabel='Label', ylabel='count'>
```

Fig. 6: Visualization of data and renaming the columns.

4.3. Preprocessing The Data

The concept of converting the raw data into clean data by removing unwanted bodies and expressing the data into computational language so that our model can understand what we are feeding to it is known as data preprocessing [21]. The reason that the dataset should be preprocessed is to check if there are any missing values, or noisy data, and any other inconsistencies before feeding it to the algorithm of the model. This process helps in improving the accuracy and quality of the dataset so that it becomes more reliable.

There are various preprocessing tools and techniques available. However, in our preprocessing, we used Natural Language Toolkit to do so. First and foremost, the thing that we did in the stage was to import or download the necessary modules for our preprocessing. We used modules like 'stopwords',



```
In [19]: import nltk
         from nltk.corpus import stopwords
         import string
         from nltk.stem import WordNetLemmatizer
         nltk.download('wordnet')
         nltk.download('omw-1.4')

         [nltk_data] Downloading package wordnet to
         [nltk_data]     C:\Users\yubar\AppData\Roaming\nltk_data...
         [nltk_data]   Package wordnet is already up-to-date!
         [nltk_data] Downloading package omw-1.4 to
         [nltk_data]     C:\Users\yubar\AppData\Roaming\nltk_data...
         [nltk_data]   Package omw-1.4 is already up-to-date!

Out[19]: True

In [20]: nltk.download('stopwords') #downloading stopwords corpus

         [nltk_data] Downloading package stopwords to
         [nltk_data]     C:\Users\yubar\AppData\Roaming\nltk_data...
         [nltk_data]   Package stopwords is already up-to-date!

Out[20]: True
```

Fig. 7: Downloading necessary modules for preprocessing.

‘WordNetLemmatizer’ etc. for the preprocessing that we were going to carry out in the following steps.

To preprocess our data, we defined a function and named it ‘process_email’. There were three main purposes of this function.

1. Remove Punctuations:

The first purpose of this function was to remove any kind of punctuations present in our dataset. For this, we created an empty list ‘nopunc’ and kept adding all the words in the data to this list leaving out all the punctuation until there was no punctuation left.

2. Remove Stop Words:

Secondly, our function ‘process_email’ removed all the stopwords from the data. The words that are most commonly and frequently used in a language are known as stop words. Words like “the”, “are”, “a”, ‘is”, etc. can be considered as stop words in English language. This kind of concept of stop words is commonly used in NLP to eliminate the words that are used widely even though they carry very little information [22].

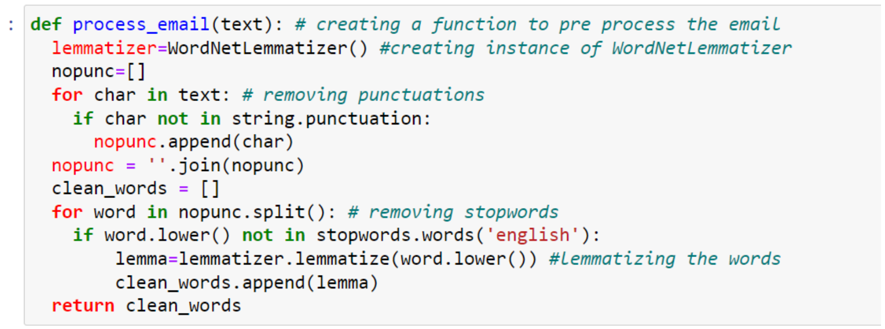
In our algorithm, we passed the new appended list ‘nopunc’ through another for loop where we looked for the words those were not stop words and sent those words to the lemmatizer for lemmatizing those words. Then we carried out the lemmatizing process which was the third and the last purpose of our function ‘process_email’.

3. Lemmatize the Words:

After removing the punctuation and stop words, we lemmatized or data. Lemmatization is the algorithmic process of identifying a word's “lemma” (dictionary form) on the basis of its intended meaning. Lemmatization deals with reducing the word or the search query to its canonical dictionary form. The root word is called a 'lemma' and the method is called lemmatization. This approach takes a part of the word into consideration in a way that it is recognized as a single element [23].

For the lemmatization, we passed every word that came from step two of our function “process_email” to the lemmatizer that we created using Word Net Lemmatizer of NLTK. The words then were appended to the list clean_words (which was initially an empty list that we created) and the list would be returned at the end by our function.

Thus, in these three simple steps, we completed our preprocessing of the data. These steps were carried out by the codes as shown in Fig.8 below.



```
: def process_email(text): # creating a function to pre process the email
    lemmatizer=WordNetLemmatizer() #creating instance of WordNetLemmatizer
    nopunc=[]
    for char in text: # removing punctuations
      if char not in string.punctuation:
        nopunc.append(char)
    nopunc = ''.join(nopunc)
    clean_words = []
    for word in nopunc.split(): # removing stopwords
      if word.lower() not in stopwords.words('english'):
          lemma=lemmatizer.lemmatize(word.lower()) #Lemmatizing the words
          clean_words.append(lemma)
    return clean_words
```

Fig. 8: Preprocessing of the data.

Additionally in the preprocessing, we also carried out vectorization. For this purpose, we used Count Vectorizer from scikit-learn library [24]. Vectorization, also known as feature extraction, is a process of encoding the words or the textual data as integers, or floating-point values in order to use those textual data as an input so that our model understands it.

So, we used Count Vectorizer from scikit-learn to convert our textual emails to a vector of token/term count. The reason why we selected this vectorizer is because it enables the pre-processing of textual data before even generating the vector representation thus making it a highly flexible feature extraction module. The scikit-learn Count Vectorizer reads each sentence and the frequency of data and assigns float values to each words according to their frequencies as follows:

Data = “Hello Udaya! the NASA wants you to go to the moon.”

$\downarrow$ preprocessing

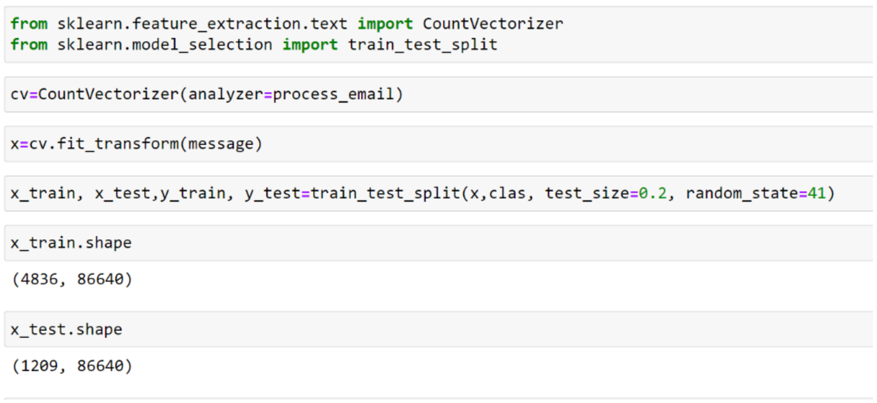
Data = [‘hello’, ‘udaya’, ‘the’, ‘nasa’, ‘wants’, ‘you’, ‘to’, ‘go’, ‘to’, ‘the’, ‘moon’]

$\downarrow$ count vectorization

Data =

hello	udaya	the	nasa	wants	you	to	go	moon
1	1	2	1	1	1	2	1	1

After the vectorization process, we split our data into training and testing bits using scikit-learn train_test_split module in the ration 4:1 [25]. This process of vectorization and splitting was done using the codes provided below:



```
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.model_selection import train_test_split
```

```
cv=CountVectorizer(analyzer=process_email)
```

```
x=cv.fit_transform(message)
```

```
x_train, x_test,y_train, y_test=train_test_split(x,clas, test_size=0.2, random_state=41)
```

```
x_train.shape
```

```
(4836, 86640)
```

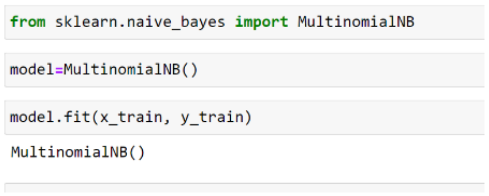
```
x_test.shape
```

```
(1209, 86640)
```

Fig. 9: vectorization and splitting of data.

4.4. Making Classifier Model and Assessing It

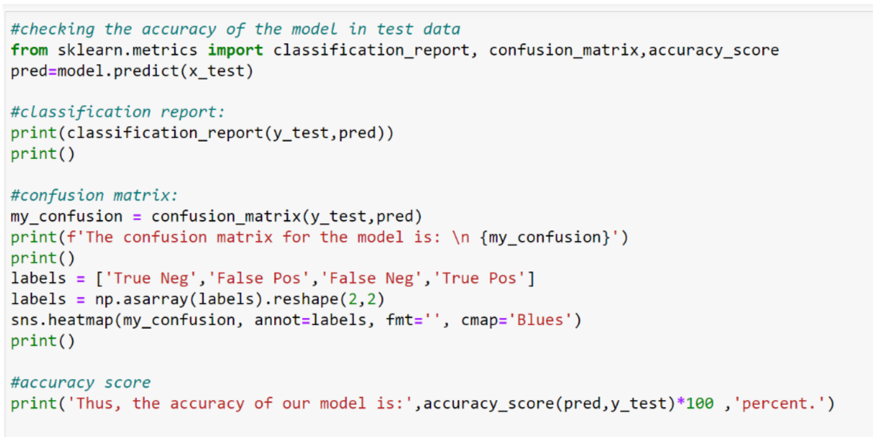
After the preprocessing of the data, we developed our classifier that could classify our input. For the classifier we used Multinomial Naïve Bayes [18]. We named our classifier ‘model’ and trained it using the preprocessed data and normalizing the data by fitting and transforming it. This process is done using following codes:



```
from sklearn.naive_bayes import MultinomialNB

model=MultinomialNB()

model.fit(x_train, y_train)
MultinomialNB()
```

Fig.10: Making instance and training of model.

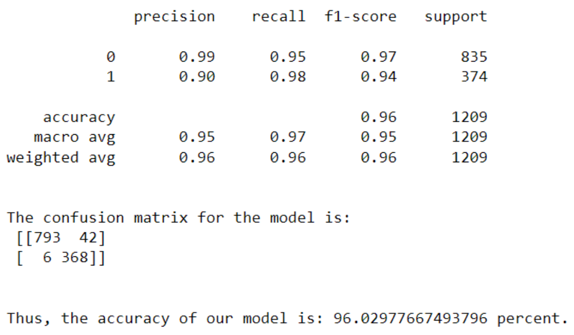
After making instance and training the model, we assessed our model using different metrics like classification report, accuracy score and confusion matrix [28]-[29] from scikit-learn [24]. In this process, we let our model make some predictions and compared these predictions with the actual statistics by using the following codes:



```
#checking the accuracy of the model in test data
from sklearn.metrics import classification_report, confusion_matrix,accuracy_score
pred=model.predict(x_test)

#classification report:
print(classification_report(y_test,pred))
print()

#confusion matrix:
my_confusion = confusion_matrix(y_test,pred)
print(f'The confusion matrix for the model is: \n {my_confusion}')
print()
labels = ['True Neg','False Pos','False Neg','True Pos']
labels = np.asarray(labels).reshape(2,2)
sns.heatmap(my_confusion, annot=labels, fmt='', cmap='Blues')
print()

#accuracy score
print('Thus, the accuracy of our model is:',accuracy_score(pred,y_test)*100 ,'percent.')
```

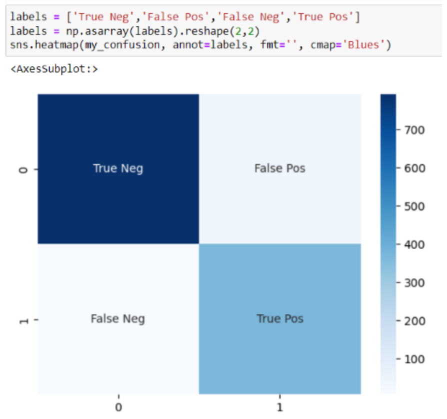
Fig. 11: Assessing the model that we built.

And we achieved the following results:



```
              precision    recall  f1-score   support

           0       0.99      0.95      0.97       835
           1       0.90      0.98      0.94       374

    accuracy                           0.96      1209
   macro avg       0.95      0.97      0.95      1209
weighted avg       0.96      0.96      0.96      1209


The confusion matrix for the model is:
 [[793  42]
 [  6 368]]


Thus, the accuracy of our model is: 96.02977667493796 percent.
```

Fig. 12: Assessment results.


```
labels = ['True Neg','False Pos','False Neg','True Pos']
labels = np.asarray(labels).reshape(2,2)
sns.heatmap(my_confusion, annot=labels, fmt='', cmap='Blues')
```

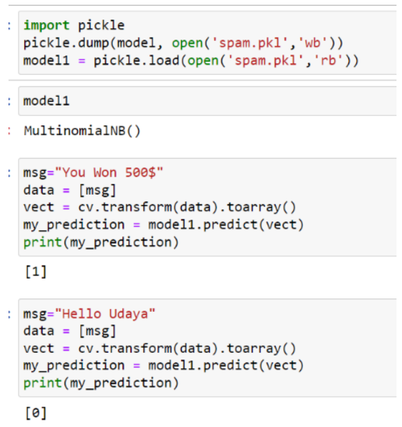
```
<AxesSubplot:>
```

Fig. 13: Confusion matrix in heatmap [28]-[29].

According to the results of the assessments that we did using different metrics from scikit-learn, our model performed about 96% of the time accurately. In this 96% of accuracy, our model mostly predicted the email those were not spam correctly and then second most correct predictions made by our model were for spam emails.

4.5 Making Graphical User Interface

In making the Graphical User Interface (GUI) for our classifier, we first used pickle [25] module to store our serialized model under the name 'spam.pkl' and then we called it back and saved it under the name 'model1' and tested if the model1 is same as our model by using the following codes:


```
import pickle
pickle.dump(model, open('spam.pkl','wb'))
model1 = pickle.load(open('spam.pkl','rb'))
```

```
model1
```

```
MultinomialNB()
```

```
msg="You Won 500$"
data = [msg]
vect = cv.transform(data).toarray()
my_prediction = model1.predict(vect)
print(my_prediction)
```

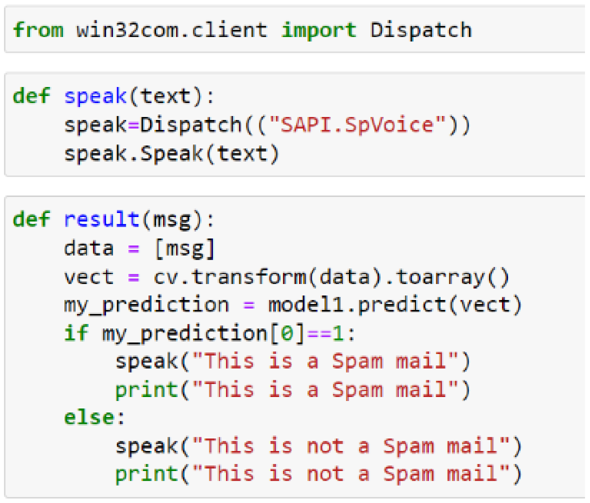
```
[1]
```

```
msg="Hello Udaya"
data = [msg]
vect = cv.transform(data).toarray()
my_prediction = model1.predict(vect)
print(my_prediction)
```

```
[0]
```

Fig. 14: Using pickle for the serialization of our classifier.

Since we designed our GUI to be able to not only show the classification but also speak it out, we used win32com.client [26] module. And to design the overall GUI, we used tkinter [27] module. In this stage, we defined a function that we named 'result' whose job was to display and speak whatever prediction was given by our model1. Which was done using the codes given in Fig. 15.



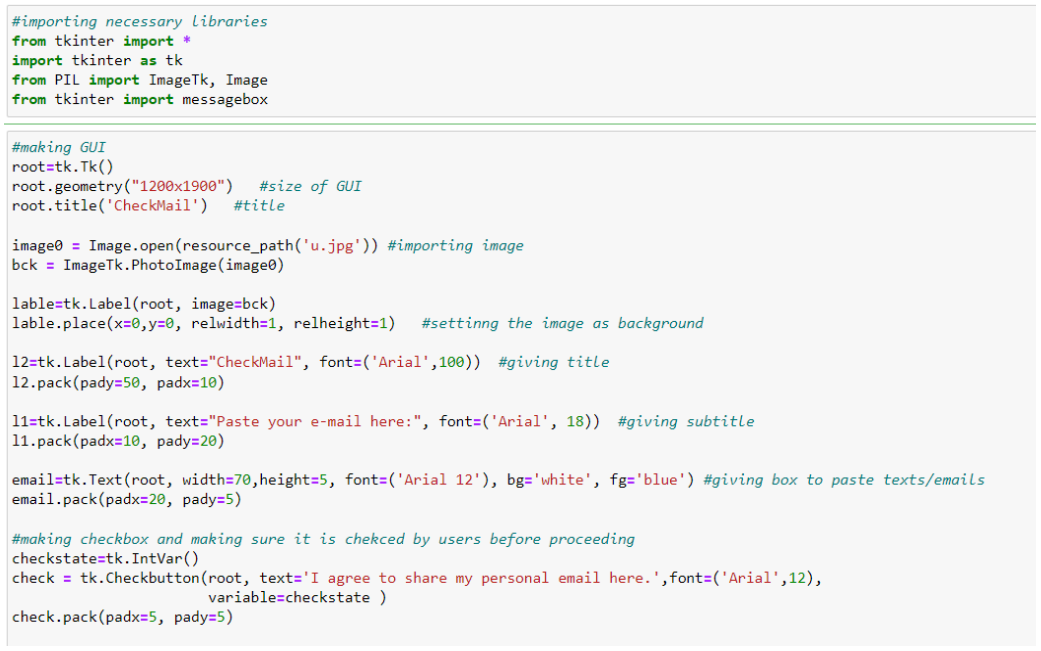
```python
from win32com.client import Dispatch
```

```python
def speak(text):
    speak=Dispatch(("SAPI.SpVoice"))
    speak.Speak(text)
```

```python
def result(msg):
    data = [msg]
    vect = cv.transform(data).toarray()
    my_prediction = model1.predict(vect)
    if my_prediction[0]==1:
        speak("This is a Spam mail")
        print("This is a Spam mail")
    else:
        speak("This is not a Spam mail")
        print("This is not a Spam mail")
```

Fig. 15: Defining functions for giving out predictions made by our classifier.

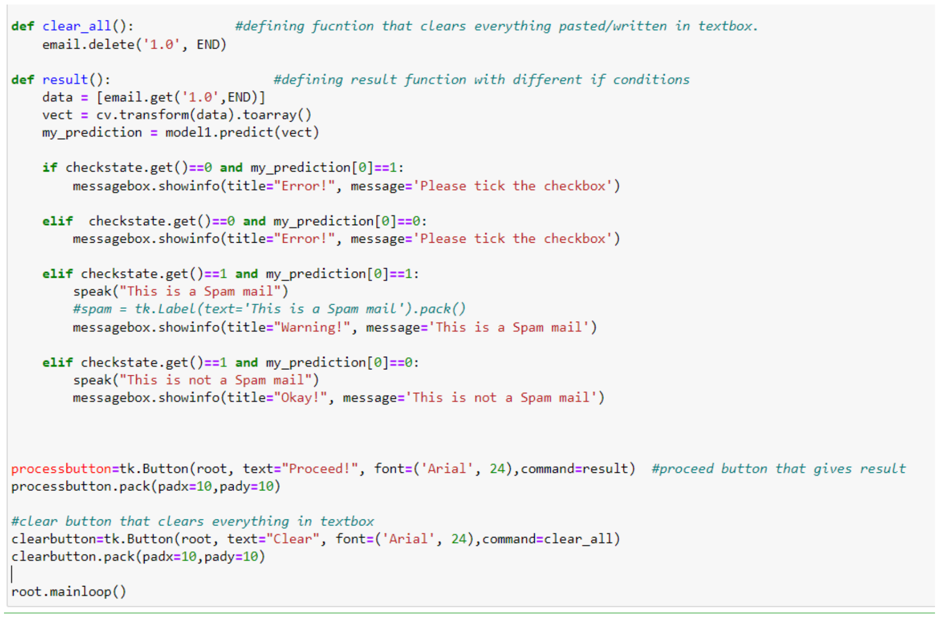
After that we designed the interface for our GUI using different functions such as message box, button, check button, etc. of tkinter. Please refer to [27] for the detailed study of tkinter and its functions and modules. We used the following codes shown in Fig. 16 (A & B) to develop our GUI completely:



```python
#importing necessary libraries
from tkinter import *
import tkinter as tk
from PIL import ImageTk, Image
from tkinter import messagebox
```

```python
#making GUI
root=tk.Tk()
root.geometry("1200x1900")   #size of GUI
root.title('CheckMail')   #title

image0 = Image.open(resource_path('u.jpg')) #importing image
bck = ImageTk.PhotoImage(image0)

lable=tk.Label(root, image=bck)
lable.place(x=0,y=0, relwidth=1, relheight=1)   #settinng the image as background

l2=tk.Label(root, text="CheckMail", font=('Arial',100))  #giving title
l2.pack(pady=50, padx=10)

l1=tk.Label(root, text="Paste your e-mail here:", font=('Arial', 18))  #giving subtitle
l1.pack(padx=10, pady=20)

email=tk.Text(root, width=70,height=5, font=('Arial 12'), bg='white', fg='blue') #giving box to paste texts/emails
email.pack(padx=20, pady=5)

#making checkbox and making sure it is chekced by users before proceeding
checkstate=tk.IntVar()
check = tk.Checkbutton(root, text='I agree to share my personal email here.',font=('Arial',12),
                       variable=checkstate )
check.pack(padx=5, pady=5)
```

Fig. 16 (A): Codes at the backend of our GUI.



```
def clear_all():            #defining fucntion that clears everything pasted/written in textbox.
    email.delete('1.0', END)

def result():                  #defining result function with different if conditions
    data = [email.get('1.0',END)]
    vect = cv.transform(data).toarray()
    my_prediction = model1.predict(vect)

    if checkstate.get()==0 and my_prediction[0]==1:
        messagebox.showinfo(title="Error!", message='Please tick the checkbox')

    elif  checkstate.get()==0 and my_prediction[0]==0:
        messagebox.showinfo(title="Error!", message='Please tick the checkbox')

    elif checkstate.get()==1 and my_prediction[0]==1:
        speak("This is a Spam mail")
        #spam = tk.Label(text='This is a Spam mail').pack()
        messagebox.showinfo(title="Warning!", message='This is a Spam mail')

    elif checkstate.get()==1 and my_prediction[0]==0:
        speak("This is not a Spam mail")
        messagebox.showinfo(title="Okay!", message='This is not a Spam mail')


processbutton=tk.Button(root, text="Proceed!", font=('Arial', 24),command=result)  #proceed button that gives result
processbutton.pack(padx=10,pady=10)

#clear button that clears everything in textbox
clearbutton=tk.Button(root, text="Clear", font=('Arial', 24),command=clear_all)
clearbutton.pack(padx=10,pady=10)

root.mainloop()
```

Fig. 16 (B): Codes at the backend of our GUI.

The result of this last bit is the GUI that we built which classifies any given email/text messages into a spam or not a spam category and saves the user from such emails and the risks that these spam emails carry. The front end of our GUI is shown in Fig.17 below.

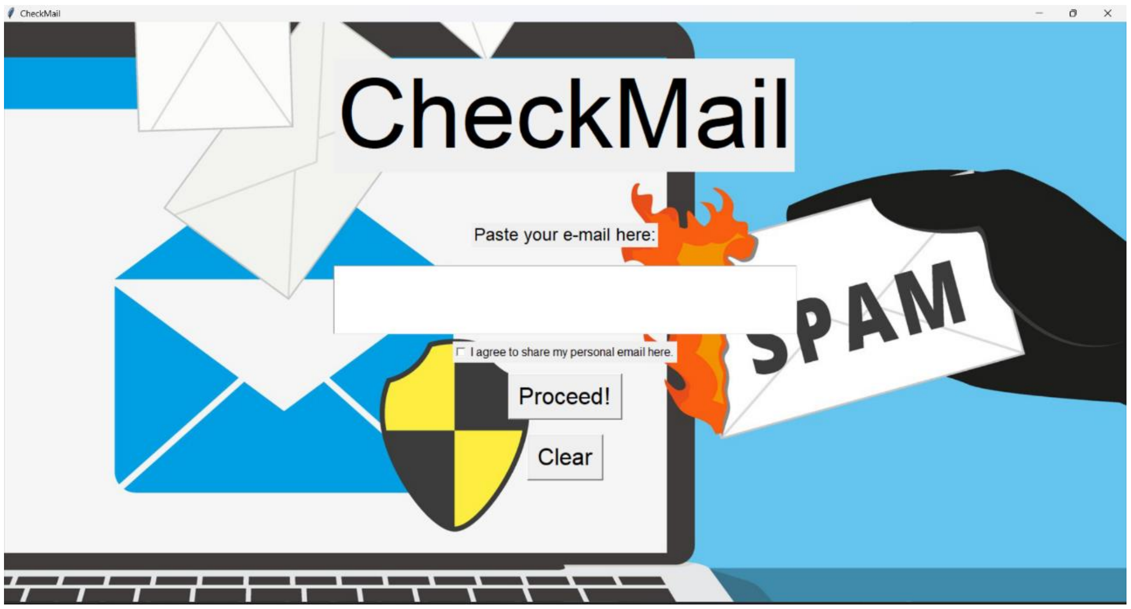
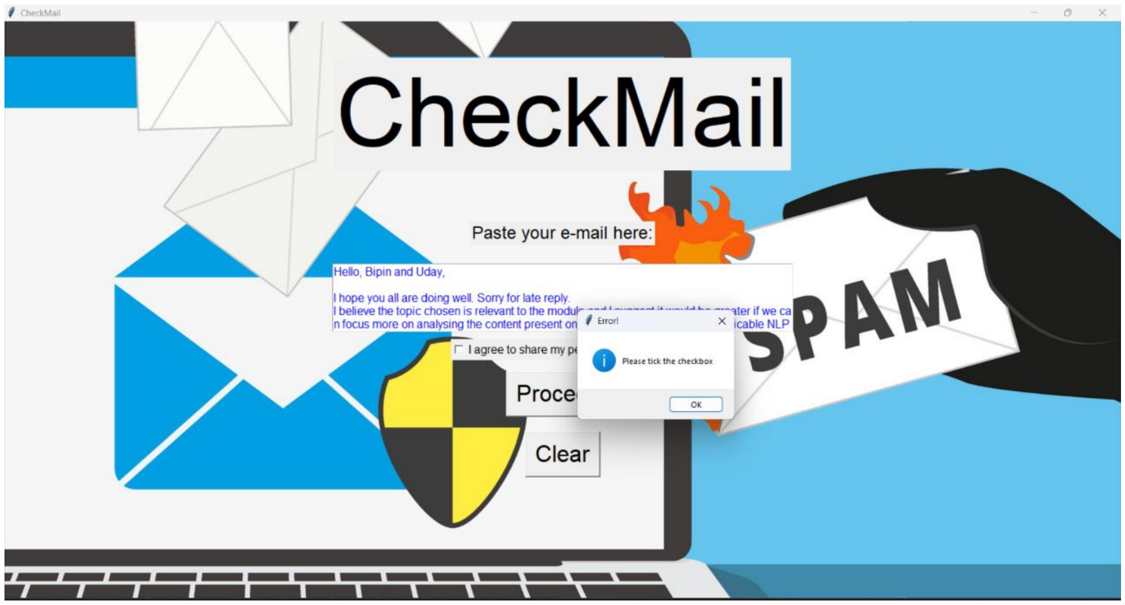


Fig. 17: Interface of CheckMail (our spam email detecting GUI).

5. Real-Time Application and Analysis of Our Classifier

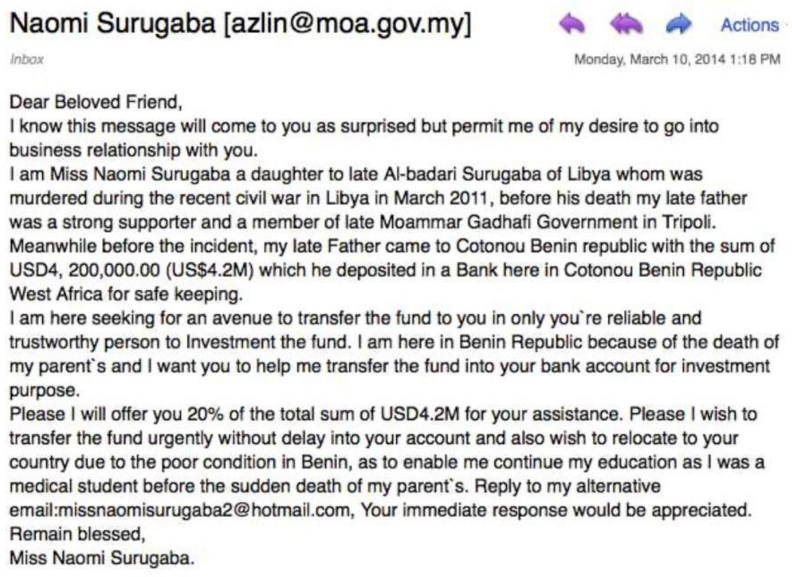
After completing the work at the backend, we tested our classifier with different test datasets with different variations and the results were as follows:



Hello, Bipin and Uday,

I hope you all are doing well. Sorry for late reply.
I believe the topic chosen is relevant to the modul   ater if we ca
n focus more on analysing the content present on   icable NLP

Fig. 18: Message shown by GUI if no consent is given by user for data sharing.



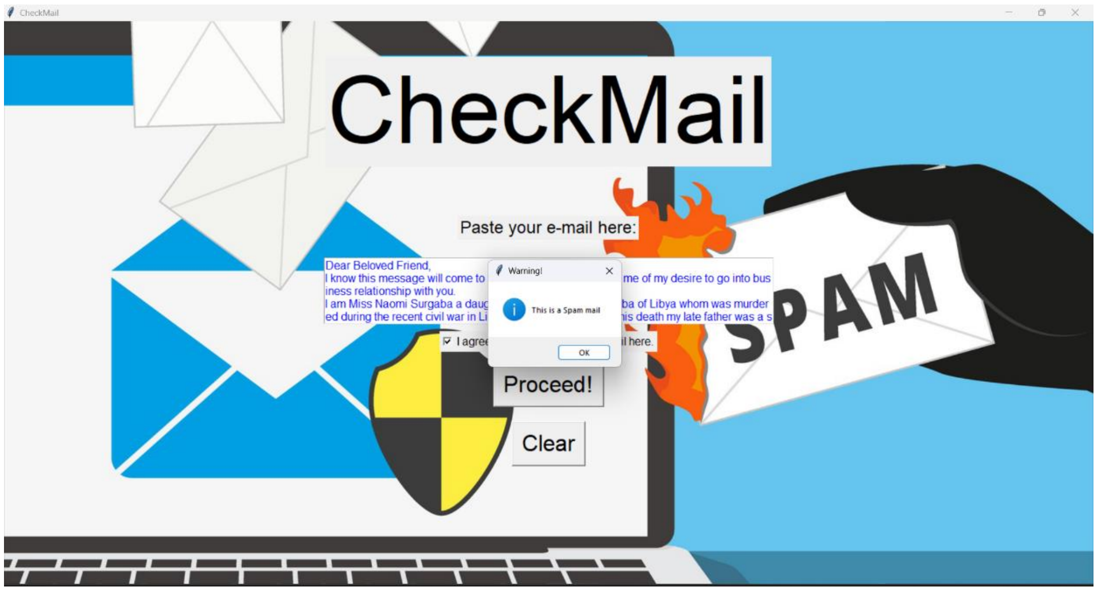
Naomi Surugaba [azlin@moa.gov.my]

Inbox

Monday, March 10, 2014 1:18 PM

Dear Beloved Friend,
I know this message will come to you as surprised but permit me of my desire to go into business relationship with you.
I am Miss Naomi Surugaba a daughter to late Al-badari Surugaba of Libya whom was murdered during the recent civil war in Libya in March 2011, before his death my late father was a strong supporter and a member of late Moammar Gadhafi Government in Tripoli.
Meanwhile before the incident, my late Father came to Cotonou Benin republic with the sum of USD4, 200,000.00 (US$4.2M) which he deposited in a Bank here in Cotonou Benin Republic West Africa for safe keeping.
I am here seeking for an avenue to transfer the fund to you in only you`re reliable and trustworthy person to Investment the fund. I am here in Benin Republic because of the death of my parent`s and I want you to help me transfer the fund into your bank account for investment purpose.
Please I will offer you 20% of the total sum of USD4.2M for your assistance. Please I wish to transfer the fund urgently without delay into your account and also wish to relocate to your country due to the poor condition in Benin, as to enable me continue my education as I was a medical student before the sudden death of my parent`s. Reply to my alternative email:missnaomisurugaba2@hotmail.com, Your immediate response would be appreciated.
Remain blessed,
Miss Naomi Surugaba.

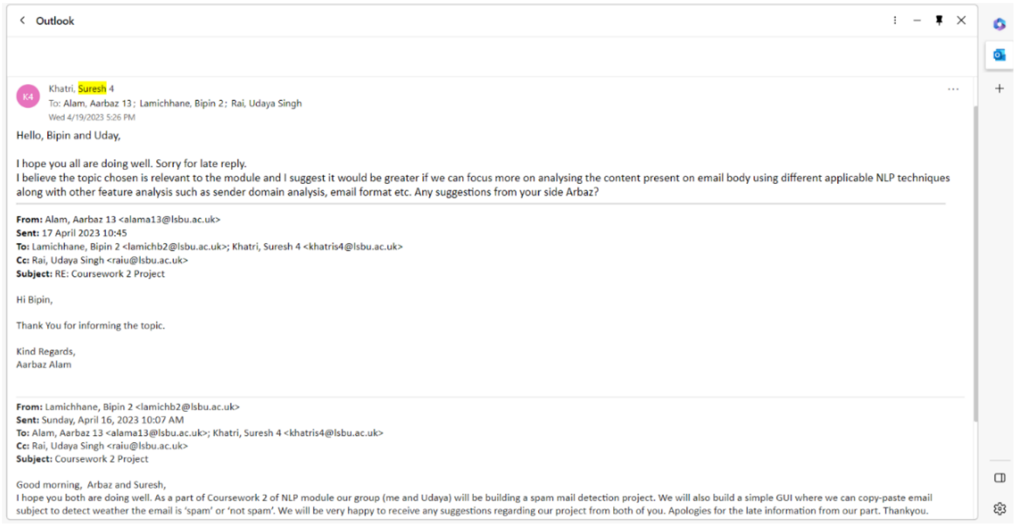
Fig. 19: Sample spam email.



Dear Beloved Friend,
I know this message will come to me of my desire to go into bus
iness relationship with you.
I am Miss Naomi Surgaba a daug ba of Libya whom was murder
ed during the recent civil war in Li his death my late father was a s

This is a Spam mail

Fig. 20: Prediction given by our model on the Fig.19 email.



Khatri, Suresh 4
To: Alam, Aarbaz 13; Lamichhane, Bipin 2; Rai, Udaya Singh
Wed 4/19/2023 5:26 PM

Hello, Bipin and Uday,

I hope you all are doing well. Sorry for late reply.
I believe the topic chosen is relevant to the module and I suggest it would be greater if we can focus more on analysing the content present on email body using different applicable NLP techniques along with other feature analysis such as sender domain analysis, email format etc. Any suggestions from your side Arbaz?

From: Alam, Aarbaz 13 <alama13@lsbu.ac.uk>
Sent: 17 April 2023 10:45
To: Lamichhane, Bipin 2 <lamichb2@lsbu.ac.uk>; Khatri, Suresh 4 <khatris4@lsbu.ac.uk>
Cc: Rai, Udaya Singh <raiu@lsbu.ac.uk>
Subject: RE: Coursework 2 Project

Hi Bipin,

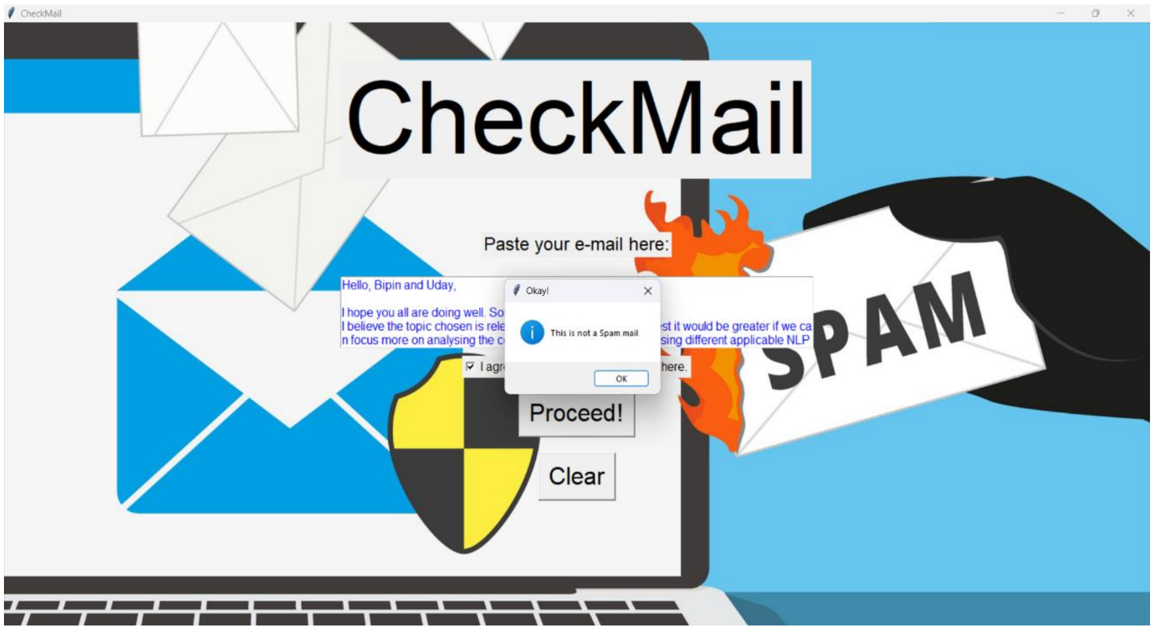
Thank You for informing the topic.

Kind Regards,
Aarbaz Alam

From: Lamichhane, Bipin 2 <lamichb2@lsbu.ac.uk>
Sent: Sunday, April 16, 2023 10:07 AM
To: Alam, Aarbaz 13 <alama13@lsbu.ac.uk>; Khatri, Suresh 4 <khatris4@lsbu.ac.uk>
Cc: Rai, Udaya Singh <raiu@lsbu.ac.uk>
Subject: Coursework 2 Project

Good morning, Arbaz and Suresh,
I hope you both are doing well. As a part of Coursework 2 of NLP module our group (me and Udaya) will be building a spam mail detection project. We will also build a simple GUI where we can copy-paste email subject to detect weather the email is 'spam' or 'not spam'. We will be very happy to receive any suggestions regarding our project from both of you. Apologies for the late information from our part. Thankyou.

Fig. 21: Sample of a regular (not spam / ham) email.



Hello, Bipin and Uday,

I hope you all are doing well. So
I believe the topic chosen is rele
n focus more on analysing the c

st it would be greater if we ca
sing different applicable NLP

This is not a Spam mail

Fig.22: Prediction given by our model on the Fig.21 email.

We had tried with different types of emails (both spam and ham) and the result that our model gave was correct almost all the time. As mentioned above, the accuracy that our model delivered was about 96% (refer to Fig. 12). Also, according to the confusion matrix [28]-[29], our model mostly predicted emails that were not spam correctly. The accuracy of prediction of emails/messages that were spam was second highest (refer to Fig.13). Also, because of the speaking ability of our model, it was accessible to people with different abilities too.

5.1. General Flowchart

In general, when we give contents of the email to our model in raw format, it preprocesses the email by removing punctuation, stop words and then lemmatizing the texts. Then it vectorizes the textual data using Count Vectorizer of scikit-learn and applies Multinomial Naïve Bayesian classifier to classify the data. In this process, it looks for the keywords which represent spam emails in the data that we fed to it. If the keyword is present, it classifies the email as 'spam' and 'not spam' otherwise. The basic flowchart of our classifier is demonstrated by Fig.22 below.

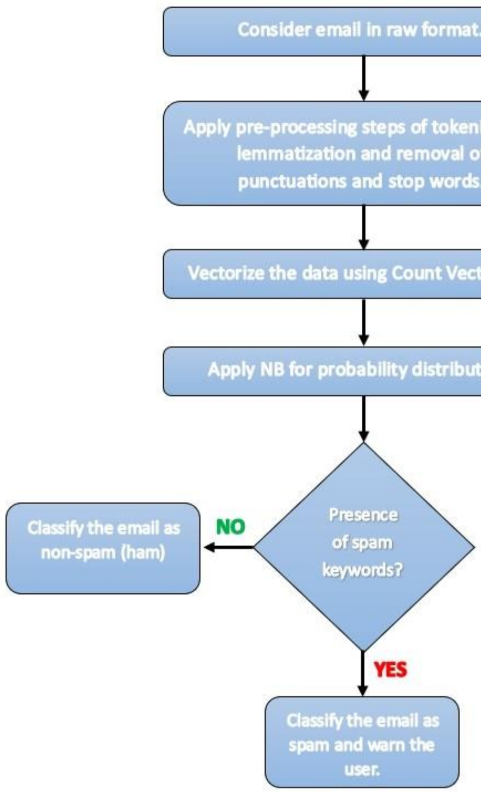

Consider email in raw format.
Apply pre-processing steps of tokenization, lemmatization and removal of punctuations and stop words.
Vectorize the data using Count Vectorizer.
Apply NB for probability distribution.
Presence of spam keywords?
NO
Classify the email as non-spam (ham)
YES
Classify the email as spam and warn the user.

Fig. 22: Basic flowchart of our model.

6. Challenges Faced and Troubleshooting

The first challenge that we faced during this project is the finding of correct dataset. We tried different datasets and only after a few hours of research we were able to find the dataset that we have used and mentioned above. Secondly, to find the desired accuracy while training and testing the dataset was another complication that we overcame by simply hit and trial. Moreover, because of the accuracy that our model was displaying, we assumed it might have been a case of overfitting. However, when we tested our model in real-world and with some more test data sets, the model performed accurately. And mainly, when we tried to convert our Python file to executable (.exe) file, we faced many errors and even after overcoming the errors the file never ran as an executable. We are still trying to troubleshoot this problem and planning to make it more advanced and automatic additionally.

7. Team Contribution

We had 2 members including me (Udaya Singh Rai) and Bipin Lamichhane in our group for this project. And rather than dividing the working area, we decided to work together on everything from scratch so that at the end of the project we both would have learnt and understood the model that we were making completely. So, we both contributed equally to generating ideas, doing research for data collection and theoretical aspects as well. However, the coding for the model development including data preprocessing and naïve bayes was done by Bipin Lamichhane.

The code was then handed over to me for further coding to develop our GUI model. Even though I did all the coding for the GUI and gave a basic layout of the interface, the idea for the graphical design of the interface was mutual.

We both sat down to troubleshoot the problems that we faced and agreed on the parameters and were able to hand out the final project as explained in previous sections. And during this entire process, we had been in contact discussing ideas and trying things all the time while working on this project. So, we made equal contributions in developing, testing, and troubleshooting for our model to its entirety.

8. Conclusion

Natural Language Processing has always been one of the most popular topics of discussion in the field of AI. And in this report, we discussed some of the modules that NLP is based on. We discussed how we can train our machines to learn natural language and implement binary classifications for human use which would take longer and more effort if had to be done by humans manually. This paper also gave a basic idea of how a classifier works and where we can apply them.

As mentioned above, when we tested our model in real-world scenario, the performance it showed was quite satisfying. However, we will try to make the model automatic by using the Cloud services that are available. We will also be working on this project to make it an executable file which will not need any human intervention in the coming days.

Reference

[1] P. M. Nadkarni, L. Ohno-Machado, and W. W. Chapman, "Natural language processing: an introduction," vol. 18, no. 5, pp. 544–551, 2011.

[2] C. D. Manning, *An introduction to information retrieval*. Cambridge university press, 2009.

[3] S. Ruder, "A review of the neural history of natural language processing," vol. 1, 2018.

[4] O. Saad, A. Darwish, and R. Faraj, "A survey of machine learning techniques for Spam filtering," vol. 12, no. 2, p. 66, 2012.

[5] M. K. Paswan, P. S. Bala, and G. Aghila, "Spam filtering: Comparative analysis of filtering techniques," 2012, pp. 170–176.

[6] A. Karim, S. Azam, B. Shanmugam, K. Kannoorpatti, and M. Alazab, "A comprehensive survey for intelligent spam email detection," vol. 7, pp. 168261–168295, 2019.

[7] E. Bauer, 15 Outrageous Email Spam Statistics that Still Ring True in 2018, Jul. 2019, [online] Available: https://www.propellercrm.com/blog/email-spam-statistics.

[8] J. Clement, "Number of e-mail users worldwide from 2017 to 2023," 2019.

[9] C. Monitor, "The shocking truth about how many emails are sent," 2019.

[10] R. Al Halaseh and J. Alqatawna, "Analyzing cybercrimes strategies: The case of phishing attack," 2016, pp. 82–88.

[11] S. Smadi, N. Aslam, and L. Zhang, "Detection of online phishing email using dynamic evolving neural network based on reinforcement learning," vol. 107, pp. 88–102, 2018.

[12] M. N. Banu and S. M. Banu, "A comprehensive study of phishing attacks," vol. 4, no. 6, pp. 783– 786, 2013.

[13] H. Hu and G. Wang, “Revisiting email spoofing attacks,” 2018.

[14] A. Attar, R. M. Rad, and R. E. Atani, “A survey of image spamming and filtering techniques,” vol. 40, pp. 71–105, 2013.

[15] C. Leung, “An analysis of the impact of phishing and anti-phishing related announcements on market value of global firms,” 2009.

[16] P. Domingos and M. Pazzani, “On the optimality of the simple Bayesian classifier under zero-one loss,” vol. 29, pp. 103–130, 1997.

[17] J. L. Hellerstein, T. S. Jayram, and I. Rish, *Recognizing end-user transactions in performance management*. IBM Thomas J. Watson Research Division Hawthorne, NY, 2000.

[18] R. Sunil, “Naïve Bayes Classifier Explained: Applications and Practice Problems of Naïve Bayes Classifier”, Analytics Vidhay, Available: https://www.analyticsvidhya.com/blog/2017/09/naive-bayes-explained/

[19] W. McKinney, “pandas: a foundational Python library for data analysis and statistics,” vol. 14, no. 9, pp. 1–9, 2011.

[20] M. L. Waskom, “Seaborn: statistical data visualization,” vol. 6, no. 60, p. 3021, 2021.

[21] K. Al-Jabery, T. Obafemi-Ajayi, G. Olbricht, and D. Wunsch, *Computational learning approaches to data analytics in biomedical applications*. Academic Press, 2019.

[22] K. Ganesan, “What Are Stop Words?”, Opinosis Analytics, Available at: https://www.opinosis-analytics.com/knowledge-base/stop-words-explained/.

[23] D. Khyani, B. S. Siddhartha, N. M. Niveditha, and B. M. Divya, “An interpretation of lemmatization and stemming in natural language processing,” vol. 22, no. 10, pp. 350–357, 2021.

[24] G. Hackeling, *Mastering Machine Learning with scikit-learn*. Packt Publishing Ltd, 2017.

[25] K. Tanaka and T. Saito, “Python deserialization denial of services attacks and their mitigations,” pp. 15–25, 2019.

[26] M. Hammond and A. Robinson, *Python programming on win32: Help for windows programmers*. “ O’Reilly Media, Inc.,” 2000.

[27] P. Hughes, “Python and tkinter programming,” vol. 2000, no. 77es, pp. 23-es, 2000.

[28] J. Browniee, “What is confusion matrix in machine learning.”, Machine Learning Mastery, 2016. Available at: https://machinelearningmastery.com/confusion-matrix-machine-learning/

[29] J. Chowdhury, “Confusing metrics around the Confusion Matrix”, Medium, 2022. Available at: https://towardsdatascience.com/confusing-metrics-around-the-confusion-matrix-6ee54e4ed603