

**REPORT**

**SPAM MAIL DETECTOR**

(Machine Learning Project)

**Under the guidance of**

**Aishwarya Saxena**

**Name :** Ritika Goyal (B.tech 2nd year)

**College :** NIT Jalandhar, Punjab

ML Summer Intern (June – July 2020)

**INDEX**

|  |  |  |
| --- | --- | --- |
| **Sr no.** | **TOPIC** | **Pg no.** |
| 1. | Introduction | 3 |
| 2. | Objective | 4 |
| 3. | Background | 4-5 |
| 4. | Hardware and software requirements | 6 |
| 5. | Coding and output | 7-13 |
| 6. | Future scope | 14 |
| 7. | Conclusion | 14 |
| 8. | Bibliography and references | 14-15 |

**INTRODUCTION**

Spam email is one of the biggest issues in the world of the internet. Spam prevents the users from having productive time usage, storage capacity and network bandwidth. The menace of spam email is increasing on a yearly basis and is responsible for more than 70% of the whole global email traffic . Users who receive spam emails that they did not request find it very irritating and disturbing. It has also resulted in untold financial losses to many users who have fallen victim to internet scams and other practices of spammers who send emails pretending to be from reputable companies with the intention to persuade individuals to disclose sensitive personal information like passwords and credit card numbers.

This paper is organized as follows, section 1 is the paper introduction, section 2 summarizes the objective of this project, section 3 gives background, next section comprises hardware and software requirements followed by code used, last section finally concludes the entire project.

**Spam Ham**

Congratulations for the new job. Want to grab some coffee tomorrow at our old canteen? I would love to know about your new environment. Feel free to call me anytime.

ONE-POUND-A-DAY DIET (back by popular demand)

FREE delicious Caesar Salad recipe included in this email!

Do you have an over-weight problem that you can't seem to beat?

Have you tried diet after diet with no results? Are you too busy to

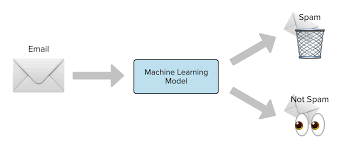
buy special diet foods? Do you want a simple, quick one-pound-a

day diet that gets you really slim, really fast?

**OBJECTIVE**

The main objective of the Spam mail detector is:

* to give knowledge to the user about fake emails and relevant emails
* to classify whether the mail is spam or not.
* To classify all related texts as spam or ham.



**BACKGROUND**

Various approaches towards spam filtering have been adopted. Naive Bayes is one of the most popular algorithms used in these approaches; we will also be using Naïve Bayes in this project.

Data is the essential ingredients before we can develop any meaningful algorithm. In our dataset, there are two types of data present, one is **ham** (non-spam) and other is **spam** data.

In order to classify our texts as spam and ham, we will create a dictionary of most commonly used words in text messages.

Since we will be needing this dictionary lot of times, so we will save this using pickle library provided by python. The pickle module implements binary protocols for serializing and de-serializing a Python object structure. *“Pickling”* is the process whereby a Python object hierarchy is converted into a byte stream, and *“unpickling”* is the inverse operation, whereby a byte stream (from a binary file or bytes-like object) is converted back into an object hierarchy.

**Naïve Bayes Classifier:**

The general term **Naive Bayes** refers to the strong independence assumptions in the model, rather than the particular distribution of each feature. A Naive Bayes model assumes that each of the features it uses are conditionally independent of one another given some class. More formally, if we want to calculate the probability of observing features f1 through fn, given some class c, under the Naive Bayes assumption the following holds:

p(f1...,fn|c) = ∏ni=1 p(fi|c)

This means that when we want to use a Naive Bayes model to classify a new example, the posterior probability is much simpler to work with:

p(c|f1,...,fn) ∝ p(c)p(f1|c)...p(fn|c)

The term **Multinomial Naive Bayes** simply lets us know that each p(fi|c)p(fi|c) is a multinomial distribution, rather than some other distribution. This works well for data which can easily be turned into counts, such as word counts in text.

In summary, Naive Bayes classifier is a general term which refers to conditional independence of each of the features in the model, while Multinomial Naive Bayes classifier is a specific instance of a Naive Bayes classifier which uses a multinomial distribution for each of the features. Since we are dealing with text classification, we will use Multinomial Naïve Bayes Classifier.

**HARDWARE AND SOFTWARE REQUIREMENTS**

**Hardware requirements:**

|  |  |
| --- | --- |
| Hardware tools | Minimum requirements |
| Processor | i3 or above |
| Hard disk | 10 GB |
| RAM | 8 GB |
| Monitor | 17” coloured |
| Mouse | Optical |
| Keyboard | 122 Keys |

**Software requirements:**

|  |  |
| --- | --- |
| Software tools | Minimum requirements |
| Operating System | Windows, Linux or MacOS |
| Platform | Windows, Linux or MacOS |
| Technology | Machine Learning Python |
| Scripting Language | Python |
| IDE | Pycharm (or Jupyter Notebook) |

**CODING**

We have two programs here

* Spam.py contains dataset , training and testing
* Detect.py contains GUI in which we can detect various mails as spam and ham.

**Spam.py**

***Libraries used:***

**import** os  
**from** collections **import** Counter  
*# Above library is used to count frequency of words which appear in the dataset.***from** sklearn.naive\_bayes **import** MultinomialNB  
**from** sklearn.model\_selection **import** train\_test\_split **as** tts  
**from** sklearn.metrics **import** accuracy\_score  
**import** \_pickle **as** c  
*# this is used to save our files created so we dont have to create them again and again.***import** matplotlib.pyplot **as** plt

***Main code:***

*#this function saves our file using pickle library***def** save(clf, name):  
 **with** open(name, **'wb'**) **as** fp:  
 c.dump(clf, fp)  
 print(**"saved"**)  
  
*#this will create a dictionary of words used in our data and will return most common 3000 words.***def** make\_dict():  
 direc = **"emails/"** files = os.listdir(direc) *# returns a list containing the names of the files in this directory* emails = [direc + email **for** email **in** files] *# this list contains all the files, basically appending paths* words = []  
 length = len(emails)  
 *#this function will give ll the words present in the data used.* **for** email **in** emails:  
 f = open(email, errors=**"ignore"**)  
 currfile = f.read()  
 words += currfile.split(**" "**)  
 print(length)  
 length -= 1  
  
 *#this will remove all unnnecssary characters like '-','+', '.' , '?'* **for** i **in** range(len(words)):  
 **if** words[i].isalpha() == **False or** len(words[i]) <= 1: *# deleting all those characters which are not alphabets* words[i] = **""** dictionary = Counter(words)  
 **del** dictionary[**""**]  
 *#print(dictionary)* **return** dictionary.most\_common(3000)  
  
*#this function will create our dataset required***def** make\_dataset(dictionary):  
 direc = **"emails/"** files = os.listdir(direc)  
  
 emails = [direc + email **for** email **in** files]  
  
 feature\_set = [] *#this will include count of words in a particular mail* labels = [] *# this will tell whether it is a spam or ham (1 for spam and 0 for ham)* length = len(emails)

**for** email **in** emails:  
 data = []  
 f = open(email, errors =**"ignore"**)  
 words = f.read().split(**' '**)  
 **for** entry **in** dictionary:  
 data.append(words.count(entry[0]))  
 *#print(data)* feature\_set.append(data)  
 **if "ham" in** email:  
 labels.append(0)  
 **if "spam" in** email:  
 labels.append(1)  
 *#print(length)* length -= 1  
 **return** feature\_set,labels

d = make\_dict()  
save(d,**"spamdict.dict"**)  
features , labels = make\_dataset(d)  
*#print(len(features) ,len(labels))  
  
  
#training and testing the data using multinomial Naive Bayes classifier*x\_train,x\_test, y\_train , y\_test = tts(features , labels , test\_size=0.2)  
clf = MultinomialNB()  
  
clf.fit(x\_train, y\_train)  
  
preds = clf.predict(x\_test)  
  
print(accuracy\_score(y\_test,preds))

*#saving the classifier so that we can use it to predict other emails.*save(clf, **"test-classifier.mdl"**)

**Detect.py**

***Libraries used:***

**import** os  
**import** \_pickle **as** c  
**from** sklearn **import** \*  
**from** collections **import** Counter  
**from** tkinter **import** \*

***Main code:***

**def** load(clf\_file):  
 **with** open(clf\_file, **'rb'**) **as** fp:  
 clf = c.load(fp)  
 **return** clf  
  
*#loading classifier and dictionary*clf = load(**"test-classifier.mdl"**)  
d= load(**"spamdict.dict"**)  
  
*# this is similar to creating dataset function in spam.py***def** detect():  
 loadinglabel.config(text = **""**)  
 features = []  
 inp = entrytext.get(**"1.0"** ,END).split()  
 *#print(inp)* **if** len(inp) == 0:  
 loadinglabel.config(text = **"Please enter the mail"** )

**else**:  
 **for** word **in** d:  
 features.append(inp.count(word[0]))  
 res = clf.predict([features])  
 **if** res == 0:  
 loadinglabel.config(text=**"Hurray.. It is Not Spam!"**, fg=**"#3AED09"**)  
 **else**:  
 loadinglabel.config(text=**"Beware it is a Spam!"**, fg=**"#810E0E"**)  
 submitbttn.config(text=**"Detect Again"**)  
 entrytext.delete(**"1.0 "**, END)  
  
  
*#main GUI*win = Tk()  
win.geometry(**"500x500"**)

win.title(**"Spam Mail Detector"**)

win.config(bg = **"white"**)

topframe = Frame(height = 80 ,bg = **"#EFF4F3"**)  
topframe.pack(side = TOP ,fill = X)

label = Label(topframe,text = **"Spam Mail Detector"** , fg = **"#CD3142"**, bg = **"#EFF4F3"** , font = **'Times 30 bold'**)  
label.place(x = 80,y=10)

enterlabel = Label(win,text = **"Paste the content of the email below"** , bg =**"white"** ,fg =**"black"** , font = **'Times 13 italic'**)  
enterlabel.place(x = 90,y = 90)

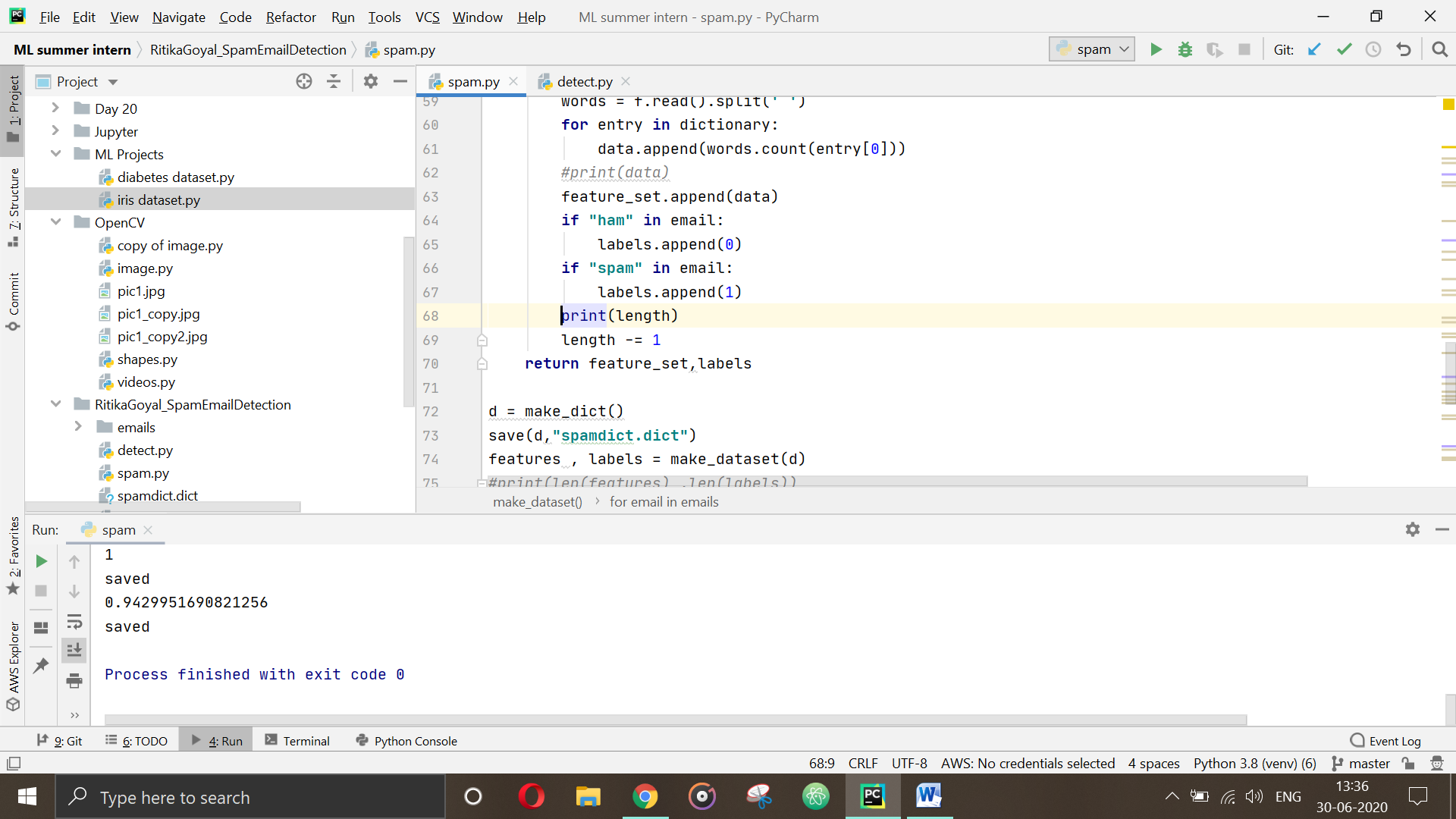
entrytext = Text(win, bd = 3, height = 10,width = 40,padx = 12,pady = 12 ,bg =**"#EFF4F3"**)  
entrytext.place(x = 80, y = 130 )

loadinglabel = Label(win,text = **""**,bg =**"white"** ,font = **'Times 15 bold'**)  
loadinglabel.place(x = 150, y = 320)

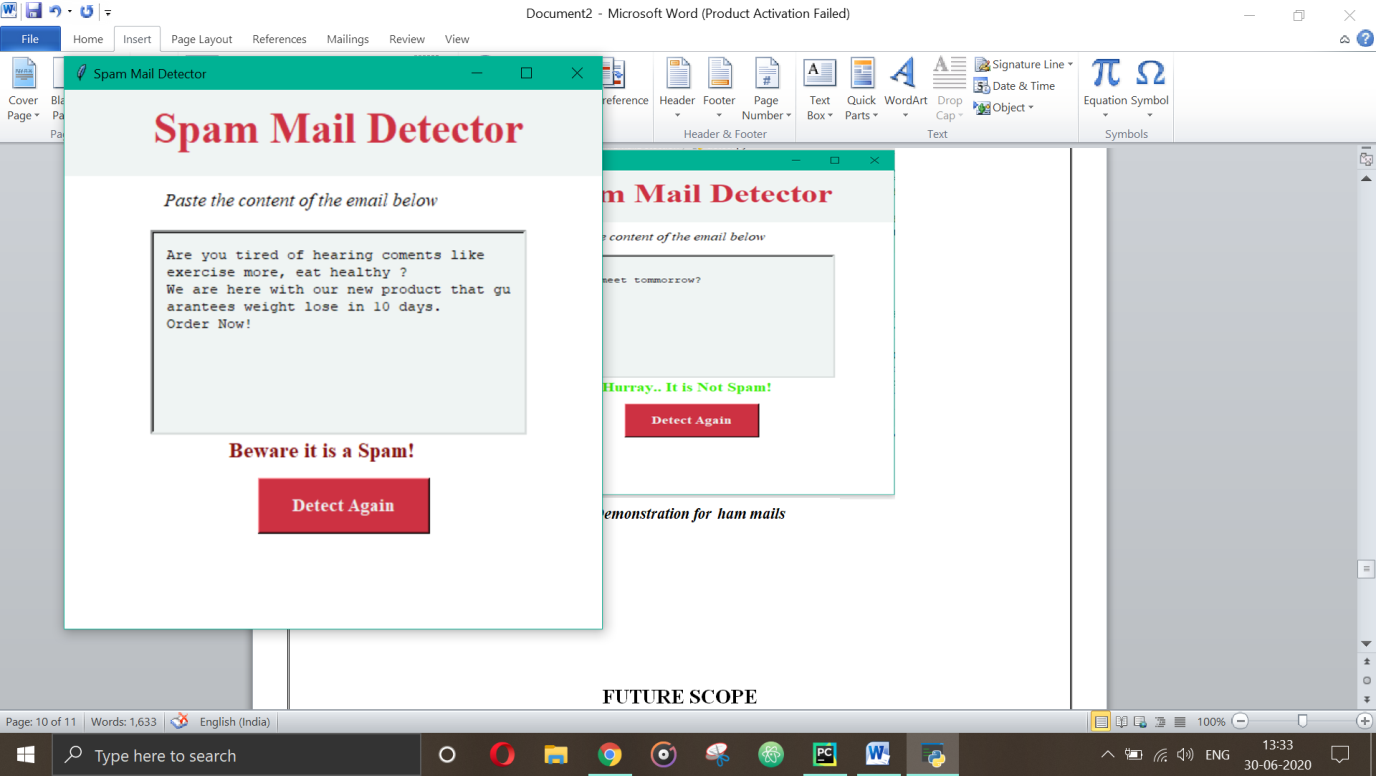
submitbttn = Button(win,text = **"Detect"** , height = 2,width =15 ,font = **'Times 13 bold'**, fg = **"#EFF4F3"** , bg = **"#CD3142"**,activebackground = **"#098273"** ,activeforeground = **"white"** ,command = detect)  
submitbttn.place(x = 180,y = 360)

win.mainloop()

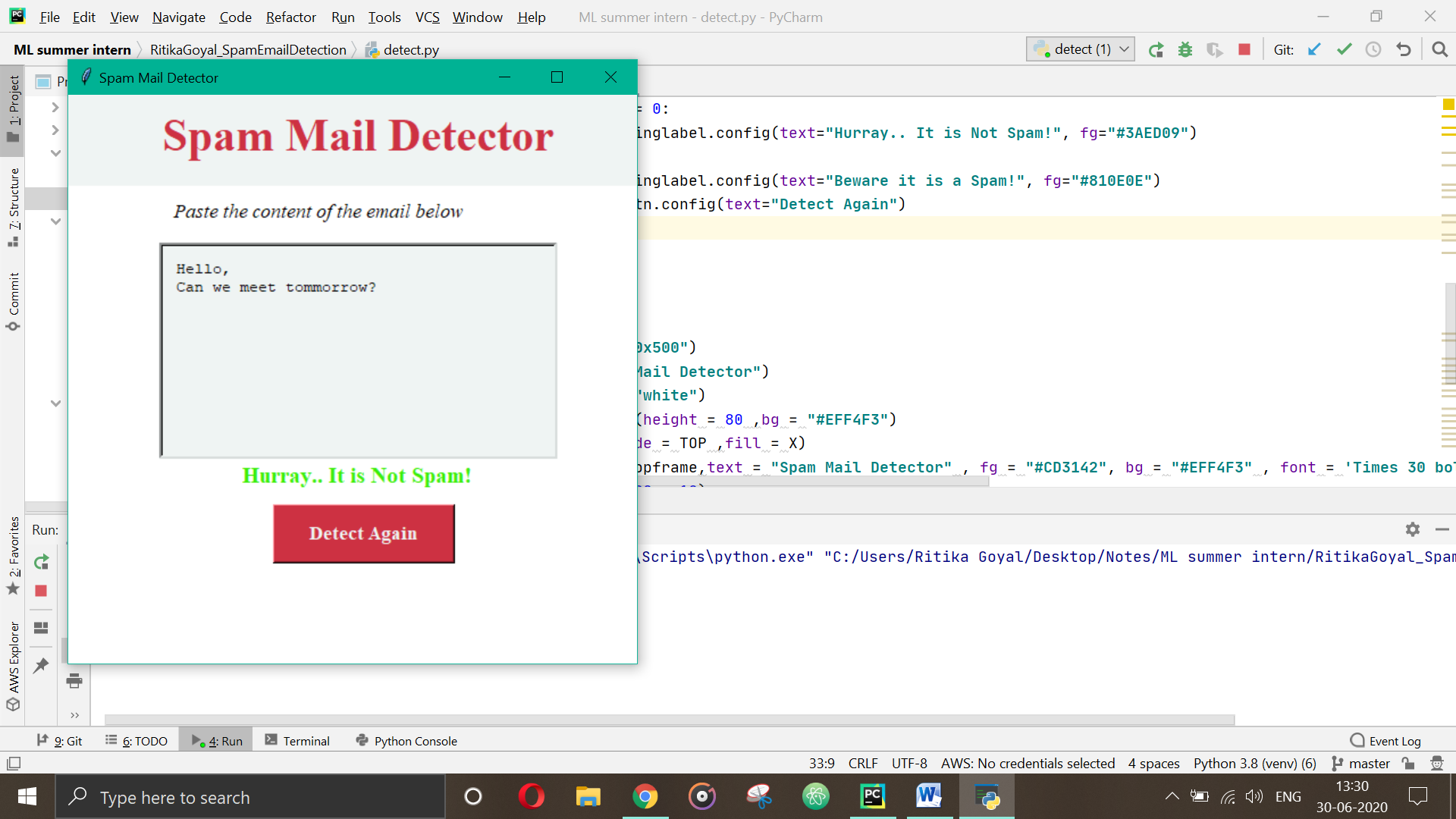
**OUTPUT SCREENSHOTS**

****

***Accuracy of our model***

****

***Demonstration for spam mails***

****

***Demonstration for ham mails***

**FUTURE SCOPE**

This project is limited to only texts related data. Images, videos and other formats of data are not included. But, in future it might be possible to include these other forms as well.

Also, the current project scope is limited to only English language but can be designed for multiple languages as well. So, let’s just see where Artificial Intelligence and machine learning can take us in future and how it will improve our life further.

**CONCLUSION**

In recent years, spam detection has received significant attention in both business and academics. In this project, we have presented an approach for detecting whether a particular mail is spam or not.

We have been successful in implementing this project and we can classify email as spam or ham. This project can also be used for detecting SMS as well as other text messages.

**BIBLIOGRAPHY AND REFERENCES**

[1] Sklearn Libraries (https://scikit-learn.org/)

[2] Algorithm reference

(https://towardsdatascience.com/) ,

(www.youtube.com) ,

(https://medium.com)

[3] Naïve Bayes­­­ study and Mutinomial Naïve Bayes

(https://monkeylearn.com/blog/practical-explanation-naive-bayes-classifier/) , (https://www.kdnuggets.com/2020/06/naive-bayes-algorithm-everything.html),

https://stats.stackexchange.com/questions/33185/difference-between-naive-bayes-multinomial-naive-bayes

[4] Pickle Library (https://docs.python.org/3/library/pickle.html)