Here to Share my expirences on creating a Spam Email Predictor using Sklearn using many algorithms and how I chose the best model for the best result 🏅👍

There was a task given to me as to build any application using the techniques learned in the Data Mining Course in this semester.I was searching for apps and was in confusion of choosing an app .Then it struck me.In the some Data Mining Session if I am not wrong it was in TF-IDF session where an example of Email Spam Classifer came into play.That time itself I thought that I must create a Email spam classifer some day.I dont know why it came into my mind but it stuck into my mind.I didn't know that this soon I would be develeoping a simple Email spam prediction app this semester itself.It was there in my mind so I thought I would build it during my job serach time.

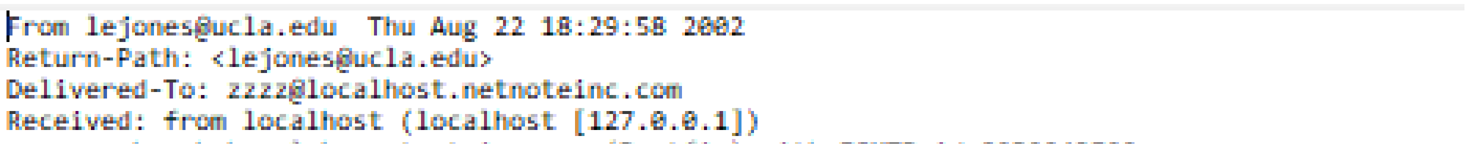
About the Dataset Used

As I was fixed in doing a email spam predictor I was searching for the dataset to be used.Then I came across the below mentioed dataset from kaggle.

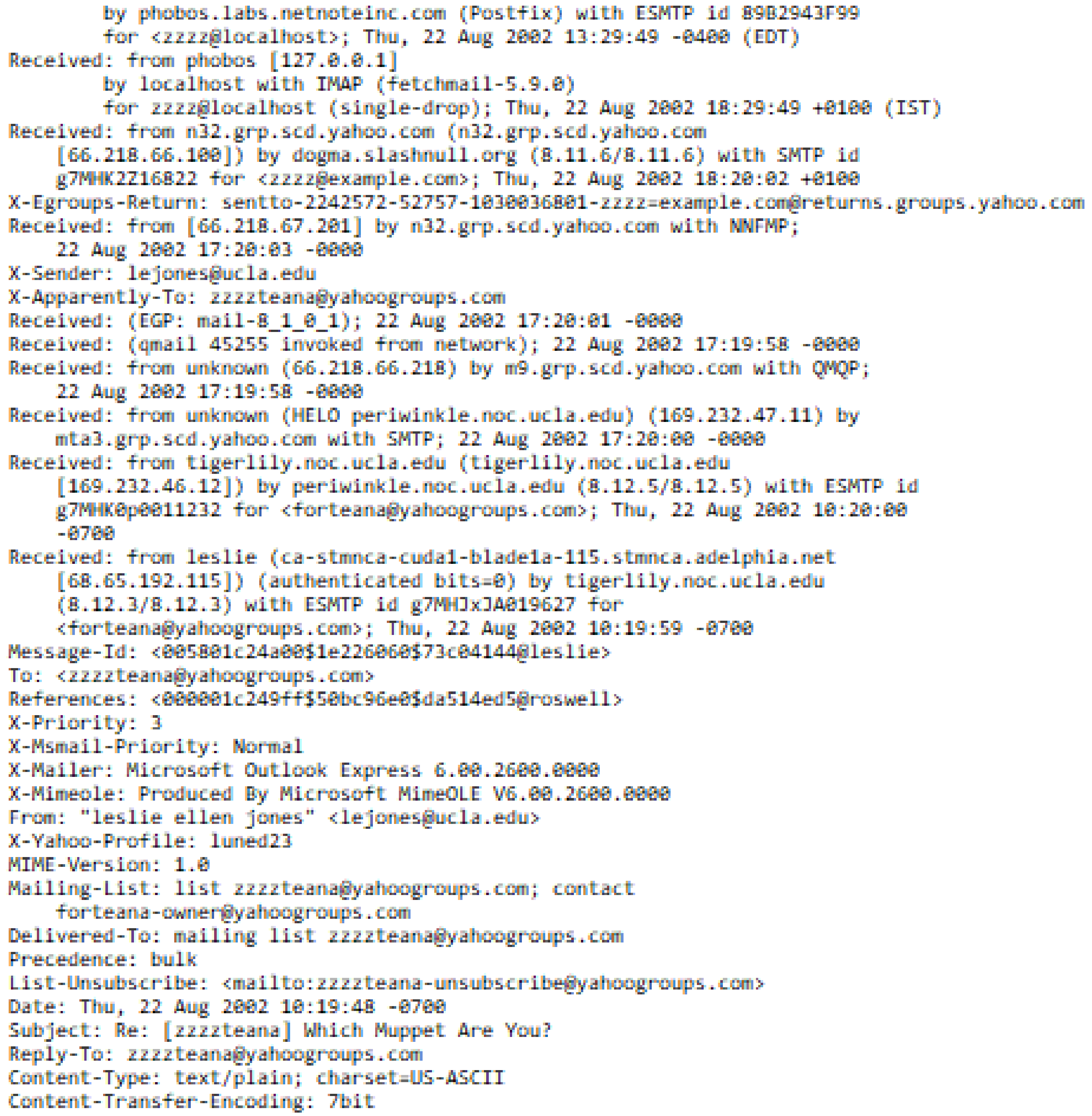
Spam Mail Dataset

The above dataset constists of two Folders Ham and Spam respectively.Each folders consits of text files.Added to there were 3052 text files[2551-Non Spam Text Files,501-Spam text Files].In each file there would be mail.

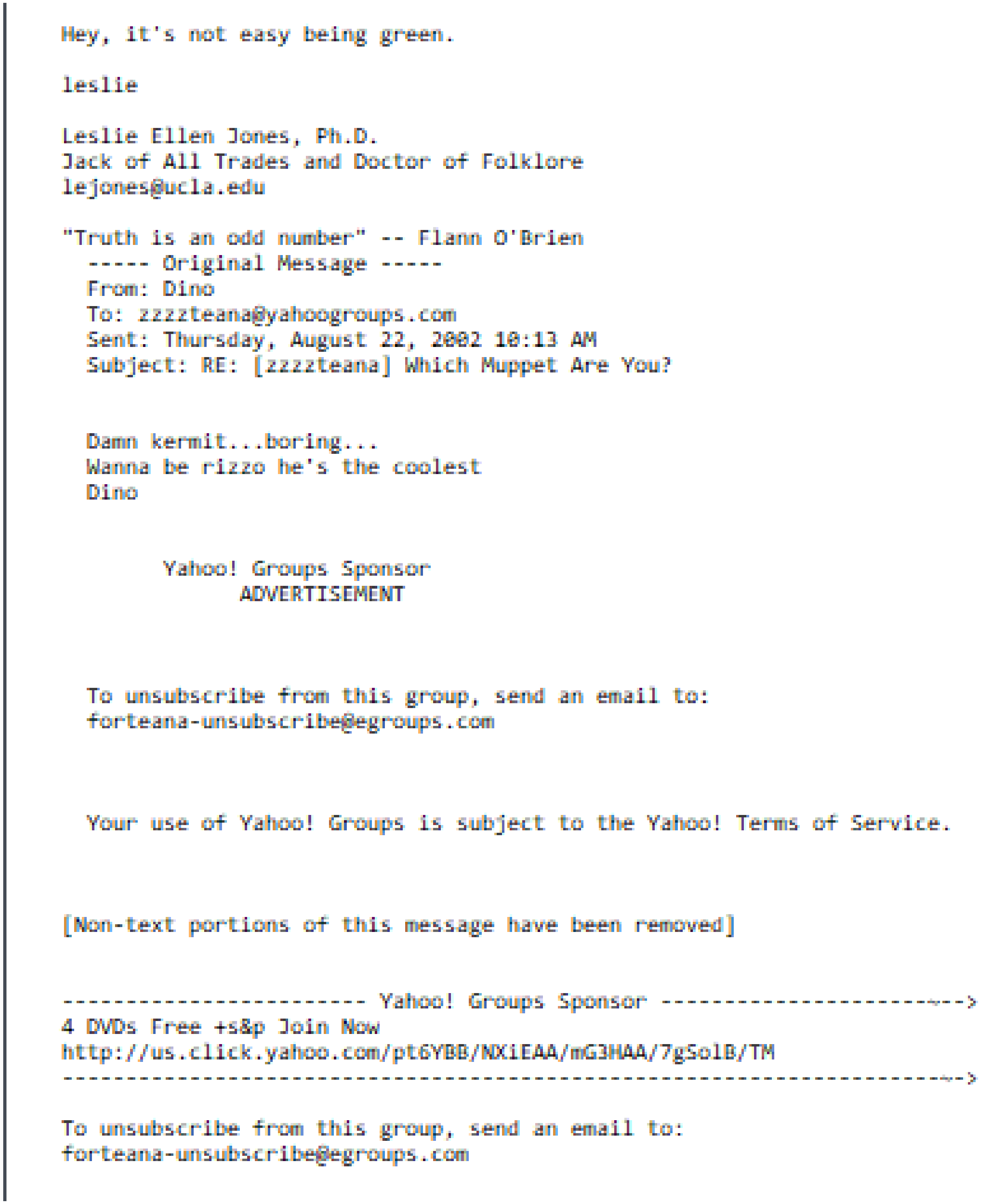
All the mails in the text files could be read by means of Email Library in python



```
From lejones@ucla.edu  Thu Aug 22 18:29:58 2002
Return-Path: <lejones@ucla.edu>
Delivered-To: zzzz@localhost.netnoteinc.com
Received: from localhost (localhost [127.0.0.1])
```



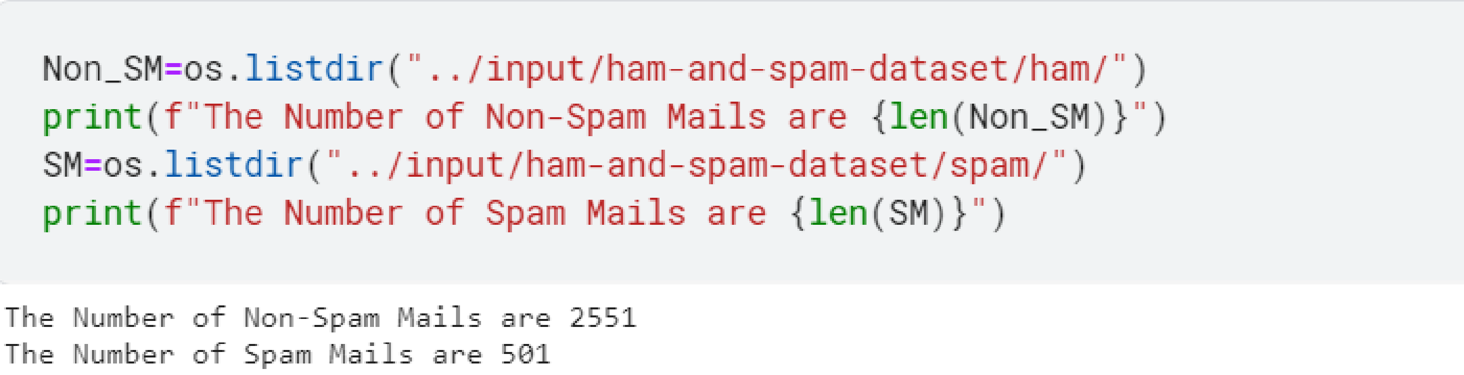
```
        by phobos.labs.netnoteinc.com (Postfix) with ESMTP id 89B2943F99
        for <zzzz@localhost>; Thu, 22 Aug 2002 13:29:49 -0400 (EDT)
Received: from phobos [127.0.0.1]
        by localhost with IMAP (fetchmail-5.9.0)
        for zzzz@localhost (single-drop); Thu, 22 Aug 2002 18:29:49 +0100 (IST)
Received: from n32.grp.scd.yahoo.com (n32.grp.scd.yahoo.com
    [66.218.66.100]) by dogma.slashnull.org (8.11.6/8.11.6) with SMTP id
    g7MHK2Z16822 for <zzzz@example.com>; Thu, 22 Aug 2002 18:20:02 +0100
X-Egroups-Return: sentto-2242572-52757-1030036801-zzzz=example.com@returns.groups.yahoo.com
Received: from [66.218.67.201] by n32.grp.scd.yahoo.com with NNFMP;
    22 Aug 2002 17:20:03 -0000
X-Sender: lejones@ucla.edu
X-Apparently-To: zzzzteana@yahoogroups.com
Received: (EGP: mail-8_1_0_1); 22 Aug 2002 17:20:01 -0000
Received: (qmail 45255 invoked from network); 22 Aug 2002 17:19:58 -0000
Received: from unknown (66.218.66.218) by m9.grp.scd.yahoo.com with QMQP;
    22 Aug 2002 17:19:58 -0000
Received: from unknown (HELO periwinkle.noc.ucla.edu) (169.232.47.11) by
    mta3.grp.scd.yahoo.com with SMTP; 22 Aug 2002 17:20:00 -0000
Received: from tigerlily.noc.ucla.edu (tigerlily.noc.ucla.edu
    [169.232.46.12]) by periwinkle.noc.ucla.edu (8.12.5/8.12.5) with ESMTP id
    g7MHK0p0011232 for <forteana@yahoogroups.com>; Thu, 22 Aug 2002 10:20:00
    -0700
Received: from leslie (ca-stmnca-cuda1-blade1a-115.stmnca.adelphia.net
    [68.65.192.115]) (authenticated bits=0) by tigerlily.noc.ucla.edu
    (8.12.3/8.12.3) with ESMTP id g7MHJxJA019627 for
    <forteana@yahoogroups.com>; Thu, 22 Aug 2002 10:19:59 -0700
Message-Id: <005801c24a00$1e226060$73c04144@leslie>
To: <zzzzteana@yahoogroups.com>
References: <000001c249ff$50bc96e0$da514ed5@roswell>
X-Priority: 3
X-Msmail-Priority: Normal
X-Mailer: Microsoft Outlook Express 6.00.2600.0000
X-Mimeole: Produced By Microsoft MimeOLE V6.00.2600.0000
From: "leslie ellen jones" <lejones@ucla.edu>
X-Yahoo-Profile: luned23
MIME-Version: 1.0
Mailing-List: list zzzzteana@yahoogroups.com; contact
    forteana-owner@yahoogroups.com
Delivered-To: mailing list zzzzteana@yahoogroups.com
Precedence: bulk
List-Unsubscribe: <mailto:zzzzteana-unsubscribe@yahoogroups.com>
Date: Thu, 22 Aug 2002 10:19:48 -0700
Subject: Re: [zzzzteana] Which Muppet Are You?
Reply-To: zzzzteana@yahoogroups.com
Content-Type: text/plain; charset=US-ASCII
Content-Transfer-Encoding: 7bit
```



```
Hey, it's not easy being green.

leslie

Leslie Ellen Jones, Ph.D.
Jack of All Trades and Doctor of Folklore
lejones@ucla.edu

"Truth is an odd number" -- Flann O'Brien
  ----- Original Message -----
  From: Dino
  To: zzzzteana@yahoogroups.com
  Sent: Thursday, August 22, 2002 10:13 AM
  Subject: RE: [zzzzteana] Which Muppet Are You?


  Damn kermit...boring...
  Wanna be rizzo he's the coolest
  Dino


          Yahoo! Groups Sponsor
                ADVERTISEMENT



  To unsubscribe from this group, send an email to:
  forteana-unsubscribe@egroups.com



  Your use of Yahoo! Groups is subject to the Yahoo! Terms of Service.



[Non-text portions of this message have been removed]


------------------------ Yahoo! Groups Sponsor ---------------------~-->
4 DVDs Free +s&p Join Now
http://us.click.yahoo.com/pt6YBB/NXiEAA/mG3HAA/7gSolB/TM
---------------------------------------------------------------------~->

To unsubscribe from this group, send an email to:
forteana-unsubscribe@egroups.com
```

Deep Dive into the code

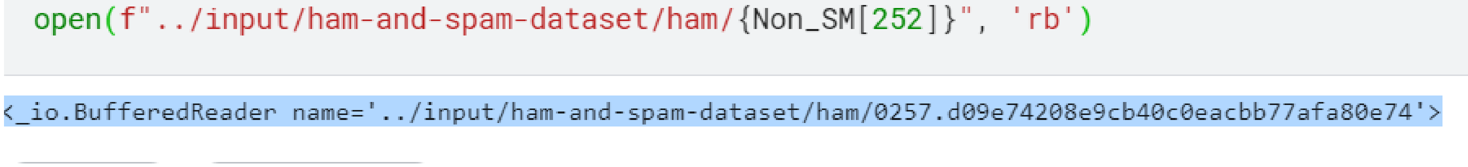
First I was going through the dataset.The first thing that came into my mind was it is a text file and hence one need to read the file and Os library is needed for that so imported the os library.For reading the contents in the file one need to open the file using the path.So first I decided to save all the file names of respective folders in a separate list



```
Non_SM=os.listdir("../input/ham-and-spam-dataset/ham/")
print(f"The Number of Non-Spam Mails are {len(Non_SM)}")
SM=os.listdir("../input/ham-and-spam-dataset/spam/")
print(f"The Number of Spam Mails are {len(SM)}")
```

```
The Number of Non-Spam Mails are 2551
The Number of Spam Mails are 501
```

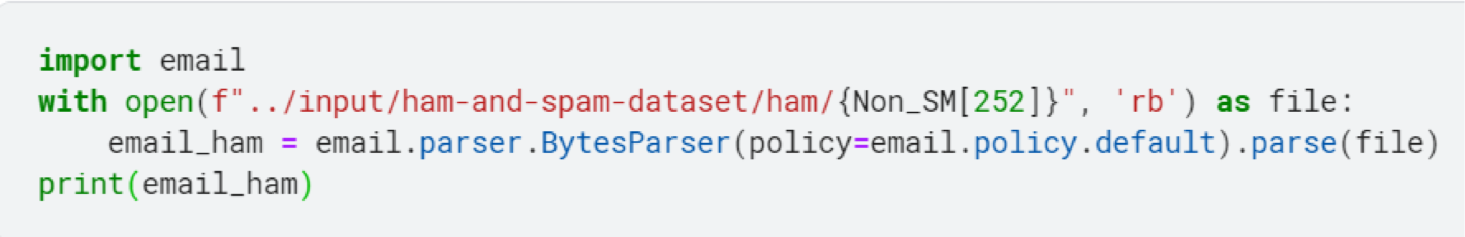
The next step is to analyze the data in the text file so I just opened the file and got this



```
open(f"../input/ham-and-spam-dataset/ham/{Non_SM[252]}", 'rb')
```

```
<_io.BufferedReader name='../input/ham-and-spam-dataset/ham/0257.d09e74208e9cb40c0eacbb77afa80e74'>
```

Then I was surfuing what could be done so that I could read the data and I came across this particualr document where I found the way to read the file here[Challange#1]



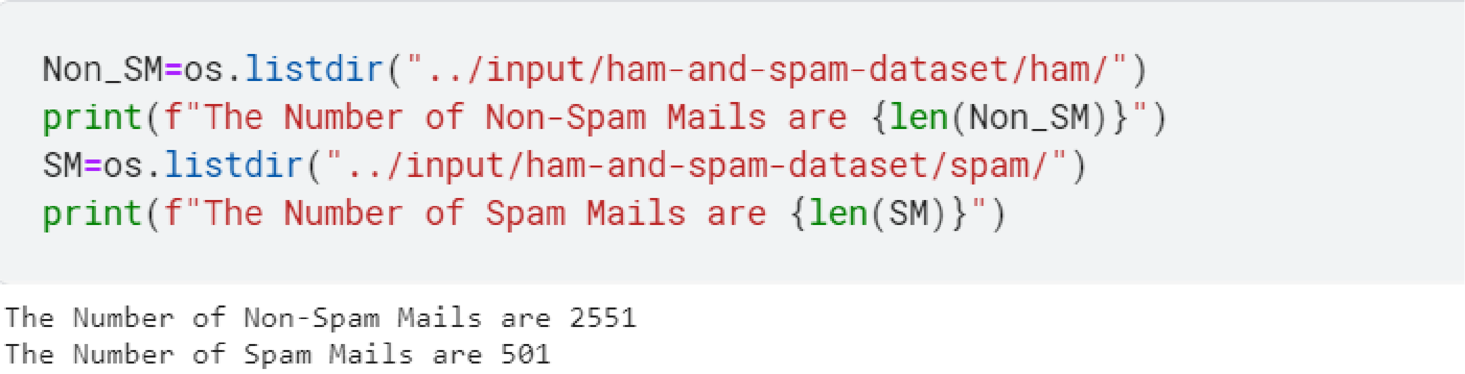
```
import email
with open(f"../input/ham-and-spam-dataset/ham/{Non_SM[252]}", 'rb') as file:
    email_ham = email.parser.BytesParser(policy=email.policy.default).parse(file)
print(email_ham)
```

The above snippet gave me the contents in the above mentioned file

EDA

Now I need to analyse the data so that to transform the data according to my wish

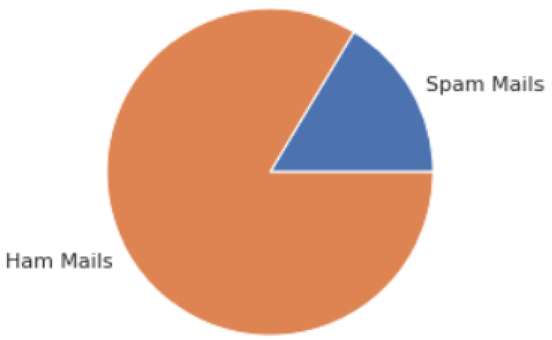
1.Number of Spam and Non-Spam mails



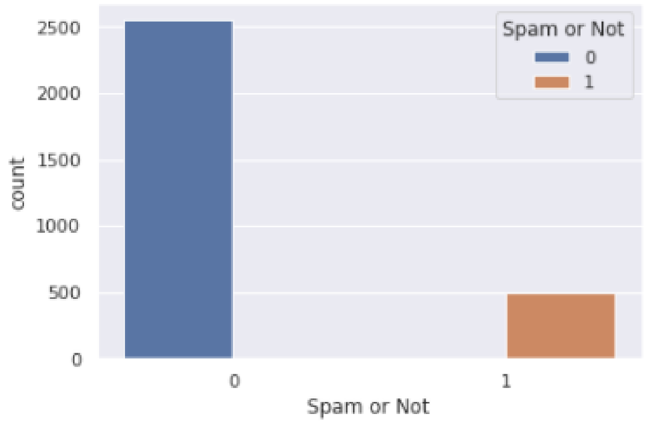
```
Non_SM=os.listdir("../input/ham-and-spam-dataset/ham/")
print(f"The Number of Non-Spam Mails are {len(Non_SM)}")
SM=os.listdir("../input/ham-and-spam-dataset/spam/")
print(f"The Number of Spam Mails are {len(SM)}")
```

```
The Number of Non-Spam Mails are 2551
The Number of Spam Mails are 501
```

```
percent_spam=len(SM)/(len(SM)+len(Non_SM))
percent_spam
percent_Non_Spam=1-percent_spam
plt.pie([percent_spam,percent_Non_Spam],labels =["Spam Mails","Ham Mails"])
plt.show()
```



```
sns.set_theme(style="darkgrid")
ax = sns.countplot(x="Spam or Not", data=cleaned_data,hue='Spam or Not') # 0-Ham Mail 1-Spam Mail
# ax = sns.countplot(x="Spam or Not", hue="Spam or No", data=cleaned_data)
```



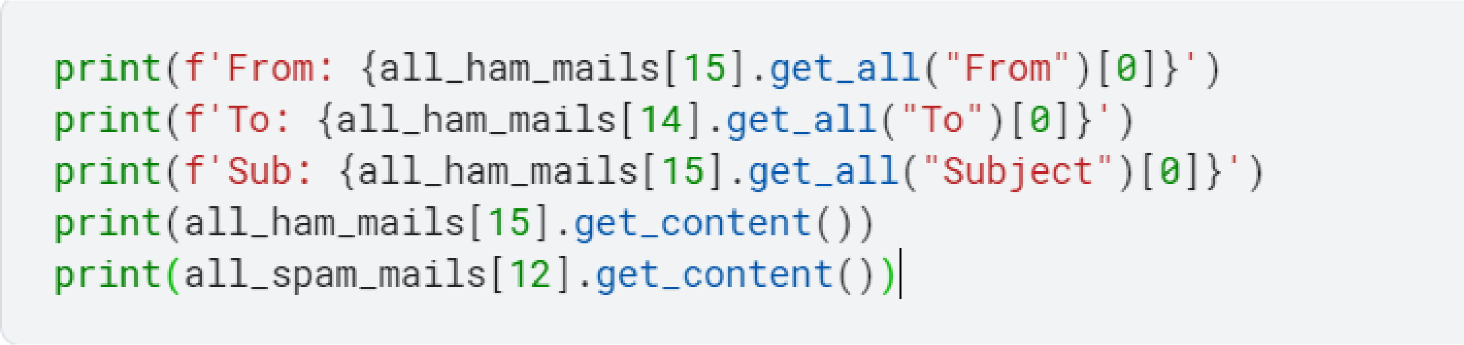
2.Reading different portions of Mail

For this first I have read all the spam mails and stored it in an list and I have also read all the non-spam mails and stored it in an array.

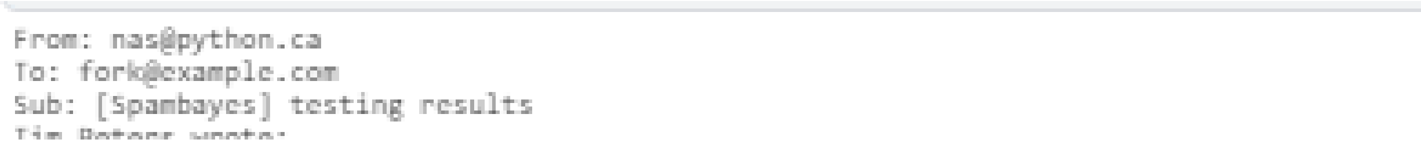
```
all_spam_mails=[]
all_ham_mails=[]
for file1 in SM:
    with open(f"../input/ham-and-spam-dataset/spam/{file1}", 'rb') as file:
        all_spam_mails.append(email.parser.BytesParser(policy=email.policy.default).parse(file))
for file1 in Non_SM:
    with open(f"../input/ham-and-spam-dataset/ham/{file1}", 'rb') as file:
        all_ham_mails.append(email.parser.BytesParser(policy=email.policy.default).parse(file))
```

Then I need to deep dive into the data and check what and all are present in the data.Then the next thing I thought was that what if I can read the From, To ,Subject and the Content of the mail separately.Then I refered this and came across the different functions present in the email library and was experimenting with the same.There are several functions that Email library has but in this I have used

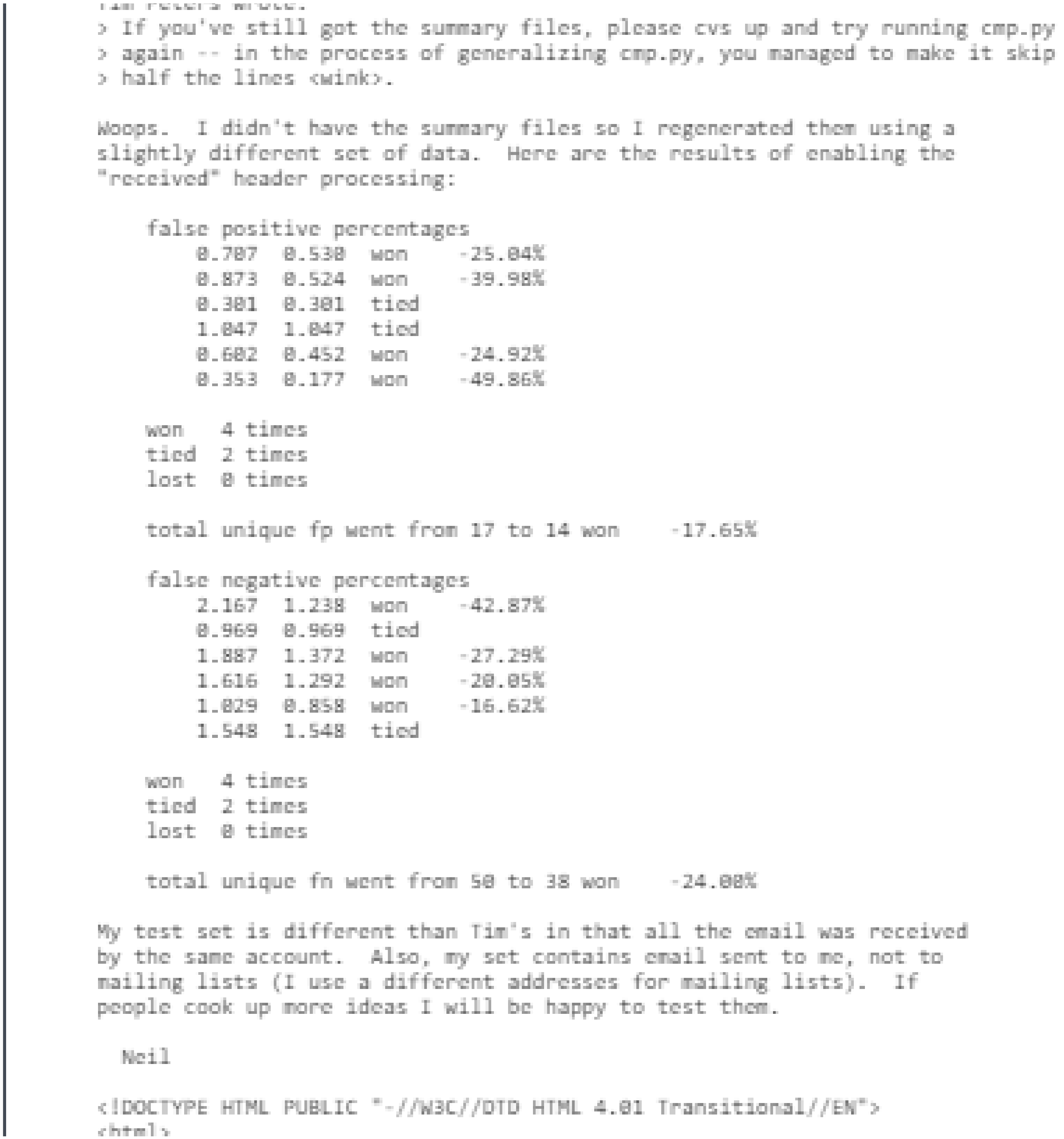
- get_all('Name'):Return a list of all the values for the field named name.
- get_content():Returns the body of the mail
- get_payload():To Get the payload of the EmailMessage



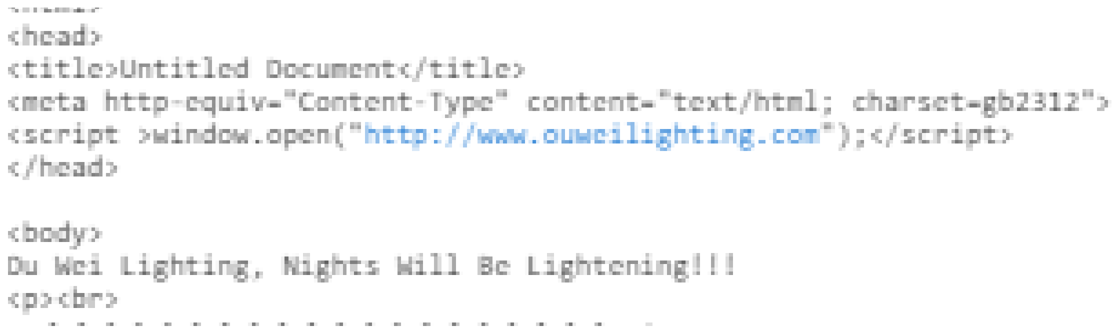
```
print(f'From: {all_ham_mails[15].get_all("From")[0]}')
print(f'To: {all_ham_mails[14].get_all("To")[0]}')
print(f'Sub: {all_ham_mails[15].get_all("Subject")[0]}')
print(all_ham_mails[15].get_content())
print(all_spam_mails[12].get_content())
```



```
From: nas@python.ca
To: fork@example.com
Sub: [Spambayes] testing results
Tim Peters wrote:
```



```
Tim Peters wrote:
> If you've still got the summary files, please cvs up and try running cmp.py
> again -- in the process of generalizing cmp.py, you managed to make it skip
> half the lines <wink>.

Woops.  I didn't have the summary files so I regenerated them using a
slightly different set of data.  Here are the results of enabling the
"received" header processing:

    false positive percentages
        0.707  0.530  won    -25.04%
        0.873  0.524  won    -39.98%
        0.301  0.301  tied
        1.047  1.047  tied
        0.602  0.452  won    -24.92%
        0.353  0.177  won    -49.86%

    won   4 times
    tied  2 times
    lost  0 times

    total unique fp went from 17 to 14 won    -17.65%

    false negative percentages
        2.167  1.238  won    -42.87%
        0.969  0.969  tied
        1.887  1.372  won    -27.29%
        1.616  1.292  won    -20.05%
        1.029  0.858  won    -16.62%
        1.548  1.548  tied

    won   4 times
    tied  2 times
    lost  0 times

    total unique fn went from 50 to 38 won    -24.00%

My test set is different than Tim's in that all the email was received
by the same account.  Also, my set contains email sent to me, not to
mailing lists (I use a different addresses for mailing lists).  If
people cook up more ideas I will be happy to test them.

  Neil

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
```



```
<head>
<title>Untitled Document</title>
<meta http-equiv="Content-Type" content="text/html; charset=gb2312">
<script >window.open("http://www.ouweilighting.com");</script>
</head>

<body>
Ou Wei Lighting, Nights Will Be Lightening!!!
<p><br>
```

Then I tried to get the types of the email from the messages in the file using get_payload()

```
def email_content_type(email):
    payload = email.get_payload()
    if isinstance(payload, list):
        return "multipart({})".format(", ".join([email_content_type(sub_email) for sub_email in payload]))
    else:
        return email.get_content_type()
```

```
email_content_type(all_spam_mails[131])
```

```
'multipart(text/plain)'
```

The output would be either 'text/plain' or 'text/html' or 'Any Multipart type'

If it is a Text/plain or text/html it is fine.But if it is a multipart text are difficult to analyse.

Multipart texts:In the case of multiple part messages, in which one or more different sets of data are combined in a single body, a "multipart" Content-Type field must appear in the entity's header. The body must then contain one or more "body parts," each preceded by an encapsulation boundary, and the last one followed by a closing boundary [as found in here]

So in this I have analysed 'text/plain 'text/html' for the multipart content type I have displayed the content as none

Feature Engineering

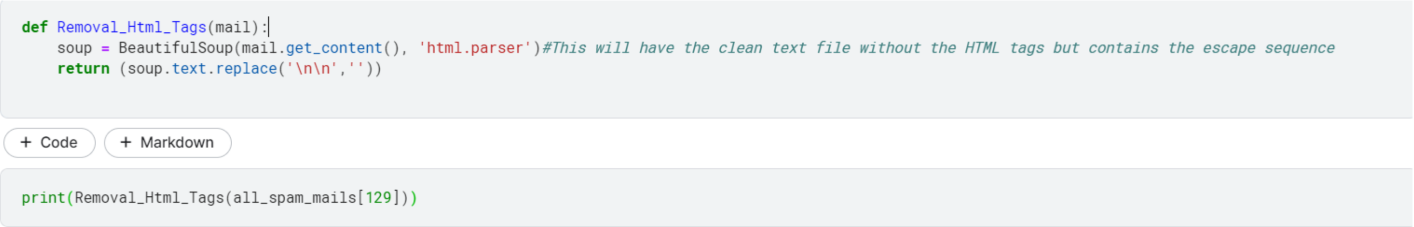
Need to Perfom some feature engineering as the data in the dataset contains lots of html tags so the primary goal is to remove the html tags to texts

1. Removal_Html_Tags

   Lots of data in the dataset constsis of data with html tags so removal of the same is necessary so as to vectorize the data and predict the output with high accuarcy. The function Removal_Html_Tags takes the argument an email.

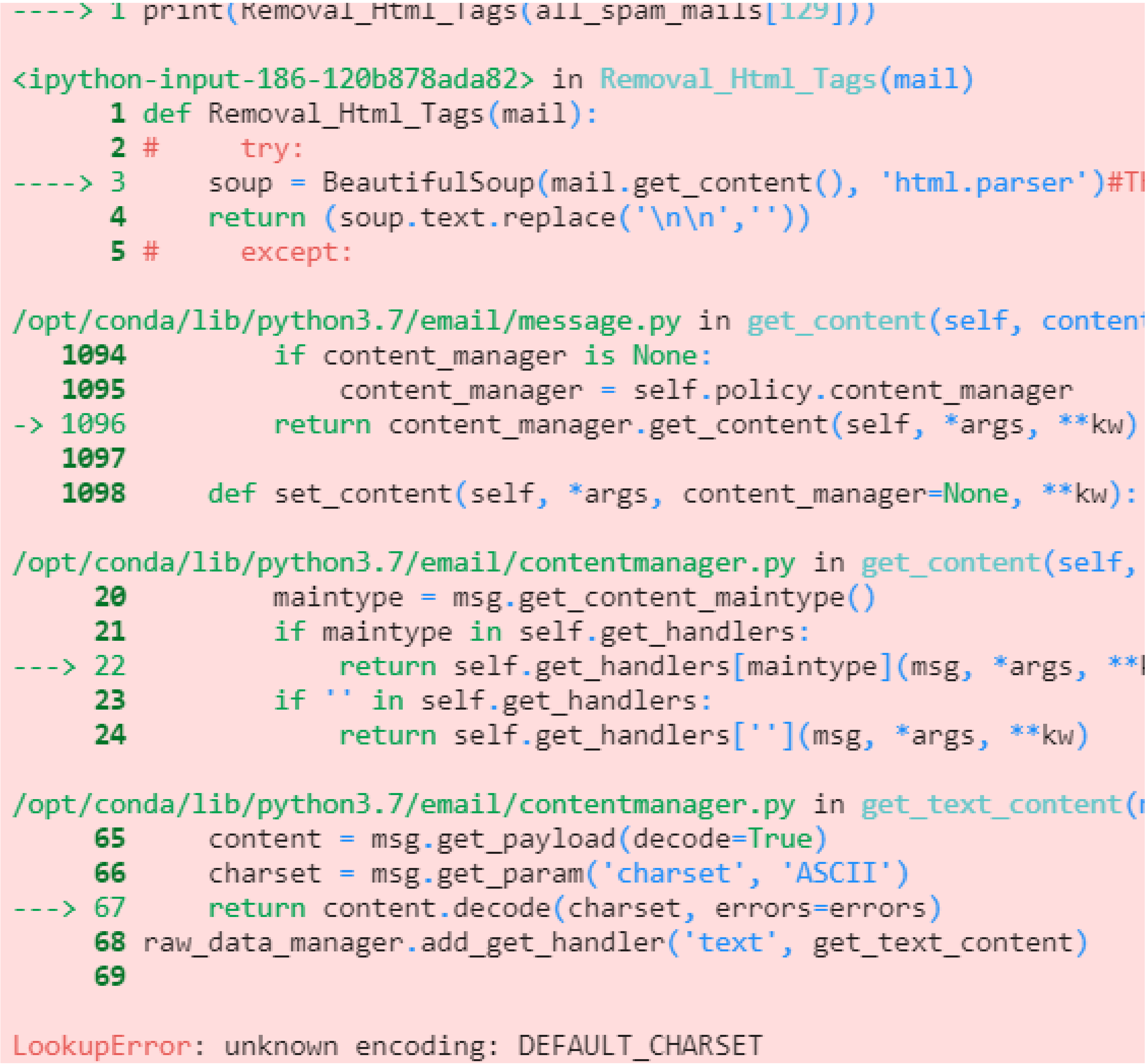
   I had no clue of how to remove the html tags from the text the I came across this and it was using BeautifulSoap.For this I need to install beautifulsoap by doing a pip install beautifulsoap4.A BeautifulSoap is a Python library for getting data out of HTML, XML, and other markup languages. Say you've found some webpages that display data relevant to your research, such as date or address information, but that do not provide any way of downloading the data directly. Beautiful Soup helps you pull particular content from a webpage, remove the HTML markup, and save the information. It is a tool for web scraping that helps you clean up and parse the documents you have pulled down from the web.

   Then I faced another issue whenever I am trying to print the text after removal of html I got an error



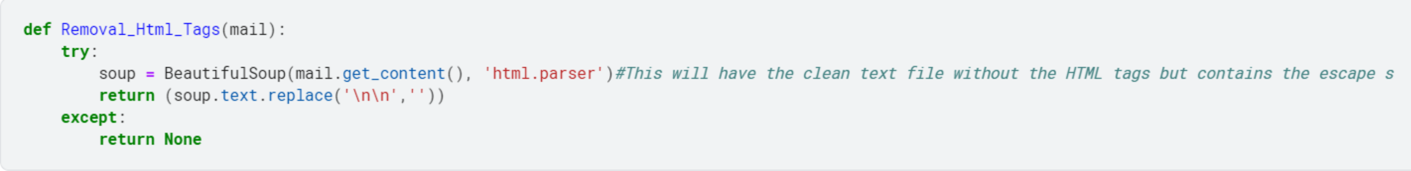
```python
def Removal_Html_Tags(mail):
    soup = BeautifulSoup(mail.get_content(), 'html.parser')#This will have the clean text file without the HTML tags but contains the escape sequence
    return (soup.text.replace('\n\n',''))
```

```python
print(Removal_Html_Tags(all_spam_mails[129]))
```

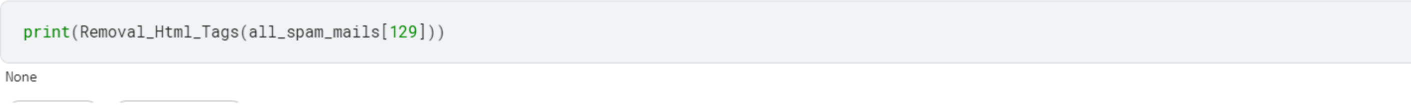


```
----> 1 print(Removal_Html_Tags(all_spam_mails[129]))

<ipython-input-186-120b878ada82> in Removal_Html_Tags(mail)
      1 def Removal_Html_Tags(mail):
      2 #     try:
----> 3     soup = BeautifulSoup(mail.get_content(), 'html.parser')#T
      4     return (soup.text.replace('\n\n',''))
      5 #     except:

/opt/conda/lib/python3.7/email/message.py in get_content(self, conten
   1094         if content_manager is None:
   1095             content_manager = self.policy.content_manager
-> 1096         return content_manager.get_content(self, *args, **kw)
   1097 
   1098     def set_content(self, *args, content_manager=None, **kw):

/opt/conda/lib/python3.7/email/contentmanager.py in get_content(self,
     20         maintype = msg.get_content_maintype()
     21         if maintype in self.get_handlers:
---> 22             return self.get_handlers[maintype](msg, *args, **
     23         if '' in self.get_handlers:
     24             return self.get_handlers[''](msg, *args, **kw)

/opt/conda/lib/python3.7/email/contentmanager.py in get_text_content(
     65     content = msg.get_payload(decode=True)
     66     charset = msg.get_param('charset', 'ASCII')
---> 67     return content.decode(charset, errors=errors)
     68 raw_data_manager.add_get_handler('text', get_text_content)
     69 

LookupError: unknown encoding: DEFAULT_CHARSET
```

I was scratching my head then I found out that for all the multipart contnet type mails this error is throwing up so I decided to keep exception handling here if the mails are of type multipart I would return None otherwise I would Return the value in which the HTML tags are removed



```
def Removal_Html_Tags(mail):
    try:
        soup = BeautifulSoup(mail.get_content(), 'html.parser')#This will have the clean text file without the HTML tags but contains the escape s
        return (soup.text.replace('\n\n',''))
    except:
        return None
```



```
print(Removal_Html_Tags(all_spam_mails[129]))
```

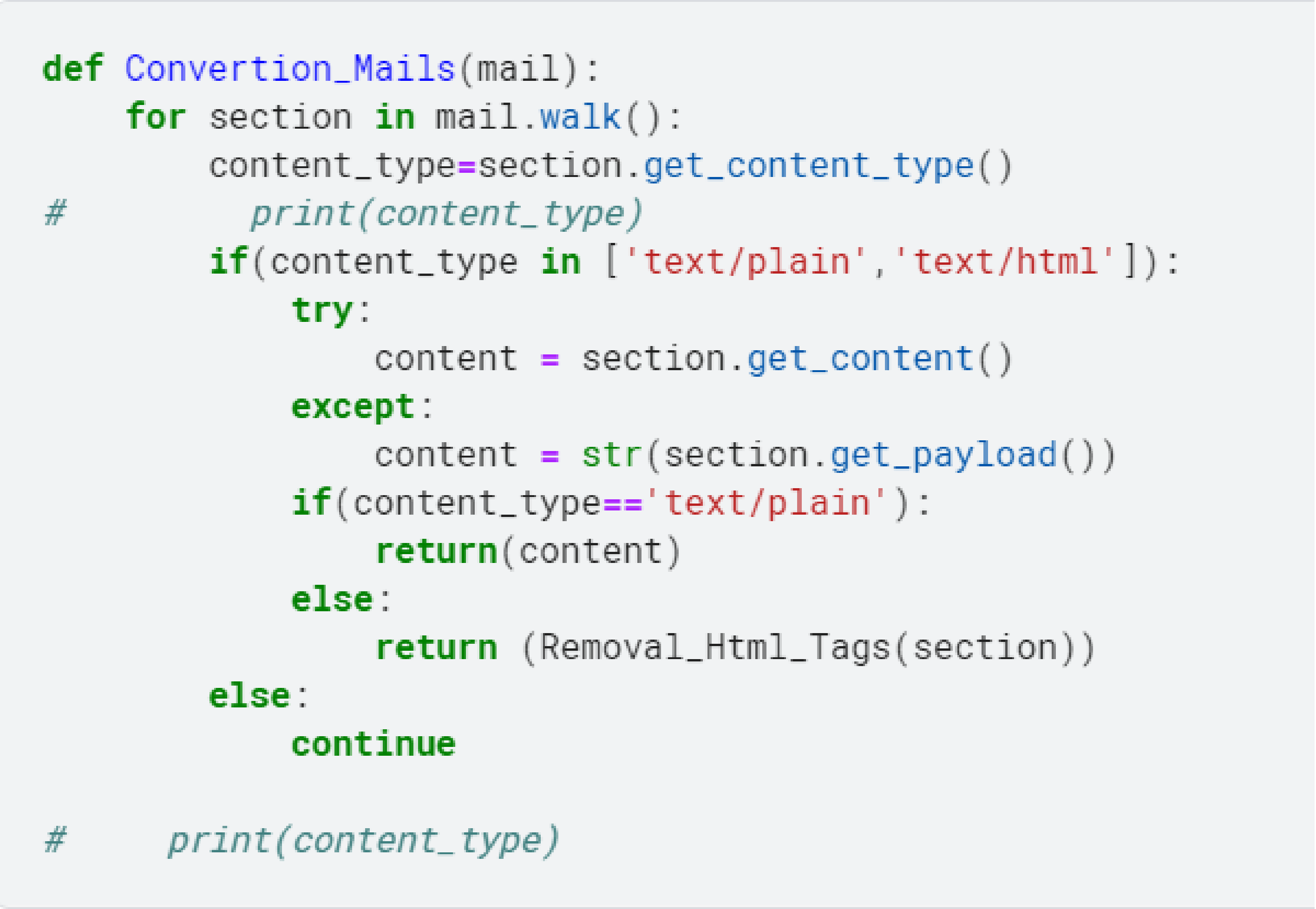
```
None
```

2. Convertion of Mail Contents to text

   The next task is to convert the mails into text and store it in an array for spam and non-spam respectivily

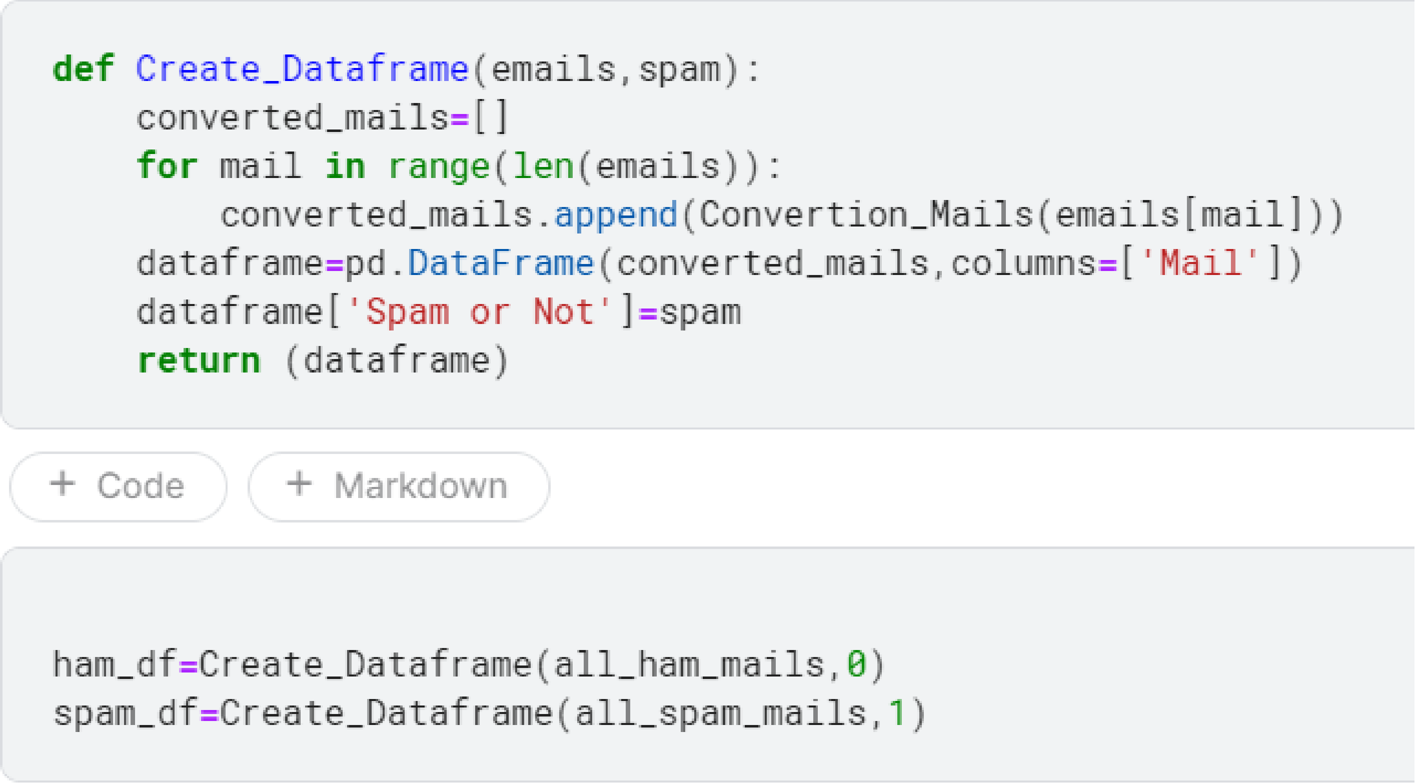
   The Function Conversion_Mail does the thing which I want that is it converts all the text/plain and the html/text to texts.

   The mail.walk() function tends to iterate through the different sections of mail such as To:,From:,Subject:,Contnet_type ect.



```
def Convertion_Mails(mail):
    for section in mail.walk():
        content_type=section.get_content_type()
#         print(content_type)
        if(content_type in ['text/plain','text/html']):
            try:
                content = section.get_content()
            except:
                content = str(section.get_payload())
            if(content_type=='text/plain'):
                return(content)
            else:
                return (Removal_Html_Tags(section))
        else:
            continue

#     print(content_type)
```

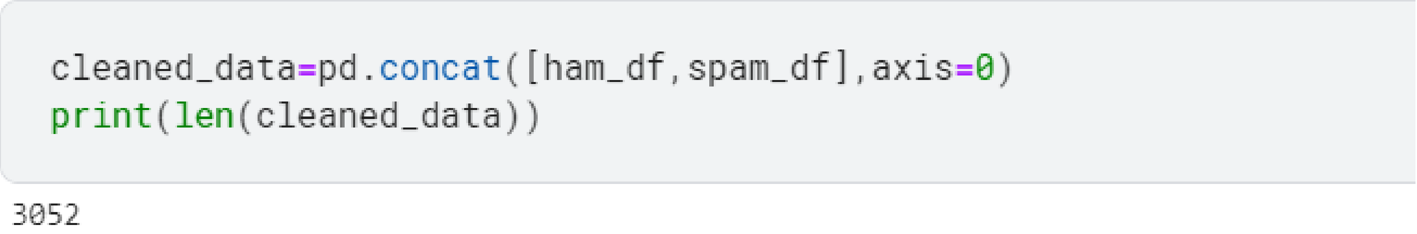
Then I have stored the converted text in an array and created a dataframe from the array which contains stored text either from non-spam or from spam.I am also adding another column to the dataframe that is the "spam or not" column which is 1 if is spam and it is non-spam if it is 0.Thus there would be two dtaframes ,one for spam mails and other for non-spam Mails, each containing converted texts and Spam or not columns.



```
def Create_Dataframe(emails,spam):
    converted_mails=[]
    for mail in range(len(emails)):
        converted_mails.append(Convertion_Mails(emails[mail]))
    dataframe=pd.DataFrame(converted_mails,columns=['Mail'])
    dataframe['Spam or Not']=spam
    return (dataframe)
```

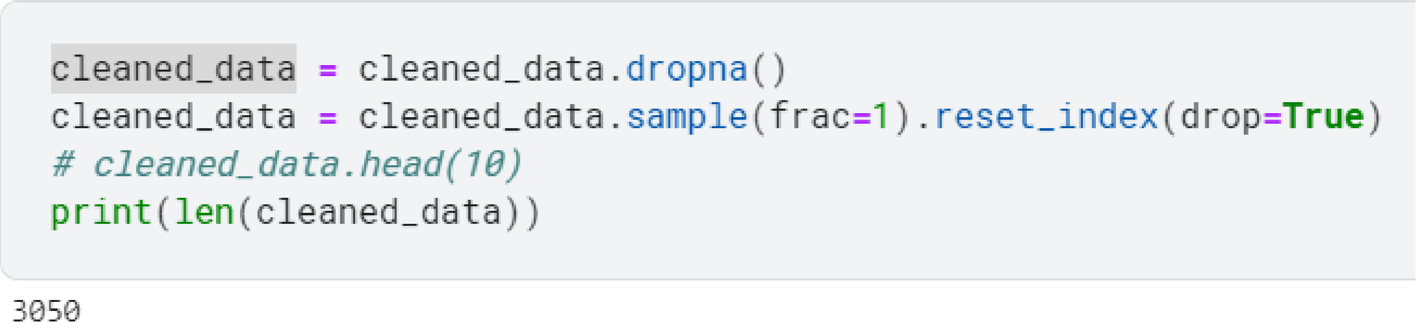
```
ham_df=Create_Dataframe(all_ham_mails,0)
spam_df=Create_Dataframe(all_spam_mails,1)
```

3. Checking for Null Values and Dropping them
   I have chekced if there are any null values if so I am dropping the null values directly.When I did 2 values had null values and hence I dropped them



```
cleaned_data=pd.concat([ham_df,spam_df],axis=0)
print(len(cleaned_data))
```

```
3052
```



```
cleaned_data = cleaned_data.dropna()
cleaned_data = cleaned_data.sample(frac=1).reset_index(drop=True)
# cleaned_data.head(10)
print(len(cleaned_data))
```

```
3050
```

4. Removing the escape sequences and puncuations like /n ,:; etc from cleaned data
   The above cleaned data in the dataframe had escape sequences and punctuations in them so removing them using the regular expression

```
for i in range(len(cleaned_data)):
    cleaned_data['Mail'][i] = re.sub(r"[^a-zA-Z0-9]+", ' ', cleaned_data['Mail'][i])
(cleaned_data)
```

Model Selection and Experimentation

I have tried using different models

- SVM (Support Vector Machine) Classifier
- Naive Bayes Classifier

- Random Forest Classifier

Not only different models have been experiemnted but also I have experimented with different vectorizers like

- CountVectorizer
- TfidfVectorizer

I am going to find the accuracy of each of the combination and select the best model.So there would be a total of 8 models for my experiemts and selecting the best one.

1. ## Naive Bayes Classifier

   ### Refrences:

   - https://www.reddit.com/r/MachineLearning/comments/2uhhbh/difference_between_binomial_multinomial_and/co8xxls/
   - https://scikit-learn.org/stable/modules/naive_bayes.html

   Naive Bayes Classifier is one of the successful method for text classification. We have following Naive Bayes Classifiers:

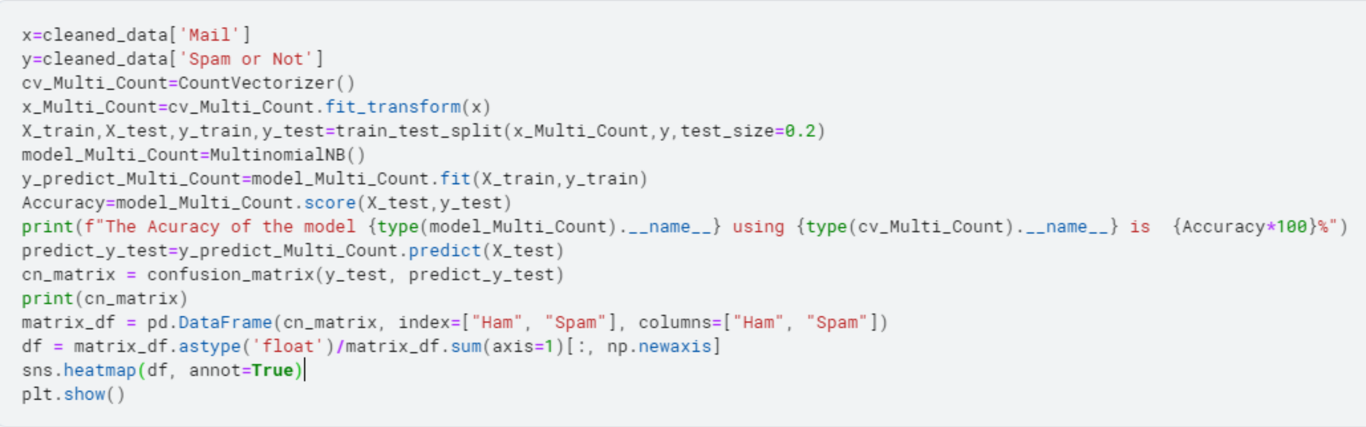
   - Bernoulli Naive Bayes:
     The binomial model is useful if your feature vectors are binary (i.e., 0s and 1s). One application would be text classification with a bag of words model where the 0s 1s are "word occurs in the document" and "word does not occur in the document"

   - Multinomial Naive Bayes:
     The multinomial naive Bayes model is typically used for discrete counts. E.g., if we have a text classification problem, we can take the idea of bernoulli trials one step further and instead of "word occurs in the document" we have "count how often word occurs in the document", you can think of it as "number of times outcome number x_i is observed over the n trials"

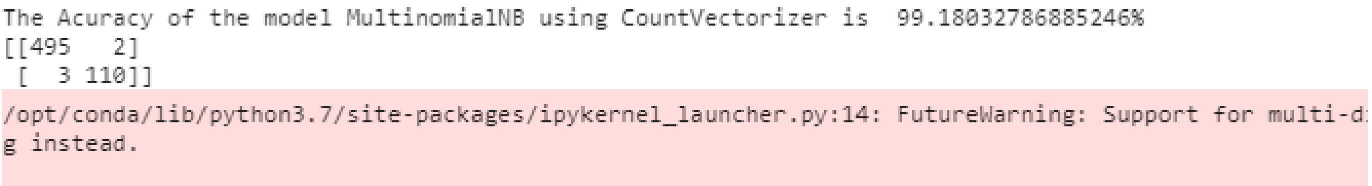
   - Gaussian Naive Bayes:
     Here, we assume that the features follow a normal distribution. Instead of discrete counts, we have continuous features (e.g., the popular Iris dataset where the features are sepal width, petal width, sepal length, petal length).

     Looking at the above description, we can conclude that only Multinomial Naive Bayes is useful in our case. But just for the confirmation, we will also try Gaussian Naive Bayes.

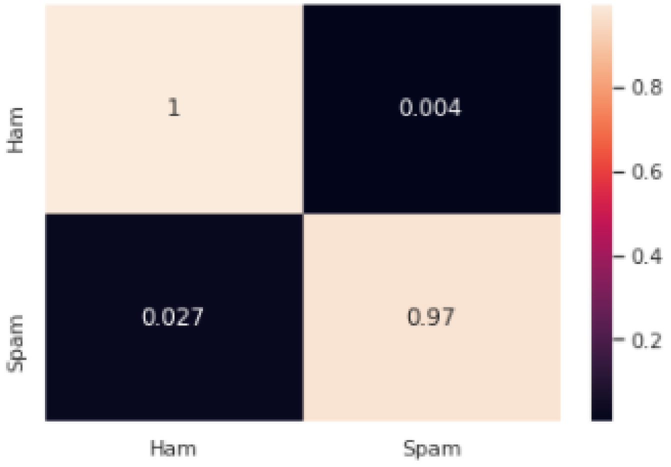
   - ## Multinomial Naive Bayes with Countvectorizer:



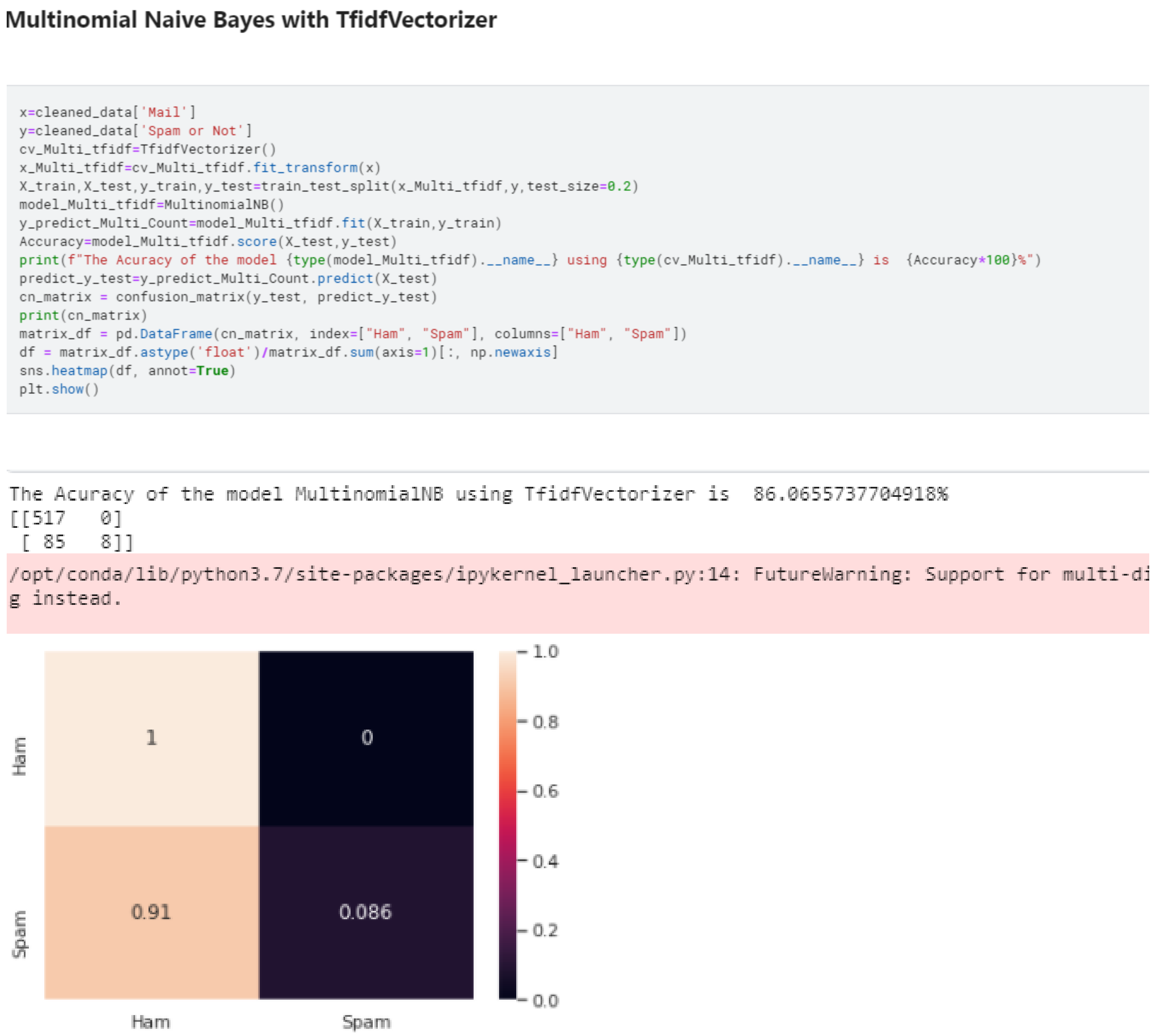
```
x=cleaned_data['Mail']
y=cleaned_data['Spam or Not']
cv_Multi_Count=CountVectorizer()
x_Multi_Count=cv_Multi_Count.fit_transform(x)
X_train,X_test,y_train,y_test=train_test_split(x_Multi_Count,y,test_size=0.2)
model_Multi_Count=MultinomialNB()
y_predict_Multi_Count=model_Multi_Count.fit(X_train,y_train)
Accuracy=model_Multi_Count.score(X_test,y_test)
print(f"The Acuracy of the model {type(model_Multi_Count).__name__} using {type(cv_Multi_Count).__name__} is  {Accuracy*100}%")
predict_y_test=y_predict_Multi_Count.predict(X_test)
cn_matrix = confusion_matrix(y_test, predict_y_test)
print(cn_matrix)
matrix_df = pd.DataFrame(cn_matrix, index=["Ham", "Spam"], columns=["Ham", "Spam"])
df = matrix_df.astype('float')/matrix_df.sum(axis=1)[:, np.newaxis]
sns.heatmap(df, annot=True)
plt.show()
```



```
The Acuracy of the model MultinomialNB using CountVectorizer is  99.18032786885246%
[[495   2]
 [  3 110]]
/opt/conda/lib/python3.7/site-packages/ipykernel_launcher.py:14: FutureWarning: Support for multi-d
g instead.
```



Multinomial Naive Bayes with TFIDFVectorizer:

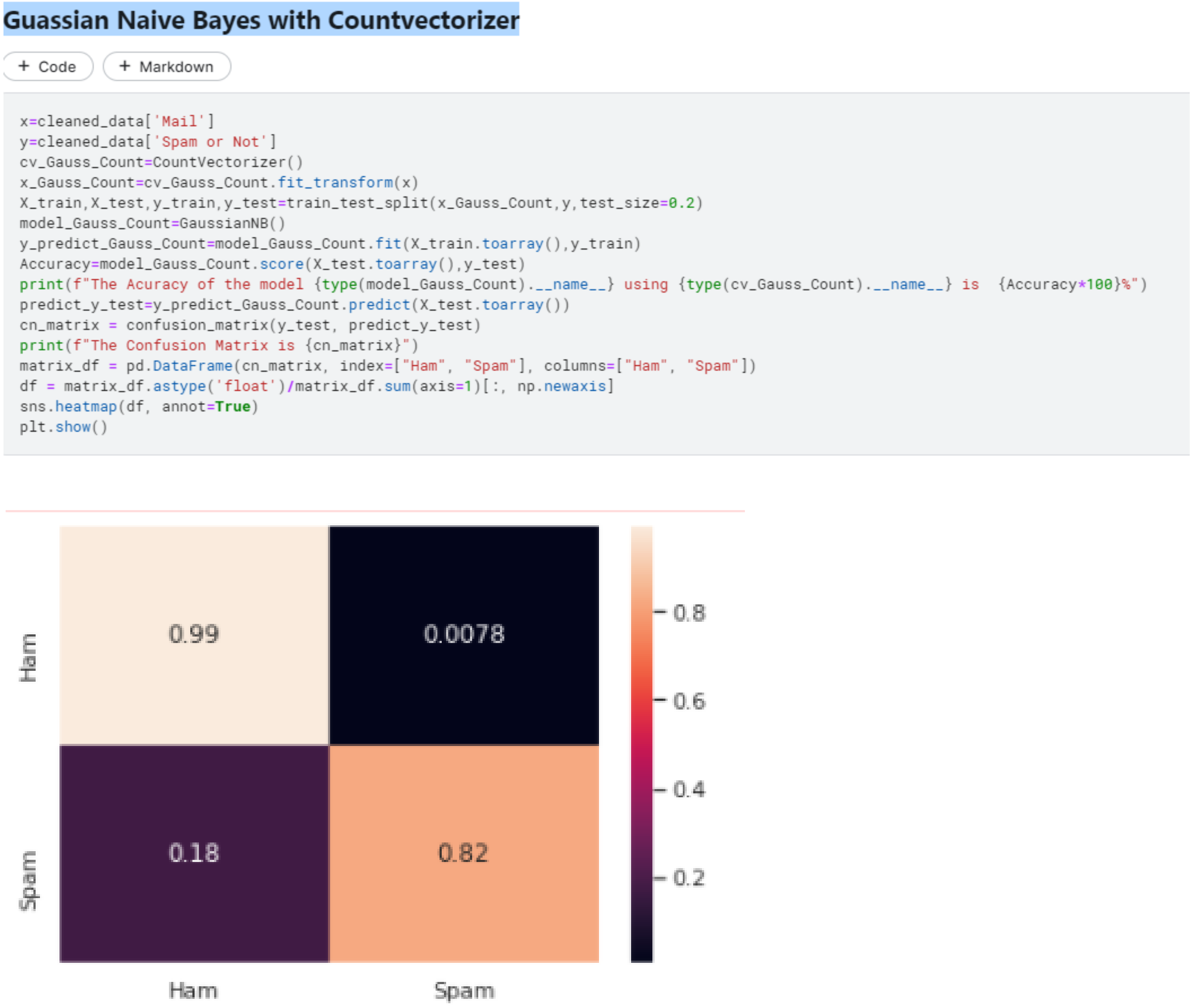


Multinomial Naive Bayes with TfidfVectorizer

```
x=cleaned_data['Mail']
y=cleaned_data['Spam or Not']
cv_Multi_tfidf=TfidfVectorizer()
x_Multi_tfidf=cv_Multi_tfidf.fit_transform(x)
X_train,X_test,y_train,y_test=train_test_split(x_Multi_tfidf,y,test_size=0.2)
model_Multi_tfidf=MultinomialNB()
y_predict_Multi_Count=model_Multi_tfidf.fit(X_train,y_train)
Accuracy=model_Multi_tfidf.score(X_test,y_test)
print(f"The Acuracy of the model {type(model_Multi_tfidf).__name__} using {type(cv_Multi_tfidf).__name__} is  {Accuracy*100}%")
predict_y_test=y_predict_Multi_Count.predict(X_test)
cn_matrix = confusion_matrix(y_test, predict_y_test)
print(cn_matrix)
matrix_df = pd.DataFrame(cn_matrix, index=["Ham", "Spam"], columns=["Ham", "Spam"])
df = matrix_df.astype('float')/matrix_df.sum(axis=1)[:, np.newaxis]
sns.heatmap(df, annot=True)
plt.show()
```

```
The Acuracy of the model MultinomialNB using TfidfVectorizer is  86.0655737704918%
[[517   0]
 [ 85   8]]
/opt/conda/lib/python3.7/site-packages/ipykernel_launcher.py:14: FutureWarning: Support for multi-di
g instead.
```

- ## Guassian Naive Bayes with Countvectorizer

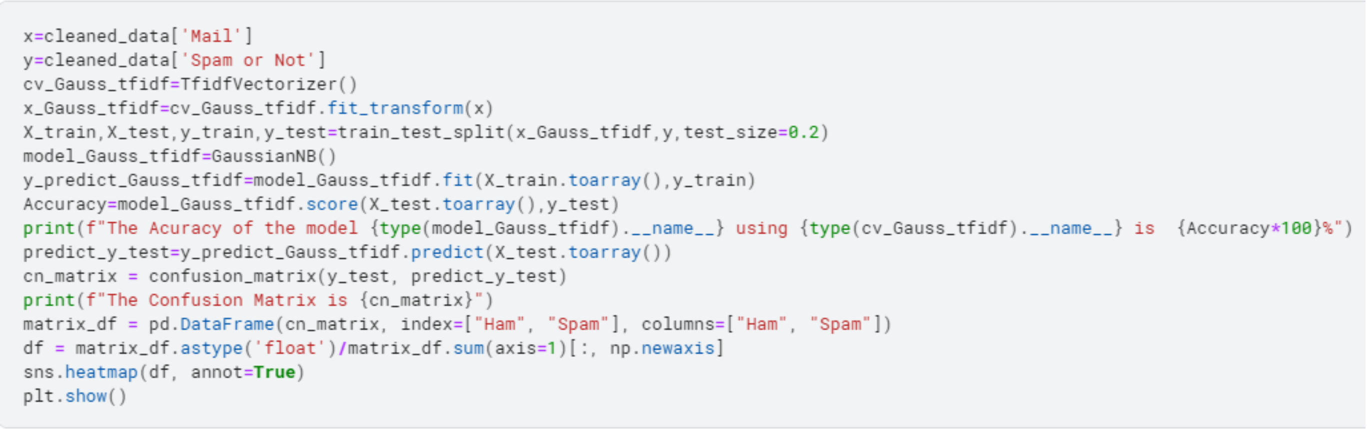


Guassian Naive Bayes with Countvectorizer

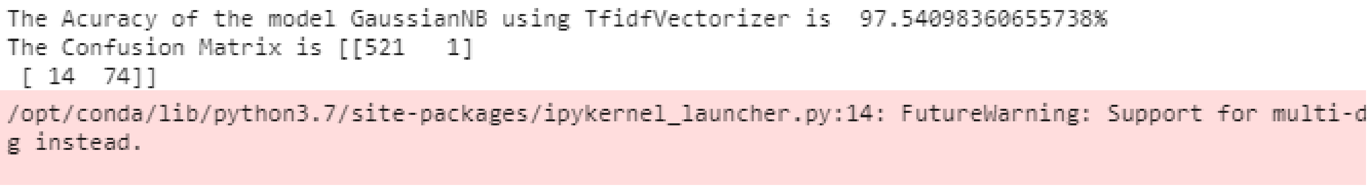
```
x=cleaned_data['Mail']
y=cleaned_data['Spam or Not']
cv_Gauss_Count=CountVectorizer()
x_Gauss_Count=cv_Gauss_Count.fit_transform(x)
X_train,X_test,y_train,y_test=train_test_split(x_Gauss_Count,y,test_size=0.2)
model_Gauss_Count=GaussianNB()
y_predict_Gauss_Count=model_Gauss_Count.fit(X_train.toarray(),y_train)
Accuracy=model_Gauss_Count.score(X_test.toarray(),y_test)
print(f"The Acuracy of the model {type(model_Gauss_Count).__name__} using {type(cv_Gauss_Count).__name__} is  {Accuracy*100}%")
predict_y_test=y_predict_Gauss_Count.predict(X_test.toarray())
cn_matrix = confusion_matrix(y_test, predict_y_test)
print(f"The Confusion Matrix is {cn_matrix}")
matrix_df = pd.DataFrame(cn_matrix, index=["Ham", "Spam"], columns=["Ham", "Spam"])
df = matrix_df.astype('float')/matrix_df.sum(axis=1)[:, np.newaxis]
sns.heatmap(df, annot=True)
plt.show()
```

- ## Guassian Naive Bayes with TFIDFVectorizer

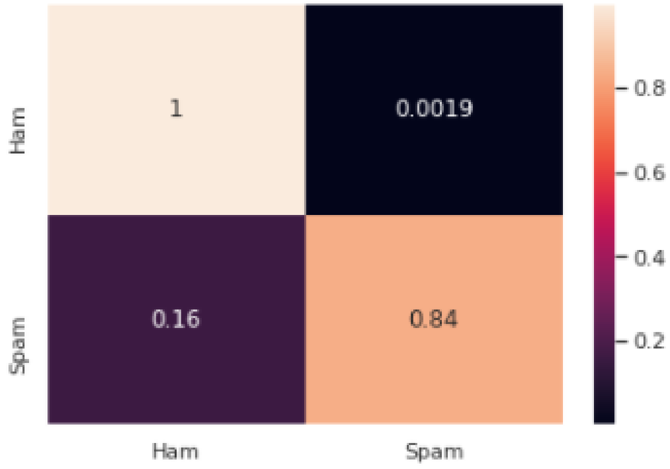
Guassian Naive Bayes with TfidfVectorizer



```
x=cleaned_data['Mail']
y=cleaned_data['Spam or Not']
cv_Gauss_tfidf=TfidfVectorizer()
x_Gauss_tfidf=cv_Gauss_tfidf.fit_transform(x)
X_train,X_test,y_train,y_test=train_test_split(x_Gauss_tfidf,y,test_size=0.2)
model_Gauss_tfidf=GaussianNB()
y_predict_Gauss_tfidf=model_Gauss_tfidf.fit(X_train.toarray(),y_train)
Accuracy=model_Gauss_tfidf.score(X_test.toarray(),y_test)
print(f"The Acuracy of the model {type(model_Gauss_tfidf).__name__} using {type(cv_Gauss_tfidf).__name__} is  {Accuracy*100}%")
predict_y_test=y_predict_Gauss_tfidf.predict(X_test.toarray())
cn_matrix = confusion_matrix(y_test, predict_y_test)
print(f"The Confusion Matrix is {cn_matrix}")
matrix_df = pd.DataFrame(cn_matrix, index=["Ham", "Spam"], columns=["Ham", "Spam"])
df = matrix_df.astype('float')/matrix_df.sum(axis=1)[:, np.newaxis]
sns.heatmap(df, annot=True)
plt.show()
```



```
The Acuracy of the model GaussianNB using TfidfVectorizer is  97.54098360655738%
The Confusion Matrix is [[521   1]
 [ 14  74]]
/opt/conda/lib/python3.7/site-packages/ipykernel_launcher.py:14: FutureWarning: Support for multi-d
g instead.
```



Looking at the results, we can confirm - for text classification, Multinomial Naive Bayes is better than Gaussian Naive Bayes.

2. SVM Classifier

References

- https://monkeylearn.com/text-classification-support-vector-machines-svm/
- https://scikit-learn.org/stable/modules/svm.html
- https://www.datacamp.com/community/tutorials/svm-classification-scikit-learn-python

There are many different algorithms we can choose from when doing text classification with machine learning. One of those is Support Vector Machines (or SVM).

Support vector machines is an algorithm that determines the best decision boundary between vectors that belong to a given group (or category) and vectors that do not belong to it. That’s it. It can be applied to any kind of vectors which encode any kind of data. This means that in order to leverage the power of svm text classification, texts have to be transformed into vectors.

As we have already transformed our texts into vectors, we can directly start training and predicting.

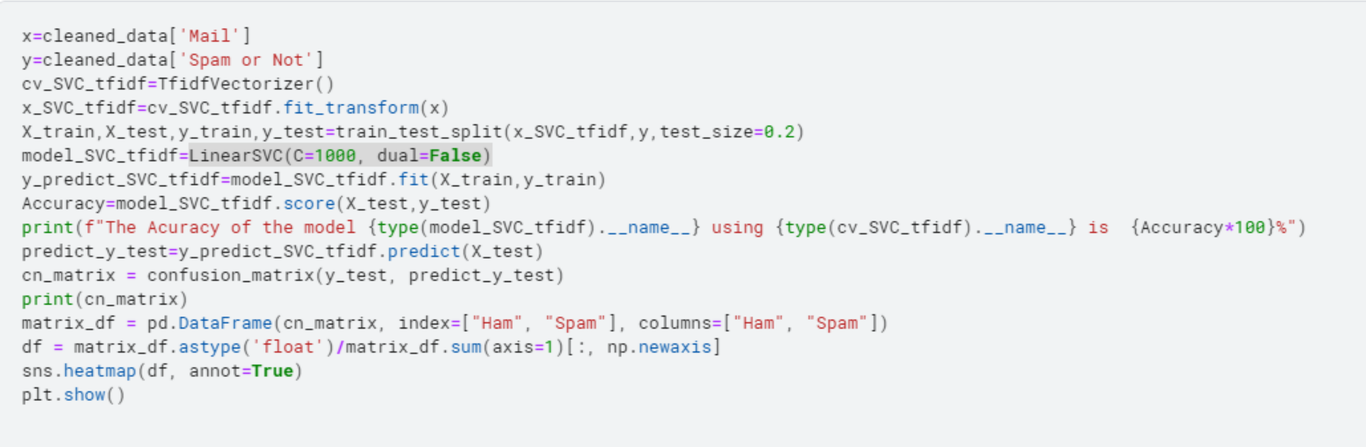
According to the documentation of 'scikit-learn': For large datasets consider using sklearn.svm.LinearSVC and not sklearn.svm.SVC So, we will go with sklearn.svm.LinearSVC

Regularization:

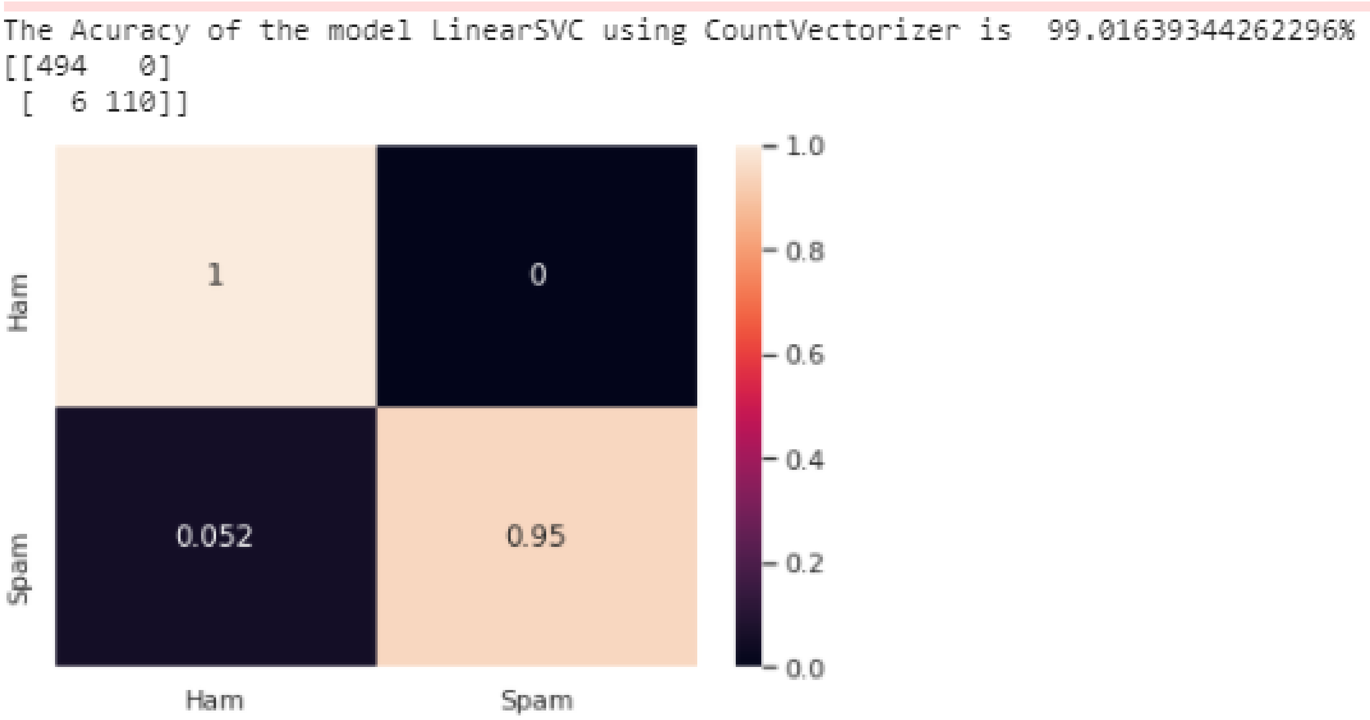
Regularization parameter in python's Scikit-learn C parameter used to maintain regularization. Here C is the penalty parameter, which represents misclassification or error term. The misclassification or error term tells the SVM optimization how much error is bearable. This is how you can control the trade-off between decision boundary and misclassification term. A smaller value of C creates a small-margin hyperplane and a larger value of C creates a larger-margin hyperplane.For this I am trying with C=1000

- ## LinearSVC(C=1000) with Countvectorizer

LinearSVC(C=1000) with TfidfVectorizer



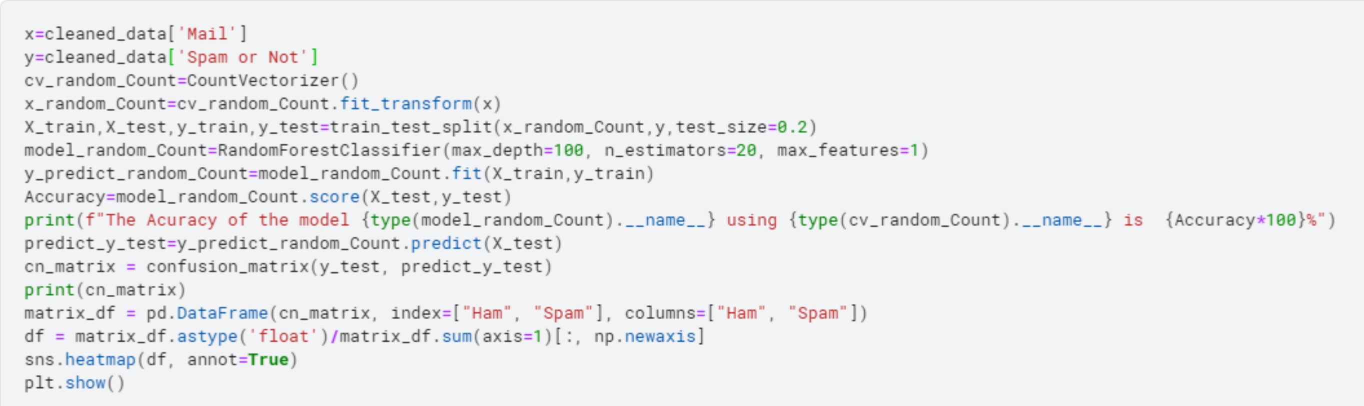
```
x=cleaned_data['Mail']
y=cleaned_data['Spam or Not']
cv_SVC_tfidf=TfidfVectorizer()
x_SVC_tfidf=cv_SVC_tfidf.fit_transform(x)
X_train,X_test,y_train,y_test=train_test_split(x_SVC_tfidf,y,test_size=0.2)
model_SVC_tfidf=LinearSVC(C=1000, dual=False)
y_predict_SVC_tfidf=model_SVC_tfidf.fit(X_train,y_train)
Accuracy=model_SVC_tfidf.score(X_test,y_test)
print(f"The Acuracy of the model {type(model_SVC_tfidf).__name__} using {type(cv_SVC_tfidf).__name__} is  {Accuracy*100}%")
predict_y_test=y_predict_SVC_tfidf.predict(X_test)
cn_matrix = confusion_matrix(y_test, predict_y_test)
print(cn_matrix)
matrix_df = pd.DataFrame(cn_matrix, index=["Ham", "Spam"], columns=["Ham", "Spam"])
df = matrix_df.astype('float')/matrix_df.sum(axis=1)[:, np.newaxis]
sns.heatmap(df, annot=True)
plt.show()
```



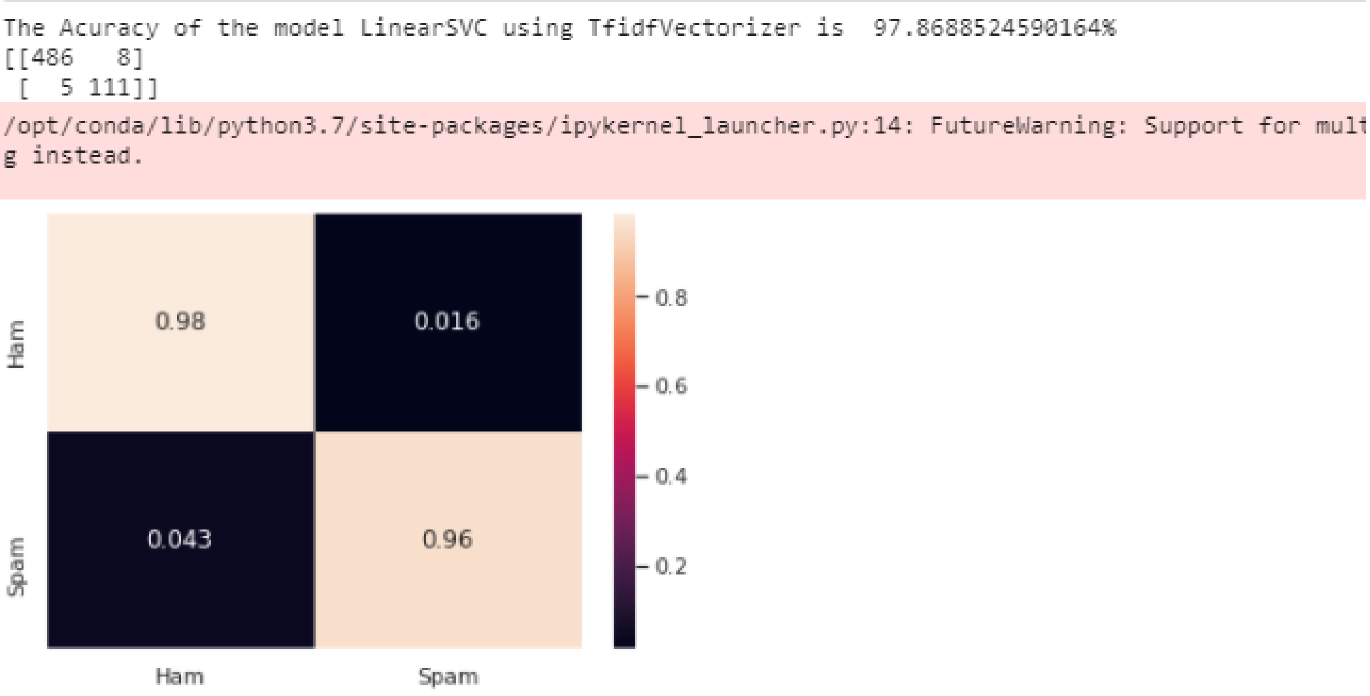
```
The Acuracy of the model LinearSVC using CountVectorizer is  99.01639344262296%
[[494   0]
 [  6 110]]
```

- [LinearSVC(C=1000) with TFIDFVectorizer]

RandomForrest with Countvectorizer



```
x=cleaned_data['Mail']
y=cleaned_data['Spam or Not']
cv_random_Count=CountVectorizer()
x_random_Count=cv_random_Count.fit_transform(x)
X_train,X_test,y_train,y_test=train_test_split(x_random_Count,y,test_size=0.2)
model_random_Count=RandomForestClassifier(max_depth=100, n_estimators=20, max_features=1)
y_predict_random_Count=model_random_Count.fit(X_train,y_train)
Accuracy=model_random_Count.score(X_test,y_test)
print(f"The Acuracy of the model {type(model_random_Count).__name__} using {type(cv_random_Count).__name__} is  {Accuracy*100}%")
predict_y_test=y_predict_random_Count.predict(X_test)
cn_matrix = confusion_matrix(y_test, predict_y_test)
print(cn_matrix)
matrix_df = pd.DataFrame(cn_matrix, index=["Ham", "Spam"], columns=["Ham", "Spam"])
df = matrix_df.astype('float')/matrix_df.sum(axis=1)[:, np.newaxis]
sns.heatmap(df, annot=True)
plt.show()
```



```
The Acuracy of the model LinearSVC using TfidfVectorizer is  97.8688524590164%
[[486   8]
 [  5 111]]
/opt/conda/lib/python3.7/site-packages/ipykernel_launcher.py:14: FutureWarning: Support for mult
g instead.
```

3. RandomForrest Classifier

References

- https://scikit-learn.org/stable/auto_examples/classification/plot_classifier_comparison.html
- https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html#sklearn.ensemble.RandomForestClassifier
- https://en.wikipedia.org/wiki/Random_forest

A random forest is a meta estimator that fits a number of decision tree classifiers on various sub-samples of the dataset and uses averaging to improve the predictive accuracy and control over-fitting.

In other words, Random forests or random decision forests are an ensemble learning method for classification, regression and other tasks that operate by constructing a multitude of decision trees at training time and outputting the class that is the mode of the classes (classification) or mean prediction (regression) of the individual trees. Random decision forests correct for decision trees' habit of overfitting to their training set.

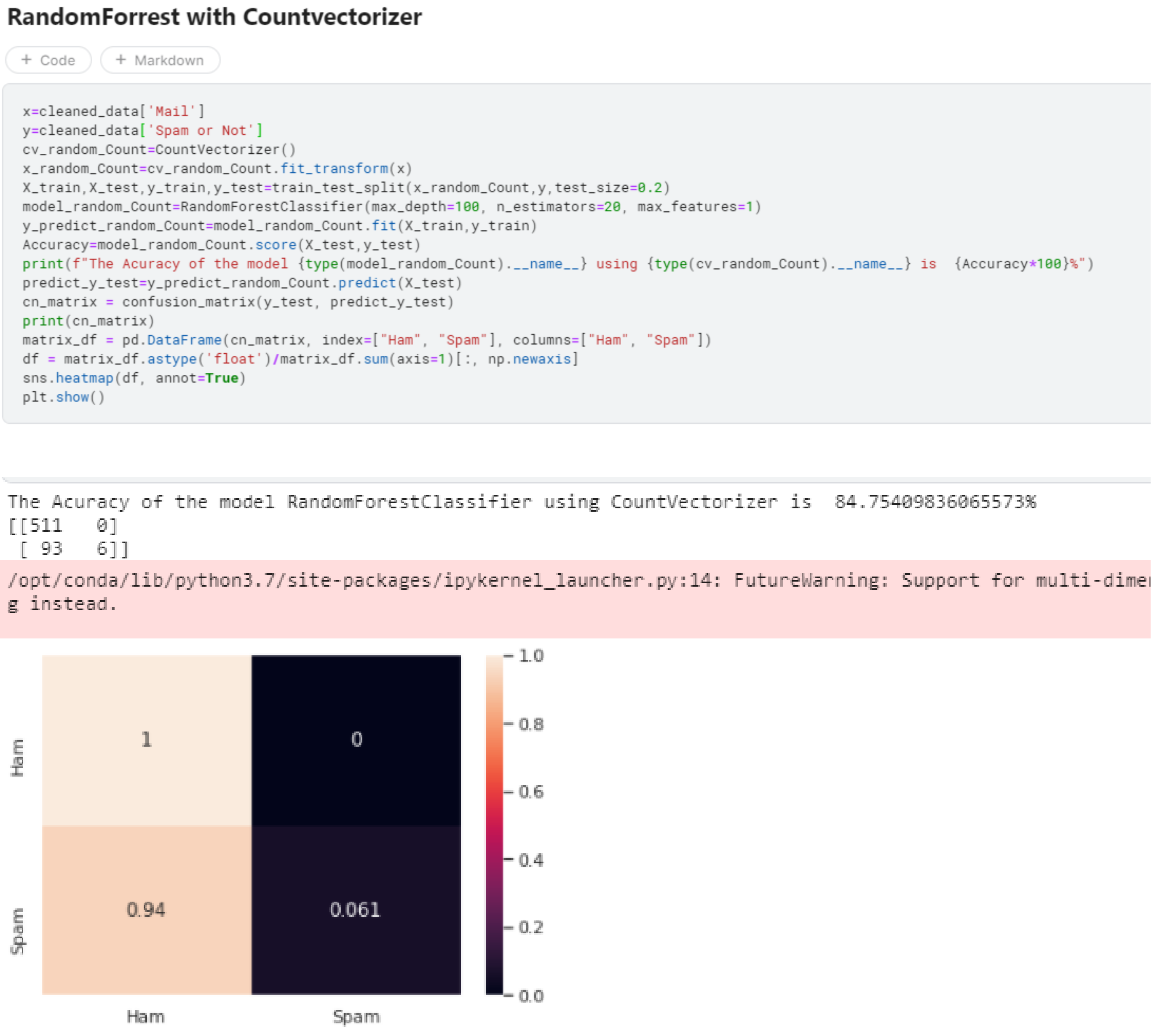
We will set the maximum depth of tree to 100 for better results.

I used this value after doing some testing on the data. (<100 : Low Accuracy, >100 : Large computing time)

The number of trees in the forest (n_estimators) is considerd as one of the most important factor for getting better accuracy.

We will use n_estimators =20

- ### RandomForrest with CountVectorizer



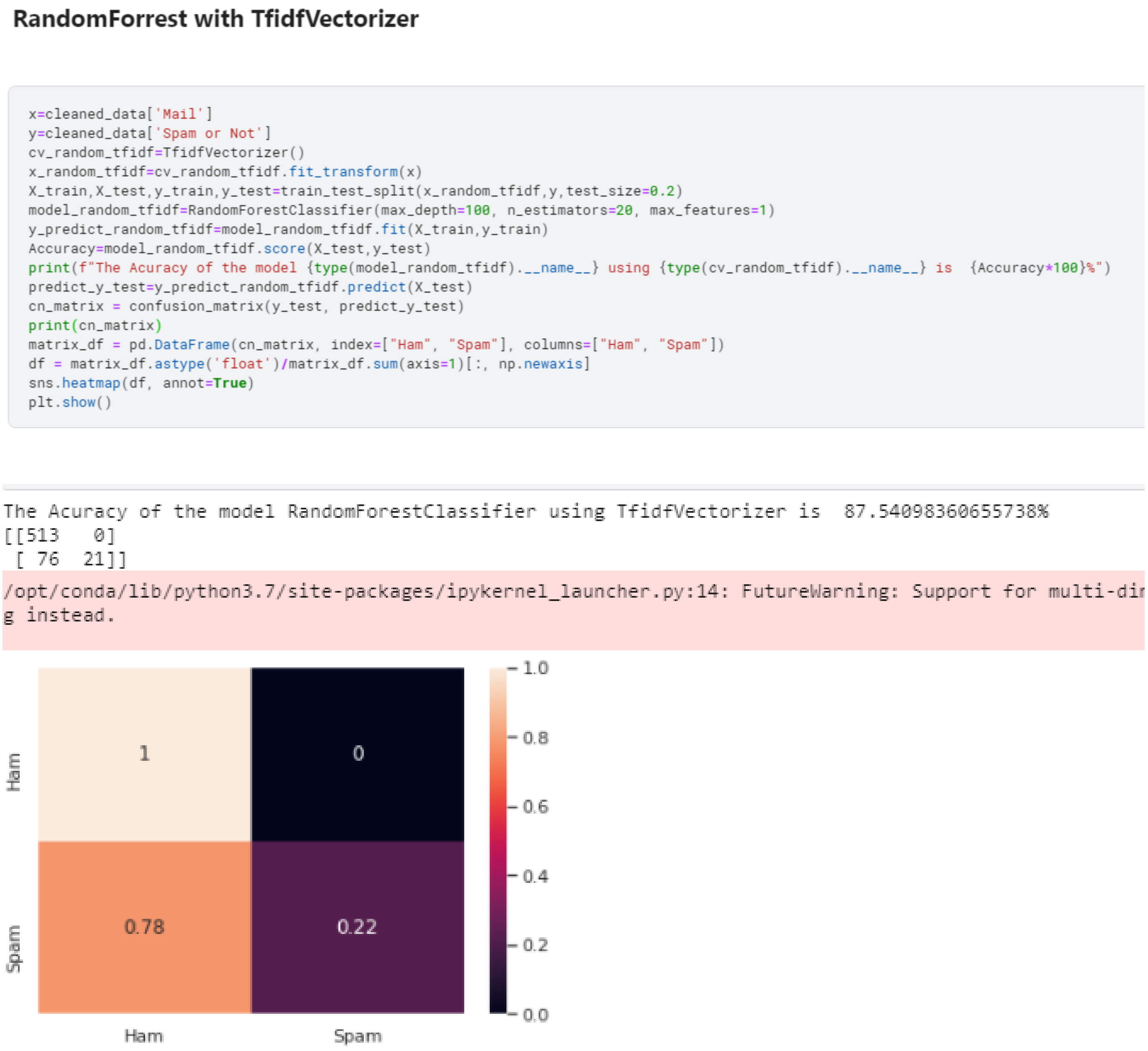
RandomForrest with Countvectorizer

```
x=cleaned_data['Mail']
y=cleaned_data['Spam or Not']
cv_random_Count=CountVectorizer()
x_random_Count=cv_random_Count.fit_transform(x)
X_train,X_test,y_train,y_test=train_test_split(x_random_Count,y,test_size=0.2)
model_random_Count=RandomForestClassifier(max_depth=100, n_estimators=20, max_features=1)
y_predict_random_Count=model_random_Count.fit(X_train,y_train)
Accuracy=model_random_Count.score(X_test,y_test)
print(f"The Acuracy of the model {type(model_random_Count).__name__} using {type(cv_random_Count).__name__} is  {Accuracy*100}%")
predict_y_test=y_predict_random_Count.predict(X_test)
cn_matrix = confusion_matrix(y_test, predict_y_test)
print(cn_matrix)
matrix_df = pd.DataFrame(cn_matrix, index=["Ham", "Spam"], columns=["Ham", "Spam"])
df = matrix_df.astype('float')/matrix_df.sum(axis=1)[:, np.newaxis]
sns.heatmap(df, annot=True)
plt.show()
```

```
The Acuracy of the model RandomForestClassifier using CountVectorizer is  84.75409836065573%
[[511   0]
 [ 93   6]]
```

```
/opt/conda/lib/python3.7/site-packages/ipykernel_launcher.py:14: FutureWarning: Support for multi-dimer
g instead.
```

- [RandomForrest with TFIDFVectorizer]



RandomForrest with TfidfVectorizer

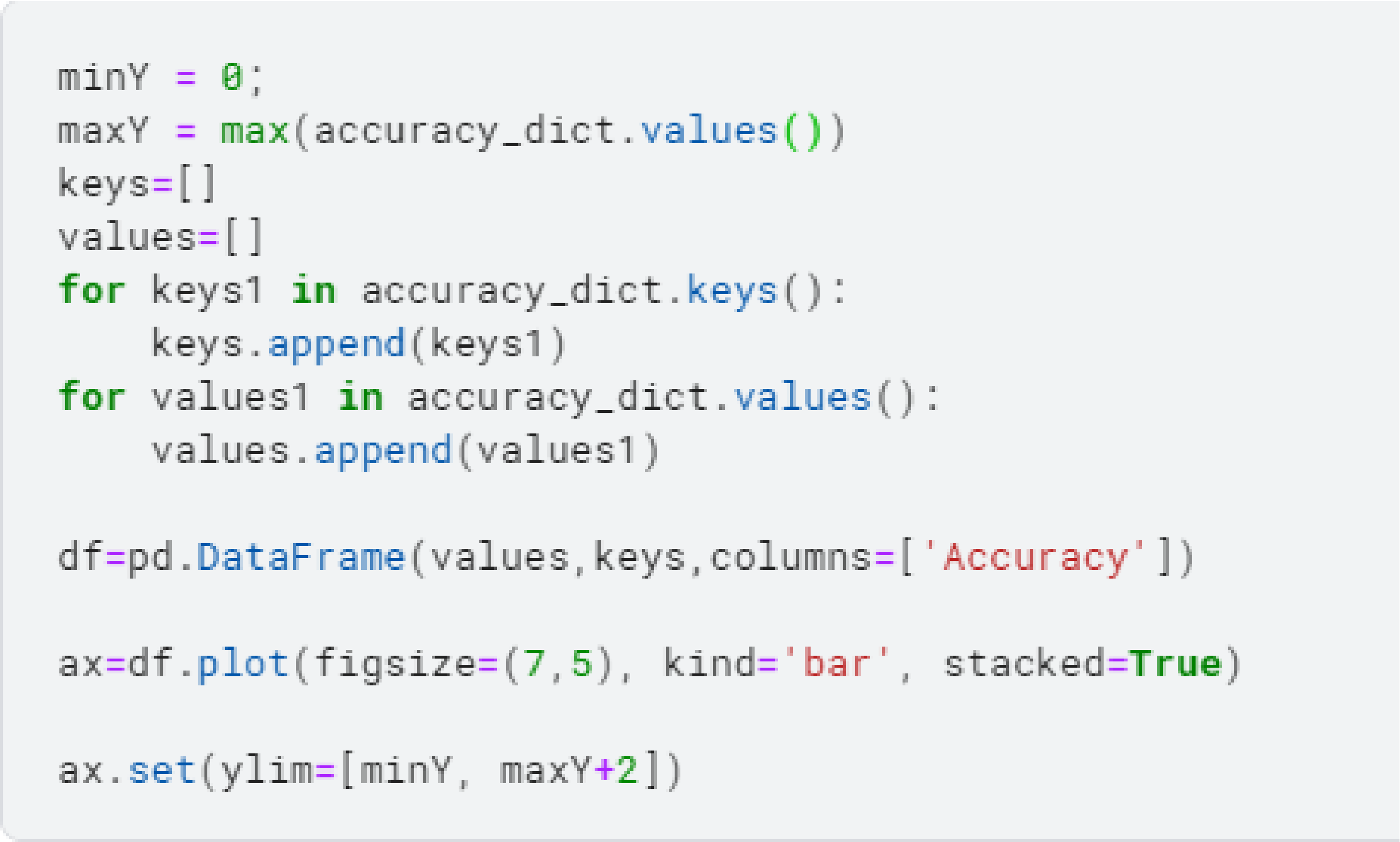
```
x=cleaned_data['Mail']
y=cleaned_data['Spam or Not']
cv_random_tfidf=TfidfVectorizer()
x_random_tfidf=cv_random_tfidf.fit_transform(x)
X_train,X_test,y_train,y_test=train_test_split(x_random_tfidf,y,test_size=0.2)
model_random_tfidf=RandomForestClassifier(max_depth=100, n_estimators=20, max_features=1)
y_predict_random_tfidf=model_random_tfidf.fit(X_train,y_train)
Accuracy=model_random_tfidf.score(X_test,y_test)
print(f"The Acuracy of the model {type(model_random_tfidf).__name__} using {type(cv_random_tfidf).__name__} is  {Accuracy*100}%")
predict_y_test=y_predict_random_tfidf.predict(X_test)
cn_matrix = confusion_matrix(y_test, predict_y_test)
print(cn_matrix)
matrix_df = pd.DataFrame(cn_matrix, index=["Ham", "Spam"], columns=["Ham", "Spam"])
df = matrix_df.astype('float')/matrix_df.sum(axis=1)[:, np.newaxis]
sns.heatmap(df, annot=True)
plt.show()
```

```
The Acuracy of the model RandomForestClassifier using TfidfVectorizer is  87.54098360655738%
[[513   0]
 [ 76  21]]
```

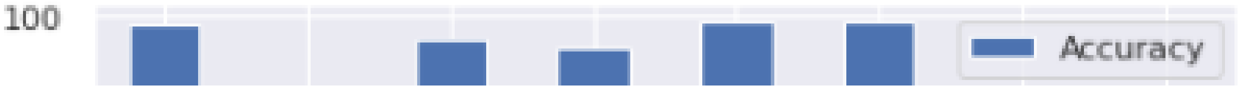
```
/opt/conda/lib/python3.7/site-packages/ipykernel_launcher.py:14: FutureWarning: Support for multi-dir
g instead.
```

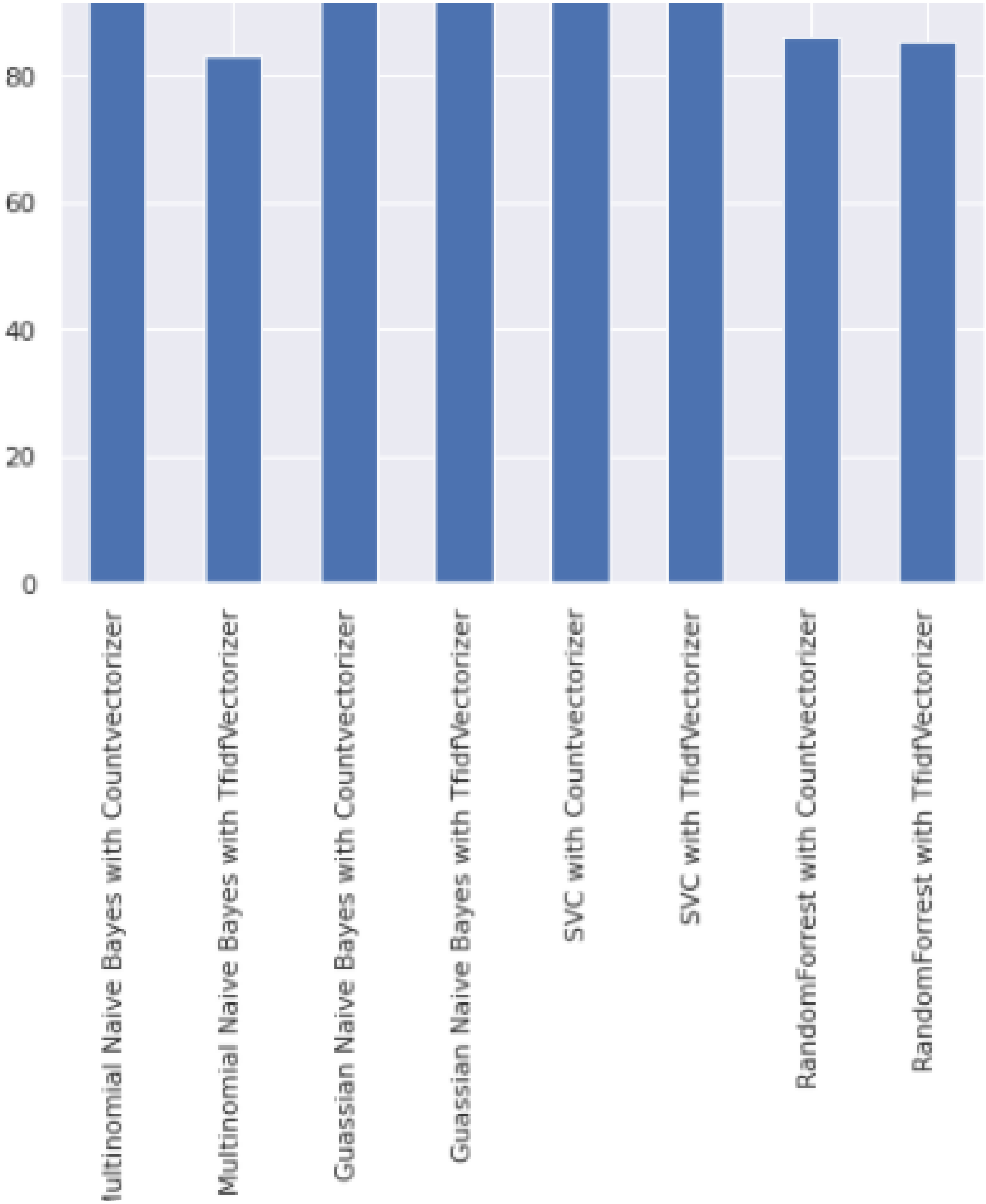
All the accuracies are good and we cannot choose the model only based on the accuracy.Thus the heatmap helps us to choose the model wisely.For RandomForrest with Countvectorizer even though the accuracy is 84% its accuarcy is bad when finding the Spam mail.From the heatmap it is seen that it predicts only 14% of the spam mail correctly.Similarly Gaussian NB with Countvectorizer also predicts only 78% of the spam mails correctly.Between SVC with Countvectorizer and Multinomial Naive Bayes with Countvectorizer need to choose one.I have chosen Multinomial Naive Bayes with Countvectorizer as of now.But seeing the heatmaps of TFIDFVectorizer MultinominalNB doesnt perform and LinearSVC performs well along with both in TFIDFVectorizer and CountVectorizer.Thus I choose SVC as my model along with the TFIDFVectorizer as my final classifier as it finds 99% of the spam mails whereas the other finds 93% of the spam mails.Thus I have chosen LinearSVC(C=1000) with TFIDFVectorizer

Efficieny Comparisons



```
minY = 0;
maxY = max(accuracy_dict.values())
keys=[]
values=[]
for keys1 in accuracy_dict.keys():
    keys.append(keys1)
for values1 in accuracy_dict.values():
    values.append(values1)

df=pd.DataFrame(values,keys,columns=['Accuracy'])

ax=df.plot(figsize=(7,5), kind='bar', stacked=True)

ax.set(ylim=[minY, maxY+2])
```




80
60
40
20
0
ultinomial Naive Bayes with Countvectorizer
Multinomial Naive Bayes with TfidfVectorizer
Guassian Naive Bayes with Countvectorizer
Guassian Naive Bayes with TfidfVectorizer
SVC with Countvectorizer
SVC with TfidfVectorizer
RandomForrest with Countvectorizer
RandomForrest with TfidfVectorizer

Eperiments Done:

- As shown above I have done Experirments in choosing the best model.So I have done experiments in choosing the classifier and as well as choosing the Vectorizer and Finally choosing the best model based on the accuracy and the heatmap

- For svm
  Experimented with the different values of C and its effect on the performance. C is the penalty parameter, which represents misclassification or error term. The misclassification or error term tells the SVM optimization how much error is bearable. This is how you can control the trade-off between decision boundary and misclassification term. A smaller value of C creates a small-margin hyperplane and a larger value of C creates a larger-margin hyperplane and selected the value of C such that the performace is boosted

- For Random Forest Classifier: Experimented through distinct values for maximum depth allowed for each tree in the forest, as well as the total number of trees in forest. Achieved a balanced model by tuning these values with multiple testing and found the best possible thresholds for our dataset and selected the appropriate value of n_estimators
- OverFit:
  After concluding the best classifier using all the hyperparameter tuning, I have gone through overfitting using maximum possible test dataset.The model was then also performing well with accuracy of 98% and predicting almost 99% of the spam mails as spam

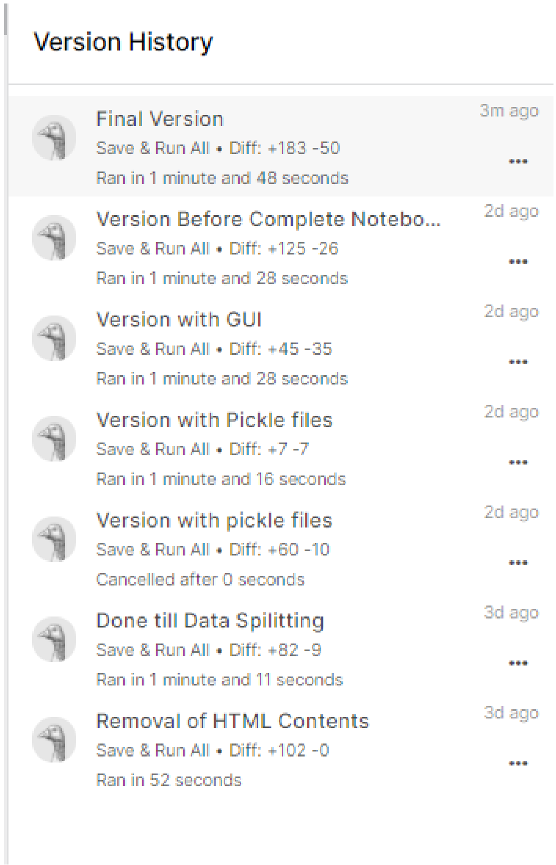
Challages Faced while doing this app:

- The first and foremost challange I face was I was unaware of how to read different parts of the email until I came acroos the email Library documentation.Once I went through the documentation and the examples using the Email library I came to know how to traverse through different portions of mail
- I had no clue of how to remove the html tags from the text the I came across this and it was using BeautifulSoap.For this I need to install beautifulsoap by doing a pip install beautifulsoap4
- Then I faced another issue whenever I am trying to print the text after removal of html I got an error and I was scratching my head then I found out that for all the multipart contnet type mails this error is throwing up so I decided to keep exception

handling here if the mails are of type multipart I would return None otherwise I would Return the value in which the HTML tags are removed

Version Control

For your reference I have added the Version History from Kaggle



Version History

Final Version
Save & Run All • Diff: +183 -50
Ran in 1 minute and 48 seconds
3m ago

Version Before Complete Notebo...
Save & Run All • Diff: +125 -26
Ran in 1 minute and 28 seconds
2d ago

Version with GUI
Save & Run All • Diff: +45 -35
Ran in 1 minute and 28 seconds
2d ago

Version with Pickle files
Save & Run All • Diff: +7 -7
Ran in 1 minute and 16 seconds
2d ago

Version with pickle files
Save & Run All • Diff: +60 -10
Cancelled after 0 seconds
2d ago

Done till Data Spilitting
Save & Run All • Diff: +82 -9
Ran in 1 minute and 11 seconds
3d ago

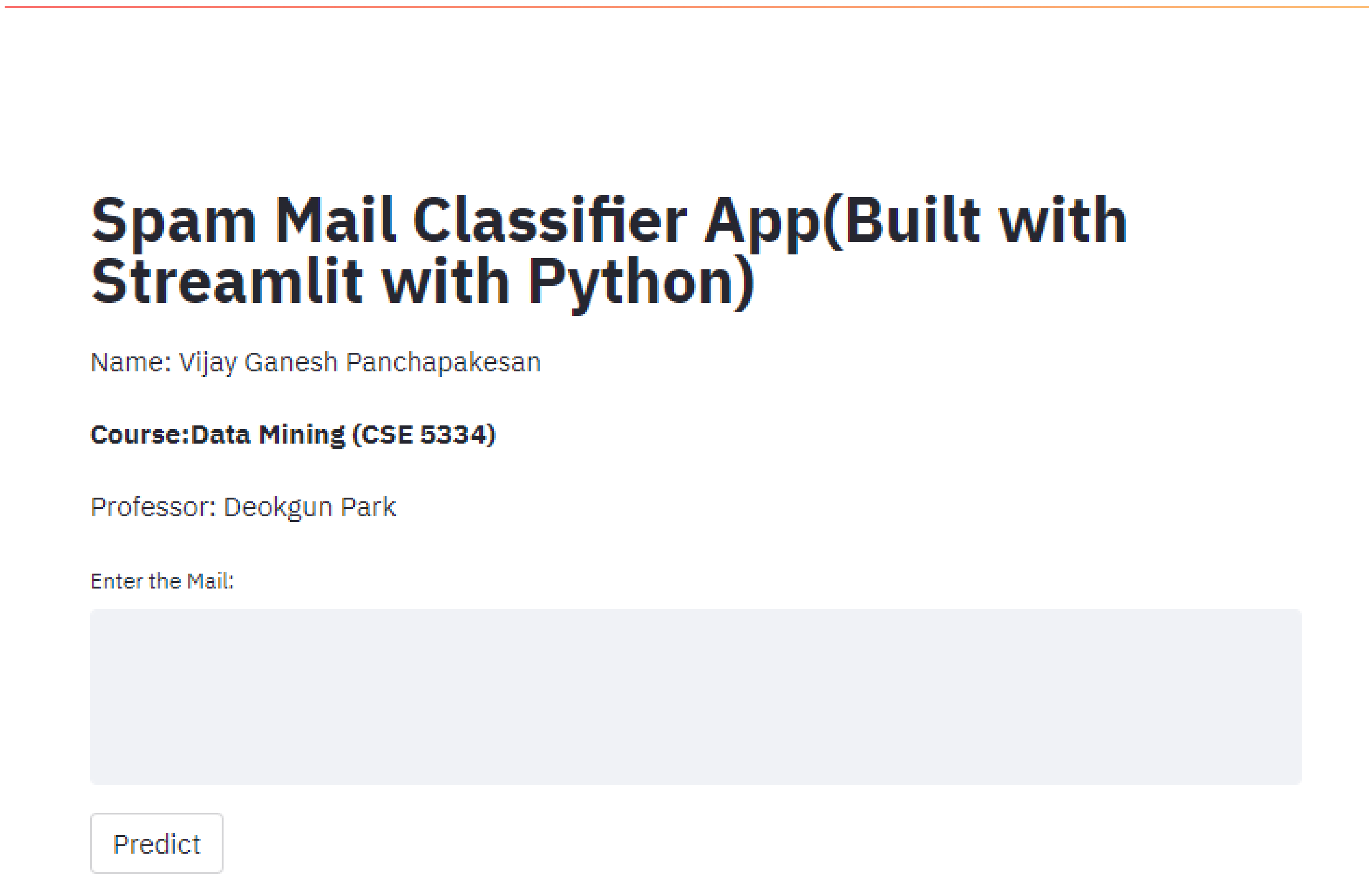
Removal of HTML Contents
Save & Run All • Diff: +102 -0
Ran in 52 seconds
3d ago

Click here to access my kaggle notebook

Youtube Demo Video Link: Demo

Simple Spam Mail Predictor App: App

Please find the screenshots of the App that I have created belwo:

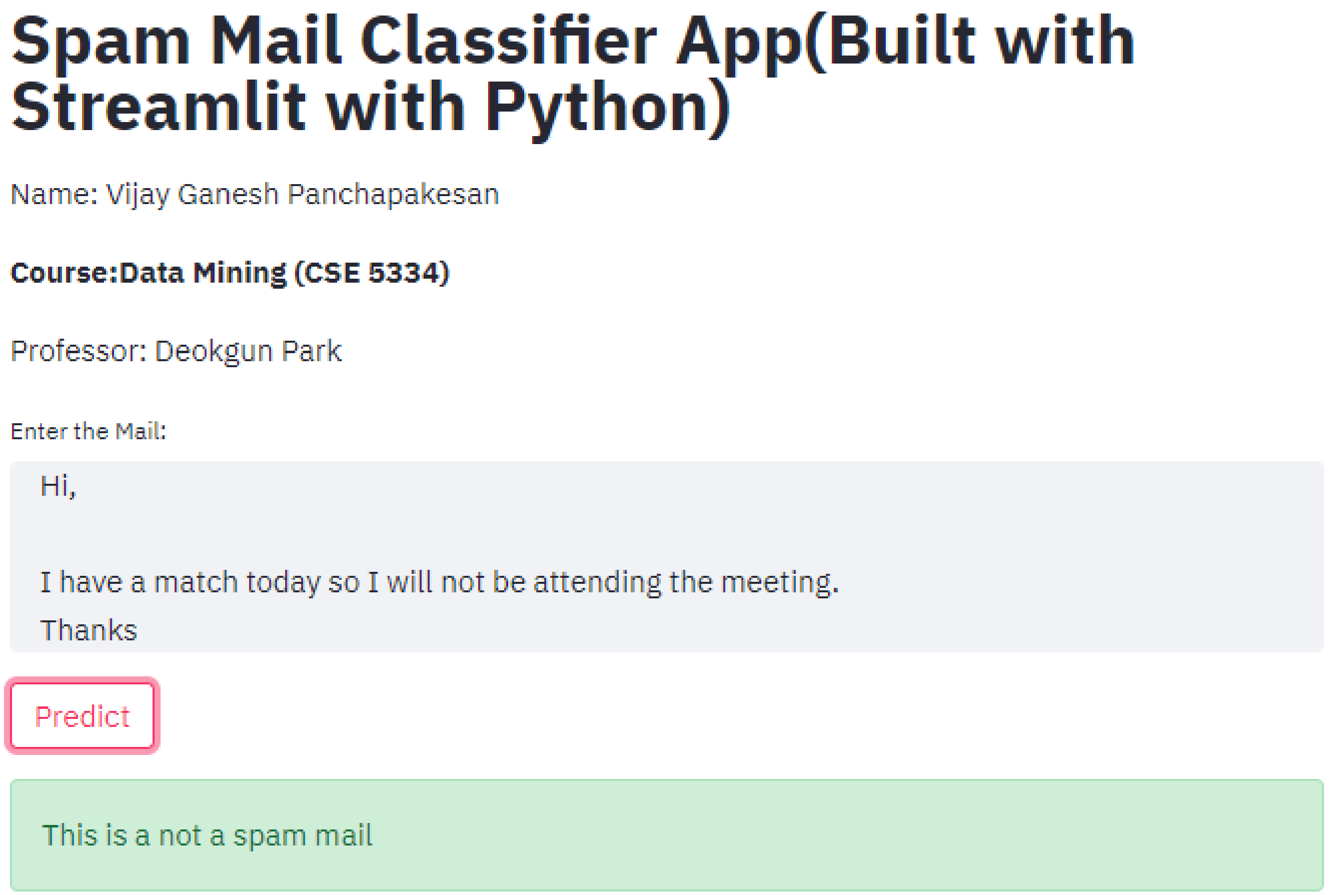


Spam Mail Classifier App(Built with Streamlit with Python)

Name: Vijay Ganesh Panchapakesan

Course:Data Mining (CSE 5334)

Professor: Deokgun Park



Spam Mail Classifier App(Built with Streamlit with Python)

Name: Vijay Ganesh Panchapakesan

Course:Data Mining (CSE 5334)

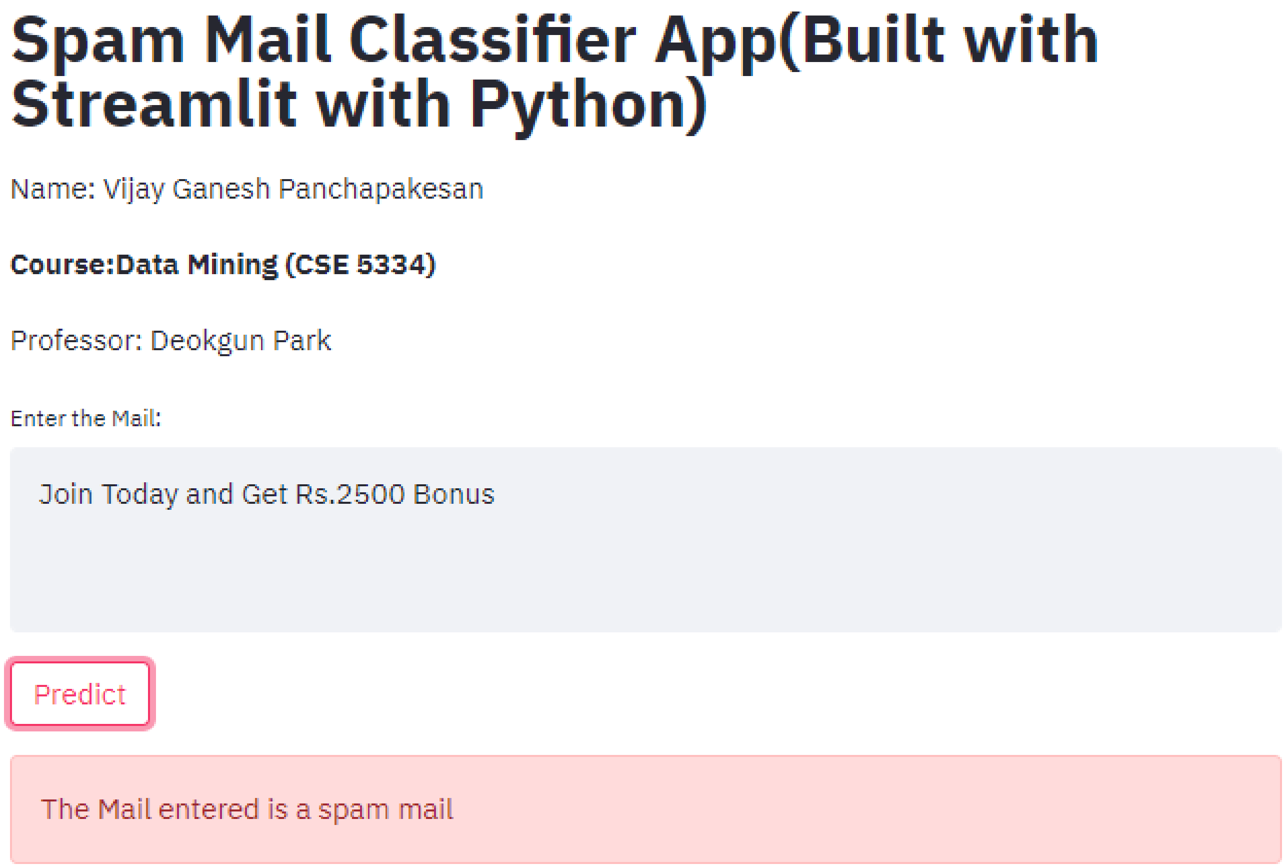
Professor: Deokgun Park

Enter the Mail:

Hi,

I have a match today so I will not be attending the meeting.
Thanks

This is a not a spam mail



Spam Mail Classifier App(Built with Streamlit with Python)

Name: Vijay Ganesh Panchapakesan

Course:Data Mining (CSE 5334)

Professor: Deokgun Park

Enter the Mail:

Join Today and Get Rs.2500 Bonus

The Mail entered is a spam mail

Refrences:

I have referenced from the following links to build my notebook and blog

- https://stackoverflow.com/questions/3207219/how-do-i-list-all-files-of-a-directory
- https://docs.python.org/3/library/email.parser.html
- https://stackoverflow.com/questions/9662346/python-code-to-remove-html-tags-from-a-string
- https://docs.python.org/3/library/email.examples.html
- https://www.kaggle.com/chetnakhanna/email-spam-ham-classification
- https://www.w3.org/Protocols/rfc1341/7_2_Multipart.html
- https://programminghistorian.org/en/lessons/intro-to-beautiful-soup