Mandar Shivaji Hanchate
Sagar Pathak

Dataset

Run the all the cell but before that upload the file

```
from google.colab import files

upload = files.upload() # to upload file on google colab
```

```
import pandas as pd

pd.set_option('display.max_columns', None) # dispaly's all colums of dataframes

data_set = pd.read_csv('SpamdetectionAssignmnet3.csv') # reading CSV file using Pandas librar

data_set
```

Choose Files No file chosen Upload widget is only available when the cell has been executed in the current browser session. Please rerun this cell to enable.

```
Saving SpamdetectionAssignmnet3.csv to SpamdetectionAssignmnet3 (1).csv
```

	Target	data
0	spam	WINNER As a valued network customer you have b...
1	spam	If you had your mobile for more than an year y...
2	ham	I am gonna be home soon
3	spam	SIX chances to win CASH From 100 to 20000 doll...
4	spam	URGENT You have won a 1 week FREE membership i...
5	ham	You have been wonderful and a blessing at all ...
6	spam	XXXMobileMovieClub To use your credit click th...
7	ham	okay I am watching here
8	spam	England vs Macedonia dont miss the team news T...
9	ham	Is that seriously how you spell his name
10	ham	did you finish your lunch already
11	ham	Alright no way I can meet up with you sooner
12	ham	Just forced myself to eat a slice I am really ...
13	ham	Did you catch the train
14	ham	tell me anything about you.
15	spam	Thanks for your subscription to Ringtone UK yo...
16	ham	Hello How are you and how did saturday go
17	spam	Rodger Burns MSG We tried to call you reply to...
18	spam	Congrats one year special cinema pass for two ...

Training Data Cleaning

21	ham	Your gonna have to pick up a burger for yourse...

```
# Converting string into indivisual words

doc = data_set.head(20)['data'].str.split(' ',expand=True)

doc
```

```
#keeping Spam related words and Ham related words in their indivisual list


Spam_word=[]
Ham_word=[]

p = 0

for i in list(data_set.head(20)['Target']):


    for j in list(doc.iloc[p,:]):

        if i == 'spam':

          Spam_word.append(j)
```

```
        else:

          Ham_word.append(j)

    p=p+1

print(Spam_word)
print(len(Spam_word))
print(Ham_word)
print(len(Ham_word))
```

```
['WINNER', 'As', 'a', 'valued', 'network', 'customer', 'you', 'have', 'been', 'selected
252
['I', 'am', 'gonna', 'be', 'home', 'soon', '', None, None, None, None, None, None, None,
308
```

Small Exercise of String Comparison

```
if 'Mandar' == 'mandar':

  print('yes')

else:

  print('No')
```

```
No
```

```
if 'Mandar' == 'Mandar ':

  print('yes')

else:

  print('No')
```

```
No
```

```
if 'Mandar' == 'Mandar':

  print('yes')

else:

  print('No')
```

```
    yes
```

```
if 'Mandar' == 'Mandar.':

  print('yes')

else:

  print('No')
```

```
    No
```

From above we can see that case of words, Space, and . matters alot which might change our output so lets take care of all these things tain and test dataset as shown below :

```
# Removing : None
# Removing : " "
# Replacing : . with nothing
# Making : Each word lowercase
# Removing : Space from word

SpamWord=[]

for i in Spam_word:
  if i != None:
    i = i.replace(".","")
    i = i.lower()
    if i.strip():
      SpamWord.append(i)

HamWord=[]

for i in Ham_word:
  if i != None:
    i = i.replace(".","")
    i = i.lower()
    if i.strip():
      HamWord.append(i)

print(SpamWord)

print(len(SpamWord))

print(HamWord)
```

```
print(len(HamWord))
```

```
['winner', 'as', 'a', 'valued', 'network', 'customer', 'you', 'have', 'been', 'selected
166
['i', 'am', 'gonna', 'be', 'home', 'soon', 'you', 'have', 'been', 'wonderful', 'and', 'a
83
```

Lets calculate number of unique words in training dataset

```
Spam_copy = SpamWord.copy()
Ham_copy = HamWord.copy()

Spam_copy.extend(Ham_copy)

unique_words = []

for i in Spam_copy:

  if i not in unique_words:

    unique_words.append(i)


print(len(unique_words))
```

```
158
```

Testing Data Cleaning

```
doc1 = data_set.tail(10)['data'].str.split(' ',expand=True)

doc1
```

```
# Removing : None
# Removing : " "
# Replacing : . with nothing
# Making : Each word lowercase
# Removing : Space from word

Final_Test_Data=list(range(10))

for i in list(range(10)):

  Final_Test_Data[i] = []    # to create list like this >> [Test_1,Test_2,Test_3,Test_4,Test_

  for j in list(doc1.iloc[i,:]):

    if j != None:
      j = j.replace(".","")
      j = j.lower()
      if j.strip():
        Final_Test_Data[i].append(j)

Final_Test_Data
```

```
[['tell', 'where', 'you', 'reached'],
 ['your',
  'gonna',
  'have',
  'to',
  'pick',
  'up',
  'a',
  'burger',
  'for',
  'yourself',
  'on',
  'your',
  'way',
  'home'],
 ['as',
  'a',
  'valued',
  'customer',
  'i',
  'am',
  'pleased',
```

```
         'to',
         'advise',
         'you',
         'that',
         'for',
         'your',
         'recent',
         'review',
         'you',
         'are',
         'awarded',
         'a',
         'bonus',
         'prize'],
        ['urgent',
         'you',
         'are',
         'awarded',
         'a',
         'complimentary',
         'trip',
         'to',
         'eurodisinc',
         'to',
         'claim',
         'text',
         'immediately'],
        ['finished', 'class', 'where', 'are', 'you'],
        ['where', 'are', 'you', 'how', 'did', 'you', 'perform'],
        ['you', 'can', 'call', 'me', 'now'],
        ['i', 'am', 'waiting', 'call', 'me', 'once', 'you', 'are', 'free'],
        ['i', 'am', 'on', 'the', 'way', 'to', 'homei'],
        ['please',
         'call',
         'our',
```

```
from operator import countOf

# number of documents = 20
print("Number documents in Train dataset : "+str(len(data_set.head(20)['data'])))
print()


# 11 number of time ham present in 20 documents
count_ham = countOf(list(data_set.head(20)['Target']), 'ham')
print("Number of time 'ham' present in 20/Training documents : "+str(count_ham))
print()


# 9 number of time ham present in 20 documents
count_spam = countOf(list(data_set.head(20)['Target']), 'spam')
print("Number of time 'spam' present in 20/Training documents : "+str(count_spam))
print()
```

```
# no of unique words

print("Number of unique wrods in Train documents : "+str(len(unique_words)))
print()


#Total number of words in Spam documents

print("Number of wrods in Spam documents : "+str(len(SpamWord)))
print()



#Total number of words in Ham documents

print("Number of wrods in Ham documents : "+str(len(HamWord)))
print()
print()


#Naive Bayes classifier : Classification on Test documents

print("************** Classification of Test Documents ***************")

print()
print()


#Probability of ham and spam

p_of_ham = count_ham/20
p_of_spam = count_spam/20


#Variables

Spam_probability_of_Doc = 0
Ham_probability_of_Doc = 0


#Function to calculate probability

def probability_calculation(doc):

  x = 1
  y = 1

  for i in doc:

    #Used laplace smoothing to avoid zero probability in Naïve Bayes
```

```
    x = x*((countOf(SpamWord, i)+1)/(len(unique_words)+len(SpamWord)))
    y = y*((countOf(HamWord, i)+1)/(len(unique_words)+len(HamWord)))

  Spam_probability_of_Doc = x*p_of_spam
  Ham_probability_of_Doc = y*p_of_ham

  if Spam_probability_of_Doc > Ham_probability_of_Doc :

    print("It is Spam","\n")

  else:

    print("It is Ham","\n")


# Feeding Test data to probability_calculation function

for i in Final_Test_Data:

  print(i)

  probability_calculation(i)
```

```
   Number documents in Train dataset : 20

   Number of time 'ham' present in 20/Training documents : 11

   Number of time 'spam' present in 20/Training documents : 9

   Number of unique wrods in Train documents : 158

   Number of wrods in Spam documents : 166

   Number of wrods in Ham documents : 83


   ************** Classification of Test Documents ***************


   ['tell', 'where', 'you', 'reached']
   It is Ham

   ['your', 'gonna', 'have', 'to', 'pick', 'up', 'a', 'burger', 'for', 'yourself', 'on', 'y
   It is Ham

   ['as', 'a', 'valued', 'customer', 'i', 'am', 'pleased', 'to', 'advise', 'you', 'that',
   It is Ham

   ['urgent', 'you', 'are', 'awarded', 'a', 'complimentary', 'trip', 'to', 'eurodisinc', 't
```

```
It is Spam

['finished', 'class', 'where', 'are', 'you']
It is Ham

['where', 'are', 'you', 'how', 'did', 'you', 'perform']
It is Ham

['you', 'can', 'call', 'me', 'now']
It is Ham

['i', 'am', 'waiting', 'call', 'me', 'once', 'you', 'are', 'free']
It is Ham

['i', 'am', 'on', 'the', 'way', 'to', 'homei']
It is Ham

['please', 'call', 'our', 'customer', 'service', 'representative', 'between', '10am-9pm
It is Spam
```

Report :

*******Test Documents *******

In test dataset, we have 10 documents :

3 documents with Spam label
7 documents with Ham label

*******OutPut of the Model *******

Actual Ham and Predicted Ham = 7

Actual Spam and Predicted Spam = 2

Actual Spam and Predicted Ham = 1

Actual Ham and Predicted Spam = 0

Total No of Documents in Test dataset = 10

Accuracy >>

= Number of Correct Predictions)/ Total No of Predictions
= ((7+2)/10)
= 0.9

Hence, we have predicted test dataset correctly with 0.9 or 90% accuracy.