Spam Checker Project made by Yllza Lela

Usign Naive Bayes algorithm to classify our data. It is a supervised, classification algorithm.

The data used here is a dataset taken from the UCI Machine Learning Repository called the "SMS Spam Collection Data Set". It contains one set of SMS messages in English of 5574 messages, tagged accordingly either "ham" (non-spam) or "spam". Can check them out here: https://archive.ics.uci.edu/ml/datasets/sms+spam+collection (https://archive.ics.uci.edu/ml/datasets/sms+spam+collection)

Importing Libraries

In [4]:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt # used for data visualization/plotting etc.
import seaborn as sns #also for data visualization
```

Importing our data

In [5]:

```
data = pd.read_csv('data.csv')
```

Showing some of the data *Not needed just helps with understanding

In [7]:

```
data.head(5) #showing first 5 rows of out data
```

Out[7]:

	text	spam
0	Subject: naturally irresistible your corporate...	1
1	Subject: the stock trading gunslinger fanny i...	1
2	Subject: unbelievable new homes made easy im ...	1
3	Subject: 4 color printing special request add...	1
4	Subject: do not have money , get software cds ...	1

In [8]:
```
data.tail(5) #last 5 datapoints
```

Out[8]:

	text	spam
5723	Subject: re : research and development charges...	0
5724	Subject: re : receipts from visit jim , than...	0
5725	Subject: re : enron case study update wow ! a...	0
5726	Subject: re : interest david , please , call...	0
5727	Subject: news : aurora 5 . 2 update aurora ve...	0

Showing number of entries, if there's null elements or not etc. *Not needed just helps with understanding

In [10]:
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5728 entries, 0 to 5727
Data columns (total 2 columns):
 #   Column  Non-Null Count  Dtype 
---  ------  --------------  ----- 
 0   text    5728 non-null   object
 1   spam    5728 non-null   int64 
dtypes: int64(1), object(1)
memory usage: 89.6+ KB
```

Visualizing our Data

In [11]:
```
ham= data[ data[ 'spam' ] == 0 ] # putting ham (non-spam) messages in one group c
```

In [17]:
```
ham.head(5) # showing the data
```

Out[17]:

	text	spam
1368	Subject: hello guys , i ' m " bugging you " f...	0
1369	Subject: sacramento weather station fyi - - ...	0
1370	Subject: from the enron india newsdesk - jan 1...	0
1371	Subject: re : powerisk 2001 - your invitation ...	0
1372	Subject: re : resco database and customer capt...	0

In [14]:
```
spam= data [ data['spam'] == 1] # same thing with spam messages
```

In [16]:
```
spam.head(5) # showing it
```

Out[16]:

	text	spam
0	Subject: naturally irresistible your corporate...	1
1	Subject: the stock trading gunslinger fanny i...	1
2	Subject: unbelievable new homes made easy im ...	1
3	Subject: 4 color printing special request add...	1
4	Subject: do not have money , get software cds ...	1

In [19]:
```
print ("Spam Percentage= ", (len(spam)/len(data)) * 100, '%' ) # Showing spam per
```

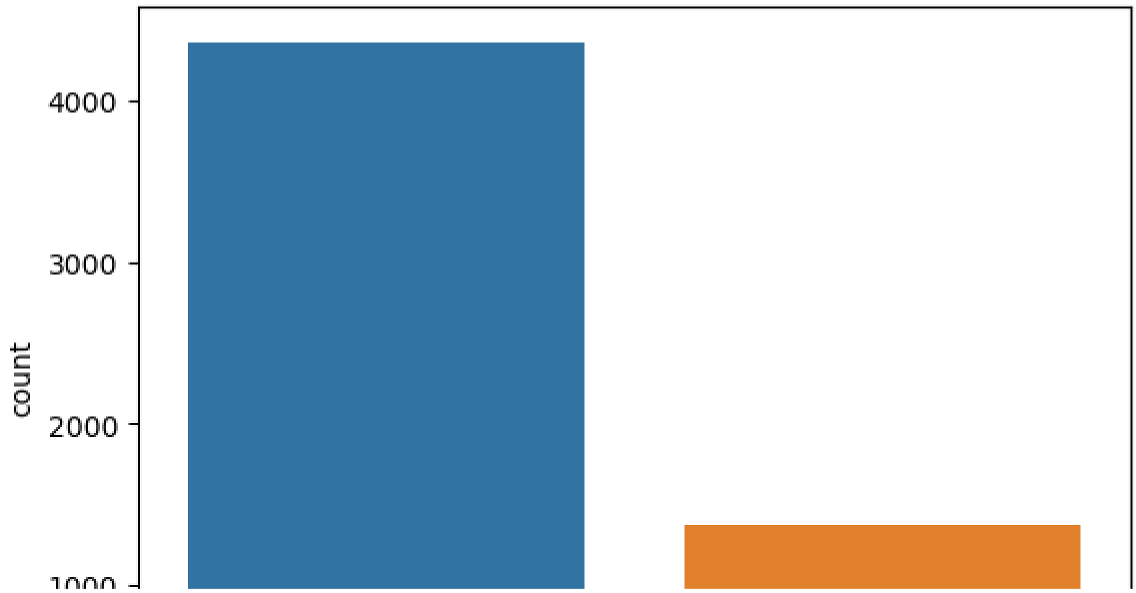
```
Spam Percentage=  23.88268156424581 %
```

In [20]:
```
print ("Ham Percentage= ", (len(ham)/len(data)) * 100, '%') # showing ham percent
```

```
Ham Percentage=  76.11731843575419 %
```

In [29]:
```
sns.countplot(x= data['spam'])
```

Out[29]:
```
<AxesSubplot:xlabel='spam', ylabel='count'>
```



Preparing our Data

In [30]:
```
from sklearn.feature_extraction.text import CountVectorizer
```

In [39]:
```
vecto = CountVectorizer()
```

In [40]:
```
data_countvectorizer = vecto.fit_transform(data['text']) #transforming the 'text
```

In [42]:
```
print(vecto.get_feature_names())  # Optional just helps to understand better
```

```
['00', '000', '0000', '000000', '00000000', '0000000000', '000000000003619',
'000000000003991', '000000000003997', '000000000005168', '000000000005409',
'000000000005411', '000000000005412', '000000000005413', '000000000005820',
'000000000006238', '000000000006452', '000000000007494', '000000000007498',
'000000000007876', '000000000010552', '000000000011185', '000000000012677',
'000000000012734', '000000000012735', '000000000012736', '000000000012738',
'000000000012741', '000000000012987', '000000000013085', '000000000013287',
'000000000015384', '000000000015793', '000000000023619', '000000000024099',
'000000000025307', '000000000025312', '000010220', '0000102317', '000010237
4', '0000102789', '0000104281', '0000104282', '0000104486', '0000104631', '00
00104730', '0000104776', '0000104778', '0000107043', '0000108729', '000066',
'0001', '000166', '0002', '000202', '0003', '0004', '0005', '0006', '00076',
'0009249480', '0009249481', '0009249504', '0009249505', '0009249506', '001',
'0011', '0015', '00193', '002', '00225', '00235424', '002813', '0029', '003',
'0031', '003399', '00343938', '004', '0044', '00453', '005', '0052', '0054',
'0057', '006', '0061', '00623', '007', '0080', '00971', '01', '010', '0100',
'01019', '0102', '0107', '01075', '0109', '011', '0110', '011000', '0115', '0
11601', '01210', '0125', '012501', '012601', '013', '014', '0141', '015', '01
500', '016', '0160', '017', '0171', '017201846', '0181', '01867', '01880', '0
```

In [43]:
```
print (data_countvectorizer.toarray())
```

```
[[0 0 0 ... 0 0 0]
 [0 0 0 ... 0 0 0]
 [0 0 0 ... 0 0 0]
 ...
 [4 0 0 ... 0 0 0]
 [0 0 0 ... 0 0 0]
 [0 0 0 ... 0 0 0]]
```

In [44]:
```
data_countvectorizer.shape
# this will show you how may samples there was (5728) and the number of words ext
```

Out[44]:
```
(5728, 37303)
```

Dividing Data and Training Model

In [48]:
```
label = data['spam'].values
```

In [47]:
```
x = data_countvectorizer # the prepared data
y = label # the spam and ham labels
```

In [51]:

```
from sklearn.model_selection import train_test_split
from sklearn.naive_bayes import MultinomialNB

x_train, x_test, y_train, y_test = train_test_split(x, y, test_size = 0.2) #divid
classifier =  MultinomialNB()
classifier.fit(x_train, y_train)  #this is our trained model, all the intelligenc
```

Out[51]:

```
MultinomialNB()
```

Evaluating our Model

In [52]:

```
from sklearn.metrics import classification_report, confusion_matrix
```

Creating 2 confusion matrices 1 for train and 1 for test

In [53]:

```
y_predict_train = classifier.predict(x_train)
y_predict_train
```

Out[53]:

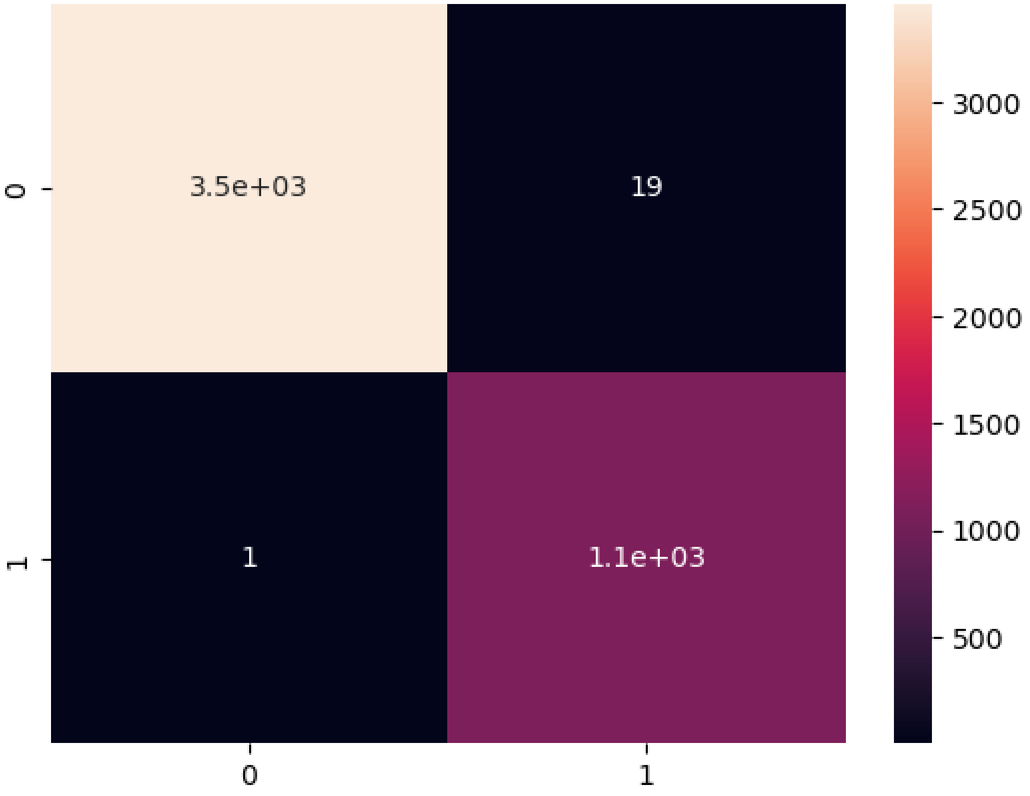
```
array([0, 0, 0, ..., 0, 0, 1], dtype=int64)
```

In [54]:

```
a = confusion_matrix(y_train, y_predict_train) # y_train = the truth # y_predict_
sns.heatmap(a, annot = True ) #visualizing
```

Out[54]:

```
<AxesSubplot:>
```



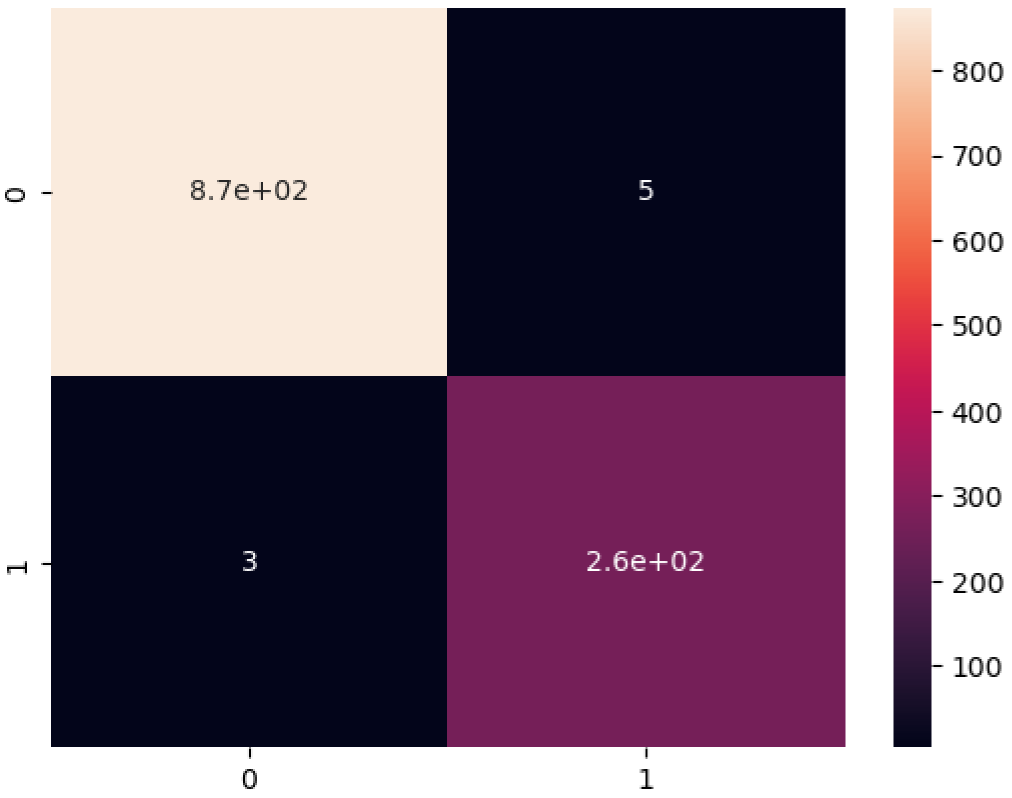
Correctly Classified: 35 and 11 hundred examples Missclassified: 20 examples. Bear in mind this is done on training sets. Not testing.

```
In [55]: y_predict_test = classifier.predict(x_test)
```

```
In [56]: b=confusion_matrix(y_test, y_predict_test)
```

```
In [57]: sns.heatmap(b, annot= True)
```

```
Out[57]: <AxesSubplot:>
```



```
In [58]: print(classification_report(y_test, y_predict_test)) #generating a report of our
```

```
              precision    recall  f1-score   support

           0       1.00      0.99      1.00       879
           1       0.98      0.99      0.99       267

    accuracy                           0.99      1146
   macro avg       0.99      0.99      0.99      1146
weighted avg       0.99      0.99      0.99      1146
```