Naive Bayes - Spam Emails Detection Project

Nawal Obaid

9/12/2022

PROBLEM STATEMENT

- The SMS Spam Collection is a set of SMS tagged messages that have been collected for SMS Spam research. It contains one set of SMS messages in English of 5,574 messages, tagged acording being ham (legitimate) or spam.
- The files contain one message per line. Each line is composed by two columns: v1 contains the label (ham or spam) and v2 contains the raw text.

Import the needed libraries

In [1]:
```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline
```

In [2]:
```
emails = pd.read_csv('emails.csv')
```

In [3]:
```
emails.head()
```

Out[3]:

	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 [4]:
```
emails.tail()
```

Out[4]:

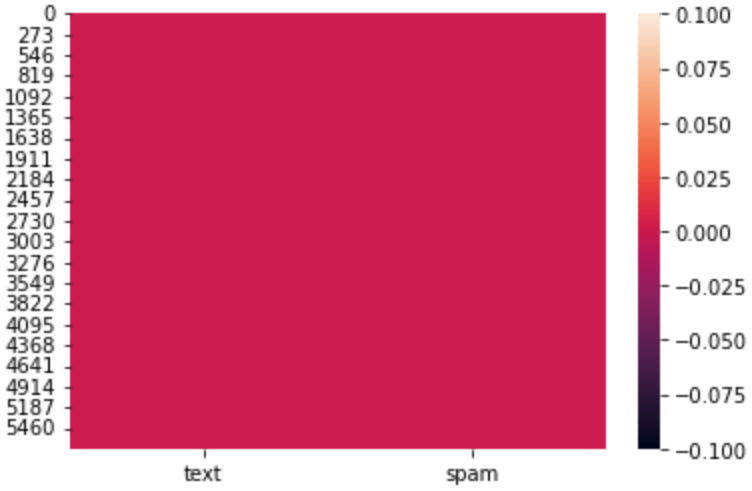
	text	spam
5723	Subject: re : research and development charges...	0
5724	Subject: re : receipts from visit jim , than...	0
5725	Subject: re : enron case study update wow ! a...	0
5726	Subject: re : interest david , please , call...	0
5727	Subject: news : aurora 5 . 2 update aurora ve...	0

In [5]:
```
emails.info()
# 5728 entries/rows
# 2 columns
# will try to detect weather an email is spam or ham
```

```
<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
```

In [10]:
```
sns.heatmap(emails.isnull())
# no missing vaules
```

Out[10]:
```
<AxesSubplot:>
```



In [18]:
```
# devide the dataset based on the value of the spam column
is_spam = emails[emails['spam']==1]
not_spam = emails[emails['spam']==0]
```

In [19]:
```
is_spam.head()
```

Out[19]:

	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 [20]:
```
not_spam.head()
```

Out[20]:

	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

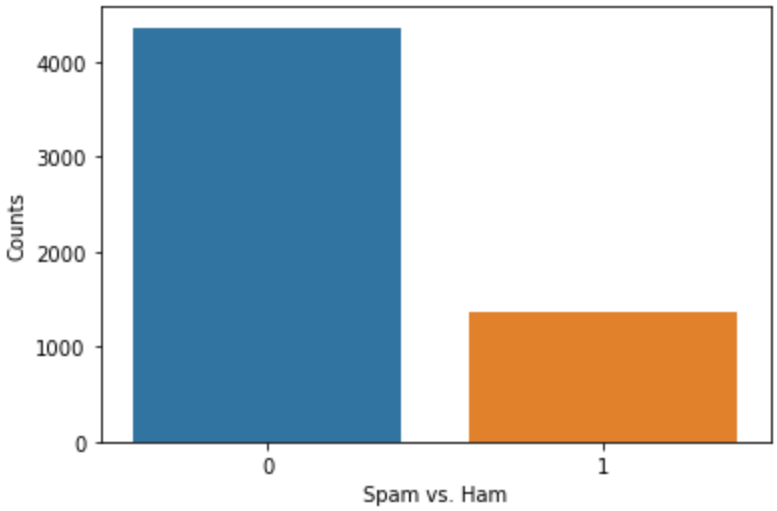
In [35]:
```
# let's get counts and percentages!
print('Number of spam emails:',len(is_spam))
print('Percentage of spam emails:', round(len(is_spam)/len(emails),2) * 100,'%')
print('Number of not spam emails:', len(not_spam))
print('Percentage of not spam emails:', round(len(not_spam)/len(emails),2) * 100,'%')
```

```
Number of spam emails: 1368
Percentage of spam emails: 24.0 %
Number of not spam emails: 4360
Percentage of not spam emails: 76.0 %
```

Data visualization

In [42]:
```
c = sns.countplot(data = emails, x = emails['spam'])
# spam is 1, not-spam or ham is 0
c.set_xlabel('Spam vs. Ham')
c.set_ylabel('Counts')
```

Out[42]:
```
Text(0, 0.5, 'Counts')
```



Applying countvectorizer to translate messages in the text column into numbers

In [46]:
```
# feature extraction using countvectorizwe
from sklearn.feature_extraction.text import CountVectorizer
vectorizer = CountVectorizer()
text_countvectorizer  = vectorizer.fit_transform(emails['text'])
# now we can perform sentiment analysis (what the user wants to say)
```

In [52]:
```
text_countvectorizer.shape
```

Out[52]:
```
(5728, 37303)
```

In [86]:
```
print(text_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 [89]:
```
#print(vectorizer.get_feature_names())
```

Train test split

In [75]:
```
X = text_countvectorizer
y = emails['spam'].values
```

In [76]:
```
X.shape
```

Out[76]:
```
(5728, 37303)
```

In [77]:
```
y.shape
```

Out[77]:
```
(5728,)
```

In [78]:
```
from sklearn.model_selection import train_test_split
```

In [79]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2)
```

In [80]:
```
from sklearn.naive_bayes import MultinomialNB
classifier = MultinomialNB()
classifier.fit(X_train, y_train)
```

Out[80]:
```
▾ MultinomialNB
MultinomialNB()
```

In [81]:
```
predictions = classifier.predict(X_test)
```

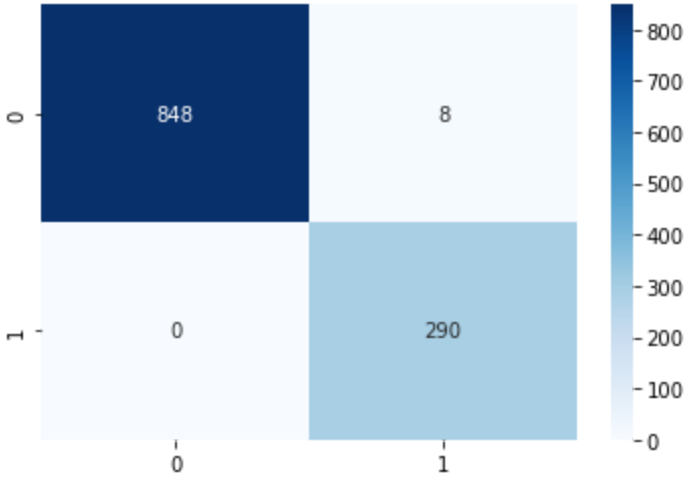
In [82]:
```
from sklearn.metrics import classification_report, confusion_matrix
```

In [83]:
```
print('**************Classification Report*************************')
print(classification_report(y_test, predictions))
print('****************Confusion Matrix*****************************')
print(confusion_matrix(y_test, predictions))
```

```
**************Classification Report*************************
              precision    recall  f1-score   support

           0       1.00      0.99      1.00       856
           1       0.97      1.00      0.99       290

    accuracy                           0.99      1146
   macro avg       0.99      1.00      0.99      1146
weighted avg       0.99      0.99      0.99      1146

****************Confusion Matrix*****************************
[[848   8]
 [  0 290]]
```

In [84]:
```
cm = confusion_matrix(y_test, predictions)
sns.heatmap(cm, annot = True, fmt = "d", cmap ='Blues')
```

Out[84]:
```
<AxesSubplot:>
```


848
8
0
290
0
1
0
1
800
700
600
500
400
300
200
100
0