Oasis Infobyte - OIBSIP - Data Science

Task 2 : Email Spam Detection Using with Machine Learning

Intern - Shravani Mahesuni

Problem Statement : Each of us has received spam emails at some point. A sort of email known as spam mail, sometimes known as junk mail, is one that is sent to a large number of people at once and typically includes cryptic messages, con games, or—most dangerously—phishing content.Python will be used in this project to create an email spam detector. After that, employ machine learning to teach the spam detector to identify and categorize emails as spam or not spam.

In [2]:
```
# Importing Python libraries from SciKit Learn for analyzing models
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.metrics import confusion_matrix,classification_report,accuracy_score,f1_score,recall_score
```

scikit-learn is a free software machine learning library for the Python programming language. It features various classification, regression and clustering algorithms including support-vector machines.

In [3]:
```
# Loading Dataset / Reading dataset
df = pd.read_csv("spam.csv",encoding="latin1")
df
```

Out[3]:

	v1	v2	Unnamed: 2	Unnamed: 3	Unnamed: 4
0	ham	Go until jurong point, crazy.. Available only ...	NaN	NaN	NaN
1	ham	Ok lar... Joking wif u oni...	NaN	NaN	NaN
2	spam	Free entry in 2 a wkly comp to win FA Cup fina...	NaN	NaN	NaN
3	ham	U dun say so early hor... U c already then say...	NaN	NaN	NaN
4	ham	Nah I don't think he goes to usf, he lives aro...	NaN	NaN	NaN
...	...	...	...	...	...
5567	spam	This is the 2nd time we have tried 2 contact u...	NaN	NaN	NaN
5568	ham	Will Ì_ b going to esplanade fr home?	NaN	NaN	NaN
5569	ham	Pity, * was in mood for that. So...any other s...	NaN	NaN	NaN
5570	ham	The guy did some bitching but I acted like i'd...	NaN	NaN	NaN
5571	ham	Rofl. Its true to its name	NaN	NaN	NaN

5572 rows × 5 columns

In [9]:
```
# Print Shape (Get the number of rows and columns)
df.shape
```

Out[9]:
```
(5169, 5)
```

In [10]:
```
# get the columns names
df.columns
```

Out[10]:
```
Index(['v1', 'v2', 'Unnamed: 2', 'Unnamed: 3', 'Unnamed: 4'], dtype='object')
```

In [14]:
```
# Check for number of rows and columns present
print('rows-->',df.shape[0])
print('columns-->',df.shape[1])
```

```
rows--> 5169
columns--> 5
```

In [15]:
```
# Check out the number of null values in the df.
df.isnull().sum()
```

Out[15]:
```
v1               0
v2               0
Unnamed: 2    5126
Unnamed: 3    5159
Unnamed: 4    5164
dtype: int64
```

In [16]:
```
# print percentage of null values
df.isnull().mean()*100
```

Out[16]:
```
v1             0.000000
v2             0.000000
Unnamed: 2    99.168118
Unnamed: 3    99.806539
Unnamed: 4    99.903269
dtype: float64
```

As you can see, there are a significant number of missing entries in the relevant columns Unnamed:2,Unnamed:3,Unnamed:4 more than 99%—so we must get rid of these null values fields.

In [17]:
```
#removing those columns which we don't want in calculation
df.drop(columns=df[['Unnamed: 2','Unnamed: 3','Unnamed: 4']],axis=1,inplace=True)
```

In [18]:
```
print(df)
```

```
        v1                                                 v2
0      ham  Go until jurong point, crazy.. Available only ...
1      ham                      Ok lar... Joking wif u oni...
2     spam  Free entry in 2 a wkly comp to win FA Cup fina...
3      ham  U dun say so early hor... U c already then say...
4      ham  Nah I don't think he goes to usf, he lives aro...
...    ...                                                ...
5567  spam  This is the 2nd time we have tried 2 contact u...
5568   ham              Will Ì_ b going to esplanade fr home?
5569   ham  Pity, * was in mood for that. So...any other s...
5570   ham  The guy did some bitching but I acted like i'd...
5571   ham                         Rofl. Its true to its name

[5169 rows x 2 columns]
```

As you can see here that extra columns successfully reduced !

In [19]:
```
# Checking after reduction
df.shape
```

Out[19]:
```
(5169, 2)
```

In [32]:
```
# renaming the column names to make them clear
df.columns = ['spam/ham','msg']
```

In [33]:
```
df.loc[df['spam/ham'] == 'spam', 'spam/ham',] = 0
df.loc[df['spam/ham'] == 'ham', 'spam/ham',] = 1
```

In [35]:
```
df
```

Out[35]:

	spam/ham	msg
0	1	Go until jurong point, crazy.. Available only ...
1	1	Ok lar... Joking wif u oni...
2	0	Free entry in 2 a wkly comp to win FA Cup fina...
3	1	U dun say so early hor... U c already then say...
4	1	Nah I don't think he goes to usf, he lives aro...
...	...	...
5567	0	This is the 2nd time we have tried 2 contact u...
5568	1	Will Ì_ b going to esplanade fr home?
5569	1	Pity, * was in mood for that. So...any other s...
5570	1	The guy did some bitching but I acted like i'd...
5571	1	Rofl. Its true to its name

5169 rows × 2 columns

In [36]:
```
x = df.msg
x
```

Out[36]:
```
0       Go until jurong point, crazy.. Available only ...
1                           Ok lar... Joking wif u oni...
2       Free entry in 2 a wkly comp to win FA Cup fina...
3       U dun say so early hor... U c already then say...
4       Nah I don't think he goes to usf, he lives aro...
                              ...
5567    This is the 2nd time we have tried 2 contact u...
5568                Will Ì_ b going to esplanade fr home?
5569    Pity, * was in mood for that. So...any other s...
5570    The guy did some bitching but I acted like i'd...
5571                           Rofl. Its true to its name
Name: msg, Length: 5169, dtype: object
```

In [37]:
```
y = df['spam/ham']
y
```

Out[37]:
```
0       1
1       1
2       0
3       1
4       1
       ..
5567    0
5568    1
5569    1
5570    1
5571    1
Name: spam/ham, Length: 5169, dtype: object
```

In [38]:
```
#Devide the whole dataset into training and testing set for model training
from sklearn.model_selection import train_test_split
```

In [39]:
```
xtrain,xtest,ytrain,ytest=train_test_split(x,y,test_size=0.2,random_state=3)
```

In [40]:
```
print(x.shape)
print(xtrain.shape)
print(xtest.shape)
```

```
(5169,)
(4135,)
(1034,)
```

In [41]:
```
xtrain,xtest
```

Out[41]:
```
(4443                  COME BACK TO TAMPA FFFFFUUUUUUU
 [illegible]
 Name: msg, Length: 4135, dtype: object,
 [illegible]
 Name: msg, Length: 1034, dtype: object)
```

In [42]:
```
ytrain,ytest
```

Out[42]:
```
(4443    1
 [illegible]
 Name: spam/ham, Length: 4135, dtype: object,
 [illegible]
 Name: spam/ham, Length: 1034, dtype: object)
```

We must convert all of the text input into numbers since, as far as we are aware, machine learning algorithms only work well with respect to numbers. I'll utilize the TfidfVectorizer method from Sklearn's feature_extraction section to accomplish this.

In [43]:
```
feat_vect=TfidfVectorizer(min_df=1,stop_words='english',lowercase=True)
feat_vect
```

Out[43]:
```
TfidfVectorizer
TfidfVectorizer(stop_words='english')
```

In [44]:
```
ytrain=ytrain.astype('int')
ytest=ytest.astype('int')
```

In [45]:
```
xtrain_vec = feat_vect.fit_transform(xtrain)
```

In [46]:
```
xtest_vec = feat_vect.transform(xtest)
```

In [47]:
```
print(xtrain)
```

```
4443                  COME BACK TO TAMPA FFFFFUUUUUUU
[illegible]
Name: msg, Length: 4135, dtype: object
```

In [48]:
```
xtrain_vec
```

Out[48]:
```
<4135x7378 sparse matrix of type '<class 'numpy.float64'>'
	with 31488 stored elements in Compressed Sparse Row format>
```

In [49]:
```
#printing x train data
print(xtrain_vec)
```

```
[illegible]
```

In [50]:
```
xtest_vec
```

Out[50]:
```
<1034x7378 sparse matrix of type '<class 'numpy.float64'>'
	with 7012 stored elements in Compressed Sparse Row format>
```

In [51]:
```
#printing x test data
print(xtest_vec)
```

```
[illegible]
```

In [52]:
```
logi = LogisticRegression()
```

In [53]:
```
logi.fit(xtrain_vec,ytrain)
```

Out[53]:
```
LogisticRegression
LogisticRegression()
```

In [54]:
```
logi.score(xtrain_vec,ytrain)
```

Out[54]:
```
[illegible]
```

In [55]:
```
logi.score(xtest_vec,ytest)
```

Out[55]:
```
[illegible]
```

In [56]:
```
pred_logi = logi.predict(xtest_vec)
pred_logi
```

Out[56]:
```
array([1, 1, 1, ..., 1, 0, 1])
```

In [57]:
```
from sklearn.metrics import confusion_matrix,classification_report,accuracy_score
```

In [58]:
```
accuracy_score(ytest,pred_logi)
```

Out[58]:
```
[illegible]
```

In [59]:
```
confusion_matrix(ytest,pred_logi)
```

Out[59]:
```
array([[ 99,  41],
       [  0, 894]], dtype=int64)
```

In [60]:
```
print(classification_report(ytest,pred_logi))
```

```
              precision    recall  f1-score   support

           0       1.00      0.71      0.83       140
           1       0.96      1.00      0.98       894

    accuracy                           0.96      1034
   macro avg       0.98      0.85      0.90      1034
weighted avg       0.96      0.96      0.96      1034
```

Final Classification report analysis

In [ ]:

In [1]:
```
df.info()
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
Cell In[1], line 1
----> 1 df.info()

NameError: name 'df' is not defined
```

In [2]:
```
df.info()
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
Cell In[2], line 1
----> 1 df.info()

NameError: name 'df' is not defined
```

In [ ]: