In [1]:

```
# This Python 3 environment comes with many helpful analytics libraries insta
# It is defined by the kaggle/python Docker image: https://github.com/kaggle/
# For example, here's several helpful packages to load

import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)

# Input data files are available in the read-only "../input/" directory
# For example, running this (by clicking run or pressing Shift+Enter) will li

import os
for dirname, _, filenames in os.walk('/kaggle/input'):
    for filename in filenames:
        print(os.path.join(dirname, filename))

# You can write up to 20GB to the current directory (/kaggle/working/) that g
# You can also write temporary files to /kaggle/temp/, but they won't be save
```

In [2]:

```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline
import warnings
warnings.filterwarnings('ignore')
from sklearn.model_selection import train_test_split
from sklearn.svm import SVC
from sklearn import metrics
```

In [3]:

```
df=pd.read_csv("../input/email-spam-classification-dataset-csv/emails.csv")
```

In [4]:
```
df
```

Out[4]:

| | Email No. | the | to | ect | and | for | of | a | you | hou | ... | connevey | jay | valued | lay | infra |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **0** | Email 1 | 0 | 0 | 1 | 0 | 0 | 0 | 2 | 0 | 0 | ... | 0 | 0 | 0 | 0 | |
| **1** | Email 2 | 8 | 13 | 24 | 6 | 6 | 2 | 102 | 1 | 27 | ... | 0 | 0 | 0 | 0 | |
| **2** | Email 3 | 0 | 0 | 1 | 0 | 0 | 0 | 8 | 0 | 0 | ... | 0 | 0 | 0 | 0 | |
| **3** | Email 4 | 0 | 5 | 22 | 0 | 5 | 1 | 51 | 2 | 10 | ... | 0 | 0 | 0 | 0 | |
| **4** | Email 5 | 7 | 6 | 17 | 1 | 5 | 2 | 57 | 0 | 9 | ... | 0 | 0 | 0 | 0 | |
| **...** | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | |
| **5167** | Email 5168 | 2 | 2 | 2 | 3 | 0 | 0 | 32 | 0 | 0 | ... | 0 | 0 | 0 | 0 | |
| **5168** | Email 5169 | 35 | 27 | 11 | 2 | 6 | 5 | 151 | 4 | 3 | ... | 0 | 0 | 0 | 0 | |
| **5169** | Email 5170 | 0 | 0 | 1 | 1 | 0 | 0 | 11 | 0 | 0 | ... | 0 | 0 | 0 | 0 | |
| **5170** | Email 5171 | 2 | 7 | 1 | 0 | 2 | 1 | 28 | 2 | 0 | ... | 0 | 0 | 0 | 0 | |
| **5171** | Email 5172 | 22 | 24 | 5 | 1 | 6 | 5 | 148 | 8 | 2 | ... | 0 | 0 | 0 | 0 | |

5172 rows × 3002 columns

In [5]:
```
df.head()
```

Out[5]:

| | Email No. | the | to | ect | and | for | of | a | you | hou | ... | connevey | jay | valued | lay | infrastru |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **0** | Email 1 | 0 | 0 | 1 | 0 | 0 | 0 | 2 | 0 | 0 | ... | 0 | 0 | 0 | 0 | |
| **1** | Email 2 | 8 | 13 | 24 | 6 | 6 | 2 | 102 | 1 | 27 | ... | 0 | 0 | 0 | 0 | |
| **2** | Email 3 | 0 | 0 | 1 | 0 | 0 | 0 | 8 | 0 | 0 | ... | 0 | 0 | 0 | 0 | |
| **3** | Email 4 | 0 | 5 | 22 | 0 | 5 | 1 | 51 | 2 | 10 | ... | 0 | 0 | 0 | 0 | |
| **4** | Email 5 | 7 | 6 | 17 | 1 | 5 | 2 | 57 | 0 | 9 | ... | 0 | 0 | 0 | 0 | |

5 rows × 3002 columns

In [6]:
```
df.info
```

Out[6]:
```
<bound method DataFrame.info of        Email No.  the  to  ect  and  for  o
f    a  you  hou  ...  connevey  \
0         Email 1    0   0    1    0    0   0    2    0    0  ...         0
1         Email 2    8  13   24    6    6   2  102    1   27  ...         0
2         Email 3    0   0    1    0    0   0    8    0    0  ...         0
3         Email 4    0   5   22    0    5   1   51    2   10  ...         0
4         Email 5    7   6   17    1    5   2   57    0    9  ...         0
...           ...  ...  ..  ...  ...  ...  ..  ...  ...  ...  ...       ...
5167  Email 5168    2   2    2    3    0   0   32    0    0  ...         0
5168  Email 5169   35  27   11    2    6   5  151    4    3  ...         0
5169  Email 5170    0   0    1    1    0   0   11    0    0  ...         0
5170  Email 5171    2   7    1    0    2   1   28    2    0  ...         0
5171  Email 5172   22  24    5    1    6   5  148    8    2  ...         0

      jay  valued  lay  infrastructure  military  allowing  ff  dry  \
0       0       0    0               0         0         0   0    0
1       0       0    0               0         0         0   1    0
2       0       0    0               0         0         0   0    0
3       0       0    0               0         0         0   0    0
4       0       0    0               0         0         0   1    0
...   ...     ...  ...             ...       ...       ...  ..  ...
5167    0       0    0               0         0         0   0    0
5168    0       0    0               0         0         0   1    0
5169    0       0    0               0         0         0   0    0
5170    0       0    0               0         0         0   1    0
5171    0       0    0               0         0         0   0    0

      Prediction
0              0
1              0
2              0
3              0
4              0
...          ...
5167           0
5168           0
5169           1
5170           1
5171           0

[5172 rows x 3002 columns]>
```

In [7]:
```
df.shape
```

Out[7]:
```
(5172, 3002)
```

In [8]:
```
df.columns
```

Out[8]:
```
Index(['Email No.', 'the', 'to', 'ect', 'and', 'for', 'of', 'a', 'you', 'hou',
       ...
       'connevey', 'jay', 'valued', 'lay', 'infrastructure', 'military',
       'allowing', 'ff', 'dry', 'Prediction'],
      dtype='object', length=3002)
```

In [9]:
```
df.isnull().sum()
```

Out[9]:
```
Email No.     0
the           0
to            0
ect           0
and           0
             ..
military      0
allowing      0
ff            0
dry           0
Prediction    0
Length: 3002, dtype: int64
```

In [10]:
```
df.dropna(inplace = True)
df.drop(['Email No.'],axis=1,inplace=True)
X = df.drop(['Prediction'],axis = 1)
y = df['Prediction']
df
```

Out[10]:

| | the | to | ect | and | for | of | a | you | hou | in | ... | connevey | jay | valued | lay | infr |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 0 | 0 | 1 | 0 | 0 | 0 | 2 | 0 | 0 | 0 | ... | 0 | 0 | 0 | 0 | |
| **1** | 8 | 13 | 24 | 6 | 6 | 2 | 102 | 1 | 27 | 18 | ... | 0 | 0 | 0 | 0 | |
| **2** | 0 | 0 | 1 | 0 | 0 | 0 | 8 | 0 | 0 | 4 | ... | 0 | 0 | 0 | 0 | |
| **3** | 0 | 5 | 22 | 0 | 5 | 1 | 51 | 2 | 10 | 1 | ... | 0 | 0 | 0 | 0 | |
| **4** | 7 | 6 | 17 | 1 | 5 | 2 | 57 | 0 | 9 | 3 | ... | 0 | 0 | 0 | 0 | |
| **...** | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | |
| **5167** | 2 | 2 | 2 | 3 | 0 | 0 | 32 | 0 | 0 | 5 | ... | 0 | 0 | 0 | 0 | |
| **5168** | 35 | 27 | 11 | 2 | 6 | 5 | 151 | 4 | 3 | 23 | ... | 0 | 0 | 0 | 0 | |
| **5169** | 0 | 0 | 1 | 1 | 0 | 0 | 11 | 0 | 0 | 1 | ... | 0 | 0 | 0 | 0 | |
| **5170** | 2 | 7 | 1 | 0 | 2 | 1 | 28 | 2 | 0 | 8 | ... | 0 | 0 | 0 | 0 | |
| **5171** | 22 | 24 | 5 | 1 | 6 | 5 | 148 | 8 | 2 | 23 | ... | 0 | 0 | 0 | 0 | |

5172 rows × 3001 columns

In [11]:
```
from sklearn.preprocessing import scale
X = scale(X)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.3, ra
```

In [12]:
```
X
```

Out[12]:
```
array([[-0.56544926, -0.64908256, -0.2938948 , ..., -0.0562853 ,
        -0.32904848, -0.07097072],
       [ 0.11575699,  0.71450797,  1.33733653, ..., -0.0562853 ,
         0.03067224, -0.07097072],
       [-0.56544926, -0.64908256, -0.2938948 , ..., -0.0562853 ,
        -0.32904848, -0.07097072],
       ...,
       [-0.56544926, -0.64908256, -0.2938948 , ..., -0.0562853 ,
        -0.32904848, -0.07097072],
       [-0.3951477 ,  0.0851585 , -0.2938948 , ..., -0.0562853 ,
         0.03067224, -0.07097072],
       [ 1.30786793,  1.86831533, -0.0102024 , ..., -0.0562853 ,
        -0.32904848, -0.07097072]])
```

In [13]:
```
X_train
```

Out[13]:
```
array([[-0.13969536, -0.22951624, -0.2938948 , ..., -0.0562853 ,
        -0.32904848, -0.07097072],
       [-0.30999692, -0.01973308, -0.0811255 , ..., -0.0562853 ,
         0.03067224, -0.07097072],
       [-0.48029848, -0.64908256, -0.2938948 , ..., -0.0562853 ,
        -0.32904848, -0.07097072],
       ...,
       [ 0.45636011,  1.23896586,  0.20256691, ..., -0.0562853 ,
         0.39039297, -0.07097072],
       [ 0.28605855, -0.01973308,  6.44379983, ..., -0.0562853 ,
         0.03067224, -0.07097072],
       [-0.48029848,  0.19005007, -0.2229717 , ..., -0.0562853 ,
        -0.32904848, -0.07097072]])
```

In [14]:
```
X_test
```

Out[14]:
```
array([[ 0.62666168,  0.0851585 , -0.2229717 , ..., -0.0562853 ,
        -0.32904848, -0.07097072],
       [-0.05454457, -0.4392994 , -0.2938948 , ..., -0.0562853 ,
        -0.32904848, -0.07097072],
       [ 2.84058199, -0.01973308, -0.0811255 , ..., -0.0562853 ,
         0.03067224, -0.07097072],
       ...,
       [ 0.11575699, -0.01973308, -0.2229717 , ..., -0.0562853 ,
        -0.32904848, -0.07097072],
       [-0.48029848, -0.4392994 , -0.2229717 , ..., -0.0562853 ,
         0.03067224, -0.07097072],
       [-0.48029848, -0.54419098, -0.2938948 , ..., -0.0562853 ,
        -0.32904848, -0.07097072]])
```

In [15]:
```
y_train
```

Out[15]:
```
3459    1
1385    0
1380    0
4462    1
3840    0
       ..
4931    0
3264    1
1653    1
2607    0
2732    1
Name: Prediction, Length: 3620, dtype: int64
```

In [16]:
```
y_test
```

Out[16]:
```
3324    0
15      0
4950    0
3964    1
2315    0
       ..
1412    1
36      0
4247    0
4858    1
4299    0
Name: Prediction, Length: 1552, dtype: int64
```

In [17]:
```
X_train.shape
```

Out[17]:
```
(3620, 3000)
```

In [18]:
```
y_train.shape
```

Out[18]:
```
(3620,)
```

In [19]:
```
X_test.shape
```

Out[19]:
```
(1552, 3000)
```

In [20]:
```
y_test.shape
```

Out[20]:
```
(1552,)
```

In [21]:
```
from sklearn.neighbors import KNeighborsClassifier
knn = KNeighborsClassifier(n_neighbors=3)
knn.fit(X_train, y_train)
y_pred = knn.predict(X_test)
```

In [22]:
```
print("Prediction",y_pred)
```

```
Prediction [1 0 0 ... 0 1 1]
```

In [23]:
```
print("KNN accuracy = ",metrics.accuracy_score(y_test,y_pred))
```

```
KNN accuracy =  0.8402061855670103
```

In [24]:
```
print("Confusion matrix",metrics.confusion_matrix(y_test,y_pred))
```

```
Confusion matrix [[900 211]
 [ 37 404]]
```

In [25]:
```
model = SVC(C=1)
model.fit(X_train, y_train)
y_pred = model.predict(X_test)
```

In [26]:
```
print("SVM accuracy = ",metrics.accuracy_score(y_test,y_pred))
```

```
SVM accuracy =  0.9329896907216495
```

In [27]:
```
metrics.confusion_matrix(y_true=y_test, y_pred=y_pred)
```

Out[27]:
```
array([[1106,    5],
       [  99,  342]])
```