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

In [3]:
```
df=pd.read_csv("C:/Users/KJCOEMR/Desktop/emails.csv")
```

In [7]:
```
df.head()
```

Out[7]:

	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	

5 rows × 3002 columns

In [9]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5172 entries, 0 to 5171
Columns: 3002 entries, Email No. to Prediction
dtypes: int64(3001), object(1)
memory usage: 118.5+ MB
```

In [11]:
```
df.isnull().sum()
```

Out[11]:
```
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 [13]: X = df.iloc[:, 1:-1].values
         y = df.iloc[:, -1].values
```

```
In [15]: from sklearn.model_selection import train_test_split
         X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.30, random_st
```

```
In [17]: from sklearn.preprocessing import StandardScaler
         sc_X = StandardScaler()
         X_train = sc_X.fit_transform(X_train)
         X_test = sc_X.transform(X_test)
```

```
In [19]: from sklearn.neighbors import KNeighborsClassifier
         classifier = KNeighborsClassifier(n_neighbors=5)
         classifier.fit(X_train, y_train)
```

```
Out[19]: ▾ KNeighborsClassifier  i ?

         KNeighborsClassifier()
```

```
In [21]: y_pred = classifier.predict(X_test)
```

```
In [23]: from sklearn.metrics import confusion_matrix, accuracy_score
         cm = confusion_matrix(y_test, y_pred)
```

```
In [25]: cm
```

```
Out[25]: array([[866, 248],
                [ 16, 422]], dtype=int64)
```

```
In [27]: from sklearn.metrics import classification_report
         cl_report=classification_report(y_test,y_pred)
         print(cl_report)
```

```
              precision    recall  f1-score   support

           0       0.98      0.78      0.87      1114
           1       0.63      0.96      0.76       438

    accuracy                           0.83      1552
   macro avg       0.81      0.87      0.81      1552
weighted avg       0.88      0.83      0.84      1552
```

```
In [29]: print("Accuracy Score for KNN : ", accuracy_score(y_pred,y_test))
```

```
Accuracy Score for KNN :  0.8298969072164949
```

```
In [31]: from sklearn.svm import SVC
```

```
In [33]: svm_classifier = SVC(kernel='linear', random_state=101)
         svm_classifier.fit(X_train, y_train)
```

Out[33]:

```
▾                SVC

SVC(kernel='linear', random_state=101)
```

In [34]:

```
y_pred_svm = svm_classifier.predict(X_test)
```

In [37]:

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

In [39]:

```
cm_svm = confusion_matrix(y_test, y_pred_svm)
print("Confusion Matrix for SVM:")
print(cm_svm)
```

```
Confusion Matrix for SVM:
[[1069   45]
 [  42  396]]
```

In [41]:

```
cl_report_svm = classification_report(y_test, y_pred_svm)
print("Classification Report for SVM:")
print(cl_report_svm)
```

```
Classification Report for SVM:
              precision    recall  f1-score   support

           0       0.96      0.96      0.96      1114
           1       0.90      0.90      0.90       438

    accuracy                           0.94      1552
   macro avg       0.93      0.93      0.93      1552
weighted avg       0.94      0.94      0.94      1552
```

In [43]:

```
accuracy_svm = accuracy_score(y_test, y_pred_svm)
print("Accuracy Score for SVM:", accuracy_svm)
```

```
Accuracy Score for SVM: 0.9439432989690721
```

In [45]:

```
print("KNN Accuracy:", accuracy_score(y_test, y_pred))
print("SVM Accuracy:", accuracy_svm)
```

```
KNN Accuracy: 0.8298969072164949
SVM Accuracy: 0.9439432989690721
```

In [47]:

```
if accuracy_svm > accuracy_score(y_test, y_pred):
    print("SVM performed better on this dataset.")
else:
    print("KNN performed better on this dataset.")
```

```
SVM performed better on this dataset.
```

In [ ]:

```

```