Import the relevant libraries

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

Read the dataset

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

Out[2]:

	Email No.	the	to	ect	and	for	of	a	you	hou	...	connevey	jay	valued	lay	infrastructure	military	allowing	ff	dry	Prediction
0	Email 1	0	0	1	0	0	0	2	0	0	...	0	0	0	0	0	0	0	0	0	0
1	Email 2	8	13	24	6	6	2	102	1	27	...	0	0	0	0	0	0	0	1	0	0
2	Email 3	0	0	1	0	0	0	8	0	0	...	0	0	0	0	0	0	0	0	0	0
3	Email 4	0	5	22	0	5	1	51	2	10	...	0	0	0	0	0	0	0	0	0	0
4	Email 5	7	6	17	1	5	2	57	0	9	...	0	0	0	0	0	0	0	1	0	0

5 rows × 3002 columns

Features and Target Selection

In [3]:
```
target = df['Prediction']
features = df.drop(['Prediction', 'Email No.'], axis=1)
```

Standardization

In [4]:
```
from sklearn.preprocessing import StandardScaler

scaler = StandardScaler()
features_scaled = scaler.fit_transform(features)
```

Train Test Split

In [5]:
```
from sklearn.model_selection import train_test_split

x_train, x_test, y_train, y_test = train_test_split(features_scaled, target, test_size=0.2, random_state=42)
```

Model Training

1. K-Nearest Neighbor (KNN)

In [6]:
```
from sklearn.neighbors import KNeighborsClassifier

knn_model = KNeighborsClassifier()
knn_model.fit(x_train, y_train)
```

Out[6]:
```
KNeighborsClassifier
KNeighborsClassifier()
```

In [7]:
```
training_accuracy = knn_model.score(x_train, y_train)
testing_accuracy = knn_model.score(x_test, y_test)
print('Training Accuracy:', training_accuracy)
print('Testing Accuracy:', testing_accuracy)
```

```
Training Accuracy: 0.8909838046893884
Testing Accuracy: 0.8338164251207729
```

In [8]:
```
from sklearn.metrics import confusion_matrix

y_pred = knn_model.predict(x_test)
confusion_matrix(y_test, y_pred)
```

Out[8]:
```
array([[583, 156],
       [ 16, 280]], dtype=int64)
```

In [9]:
```
from sklearn.metrics import classification_report

y_pred = knn_model.predict(x_test)
print(classification_report(y_test, y_pred))
```

```
              precision    recall  f1-score   support

           0       0.97      0.79      0.87       739
           1       0.64      0.95      0.77       296

    accuracy                           0.83      1035
   macro avg       0.81      0.87      0.82      1035
weighted avg       0.88      0.83      0.84      1035
```

In [10]:
```
y_pred
```

Out[10]:
```
array([0, 0, 1, ..., 0, 1, 0], dtype=int64)
```

In [11]:
```
x_test
```

Out[11]:
```
array([[-0.48029848, -0.54419098, -0.2938948 , ..., -0.0562853 ,
        -0.32904848, -0.07097072],
       [ 0.5415109 ,  1.44874902,  4.31610679, ..., -0.0562853 ,
        -0.32904848, -0.07097072],
       [-0.48029848,  0.81939955, -0.1520486 , ..., -0.0562853 ,
         1.46955514, -0.07097072],
       ...,
       [-0.56544926, -0.64908256, -0.2938948 , ..., -0.0562853 ,
        -0.32904848, -0.07097072],
       [-0.56544926, -0.54419098, -0.2938948 , ..., -0.0562853 ,
        -0.32904848, -0.07097072],
       [ 2.07422496,  1.23896586,  1.47918273, ..., -0.0562853 ,
        -0.32904848, -0.07097072]])
```

2. Support Vector Machine (SVM)

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

```
svc_model = SVC()
svc_model.fit(x_train, y_train)
```

In [ ]:
```
training_accuracy = svc_model.score(x_train, y_train)
testing_accuracy = svc_model.score(x_test, y_test)
print('Training Accuracy:', training_accuracy)
print('Testing Accuracy:', testing_accuracy)
```

In [ ]:
```
from sklearn.metrics import confusion_matrix

y_pred = svc_model.predict(x_test)
confusion_matrix(y_test, y_pred)
```

In [ ]:
```
from sklearn.metrics import classification_report

y_pred = svc_model.predict(x_test)
print(classification_report(y_test, y_pred))
```