In [ ]:
```
# Classify the email using the binary classification method. Email Spam detection has two
# states: a) Normal State – Not Spam, b) Abnormal State – Spam. Use K-Nearest Neighbors and
# Support Vector Machine for classification. Analyze their performance.
# Dataset link: The emails.csv dataset on the Kaggle
# https://www.kaggle.com/datasets/balaka18/email-spam-classification-dataset-csv
```

In [5]:
```
# Import necessary libraries
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsClassifier
from sklearn.svm import SVC
from sklearn.metrics import accuracy_score, classification_report

# Load the dataset
data = pd.read_csv("emails.csv")  # Replace with the actual path to the dataset
data
```

Out[5]:

	Email No.	the	to	ect	and	for	of	a	you	hou	...	connevey	jay	valued	lay	infrastructure	military	allowing	ff	dry
0	Email 1	0	0	1	0	0	0	2	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
2	Email 3	0	0	1	0	0	0	8	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
4	Email 5	7	6	17	1	5	2	57	0	9	...	0	0	0	0	0	0	0	1	0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
5167	Email 5168	2	2	2	3	0	0	32	0	0	...	0	0	0	0	0	0	0	0	0
5168	Email 5169	35	27	11	2	6	5	151	4	3	...	0	0	0	0	0	0	0	1	0
5169	Email 5170	0	0	1	1	0	0	11	0	0	...	0	0	0	0	0	0	0	0	0
5170	Email 5171	2	7	1	0	2	1	28	2	0	...	0	0	0	0	0	0	0	1	0
5171	Email 5172	22	24	5	1	6	5	148	8	2	...	0	0	0	0	0	0	0	0	0

5172 rows × 3002 columns

In [15]:
```
# 1. Data Preprocessing - Handle missing values if necessary
data.drop(['Email No.'],axis=1, inplace=True)
# 2. Feature Selection/Engineering - Select relevant features
```

In [16]:
```
# 3. Split the data into training and testing sets
X = data.drop("Prediction", axis=1)  # Features
y = data["Prediction"]  # Target variable
print("Features: ",X)
print("Target: ",y)
```

```
Features:        the  to  ect  and  for  of    a  you  hou  in  ...  enhancements  \
0          0   0    1    0    0   0    2    0    0   0  ...             0
1          8  13   24    6    6   2  102    1   27  18  ...             0
2          0   0    1    0    0   0    8    0    0   4  ...             0
3          0   5   22    0    5   1   51    2   10   1  ...             0
4          7   6   17    1    5   2   57    0    9   3  ...             0
...      ...  ..  ...  ...  ...  ..  ...  ...  ...  ..  ...           ...
5167       2   2    2    3    0   0   32    0    0   5  ...             0
5168      35  27   11    2    6   5  151    4    3  23  ...             0
5169       0   0    1    1    0   0   11    0    0   1  ...             0
5170       2   7    1    0    2   1   28    2    0   8  ...             0
5171      22  24    5    1    6   5  148    8    2  23  ...             0

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

[5172 rows x 3000 columns]
Target:  0       0
1       0
2       0
3       0
4       0
       ..
5167    0
5168    0
5169    1
5170    1
5171    0
Name: Prediction, Length: 5172, dtype: int64
```

In [17]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)
```

In [18]:
```
# 4. Model Building
# K-Nearest Neighbors
knn_model = KNeighborsClassifier(n_neighbors=5)
knn_model.fit(X_train, y_train)

# Support Vector Machine
svm_model = SVC()
svm_model.fit(X_train, y_train)
```

Out[18]:
```
SVC()
```

In [19]:
```
# 5. Model Evaluation
# K-Nearest Neighbors
knn_predictions = knn_model.predict(X_test)
knn_accuracy = accuracy_score(y_test, knn_predictions)
knn_report = classification_report(y_test, knn_predictions)
```

```
C:\Users\rohit\anaconda3\lib\site-packages\sklearn\neighbors\_classification.py:228: FutureWarning: Unlike other reduction functions (e.g. `skew`, `kurtosis`), the default behavior of `mode` typically preserves the axis it acts along. In SciPy 1.11.0, this behavior will change: the default value of `keepdims` will become False, the `axis` over which the statistic is taken will be eliminated, and the value None will no longer be accepted. Set `keepdims` to True or False to avoid this warning.
  mode, _ = stats.mode(_y[neigh_ind, k], axis=1)
```

In [24]:
```
print(knn_predictions)
```

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

In [21]:
```
# Print or visualize the evaluation results
print("K-Nearest Neighbors Accuracy:")
print(knn_accuracy)
print("K-Nearest Neighbors Classification Report:")
print(knn_report)
```

```
K-Nearest Neighbors Accuracy:
0.8608247422680413
K-Nearest Neighbors Classification Report:
              precision    recall  f1-score   support

           0       0.93      0.87      0.90      1097
           1       0.73      0.83      0.78       455

    accuracy                           0.86      1552
   macro avg       0.83      0.85      0.84      1552
weighted avg       0.87      0.86      0.86      1552
```

In [20]:
```
# Support Vector Machine
svm_predictions = svm_model.predict(X_test)
svm_accuracy = accuracy_score(y_test, svm_predictions)
svm_report = classification_report(y_test, svm_predictions)
```

In [25]:
```
print(svm_predictions)
```

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

In [22]:
```
print("Support Vector Machine Accuracy:")
print(svm_accuracy)
print("Support Vector Machine Classification Report:")
print(svm_report)
```

```
Support Vector Machine Accuracy:
0.803479381443299
Support Vector Machine Classification Report:
              precision    recall  f1-score   support

           0       0.79      0.99      0.88      1097
           1       0.92      0.36      0.52       455

    accuracy                           0.80      1552
   macro avg       0.85      0.67      0.70      1552
weighted avg       0.83      0.80      0.77      1552
```

In [ ]: