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 [1]:
```
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsClassifier
from sklearn.svm import SVC
from sklearn.metrics import confusion_matrix, classification_report,accuracy_score
```

In [2]:
```
df=pd.read_csv("emails.csv") #Reading the Dataset
df.head()
```

Out[2]:

	Email No.	the	to	ect	and	for	of	a	you	hou	...	connevey	jay	valued	lay	infrastructure	military	allowing	ff	dry	Predict
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	

5 rows × 3002 columns

In [3]:
```
df.drop(columns=['Email No.'], inplace=True) #Dropping Email No. as it is irrelevant.
df.head()
```

Out[3]:

	the	to	ect	and	for	of	a	you	hou	in	...	connevey	jay	valued	lay	infrastructure	military	allowing	ff	dry	Prediction
0	0	0	1	0	0	0	2	0	0	0	...	0	0	0	0	0	0	0	0	0	0
1	8	13	24	6	6	2	102	1	27	18	...	0	0	0	0	0	0	0	1	0	0
2	0	0	1	0	0	0	8	0	0	4	...	0	0	0	0	0	0	0	0	0	0
3	0	5	22	0	5	1	51	2	10	1	...	0	0	0	0	0	0	0	0	0	0
4	7	6	17	1	5	2	57	0	9	3	...	0	0	0	0	0	0	0	1	0	0

5 rows × 3001 columns

In [4]:
```
#Splitting the Dataset
X=df.iloc[:, :df.shape[1]-1]
Y=df.iloc[:, -1]
X_train, X_test, Y_train, Y_test=train_test_split(X, Y, test_size=0.20, random_state=8)
```

In [5]:
```
def apply_model(model):#Model to print the scores of various models
    model.fit(X_train,Y_train)
    print("Training score = ",model.score(X_train,Y_train))
    print("Testing score = ",model.score(X_test,Y_test))
    print("Accuracy = ",model.score(X_test,Y_test))
    Y_pred = model.predict(X_test)
    print("Predicted values:\n",Y_pred)
```

```
    print("Confusion Matrix:\n",confusion_matrix(Y_test,Y_pred))
    print("Classification Report:\n",classification_report(Y_test,Y_pred))
```

In [6]:

```
knn = KNeighborsClassifier(n_neighbors=17) #KNN Model
apply_model(knn)
```

```
Training score =  0.883248730964467
Testing score =  0.8695652173913043
Accuracy =  0.8695652173913043
Predicted values:
 [0 0 0 ... 0 0 0]
Confusion Matrix:
 [[653  73]
 [ 62 247]]
Classification Report:
              precision    recall  f1-score   support

           0       0.91      0.90      0.91       726
           1       0.77      0.80      0.79       309

    accuracy                           0.87      1035
   macro avg       0.84      0.85      0.85      1035
weighted avg       0.87      0.87      0.87      1035
```

In [7]:

```
svm = SVC(kernel='linear',random_state=3,max_iter=10000) #SVM Model
apply_model(svm)
```

```
C:\Users\candr\anaconda3\lib\site-packages\sklearn\svm\_base.py:284: ConvergenceWarning: Solver terminated early (ma
x_iter=10000).  Consider pre-processing your data with StandardScaler or MinMaxScaler.
  warnings.warn(
Training score =  0.9951655789219241
Testing score =  0.9671497584541063
Accuracy =  0.9671497584541063
Predicted values:
 [0 1 0 ... 1 0 0]
Confusion Matrix:
 [[710  16]
 [ 18 291]]
Classification Report:
              precision    recall  f1-score   support

           0       0.98      0.98      0.98       726
           1       0.95      0.94      0.94       309

    accuracy                           0.97      1035
   macro avg       0.96      0.96      0.96      1035
weighted avg       0.97      0.97      0.97      1035
```