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.svm import SVC
from sklearn.neighbors import KNeighborsClassifier
from sklearn import metrics
```

In [2]:

```
df = pd.read_csv('emails.csv')
df
```

Out[2]:

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

5172 rows × 3002 columns

In [3]:

```
df.shape
```

Out[3]:

```
(5172, 3002)
```

In [4]:
```
df.isnull().any()
```

Out[4]:
```
Email No.     False
the           False
to            False
ect           False
and           False
              ...
military      False
allowing      False
ff            False
dry           False
Prediction    False
Length: 3002, dtype: bool
```

In [5]:
```
df.drop(columns='Email No.', inplace=True)
df
```

Out[5]:

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

5172 rows × 3001 columns

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

Out[6]:
```
Index(['the', 'to', 'ect', 'and', 'for', 'of', 'a', 'you', 'hou', 'in',
       ...
       'connevey', 'jay', 'valued', 'lay', 'infrastructure', 'military',
       'allowing', 'ff', 'dry', 'Prediction'],
      dtype='object', length=3001)
```

In [7]:
```
df.Prediction.unique()
```

Out[7]:
```
array([0, 1], dtype=int64)
```

In [8]:
```
df['Prediction'] = df['Prediction'].replace({0:'Not spam', 1:'Spam'})
```

In [9]:
```
df
```

Out[9]:

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

5172 rows × 3001 columns

KNN

In [10]:
```
X = df.drop(columns='Prediction',axis = 1)
Y = df['Prediction']
```

In [11]:
```
X.columns
```

Out[11]:
```
Index(['the', 'to', 'ect', 'and', 'for', 'of', 'a', 'you', 'hou', 'in',
       ...
       'enhancements', 'connevey', 'jay', 'valued', 'lay', 'infrastructur
e',
       'military', 'allowing', 'ff', 'dry'],
      dtype='object', length=3000)
```

In [12]:
```
Y.head()
```

Out[12]:
```
0    Not spam
1    Not spam
2    Not spam
3    Not spam
4    Not spam
Name: Prediction, dtype: object
```

In [13]:
```
x_train, x_test, y_train, y_test = train_test_split(X, Y, test_size=0.2, ra
```