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.

In [1]:
```
import pandas as pd
import numpy as np
```

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	infrastr
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 [3]:
```
df.columns
```

Out[3]:
```
Index(['Email No.', 'the', 'to', 'ect', 'and', 'for', 'of', 'a', 'you', 'hou',
       ...
       'connevey', 'jay', 'valued', 'lay', 'infrastructure', 'military',
       'allowing', 'ff', 'dry', 'Prediction'],
      dtype='object', length=3002)
```

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

Out[4]:
```
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 [5]:
```
df.dropna(inplace =True)
```

In [6]:
```
df.drop(['Email No.'],axis=1,inplace=True)
X=df.drop(['Prediction'],axis=1)
Y=df['Prediction']
```

In [13]:
```
from sklearn.preprocessing import scale
from sklearn.model_selection import train_test_split
X=scale(X)
X_train,X_test,Y_train,Y_test=train_test_split(X,Y,test_size=0.3,random_state=2)
```

KNN Classifier

In [14]:
```
from sklearn.neighbors import KNeighborsClassifier
knn=KNeighborsClassifier(n_neighbors=7)
knn.fit(X_train,Y_train)
y_pred = knn.predict(X_test)
print("Prediction",y_pred)
```

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

In [19]:
```
from sklearn import metrics
print("KNN Accuracy",metrics.accuracy_score(Y_test,y_pred))
```

```
KNN Accuracy 0.773840206185567
```

In [22]:
```
print("KNN Confusion Matrix : \n",metrics.confusion_matrix(Y_test,y_pred))
```

```
KNN Confusion Matrix :
 [[773 331]
 [ 20 428]]
```

SVM

In [23]:
```
from sklearn.svm import SVC
```

In [24]:
```
model=SVC(C=1)
model.fit(X_train,Y_train)
y_pred=model.predict(X_test)
print("Prediction : ",y_pred)
```

```
Prediction :  [0 0 1 ... 0 0 1]
```

In [25]:
```
print("SVM Accuracy",metrics.accuracy_score(Y_test,y_pred))
```

```
SVM Accuracy 0.9362113402061856
```

In [26]:
```
print("SVM Confusion Matrix : \n",metrics.confusion_matrix(Y_test,y_pred))
```

```
SVM Confusion Matrix :
 [[1096    8]
 [  91  357]]
```

In [ ]:
```

```