Name :

**Assignment 2**

Assignment No 2

1. 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 [19]:

**import pandas as pd import numpy as np**

**import seaborn as sns**

**import matplotlib.pyplot as plt**

%**matplotlib** inline

**import warnings**

warnings.filterwarnings('ignore')

**from sklearn.model\_selection import** train\_test\_split

**from sklearn.svm import** SVC

**from sklearn import** metrics

In [20]:

df=pd.read\_csv('emails.csv')

In [21]:

df.head()

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Out[21]: |  | | | | | | | | | | | | | | | | | | | |
|  |  | **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

In [22]:

Out[22]:

df.columns

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 [23]:

df.isnull().sum()

|  |  |  |
| --- | --- | --- |
| Out[23]: | 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 [24]:

df.dropna(inplace = **True**)

In [25]:

df.drop(['Email No.'],axis=1,inplace=**True**) X = df.drop(['Prediction'],axis = 1)

y = df['Prediction']

In [26]:

**KNN classifier**

**from sklearn.preprocessing import** scale X = scale(X)

*# split into train and test*

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size = 0.3, random\_state = 42)

|  |  |  |
| --- | --- | --- |
| In | [35]: | **from sklearn.neighbors import** KNeighborsClassifier knn = KNeighborsClassifier(n\_neighbors=7)  knn.fit(X\_train, y\_train) y\_pred = knn.predict(X\_test) |
|  |  |  |
| In | [36]: | print("Prediction",y\_pred) |
|  |  | Prediction [0 0 1 ... 1 1 1] |
| In | [37]: | print("KNN accuracy = ",metrics.accuracy\_score(y\_test,y\_pred)) |
|  |  | KNN accuracy = 0.8009020618556701 |
| In | [39]: | print("Confusion matrix",metrics.confusion\_matrix(y\_test,y\_pred)) |
|  |  | Confusion matrix [[804 293] |

[ 16 439]]

In [27]:

**SVM classifier**

*# cost C = 1*

model = SVC(C = 1)

*# fit*

model.fit(X\_train, y\_train)

*# predict*

y\_pred = model.predict(X\_test)

In [28]:

Out[28]:

In [29]:

metrics.confusion\_matrix(y\_true=y\_test, y\_pred=y\_pred)

array([[1091, 6],

[ 90, 365]])

print("SVM accuracy = ",metrics.accuracy\_score(y\_test,y\_pred))

SVM accuracy = 0.9381443298969072