Madhur Jaripatke

Roll No. 52

BE A Computer

RMDSSOE, Warje, Pune

Classify emails 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

Importing Libraries

In [1]:
```
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.preprocessing import scale
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsClassifier
from sklearn.svm import SVC
from sklearn import metrics
```

Loading The Dataset

In [2]:
```
df = pd.read_csv('./Datasets/emails.csv')
df.head()
```

Out[2]:

	Email No.	the	to	ect	and	for	of	a	you	hou	...	connevey	jay	valued
0	Email 1	0	0	1	0	0	0	2	0	0	...	0	0	0
1	Email 2	8	13	24	6	6	2	102	1	27	...	0	0	0
2	Email 3	0	0	1	0	0	0	8	0	0	...	0	0	0
3	Email 4	0	5	22	0	5	1	51	2	10	...	0	0	0
4	Email 5	7	6	17	1	5	2	57	0	9	...	0	0	0

5 rows × 3002 columns

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

Out[3]:
```
Index(['Email No.', 'the', 'to', 'ect', 'and', 'for', 'of', 'a', 'you', 'ho
u',
       ...
       '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
```

Data Preprocessing

In [5]:
```
df.dropna(inplace=True)
df.drop(['Email No.'], axis=1, inplace=True)
x = df.drop(['Prediction'], axis = 1)
y = df['Prediction']
```

In [6]:
```
x = scale(x)
# Split into train and test data
```

```
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.3,
                                                    random_state=42)
```

Model Building

In [7]:
```
knn = KNeighborsClassifier(n_neighbors=7)
knn.fit(x_train, y_train)
y_pred = knn.predict(x_test)
```

In [8]:
```
model = SVC(C = 1)
# Fit model
model.fit(x_train, y_train)
# Predict
y_pred = model.predict(x_test)
```

In [9]:
```
cs = metrics.confusion_matrix(y_true=y_test, y_pred=y_pred)
print('Confusion Matrix:\n',cs)
```

```
Confusion Matrix:
 [[1091    6]
 [  90  365]]
```

Results

In [10]:
```
print('Predictions:\n', y_pred)
print('\nKNN accuracy = \n', metrics.accuracy_score(y_test,y_pred))
print('\nConfusion matrix:\n', metrics.confusion_matrix(y_test,y_pred))
print('\nSVM accuracy = \n', metrics.accuracy_score(y_test,y_pred))
```

```
Predictions:
 [0 0 1 ... 0 0 1]

KNN accuracy =
 0.9381443298969072

Confusion matrix:
 [[1091    6]
 [  90  365]]

SVM accuracy =
 0.9381443298969072
```