Machine Learning Ass. 2

BE_34_Samruddhi Khairnar

Classify the email using the binary classification method.

Email Spam detection has two states:

a) Normal State - Not Spam,

b) Abnormal State - Spam.

Dataset link: https://www.kaggle.com/datasets/balaka18/email-spam-classification-dataset-csv (https://www.kaggle.com/datasets/balaka18/email-spam-classification-dataset-csv)

In [175]:
```
from google.colab import drive
drive.mount('/content/drive')
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, c
all drive.mount("/content/drive", force_remount=True).
```

In [176]:
```
%cd /content/drive/MyDrive/BE_Datasets/
```

```
/content/drive/MyDrive/BE_Datasets
```

In [177]:
```
import pandas as pd
df = pd.read_csv('emails.csv')
```

In [178]:
```
df.head()
```

Out[178]:

	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	0.0	0.0	
1	Email 2	8	13	24	6	6	2	102	1	27	...	0.0	0.0	0.0	0.0	
2	Email 3	0	0	1	0	0	0	8	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	
4	Email 5	7	6	17	1	5	2	57	0	9	...	0.0	0.0	0.0	0.0	

5 rows × 3002 columns

In [179]:
```
df.shape
```
Out[179]:
```
(518, 3002)
```

In [180]:
```
df.isna().sum().sum()
```
Out[180]:
```
1345
```

In [181]:
```
df.fillna(df.mean() ,inplace=True, axis=0)
```
```
<ipython-input-181-3a76dfeab0bf>:1: FutureWarning: The default value of nu
meric_only in DataFrame.mean is deprecated. In a future version, it will d
efault to False. In addition, specifying 'numeric_only=None' is deprecate
d. Select only valid columns or specify the value of numeric_only to silen
ce this warning.
  df.fillna(df.mean() ,inplace=True, axis=0)
```

In [182]:
```
df.isna().sum().sum()
```
Out[182]:
```
0
```

In [183]:
```
df = df.astype({'Prediction':'int'})
```

Use K-Nearest Neighbors and Support Vector Machine for classification.

In [184]:
```
from sklearn.model_selection import train_test_split
```

In [185]:
```
train, test = train_test_split(df, test_size = 0.2, shuffle = True)
X_train, y_train = train.drop(['Prediction', 'Email No.'], axis='columns').
X_test, y_test = test.drop(['Prediction', 'Email No.'], axis='columns').val
```

In [186]:
```
y_test[0]
```
Out[186]:
```
0
```

In [187]:
```
from sklearn.neighbors import KNeighborsClassifier
knn = KNeighborsClassifier()
knn.fit(X_train, y_train)
```
Out[187]:
```
KNeighborsClassifier()
```
In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [188]:
```
from sklearn.svm import SVC
svc = SVC()
svc.fit(X_train, y_train)
```

Out[188]:
```
SVC()
```

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

Analyze their performance.

In [189]:
```
svc.predict(X_test)
```

Out[189]:
```
array([0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0])
```

In [190]:
```
svc.score(X_test, y_test)
```

Out[190]:
```
0.7788461538461539
```

In [191]:
```
knn.predict(X_test)
```

Out[191]:
```
array([0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 1, 0, 1,
       1, 0, 0, 0, 1, 1, 1, 0, 0, 0, 0, 1, 0, 0, 1, 0, 1, 1, 1, 0, 0, 0,
       0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0])
```

In [192]:
```
knn.score(X_test, y_test)
```

Out[192]:
```
0.8846153846153846
```