

Assignment 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>

```
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
```

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

```
df.head()
```

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

	valued	lay	infrastructure	military	allowing	ff	dry
0	0	0	0	0	0	0	0
1	0	0	0	0	0	1	0
2	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0
4	0	0	0	0	0	1	0

```
[5 rows x 3002 columns]
```

```

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)

df.isnull().sum()
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

df.dropna(inplace = True)

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

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)

```

##KNN classifier

```

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 ... 1 1 1]

print("KNN accuracy = ",metrics.accuracy_score(y_test,y_pred))

```

```
KNN accuracy = 0.8009020618556701
print("Confusion matrix",metrics.confusion_matrix(y_test,y_pred))
Confusion matrix [[804 293]
 [ 16 439]]
```

bold text## SVM classifier

```
# cost C = 1
model = SVC(C = 1)

# fit
model.fit(X_train, y_train)

# predict
y_pred = model.predict(X_test)

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
```