

Course: Laboratory Practice III

Course Code: 410246

Name: Asmeeta Lalit Bardiya

Class: BE

Roll No. : 22

Div: A

Title: 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 numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.model_selection import GridSearchCV
from sklearn.svm import SVC
```

```
data = pd.read_csv('/content/emails (1).csv')
```

```
data
```

	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
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
5167	Email 5168	2	2	2	3	0	0	32	0	0	...	0	0	0
5168	Email	35	27	11	2	6	5	151	4	3	...	0	0	0

```
data.head()
```

	Email No.	the	to	ect	and	for	of	a	you	hou	...	connevey	jay	valued	1
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	

Email

```
data.tail()
```

	Email No.	the	to	ect	and	for	of	a	you	hou	...	connevey	jay	valued	
5167	Email 5168	2	2	2	3	0	0	32	0	0	...	0	0	0	
5168	Email 5169	35	27	11	2	6	5	151	4	3	...	0	0	0	
5169	Email 5170	0	0	1	1	0	0	11	0	0	...	0	0	0	

Email

```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5172 entries, 0 to 5171
Columns: 3002 entries, Email No. to Prediction
dtypes: int64(3001), object(1)
memory usage: 118.5+ MB
```

```
data.describe()
```

	the	to	ect	and	for	of
<b>count</b>	5172.000000	5172.000000	5172.000000	5172.000000	5172.000000	5172.000000
<b>mean</b>	6.640565	6.188128	5.143852	3.075599	3.124710	2.627030
<b>std</b>	11.745009	9.534576	14.101142	6.045970	4.680522	6.229845

```
data.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
```

```
data.drop('Email No.',axis=1,inplace=True)
```

```
data=data.dropna()
```

```
data.head()
```

	the	to	ect	and	for	of	a	you	hou	in	...	connevey	jay	valued	lay
<b>0</b>	0	0	1	0	0	0	2	0	0	0	...	0	0	0	0
<b>1</b>	8	13	24	6	6	2	102	1	27	18	...	0	0	0	0
<b>2</b>	0	0	1	0	0	0	8	0	0	4	...	0	0	0	0
<b>3</b>	0	5	22	0	5	1	51	2	10	1	...	0	0	0	0
<b>4</b>	7	6	17	1	5	2	57	0	9	3	...	0	0	0	0

5 rows × 3001 columns

```
x=data.drop('Prediction',axis=1)
y=data['Prediction']
```

```
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.3,random_state=0)
```

```
from sklearn.neighbors import KNeighborsClassifier
model=KNeighborsClassifier(n_neighbors=5)
model.fit(x_train,y_train)
```

```
predicted= model.predict(x_test)
```

```
print("Predicited Results: ",predicted)
```

```
Predicited Results:  [0 0 0 ... 0 1 1]
```

```
from sklearn import metrics
```

```
from sklearn.metrics import confusion_matrix, accuracy_score,recall_score,precision_score
```

```
confusion_matrix=metrics.confusion_matrix(y_test,predicted)
```

```
print(confusion_matrix)
```

```
[[969 142]
 [ 79 362]]
```

```
print("Accuracy score of K Neighbours Classifier")
```

```
print(metrics.accuracy_score(y_test,predicted))
```

```
Accuracy score of K Neighbours Classifier
0.8576030927835051
```

```
ps=precision_score(y_test,predicted)
```

```
print("Precision score of K neighbours Classifier is:")
```

```
print(ps)
```

```
print("Recall score of K Neighbours classifier is:")
```

```
rs=recall_score(y_test,predicted)
```

```
print(rs)
```

```
Precision score of K neighbours Classifier is:
```

```
0.7182539682539683
```

```
Recall score of K Neighbours classifier is:
```

```
0.8208616780045351
```

```
model2=SVC()
```

```
model2.fit(x_train,y_train)
```

```
prediction2=model2.predict(x_test)
```

```
confusion_matrix2=metrics.confusion_matrix(y_test,prediction2)
```

```
print(confusion_matrix2)
```

```
[[1086  25]
 [ 284 157]]
```

```
print("Predicited Results: ",prediction2)
```

```
Predicited Results:  [0 0 0 ... 0 0 0]
```

```
print("accuracy score: ",metrics.accuracy_score(y_test,prediction2))
```

```
accuracy score: 0.8009020618556701
```

```
print("Precision score of Support Vector Machine(SVM) is:")
```

```
print(ps)
```

```
print("Recall score of Support Vector Machine(SVM) is:")
```

```
rs=recall_score(y_test,prediction2)
```

```
print(rs)
```

```
Precision score of Support Vector Machine(SVM) is:
```

```
0.7182539682539683
```

```
Recall score of Support Vector Machine(SVM) is:
```

```
0.35600907029478457
```

```
print(data.corr)
```

```
<bound method DataFrame.corr of
0      0      0      1      0      0      0      2      0      0      0      ...      0      0
1      8     13     24      6      6      2    102      1     27     18      ...      0      0
2      0      0      1      0      0      0      8      0      0      4      ...      0      0
3      0      5     22      0      5      1     51      2     10      1      ...      0      0
4      7      6     17      1      5      2     57      0      9      3      ...      0      0
...     ...     ..     ...     ...     ...     ..     ...     ...     ...     ..     ...     ...     ...
5167     2      2      2      3      0      0     32      0      0      5      ...      0      0
5168    35     27     11      2      6      5    151      4      3     23      ...      0      0
5169     0      0      1      1      0      0     11      0      0      1      ...      0      0
5170     2      7      1      0      2      1     28      2      0      8      ...      0      0
5171    22     24      5      1      6      5    148      8      2     23      ...      0      0
```

```
      valued  lay  infrastructure  military  allowing  ff  dry  Prediction
0          0    0                0          0          0  0  0          0
1          0    0                0          0          0  1  0          0
2          0    0                0          0          0  0  0          0
3          0    0                0          0          0  0  0          0
4          0    0                0          0          0  1  0          0
...     ...     ...                ...          ...          ...  ..  ...          ...
5167     0    0                0          0          0  0  0          0
5168     0    0                0          0          0  1  0          0
5169     0    0                0          0          0  0  0          1
5170     0    0                0          0          0  1  0          1
5171     0    0                0          0          0  0  0          0
```

```
[5172 rows x 3001 columns]>
```

