

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
from sklearn.model_selection import train_test_split
from sklearn.svm import SVC
from sklearn.neighbors import KNeighborsClassifier
from sklearn import metrics

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

	convey	Email No.	the	to	ect	and	for	of	a	you	hou	...
0		Email 1	0	0	1	0	0	0	2	0	0	...
1		Email 2	8	13	24	6	6	2	102	1	27	...
2		Email 3	0	0	1	0	0	0	8	0	0	...
3		Email 4	0	5	22	0	5	1	51	2	10	...
4		Email 5	7	6	17	1	5	2	57	0	9	...
...	
5167		Email 5168	2	2	2	3	0	0	32	0	0	...
5168		Email 5169	35	27	11	2	6	5	151	4	3	...
5169		Email 5170	0	0	1	1	0	0	11	0	0	...
5170		Email 5171	2	7	1	0	2	1	28	2	0	...
5171		Email 5172	22	24	5	1	6	5	148	8	2	...
0												
	jay	valued	lay	infrastructure		military		allowing	ff	dry	\	
0	0	0	0		0		0		0	0	0	
1	0	0	0		0		0		0	1	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		0	1	0	
...	
5167	0	0	0		0		0		0	0	0	

5168	0	0	0	0	0	0	1	0
5169	0	0	0	0	0	0	0	0
5170	0	0	0	0	0	0	1	0
5171	0	0	0	0	0	0	0	0

Prediction

0	0
1	0
2	0
3	0
4	0
...	...
5167	0
5168	0
5169	1
5170	1
5171	0

[5172 rows x 3002 columns]

df.shape

(5172, 3002)

df.isnull().any()

Email No.	False
the	False
to	False
ect	False
and	False

...	...
military	False
allowing	False
ff	False
dry	False
Prediction	False

Length: 3002, dtype: bool

df.drop(columns='Email No.', inplace=True)

df

	the	to	ect	and	for	of	a	you	hou	in	...	connevey
jay \												
0	0	0	1	0	0	0	2	0	0	0	...	0
1	8	13	24	6	6	2	102	1	27	18	...	0
2	0	0	1	0	0	0	8	0	0	4	...	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]

df.columns
Index(['the', 'to', 'ect', 'and', 'for', 'of', 'a', 'you', 'hou',
      'in',
      ...,
      'connevey', 'jay', 'valued', 'lay', 'infrastructure',
      'military',

```


5167	0	0	0	0	0	0	0	0	Not
spam									
5168	0	0	0	0	0	1	0	0	Not
spam									
5169	0	0	0	0	0	0	0	0	
Spam									
5170	0	0	0	0	0	1	0	0	
Spam									
5171	0	0	0	0	0	0	0	0	Not
spam									

[5172 rows x 3001 columns]

KNN

```
X = df.drop(columns='Prediction',axis = 1)
```

```
Y = df['Prediction']
```

```
X.columns
```

```
Index(['the', 'to', 'ect', 'and', 'for', 'of', 'a', 'you', 'hou',  
'in',
```

```
...,  
      'enhancements', 'connevey', 'jay', 'valued', 'lay',  
'infrastructure',  
      'military', 'allowing', 'ff', 'dry'],  
      dtype='object', length=3000)
```

```
Y.head()
```

```
0    Not spam  
1    Not spam  
2    Not spam  
3    Not spam  
4    Not spam
```

```
Name: Prediction, dtype: object
```

```
x_train, x_test, y_train, y_test = train_test_split(X, Y,  
test_size=0.2, random_state=1)
```

```
KN = KNeighborsClassifier
```

```
knn = KN(n_neighbors=7)
```

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

```
y_pred = knn.predict(x_test)
```

```
print("Prediction: \n")
```

```
print(y_pred)
```

Prediction:

```
['Not spam' 'Spam' 'Not spam' ... 'Not spam' 'Not spam' 'Not spam']
```

Accuracy

```
M = metrics.accuracy_score(y_test,y_pred)
```

```
print("KNN accuracy: ", M)
```

```
KNN accuracy:  0.8714975845410629
```

```
C = metrics.confusion_matrix(y_test,y_pred)
```

```
print("Confusion matrix: ", C)
```

```
Confusion matrix:  [[635  84]
 [ 49 267]]
```

SVM Classifier

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

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

```
y_pred = model.predict(x_test)    # predict
```

```
kc = metrics.confusion_matrix(y_test, y_pred)
```

```
print("SVM accuracy: ", kc)
```

```
SVM accuracy:  [[700  19]
 [189 127]]
```