

Importing necessary Libraries and loading dataset

```
import pandas as pd
import numpy as np

import kagglehub

# Download latest version
path = kagglehub.dataset_download("balaka18/email-spam-classification-dataset-csv")

print("Path to dataset files:", path)

Downloading from https://www.kaggle.com/api/v1/datasets/download/balaka18/email-spam-classification-dataset-csv?dataset_version_number=1...
100%|██████████| 1.66M/1.66M [00:00<00:00, 103MB/s]Extracting files...
Path to dataset files: /root/.cache/kagglehub/datasets/balaka18/email-spam-classification-dataset-csv/versions/1

df = pd.read_csv(path + "/emails.csv")

import os
print(os.listdir(path))

['emails.csv']
```

Preprocessing of DATA

```
df.head(2)
```

	Email No.	the	to	ect	and	for	of	a	you	hou	...	connevey	jay	valued	lay	infrastructure	military	allowing	ff	dry	Prediction
0	Email 1	0	0	1	0	0	0	2	0	0	...	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	1	0	0

2 rows × 3002 columns

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

df.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5172 entries, 0 to 5171
Columns: 3001 entries, the to Prediction
dtypes: int64(3001)
memory usage: 118.4 MB

df.describe()
```

	the	to	ect	and	for	of	a	you	hou	in	...	connevey	jay	valued	
count	5172.000000	5172.000000	5172.000000	5172.000000	5172.000000	5172.000000	5172.000000	5172.000000	5172.000000	5172.000000	...	5172.000000	5172.000000	5172.000000	51
mean	6.640565	6.188128	5.143852	3.075599	3.124710	2.627030	55.517401	2.466551	2.024362	10.600155	...	0.005027	0.012568	0.010634	
std	11.745009	9.534576	14.101142	6.045970	4.680522	6.229845	87.574172	4.314444	6.967878	19.281892	...	0.105788	0.199682	0.116693	
min	0.000000	0.000000	1.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	...	0.000000	0.000000	0.000000	
25%	0.000000	1.000000	1.000000	0.000000	1.000000	0.000000	12.000000	0.000000	0.000000	1.000000	...	0.000000	0.000000	0.000000	
50%	3.000000	3.000000	1.000000	1.000000	2.000000	1.000000	28.000000	1.000000	0.000000	5.000000	...	0.000000	0.000000	0.000000	
75%	8.000000	7.000000	4.000000	3.000000	4.000000	2.000000	62.250000	3.000000	1.000000	12.000000	...	0.000000	0.000000	0.000000	
max	210.000000	132.000000	344.000000	89.000000	47.000000	77.000000	1898.000000	70.000000	167.000000	223.000000	...	4.000000	7.000000	2.000000	
8 rows × 3001 columns															

df.shape

(5172, 3001)

df.isnull().sum()	
	0
the	0
to	0
ect	0
and	0
for	0
...	...
military	0
allowing	0
ff	0
dry	0
Prediction	0
3001 rows × 1 columns	
dtype: int64	

df['Prediction'].value_counts()

	count
Prediction	
0	3672
1	1500

dtype: int64

Spliting Data

```
y = df['Prediction']
```

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

```
from sklearn.model_selection import train_test_split
```

```
x_train,x_test,y_train,y_test = train_test_split(X,y,test_size=0.2,random_state=0)
```

```
from sklearn.neighbors import KNeighborsClassifier
```

```
KNN = KNeighborsClassifier(n_neighbors=2)
```

Training K Nearest Neighbours Algorithm

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

```
▼ KNeighborsClassifier ⓘ ?
KNeighborsClassifier(n_neighbors=2)
```

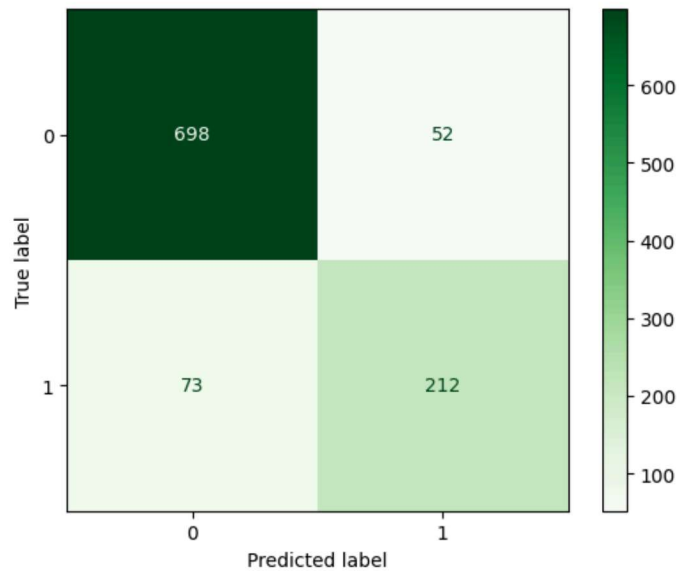
```
y_pred = KNN.predict(x_test)
```

```
from sklearn.metrics import accuracy_score
accuracy = accuracy_score(y_test,y_pred)
print("Accuracy :",accuracy)
```

```
Accuracy : 0.8792270531400966
```

```
import matplotlib.pyplot as plt
```

```
from sklearn.metrics import confusion_matrix, ConfusionMatrixDisplay
cm = confusion_matrix(y_test,y_pred)
disp = ConfusionMatrixDisplay(confusion_matrix=cm)
disp.plot(cmap='Greens')
plt.show()
```



```
from sklearn.metrics import classification_report
print(classification_report(y_test,y_pred))
```

	precision	recall	f1-score	support
0	0.91	0.93	0.92	750
1	0.80	0.74	0.77	285
accuracy			0.88	1035
macro avg	0.85	0.84	0.85	1035
weighted avg	0.88	0.88	0.88	1035

Training Support Vector Machines Classifier Algorithm

```
from sklearn.svm import SVC
```

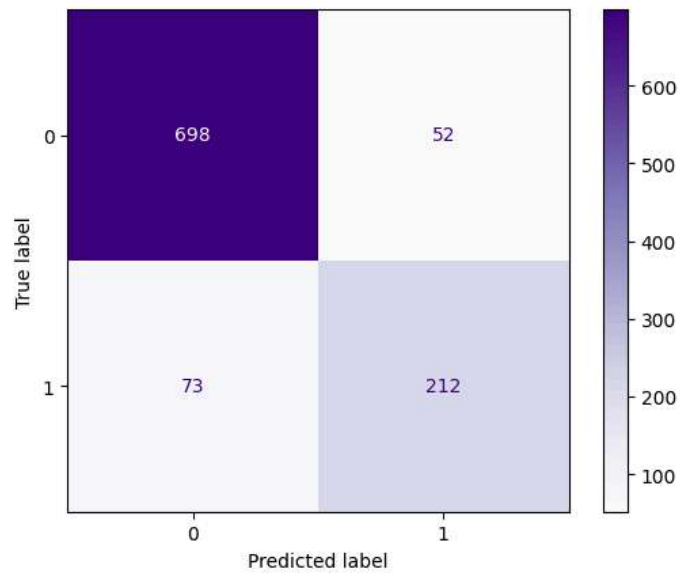
```
svc = SVC(kernel='linear')
svc.fit(x_train,y_train)
```

```
▼ SVC ⓘ ?
SVC(kernel='linear')
```

```
accuracy = svc.score(x_test,y_test)
print("Accuracy :",accuracy)
```

```
Accuracy : 0.9439613526570049
```

```
cf = confusion_matrix(y_test,y_pred)
disp = ConfusionMatrixDisplay(confusion_matrix=cf)
disp.plot(cmap='Purples')
plt.show()
```



```
print(classification_report(y_test,y_pred))
```

	precision	recall	f1-score	support
0	0.91	0.93	0.92	750
1	0.80	0.74	0.77	285
accuracy			0.88	1035
macro avg	0.85	0.84	0.85	1035
weighted avg	0.88	0.88	0.88	1035

Start coding or [generate](#) with AI.