```
#ADITYA WAVHALE
#BEB59

import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt

from sklearn.metrics import classification_report, confusion_matrix, ConfusionMatrixDisplay, precisio

df = pd.read_csv("emails.csv")

df
```

	Email No.	the	to	ect	and	for	of	a	you	hou	...	connevey	jay	valued	lay	infrastructure	military	allowing	ff	dry
0	Email 1	0	0	1	0	0	0	2	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
2	Email 3	0	0	1	0	0	0	8	0	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	0
4	Email 5	7	6	17	1	5	2	57	0	9	...	0	0	0	0	0	0	0	1	0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
5167	Email 5168	2	2	2	3	0	0	32	0	0	...	0	0	0	0	0	0	0	0	0
5168	Email 5169	35	27	11	2	6	5	151	4	3	...	0	0	0	0	0	0	0	1	0

```
df = df.drop('Email No.', axis=1)

df.shape
```

```
(5172, 3001)
```

```
df.describe()
```

	the	to	ect	and	for	of	a	you	hou	in
count	5172.000000	5172.000000	5172.000000	5172.000000	5172.000000	5172.000000	5172.000000	5172.000000	5172.000000	5172.000000
mean	6.640565	6.188128	5.143852	3.075599	3.124710	2.627030	55.517401	2.466551	2.024362	10.600155
std	11.745009	9.534576	14.101142	6.045970	4.680522	6.229845	87.574172	4.314444	6.967878	19.281892
min	0.000000	0.000000	1.000000	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
50%	3.000000	3.000000	1.000000	1.000000	2.000000	1.000000	28.000000	1.000000	0.000000	5.000000
75%	8.000000	7.000000	4.000000	3.000000	4.000000	2.000000	62.250000	3.000000	1.000000	12.000000
max	210.000000	132.000000	344.000000	89.000000	47.000000	77.000000	1898.000000	70.000000	167.000000	223.000000

8 rows × 3001 columns

```
df.describe()
```

What can I help you build?

	the	to	ect	and	for	of	a	you	hou	in
count	5172.000000	5172.000000	5172.000000	5172.000000	5172.000000	5172.000000	5172.000000	5172.000000	5172.000000	5172.000000
mean	6.640565	6.188128	5.143852	3.075599	3.124710	2.627030	55.517401	2.466551	2.024362	10.600155
std	11.745009	9.534576	14.101142	6.045970	4.680522	6.229845	87.574172	4.314444	6.967878	19.281892
min	0.000000	0.000000	1.000000	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
50%	3.000000	3.000000	1.000000	1.000000	2.000000	1.000000	28.000000	1.000000	0.000000	5.000000
75%	8.000000	7.000000	4.000000	3.000000	4.000000	2.000000	62.250000	3.000000	1.000000	12.000000
max	210.000000	132.000000	344.000000	89.000000	47.000000	77.000000	1898.000000	70.000000	167.000000	223.000000

8 rows × 3001 columns

```
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['Prediction'].value_counts()
```

Prediction	count
0	3672
1	1500

dtype: int64

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

from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.20)

from sklearn.neighbors import KNeighborsClassifier
neigh = KNeighborsClassifier(n_neighbors = 2)
neigh.fit(X_train, y_train)
```

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

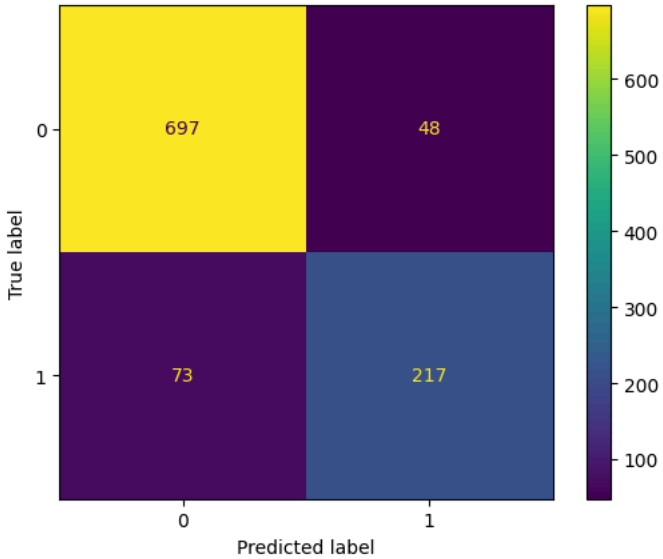
```
y_pred = neigh.predict(X_test)

neigh.score(X_train, y_train)
neigh.score(X_test, y_test)
```

```
0.8830917874396135
```

```
print("Confusion Matrix: ")
cm = confusion_matrix(y_test, y_pred)
cm
```

```
Confusion Matrix:
array([[697,  48],
       [ 73, 217]])
```

```
mat = ConfusionMatrixDisplay(confusion_matrix = cm)
mat.plot()
plt.show()
```




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

```
              precision    recall  f1-score   support

           0       0.91      0.94      0.92       745
           1       0.82      0.75      0.78       290

    accuracy                           0.88      1035
   macro avg       0.86      0.84      0.85      1035
weighted avg       0.88      0.88      0.88      1035
```

```
print("accuracy_score: ")
accuracy_score(y_test, y_pred)
```

```
accuracy_score:
0.8830917874396135
```

```
print("precision_score: ")
precision_score(y_test, y_pred)
```

```
precision_score:
0.8188679245283019
```

```
print("recall_score: ")
recall_score(y_test, y_pred)
```

```
recall_score:
0.7482758620689656
```

```
print("Error: ")
1-accuracy_score(y_test, y_pred)
```

```
Error:
0.11690821256038653
```

```
from sklearn.svm import SVC
SVM = SVC(gamma = 'auto')
SVM.fit(X_train, y_train)
```

```
SVC
SVC(gamma='auto')
```

```
y_pred = SVM.predict(X_test)
```

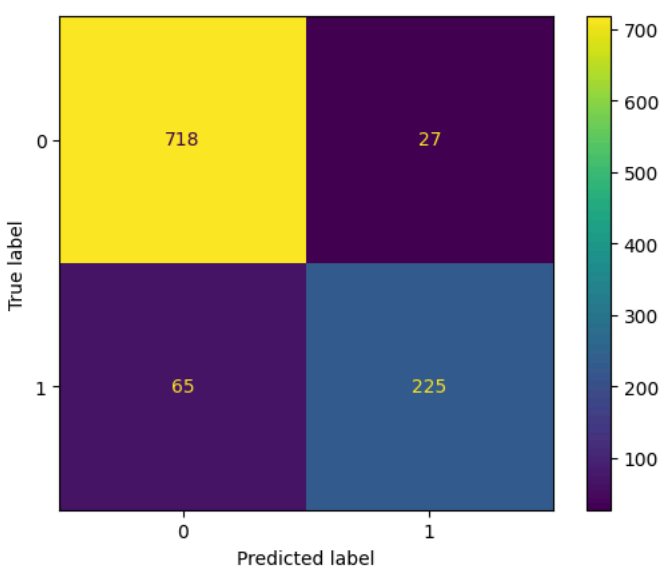
```
SVM.score(X_train, y_train)
SVM.score(X_test, y_test)
```

```
0.9111111111111111
```

```
print("Confusion Matrix: ")
cm = confusion_matrix(y_test, y_pred)
cm
```

```
Confusion Matrix:
array([[718,  27],
       [ 65, 225]])
```

```
mat = ConfusionMatrixDisplay(confusion_matrix = cm)
mat.plot()
plt.show()
```




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

```
              precision    recall  f1-score   support

           0       0.92      0.96      0.94       745
           1       0.89      0.78      0.83       290

    accuracy                           0.91      1035
   macro avg       0.90      0.87      0.89      1035
weighted avg       0.91      0.91      0.91      1035
```

Start coding or generate with AI.