In [1]:
```
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
```

In [2]:
```
data = pd.read_csv("emails.csv")
data
```

Out[2]:

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

5172 rows × 3002 columns

In [3]:
```
data = data.drop('Email No.', axis=1)
```

In [4]:
```
data.shape
```

Out[4]:
```
(5172, 3001)
```

In [5]:
```
data.describe()
```

Out[5]:

	the	to	ect	and	for	of	
count	5172.000000	5172.000000	5172.000000	5172.000000	5172.000000	5172.000000	517
mean	6.640565	6.188128	5.143852	3.075599	3.124710	2.627030	5
std	11.745009	9.534576	14.101142	6.045970	4.680522	6.229845	8

	the	to	ect	and	for	of	
min	0.000000	0.000000	1.000000	0.000000	0.000000	0.000000	
25%	0.000000	1.000000	1.000000	0.000000	1.000000	0.000000	1
50%	3.000000	3.000000	1.000000	1.000000	2.000000	1.000000	2
75%	8.000000	7.000000	4.000000	3.000000	4.000000	2.000000	6
max	210.000000	132.000000	344.000000	89.000000	47.000000	77.000000	189

In [6]:
```
data.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
```

In [7]:
```
data['Prediction'].value_counts()
```

Out[7]:
```
0    3672
1    1500
Name: Prediction, dtype: int64
```

In [8]:
```
X = data.drop('Prediction', axis = 1)
y = data['Prediction']
```

In [9]:
```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size
```

In [10]:
```
from sklearn.neighbors import KNeighborsClassifier
neigh = KNeighborsClassifier(n_neighbors = 2)
neigh.fit(X_train, y_train)
```

Out[10]:
```
KNeighborsClassifier(n_neighbors=2)
```

In [11]:
```
y_pred = neigh.predict(X_test)
```

In [12]:
```
neigh.score(X_train, y_train)
neigh.score(X_test, y_test)
```

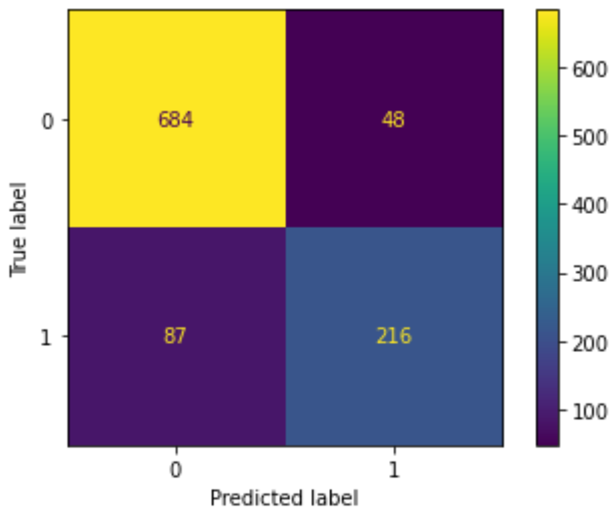
Out[12]:
```
0.8695652173913043
```

In [13]:
```
print("Confusion Matrix: ")
cm = confusion_matrix(y_test, y_pred)
cm
```

```
Confusion Matrix:
```

Out[13]:
```
array([[684,  48],
       [ 87, 216]], dtype=int64)
```

In [14]:
```
mat = ConfusionMatrixDisplay(confusion_matrix = cm)
mat.plot()
plt.show()
```



In [15]:
```
print(classification_report(y_test, y_pred))
```

```
              precision    recall  f1-score   support

           0       0.89      0.93      0.91       732
           1       0.82      0.71      0.76       303

    accuracy                           0.87      1035
   macro avg       0.85      0.82      0.84      1035
weighted avg       0.87      0.87      0.87      1035
```

In [16]:
```
print("accuracy_score: ")
accuracy_score(y_test, y_pred)
```

```
accuracy_score:
```

Out[16]:
```
0.8695652173913043
```

In [17]:
```
print("precision_score: ")
precision_score(y_test, y_pred)
```

```
precision_score:
```

Out[17]:
```
0.8181818181818182
```

In [18]:
```
print("recall_score: ")
recall_score(y_test, y_pred)
```

```
recall_score:
```

Out[18]:
```
0.7128712871287128
```

In [19]:
```
print("Error: ")
1-accuracy_score(y_test, y_pred)
```

```
Error:
```

Out[19]:
```
0.13043478260869568
```

In [23]:
```
from sklearn.svm import SVC
SVM = SVC(gamma = 'auto')
```

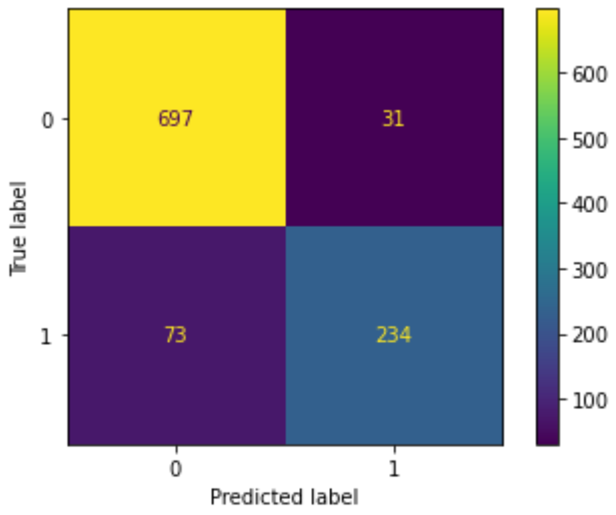
```
        SVM.fit(X_train, y_train)
Out[23]: SVC(gamma='auto')
```

```
In [22]: y_pred = SVC.predict(y_test)

         ---------------------------------------------------------------------------
         TypeError                                 Traceback (most recent call last)
         <ipython-input-22-dcf354cc4f53> in <module>
         ----> 1 y_pred = SVC.predict(y_test)

         TypeError: predict() missing 1 required positional argument: 'X'
```

```
In [44]: SVM.score(X_train, y_train)
         SVM.score(X_test, y_test)
Out[44]: 0.8995169082125604
```

```
In [45]: print("Confusion Matrix: ")
         cm = confusion_matrix(y_test, y_pred)
         cm
         Confusion Matrix:

Out[45]: array([[697,  31],
                [ 73, 234]], dtype=int64)
```

```
In [46]: mat = ConfusionMatrixDisplay(confusion_matrix = cm)
         mat.plot()
         plt.show()
```



```
In [47]: print(classification_report(y_test, y_pred))

                       precision    recall  f1-score   support

                    0       0.91      0.96      0.93       728
                    1       0.88      0.76      0.82       307

             accuracy                           0.90      1035
            macro avg       0.89      0.86      0.87      1035
         weighted avg       0.90      0.90      0.90      1035
```

```
In [ ]:
```