In [1]:
```
import pandas as pd
```

In [2]:
```
df = pd.read_csv('emails.csv')
```

In [3]:
```
df.shape
```

Out[3]:
```
(5172, 3002)
```

In [6]:
```
df.head()
```

Out[6]:

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

5 rows × 3002 columns

In [8]:
```
df.isnull().sum()
```

Out[8]:
```
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
```

In [11]:
```
# input data
x = df.drop(['Email No.','Prediction'], axis = 1)

# output data
y = df['Prediction']
```

In [12]:
```
x.shape
```

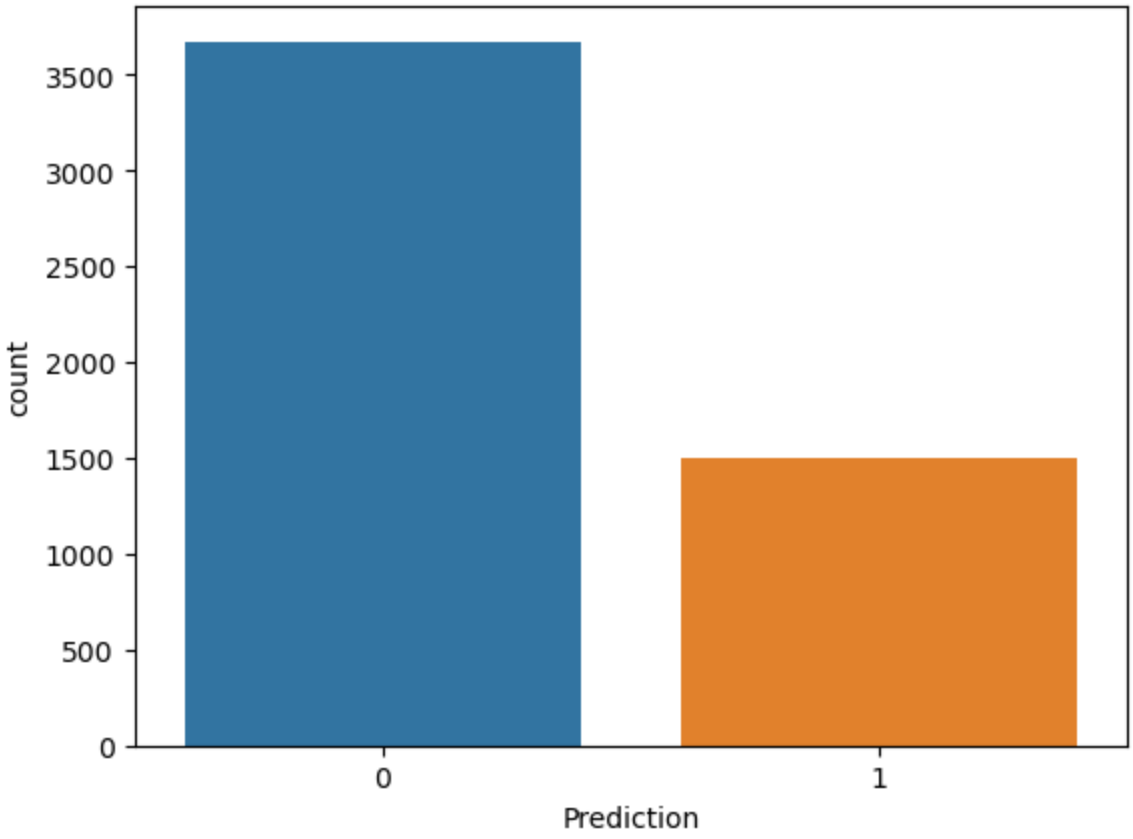
Out[12]:
```
(5172, 3000)
```

In [13]:
```
set(x.dtypes)
```

Out[13]:
```
{dtype('int64')}
```

In [14]:
```
import seaborn as sns
sns.countplot(x = y)
```

Out[14]:
```
<Axes: xlabel='Prediction', ylabel='count'>
```



In [15]:
```
y.value_counts()
```

Out[15]:
```
0    3672
1    1500
Name: Prediction, dtype: int64
```

In [16]:
```
# feautre scaling
from sklearn.preprocessing import MinMaxScaler
scaler = MinMaxScaler()
x_scaled = scaler.fit_transform(x)
```

In [17]:
```
x_scaled
```

Out[17]:
```
array([[0.        , 0.        , 0.        , ..., 0.        , 0.        ,
        0.        ],
       [0.03809524, 0.09848485, 0.06705539, ..., 0.        , 0.00877193,
        0.        ],
       [0.        , 0.        , 0.        , ..., 0.        , 0.        ,
        0.        ],
       ...,
       [0.        , 0.        , 0.        , ..., 0.        , 0.        ,
        0.        ],
       [0.00952381, 0.0530303 , 0.        , ..., 0.        , 0.00877193,
        0.        ],
       [0.1047619 , 0.18181818, 0.01166181, ..., 0.        , 0.        ,
        0.        ]])
```

In [19]:
```
# cross validation
from sklearn.model_selection import train_test_split as tts
xtrain,xtest,ytrain,ytest = tts(x_scaled,y, test_size=0.25)
```

In [20]:
```
x_scaled.shape
```

Out[20]: (5172, 3000)

In [21]:
```
xtrain.shape
```

Out[21]: (3879, 3000)

In [22]:
```
xtest.shape
```

Out[22]: (1293, 3000)

In [24]:
```
# import the class
from sklearn.neighbors import KNeighborsClassifier as knc
```

In [25]:
```
# create the object
knn = knc(n_neighbors = 5)
```

In [26]:
```
# train the algorithm
knn.fit(xtrain, ytrain)
```

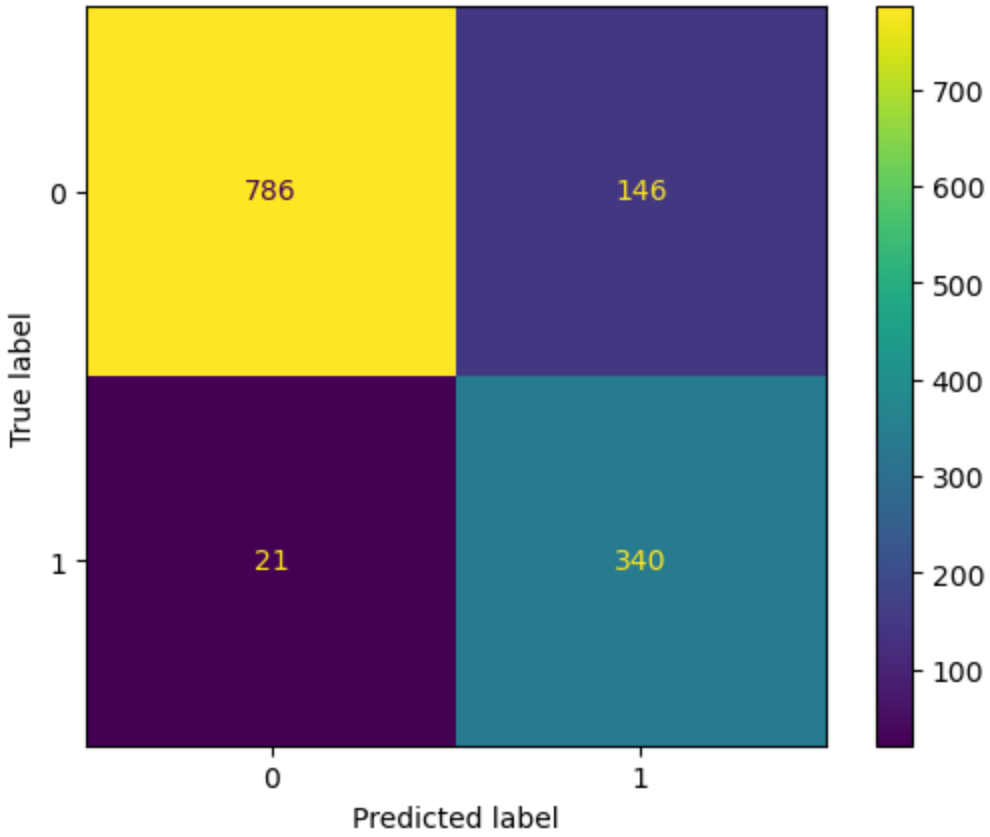
Out[26]:
```
▾ KNeighborsClassifier

KNeighborsClassifier()
```

In [27]:
```
# predict in test data
y_pred = knn.predict(xtest)
```

In [31]:
```
# import the evaluation metrics
from sklearn.metrics import ConfusionMatrixDisplay as cmd
from sklearn.metrics import accuracy_score as score
from sklearn.metrics import classification_report as report
```

In [33]:
```
cmd.from_predictions(ytest, y_pred)
```

Out[33]: <sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay at 0x229f76af970>



In [36]:
```
ytest.value_counts()
```

Out[36]:
```
0    932
1    361
Name: Prediction, dtype: int64
```

In [37]:
```
score(ytest, y_pred)
```

Out[37]: 0.8708430007733952

In [38]:
```
print(report(ytest, y_pred))
```

```
              precision    recall  f1-score   support

           0       0.97      0.84      0.90       932
           1       0.70      0.94      0.80       361

    accuracy                           0.87      1293
   macro avg       0.84      0.89      0.85      1293
weighted avg       0.90      0.87      0.88      1293
```

In [39]:
```
import numpy as np
import matplotlib.pyplot as plt
```

In [42]:
```
error = []
i = 1
for k in range(1, 40):
```

```
    knn = knc(n_neighbors = k)
    knn.fit(xtrain, ytrain)
    pred = knn.predict(xtest)
    error.append(np.mean(pred != ytest))
```

In [43]:
```
error
```

Out[43]:
```
[0.10827532869296211,
 0.0951276102088167,
 0.12374323279195669,
 0.10904872389791183,
 0.12915699922660479,
 0.12606341840680588,
 0.15081206496519722,
 0.15081206496519722,
 0.177107501933488,
 0.16937354988399073,
 0.19721577726218098,
 0.1902552204176334,
 0.21191028615622584,
 0.2088167053364269,
 0.22583139984532097,
 0.2242846094354215,
 0.23588553750966745,
 0.23588553750966745,
 0.24361948955916474,
 0.24439288476411447,
 0.2614075792730085,
 0.262954369682908,
 0.2776488785769528,
 0.27842227378190254,
 0.2861562258313998,
 0.2877030162412993,
 0.29466357308584684,
 0.2962103634957463,
 0.3062645011600928,
 0.3039443155452436,
 0.3109048723897912,
 0.308584686774942,
 0.31554524361948955,
 0.31322505800464034,
 0.3186388244392885,
 0.31786542923433875,
 0.32559938128383603,
 0.32559938128383603,
 0.33101314771848417]
```

In [60]:
```
knn = knc(n_neighbors = 2)
```

In [61]:
```
knn.fit(xtrain,ytrain)
```

Out[61]:
```
KNeighborsClassifier
KNeighborsClassifier(n_neighbors=2)
```

In [62]:
```
y_pred = knn.predict(xtest)
```

In [63]:
```
score(ytest, y_pred)
```

Out[63]:
```
0.9048723897911833
```

In [64]:
```
# SVM MODEL
```

In [65]:
```
from sklearn.svm import SVC
```

In [66]:
```
svm = SVC(kernel='linear')
```

In [67]:
```
svm.fit(xtrain,ytrain)
```

Out[67]:
```
SVC(kernel='linear')
```

In [68]:
```
y_pred = svm.predict(xtest)
```

In [69]:
```
score(ytest, y_pred)
```

Out[69]:
```
0.9651972157772621
```

In [72]:
```
svm = SVC(kernel='rbf')
```

In [73]:
```
svm.fit(xtrain, ytrain)
```

Out[73]:
```
SVC()
```

In [74]:
```
y_pred = svm.predict(xtest)
```

In [75]:
```
score(ytest, y_pred)
```

Out[75]:
```
0.9443155452436195
```

In [76]:
```
svm = SVC(kernel='poly')
```

In [77]:
```
svm.fit(xtrain, ytrain)
```

Out[77]:
```
SVC(kernel='poly')
```

In [78]:
```
y_pred = svm.predict(xtest)
```

In [79]:
```
score(ytest, y_pred)
```

Out[79]:
```
0.7648878576952823
```

In [80]:
```
svm = SVC(kernel='sigmoid')
```

In [81]:
```
svm.fit(xtrain, ytrain)
```

Out[81]:
```
SVC(kernel='sigmoid')
```

In [82]:
```
y_pred = svm.predict(xtest)
```

In [83]:
```
score(ytest, y_pred)
```

Out[83]:
```
0.8329466357308585
```

In [ ]:
```

```