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

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

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

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

In [4]:
```
df.head()
```

Out[4]:

	Email No.	the	to	ect	and	for	of	a	you	hou	...	connevey	jay	valued	lay	infrastructu
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 [5]:
```
#input data
x=df.drop(['Email No.','Prediction'],axis=1)
#output data
y=df['Prediction']
```

In [6]:
```
x.shape
```

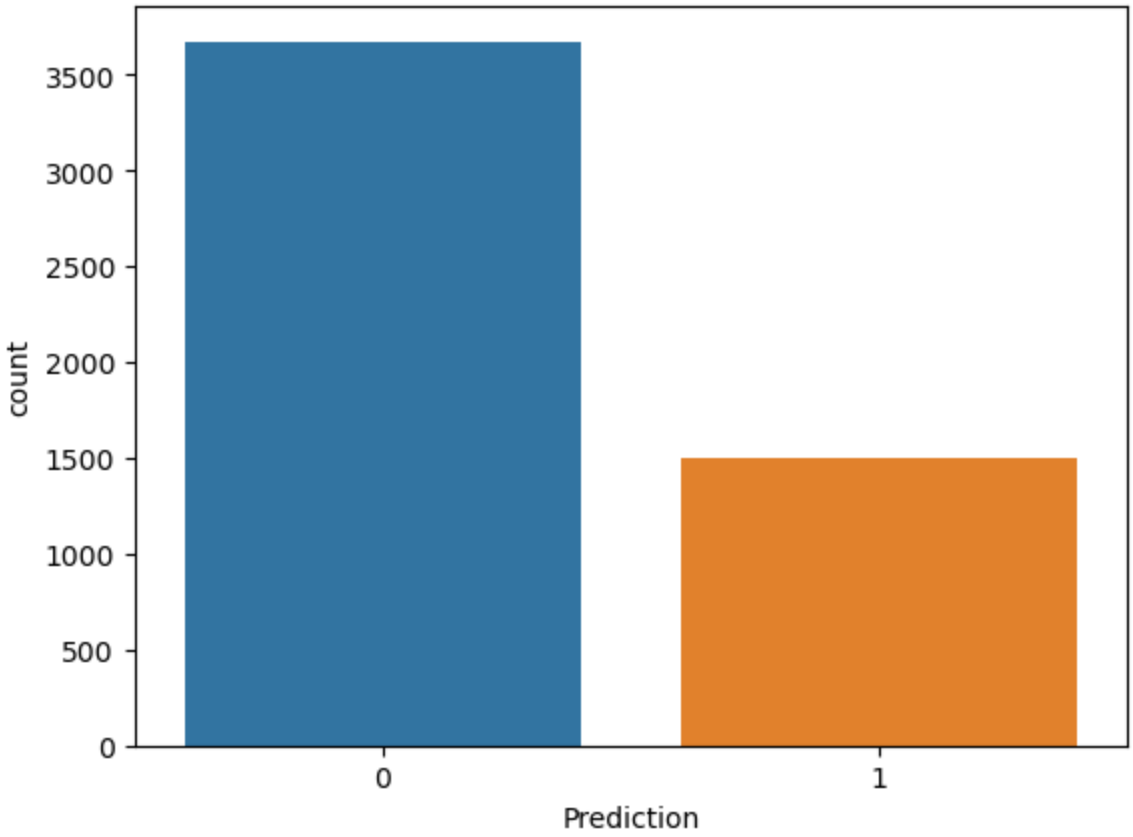
Out[6]:
```
(5172, 3000)
```

In [8]:
```
set(x.dtypes)
```

Out[8]:
```
{dtype('int64')}
```

In [9]:
```
import seaborn as sns
sns.countplot(x=y)
```

Out[9]: <Axes: xlabel='Prediction', ylabel='count'>



In [10]:
```
y.value_counts()
```

Out[10]:
```
Prediction
0    3672
1    1500
Name: count, dtype: int64
```

In [11]:
```
#feature scale
from sklearn.preprocessing import MinMaxScaler
scaler=MinMaxScaler()
x_scaled=scaler.fit_transform(x)
```

In [12]:
```
x_scaled
```

Out[12]:
```
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 [13]:
```
#cross validation
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(
x_scaled,y,random_state=0,test_size=0.25)
```

In [14]:
```
x_scaled.shape
```

Out[14]:
```
(5172, 3000)
```

In [15]:
```
x_train.shape
```

Out[15]:
```
(3879, 3000)
```

In [16]:
```
x_test.shape
```

Out[16]:
```
(1293, 3000)
```

In [35]:
```
#import the class
from sklearn.neighbors import KNeighborsClassifier
from sklearn import metrics
```

In [36]:
```
#create object
knn=KNeighborsClassifier(n_neighbors=5)
```

In [21]:
```
knn.fit(x_train,y_train)
```

Out[21]:
```
KNeighborsClassifier()
```

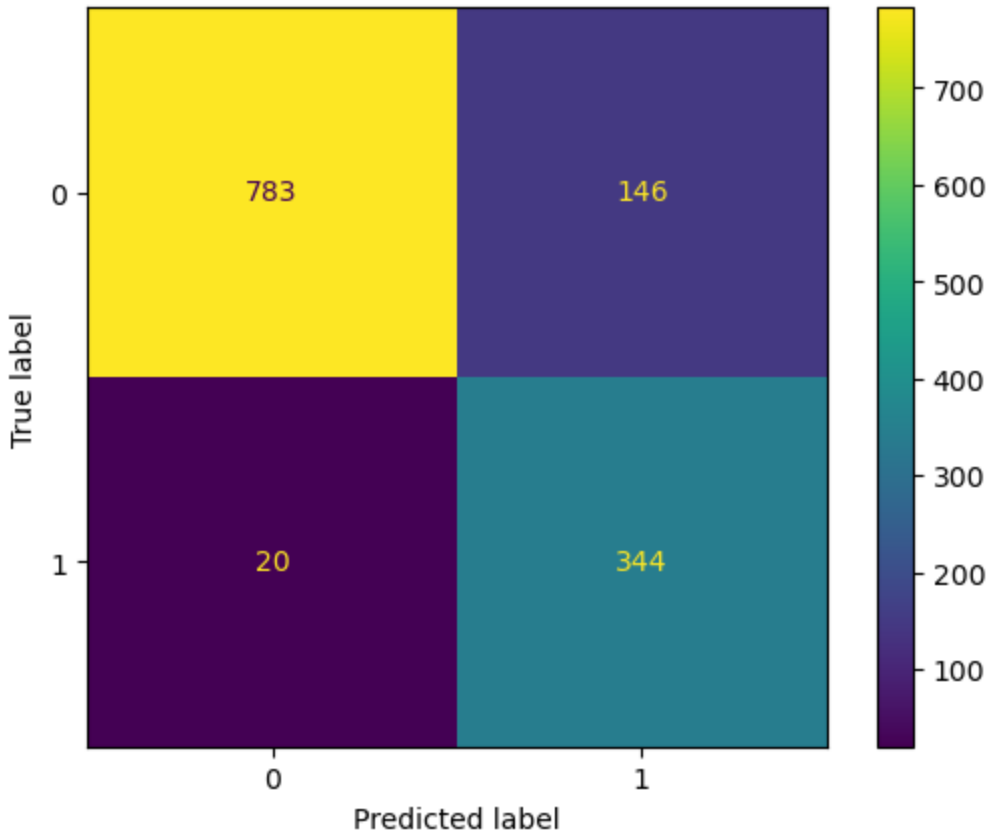
In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [22]:
```
#predict on test data
y_pred=knn.predict(x_test)
```

In [23]:
```
#import the evaluation metrices
from sklearn.metrics import ConfusionMatrixDisplay,accuracy_score
from sklearn.metrics import classification_report
```

In [24]:
```
ConfusionMatrixDisplay.from_predictions(y_test,y_pred)
```

Out[24]:
```
<sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay at 0x1fa69404a
90>
```



In [25]:
```
y_test.value_counts()
```

Out[25]:
```
Prediction
0    929
1    364
Name: count, dtype: int64
```

In [26]:
```
accuracy_score(y_test,y_pred)
```

Out[26]:
```
0.871616395978345
```

In [29]:
```
print(classification_report(y_test,y_pred))
```

```
              precision    recall  f1-score   support

           0       0.98      0.84      0.90       929
           1       0.70      0.95      0.81       364

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

In [37]:
```
metrics.confusion_matrix(y_test,y_pred)
```

Out[37]:
```
array([[911,  18],
       [ 12, 352]], dtype=int64)
```

In [39]:
```
from sklearn.svm import SVC
```

In [40]:
```
svm=SVC(kernel='linear')
```

In [41]:
```
svm.fit(x_train,y_train)
```

Out[41]:
```
SVC(kernel='linear')
```

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [42]:
```
y_pred=svm.predict(x_test)
```

In [43]:
```
accuracy_score(y_test,y_pred)
```

Out[43]:
```
0.9767981438515081
```

In [44]:
```
#Linear SVM:0.9767981438515081
metrics.confusion_matrix(y_test,y_pred)
```

Out[44]:
```
array([[911,  18],
       [ 12, 352]], dtype=int64)
```

In [ ]:
```

```