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

In [2]:
```
df=pd.read_csv('C:/shubhangi/2023-24/LP-III_ML/Assignment 2/emails.csv')
```

In [3]:
```
df.head()
```

Out[3]:

	Email No.	the	to	ect	and	for	of	a	you	hou	...	connevey	jay	valued	lay	infrastructure	military
0	Email 1	0	0	1	0	0	0	2	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
2	Email 3	0	0	1	0	0	0	8	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
4	Email 5	7	6	17	1	5	2	57	0	9	...	0	0	0	0	0	0

5 rows × 3002 columns

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

Out[4]:
```
(5172, 3002)
```

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 [7]:
```
x.dtypes
```

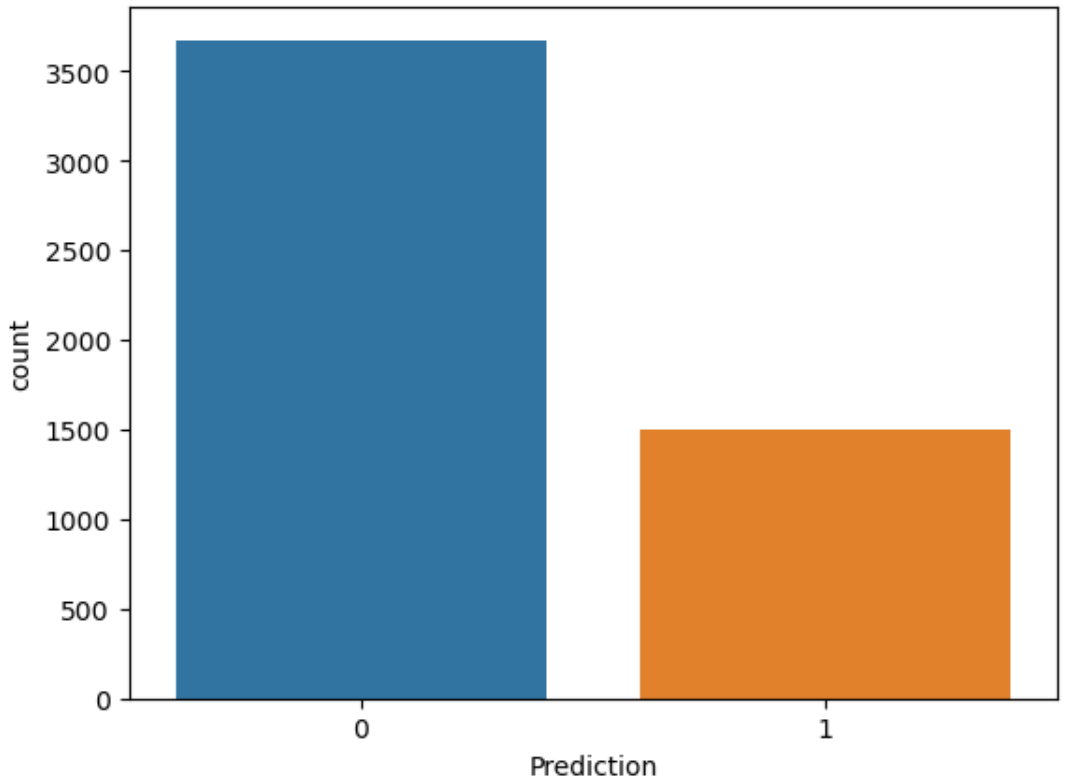
Out[7]:
```
the               int64
to                int64
ect               int64
and               int64
for               int64
                  ...
infrastructure    int64
military          int64
allowing          int64
ff                int64
dry               int64
Length: 3000, dtype: object
```

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

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

In [10]:
```
import seaborn as sns
sns.countplot(x=y)
```

Out[10]:
```
<Axes: xlabel='Prediction', ylabel='count'>
```



In [11]:
```
y.value_counts()
```

Out[11]:
```
0    3672
1    1500
Name: Prediction, dtype: int64
```

In [14]:
```
pip install scikit-learn
```

```
Requirement already satisfied: scikit-learn in c:\programdata\anaconda3\lib\site-packa
ges (1.2.2)
Requirement already satisfied: numpy>=1.17.3 in c:\programdata\anaconda3\lib\site-pack
ages (from scikit-learn) (1.24.3)
Requirement already satisfied: scipy>=1.3.2 in c:\programdata\anaconda3\lib\site-packa
ges (from scikit-learn) (1.10.1)
Requirement already satisfied: joblib>=1.1.1 in c:\programdata\anaconda3\lib\site-pack
ages (from scikit-learn) (1.2.0)
Requirement already satisfied: threadpoolctl>=2.0.0 in c:\programdata\anaconda3\lib\si
te-packages (from scikit-learn) (2.2.0)
Note: you may need to restart the kernel to use updated packages.
```

In [15]:
```
from sklearn.preprocessing import MinMaxScaler
```

In [16]:
```
scaler=MinMaxScaler()
```

In [17]:
```
x_scaled=scaler.fit_transform(x)
```

In [18]:
```
x_scaled
```

Out[18]:
```
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
```

In [20]:
```
x_train,x_test,y_train,y_test=train_test_split(x_scaled,y,random_state=0,test_size=0.25
```

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

Out[22]:
```
(5172, 3000)
```

In [23]:
```
x_train.shape
```

Out[23]:
```
(3879, 3000)
```

In [24]:
```
x_test.shape
```

Out[24]:
```
(1293, 3000)
```

In [26]:
```
from sklearn.neighbors import  KNeighborsClassifier
```

In [27]:
```
knn=KNeighborsClassifier(n_neighbors=5)
```

In [28]:
```
knn.fit(x_train,y_train)
```

Out[28]:
```
KNeighborsClassifier

KNeighborsClassifier()
```

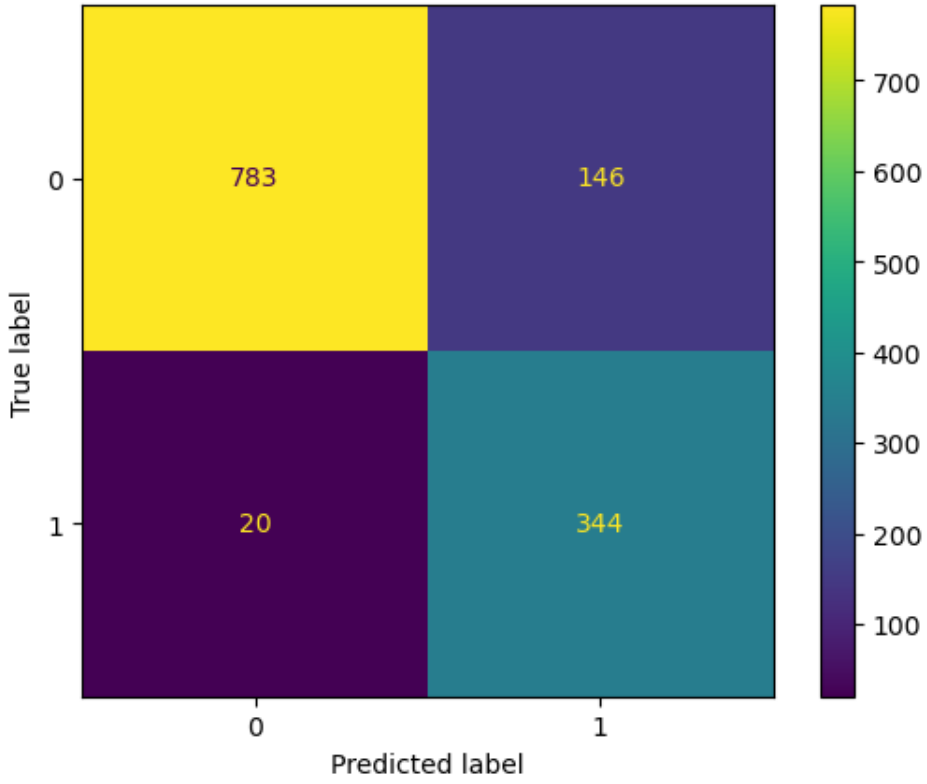
In [29]:
```
#predict on test data
y_pred=knn.predict(x_test)
```

In [31]:
```
#import evaluation matrix
from sklearn.metrics import ConfusionMatrixDisplay,accuracy_score
```

In [32]:
```
from sklearn.metrics import classification_report
```

In [33]:
```
ConfusionMatrixDisplay.from_predictions(y_test,y_pred)
```

Out[33]:
```
<sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay at 0x207c27b6c50>
```



In [34]:
```
y_test.value_counts()
```

Out[34]:
```
0    929
1    364
Name: Prediction, dtype: int64
```

In [35]:
```
accuracy_score(y_test,y_pred)
```

Out[35]:
```
0.871616395978345
```

In [36]:
```
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]:
```
import numpy as np
import matplotlib.pyplot as plt
```

In [40]:
```
error = []
for k in range(1,41):
    knn=  KNeighborsClassifier(n_neighbors=k)
    knn.fit(x_train,y_train)
    pred=knn.predict(x_test)
    error.append(np.mean(pred!=y_test))
```

In [39]:
```
error
```

Out[39]:
```
[0.10827532869296211,
 0.10982211910286156,
 0.12296983758700696,
 0.11523588553750967,
 0.12838360402165508,
 0.1214230471771075,
 0.15158546017014696,
 0.14849187935034802,
 0.17246713070378963,
 0.16705336426914152,
 0.1871616395978345,
 0.18329466357308585,
 0.21500386697602475,
 0.21345707656612528,
 0.22815158546017014,
 0.2266047950502707,
 0.23588553750966745,
 0.23356535189481825,
 0.2459396751740139,
 0.24361948955916474,
 0.2559938128383604,
 0.2552204176334107,
 0.2699149265274555,
 0.2691415313225058,
 0.2822892498066512,
 0.28306264501160094,
 0.2954369682907966,
 0.2923433874709977,
 0.3039443155452436,
 0.300077339520495,
 0.30549110595514306,
 0.30549110595514306,
 0.31245166279969067,
 0.31245166279969067,
 0.3194122196442382,
 0.317092034029389,
 0.32637277648878577,
 0.32559938128383603,
 0.33410672853828305,
 0.3325599381283836]
```

In [41]:
```
knn=KNeighborsClassifier(n_neighbors=1)
```

In [42]:
```
knn.fit(x_train,y_train)
```

Out[42]:
```
KNeighborsClassifier
KNeighborsClassifier(n_neighbors=1)
```

In [43]:
```
y_pred=knn.predict(x_test)
```

In [44]:
```
accuracy_score(y_test,y_pred)
```

Out[44]:
```
0.8917246713070379
```

In [45]:
```
from sklearn.svm import SVC
```

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

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

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

In [54]:
```
accuracy_score(y_test,y_pred)
```

Out[54]:
```
0.9767981438515081
```

In [ ]:
```
#linear 0.9767981438515081
```