Practical 2

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]:
```
<bound method NDFrame.head of         Email No.  the  to  ect  and  for  of    a  y
ou  hou  ...  connevey  \
0         Email 1    0   0    1    0    0   0    2    0    0  ...          0
1         Email 2    8  13   24    6    6   2  102    1   27  ...          0
2         Email 3    0   0    1    0    0   0    8    0    0  ...          0
3         Email 4    0   5   22    0    5   1   51    2   10  ...          0
4         Email 5    7   6   17    1    5   2   57    0    9  ...          0
...           ...  ...  ..  ...  ...  ...  ..  ...  ...  ...  ...        ...
5167   Email 5168    2   2    2    3    0   0   32    0    0  ...          0
5168   Email 5169   35  27   11    2    6   5  151    4    3  ...          0
5169   Email 5170    0   0    1    1    0   0   11    0    0  ...          0
5170   Email 5171    2   7    1    0    2   1   28    2    0  ...          0
5171   Email 5172   22  24    5    1    6   5  148    8    2  ...          0

      jay  valued  lay  infrastructure  military  allowing  ff  dry  \
0       0       0    0               0         0         0   0    0
1       0       0    0               0         0         0   1    0
2       0       0    0               0         0         0   0    0
3       0       0    0               0         0         0   0    0
4       0       0    0               0         0         0   1    0
...   ...     ...  ...             ...       ...       ...  ..  ...
5167    0       0    0               0         0         0   0    0
5168    0       0    0               0         0         0   1    0
5169    0       0    0               0         0         0   0    0
5170    0       0    0               0         0         0   1    0
5171    0       0    0               0         0         0   0    0

      Prediction
0              0
1              0
2              0
3              0
4              0
...          ...
5167           0
5168           0
5169           1
5170           1
5171           0

[5172 rows x 3002 columns]>
```

In [5]:
```
x=df.drop(['Email No.','Prediction'],axis=1)
y=df['Prediction']
```

In [7]:
```
x.shape
```

Out[7]:
```
(5172, 3000)
```

In [8]:
```
x.info
```

Out[8]:
```
<bound method DataFrame.info of        the  to  ect  and  for  of       a  you  hou  i
n  ...  enhancements  \
0         0   0    1    0    0   0    2    0    0   0  ...             0
1         8  13   24    6    6   2  102    1   27  18  ...             0
2         0   0    1    0    0   0    8    0    0   4  ...             0
3         0   5   22    0    5   1   51    2   10   1  ...             0
4         7   6   17    1    5   2   57    0    9   3  ...             0
...     ...  ..  ...  ...  ...  ..  ...  ...  ...  ..  ...           ...
5167      2   2    2    3    0   0   32    0    0   5  ...             0
5168     35  27   11    2    6   5  151    4    3  23  ...             0
5169      0   0    1    1    0   0   11    0    0   1  ...             0
5170      2   7    1    0    2   1   28    2    0   8  ...             0
5171     22  24    5    1    6   5  148    8    2  23  ...             0

      connevey   jay  valued   lay  infrastructure   military   allowing  ff  dry
0            0    0       0    0                0          0          0   0    0
1            0    0       0    0                0          0          0   1    0
2            0    0       0    0                0          0          0   0    0
3            0    0       0    0                0          0          0   0    0
4            0    0       0    0                0          0          0   1    0
...        ...  ...     ...  ...              ...        ...        ...  ..  ...
5167         0    0       0    0                0          0          0   0    0
5168         0    0       0    0                0          0          0   1    0
5169         0    0       0    0                0          0          0   0    0
5170         0    0       0    0                0          0          0   1    0
5171         0    0       0    0                0          0          0   0    0

[5172 rows x 3000 columns]>
```

In [9]:
```
x.dtypes
```

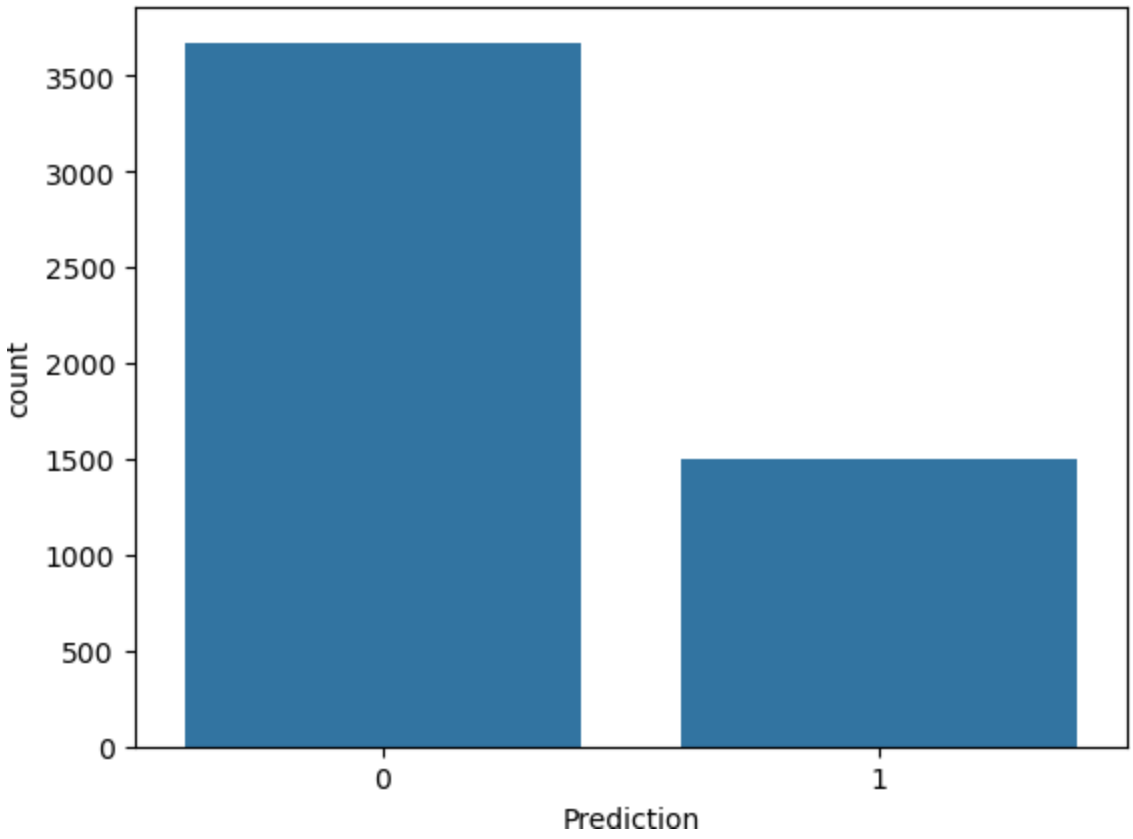
Out[9]:
```
the               int64
to                int64
ect               int64
and               int64
for               int64
                  ...
infrastructure    int64
military          int64
allowing          int64
ff                int64
dry               int64
Length: 3000, dtype: object
```

In [10]:
```
set(x.dtypes)
```

Out[10]:
```
{dtype('int64')}
```

In [11]:
```
import seaborn as sns
sns.countplot(x=y)
```

Out[11]:
```
<Axes: xlabel='Prediction', ylabel='count'>
```



In [12]:
```
y.value_counts()
```

Out[12]:
```
Prediction
0    3672
1    1500
Name: count, dtype: int64
```

In [13]:
```
from sklearn.preprocessing import MinMaxScaler
scaler=MinMaxScaler()
x_scaled=scaler.fit_transform(x)
```

In [14]:
```
x_scaled
```

Out[14]:
```
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 [15]:
```
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=
```

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

Out[16]:
```
(5172, 3000)
```

In [17]:
```
x_train.shape
```

Out[17]:
```
(3620, 3000)
```

In [18]:
```
x_test.shape
```

Out[18]:
```
(1552, 3000)
```

In [21]:
```
from sklearn.neighbors import KNeighborsClassifier
knn=KNeighborsClassifier(n_neighbors=5)
knn.fit(x_train,y_train)
```

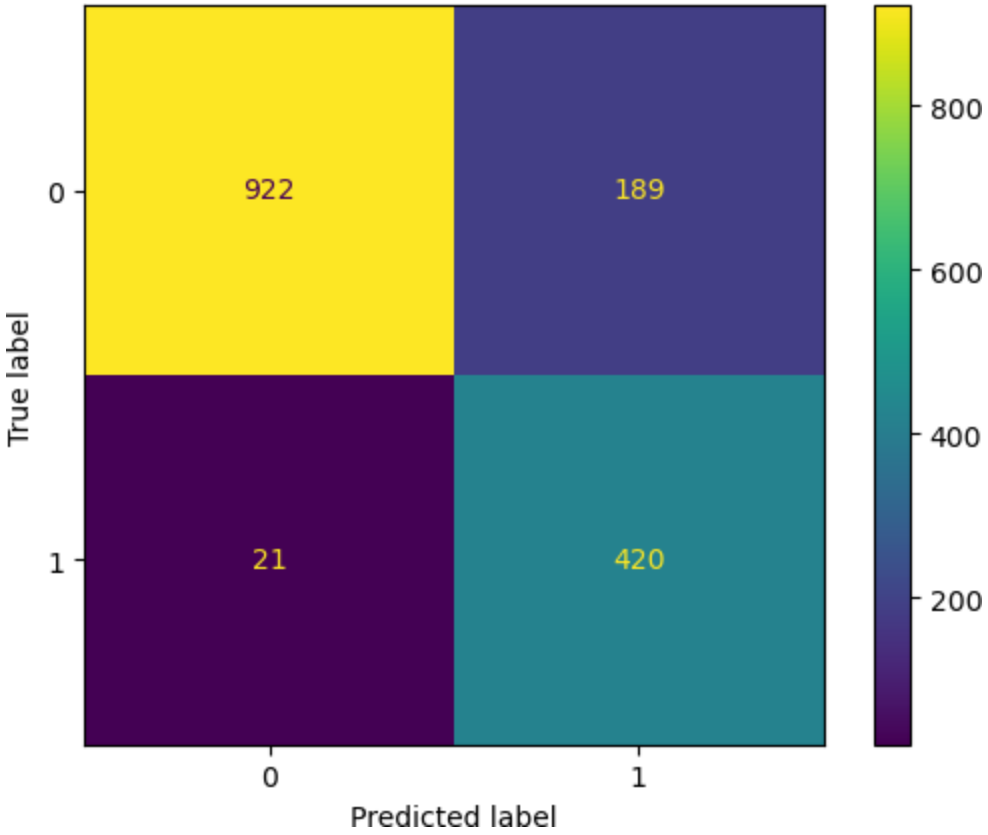
Out[21]:
```
KNeighborsC assifier
KNeighborsClassifier()
```

In [22]:
```
y_pred=knn.predict(x_test)
```

In [23]:
```
from sklearn.metrics import ConfusionMatrixDisplay, accuracy_score,classification_r
```

In [25]:
```
ConfusionMatrixDisplay.from_predictions(y_test,y_pred)
```

Out[25]:
```
<sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay at 0x1ac2128e4b0>
```



In [26]:
```
y_test.value_counts()
```

Out[26]:
```
Prediction
0    1111
1     441
Name: count, dtype: int64
```

In [27]:
```
accuracy_score(y_test,y_pred)
```

Out[27]:
```
0.8646907216494846
```

In [28]:
```
print(classification_report(y_test,y_pred))
```

```
              precision    recall  f1-score   support

           0       0.98      0.83      0.90      1111
           1       0.69      0.95      0.80       441

    accuracy                           0.86      1552
   macro avg       0.83      0.89      0.85      1552
weighted avg       0.90      0.86      0.87      1552
```

In [31]:
```
import numpy as np
import matplotlib.pyplot as plt
```

In [32]:
```
error=[]
for k in range(1,41):
    knn=KNeighborsClassifier(n_neighbors=k)
```

```python
    knn.fit(x_train,y_train)
    ypred=knn.predict(x_test)
    error.append(np.mean(y_pred!=y_test))
```

In [33]:

```python
error
```

Out[33]:

```
[0.13530927835051546,
 0.13530927835051546,
 0.13530927835051546,
 0.13530927835051546,
 0.13530927835051546,
 0.13530927835051546,
 0.13530927835051546,
 0.13530927835051546,
 0.13530927835051546,
 0.13530927835051546,
 0.13530927835051546,
 0.13530927835051546,
 0.13530927835051546,
 0.13530927835051546,
 0.13530927835051546,
 0.13530927835051546,
 0.13530927835051546,
 0.13530927835051546,
 0.13530927835051546,
 0.13530927835051546,
 0.13530927835051546,
 0.13530927835051546,
 0.13530927835051546,
 0.13530927835051546,
 0.13530927835051546,
 0.13530927835051546,
 0.13530927835051546,
 0.13530927835051546,
 0.13530927835051546,
 0.13530927835051546,
 0.13530927835051546,
 0.13530927835051546,
 0.13530927835051546,
 0.13530927835051546,
 0.13530927835051546,
 0.13530927835051546,
 0.13530927835051546,
 0.13530927835051546,
 0.13530927835051546,
 0.13530927835051546]
```

In [34]:

```python
knn=KNeighborsClassifier(n_neighbors=1)
knn.fit(x_train,y_train)
```

Out[34]:

```
▾ KNeighborsClassifier
KNeighborsClassifier(n_neighbors=1)
```

In [35]:
```
y_pred=knn.predict(x_test)
```

In [36]:
```
accuracy_score(y_test,y_pred)
```

Out[36]:
```
0.8917525773195877
```

In [37]:
```
from sklearn.svm import SVC
svm=SVC(kernel='linear')
svm.fit(x_train,y_train)
```

Out[37]:
```
SVC(kernel='linear')
```

In [38]:
```
y_pred=svm.predict(x_test)
```

In [39]:
```
accuracy_score(y_test,y_pred)
```

Out[39]:
```
0.9755154639175257
```

In [ ]: