In [2]:
```
#Title : Classify the email using the binary classification method.Email Spam
#b)Abnormal State-Spam.Use K-Nearest Neighbors and Support Vector Machine for
#link:The emails.csv dataset on the Kaggle https://www.kaggle.com/datasets/bal
```

In [3]:
```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline
import warnings
warnings.filterwarnings('ignore')
from sklearn.model_selection import train_test_split
from sklearn import metrics
```

In [4]:
```
df=pd.read_csv("emails.csv")
df
```

Out[4]:

	Email No.	the	to	ect	and	for	of	a	you	hou	...	connevey	jay	valued	lay	infrastru
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 [5]:
```
df.isnull()
```

Out[5]:

	Email No.	the	to	ect	and	for	of	a	you	hou	...	connevey	jay	va
0	False	False	False	False	False	False	False	False	False	False	...	False	False	
1	False	False	False	False	False	False	False	False	False	False	...	False	False	
2	False	False	False	False	False	False	False	False	False	False	...	False	False	
3	False	False	False	False	False	False	False	False	False	False	...	False	False	
4	False	False	False	False	False	False	False	False	False	False	...	False	False	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	
5167	False	False	False	False	False	False	False	False	False	False	...	False	False	
5168	False	False	False	False	False	False	False	False	False	False	...	False	False	
5169	False	False	False	False	False	False	False	False	False	False	...	False	False	
5170	False	False	False	False	False	False	False	False	False	False	...	False	False	
5171	False	False	False	False	False	False	False	False	False	False	...	False	False	

5172 rows × 3002 columns

In [6]:
```
df.isnull().sum
```

Out[6]:
```
<bound method NDFrame._add_numeric_operations.<locals>.sum of       Email No.
the     to    ect    and    for     of      a    you  \
0         False  False  False  False  False  False  False  False  False
1         False  False  False  False  False  False  False  False  False
2         False  False  False  False  False  False  False  False  False
3         False  False  False  False  False  False  False  False  False
4         False  False  False  False  False  False  False  False  False
...         ...    ...    ...    ...    ...    ...    ...    ...    ...
5167      False  False  False  False  False  False  False  False  False
5168      False  False  False  False  False  False  False  False  False
5169      False  False  False  False  False  False  False  False  False
5170      False  False  False  False  False  False  False  False  False
5171      False  False  False  False  False  False  False  False  False

        hou  ...  connevey    jay  valued    lay  infrastructure  military  \
0     False  ...     False  False   False  False           False     False
1     False  ...     False  False   False  False           False     False
2     False  ...     False  False   False  False           False     False
3     False  ...     False  False   False  False           False     False
4     False  ...     False  False   False  False           False     False
...     ...  ...       ...    ...     ...    ...             ...       ...
5167  False  ...     False  False   False  False           False     False
5168  False  ...     False  False   False  False           False     False
5169  False  ...     False  False   False  False           False     False
5170  False  ...     False  False   False  False           False     False
5171  False  ...     False  False   False  False           False     False

      allowing     ff    dry  Prediction
0        False  False  False       False
1        False  False  False       False
2        False  False  False       False
3        False  False  False       False
4        False  False  False       False
...        ...    ...    ...         ...
5167     False  False  False       False
5168     False  False  False       False
5169     False  False  False       False
5170     False  False  False       False
5171     False  False  False       False

[5172 rows x 3002 columns]>
```

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

Out[7]:
```
(5172, 3002)
```

In [8]:
```
df.columns
```

Out[8]:
```
Index(['Email No.', 'the', 'to', 'ect', 'and', 'for', 'of', 'a', 'you', 'ho
u',
       ...
       'connevey', 'jay', 'valued', 'lay', 'infrastructure', 'military',
       'allowing', 'ff', 'dry', 'Prediction'],
      dtype='object', length=3002)
```

In [9]:
```
x=df.drop(['Email No.','Prediction'],axis=1)
y=df['Prediction']
x.shape
```

Out[9]:
```
(5172, 3000)
```

In [10]:
```
y.shape
```

Out[10]:
```
(5172,)
```

In [11]:
```
from sklearn.preprocessing import MinMaxScaler
scaler=MinMaxScaler()
x_scale=scaler.fit_transform(x)
x_scale.shape
```

Out[11]:
```
(5172, 3000)
```

In [12]:
```
from sklearn.model_selection import train_test_split
```

In [13]:
```
x_train,x_test,y_train,y_test=train_test_split(x_scale,y,test_size=0.25,random
x_train.shape
```

Out[13]:
```
(3879, 3000)
```

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

Out[14]:
```
(3879,)
```

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

Out[15]:
```
(1293, 3000)
```

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

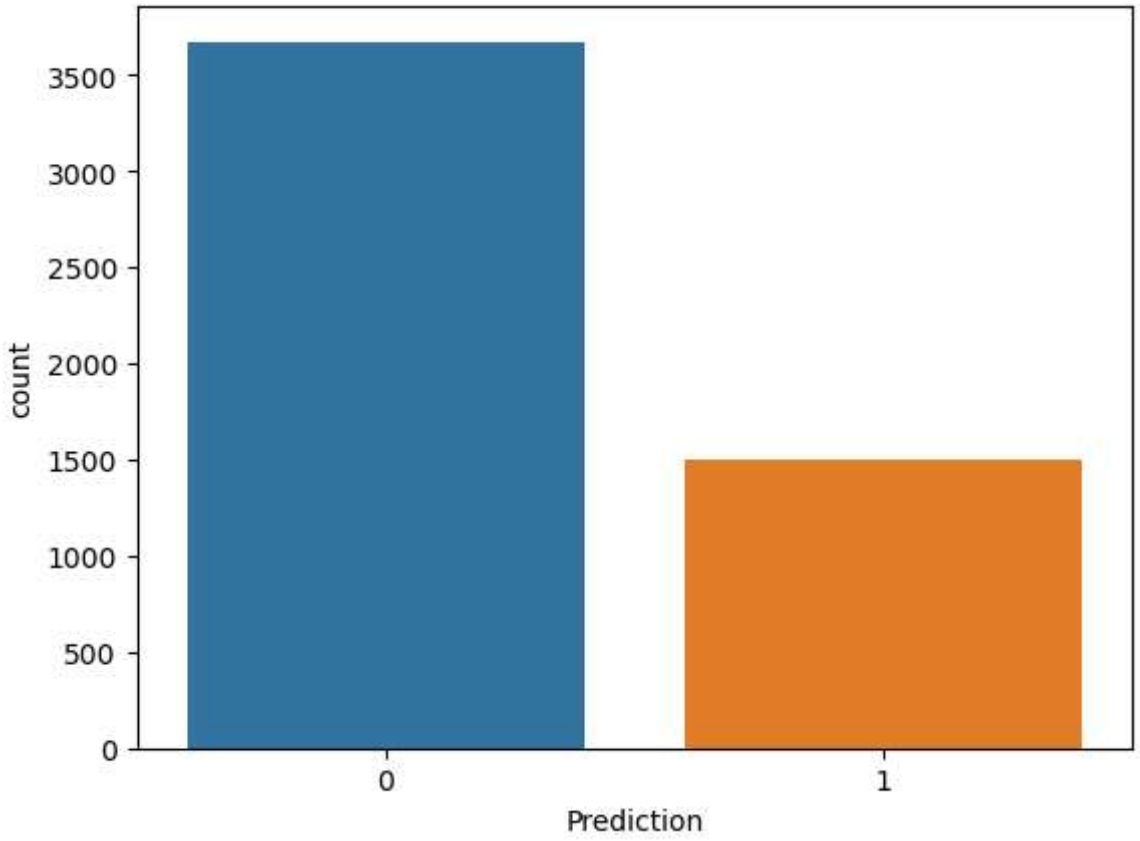
Out[16]:
```
(1293,)
```

In [17]:
```
 set(x.dtypes)
```

Out[17]:
```
{dtype('int64')}
```

In [18]:
```
sns.countplot(x=y)
```

Out[18]:
```
<AxesSubplot:xlabel='Prediction', ylabel='count'>
```



In [19]:
```
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import ConfusionMatrixDisplay,accuracy_score,classificat
```

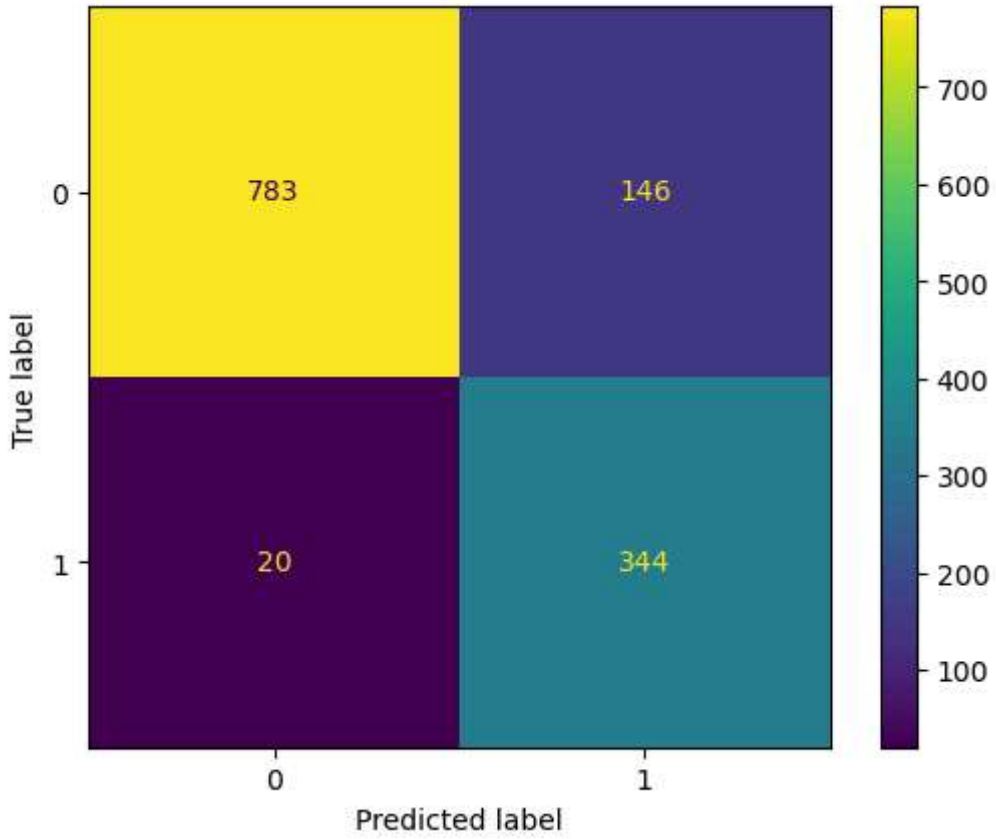
In [20]:
```
knn= KNeighborsClassifier(n_neighbors=5)
knn.fit(x_train,y_train)
```

Out[20]:
```
KNeighborsClassifier()
```

In [21]:
```
y_pred=knn.predict(x_test)
```

In [22]:
```
ConfusionMatrixDisplay.from_predictions(y_test,y_pred)
```

Out[22]:
```
<sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay at 0x2666ee0d7
c0>
```



In [23]:
```
y_test.value_counts()
```

Out[23]:
```
0    929
1    364
Name: Prediction, dtype: int64
```

In [24]:
```
accuracy_score(y_test,y_pred)
```

Out[24]:
```
0.871616395978345
```

In [25]:
```
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 [27]:
```
from sklearn.svm import SVC
```

In [28]:
```
svm=SVC(kernel='sigmoid')
```

In [29]:
```
svm.fit(x_train,y_train)
```

Out[29]:
```
SVC(kernel='sigmoid')
```

In [30]:
```
y_pred=svm.predict(x_test)
print("svm  accuracy=",accuracy_score(y_test,y_pred))
```

```
svm     accuracy= 0.839907192575406
```

In [31]:
```
 svm=SVC(kernel='linear')
 svm.fit(x_train,y_train)
```

Out[31]:
```
SVC(kernel='linear')
```

In [32]:
```
y_pred=svm.predict(x_test)
print("svm  accuracy=",accuracy_score(y_test,y_pred))
```

```
svm     accuracy= 0.9767981438515081
```

In [33]:
```
svm=SVC(kernel='rbf')
svm.fit(x_train,y_train)
```

Out[33]:
```
SVC()
```

In [34]:
```
y_pred=svm.predict(x_test)
print("svm  accuracy=",accuracy_score(y_test,y_pred))
```

```
svm     accuracy= 0.9450889404485692
```

In [35]:
```
svm=SVC(kernel='poly')
svm.fit(x_train,y_train)
```

Out[35]:
```
SVC(kernel='poly')
```

In [37]:
```
y_pred=svm.predict(x_test)
print("svm  accuracy=",accuracy_score(y_test,y_pred))
```

```
svm     accuracy= 0.7548337200309359
```