In [2]:
```
import pandas as pd
```

In [3]:
```
df=pd.read_csv("emails.csv")
```

In [4]:
```
df.sample(5)
```

Out [4]:

	Email No.	the	to	ect	and	for	of	a	you	hou	...	connevey	jay	valued	lay	infrastructure	military	allowing	ff	dry	Predi
4321	Email 4322	0	0	1	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
4558	Email 4559	18	14	4	3	5	0	93	1	2	...	0	0	0	0	0	0	0	0	0	0
2706	Email 2707	1	4	1	1	1	1	22	1	0	...	0	0	0	0	0	0	0	1	0	1
2665	Email 2666	4	0	1	0	5	0	20	0	0	...	0	0	0	0	0	0	0	0	0	0
4834	Email 4835	2	2	2	4	7	1	60	0	0	...	0	0	0	0	0	0	0	0	0	0

5 rows × 3002 columns

In [5]:
```
df.shape
```

Out [5]:
```
(5172, 3002)
```

In [7]:
```
# The last column has the labels for prediction : 1 for spam, 0 for not spam.
# The remaining 3000 columns are the 3000 most common words in all the emails, after excluding the n
```

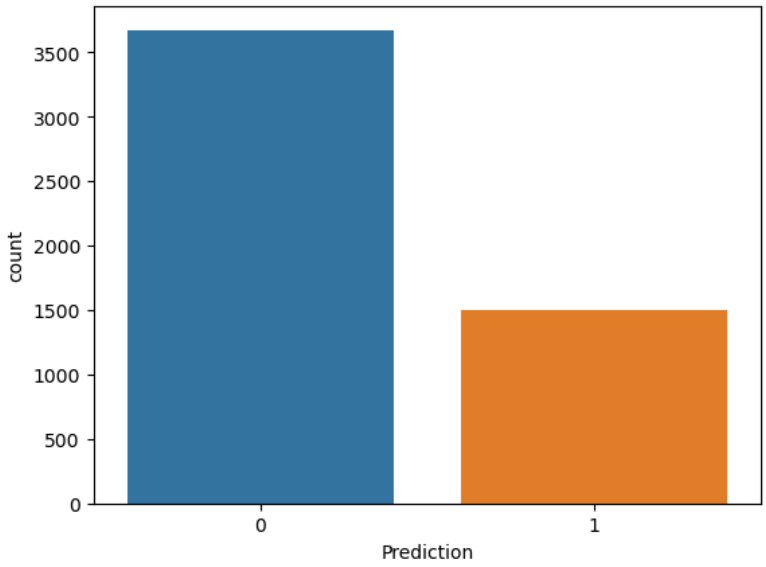
In [8]:
```
# split input and output data
X=df.drop(['Email No.','Prediction'],axis=1)
Y=df['Prediction']
```

In [9]:
```
X.shape
```

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

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

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



In [13]:
```
from sklearn.model_selection import train_test_split
X_train,X_test,Y_train,Y_test=train_test_split(X,Y,test_size=0.3,random_state=42)
```

In [17]:
```
# k- nearest neighbors

from sklearn.neighbors import KNeighborsClassifier
knn_m=KNeighborsClassifier(n_neighbors=5)
```

```
# the n_neighbors parameter specifies the number of nearest neighbors to consider when making a pred
knn_m.fit(X_train,Y_train)
```

Out [17]:
```
KNeighborsClassifier
KNeighborsClassifier()
```

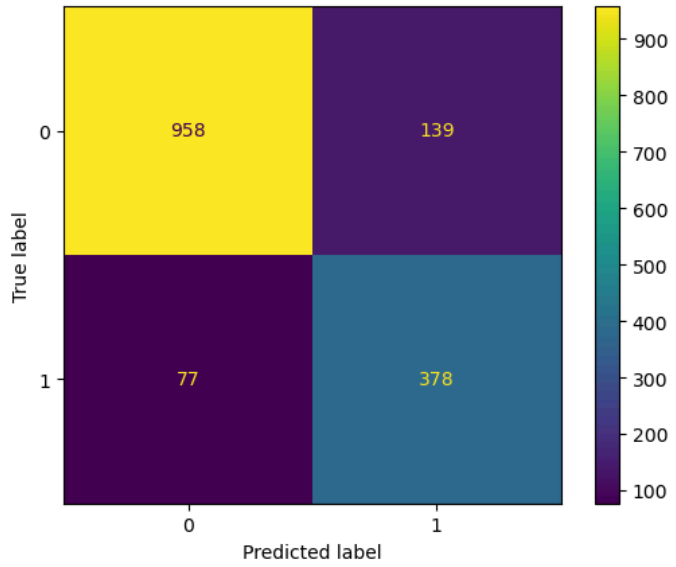
In [19]:
```
y_pred_knn=knn_m.predict(X_test)
y_pred_knn
```

Out [19]:
```
array([0, 0, 1, ..., 0, 0, 0], dtype=int64)
```

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

ConfusionMatrixDisplay.from_predictions(Y_test,y_pred_knn)
```

Out [23]:
```
<sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay at 0x1a8fe5c1110>
```



In [27]:
```
knn_accuracy = accuracy_score(Y_test, y_pred_knn)
knn_report = classification_report(Y_test, y_pred_knn)

print("Accuracy : ",knn_accuracy)
print("Report : ",knn_report)
```

```
Accuracy :  0.8608247422680413
Report :                precision    recall  f1-score   support

           0       0.93      0.87      0.90      1097
           1       0.73      0.83      0.78       455

    accuracy                           0.86      1552
   macro avg       0.83      0.85      0.84      1552
weighted avg       0.87      0.86      0.86      1552
```

In [29]:
```
## Support Vector Machine

from sklearn.svm import SVC
svm_model=SVC()
```

In [30]:
```
svm_model.fit(X_test,Y_test)
```

Out [30]:
```
SVC
SVC()
```

In [34]:
```
y_pred_svm=svm_model.predict(X_test)
y_pred_svm
```

Out [34]:
```
array([0, 0, 1, ..., 0, 0, 0], dtype=int64)
```

In [37]:
```
svm_accuracy = accuracy_score(Y_test, y_pred_svm)
svm_report = classification_report(Y_test, y_pred_svm)

print("Accuracy : ",svm_accuracy)
print("Report : ",svm_report)
```

```
Accuracy :  0.7744845360824743
Report :                precision    recall  f1-score   support
```

```
                   0       0.76      0.99      0.86      1097
                   1       0.89      0.26      0.41       455

            accuracy                           0.77      1552
           macro avg       0.83      0.63      0.63      1552
        weighted avg       0.80      0.77      0.73      1552
```

In [ ]: