```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

df = pd.read_csv('emails.csv')
df.head()
```

```
  Email No.  the  to  ect  and  for  of    a  you  hou  ...  connevey
jay  \
0   Email 1    0   0    1    0    0   0    2    0    0  ...         0
0
1   Email 2    8  13   24    6    6   2  102    1   27  ...         0
0
2   Email 3    0   0    1    0    0   0    8    0    0  ...         0
0
3   Email 4    0   5   22    0    5   1   51    2   10  ...         0
0
4   Email 5    7   6   17    1    5   2   57    0    9  ...         0
0

   valued  lay  infrastructure  military  allowing  ff  dry
Prediction
0       0    0               0         0         0   0    0
0
1       0    0               0         0         0   1    0
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   1    0
0

[5 rows x 3002 columns]
```

```
df.isnull().sum()
```

```
Email No.     0
the           0
to            0
ect           0
and           0
             ..
military      0
allowing      0
ff            0
dry           0
Prediction    0
Length: 3002, dtype: int64
```

```
df.dropna(how='any',inplace=True)
```

```
x = df.iloc[:,1:-1].values
y = df.iloc[:,-1].values

from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test =
train_test_split(x,y,test_size=0.25,random_state=10)

from sklearn.metrics import ConfusionMatrixDisplay, confusion_matrix,
accuracy_score, precision_score, recall_score
from sklearn.metrics import PrecisionRecallDisplay, RocCurveDisplay

def report(classifier, x_test, y_test):
    y_pred = classifier.predict(x_test)

    # Confusion Matrix
    cm = confusion_matrix(y_test, y_pred)
    display = ConfusionMatrixDisplay(cm,
display_labels=classifier.classes_)
    display.plot()

    # Accuracy, Precision, Recall
    print(f"Accuracy:  {accuracy_score(y_test, y_pred)}")
    print(f"Precision Score:  {precision_score(y_test, y_pred,
average='weighted')}")
    print(f"Recall Score:  {recall_score(y_test, y_pred,
average='weighted')}")

    # Precision-Recall Curve and ROC Curve
    PrecisionRecallDisplay.from_estimator(classifier, x_test, y_test)
    RocCurveDisplay.from_estimator(classifier, x_test, y_test)
```

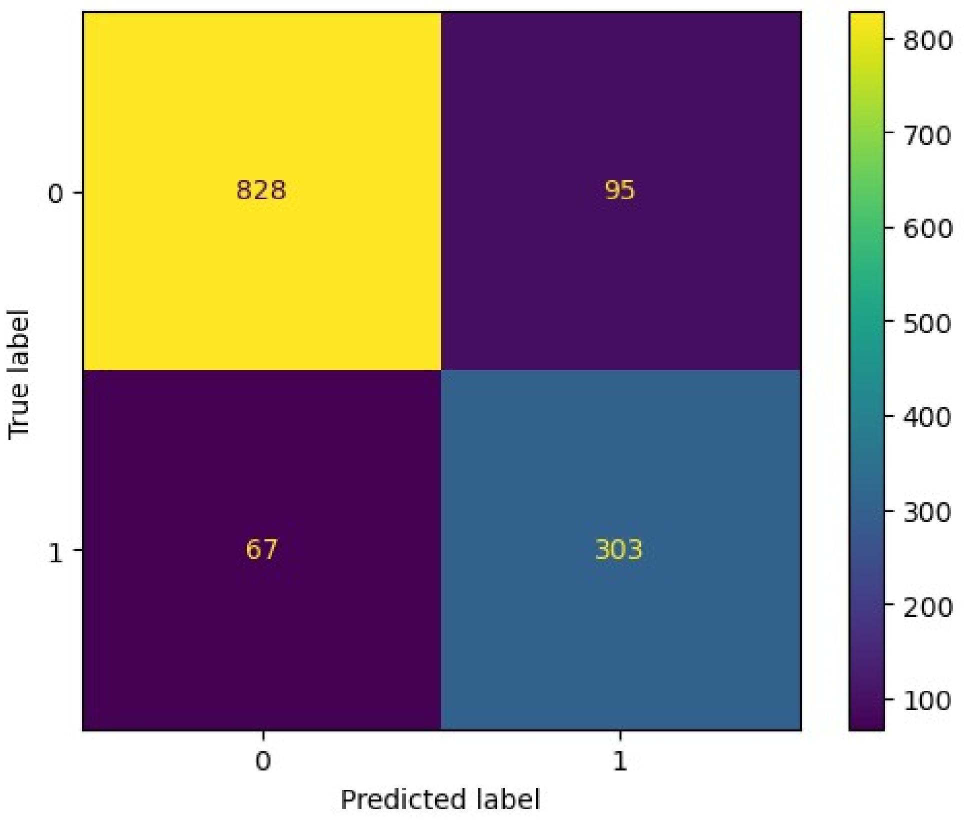
K-Nearest Neighbours Classifier

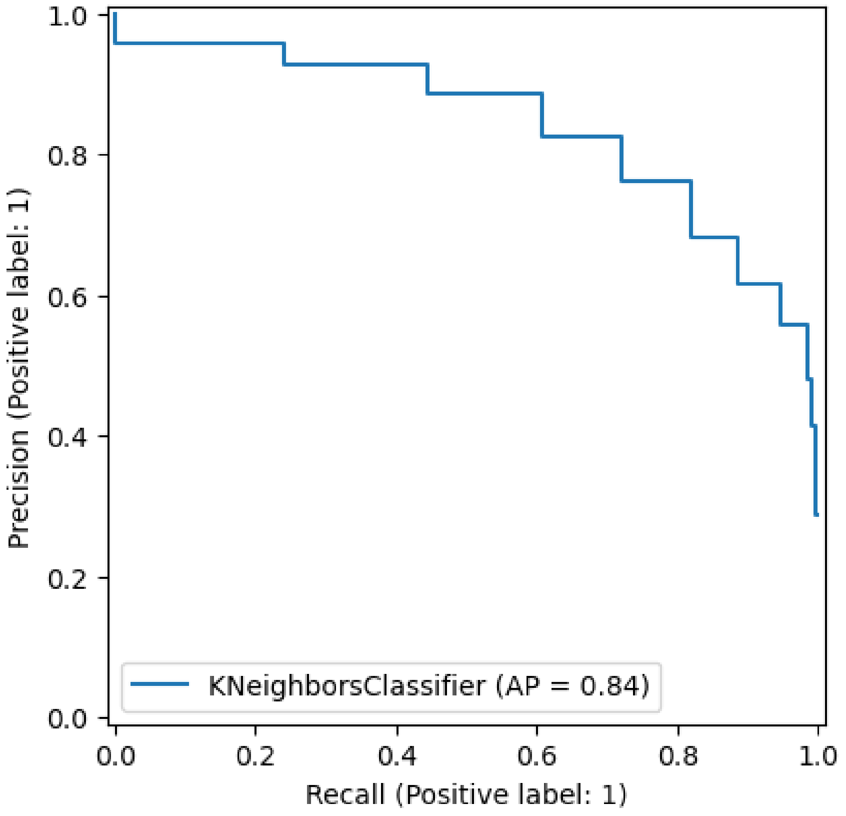
```
from sklearn.neighbors import KNeighborsClassifier

kNN = KNeighborsClassifier(n_neighbors=10)
kNN.fit(x_train,y_train)
```

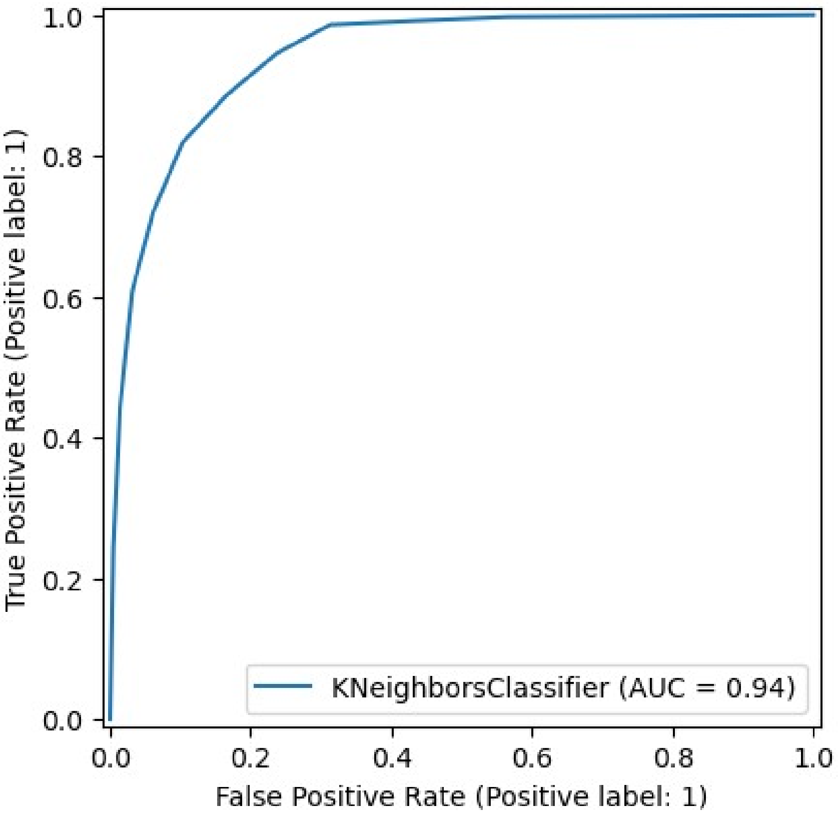
```
KNeighborsClassifier(n_neighbors=10)
```

```
report(kNN, x_test, y_test)
```

```
Accuracy:  0.8747099767981439
Precision Score:  0.8782577940448201
Recall Score:  0.8747099767981439
```


828
95
67
303
True label
0
1
Predicted label
0
1
800
700
600
500
400
300
200
100


Precision (Positive label: 1)
1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0
Recall (Positive label: 1)
KNeighborsClassifier (AP = 0.84)

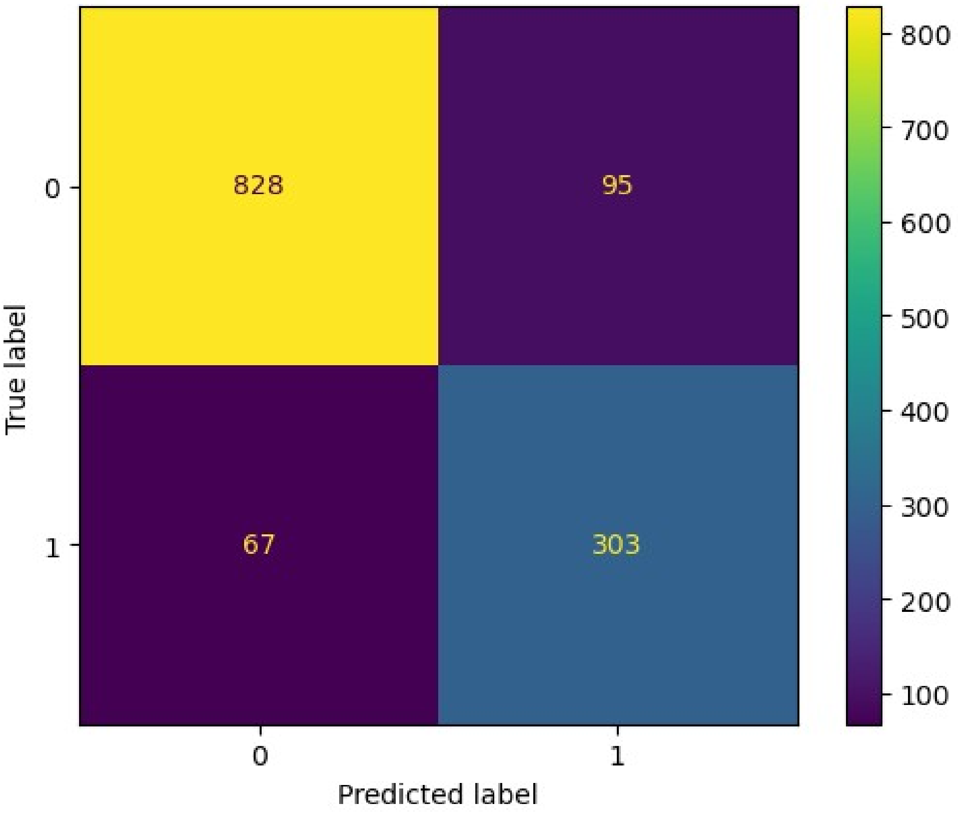


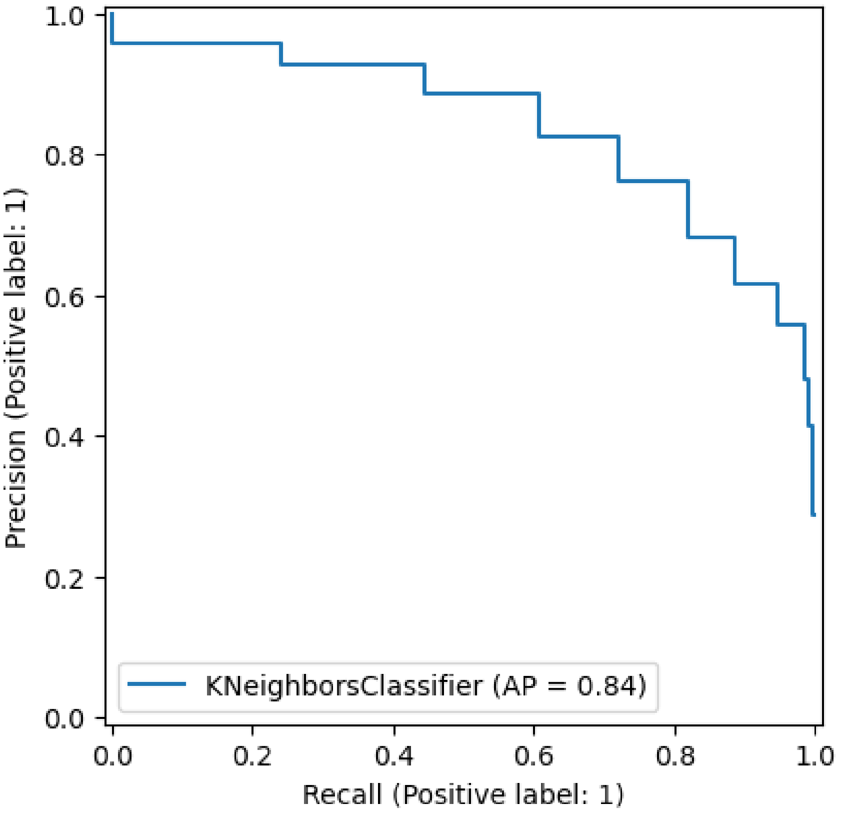
```
from sklearn.svm import SVC
svm = SVC(gamma='auto',random_state=10)
svm.fit(x_train,y_train)
```

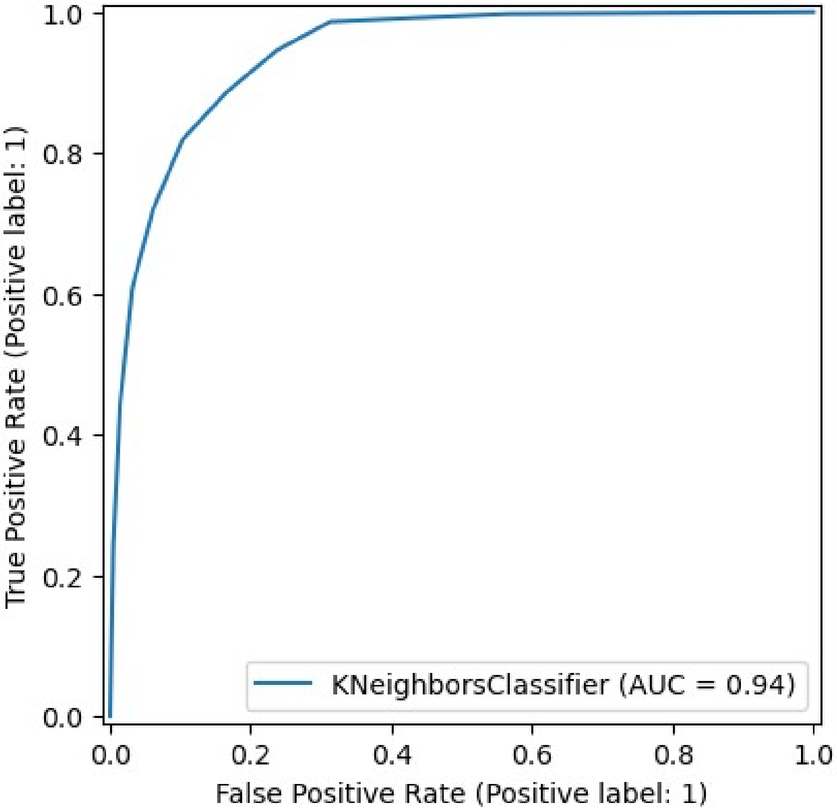
```
SVC(gamma='auto', random_state=10)
```

```
report(kNN, x_test, y_test)
```

```
Accuracy:  0.8747099767981439
Precision Score:  0.8782577940448201
Recall Score:  0.8747099767981439
```


828
95
67
303
True label
0
1
Predicted label
0
1
800
700
600
500
400
300
200
100


Precision (Positive label: 1)
1.0
0.8
0.6
0.4
0.2
0.0
Recall (Positive label: 1)
0.0
0.2
0.4
0.6
0.8
1.0
KNeighborsClassifier (AP = 0.84)


1.0
0.8
0.6
0.4
0.2
0.0
True Positive Rate (Positive label: 1)
KNeighborsClassifier (AUC = 0.94)
0.0
0.2
0.4
0.6
0.8
1.0
False Positive Rate (Positive label: 1)