```
import pandas as pd
import numpy as np
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.shape
```

```
(5172, 3002)
```

```
df.describe()
```

```
               the           to          ect          and          for
\
count  5172.000000  5172.000000  5172.000000  5172.000000  5172.000000

mean      6.640565     6.188128     5.143852     3.075599     3.124710

std      11.745009     9.534576    14.101142     6.045970     4.680522
```

```
min       0.000000    0.000000    1.000000    0.000000    0.000000

25%       0.000000    1.000000    1.000000    0.000000    1.000000

50%       3.000000    3.000000    1.000000    1.000000    2.000000

75%       8.000000    7.000000    4.000000    3.000000    4.000000

max     210.000000  132.000000  344.000000   89.000000   47.000000


                of           a         you         hou          in
...  \
count  5172.000000 5172.000000 5172.000000 5172.000000 5172.000000
...
mean      2.627030   55.517401    2.466551    2.024362   10.600155
...
std       6.229845   87.574172    4.314444    6.967878   19.281892
...
min       0.000000    0.000000    0.000000    0.000000    0.000000
...
25%       0.000000   12.000000    0.000000    0.000000    1.000000
...
50%       1.000000   28.000000    1.000000    0.000000    5.000000
...
75%       2.000000   62.250000    3.000000    1.000000   12.000000
...
max      77.000000 1898.000000   70.000000  167.000000  223.000000
...

          connevey         jay      valued         lay
infrastructure  \
count  5172.000000 5172.000000 5172.000000 5172.000000
5172.000000
mean      0.005027    0.012568    0.010634    0.098028
0.004254
std       0.105788    0.199682    0.116693    0.569532
0.096252
min       0.000000    0.000000    0.000000    0.000000
0.000000
25%       0.000000    0.000000    0.000000    0.000000
0.000000
50%       0.000000    0.000000    0.000000    0.000000
0.000000
75%       0.000000    0.000000    0.000000    0.000000
0.000000
max       4.000000    7.000000    2.000000   12.000000
3.000000

          military    allowing          ff         dry  Prediction
```

```
count  5172.000000  5172.000000  5172.000000  5172.000000  5172.000000
mean      0.006574     0.004060     0.914733     0.006961     0.290023
std       0.138908     0.072145     2.780203     0.098086     0.453817
min       0.000000     0.000000     0.000000     0.000000     0.000000
25%       0.000000     0.000000     0.000000     0.000000     0.000000
50%       0.000000     0.000000     0.000000     0.000000     0.000000
75%       0.000000     0.000000     1.000000     0.000000     1.000000
max       4.000000     3.000000   114.000000     4.000000     1.000000

[8 rows x 3001 columns]
```

```
df = df.drop("Email No.", axis=1)
```

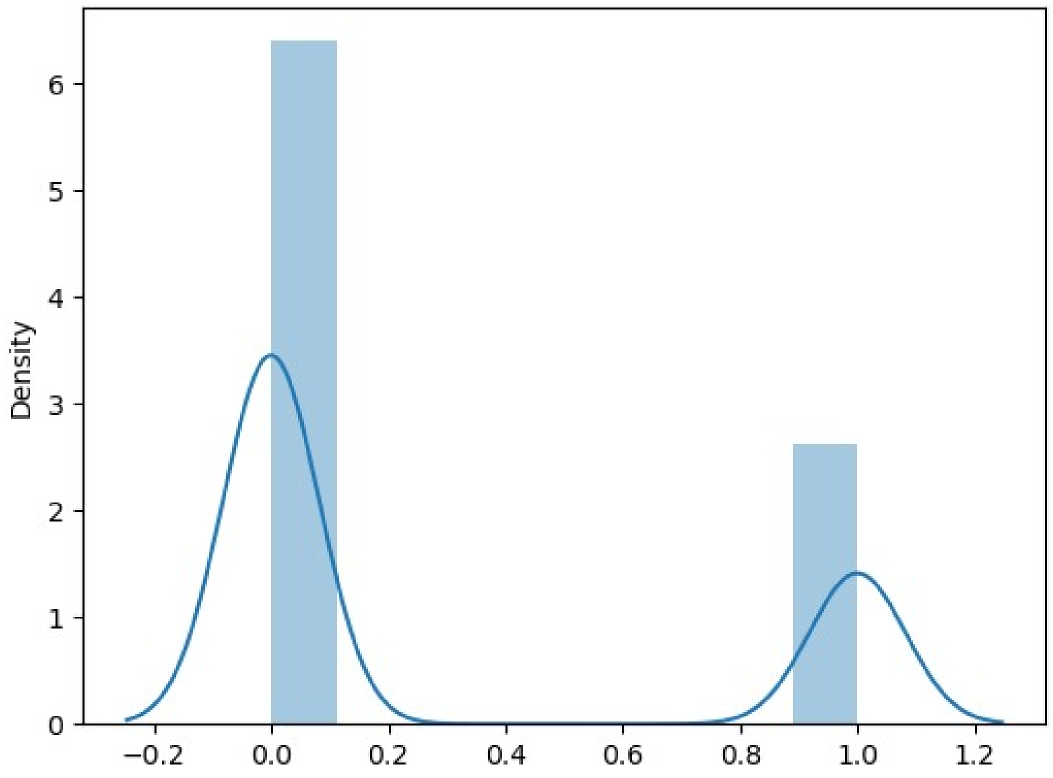
```
df.isna().sum()
```

```
the             0
to              0
ect             0
and             0
for             0
               ..
military        0
allowing        0
ff              0
dry             0
Prediction      0
Length: 3001, dtype: int64
```

```
sns.distplot(x=df["Prediction"])
plt.show()
```

```
C:\Users\prajw\AppData\Local\Temp\ipykernel_26344\422357721.py:1: 
UserWarning: 

`distplot` is a deprecated function and will be removed in seaborn 
v0.14.0.

Please adapt your code to use either `displot` (a figure-level 
function with
similar flexibility) or `histplot` (an axes-level function for 
histograms).

For a guide to updating your code to use the new functions, please see
```

```
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

  sns.distplot(x=df["Prediction"])
```



```python
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,
    precision_recall_curve,
    roc_curve,
)

def report(classifier, x_test, y_test):
    # Predict the labels
    y_pred = classifier.predict(x_test)
    # Confusion Matrix
    cm = confusion_matrix(y_test, y_pred)
    display = ConfusionMatrixDisplay(confusion_matrix=cm,
display_labels=classifier.classes_)
    display.plot()

    # Metrics
    print(f"Accuracy:  {accuracy_score(y_test, y_pred):.2f}")
    print(f"Precision Score:  {precision_score(y_test, y_pred,
average='weighted'):.2f}")
    print(f"Recall Score:  {recall_score(y_test, y_pred,
average='weighted'):.2f}")

    # Precision-Recall Curve and ROC Curve (for binary or
probabilistic classifiers)
    try:
        precision_recall_curve(classifier, x_test, y_test)
        roc_curve(classifier, x_test, y_test)
    except AttributeError:
        print("plot_precision_recall_curve and plot_roc_curve require
a classifier with 'predict_proba' or 'decision_function'.")

# Usage:
# report(classifier, x_test, y_test)
```

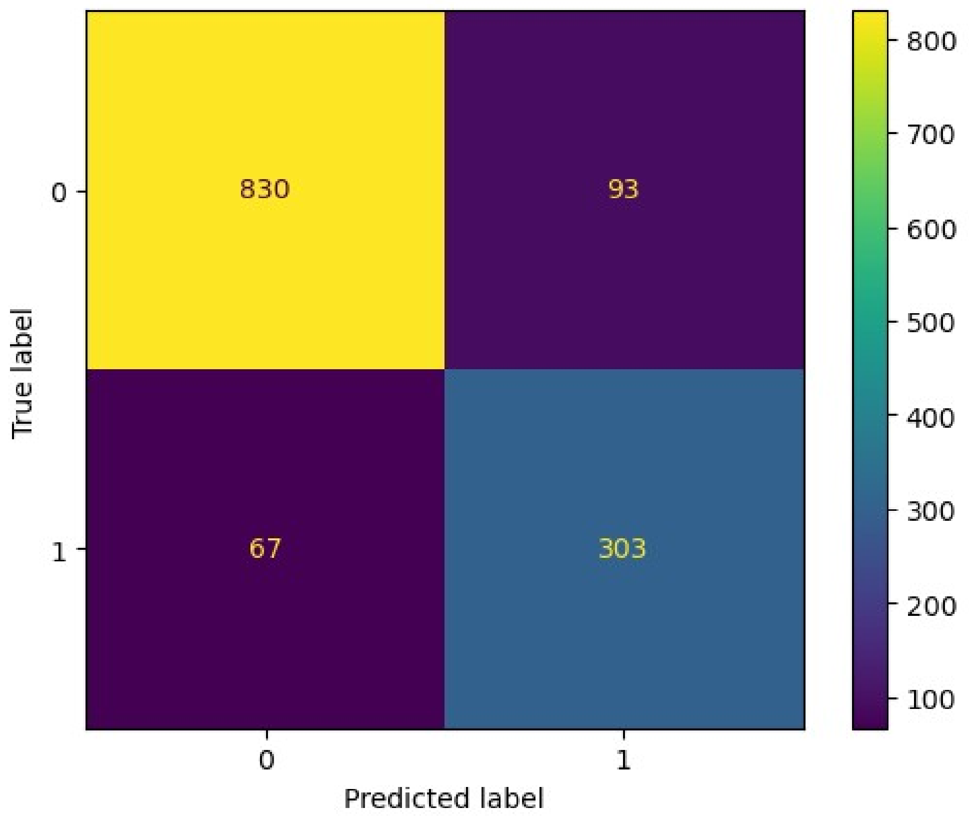
```
from sklearn.neighbors import KNeighborsClassifier

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

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

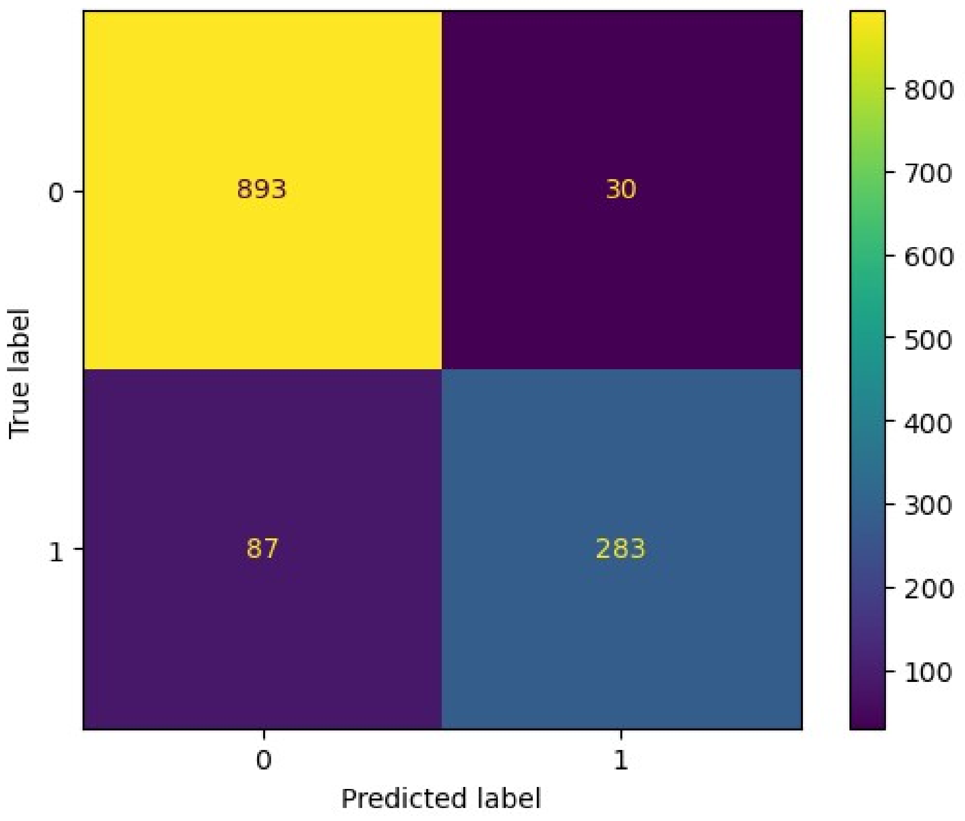
```
report(kNN, x_test, y_test)
```

```
Accuracy:  0.88
Precision Score:  0.88
Recall Score:  0.88

-----------------------------------------------------------------------
-----
TypeError                                 Traceback (most recent call
last)
Cell In[35], line 1
----> 1 report(kNN, x_test, y_test)

Cell In[31], line 27, in report(classifier, x_test, y_test)
     25 # Precision-Recall Curve and ROC Curve (for binary or
probabilistic classifiers)
```

```
     26 try:
---> 27     precision_recall_curve(classifier, x_test, y_test)
     28     roc_curve(classifier, x_test, y_test)
     29 except AttributeError:

File ~\anaconda3\Lib\site-packages\sklearn\utils\
_param_validation.py:189, in
validate_params.<locals>.decorator.<locals>.wrapper(*args, **kwargs)
    186 func_sig = signature(func)
    188 # Map *args/**kwargs to the function signature
--> 189 params = func_sig.bind(*args, **kwargs)
    190 params.apply_defaults()
    192 # ignore self/cls and positional/keyword markers

File ~\anaconda3\Lib\inspect.py:3212, in Signature.bind(self, *args,
**kwargs)
   3207 def bind(self, /, *args, **kwargs):
   3208     """Get a BoundArguments object, that maps the passed
`args`
   3209     and `kwargs` to the function's signature.  Raises
`TypeError`
   3210     if the passed arguments can not be bound.
   3211     """
-> 3212     return self._bind(args, kwargs)

File ~\anaconda3\Lib\inspect.py:3138, in Signature._bind(self, args,
kwargs, partial)
   3134 else:
   3135     if param.kind in (_VAR_KEYWORD, _KEYWORD_ONLY):
   3136         # Looks like we have no parameter for this positional
   3137         # argument
-> 3138         raise TypeError(
   3139             'too many positional arguments') from None
   3141     if param.kind == _VAR_POSITIONAL:
   3142         # We have an '*args'-like argument, let's fill it with
   3143         # all positional arguments we have left and move on to
   3144         # the next phase
   3145         values = [arg_val]

TypeError: too many positional arguments
```



```
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(svm,x_test,y_test)

Accuracy:  0.91
Precision Score:  0.91
Recall Score:  0.91

----------------------------------------------------------------------
-----
TypeError                                 Traceback (most recent call
last)
Cell In[38], line 1
----> 1 report(svm,x_test,y_test)

Cell In[31], line 27, in report(classifier, x_test, y_test)
     25 # Precision-Recall Curve and ROC Curve (for binary or
probabilistic classifiers)
     26 try:
---> 27     precision_recall_curve(classifier, x_test, y_test)
     28     roc_curve(classifier, x_test, y_test)
     29 except AttributeError:

File ~\anaconda3\Lib\site-packages\sklearn\utils\
_param_validation.py:189, in
validate_params.<locals>.decorator.<locals>.wrapper(*args, **kwargs)
    186 func_sig = signature(func)
    188 # Map *args/**kwargs to the function signature
--> 189 params = func_sig.bind(*args, **kwargs)
    190 params.apply_defaults()
    192 # ignore self/cls and positional/keyword markers

File ~\anaconda3\Lib\inspect.py:3212, in Signature.bind(self, *args,
**kwargs)
   3207 def bind(self, /, *args, **kwargs):
   3208     """Get a BoundArguments object, that maps the passed
`args`
   3209     and `kwargs` to the function's signature.  Raises
`TypeError`
   3210     if the passed arguments can not be bound.
   3211     """
-> 3212     return self._bind(args, kwargs)

File ~\anaconda3\Lib\inspect.py:3138, in Signature._bind(self, args,
kwargs, partial)
   3134 else:
   3135     if param.kind in (_VAR_KEYWORD, _KEYWORD_ONLY):
   3136         # Looks like we have no parameter for this positional
   3137         # argument
-> 3138         raise TypeError(
   3139             'too many positional arguments') from None
   3141     if param.kind == _VAR_POSITIONAL:
   3142         # We have an '*args'-like argument, let's fill it with
   3143         # all positional arguments we have left and move on to
   3144         # the next phase
   3145         values = [arg_val]

TypeError: too many positional arguments
```


893
30
87
283
True label
Predicted label
0
1
0
1
800
700
600
500
400
300
200
100