

Sahil Changede - BE-A2-14125

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]:

	Email No.	the	to	ect	and	for	of	a	you	hou	...	connevey	jay	valued	lay	infrastructui
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	

5 rows × 3002 columns

In [5]: `x = df.drop(['Email No.', 'Prediction'], axis = 1)`  
`y = df['Prediction']`In [6]: `x.shape`

Out[6]: (5172, 3000)

In [7]: `x.info()`

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5172 entries, 0 to 5171
Columns: 3000 entries, the to dry
dtypes: int64(3000)
memory usage: 118.4 MB
```

In [8]: `x.dtypes`

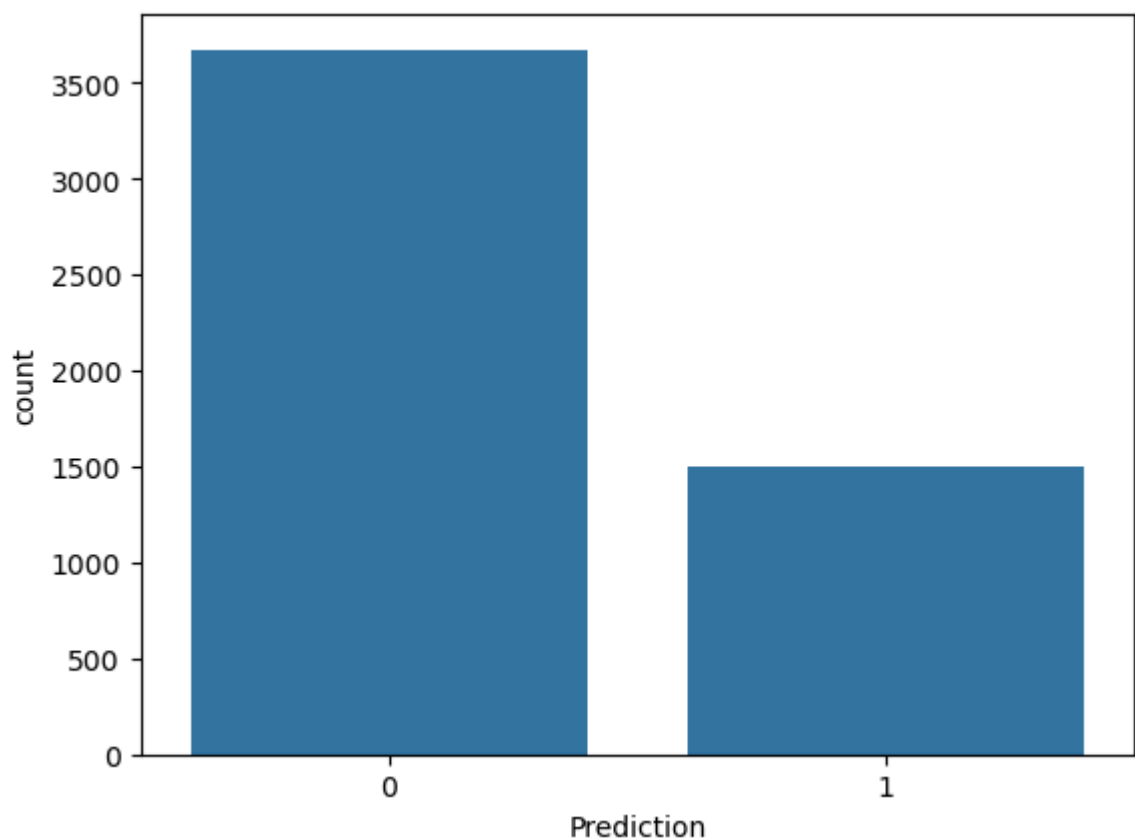
```
Out[8]: the          int64
        to          int64
        ect        int64
        and        int64
        for        int64
        ...
        infrastructure int64
        military      int64
        allowing      int64
        ff            int64
        dry           int64
        Length: 3000, dtype: object
```

```
In [9]: set(x.dtypes)
```

```
Out[9]: {dtype('int64')}
```

```
In [10]: import seaborn as sns
         sns.countplot(x=y)
```

```
Out[10]: <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 [27]: 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
```

```
In [28]: x_scaled.shape
```

```
Out[28]: (5172, 3000)
```

```
In [29]: x_train.shape
```

```
Out[29]: (3620, 3000)
```

```
In [33]: x_test.shape
```

```
Out[33]: (1552, 3000)
```

```
In [34]: from sklearn.neighbors import KNeighborsClassifier
knn = KNeighborsClassifier(n_neighbors = 5)
knn.fit(X_train, y_train)
```

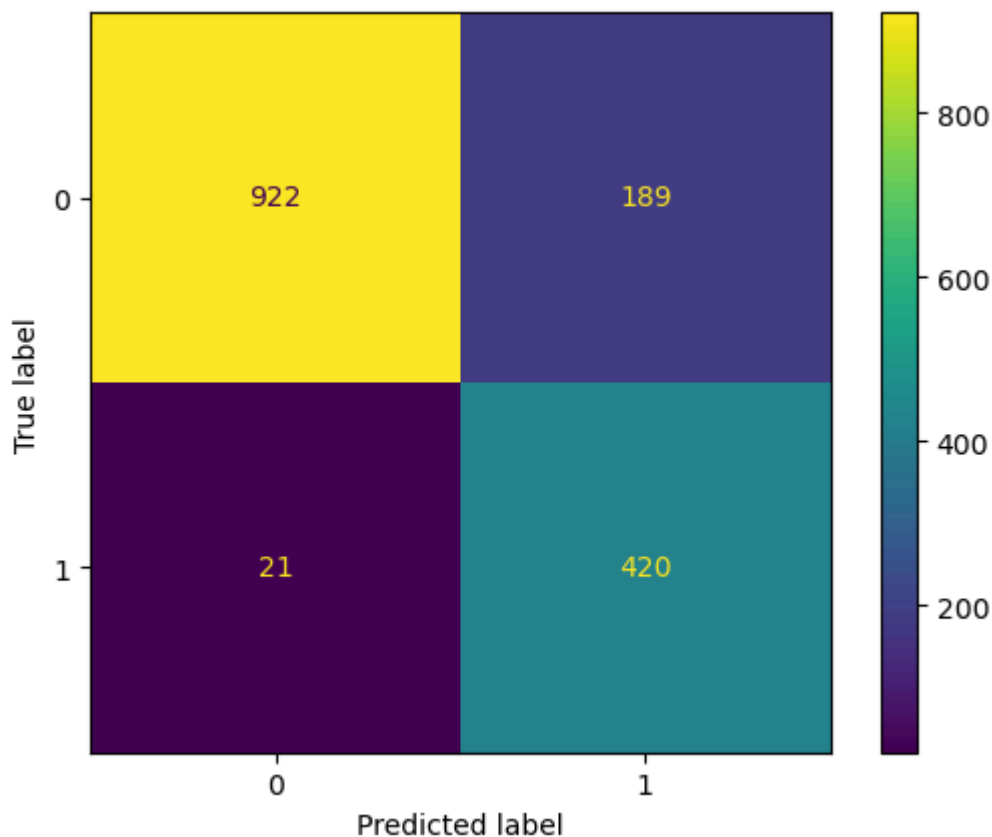
```
Out[34]: ▼ KNeighborsClassifier ⓘ ⓘ
KNeighborsClassifier()
```

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

```
In [36]: from sklearn.metrics import ConfusionMatrixDisplay,accuracy_score,classification_re
```

```
In [37]: ConfusionMatrixDisplay.from_predictions(y_test, y_pred)
```

```
Out[37]: <sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay at 0x1c147ddab70>
```



```
In [38]: y_test.value_counts()
```

```
Out[38]: Prediction
0      1111
1       441
Name: count, dtype: int64
```

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

```
Out[39]: 0.8646907216494846
```

```
In [41]: 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

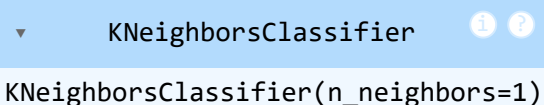
```
In [42]: import numpy as np
import matplotlib.pyplot as plt
```

```
In [43]: error = []
for k in range(1,41):
    knn = KNeighborsClassifier(n_neighbors =k)
    knn.fit(X_train, y_train)
    y_pred = knn.predict(X_test)
    error.append(np.mean(y_pred !=y_test))
```

```
In [44]: error
```

```
Out[44]: [0.10824742268041238,
0.10502577319587629,
0.11855670103092783,
0.11082474226804123,
0.13530927835051546,
0.12886597938144329,
0.15914948453608246,
0.15528350515463918,
0.17719072164948454,
0.17010309278350516,
0.19974226804123713,
0.19652061855670103,
0.21520618556701032,
0.21198453608247422,
0.22809278350515463,
0.22551546391752578,
0.23904639175257733,
0.23646907216494845,
0.2538659793814433,
0.25193298969072164,
0.2654639175257732,
0.26417525773195877,
0.27448453608247425,
0.27512886597938147,
0.28865979381443296,
0.2867268041237113,
0.3015463917525773,
0.3002577319587629,
0.3086340206185567,
0.30605670103092786,
0.3131443298969072,
0.3125,
0.31894329896907214,
0.3176546391752577,
0.32989690721649484,
0.3279639175257732,
0.33634020618556704,
0.33505154639175255,
0.34085051546391754,
0.3389175257731959]
```

```
In [45]: knn = KNeighborsClassifier(n_neighbors =1)
knn.fit(X_train, y_train)
```

```
Out[45]: 
KNeighborsClassifier(n_neighbors=1)
```

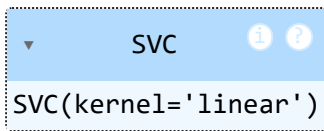
```
In [46]: y_pred = knn.predict(X_test)
```

```
In [47]: accuracy_score(y_test, y_pred)
```

```
Out[47]: 0.8917525773195877
```

```
In [50]: from sklearn.svm import SVC
svm = SVC(kernel = 'linear')
svm.fit(X_train, y_train)
```

Out[50]:

In [51]: `y_pred = svm.predict(X_test)`In [52]: `accuracy_score(y_test, y_pred)`

Out[52]: 0.9755154639175257