```
In [1]: import pandas as pd
```

```
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  infrastructure  military  allowing  ff  dry  Prediction
   0      Email 1    0   0    1    0    0   0    2    0    0  ...         0    0       0    0               0         0         0   0    0           0
   1      Email 2    8  13   24    6    6   2  102    1   27  ...         0    0       0    0               0         0         0   1    0           0
   2      Email 3    0   0    1    0    0   0    8    0    0  ...         0    0       0    0               0         0         0   0    0           0
   3      Email 4    0   5   22    0    5   1   51    2   10  ...         0    0       0    0               0         0         0   0    0           0
   4      Email 5    7   6   17    1    5   2   57    0    9  ...         0    0       0    0               0         0         0   1    0           0
 ...          ...  ...  ...  ...  ...  ...  ...  ...  ...  ...  ...       ...  ...     ...  ...             ...       ...       ...  ...  ...        ...
5167   Email 5168    2   2    2    3    0   0   32    0    0  ...         0    0       0    0               0         0         0   0    0           0
5168   Email 5169   35  27   11    2    6   5  151    4    3  ...         0    0       0    0               0         0         0   1    0           0
5169   Email 5170    0   0    1    1    0   0   11    0    0  ...         0    0       0    0               0         0         0   0    0           1
5170   Email 5171    2   7    1    0    2   1   28    2    0  ...         0    0       0    0               0         0         0   1    0           1
5171   Email 5172   22  24    5    1    6   5  148    8    2  ...         0    0       0    0               0         0         0   0    0           0

5172 rows × 3002 columns
```

```
In [5]: df.describe()
```

```
Out[5]:
              the           to          ect          and          for           of            a          you          hou           in  ...  c
count  5172.000000  5172.000000  5172.000000  5172.000000  5172.000000  5172.000000  5172.000000  5172.000000  5172.000000  5172.000000  ...  517
mean      6.640565     6.188128     5.143852     3.075599     3.124710     2.627030    55.517401     2.466551     2.024362    10.600155  ...  (
std      11.745009     9.534576    14.101142     6.045970     4.680522     6.229845    87.574172     4.314444     6.967878    19.281892  ...  (
min       0.000000     0.000000     1.000000     0.000000     0.000000     0.000000     0.000000     0.000000     0.000000     0.000000  ...  (
25%       0.000000     1.000000     1.000000     0.000000     1.000000     0.000000    12.000000     0.000000     0.000000     1.000000  ...  (
50%       3.000000     3.000000     1.000000     1.000000     2.000000     1.000000    28.000000     1.000000     0.000000     5.000000  ...  (
75%       8.000000     7.000000     4.000000     3.000000     4.000000     2.000000    62.250000     3.000000     1.000000    12.000000  ...  (
max     210.000000   132.000000   344.000000    89.000000    47.000000    77.000000  1898.000000    70.000000   167.000000   223.000000  ...  4

8 rows × 3001 columns
```

```
In [6]: X = df.drop(columns=["Email No.","Prediction"])
        X
```

```
Out[6]:
      the  to  ect  and  for  of    a  you  hou  in  ...  enhancements  connevey  jay  valued  lay  infrastructure  military  allowing  ff  dry
   0    0   0    1    0    0   0    2    0    0   0  ...             0         0    0       0    0               0         0         0   0    0
   1    8  13   24    6    6   2  102    1   27  18  ...             0         0    0       0    0               0         0         0   1    0
   2    0   0    1    0    0   0    8    0    0   4  ...             0         0    0       0    0               0         0         0   0    0
   3    0   5   22    0    5   1   51    2   10   1  ...             0         0    0       0    0               0         0         0   0    0
   4    7   6   17    1    5   2   57    0    9   3  ...             0         0    0       0    0               0         0         0   1    0
 ...  ...  ...  ...  ...  ...  ...  ...  ...  ... ...  ...           ...       ...  ...     ...  ...             ...       ...       ...  ...  ...
5167    2   2    2    3    0   0   32    0    0   5  ...             0         0    0       0    0               0         0         0   0    0
5168   35  27   11    2    6   5  151    4    3  23  ...             0         0    0       0    0               0         0         0   1    0
5169    0   0    1    1    0   0   11    0    0   1  ...             0         0    0       0    0               0         0         0   0    0
5170    2   7    1    0    2   1   28    2    0   8  ...             0         0    0       0    0               0         0         0   1    0
5171   22  24    5    1    6   5  148    8    2  23  ...             0         0    0       0    0               0         0         0   0    0

5172 rows × 3000 columns
```

In [7]:

```
y = df['Prediction']
y
```

Out[7]:

```
0       0
1       0
2       0
3       0
4       0
       ..
5167    0
5168    0
5169    1
5170    1
5171    0
Name: Prediction, Length: 5172, dtype: int64
```

In [9]:

```
from sklearn.preprocessing import StandardScaler
scalar = StandardScaler()
Scaled_x = scalar.fit_transform(X)
```

In [10]:

```
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test = train_test_split(Scaled_x,y)
```

In [11]:

```
from sklearn.neighbors import KNeighborsClassifier
knn = KNeighborsClassifier(n_neighbors=2)
knn.fit(x_train,y_train)
```

Out[11]:

```
KNeighborsClassifier(n_neighbors=2)
```

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [12]:

```
y_pred = knn.predict(x_test)
```

In [13]:

```
from sklearn.metrics import accuracy_score,confusion_matrix
```

In [14]:

```
acc = accuracy_score(y_test,y_pred)
acc
```

Out[14]:

```
0.8979118329466357
```

In [15]:

```
cm = confusion_matrix(y_test,y_pred)
```

In [16]:

```
cm
```

Out[16]:

```
array([[834,  70],
       [ 62, 327]])
```

In [18]:

```
from sklearn.svm import SVC, LinearSVC
import time
import math
```

In [19]:

```
start = time.time()
model = SVC(kernel='poly', C=2)
model.fit(x_train,y_train)
pred = model.predict(x_test)
acc = accuracy_score(y_test,pred)
print(round(acc*100,1), "%")
end = time.time()
print(f"{end-start:.5f} sec")
```

```
75.7 %
30.05556 sec
```

In [20]:

```
start = time.time()
model = SVC(kernel='rbf', C=2)
model.fit(x_train,y_train)
pred = model.predict(x_test)
acc = accuracy_score(y_test,pred)
print(round(acc*100,1), "%")
end = time.time()
print(f"{end-start:.5f} sec")
```

```
94.3 %
17.35720 sec
```

In [21]:

```
start = time.time()
model = SVC(kernel='sigmoid', C=2)
model.fit(x_train,y_train)
pred = model.predict(x_test)
acc = accuracy_score(y_test,pred)
print(round(acc*100,1), "%")
end = time.time()
print(f"{end-start:.5f} sec")
```

```
89.8 %
9.87208 sec
```

In [22]:

```
start = time.time()
model = LinearSVC(C=2)
model.fit(x_train,y_train)
pred = model.predict(x_test)
acc = accuracy_score(y_test,pred)
print(round(acc*100,1), "%")
end = time.time()
print(f"{end-start:.5f} sec")
```

```
/home/student/.local/lib/python3.10/site-packages/sklearn/svm/_classes.py:31: FutureWarning: The default value of `dua
l` will change from `True` to `'auto'` in 1.5. Set the value of `dual` explicitly to suppress the warning.
  warnings.warn(
```

```
92.6 %
6.39500 sec
```

```
/home/student/.local/lib/python3.10/site-packages/sklearn/svm/_base.py:1237: ConvergenceWarning: Liblinear failed to c
onverge, increase the number of iterations.
  warnings.warn(
```

In [ ]: