```
import pandas as pd
import numpy as np
from sklearn.metrics import accuracy_score
from sklearn.metrics import precision_score, recall_score, f1_score, roc_auc_score
from sklearn.metrics import confusion_matrix
import seaborn as sns
import matplotlib.pyplot as plt


df = pd.read_csv("emails.csv")
```

```
df
```

	Email No.	the	to	ect	and	for	of	a	you	hou	...	connevey	jay	valued	lay	infrastructure	military	allowing	ff	du
0	Email 1	0	0	1	0	0	0	2	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	
2	Email 3	0	0	1	0	0	0	8	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	
4	Email 5	7	6	17	1	5	2	57	0	9	...	0	0	0	0	0	0	0	1	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
5167	Email 5168	2	2	2	3	0	0	32	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	
5169	Email 5170	0	0	1	1	0	0	11	0	0	...	0	0	0	0	0	0	0	0	
5170	Email 5171	2	7	1	0	2	1	28	2	0	...	0	0	0	0	0	0	0	1	
5171	Email 5172	22	24	5	1	6	5	148	8	2	...	0	0	0	0	0	0	0	0	

5172 rows × 3002 columns

```
df.describe()
```

	the	to	ect	and	for	of	a	you	hou	in
count	5172.000000	5172.000000	5172.000000	5172.000000	5172.000000	5172.000000	5172.000000	5172.000000	5172.000000	5172.000000
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

8 rows × 3001 columns

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5172 entries, 0 to 5171
Columns: 3002 entries, Email No. to Prediction
dtypes: int64(3001), object(1)
memory usage: 118.5+ MB
```

```
df.isnull()
```

	Email No.	the	to	ect	and	for	of	a	you	hou	...	connevey
0	False	False	False	False	False	False	False	False	False	False	...	False
1	False	False	False	False	False	False	False	False	False	False	...	False
2	False	False	False	False	False	False	False	False	False	False	...	False
3	False	False	False	False	False	False	False	False	False	False		False

Checking Null Value

```
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
```

```
null_count = df.isnull().sum()
#Filter column with value more than one null value
column_null = null_count[null_count >= 1]
column_null
```

```
Series([], dtype: int64)
```

```
df.isnull().sum()
# all row having value as null
```

```
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
```

```
# remove email column as it is not for use(i.e unnecessary)

x = df.iloc[:,1:3001]
```

```
x.head()
```

	the	to	ect	and	for	of	a	you	hou	in	...	enhancements	connevey	jay	valued	lay	infrastructure	military	allowing	ff
0	0	0	1	0	0	0	2	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
2	0	0	1	0	0	0	8	0	0	4	...	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
4	7	6	17	1	5	2	57	0	9	3	...	0	0	0	0	0	0	0	0	1

5 rows × 3000 columns

```
# output class

y = df.iloc[:,-1]
y.head()
```

```
0    0
1    0
2    0
3    0
4    0
Name: Prediction, dtype: int64
```

Training and Testing Model

```
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(x,y, test_size=.2, random_state = 12)
```

```
x_test
```

	the	to	ect	and	for	of	a	you	hou	in	...	enhancements	connevey	jay	valued	lay	infrastructure	military	allowing
4075	2	1	3	0	1	2	13	1	0	3	...	0	0	0	0	0	0	0	0
4835	10	7	3	8	3	5	99	0	0	14	...	0	0	0	0	1	0	0	0
4439	6	5	4	2	0	9	57	3	1	0	...	0	0	0	0	0	0	0	0
3910	4	5	5	1	3	2	44	0	2	5	...	0	0	0	0	0	0	0	0
2398	0	1	1	0	4	0	14	0	0	6	...	0	0	0	0	0	0	0	0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
4367	18	11	16	7	5	3	126	1	14	29	...	0	0	0	0	0	0	0	0
2513	0	4	1	2	0	0	27	0	1	6	...	0	0	0	0	0	0	0	0
1662	2	4	2	3	1	1	104	0	0	18	...	0	0	0	0	1	0	0	0
3810	7	2	1	2	1	1	26	0	1	3	...	0	0	0	0	0	0	0	0
570	9	31	7	14	4	3	519	1	2	69	...	0	0	0	0	1	0	0	0

1035 rows × 3000 columns

```
# Feature Scaling

from sklearn.preprocessing import StandardScaler

sc = StandardScaler()
```

```
x_train = sc.fit_transform(x_train)
x_test = sc.fit_transform(x_test)
```

K-Nearest Neighbour

```
# Fitting K-NN classifier to the training set
```

```
from sklearn.neighbors import KNeighborsClassifier
knn = KNeighborsClassifier(n_neighbors=5, metric='minkowski', p=2)
knn.fit(x_train, y_train)
```

```
KNeighborsClassifier
KNeighborsClassifier()
```

Prediction

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

```
array([1, 0, 0, ..., 1, 1, 1])
```

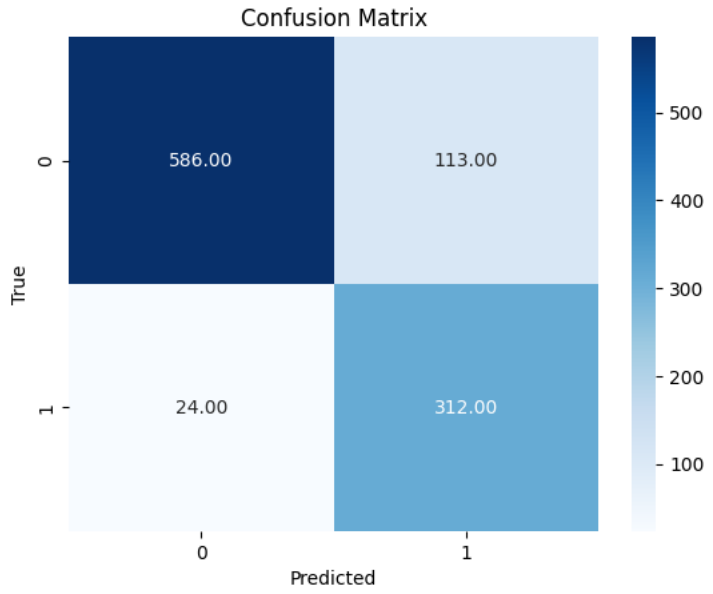
Confusion Matric

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

```
array([[586, 113],
       [ 24, 312]])
```

```
# Confusion matrix graph using heatmap

sns.heatmap(cm, annot=True, fmt=".2f", cmap= "Blues")
plt.xlabel("Predicted")
plt.ylabel("True")
plt.title("Confusion Matrix")
plt.show()
```




```
accuracy_score(y_test, y_pred)
```

```
0.8676328502415459
```

Support Vector Machine

```
from sklearn.svm import SVC
svc = SVC(kernel='linear', random_state=0)
svc.fit(x_train, y_train)
```

```
SVC(kernel='linear', random_state=0)
```

Prediction

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

```
array([0, 0, 0, ..., 1, 0, 1])
```

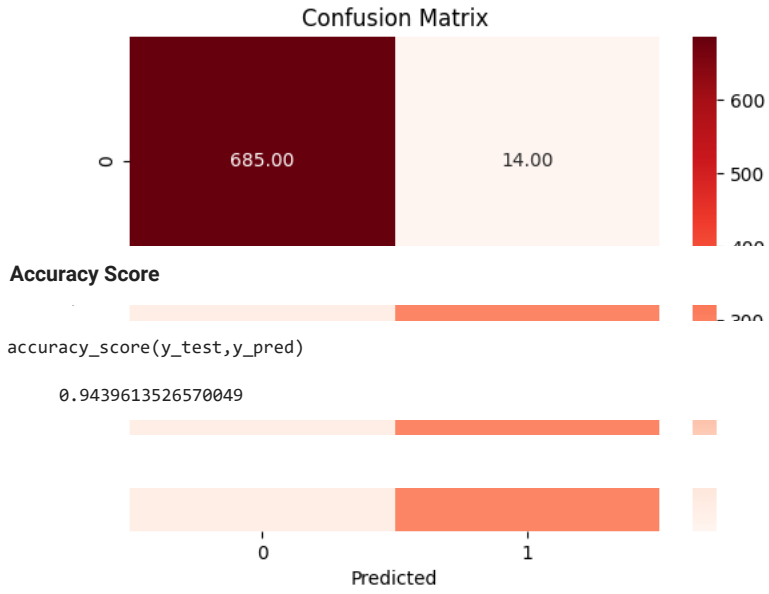
Confusion Matrix

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

```
array([[685,  14],
       [ 44, 292]])
```

```
#confusion matrix graph representation

sns.heatmap(cm, annot=True, fmt = '.2f', cmap='Reds')
plt.xlabel("Predicted")
plt.ylabel("True")
plt.title("Confusion Matrix")
plt.show()
```



Accuracy Score

```
accuracy_score(y_test,y_pred)
```

```
0.9439613526570049
```