In [ ]:
```
#Tushar Holkar
# Roll no: A28
```

In [ ]:
```
#Problem Statement
#Emails Spam detection  using Binary Classification
```

In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import plotly.express as px
import warnings
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsClassifier
from sklearn.svm import SVC
from sklearn.utils import resample
from sklearn import metrics
from tqdm.notebook import tqdm
%matplotlib inline
warnings.filterwarnings("ignore")
```

In [5]:
```
df = pd.read_csv("C:/Users/KJCOEMR/Downloads/emails.csv")
```

In [7]:
```
df.head()
```

Out[7]:

	Email No.	the	to	ect	and	for	of	a	you	hou	...	connevey	jay	valued	lay
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 [27]:
```
df.columns
```

Out[27]:
```
Index(['the', 'to', 'ect', 'and', 'for', 'of', 'a', 'you', 'hou', 'in',
       ...
       'connevey', 'jay', 'valued', 'lay', 'infrastructure', 'military',
       'allowing', 'ff', 'dry', 'Prediction'],
      dtype='object', length=3001)
```

In [29]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5172 entries, 0 to 5171
Columns: 3001 entries, the to Prediction
dtypes: int64(3001)
memory usage: 118.4 MB
```

In [33]:
```
df.isnull().sum()
```

Out[33]:
```
the           0
to            0
ect           0
and           0
for           0
             ..
military      0
allowing      0
ff            0
dry           0
Prediction    0
Length: 3001, dtype: int64
```

In [35]:
```
df.head(10)
```

Out[35]:

	the	to	ect	and	for	of	a	you	hou	in	...	connevey	jay	valued	lay	inf
0	0	0	1	0	0	0	2	0	0	0	...	0	0	0	0	
1	8	13	24	6	6	2	102	1	27	18	...	0	0	0	0	
2	0	0	1	0	0	0	8	0	0	4	...	0	0	0	0	
3	0	5	22	0	5	1	51	2	10	1	...	0	0	0	0	
4	7	6	17	1	5	2	57	0	9	3	...	0	0	0	0	
5	4	5	1	4	2	3	45	1	0	16	...	0	0	0	0	
6	5	3	1	3	2	1	37	0	0	9	...	0	0	0	0	
7	0	2	2	3	1	2	21	6	0	2	...	0	0	0	0	
8	2	2	3	0	0	1	18	0	0	3	...	0	0	0	0	
9	4	4	35	0	1	0	49	1	16	9	...	0	0	0	0	

10 rows × 3001 columns

In [39]:
```
df.tail(10)
```

Out[39]:

	the	to	ect	and	for	of	a	you	hou	in	...	connevey	jay	valued	lay
5162	2	3	1	2	1	2	32	0	0	7	...	0	0	0	0
5163	0	0	1	0	0	0	1	0	0	0	...	0	0	0	0
5164	21	18	3	1	6	4	106	1	2	18	...	0	0	0	0
5165	1	0	1	0	3	1	12	1	0	2	...	0	0	0	1
5166	1	0	1	1	0	0	4	0	0	0	...	0	0	0	0
5167	2	2	2	3	0	0	32	0	0	5	...	0	0	0	0
5168	35	27	11	2	6	5	151	4	3	23	...	0	0	0	0
5169	0	0	1	1	0	0	11	0	0	1	...	0	0	0	0
5170	2	7	1	0	2	1	28	2	0	8	...	0	0	0	0
5171	22	24	5	1	6	5	148	8	2	23	...	0	0	0	0

10 rows × 3001 columns

In [75]:
```
df.drop(columns=["Email No."], inplace=True, errors='ignore')
```

In [71]:
```
df.tail()
```

Out[71]:

	the	to	ect	and	for	of	a	you	hou	in	...	connevey	jay	valued	lay
5167	2	2	2	3	0	0	32	0	0	5	...	0	0	0	0
5168	35	27	11	2	6	5	151	4	3	23	...	0	0	0	0
5169	0	0	1	1	0	0	11	0	0	1	...	0	0	0	0
5170	2	7	1	0	2	1	28	2	0	8	...	0	0	0	0
5171	22	24	5	1	6	5	148	8	2	23	...	0	0	0	0

5 rows × 3001 columns

In [84]:
```
df.isnull().any().value_counts()
```

Out[84]:
```
False    3001
Name: count, dtype: int64
```

In [86]:
```
X=df.iloc[:, :df.shape[1]-1]
y=df.iloc[:, -1]
X.shape,y.shape
```

Out[86]:
```
((5172, 3000), (5172,))
```

In [88]:
```
from sklearn.model_selection import train_test_split
```

In [90]:
```
x_train,x_test,y_train,y_test=train_test_split(X,y,test_size=0.15)
```

In [94]:
```
from sklearn.linear_model import LogisticRegression
```

```
from sklearn.svm import SVC, LinearSVC
from sklearn.neural_network import MLPClassifier
```

In [98]:
```
models = {
    "Logistic Regression": LogisticRegression(solver='lbfgs', max_iter=2000),
    "Linear SVM": LinearSVC(max_iter=3000),
    "Polynomial SVM": SVC(kernel='poly', degree=2),
    "RBF SVM": SVC(kernel='rbf'),
    "Sigmoid SVM": SVC(kernel='sigmoid'),
    "Multi-layer Perceptron Classification": MLPClassifier(hidden_layer_sizes=[2
}
```

In [100…
```
from sklearn.metrics import accuracy_score
```

In [105…
```
for model_name, model in models.items():
    y_pred = model.fit(x_train, y_train).predict(x_test)
    print(f"Accuracy for {model_name} model is: {accuracy_score(y_test, y_pred)}
```

```
Accuracy for Logistic Regression model is: 0.9716494845360825
Accuracy for Linear SVM model is: 0.9484536082474226
Accuracy for Polynomial SVM model is: 0.7435567010309279
Accuracy for RBF SVM model is: 0.8118556701030928
Accuracy for Sigmoid SVM model is: 0.5927835051546392
Accuracy for Multi-layer Perceptron Classification model is: 0.979381443298969
```

In [ ]:
```

```