In [ ]:
```
Name: Vrushali Khade

Div: A
Roll No: 44
```

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

In [18]:
```
df=pd.read_csv('/home/student/Downloads/emails.csv')
```

In [19]:
```
df.head()
```

Out[19]:

	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

5 rows × 3002 columns

In [20]:
```
df.columns
```

Out[20]:
```
Index(['Email No.', 'the', 'to', 'ect', 'and', 'for', 'of', 'a', 'you', 'hou',
       'connevey', 'jay', 'valued', 'lay', 'infrastructure', 'military',
       'allowing', 'ff', 'dry', 'Prediction'],
      dtype='object', length=3002)
```

In [21]:
```
df.info
```

Out[21]:
```
<bound method DataFrame.info of        Email No.  the  to  ect  and  for  of    a  you  hou  ...  connevey  \
0         Email 1    0   0    1    0    0   0    2    0    0  ...         0
1         Email 2    8  13   24    6    6   2  102    1   27  ...         0
2         Email 3    0   0    1    0    0   0    8    0    0  ...         0
3         Email 4    0   5   22    0    5   1   51    2   10  ...         0
4         Email 5    7   6   17    1    5   2   57    0    9  ...         0
...           ...  ...  ..  ...  ...  ...  ..  ...  ...  ...  ...       ...
5167  Email 5168    2   2    2    3    0   0   32    0    0  ...         0
5168  Email 5169   35  27   11    2    6   5  151    4    3  ...         0
5169  Email 5170    0   0    1    1    0   0   11    0    0  ...         0
5170  Email 5171    2   7    1    0    2   1   28    2    0  ...         0
5171  Email 5172   22  24    5    1    6   5  148    8    2  ...         0

      jay  valued  lay  infrastructure  military  allowing  ff  dry  \
0       0       0    0               0         0         0   0    0
1       0       0    0               0         0         0   1    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         0   1    0
...   ...     ...  ...             ...       ...       ...  ..  ...
5167    0       0    0               0         0         0   0    0
5168    0       0    0               0         0         0   1    0
5169    0       0    0               0         0         0   0    0
5170    0       0    0               0         0         0   1    0
5171    0       0    0               0         0         0   0    0

      Prediction
0              0
1              0
2              0
3              0
4              0
...          ...
5167           0
5168           0
5169           1
5170           1
5171           0

[5172 rows x 3002 columns]>
```

In [22]:
```
df.isnull().sum()
```

Out[22]:
```
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
```

In [23]:
```
df.head(10)
```

Out[23]:

	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
5	Email 6	4	5	1	4	2	3	45	1	0	...	0	0	0	0	0	0	0	0	0	1
6	Email 7	5	3	1	3	2	1	37	0	0	...	0	0	0	0	0	0	0	0	0	0
7	Email 8	0	2	2	3	1	2	21	6	0	...	0	0	0	0	0	0	0	1	0	1
8	Email 9	2	2	3	0	0	1	18	0	0	...	0	0	0	0	0	0	0	0	0	0
9	Email 10	4	4	35	0	1	0	49	1	16	...	0	0	0	0	0	0	0	0	0	0

10 rows × 3002 columns

In [24]:
```
df.drop(columns=['Email No.'],inplace=True)
df.tail()
```

Out[24]:

	the	to	ect	and	for	of	a	you	hou	in	...	connevey	jay	valued	lay	infrastructure	military	allowing	ff	dry	Prediction
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	1	0	0
5169	0	0	1	1	0	0	11	0	0	1	...	0	0	0	0	0	0	0	0	0	1
5170	2	7	1	0	2	1	28	2	0	8	...	0	0	0	0	0	0	0	1	0	1
5171	22	24	5	1	6	5	148	8	2	23	...	0	0	0	0	0	0	0	0	0	0

5 rows × 3001 columns

In [25]:
```
df.isnull().any().value_counts()
```

Out[25]:
```
False    3001
dtype: int64
```

In [26]:
```
#Separate Features And Labels

x=df.iloc[:, :df.shape[1]-1]
y=df.iloc[:,-1]
x.shape,y.shape
```

Out[26]:
```
((5172, 3000), (5172,))
```

In [27]:
```
#Train
```

In [28]:
```
from sklearn.model_selection import train_test_split
```

In [29]:
```
x_train,x_test,y_train,y_test = train_test_split(x,y,test_size=0.15)
```

In [30]:
```
#Ml Models
```

In [31]:
```
from sklearn.linear_model import LogisticRegression
from sklearn.svm import SVC,LinearSVC
from sklearn.neural_network import MLPClassifier
```

In [32]:
```
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 Perception Classification":MLPClassifier(hidden_layer_sizes=[20,20])
        }
```

In [33]:
```
#Predict Accuracy
```

In [34]:
```
from sklearn.metrics import accuracy_score
```

In [35]:
```
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 : 0.9652061855670103
/home/student/anaconda3/lib/python3.10/site-packages/sklearn/svm/_base.py:1244: ConvergenceWarning: Liblinear f
ailed to converge, increase the number of iterations.
  warnings.warn(
Accuracy for Linear SVM model is : 0.9458762886597938
Accuracy for Polynomial SVM model is : 0.7667525773195877
Accuracy for RBF SVM model is : 0.8195876288659794
Accuracy for Sigmoid SVM model is : 0.6494845360824743
Accuracy for Multi-layer Perception Classification model is : 0.9742268041237113
```

In [ ]: