```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.svm import SVC, LinearSVC
from sklearn.neighbors import KNeighborsClassifier
from sklearn import metrics
from sklearn import preprocessing
```

Loading the Dataset

First we load the dataset and find out the number of columns, rows, NULL values, etc.

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

```
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.head()
```

	Email No.	the	to	ect	and	for	of	a	you	hou	...	connevey	jay	valued	lay	infrastructure	military	allowing	ff	dry	Predicti
0	Email 1	0	0	1	0	0	0	2	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	
2	Email 3	0	0	1	0	0	0	8	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	
4	Email 5	7	6	17	1	5	2	57	0	9	...	0	0	0	0	0	0	0	1	0	

5 rows × 3002 columns

```
df.dtypes
```

	0
Email No.	object
the	int64
to	int64
ect	int64
and	int64
...	...
military	int64
allowing	int64
ff	int64
dry	int64
Prediction	int64

3002 rows × 1 columns

dtype: object

Cleaning

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

```
df.dropna(inplace=True)
```

```
df.isna().sum()
```

	0
the	0
to	0
ect	0
and	0
for	0
...	...
military	0
allowing	0
ff	0
dry	0
Prediction	0

3001 rows × 1 columns

dtype: int64

```
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

Separating the features and the labels

```
X=df.iloc[:, :df.shape[1]-1]       #Independent Variables
y=df.iloc[:, -1]                   #Dependent Variable
X.shape, y.shape
```

```
((5172, 3000), (5172,))
```

Splitting the Dataset

Training and Test Set

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.15, random_state=8)
```

Machine Learning models

The following 5 models are used:

1. K-Nearest Neighbors
2. Linear SVM
3. Polynomial SVM
4. RBF SVM
5. Sigmoid SVM

```
models = {
    "K-Nearest Neighbors": KNeighborsClassifier(n_neighbors=2),
    "Linear SVM":LinearSVC(random_state=8, max_iter=900000),
    "Polynomical SVM":SVC(kernel="poly", degree=2, random_state=8),
    "RBF SVM":SVC(kernel="rbf", random_state=8),
    "Sigmoid SVM":SVC(kernel="sigmoid", random_state=8)
}
```

Fit and predict on each model

Each model is trained using the train set and predictions are made based on the test set. Accuracy scores are calculated for each model.

```
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 \t: {metrics.accuracy_score(y_test, y_pred)}")
```

```
Accuracy for K-Nearest Neighbors model  : 0.8878865979381443
Accuracy for Linear SVM model   : 0.9780927835051546
Accuracy for Polynomical SVM model      : 0.7615979381443299
Accuracy for RBF SVM model      : 0.8182989690721649
Accuracy for Sigmoid SVM model  : 0.6237113402061856
```