```
import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)
from sklearn.model_selection import train_test_split
from sklearn.naive_bayes import MultinomialNB
from sklearn.svm import SVC
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score
```

```
df = pd.read_csv("/content/emails.csv")
df.head(2)
```

	Email No.	the	to	ect	and	for	of	a	you	hou	...	connevey	jay	valued	lay	infrastructure	military	allowing	ff	dry	Predict
0	Email 1	0	0	1	0	0	0	2	0	0	...	0	0	0	0	0	0	0	0	0	

```
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
```

```
df.describe()
```

	the	to	ect	and	for	of	a	you	hou	
count	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	1
std	11.745009	9.534576	14.101142	6.045970	4.680522	6.229845	87.574172	4.314444	6.967878	1
min	0.000000	0.000000	1.000000	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	
50%	3.000000	3.000000	1.000000	1.000000	2.000000	1.000000	28.000000	1.000000	0.000000	
75%	8.000000	7.000000	4.000000	3.000000	4.000000	2.000000	62.250000	3.000000	1.000000	1
max	210.000000	132.000000	344.000000	89.000000	47.000000	77.000000	1898.000000	70.000000	167.000000	22

8 rows × 3001 columns

Creating the NB Model

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

	the	to	ect	and	for	of	a	you	hou	in	...	enhancements	connevey	jay	valued	lay	infrastructure	military
0	0	0	1	0	0	0	2	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
2	0	0	1	0	0	0	8	0	0	4	...	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
4	7	6	17	1	5	2	57	0	9	3	...	0	0	0	0	0	0	0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
5171	22	24	5	1	6	5	148	8	2	23	...	0	0	0	0	0	0	0

```
Y = df.iloc[:,-1].values
Y
```

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

```
train_x,test_x,train_y,test_y = train_test_split(X,Y,test_size = 0.25)
```

Naive Bayes

```
mnb = MultinomialNB(alpha=1.9)         # alpha by default is 1. alpha must always be > 0.
# alpha is the '1' in the formula for Laplace Smoothing (P(words))
mnb.fit(train_x,train_y)
y_pred1 = mnb.predict(test_x)
print("Accuracy Score for Naive Bayes : ", accuracy_score(y_pred1,test_y))
```

```
Accuracy Score for Naive Bayes :  0.9365815931941222
```

Support Vector Machines

Support Vector Machine is the most sought after algorithm for classic classification problems. SVMs work on the algorithm of Maximal Margin, i.e, to find the maximum margin or threshold between the support vectors of the two classes (in binary classification). The most effective Support vector machines are the soft maximal margin classifier, that allows one misclassification, i.e, the model starts with low bias(slightly poor performance) to ensure low variance later.

Let us see the model performance.

```
svc = SVC(C=1.0,kernel='rbf',gamma='auto')
svc.fit(train_x,train_y)
y_pred2 = svc.predict(test_x)
print("Accuracy Score for SVC : ", accuracy_score(y_pred2,test_y))
```

```
Accuracy Score for SVC :  0.888631090487239
```

As expected, SVM's performance is slightly poorer than Multinomia Naive Bayes

Random Forests (Bagging)

Ensemble methods turn any feeble model into a highly powerful one. Let us see if ensemble model can perform better than Naive Bayes

```
rfc = RandomForestClassifier(n_estimators=100,criterion='gini')
# n_estimators = No. of trees in the forest
# criterion = basis of making the decision tree split, either on gini impurity('gini'), or on infromation gain('entropy')
rfc.fit(train_x,train_y)
y_pred3 = rfc.predict(test_x)
print("Accuracy Score of Random Forest Classifier : ", accuracy_score(y_pred3,test_y))
```

```
Accuracy Score of Random Forest Classifier :  0.9682907965970611
```

Ending notes:

This was a purely comparative study to check the workability of the dataset that I created, and to check how conventional models perform on my dataset. In my next kernel, I will show the code behind the extraction of this dataset from the raw text files.

✓ 2s completed at 9:59 PM