COURSE 3

PROJECT REPORT : EMAIL SPAM CLASSIFICATION

APRIL 2022

Overview

In today’s globalized world, email is a primary source of communication. This communication can vary from personal, business, corporate to government. With the rapid increase in email usage, there has also been an increase in SPAM emails. SPAM emails, also known as junk email, involve identical messages sent to numerous recipients by email. Apart from being annoying, spam emails can also pose a security threat to computer systems. It is estimated that spam cost businesses the order of $100 billion (about$310 per person in the US) in 2007. In this project, we will try to perform automatic spam filtering to use emails effectively. We try to identify patterns using classification algorithms to enable us to classify the emails as HAM or SPAM.

About Dataset

We have taken our data set from Kaggle, and this is a csv file containing related information of 5172 randomly picked email files and their respective labels for spam or not-spam classification. The csv file contains 5172 rows, each row for each email. There are 3002 columns. The first column indicates Email name. The name has been set with numbers and not recipients' name to protect privacy. The last column has the labels for prediction: 1 for spam, 0 for not spam or Ham. The remaining 3000 columns are the 3000 most common words in all the emails, after excluding the non-alphabetical characters/words. For each row, the count of each word(column) in that email(row) is stored in the respective cells. Thus, information regarding all 5172 emails is stored in a compact data frame rather than as separate text files.

Data Preprocessing

In [1]:
```
import pandas as pd,numpy as np,matplotlib as plt, os,seaborn as sns
```

In [2]:
```
os.chdir('data')
```

In [3]:
```
data=pd.read_csv('emails.csv')
```

In [4]:
```
data.head()
```

Out[4]:

	Email No.	the	to	ect	and	for	of	a	you	hou	...	connevey	jay	valued	lay	infrastructure
0	Email 1	0	0	1	0	0	0	2	0	0	...	0	0	0	0	C
1	Email 2	8	13	24	6	6	2	102	1	27	...	0	0	0	0	C
2	Email 3	0	0	1	0	0	0	8	0	0	...	0	0	0	0	C
3	Email 4	0	5	22	0	5	1	51	2	10	...	0	0	0	0	C
4	Email 5	7	6	17	1	5	2	57	0	9	...	0	0	0	0	C

5 rows × 3002 columns

In [5]:
```
data.shape
```

Out[5]:
```
(5172, 3002)
```

In [6]:
```
data.isnull().sum().all()
```

Out[6]:
```
False
```

In [7]:
```
data.dtypes
```

Out[7]:
```
Email No.     object
the            int64
to             int64
ect            int64
and            int64
               ...
military       int64
allowing       int64
ff             int64
dry            int64
Prediction     int64
Length: 3002, dtype: object
```

In [8]:
```
data.describe()
```

Out[8]:

	the	to	ect	and	for	of	a
count	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
std	11.745009	9.534576	14.101142	6.045970	4.680522	6.229845	87.574172
min	0.000000	0.000000	1.000000	0.000000	0.000000	0.000000	0.000000
25%	0.000000	1.000000	1.000000	0.000000	1.000000	0.000000	12.000000
50%	3.000000	3.000000	1.000000	1.000000	2.000000	1.000000	28.000000
75%	8.000000	7.000000	4.000000	3.000000	4.000000	2.000000	62.250000
max	210.000000	132.000000	344.000000	89.000000	47.000000	77.000000	1898.000000

8 rows × 3001 columns

Splitting the Data Into Train and Test Set

We have split our data set into 70:30. 70 percent for training and 30 percent for testing.

In [9]:
```
from sklearn.model_selection import train_test_split
```

In [10]:
```
X=data.iloc[:,1:3001]
```

In [11]:
```
y=data.iloc[:,-1]
```

In [12]:
```
X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.3,random_state=42)
```

Model Development

SVM

In [14]:
```
from sklearn.svm import SVC
```

In [15]:
```
SVM = SVC(kernel='rbf',gamma='auto',C=10.0)
SVM = SVM.fit(X_train,y_train)
y_predict=SVM.predict(X_test)
```

In [16]:
```
from sklearn.metrics import accuracy_score
a1=accuracy_score(y_predict,y_test)
print("Accuracy Score for SVC : ", accuracy_score(y_predict,y_test))
```

```
Accuracy Score for SVC :  0.9213917525773195
```

Naive Bayes

In [18]:
```
from sklearn.naive_bayes import MultinomialNB
```

In [19]:
```
mnb = MultinomialNB(alpha=1)
mnb = mnb.fit(X_train,y_train)
y_pred2 = mnb.predict(X_test)
```

In [20]:
```
print("Accuracy Score NB : ", accuracy_score(y_pred2,y_test))
a2=accuracy_score(y_pred2,y_test)
```

```
Accuracy Score NB :  0.9484536082474226
```

As expected naive bayes performs slightly better than svc

Decision Trees

In [23]:
```
from sklearn.tree import DecisionTreeClassifier
```

In [24]:
```
DTC = DecisionTreeClassifier(criterion='gini')
DTC=DTC.fit(X_train,y_train)
y_pred3=DTC.predict(X_test)
```

In [25]:
```
print("Accuracy Score for Decision Tree", accuracy_score(y_pred3,y_test))
a3=accuracy_score(y_pred3,y_test)
```

```
Accuracy Score for Decision Tree 0.9162371134020618
```

Ensemble Models

Random Forest Classifier

In [27]:
```
from sklearn.ensemble import RandomForestClassifier
```

In [28]:
```
RFC = RandomForestClassifier(n_estimators=50)
RFC = RFC.fit(X_train,y_train)
y_pred4 = RFC.predict(X_test)
```

In [29]:
```
print("Accuracy Score for RFC",accuracy_score(y_pred4,y_test))
a4=accuracy_score(y_pred4,y_test)
```

```
Accuracy Score for RFC 0.9710051546391752
```

Extra Trees Classifier

In [30]:
```
from sklearn.ensemble import ExtraTreesClassifier
```

In [31]:
```
ETC = ExtraTreesClassifier(n_estimators=50)
ETC = ETC.fit(X_train,y_train)
y_pred5 = ETC.predict(X_test)
```

In [32]:
```
print("Accuracy Score for Extra Tree Classifier",accuracy_score(y_pred5,y_test))
a5=accuracy_score(y_pred5,y_test)
```

```
Accuracy Score for Extra Tree Classifier 0.9806701030927835
```

Boosting

In [34]:
```
from sklearn.ensemble import GradientBoostingClassifier

GBC = GradientBoostingClassifier(learning_rate=0.1,max_features=100,subsample=0.5
GBC = GBC.fit(X_train,y_train)
y_pred6 = GBC.predict(X_test)
```

In [35]:
```
print("Accuracy Score for Gradient Boosting Classifier",accuracy_score(y_pred6,y_
a6=accuracy_score(y_pred6,y_test)
```

```
Accuracy Score for Gradient Boosting Classifier 0.9748711340206185
```

Extra Tree Classifier is performing slightly better than Gradient Boosting Clsassifier

In [37]:
```
from sklearn.ensemble import AdaBoostClassifier
```

In [38]:
```
ABC = AdaBoostClassifier(base_estimator=RandomForestClassifier(),
                        learning_rate=0.001,n_estimators=200)

ABC = ABC.fit(X_train,y_train)
y_pred7 = ABC.predict(X_test)
```

In [39]:
```
print("Accuracy Score for Ada Boost Classifier",accuracy_score(y_pred7,y_test))
a7=accuracy_score(y_pred7,y_test)
```

```
Accuracy Score for Ada Boost Classifier 0.9735824742268041
```

Result and Key Findings

In [40]:
```
accuracy = {'Model':['Support Vector Machine','Naive Bayes','Decision Trees','Rar
                     'Extra Trees','Gradient Boost','Ada Boost'],
           'Accuracy Score':[a1,a2,a3,a4,a5,a6,a7]}
accuracy= pd.DataFrame(accuracy)
accuracy.set_index(['Model'],inplace=True)
accuracy
```

Out[40]:

Model	Accuracy Score
Support Vector Machine	0.921392
Naive Bayes	0.948454
Decision Trees	0.916237
Random Forest	0.971005
Extra Trees	0.980670
Gradient Boost	0.974871
Ada Boost	0.973582

As we can see from the above table that Ensemble Methods such as Random Forest gives accuracy of approximately 97 percent and Extra Trees gives best performace with an accuracy of 98 percent. It was expected that boosting methods such as Gradient Boost and Ada Boost will perform better but it is found that they performs slightly poorer than Extra Trees and slightly better than Random Forest.

Suggestions for Next Step

We have trained a pretty good model but for further improving performace of our models we can apply Deep Learning And Natural Language Processing Techniques.We have only used a section of Data Set so we can increase the size of data set.