INTRODUCTION

Email spam, also known as junk email, is unsolicited bulk messages sent through email. The use of spam has been growing in popularity since the early 1990s and is a problem faced by most email users. This is a dark side of email marketing. Since the 1990s, spam email has been becoming a more advanced phenomenon in terms of its outreach and the technical solutions for bypassing restrictions.

PROBLEM DEFINITION

The data consists of 2893 rows and 3 columns in text format out of which one feature column is label in which 0 means not spam or 1: spam message.

The columns of the dataset are:

Subject: which contains subject of the email.

Message: contains the body of the mail.

Label: 0 means no spam message 1 means spam message.

Let's now import necessary libraries to start on our dataset such as

```
#importing necessary libraries
import pandas as pd
import numpy as np
import nltk
```

DATA ANALYSIS

```
#reading the file
data=pd.read_csv('messages.csv')
data
```

	subject	message	label
0	job posting - apple-iss research center	content - length : 3386 apple-iss research cen...	0
1	NaN	lang classification grimes , joseph e . and ba...	0
2	query : letter frequencies for text identifica...	i am posting this inquiry for sergei atamas (...	0
3	risk	a colleague and i are researching the differin...	0
4	request book information	earlier this morning i was on the phone with a...	0
...	...	...	...
2888	love your profile - ysuolvpv	hello thanks for stopping by ! ! we have taken...	1
2889	you have been asked to join kiddin	the list owner of : " kiddin " has invited you...	1
2890	anglicization of composers ' names	judging from the return post , i must have sou...	0
2891	re : 6 . 797 , comparative method : n - ary co...	gotcha ! there are two separate fallacies in t...	0
2892	re : american - english in australia	hello ! i ' m working on a thesis concerning a...	0

2893 rows × 3 columns

As we are further moving to data analysis, we get to know that our data set consist of some 2893 rows and 3 columns in which 1 column is our target called as 'label'. We also observed that subject and message columns is having text data which we have to convert it into machine learning understood language.

Pre-Processing Pipeline

Data pre-processing is a data mining technique that involves transforming raw data into an understandable format. Real-world data is often incomplete, inconsistent, and/or lacking in certain behaviours or trends, and is likely to contain many errors. Data pre-processing is a proven method of resolving such issues.

```
from nltk.corpus import stopwords
from nltk.stem.porter import PorterStemmer
```

```
c=[]
for i in range(0,len(data)):
    clean1=re.sub('[^a-zA-Z]',' ',str(data['subject'][i]))
    clean1=clean1.lower()
    clean1=clean1.split()
    clean2=re.sub('[^a-zA-Z]',' ',str(data['message'][i]))
    clean2=clean2.lower()
    clean2=clean2.split()
    clean=clean1+clean2

    clean=[PorterStemmer().stem(word) for word in clean if not word in stopwords.words('english')]
    clean=' '.join(clean)
    c.append(clean)
```

Text data requires special preparation before you can start using it for predictive modeling.

The text must be parsed to remove words, called tokenization. Then the words need to be encoded as integers or floating-point values for use as input to a machine learning algorithm, called feature extraction (or vectorization).

As we can see our data is in text format for converting the text data into useful values, we will first remove unwanted small capital full stops etc from the subject and message columns. In the above step we have done this by importing stop words and Porter Stemer.

After successful removal of unwanted data now we will convert the relevant text into number form with the help of count vectorizer. For doing this we will import count vectorizer from Sklearn package as shown in below image.

The Count Vectorizer provides a simple way to both tokenize a collection of text documents and build a vocabulary of known words, but also to encode new documents using that vocabulary.

```
from sklearn.feature_extraction.text import CountVectorizer
cv=CountVectorizer(max_features=2500)
x=cv.fit_transform(c).toarray()
```

Finally, after parsing all the relevant columns text into vector, it will look like this shown below.

```
predict
```

```
array([0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0,
       0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 1, 0, 0, 0, 1, 0,
       0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0,
       1, 0, 1, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0,
       1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 1, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 1, 1, 1, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 1, 0,
       1, 0, 0, 1, 0, 1, 1, 1, 0, 0, 0, 0, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0,
       0, 1, 0, 1, 0, 1, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 1, 1, 0,
       0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0,
       0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 1, 1, 0, 0,
       0, 0, 1, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 1, 0, 0, 0, 1, 1, 1, 0,
       0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0,
       0, 0, 0, 1, 1, 0, 1, 0, 1, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0,
       0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 1, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0,
       0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0,
       0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 1, 0, 1, 0], dtype=int64)
```

Building Machine Learning Models:

Finally, now by doing this we can move further splitting our data using train test split keeping 75% of the data for training the model and rest 30% for the testing. Now our data can be passed to the required models. We will perform on 6 different models: Logistic Regression, SVC, GaussianNB, DecisionTreeClassifier, KNeighborsClassifier, RandomForestClassifier. For doing that we will import the necessary models from Sklearn. During training the models, we have trained our models with on random state=0

```
from sklearn.linear_model import LogisticRegression
from sklearn.svm import SVC
from sklearn.naive_bayes import GaussianNB
from sklearn.tree import DecisionTreeClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.ensemble import RandomForestClassifier
```

After importing the models we will run our training and testing data into the models to check which one model out from the six models is performing the best for our given dataset.

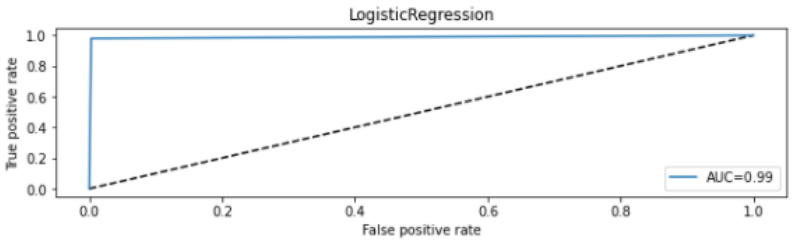
```
list=[LogisticRegression(),GaussianNB(),SVC(),DecisionTreeClassifier(),KNeighborsClassifier(),RandomForestClassifier()]
for g in list:
    print('for the algorithm: ',g)
    x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=.30,random_state=0)
    g.fit(x_train,y_train)
    pred=g.predict(x_test)
    accuracy=accuracy_score(pred,y_test)
    scr=g.score(x_train,y_train)
    pred=g.predict(x_test)
    accuracy=accuracy_score(pred,y_test)
    print('The score is: ',scr)
    print('Accuracy score is: ',accuracy)
    print('confusion matrix: \n',confusion_matrix(pred,y_test))
    print('classification report: \n',classification_report(pred,y_test))
```

After running the models, we got to know that Logistic Regression model is performing well from all models. As the accuracy score for it is 0.9942396313364056 which is the best among all models. So now moving further with this step now we will run some multiple metrics on these 6 models such as f1 score, roc,auc, confusion matrix. These all metrics helps in building a close to perfect model. As f1 score do give precise check of score auc roc curve show if the model is under fit or overfit.

Concluding Marks

Performing fitting of data into different models, applying multiple matrices for better check of models, we observe that the best following results goes with the logistic regression model shown below.

```
LogisticRegression

LogisticRegression()


Accuracy Score:  0.9942396313364056


F1 Score:  0.9826989619377162
AUC_ROC_Score 0.9882720465493395


[[721   2]
 [  3 142]]


Classification Report:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00       723
           1       0.99      0.98      0.98       145

    accuracy                           0.99       868
   macro avg       0.99      0.99      0.99       868
weighted avg       0.99      0.99      0.99       868


AxesSubplot(0.125,0.808774;0.775x0.0712264)
```



Hence, final concluding with the logistic Regression model which predicted the result without overfitting the data and better than other models.

You can go through the complete codes of the above program in given Github link:

https://github.com/sachinkoli92/Email-spam-detection-Project/blob/main/spam%20detection.ipynb