Bhumi Reddy Chenna Kesava Reddy

Task-01: Email Spam Detection

Dataset link: https://www.kaggle.com/datasets/venky73/spam-mails-dataset?resource=download

SIMPLBYTE

Import Libraries

In [10]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
```

Import Dataset

In [11]:
```
df=pd.read_csv("spam_ham_dataset.csv")
df
```

Out[11]:

	Unnamed: 0	label	text	label_num
0	605	ham	Subject: enron methanol ; meter # : 988291\r\n...	0
1	2349	ham	Subject: hpl nom for january 9 , 2001\r\n(see...	0
2	3624	ham	Subject: neon retreat\r\nho ho ho , we ' re ar...	0
3	4685	spam	Subject: photoshop , windows , office . cheap ...	1
4	2030	ham	Subject: re : indian springs\r\nthis deal is t...	0
...	...	...	...	...
5166	1518	ham	Subject: put the 10 on the ft\r\nthe transport...	0
5167	404	ham	Subject: 3 / 4 / 2000 and following noms\r\nhp...	0
5168	2933	ham	Subject: calpine daily gas nomination\r\n>\r\n...	0
5169	1409	ham	Subject: industrial worksheets for august 2000...	0
5170	4807	spam	Subject: important online banking alert\r\ndea...	1

5171 rows × 4 columns

In [12]:
```
df.shape
```

Out[12]:
```
(5171, 4)
```

In [13]:
```
df.columns
```

Out[13]:
```
Index(['Unnamed: 0', 'label', 'text', 'label_num'], dtype='object')
```

In [14]:
```
df.isna()
```

Out[14]:

	Unnamed: 0	label	text	label_num
0	False	False	False	False
1	False	False	False	False
2	False	False	False	False
3	False	False	False	False
4	False	False	False	False
...	...	...	...	...
5166	False	False	False	False
5167	False	False	False	False
5168	False	False	False	False
5169	False	False	False	False
5170	False	False	False	False

5171 rows × 4 columns

In [15]:
```
df.isna().sum()
```

Out[15]:
```
Unnamed: 0    0
label         0
text          0
label_num     0
dtype: int64
```

In [16]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5171 entries, 0 to 5170
Data columns (total 4 columns):
 #   Column      Non-Null Count  Dtype 
---  ------      --------------  ----- 
 0   Unnamed: 0  5171 non-null   int64 
 1   label       5171 non-null   object
 2   text        5171 non-null   object
 3   label_num   5171 non-null   int64 
dtypes: int64(2), object(2)
memory usage: 161.7+ KB
```

In [17]:
```
df.describe()
```

Out[17]:

	Unnamed: 0	label_num
count	5171.000000	5171.000000
mean	2585.000000	0.289886

	Unnamed: 0	label_num
std	1492.883452	0.453753
min	0.000000	0.000000
25%	1292.500000	0.000000
50%	2585.000000	0.000000
75%	3877.500000	1.000000
max	5170.000000	1.000000

Cleaning the data

In [18]:

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

Downloading the stopwords package

In [21]:

```
nltk.download('stopwords')
```

```
[nltk_data] Downloading package stopwords to
[nltk_data]     C:\Users\chenn\AppData\Roaming\nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
```

Out[21]:

```
True
```

Remove all characters excluding Alphabets, stopwords and stem words

In [22]:

```
def email(text):
  email_review=re.sub('[^a-zA-Z]',' ',text)
  email_review=email_review.lower().split()
  email_review.remove('subject')
  ps=PorterStemmer()
  all_stopwords=stopwords.words('english')
  all_stopwords.remove('not')
  email_review=[ps.stem(word) for word in email_review if word not in set(all_stopwo
  return email_review
df['text'].head(10).apply(email)
```

Out[22]:

```
0    [enron, methanol, meter, follow, note, gave, m...
1    [hpl, nom, januari, see, attach, file, hplnol,...
2    [neon, retreat, ho, ho, ho, around, wonder, ti...
3    [photoshop, window, offic, cheap, main, trend,...
4    [indian, spring, deal, book, teco, pvr, revenu...
5    [ehronlin, web, address, chang, messag, intend...
6    [spring, save, certif, take, save, use, custom...
7    [look, medic, `, best, sourc, difficult, make,...
8    [nom, actual, flow, agre, forward, melissa, jo...
9    [nomin, oct, see, attach, file, hplnl, xl, hpl...
Name: text, dtype: object
```

Bag of words creation

In [23]:
```
from sklearn.feature_extraction.text import CountVectorizer
x=CountVectorizer(analyzer=email).fit_transform(df['text']).toarray()
y=df.iloc[:,-1].values
```

In [24]:
```
print(x.shape)
print()
print(x)
```

```
(5171, 37917)

[[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 0 ... 0 0 0]]
```

In [28]:
```
print(y.shape)
print()
print(y)
```

```
(5171,)

[0 0 0 ... 0 0 1]
```

Splitting the data into 70% training and 30% test dataset

In [29]:
```
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.3,random_state=42)
```

Implementing Naive bayes Classification

In [30]:
```
from sklearn.naive_bayes import GaussianNB
classifier=GaussianNB()
classifier.fit(x_train,y_train)
```

Out[30]:
```
GaussianNB()
```

predicting the Test Results

In [31]:
```
y_pred=classifier.predict(x_test)
print(np.concatenate((y_pred.reshape(len(y_pred),1),y_test.reshape(len(y_test),1)),1
```

```
[[0 0]
 [1 1]
 [0 0]
 ...
 [0 0]
 [0 1]
 [0 0]]
```

Evaluating the performance of the model

In [32]:
```
from sklearn.metrics import accuracy_score,confusion_matrix,classification_report
cm=confusion_matrix(y_pred,y_test)
```

```
ac=accuracy_score(y_pred,y_test)
cr=classification_report(y_pred,y_test)
print('Classification report is:',cr)
print()
print('confusion matrix is:\n',cm)
print()
print('Accuracy score is:\n',ac)
```

```
Classification report is:                precision    recall  f1-score   support

           0       0.98      0.96      0.97      1138
           1       0.90      0.93      0.92       414

    accuracy                           0.95      1552
   macro avg       0.94      0.95      0.94      1552
weighted avg       0.96      0.95      0.95      1552


confusion matrix is:
 [[1094   44]
 [  27  387]]

Accuracy score is:
 0.9542525773195877
```

Accuracy=95%

Thank you