Summary

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. 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 are stored in a compact dataframe rather than as separate text files.

Presented By: Asad

Importing Libraries

In [1]:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
%matplotlib inline
import seaborn as sns
```

Exploring the data

In [2]:

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

In [3]:

```
df.head() # 0 for ham, 1 for spam
```

Out[3]:

	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

In [4]:

```
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 [5]:

```
df.drop(['Unnamed: 0','label'],axis=1,inplace=True)
```

In [6]:

```
df.head()
```

Out[6]:

	text	label_num
0	Subject: enron methanol ; meter # : 988291\r\n...	0
1	Subject: hpl nom for january 9 , 2001\r\n(see...	0
2	Subject: neon retreat\r\nho ho ho , we ' re ar...	0
3	Subject: photoshop , windows , office . cheap ...	1
4	Subject: re : indian springs\r\nthis deal is t...	0

Visualizing the data

In [7]:

```
df['label_num'].value_counts()
```

Out[7]:

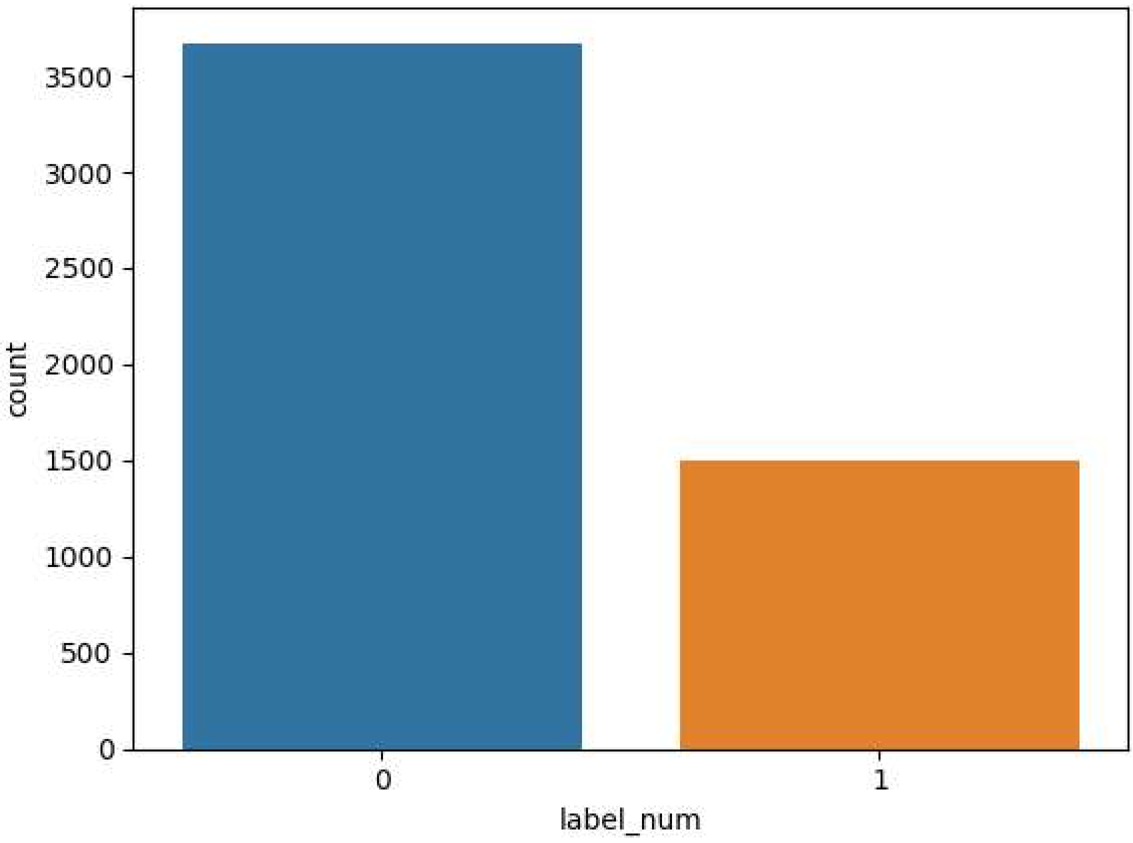
```
label_num
0    3672
1    1499
Name: count, dtype: int64
```

In [8]:

```
sns.countplot(x=df['label_num'])
```

Out[8]:

```
<Axes: xlabel='label_num', ylabel='count'>
```



Text Feature Extraction

In [9]:

```
from sklearn.feature_extraction.text import CountVectorizer
```

In [10]:

```
X=df['text']
y=df['label_num']
vec=CountVectorizer()
X_count=vec.fit_transform(X)
```

Splitting the data

In [11]:

```
from sklearn.model_selection import train_test_split
```

In [12]:

```
X_train, X_test, y_train, y_test =train_test_split(X_count,y,test_size=0.25,random_
```

Instantiating the model

In [13]:

```
from sklearn.naive_bayes import MultinomialNB
```

In [14]:

```
model=MultinomialNB()
```

Training and Testing

In [15]:

```
model.fit(X_train,y_train)
```

Out[15]:

```
▾ MultinomialNB
MultinomialNB()
```

In [16]:

```
predictions=model.predict(X_test)
```

In [17]:

```
predicted_df=pd.DataFrame({'Predicted':predictions,'Actual':y_test})
```

In [18]:

```
predicted_df.head(10)
```

Out[18]:

	Predicted	Actual
1309	0	0
4407	1	1
2577	1	1
1332	1	1
94	1	1
1623	1	1
4178	1	1
3476	0	0
4834	0	0
234	0	0

Report of the model

In [19]:

```
from sklearn.metrics import ConfusionMatrixDisplay,confusion_matrix,classification_
```

In [20]:

```
print(classification_report(y_test,predictions))
```

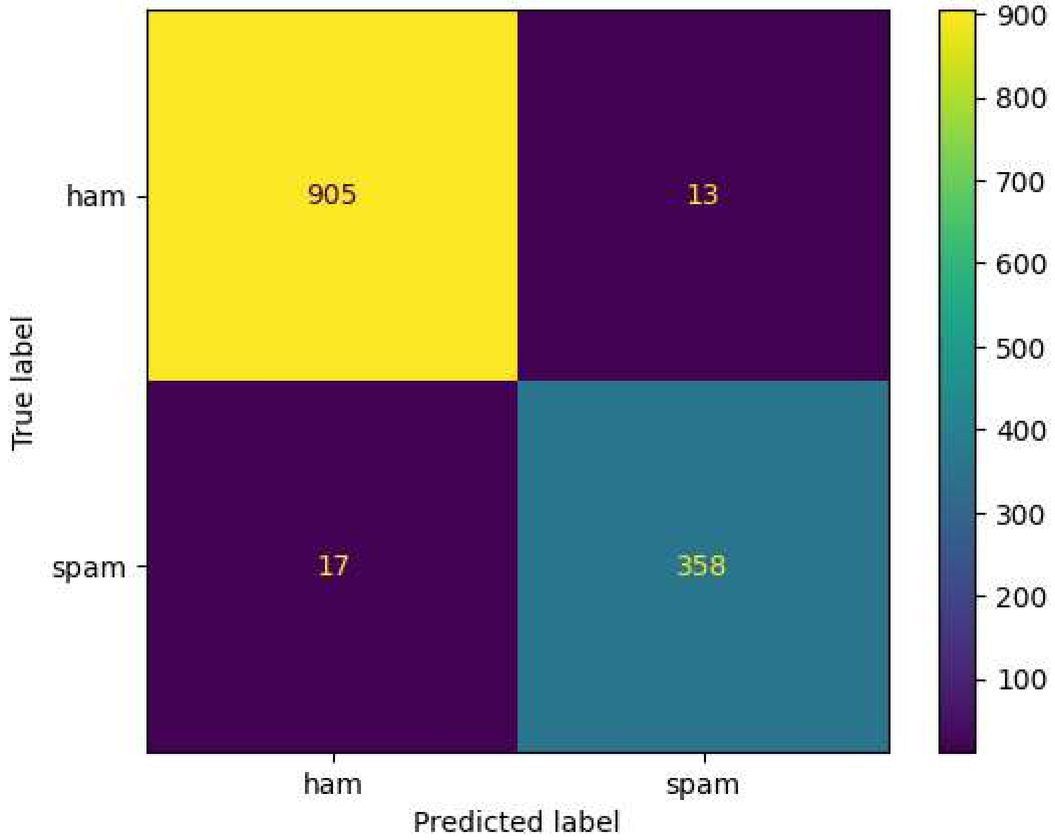
```
              precision    recall  f1-score   support

           0       0.98      0.99      0.98       918
           1       0.96      0.95      0.96       375

    accuracy                           0.98      1293
   macro avg       0.97      0.97      0.97      1293
weighted avg       0.98      0.98      0.98      1293
```

In [21]:

```
cm=confusion_matrix(y_test,predictions)
disp=ConfusionMatrixDisplay(cm,display_labels=['ham','spam'])
disp.plot()
```

Out[21]:

```
<sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay at 0x20b48fbe910>
```



Testing with RealTime data

In [22]:
```
mail="""Subject: Congratulations! You have won a free trip to Hawaii! Body: Dear Va

You are one of the lucky winners of our monthly sweepstakes! You have won a free tr

This is a once-in-a-lifetime opportunity to enjoy the sun, sand, and surf of Hawaii

Sincerely, The Travel Club"""
```

In [23]:
```
mail_count=vec.transform([mail])
```

In [24]:
```
if model.predict(mail_count)==1:
    print('Given mail is Spam')
else:
    print('Given mail is not Spam')
```

```
Given mail is Spam
```