In [1]:
```
import numpy as np
import pandas as pd
```

In [2]:
```
dt=pd.read_csv("sms.tsv",delimiter='\t',header=None)
```

In [3]:
```
dt.head(10)
```

Out[3]:
```
      0                                                  1
0   ham    Go until jurong point, crazy.. Available only ...
1   ham                        Ok lar... Joking wif u oni...
2  spam    Free entry in 2 a wkly comp to win FA Cup fina...
3   ham    U dun say so early hor... U c already then say...
4   ham    Nah I don't think he goes to usf, he lives aro...
5  spam    FreeMsg Hey there darling it's been 3 week's n...
6   ham    Even my brother is not like to speak with me. ...
7   ham    As per your request 'Melle Melle (Oru Minnamin...
8  spam    WINNER!! As a valued network customer you have...
9  spam    Had your mobile 11 months or more? U R entitle...
```

In [18]:
```
dt.tail(15)
```

Out[18]:

	0	1
5557	ham	No. I meant the calculation is the same. That ...
5558	ham	Sorry, I'll call later
5559	ham	if you aren't here in the next &lt;#&gt; hou...
5560	ham	Anything lor. Juz both of us lor.
5561	ham	Get me out of this dump heap. My mom decided t...
5562	ham	Ok lor... Sony ericsson salesman... I ask shuh...
5563	ham	Ard 6 like dat lor.
5564	ham	Why don't you wait 'til at least wednesday to ...
5565	ham	Huh y lei...
5566	spam	REMINDER FROM O2: To get 2.50 pounds free call...
5567	spam	This is the 2nd time we have tried 2 contact u...
5568	ham	Will ü b going to esplanade fr home?
5569	ham	Pity, * was in mood for that. So...any other s...
5570	ham	The guy did some bitching but I acted like i'd...
5571	ham	Rofl. Its true to its name

In [4]:
```
dt.columns=['categorry','email']
```

In [5]:
```
import string
import re
import nltk
nltk.download('stopwords')

```

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

Out[5]:
```
True
```

In [6]:
```
from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.feature_extraction.text import  CountVectorizer
```

In [7]:
```
#sw=nltk.corpus.stopwords.words('english')
sw= nltk.corpus.stopwords.words('english')
ps=nltk.PorterStemmer()
analyzer = CountVectorizer().build_analyzer()
```

In [19]:
```
ps
```
Out[19]:
```
<PorterStemmer>
```

In [8]:
```
def clean_text(text) :
    ### Stemming of words
    stemmed_words = (ps.stem(w) for w in analyzer(text))
    ### Remove the words in stop words list
    non_stop_words = [ word for word in list(set(stemmed_words) - set(sw
    return non_stop_words

```

In [9]:
```
tfidf_vectorizer = TfidfVectorizer( analyzer=clean_text,max_features = 1
```

In [10]:
```
feature_vector = tfidf_vectorizer.fit_transform( dt['email'] )
```

In [11]:
```
features = tfidf_vectorizer.get_feature_names()
```

In [12]:
```
dt_dataframe=pd.DataFrame(feature_vector.toarray(),columns=features)
```

In [13]:
```
dt_dataframe.head(15)
```

Out[13]:

	00	000	03	04	0800	08000839402	08000930705	08712460324	10	100	...
0	0.0	0.000000	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.000000	...
1	0.0	0.000000	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.000000	...
2	0.0	0.000000	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.000000	...
3	0.0	0.000000	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.000000	...
4	0.0	0.000000	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.000000	...
5	0.0	0.000000	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.000000	...
6	0.0	0.000000	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.000000	...
7	0.0	0.000000	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.000000	...
8	0.0	0.000000	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.000000	...
9	0.0	0.000000	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.000000	...
10	0.0	0.000000	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.000000	...
11	0.0	0.279921	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.261887	...
12	0.0	0.332396	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.310981	...
13	0.0	0.000000	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.000000	...
14	0.0	0.000000	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.000000	...

15 rows × 1000 columns

In [14]:
```
dt_dataframe.shape
```

Out[14]:
```
(5572, 1000)
```

In [15]:
```
dt['categorry']=dt['categorry'].replace(['ham','spam'],[0,1])
```

In [17]:
```
dt['categorry'].value_counts()
```

Out[17]:
```
0    4825
1     747
Name: categorry, dtype: int64
```

In [18]:
```
X_train,X_test,y_train,y_test=train_test_split(dt_dataframe,dt['categorr
```

In [21]:
```
print(len(X_train))
print(len(X_test))
print(len(y_train))
print(len(y_test))
```

```
4457
1115
4457
1115
```

In [22]:
```
from sklearn.linear_model import LogisticRegression
```

In [23]:
```
logreg=LogisticRegression()

```

In [24]:
```
logreg.fit(X_train,y_train)
```

Out[24]:
```
LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=Tru
e,
                   intercept_scaling=1, l1_ratio=None, max_iter=100,
                   multi_class='auto', n_jobs=None, penalty='l2',
                   random_state=None, solver='lbfgs', tol=0.0001, verbose=
0,
                   warm_start=False)
```

In [25]:
```
test_predicted=logreg.predict(X_test)
```

In [26]:
```
test_predicted
```

Out[26]:
```
array([0, 0, 0, ..., 0, 0, 0], dtype=int64)
```

In [27]:
```
from sklearn import metrics
```

In [28]:
```
print(metrics.classification_report(y_test,test_predicted))
```

```
              precision    recall  f1-score   support

           0       0.97      1.00      0.99       962
           1       0.98      0.82      0.90       153

    accuracy                           0.97      1115
   macro avg       0.98      0.91      0.94      1115
weighted avg       0.97      0.97      0.97      1115
```

In [30]:
```
print(metrics.confusion_matrix(y_test,test_predicted))
```

```
[[960   2]
 [ 27 126]]
```