In [1]:
```
import pandas as pd
from nltk.corpus import stopwords
```

## 1. Open “SMSSpamCollection” file and load into DataFrame. It contains two columns “label” and “text" ¶

In [2]:
```
df = pd.read_csv("SMSSpamCollection.csv")
```

In [3]:
```
sms=df.drop(['Unnamed: 2','Unnamed: 3','Unnamed: 4'],axis=1)
sms
```

Out[3]:

| | label | text |
|---|---|---|
| **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... |
| **...** | ... | ... |
| **5567** | spam | This is the 2nd time we have tried 2 contact u... |
| **5568** | ham | Will  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 |

5572 rows × 2 columns

## 2. How many sms messages are there?

In [4]:
```
len(sms)
```

Out[4]:
```
5572
```

## 3. How many “ham” and “spam” messages?. You need to groupby() label column.

In [5]:
```
a=sms.groupby('label').count()
a
```

Out[5]:

| label | text |
|---|---|
| ham | 4825 |
| spam | 747 |

## 4. Split the dataset into training set and test set (Use 20% of data for testing).

In [6]:
```
X = sms.text
y = sms.label
```

In [7]:
```
from sklearn.model_selection import train_test_split
```

In [8]:
```
X_train,X_test,y_train,y_test = train_test_split(X,y,train_size=0.8,test_size=0.2
```

## 5. Create a function that will remove all punctuation characters and stop words, as below

In [9]:
```
def process_text(msg):
    punctuations = '''!()-[]{};:'"\,<>./?@#$%^&*_~'''
    nopunc =[char for char in msg if char not in punctuations]
    nopunc=''.join(nopunc)
    return [word for word in nopunc.split()
    if word.lower() not in stopwords.words('english')]
```

## 6. Create TfIdfVectorizer as below and perform vectorization on X_train, using fit_perform() method.

In [10]:
```
from sklearn.feature_extraction.text import TfidfVectorizer
```

In [11]:
```
tf2=TfidfVectorizer(use_idf=True,analyzer=process_text,ngram_range=(1,3),min_df =
tf2
```

Out[11]:
```
TfidfVectorizer(analyzer=<function process_text at 0x000001D510FDA820>,
                ngram_range=(1, 3), stop_words='english')
```

In [12]:
```
m2=tf2.fit_transform(X_train)
my2=tf2.transform(X_test)
```

In [13]:
```
m2.shape
```

Out[13]:
```
(4457, 9960)
```

In [14]:
```
my2.shape
```

Out[14]:
```
(1115, 9960)
```

## 7. Create MultinomialNB model and perform training on X_train and y_train using fit() method

In [15]:
```
x_train,x_test,Y_train,Y_test = train_test_split(X,y,train_size=0.8,test_size=0.2
```

In [16]:
```
from sklearn.naive_bayes import MultinomialNB
```

In [17]:
```
clf = MultinomialNB()
```

In [18]:
```
clf.fit(m2,y_train)
```

Out[18]:
```
MultinomialNB()
```

## 8. Predict labels on the test set, using predict() method

In [19]:
```
y_pred = clf.predict(my2)
y_pred
```

Out[19]:
```
array(['ham', 'ham', 'ham', ..., 'ham', 'ham', 'ham'], dtype='<U4')
```

## 9. Print confusion_matrix and classification_report

In [20]:
```
from sklearn.metrics import confusion_matrix
```

In [21]:
```
confusion_matrix(y_test,y_pred)
```

Out[21]:
```
array([[952,   0],
       [ 47, 116]], dtype=int64)
```

In [22]:
```
from sklearn.metrics import classification_report
```

In [32]:
```
target_names = ['class 0', 'class 1']
print(classification_report(y_test,y_pred,target_names=target_names))
```

```
              precision    recall  f1-score   support

     class 0       0.95      1.00      0.98       952
     class 1       1.00      0.71      0.83       163

    accuracy                           0.96      1115
   macro avg       0.98      0.86      0.90      1115
weighted avg       0.96      0.96      0.95      1115
```

## 10. Modify ngram_range=(1,2) and perform Steps 7 to 9.

In [24]:
```
tf3=TfidfVectorizer(use_idf=True,analyzer=process_text,ngram_range=(1,2),min_df =
tf3
```

Out[24]:
```
TfidfVectorizer(analyzer=<function process_text at 0x000001D510FDA820>,
                ngram_range=(1, 2), stop_words='english')
```

In [25]:
```
m3=tf3.fit_transform(X_train)
my3=tf3.transform(X_test)
```

In [26]:
```
m3.shape
```

Out[26]:
```
(4457, 9960)
```

In [27]:
```
my3.shape
```

Out[27]:
```
(1115, 9960)
```

In [28]:
```
clf.fit(m3,y_train)
```

Out[28]:
```
MultinomialNB()
```

In [29]:
```
y_pred3 = clf.predict(my3)
y_pred3
```

Out[29]:
```
array(['ham', 'ham', 'ham', ..., 'ham', 'ham', 'ham'], dtype='<U4')
```

In [30]:
```
confusion_matrix(y_test,y_pred3)
```

Out[30]:
```
array([[952,   0],
       [ 47, 116]], dtype=int64)
```

In [31]:
```
target_names = ['class 0', 'class 1']
print(classification_report(y_test,y_pred3,target_names=target_names))
```

```
              precision    recall  f1-score   support

     class 0       0.95      1.00      0.98       952
     class 1       1.00      0.71      0.83       163

    accuracy                           0.96      1115
   macro avg       0.98      0.86      0.90      1115
weighted avg       0.96      0.96      0.95      1115
```