Spam Ham Classification

In [2]:
```
# import library
import pandas as pd
import string
from nltk.corpus import stopwords
```

In [3]:
```
# Load the data
df_spam = pd.read_csv('SpamCollection', sep='\t',names=['response','message'])
# sep is separator \t is tab, names is defining columns
```

In [4]:
```
# first five records
df_spam.head()
```

Out[4]:

	response	message
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...

In [5]:
```
# view more information
df_spam.describe()
```

Out[5]:

	response	message
count	5572	5572
unique	2	5169
top	ham	Sorry, I'll call later
freq	4825	30

In [6]:
```
# value_counts
df_spam['response'].value_counts()
```

Out[6]:
```
ham     4825
spam     747
Name: response, dtype: int64
```

In [7]:
```
# view response
df_spam.groupby('response').describe()
```

Out[7]:

	message			
	count	unique	top	freq
response				
ham	4825	4516	Sorry, I'll call later	30
spam	747	653	Please call our customer service representativ...	4

In [8]:
```
# calculate the length of each message
df_spam['length']= df_spam['message'].apply(len)
```

In [9]:
```
# view the first five message
df_spam.head()
```

Out[9]:

	response	message	length
0	ham	Go until jurong point, crazy.. Available only ...	111
1	ham	Ok lar... Joking wif u oni...	29
2	spam	Free entry in 2 a wkly comp to win FA Cup fina...	155
3	ham	U dun say so early hor... U c already then say...	49
4	ham	Nah I don't think he goes to usf, he lives aro...	61

In [10]:
```
# define a function to remove punctuations and stopwords
def message_text_process(mess):
  # remove punctuation
  no_punc = [char for char in mess if char not in string.punctuation]  # using list comprehension
  # join to form sentences
  no_punc = ''.join(no_punc)
  # now eliminate stop words
  return[word for word in no_punc.split() if word.lower not in stopwords.words('english')] # using list comprehension
```

In [12]:
```
# verify that is working
import nltk
nltk.download('stopwords')
df_spam['message'].head().apply(message_text_process)
```

```
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
```

Out[12]:
```
0    [Go, until, jurong, point, crazy, Available, o...
1                      [Ok, lar, Joking, wif, u, oni]
2    [Free, entry, in, 2, a, wkly, comp, to, win, F...
3    [U, dun, say, so, early, hor, U, c, already, t...
4    [Nah, I, dont, think, he, goes, to, usf, he, l...
Name: message, dtype: object
```

List comprehension

In [13]:
```
# expaination of code message_text_process
# Calculate square of numbers from 1 to 9
sq= [] # empty list
for x in range(10):
  sq.append(x**2)
sq
```

Out[13]:
```
[0, 1, 4, 9, 16, 25, 36, 49, 64, 81]
```

In [14]:
```
# same code using list comprehension
sq_lc = [x**2 for x in range(10)]
sq_lc
```

Out[14]:
```
[0, 1, 4, 9, 16, 25, 36, 49, 64, 81]
```

In [15]:
```
# filter the negative numbers
vec= [-4,-2,0,2,4]
vec_pos = [x for x in vec if x>=0]
vec_pos
```

Out[15]:
```
[0, 2, 4]
```

In [16]:
```
vec_p=[]
for x in vec:
  if x>=0:
    vec_p.append(x)
vec_p
```

Out[16]:
```
[0, 2, 4]
```

In [18]:
```
# import library for count vectorizer
from sklearn.feature_extraction.text import CountVectorizer
```

In [19]:
```
# bag of word using count vectoriser
bag_of_words = CountVectorizer(analyzer=message_text_process).fit(df_spam['message'])
```

In [20]:
```
# print the length of bag_of_words
print(len(bag_of_words.vocabulary_))
```

```
11747
```

In [21]:
```
# transform the bag_of_words
message_bagofwords = bag_of_words.transform(df_spam['message'])
```

In [22]:
```
# apply tfidf transformer
from sklearn.feature_extraction.text import TfidfTransformer
tdidf_transformer = TfidfTransformer().fit(message_bagofwords)
```

In [23]:
```
# transform
message_tfidf=tdidf_transformer.transform(message_bagofwords)
```

In [24]:
```
# shape
print(message_tfidf.shape)
```

```
(5572, 11747)
```

In [25]:
```
#import Naive Bayes algorithm for classification
from sklearn.naive_bayes import MultinomialNB
spam_detect_model = MultinomialNB().fit(message_tfidf,df_spam['response'])
```

In [27]:
```
# check model for predcited and actual values for messsage 5
message = df_spam['message'][4]
message
```

Out[27]:
```
"Nah I don't think he goes to usf, he lives around here though"
```

In [28]:
```
# predict using model
bag_of_words_for_message = bag_of_words.transform([message])
tfidf = tdidf_transformer.transform(bag_of_words_for_message)

print('predicted' , spam_detect_model.predict(tfidf)[0])
print('actual' , df_spam.response[4])
```

```
predicted ham
actual ham
```

In [29]:
```
# check model for predcited and actual values for messsage2
message = df_spam['message'][1]
message
```

Out[29]:
```
'Ok lar... Joking wif u oni...'
```

In [30]:
```
# predict using model
bag_of_words_for_message = bag_of_words.transform([message])
tfidf = tdidf_transformer.transform(bag_of_words_for_message)

print('predicted' , spam_detect_model.predict(tfidf)[0])
print('actual' , df_spam.response[1])
```

```
predicted ham
actual ham
```