In [1]:
```python
import pandas as pd

sms_spam = pd.read_csv('spam - spam.csv')

print(sms_spam.shape)
sms_spam.head()
```

```
(5572, 2)
```

Out[1]:

	Category	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 [3]:
```python
sms_spam['Category'].value_counts(normalize=True)
```

Out[3]:
```
Category
ham     0.865937
spam    0.134063
Name: proportion, dtype: float64
```

In [5]:
```python
# Randomize the dataset
data_randomized = sms_spam.sample(frac=1, random_state=1)

# Calculate index for split
training_test_index = round(len(data_randomized) * 0.8)

# Split into training and test sets
training_set = data_randomized[:training_test_index].reset_index(drop=True)
test_set = data_randomized[training_test_index:].reset_index(drop=True)

print(training_set.shape)
print(test_set.shape)
```

```
(4458, 2)
(1114, 2)
```

In [7]:
```python
training_set['Category'].value_counts(normalize=True)
test_set['Category'].value_counts(normalize=True)
```

Out[7]:
```
Category
ham     0.868043
spam    0.131957
Name: proportion, dtype: float64
```

In [9]:
```python
training_set.head(3)
```

Out[9]:

	Category	Message
0	ham	Yep, by the pretty sculpture
1	ham	Yes, princess. Are you going to make me moan?
2	ham	Welp apparently he retired

In [15]:
```
training_set['Message'] = training_set['Message'].str.replace(r'[^\w\s]', ' ', rege
training_set['Message'] = training_set['Message'].str.lower()  # Converts text to l
training_set.head(3)
```

Out[15]:

	Category	Message
0	ham	yep by the pretty sculpture
1	ham	yes princess are you going to make me moan
2	ham	welp apparently he retired

In [22]:
```
training_set['Message'] = training_set['Message'].str.split()

vocabulary = []
for sms in training_set['Message']:
   for word in sms:
      vocabulary.append(word)

vocabulary = list(set(vocabulary))
```

In [24]:
```
len(vocabulary)
```

Out[24]:
```
7778
```

In [26]:
```
word_counts_per_sms = {'secret': [2,1,1],
                       'prize': [2,0,1],
                       'claim': [1,0,1],
                       'now': [1,0,1],
                       'coming': [0,1,0],
                       'to': [0,1,0],
                       'my': [0,1,0],
                       'party': [0,1,0],
                       'winner': [0,0,1]
                      }

word_counts = pd.DataFrame(word_counts_per_sms)
word_counts.head()
```

```
Out[26]:
```

	secret	prize	claim	now	coming	to	my	party	winner
0	2	2	1	1	0	0	0	0	0
1	1	0	0	0	1	1	1	1	0
2	1	1	1	1	0	0	0	0	1

```
In [30]: word_counts_per_sms = {unique_word: [0] * len(training_set['Message']) for unique_w

         for index, sms in enumerate(training_set['Message']):
            for word in sms:
               word_counts_per_sms[word][index] += 1
```

```
In [32]: word_counts = pd.DataFrame(word_counts_per_sms)
         word_counts.head()
```

```
Out[32]:
```

	wiv	50perwksub	thin	into	dom	08717111821	pours	something	letter	tb	...	ba
0	0	0	0	0	0	0	0	0	0	0	...	
1	0	0	0	0	0	0	0	0	0	0	...	
2	0	0	0	0	0	0	0	0	0	0	...	
3	0	0	0	0	0	0	0	0	0	0	...	
4	0	0	0	0	0	0	0	0	0	0	...	

5 rows × 7778 columns

```
In [34]: training_set_clean = pd.concat([training_set, word_counts], axis=1)
         training_set_clean.head()
```

Out[34]:

	Category	Message	wiv	50perwksub	thin	into	dom	08717111821	pours	someth
0	ham	[yep, by, the, pretty, sculpture]	0	0	0	0	0	0	0	
1	ham	[yes, princess, are, you, going, to, make, me,...	0	0	0	0	0	0	0	
2	ham	[welp, apparently, he, retired]	0	0	0	0	0	0	0	
3	ham	[havent]	0	0	0	0	0	0	0	
4	ham	[i, forgot, 2, ask, ü, all, smth, there, s, a,...	0	0	0	0	0	0	0	

5 rows × 7780 columns

In [38]:

```python
# Isolating spam and ham messages first
spam_messages = training_set_clean[training_set_clean['Category'] == 'spam']
ham_messages = training_set_clean[training_set_clean['Category'] == 'ham']

# P(Spam) and P(Ham)
p_spam = len(spam_messages) / len(training_set_clean)
p_ham = len(ham_messages) / len(training_set_clean)

# N_Spam
n_words_per_spam_message = spam_messages['Message'].apply(len)
n_spam = n_words_per_spam_message.sum()

# N_Ham
n_words_per_ham_message = ham_messages['Message'].apply(len)
n_ham = n_words_per_ham_message.sum()

# N_Vocabulary
n_vocabulary = len(vocabulary)

# Laplace smoothing
alpha = 1
```

In [40]:

```python
# Initiate parameters
parameters_spam = {unique_word:0 for unique_word in vocabulary}
parameters_ham = {unique_word:0 for unique_word in vocabulary}

# Calculate parameters
```

```python
for word in vocabulary:
   n_word_given_spam = spam_messages[word].sum() # spam_messages already defined
   p_word_given_spam = (n_word_given_spam + alpha) / (n_spam + alpha*n_vocabulary)
   parameters_spam[word] = p_word_given_spam

   n_word_given_ham = ham_messages[word].sum() # ham_messages already defined
   p_word_given_ham = (n_word_given_ham + alpha) / (n_ham + alpha*n_vocabulary)
   parameters_ham[word] = p_word_given_ham
```

In [44]:
```python
import re

def classify(message):
   '''
   message: a string
   '''

   # Use raw string for regular expression
   message = re.sub(r'\W', ' ', message)  # Removes punctuation
   message = message.lower().split()

   p_spam_given_message = p_spam
   p_ham_given_message = p_ham

   for word in message:
      if word in parameters_spam:
         p_spam_given_message *= parameters_spam[word]

      if word in parameters_ham:
         p_ham_given_message *= parameters_ham[word]

   print('P(Spam|message):', p_spam_given_message)
   print('P(Ham|message):', p_ham_given_message)

   if p_ham_given_message > p_spam_given_message:
      print('Label: Ham')
   elif p_ham_given_message < p_spam_given_message:
      print('Label: Spam')
   else:
      print('Equal probabilities, have a human classify this!')
```

In [46]:
```python
classify('WINNER!! This is the secret code to unlock the money: C3421.')
```

```
P(Spam|message): 1.2923061134414878e-25
P(Ham|message): 1.938145870890239e-27
Label: Spam
```

In [48]:
```python
classify("Sounds good, Tom, then see u there")
```

```
P(Spam|message): 2.423500921528076e-25
P(Ham|message): 3.689516028273414e-21
Label: Ham
```

In [62]:
```python
import re

def classify_test_set(message):
   '''
```

```
    message: a string
    '''

    # Use raw string for regular expression
    message = re.sub(r'\W', ' ', message)  # Removes punctuation
    message = message.lower().split()

    p_spam_given_message = p_spam
    p_ham_given_message = p_ham

    for word in message:
        if word in parameters_spam:
            p_spam_given_message *= parameters_spam[word]

        if word in parameters_ham:
            p_ham_given_message *= parameters_ham[word]

    if p_ham_given_message > p_spam_given_message:
        return 'ham'
    elif p_spam_given_message > p_ham_given_message:
        return 'spam'
    else:
        return 'needs human classification'
import re

def classify_test_set(message):
    '''
    message: a string
    '''

    # Use raw string for regular expression
    message = re.sub(r'\W', ' ', message)  # Removes punctuation
    message = message.lower().split()

    p_spam_given_message = p_spam
    p_ham_given_message = p_ham

    for word in message:
        if word in parameters_spam:
            p_spam_given_message *= parameters_spam[word]

        if word in parameters_ham:
            p_ham_given_message *= parameters_ham[word]

    if p_ham_given_message > p_spam_given_message:
        return 'ham'
    elif p_spam_given_message > p_ham_given_message:
        return 'spam'
    else:
        return 'needs human classification'
```

In [64]:
```
test_set['predicted'] = test_set['Message'].apply(classify_test_set)
test_set.head()
```

Out[64]:

	Category	Message	predicted
0	ham	Later i guess. I needa do mcat study too.	ham
1	ham	But i haf enuff space got like 4 mb...	ham
2	spam	Had your mobile 10 mths? Update to latest Oran...	spam
3	ham	All sounds good. Fingers . Makes it difficult ...	ham
4	ham	All done, all handed in. Don't know if mega sh...	ham

In [66]:

```python
correct = 0
total = test_set.shape[0]

for row in test_set.iterrows():
   row = row[1]
   if row['Category'] == row['predicted']:
      correct += 1

print('Correct:', correct)
print('Incorrect:', total - correct)
print('Accuracy:', correct/total)
```

```
Correct: 1098
Incorrect: 16
Accuracy: 0.9856373429084381
```

In [ ]: