In [4]:
```
import pandas as pd

sms_spam = pd.read_csv('SMSSpamCollection', sep='\t',
header=None, names=['Label', 'SMS'])

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

```
(5572, 2)
```

Out[4]:

	Label	SMS
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]:
```
sms_spam['Label'].value_counts(normalize=True)
```

Out[5]:
```
Label
ham     0.865937
spam    0.134063
Name: proportion, dtype: float64
```

In [6]:
```
# 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]:
```
training_set['Label'].value_counts(normalize=True)
```

Out[7]:
```
Label
ham     0.86541
spam    0.13459
Name: proportion, dtype: float64
```

In [8]:
```
test_set['Label'].value_counts(normalize=True)
```

Out[8]:
```
Label
ham     0.868043
spam    0.131957
Name: proportion, dtype: float64
```

In [9]:
```
# Before cleaning
training_set.head(3)
```

Out[9]:

	Label	SMS
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 [10]:
```
# After cleaning
training_set['SMS'] = training_set['SMS'].str.replace(
   '\W', ' ') # Removes punctuation
training_set['SMS'] = training_set['SMS'].str.lower()
training_set.head(3)
```

Out[10]:

	Label	SMS
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 [11]:
```
training_set['SMS'] = training_set['SMS'].str.split()

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

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

In [12]:
```
len(vocabulary)
```

Out[12]:
```
11860
```

In [13]:
```
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[13]:

	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 [18]:
```
word_counts_per_sms = {unique_word: [0] * len(training_set['SMS']) for unique_word

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

In [17]:
```
word_counts = pd.DataFrame(word_counts_per_sms)
word_counts.head()
```

Out[17]:

	out?	rental	x49.	txt>	helen,	out--if	dat..	both!	comes..	tahan	...	driving...	concern	exp
0	0	0	0	0	0	0	0	0	0	0	...	0	0	
1	0	0	0	0	0	0	0	0	0	0	...	0	0	
2	0	0	0	0	0	0	0	0	0	0	...	0	0	
3	0	0	0	0	0	0	0	0	0	0	...	0	0	
4	0	0	0	0	0	0	0	0	0	0	...	0	0	

5 rows × 11860 columns

In [16]:
```
training_set_clean = pd.concat([training_set, word_counts], axis=1)
training_set_clean.head()
```

Out[16]:

	Label	SMS	out?	rental	x49.	txt>	helen,	out--if	dat..	both!	...	driving...	concern	e
0	ham	[yep,, by, the, pretty, sculpture]	0	0	0	0	0	0	0	0	...	0	0	
1	ham	[yes,, princess., are, you, going, to, make, m...	0	0	0	0	0	0	0	0	...	0	0	
2	ham	[welp, apparently, he, retired]	0	0	0	0	0	0	0	0	...	0	0	
3	ham	[havent.]	0	0	0	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	0	...	0	0	

5 rows × 11862 columns

In [19]:
```
# Isolating spam and ham messages first
spam_messages = training_set_clean[training_set_clean['Label'] == 'spam']
ham_messages = training_set_clean[training_set_clean['Label'] == '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['SMS'].apply(len)
n_spam = n_words_per_spam_message.sum()
```

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

# N_Vocabulary
n_vocabulary = len(vocabulary)

# Laplace smoothing
alpha = 1
```

In [20]:
```
# Initiate parameters
parameters_spam = {unique_word:0 for unique_word in vocabulary}
parameters_ham = {unique_word:0 for unique_word in vocabulary}

# Calculate parameters
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 [21]:
```
import re

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

    message = re.sub('\W', ' ', message)
    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 proabilities, have a human classify this!')
```

In [22]:
```
classify('WINNER!! This is the secret code to unlock the money: C3421.')
```

```
P(Spam|message): 1.1680023632078457e-26
P(Ham|message): 6.088544142463393e-28
Label: Spam
```

In [23]:
```
classify("Sounds good, Tom, then see u there")
```

```
P(Spam|message): 2.234299283967944e-26
P(Ham|message): 8.376346103813855e-22
Label: Ham
```

In [24]:
```
def classify_test_set(message):
    '''
    message: a string
    '''

    message = re.sub('\W', ' ', message)
    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 [25]:
```
test_set['predicted'] = test_set['SMS'].apply(classify_test_set)
test_set.head()
```

Out[25]:

	Label	SMS	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 [26]:
```
correct = 0
total = test_set.shape[0]

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

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

```
Correct: 1090
Incorrect: 24
Accuracy: 0.9784560143626571
```

In [ ]:
```

```