```
In [1]: pwd

Out[1]: '/Users/rebirth/KEEP/621/LEARN/SPAM'
```

```
In [2]: rootdir = '/Users/rebirth/KEEP/621/LEARN/SPAM/'
```

```
In [3]: import nltk.classify.util
        from nltk.classify import NaiveBayesClassifier
        from nltk.corpus import movie_reviews
        from nltk.corpus import stopwords
        from nltk.tokenize import word_tokenize
        import os
        import random
```

```
In [4]: # Loop through all DIRs, sub-DIRs and files, print them all.
        # For files: print number(files)

        for directories, subdirs, files in os.walk(rootdir):
            print(directories, subdirs, len(files))
```

```
/Users/rebirth/KEEP/621/LEARN/SPAM/ ['enron1', 'enron6', 'enron5', '.ip
ynb_checkpoints', 'enron2', 'enron3', 'enron4'] 10
/Users/rebirth/KEEP/621/LEARN/SPAM/enron1 ['spam', 'ham'] 2
/Users/rebirth/KEEP/621/LEARN/SPAM/enron1/spam [] 1500
/Users/rebirth/KEEP/621/LEARN/SPAM/enron1/ham [] 3672
/Users/rebirth/KEEP/621/LEARN/SPAM/enron6 ['spam', 'ham'] 1
/Users/rebirth/KEEP/621/LEARN/SPAM/enron6/spam [] 4500
/Users/rebirth/KEEP/621/LEARN/SPAM/enron6/ham [] 1500
/Users/rebirth/KEEP/621/LEARN/SPAM/enron5 ['spam', 'ham'] 1
/Users/rebirth/KEEP/621/LEARN/SPAM/enron5/spam [] 3675
/Users/rebirth/KEEP/621/LEARN/SPAM/enron5/ham [] 1500
/Users/rebirth/KEEP/621/LEARN/SPAM/.ipynb_checkpoints [] 3
/Users/rebirth/KEEP/621/LEARN/SPAM/enron2 ['spam', 'ham'] 1
/Users/rebirth/KEEP/621/LEARN/SPAM/enron2/spam [] 1496
/Users/rebirth/KEEP/621/LEARN/SPAM/enron2/ham [] 4361
/Users/rebirth/KEEP/621/LEARN/SPAM/enron3 ['spam', 'ham'] 1
/Users/rebirth/KEEP/621/LEARN/SPAM/enron3/spam [] 1500
/Users/rebirth/KEEP/621/LEARN/SPAM/enron3/ham [] 4012
/Users/rebirth/KEEP/621/LEARN/SPAM/enron4 ['spam', 'ham'] 1
/Users/rebirth/KEEP/621/LEARN/SPAM/enron4/spam [] 4500
/Users/rebirth/KEEP/621/LEARN/SPAM/enron4/ham [] 1500
```

```
In [5]: print(os.path.split("/Users/rebirth/KEEP/621/LEARN/SPAM/enron1/ham"))
        print(os.path.split("/Users/rebirth/KEEP/621/LEARN/SPAM/enron1/ham")[0])
        print(os.path.split("/Users/rebirth/KEEP/621/LEARN/SPAM/enron1/ham")[1])
```

```
('/Users/rebirth/KEEP/621/LEARN/SPAM/enron1', 'ham')
/Users/rebirth/KEEP/621/LEARN/SPAM/enron1
ham
```

In [6]:
```
# Loop through 'ham' and 'spam' files, print them all.
# For files: print number(files)
for directories, subdirs, files in os.walk(rootdir):
    if (os.path.split(directories)[1] == 'ham'):
        print(directories, subdirs, len(files))

    if (os.path.split(directories)[1] == 'spam'):
        print(directories, subdirs, len(files))
```

```
/Users/rebirth/KEEP/621/LEARN/SPAM/enron1/spam [] 1500
/Users/rebirth/KEEP/621/LEARN/SPAM/enron1/ham [] 3672
/Users/rebirth/KEEP/621/LEARN/SPAM/enron6/spam [] 4500
/Users/rebirth/KEEP/621/LEARN/SPAM/enron6/ham [] 1500
/Users/rebirth/KEEP/621/LEARN/SPAM/enron5/spam [] 3675
/Users/rebirth/KEEP/621/LEARN/SPAM/enron5/ham [] 1500
/Users/rebirth/KEEP/621/LEARN/SPAM/enron2/spam [] 1496
/Users/rebirth/KEEP/621/LEARN/SPAM/enron2/ham [] 4361
/Users/rebirth/KEEP/621/LEARN/SPAM/enron3/spam [] 1500
/Users/rebirth/KEEP/621/LEARN/SPAM/enron3/ham [] 4012
/Users/rebirth/KEEP/621/LEARN/SPAM/enron4/spam [] 4500
/Users/rebirth/KEEP/621/LEARN/SPAM/enron4/ham [] 1500
```

In [7]:
```
# Loop through 'ham' and 'spam' files, read them and append them to ham/
spam lists.

ham_list=[]
spam_list=[]

for directories, subdirs, files in os.walk(rootdir):
    if (os.path.split(directories)[1] == 'ham'):
        for filename in files:
            with open(os.path.join(directories, filename), encoding="lat
in-1") as f:
                data = f.read()
                ham_list.append(data)


    if (os.path.split(directories)[1] == 'spam'):
        for filename in files:
            with open(os.path.join(directories, filename), encoding="lat
in-1") as f:
                data = f.read()
                ham_list.append(data)

print(ham_list[0])
print(spam_list[0])
```

```
Subject: what up , , your cam babe
what are you looking for ?
if your looking for a companion for friendship , love , a date , or jus
t good ole '
fashioned * * * * * * , then try our brand new site ; it was developed
and created
to help anyone find what they ' re looking for . a quick bio form and y
ou ' re
on the road to satisfaction in every sense of the word . . . . no matte
r what
that may be !
try it out and youll be amazed .
have a terrific time this evening
copy and pa ste the add . ress you see on the line below into your brow
ser to come to the site .
http : / / www . meganbang . biz / bld / acc /
no more plz
http : / / www . naturalgolden . com / retract /
counterattack aitken step preemptive shoehorn scaup . electrocardiograp
h movie honeycomb . monster war brandywine pietism byrne catatonia . en
comia lookup intervenor skeleton turn catfish .


---------------------------------------------------------------------------
IndexError                                Traceback (most recent call l
ast)
<ipython-input-7-36664e76601e> in <module>()
     19
     20 print(ham_list[0])
---> 21 print(spam_list[0])

IndexError: list index out of range
```

In [8]:

```python
# Function to return a dictionary of the form (Word1: True, Word2: True,
 Word3: True)

def create_word_features(words):
    my_dict = dict([(word, True) for word in words])
    return my_dict

create_word_features(["I", "came", "saw", "and", "conquered"])
```

Out[8]:

```
{'I': True, 'came': True, 'saw': True, 'and': True, 'conquered': True}
```

In [9]:
```
ham_list=[]
spam_list=[]

#Break sentences into words ising word_tokenize
#Use create_word_features()
for directories, subdirs, files in os.walk(rootdir):
    if (os.path.split(directories)[1]  == 'ham'):
        for filename in files:
            with open(os.path.join(directories, filename), encoding="lat
in-1") as f:
                data = f.read()

                # Data read is a single, big string; needs to be broken
 into words.
                words = word_tokenize(data)
                ham_list.append((create_word_features(words), "ham"))

    if (os.path.split(directories)[1]  == 'spam'):
        for filename in files:
            with open(os.path.join(directories, filename), encoding="lat
in-1") as f:
                data = f.read()

                # Data read is a single, big string; needs to be broken
 into words.
                words = word_tokenize(data)
                spam_list.append((create_word_features(words), "spam"))
print(ham_list[0])
print(spam_list[0])
```

```
({'Subject': True, ':': True, 'ena': True, 'sales': True, 'on': True,
'hpl': True, 'just': True, 'to': True, 'update': True, 'you': True, 'th
is': True, 'project': True, "'": True, 's': True, 'status': True, 'base
d': True, 'a': True, 'new': True, 'report': True, 'that': True, 'scot
t': True, 'mills': True, 'ran': True, 'for': True, 'me': True, 'from':
True, 'sitara': True, ',': True, 'i': True, 'have': True, 'come': True,
'up': True, 'with': True, 'the': True, 'following': True, 'counterparti
es': True, 'as': True, 'ones': True, 'which': True, 'is': True, 'sellin
g': True, 'gas': True, 'off': True, 'of': True, 'pipe': True, '.': Tru
e, 'altrade': True, 'transaction': True, 'l': True, 'c': True, 'gulf':
True, 'utilities': True, 'company': True, 'brazoria': True, 'city': Tru
e, 'panther': True, 'pipeline': True, 'inc': True, 'central': True, 'il
linois': True, 'light': True, 'praxair': True, 'power': True, 'and': Tr
ue, 'reliant': True, 'energy': True, '-': True, 'entex': True, 'ces': T
rue, 'equistar': True, 'chemicals': True, 'lp': True, 'hl': True, '&':
True, 'p': True, 'corpus': True, 'christi': True, 'marketing': True, 's
outhern': True, 'union': True, 'd': True, 'h': True, 'texas': True, 'fu
el': True, 'duke': True, 'field': True, 'services': True, 'txu': True,
'distribution': True, 'carbide': True, 'corporation': True, 'unit': Tru
e, 'transmission': True, 'since': True, 'm': True, 'not': True, 'sure':
True, 'exactly': True, 'what': True, 'gets': True, 'entered': True, 'in
to': True, 'pat': True, 'clynes': True, 'suggested': True, 'check': Tru
e, 'daren': True, 'farmer': True, 'make': True, 'missing': True, 'somet
hing': True, '(': True, 'did': True, 'below': True, ')': True, 'while':
True, 'am': True, 'waiting': True, 'response': True, 'him': True, '/':
True, 'or': True, 'mary': True, 'smith': True, 'will': True, 'begin': T
rue, 'gathering': True, 'contractual': True, 'volumes': True, 'under':
True, 'above': True, 'contracts': True, 'forwarded': True, 'by': True,
'cheryl': True, 'dudley': True, 'hou': True, 'ect': True, '05': True,
'10': True, '2000': True, '07': True, '56': True, 'king': True, '08': T
rue, '04': True, '11': True, 'pm': True, 'sent': True, 'j': True, '@':
True, 'cc': True, 'subject': True, 'working': True, 'brenda': True, 'he
rod': True, 'was': True, 'wondering': True, 'if': True, 'one': True, 'c
ould': True, 'tell': True, 'right': True, 'track': True, 'get': True,
'everything': True, 'she': True, 'looking': True, 'trying': True, 'draf
t': True, 'long': True, 'term': True, 'transport': True, 'storage': Tru
e, 'agreement': True, 'between': True, 'hplc': True, 'allow': True, 'mo
ve': True, 'their': True, 'markets': True, 'in': True, 'order': True,
'accomplish': True, 'needs': True, 'know': True, 'all': True, 'customer
s': True, 'doing': True, 'had': True, 'run': True, 'showing': True, 'bu
y': True, 'sell': True, 'activity': True, '7': True, '99': True, 'elimi
nate': True, 'buys': True, 'desk': True, 'deals': True, 'give': True,
'need': True, '?': True, 'are': True, 'there': True, 'done': True, 'wou
ldn': True, 't': True, 'show': True, 'someone': True, 'mentioned': Tru
e, 'about': True, 'where': True, 'transports': True, 'it': True, 'own':
True, 'behalf': True, 'then': True, 'sells': True, 'customer': True, 'a
t': True, 'same': True, 'spot': True, 'do': True, 'like': True, 'happe
n': True, 'would': True, 'they': True, 'anything': True, 'else': True,
'real': True, 'familiar': True, 'how': True, 'some': True, 'these': Tru
e, 'nowadays': True, 'so': True, 'very': True, 'receptive': True, 'an
y': True, 'ideas': True, 'suggestions': True, 'help': True, 'can': Tru
e, 'offer': True, '!': True, 'thanks': True, 'advance': True}, 'ham')
({'Subject': True, ':': True, 'what': True, 'up': True, ',': True, 'you
r': True, 'cam': True, 'babe': True, 'are': True, 'you': True, 'lookin
g': True, 'for': True, '?': True, 'if': True, 'a': True, 'companion': T
rue, 'friendship': True, 'love': True, 'date': True, 'or': True, 'jus
t': True, 'good': True, 'ole': True, "'": True, 'fashioned': True, '*':
```

```
True, 'then': True, 'try': True, 'our': True, 'brand': True, 'new': Tru
e, 'site': True, ';': True, 'it': True, 'was': True, 'developed': True,
'and': True, 'created': True, 'to': True, 'help': True, 'anyone': True,
'find': True, 'they': True, 're': True, '.': True, 'quick': True, 'bi
o': True, 'form': True, 'on': True, 'the': True, 'road': True, 'satisfa
ction': True, 'in': True, 'every': True, 'sense': True, 'of': True, 'wo
rd': True, 'no': True, 'matter': True, 'that': True, 'may': True, 'be':
True, '!': True, 'out': True, 'youll': True, 'amazed': True, 'have': Tr
ue, 'terrific': True, 'time': True, 'this': True, 'evening': True, 'cop
y': True, 'pa': True, 'ste': True, 'add': True, 'ress': True, 'see': Tr
ue, 'line': True, 'below': True, 'into': True, 'browser': True, 'come':
True, 'http': True, '/': True, 'www': True, 'meganbang': True, 'biz': T
rue, 'bld': True, 'acc': True, 'more': True, 'plz': True, 'naturalgolde
n': True, 'com': True, 'retract': True, 'counterattack': True, 'aitke
n': True, 'step': True, 'preemptive': True, 'shoehorn': True, 'scaup':
True, 'electrocardiograph': True, 'movie': True, 'honeycomb': True, 'mo
nster': True, 'war': True, 'brandywine': True, 'pietism': True, 'byrn
e': True, 'catatonia': True, 'encomia': True, 'lookup': True, 'interven
or': True, 'skeleton': True, 'turn': True, 'catfish': True}, 'spam')
```

In [10]:
```
combined_list = ham_list + spam_list
print(len(combined_list))
random.shuffle(combined_list)
```

```
33716
```

In [11]:
```
#Creating test and train section; 67%:training, 33%:test
training_part=int(len(combined_list)*0.67)
print(len(combined_list))
training_set = combined_list[:training_part]
test_set = combined_list[training_part:]
print(len(training_set))
print(len(test_set))
```

```
33716
22589
11127
```

In [12]:
```
#Creating Naïve Bayes filter
classifier = NaiveBayesClassifier.train(training_set)

#Finding accuracy using test-data
accuracy=nltk.classify.util.accuracy(classifier, test_set)

print("Accuracy is: ",accuracy*100)
```

```
Accuracy is:  98.59800485306013
```

```
In [13]: classifier.show_most_informative_features(20)
```

```
Most Informative Features
              scheduling = True              ham : spam   =    488.4 :
1.0
                     php = True             spam : ham    =    403.3 :
1.0
                     sex = True             spam : ham    =    317.5 :
1.0
                    meds = True             spam : ham    =    285.1 :
1.0
                     713 = True              ham : spam   =    258.9 :
1.0
                crenshaw = True              ham : spam   =    251.4 :
1.0
                     hpl = True              ham : spam   =    234.8 :
1.0
                   corel = True             spam : ham    =    214.5 :
1.0
                     eol = True              ham : spam   =    203.1 :
1.0
             medications = True             spam : ham    =    201.8 :
1.0
                     ect = True              ham : spam   =    200.3 :
1.0
                   daren = True              ham : spam   =    198.8 :
1.0
                  louise = True              ham : spam   =    193.4 :
1.0
              macromedia = True             spam : ham    =    186.5 :
1.0
                        = True             ham : spam   =    162.6 :
1.0
                    pill = True             spam : ham    =    162.4 :
1.0
                 parsing = True              ham : spam   =    141.9 :
1.0
                     853 = True              ham : spam   =    125.3 :
1.0
             derivatives = True              ham : spam   =    116.4 :
1.0
                    dose = True             spam : ham    =    115.4 :
1.0
```

In [15]:
```
#Classify as ham/spam
#Break into words using word_tokenize
#Create_word_features
#Use classify function

msg1='''Visit this link for premium membership'''

msg2='''Nik, this is Mr. Carmichael with REACH. I'm glad to let you know
 that your pay has been increased by 300,000%.'''

msg3='''ROC curve or Precision/Recall curve while varying threshold para
meters'''

msg4='''We need to finish both assignments by today.'''

msg5='''Click here to get laid'''

msg6='''Mr. walters informed me that he rec'd a call from our mktg. Plea
se look into this for me and see if you can provide me
an assessment. Thanks.'''

msg7='''U dun say so early hor... U c already then say...'''

msg8='''Run around the block next time.'''

msg9='''Go until jurong point, crazy.. Available only in bugis n great w
orld la e buffet... Cine there got amore wat...'''

msg10='''Ok lar... Joking wif u oni...'''

msg11='''Free entry in 2 a wkly comp to win FA Cup final tkts 21st May 2
005. Text FA to 87121 to receive entry question(std txt rate)T&C's apply
 08452810075over18's'''

msg12='''Nah I don't think he goes to usf, he lives around here thoug
h'''

msg13='''FreeMsg Hey there darling it's been 3 week's now and no word ba
ck! I'd like some fun you up for it still? Tb ok! XxX std chgs to send,
 £1.50 to rcv
'''
msg14='''Even my brother is not like to speak with me. They treat me lik
e aids patent'''

msg15='''WINNER!! As a valued network customer you have been selected to
 receivea £900 prize reward! To claim call 09061701461. Claim code KL34
1. Valid 12 hours only.'''

msg16='''Had your mobile 11 months or more? U R entitled to Update to th
e latest colour mobiles with camera for Free! Call The Mobile Update Co
 FREE on 08002986030'''

msg17='''I'm gonna be home soon and i don't want to talk about this stuf
f anymore tonight, k? I've cried enough today.'''

msg18='''SIX chances to win CASH! From 100 to 20,000 pounds txt> CSH11 a
```

```
nd send to 87575. Cost 150p/day, 6days, 16+ TsandCs apply Reply HL 4 inf
o'''

msg19='''URGENT! You have won a 1 week FREE membership in our £100,000 P
rize Jackpot! Txt the word: CLAIM to No: 81010 T&C www.dbuk.net LCCLTD P
OBOX 4403LDNW1A7RW18'''

msg20='''I've been searching for the right words to thank you for this b
reather. I promise i wont take your help for granted and will fulfil my
 promise. You have been wonderful and a blessing at all times.'''


msg21='''I HAVE A DATE ON SUNDAY WITH WILL!!'''

msg22='''XXXMobileMovieClub: To use your credit, click the WAP link in t
he next txt message or click here>> http://wap. xxxmobilemovieclub.com?n
=QJKGIGHJJGCBL'''

msg23='''Oh k...i'm watching here:)'''

msg24='''Eh u remember how 2 spell his name... Yes i did. He v naughty m
ake until i v wet.'''

msg25='''Fine if thats the way u feel. Thats the way its gota b'''

msg26='''England v Macedonia - dont miss the goals/team news. Txt ur nat
ional team to 87077 eg ENGLAND to 87077 Try:WALES, SCOTLAND 4txt/ú1.20 P
OBOXox36504W45WQ 16+'''

msg27='''Is that seriously how you spell his name?'''

msg28='''Thanks for your subscription to Ringtone UK your mobile will be
 charged £5/month Please confirm by replying YES or NO. If you reply NO
 you will not be charged
'''

msg29='''Yup... Ok i go home look at the timings then i msg ü again... X
uhui going to learn on 2nd may too but her lesson is at 8am'''

msg30='''Oops, I'll let you know when my roommate's done'''

msg31='''Urgent UR awarded a complimentary trip to EuroDisinc Trav, Aco&
Entry41 Or £1000. To claim txt DIS to 87121 18+6*£1.50(moreFrmMob. ShrAc
omOrSglSuplt)10, LS1 3AJ'''

msg32='''SMS. ac Sptv: The New Jersey Devils and the Detroit Red Wings p
lay Ice Hockey. Correct or Incorrect? End? Reply END SPTV'''

msg33='''K fyi x has a ride early tomorrow morning but he's crashing at
 our place tonight'''

msg34='''Its a part of checking IQ'''

msg35='''Please call our customer service representative on 0800 169 603
1 between 10am-9pm as you have WON a guaranteed £1000 cash or £5000 priz
e!'''
```

```
msg36='''07732584351 - Rodger Burns - MSG = We tried to call you re your
 reply to our sms for a free nokia mobile + free camcorder. Please call
 now 08000930705 for delivery'''

msg37='''A gram usually runs like  &lt;#&gt; , a half eighth is smarter
 though and gets you almost a whole second gram for  &lt;#&gt;'''

msg38='''i see. When we finish we have loads of loans to pay'''

msg39='''HEY GIRL. HOW R U? HOPE U R WELL ME AN DEL R BAK! AGAIN LONG TI
ME NO C! GIVE ME A CALL SUM TIME FROM LUCYxx'''

msg40='''Hello, my love. What are you doing? Did you get to that intervi
ew today? Are you you happy? Are you being a good boy? Do you think of m
e?Are you missing me ?'''
```

In [16]:
```
words = word_tokenize(msg1)
features=create_word_features(words)
print("Message 1 is: ",classifier.classify(features))
```

```
Message 1 is:  spam
```

In [17]:
```
words = word_tokenize(msg2)
features=create_word_features(words)
print("Message 2 is: ",classifier.classify(features))
```

```
Message 2 is:  ham
```

In [18]:
```
words = word_tokenize(msg3)
features=create_word_features(words)
print("Message 3 is: ",classifier.classify(features))
```

```
Message 3 is:  ham
```

In [19]:
```
words = word_tokenize(msg4)
features=create_word_features(words)
print("Message 4 is: ",classifier.classify(features))
```

```
Message 4 is:  ham
```

In [20]:
```
words = word_tokenize(msg5)
features=create_word_features(words)
print("Message 5 is: ",classifier.classify(features))
```

```
Message 5 is:  spam
```

In [21]:
```
words = word_tokenize(msg6)
features=create_word_features(words)
print("Message 6 is: ",classifier.classify(features))
```

```
Message 6 is:  ham
```

In [22]:
```
words = word_tokenize(msg7)
features=create_word_features(words)
print("Message 7 is: ",classifier.classify(features))
```

```
Message 7 is:  ham
```

In [23]:
```
words = word_tokenize(msg8)
features=create_word_features(words)
print("Message 8 is: ",classifier.classify(features))
```

```
Message 8 is:  ham
```

In [24]:
```
words = word_tokenize(msg9)
features=create_word_features(words)
print("Message 9 is: ",classifier.classify(features))
```

```
Message 9 is:  spam
```

In [25]:
```
words = word_tokenize(msg10)
features=create_word_features(words)
print("Message 10 is: ",classifier.classify(features))
```

```
Message 10 is:  spam
```

In [26]:
```
words = word_tokenize(msg11)
features=create_word_features(words)
print("Message 11 is: ",classifier.classify(features))
```

```
Message 11 is:  ham
```

In [27]:
```
words = word_tokenize(msg12)
features=create_word_features(words)
print("Message 12 is: ",classifier.classify(features))
```

```
Message 12 is:  ham
```

In [28]:
```
words = word_tokenize(msg13)
features=create_word_features(words)
print("Message 13 is: ",classifier.classify(features))
```

```
Message 13 is:  ham
```

In [29]:
```
words = word_tokenize(msg14)
features=create_word_features(words)
print("Message 14 is: ",classifier.classify(features))
```

```
Message 14 is:  spam
```

In [30]:
```
words = word_tokenize(msg15)
features=create_word_features(words)
print("Message 15 is: ",classifier.classify(features))
```

```
Message 15 is:  spam
```

In [31]:
```
words = word_tokenize(msg16)
features=create_word_features(words)
print("Message 16 is: ",classifier.classify(features))
```
```
Message 16 is:  spam
```

In [32]:
```
words = word_tokenize(msg17)
features=create_word_features(words)
print("Message 17 is: ",classifier.classify(features))
```
```
Message 17 is:  spam
```

In [33]:
```
words = word_tokenize(msg18)
features=create_word_features(words)
print("Message 18 is: ",classifier.classify(features))
```
```
Message 18 is:  ham
```

In [34]:
```
words = word_tokenize(msg19)
features=create_word_features(words)
print("Message 19 is: ",classifier.classify(features))
```
```
Message 19 is:  spam
```

In [35]:
```
words = word_tokenize(msg20)
features=create_word_features(words)
print("Message 20 is: ",classifier.classify(features))
```
```
Message 20 is:  spam
```

In [36]:
```
words = word_tokenize(msg21)
features=create_word_features(words)
print("Message 21 is: ",classifier.classify(features))
```
```
Message 21 is:  spam
```

In [37]:
```
words = word_tokenize(msg22)
features=create_word_features(words)
print("Message 22 is: ",classifier.classify(features))
```
```
Message 22 is:  ham
```

In [38]:
```
words = word_tokenize(msg23)
features=create_word_features(words)
print("Message 23 is: ",classifier.classify(features))
```
```
Message 23 is:  spam
```

In [39]:
```
words = word_tokenize(msg24)
features=create_word_features(words)
print("Message 24 is: ",classifier.classify(features))
```
```
Message 24 is:  spam
```

In [40]:
```
words = word_tokenize(msg25)
features=create_word_features(words)
print("Message 25 is: ",classifier.classify(features))
```
```
Message 25 is:  spam
```

In [41]:
```
words = word_tokenize(msg26)
features=create_word_features(words)
print("Message 26 is: ",classifier.classify(features))
```
```
Message 26 is:  spam
```

In [42]:
```
words = word_tokenize(msg27)
features=create_word_features(words)
print("Message 27 is: ",classifier.classify(features))
```
```
Message 27 is:  spam
```

In [43]:
```
words = word_tokenize(msg28)
features=create_word_features(words)
print("Message 28 is: ",classifier.classify(features))
```
```
Message 28 is:  spam
```

In [44]:
```
words = word_tokenize(msg29)
features=create_word_features(words)
print("Message 29 is: ",classifier.classify(features))
```
```
Message 29 is:  ham
```

In [45]:
```
words = word_tokenize(msg30)
features=create_word_features(words)
print("Message 30 is: ",classifier.classify(features))
```
```
Message 30 is:  ham
```

In [46]:
```
words = word_tokenize(msg31)
features=create_word_features(words)
print("Message 31 is: ",classifier.classify(features))
```
```
Message 31 is:  ham
```

In [47]:
```
words = word_tokenize(msg32)
features=create_word_features(words)
print("Message 32 is: ",classifier.classify(features))
```
```
Message 32 is:  spam
```

In [48]:
```
words = word_tokenize(msg33)
features=create_word_features(words)
print("Message 33 is: ",classifier.classify(features))
```
```
Message 33 is:  ham
```

In [49]:
```
words = word_tokenize(msg34)
features=create_word_features(words)
print("Message 34 is: ",classifier.classify(features))
```

```
Message 34 is:  ham
```

In [50]:
```
words = word_tokenize(msg35)
features=create_word_features(words)
print("Message 35 is: ",classifier.classify(features))
```

```
Message 35 is:  spam
```

In [51]:
```
words = word_tokenize(msg36)
features=create_word_features(words)
print("Message 36 is: ",classifier.classify(features))
```

```
Message 36 is:  spam
```

In [52]:
```
words = word_tokenize(msg37)
features=create_word_features(words)
print("Message 37 is: ",classifier.classify(features))
```

```
Message 37 is:  ham
```

In [53]:
```
words = word_tokenize(msg38)
features=create_word_features(words)
print("Message 38 is: ",classifier.classify(features))
```

```
Message 38 is:  ham
```

In [54]:
```
words = word_tokenize(msg39)
features=create_word_features(words)
print("Message 39 is: ",classifier.classify(features))
```

```
Message 39 is:  spam
```

In [55]:
```
words = word_tokenize(msg40)
features=create_word_features(words)
print("Message 40 is: ",classifier.classify(features))
```

```
Message 40 is:  ham
```