In [1]:
```
import numpy as np
import pprint as pr
import math
```

In [3]:
```
def loadDataSet():
    #EVERY LIST IS A SENTENCE
    postingList=[
        ['my', 'dog', 'has', 'flea','problems', 'help', 'please'],
        ['maybe', 'not', 'take', 'him', 'to', 'dog', 'park', 'stupid'],
        ['my', 'dalmation', 'is', 'so', 'cute','I', 'love', 'him'],
        ['stop', 'posting', 'stupid', 'worthless', 'garbage'],
        ['mr', 'licks', 'ate', 'my', 'steak', 'how','to', 'stop', 'him'],
        ['quit', 'buying', 'worthless', 'dog', 'food', 'stupid']
    ]
    classVec = [0,1,0,1,0,1] #1 is abusive, 0 not
    return postingList,classVec
```

In [4]:
```
dataSet , labels = loadDataSet()
```

In [5]:
```
def createVocabList(dataSet1):
    ''' RETURNS A UNIQUE LIST OF WORDS IN THE VOCABULARY'''
    vocabSet = set([]) # EMPTY SET
    for document in dataSet1:
        # 'UNION' SET OPERATION - |
        vocabSet = vocabSet | set(document)
    return list(vocabSet)
```

In [9]:
```
myVocabList = createVocabList(dataSet)
myVocabList.sort()
```

In [10]:
```
print(myVocabList)
```

```
['I', 'ate', 'buying', 'cute', 'dalmation', 'dog', 'flea', 'food', 'garbage',
'has', 'help', 'him', 'how', 'is', 'licks', 'love', 'maybe', 'mr', 'my', 'no
t', 'park', 'please', 'posting', 'problems', 'quit', 'so', 'steak', 'stop',
'stupid', 'take', 'to', 'worthless']
```

In [11]:
```
def setOfWords2Vec(vocabList, inputSet):
    '''
    vocabList is the list of unique words in vocabulary
    inputSet is the list of words to be tested (A line or doc)
    '''
    returnVec = [0] * len(vocabList)  # [0,0,0,.....n]
    for word in inputSet:
        if word in vocabList:
            returnVec[vocabList.index(word) ] = 1  # WORD PRESENT
        else:
            print("the word: %s is not in my Vocabulary!" %word )
    return returnVec
```

In [16]:
```
#EX: LETS TRY OUR FUNC ON first line of dataset
vocVector = setOfWords2Vec( myVocabList , dataSet[0] )
print(myVocabList,dataSet[0],vocVector,sep='\n\n')
```

```
['I', 'ate', 'buying', 'cute', 'dalmation', 'dog', 'flea', 'food', 'garbage',
'has', 'help', 'him', 'how', 'is', 'licks', 'love', 'maybe', 'mr', 'my', 'no
t', 'park', 'please', 'posting', 'problems', 'quit', 'so', 'steak', 'stop',
'stupid', 'take', 'to', 'worthless']

['my', 'dog', 'has', 'flea', 'problems', 'help', 'please']

[0, 0, 0, 0, 0, 1, 1, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 1, 0,
0, 0, 0, 0, 0, 0, 0]
```

In [35]:
```
def trainNB0(trainMatrix,trainCategory):
    '''
    Function train Naive Bayes
    INPUTS:
     1) trainMatrix   - dataSet which is a list of lists.
        Innerlist is a vocabulary vector which has 0 or 1 for corresponding
        words in the vocabulary list
     2) trainCategory - labels for the sentences . (abusive - 1 or not abusive
-0)

    OUTPUTS:
    probability vectors p0Vect and p1Vect and value pAbusive
    p0Vect is a vector of probability of occurences of word given that the sen
tence is non-abusive
    p1Vect is a vector of probability of occurences of word given that sentenc
e is non-abusive
    '''
    numTrainDocs = len(trainMatrix)   #no of sentences
    numWords     = len(trainMatrix[0]) # no of words in vocabulary list

    # sum(trainCategory) will return the no of ones in the labels i.e the no o
f
    # abusive documents/lines in the matrix.
    pAbusive = sum(trainCategory)/float(numTrainDocs)

    p0Num = np.ones(numWords)
    p1Num = np.ones(numWords) #THESE ARE VECTORS!!

    '''
    We set intial count to all 1's and not 0's bcoz if we keep all 0's then if
the probability
    for even a single word is 0 then when we multiply probabilities for obtain
ing the
    probability the answer will become 0 even if there is one 0 in probability
vector
    '''
    p0Denom = 2.0   #THE DENOMINATOR FOR BOTH VECTORS ARE SET TO 2.0
    p1Denom = 2.0

    for i in range(numTrainDocs):
        if trainCategory[i] == 1: #line or doc is abusive

            # add the entire corresponding vocabulary list vector to p1Num
            p1Num += trainMatrix[i]
            p1Denom += sum(trainMatrix[i]) # add the no of words present in li
ne to the den

        else:            #line or doc is non-abusive
            # add the entire corresponding vocabulary list to p0Num
            p0Num += trainMatrix[i]
            p0Denom += sum(trainMatrix[i])
    '''
    at the end of the loop ,
    p0Denom will be the no. of words belonging to data pieces in the training
 set having non-abusive label
    p1Denom will be the no. of words belonging to data pieces in the training
```

```
 set having abusive label
    '''
    p1Vect = np.log(p1Num/p1Denom)
    p0Vect = np.log(p0Num/p0Denom)

    #NOW WE HAVE THE probability vectors p1Vect and p0Vect

    return p0Vect,p1Vect,pAbusive
```

In [36]:
```
#CREATE THE TRAINING MATRIX
trainMat = []
for line in dataSet:
    trainMat.append( setOfWords2Vec(myVocabList , line) )
```

In [43]:
```
#WHAT DOES TRAINMAT LOOK LIKE?
for vocabList1 in trainMat:
    print(vocabList1,end='\n\n')
```

```
[0, 0, 0, 0, 0, 1, 1, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 1, 0,
0, 0, 0, 0, 0, 0, 0]

[0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 1, 1, 0, 0, 0, 0,
0, 0, 0, 1, 1, 1, 0]

[1, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0,
1, 0, 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, 0, 1, 0, 0,
0, 0, 1, 1, 0, 0, 1]

[0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 1, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0,
0, 1, 1, 0, 0, 1, 0]

[0, 0, 1, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1,
0, 0, 0, 1, 0, 0, 1]
```

In [38]:
```
def classifyNB(vec2Classify, p0Vec, p1Vec, pClass1):
    '''
    THE FUNCTION WHICH DOES THE ACTUAL CLASSIFICATION BASED ON THE probability
    vectors p0Vec and p1Vec genereted by trainNB0 and comparing the final

    vec2Classify is a vocabulary list format having 0 or 1 indicating presence
of words.
    pClass1 is the probability of the vector being abusive
    '''
    p1 = sum(vec2Classify * p1Vec) + np.log(pClass1)
    p0 = sum(vec2Classify * p0Vec) + np.log(1.0 - pClass1)
    if p1 > p0:
        return 1
    else:
        return 0
```

In [39]:
```
p0Vect,p1Vect,probAbusive =  trainNB0(trainMat , labels)
print( p0Vect , p1Vect ,probAbusive,sep='\n\n')
```

```
[-2.56494936 -2.56494936 -3.25809654 -2.56494936 -2.56494936 -2.56494936
 -2.56494936 -3.25809654 -3.25809654 -2.56494936 -2.56494936 -2.15948425
 -2.56494936 -2.56494936 -2.56494936 -2.56494936 -3.25809654 -2.56494936
 -1.87180218 -3.25809654 -3.25809654 -2.56494936 -3.25809654 -2.56494936
 -3.25809654 -2.56494936 -2.56494936 -2.56494936 -3.25809654 -3.25809654
 -2.56494936 -3.25809654]

[-3.04452244 -3.04452244 -2.35137526 -3.04452244 -3.04452244 -1.94591015
 -3.04452244 -2.35137526 -2.35137526 -3.04452244 -3.04452244 -2.35137526
 -3.04452244 -3.04452244 -3.04452244 -3.04452244 -2.35137526 -3.04452244
 -3.04452244 -2.35137526 -2.35137526 -3.04452244 -2.35137526 -3.04452244
 -2.35137526 -3.04452244 -3.04452244 -2.35137526 -1.65822808 -2.35137526
 -2.35137526 -1.94591015]

0.5
```

In [40]:
```
#TESTING SAMPLE1
testEntry = ['love', 'my', 'dalmation']
thisDoc = np.array(setOfWords2Vec(myVocabList, testEntry))
print(testEntry,'classified as: ',classifyNB(thisDoc,p0Vect,p1Vect,probAbusive
) )
```

```
['love', 'my', 'dalmation'] classified as:  0
```

In [42]:
```
#TESTING SAMPLE2
testEntry = ['stupid', 'garbage']
thisDoc = np.array(setOfWords2Vec(myVocabList, testEntry))
print(testEntry,'classified as: ',classifyNB(thisDoc,p0Vect,p1Vect,probAbusive
) )
```

```
['stupid', 'garbage'] classified as:  1
```

In [ ]:
```

```