3a) I construct a naive Bayes classifier to filter spam emails and report the error for a number of training sets. My code is below:

```
import numpy as np
import heapq

probwordgivspam, probwordgivgood, spamsum, goodsum, logprobs = [0]*1448, [0]*1448, [0]*1448,
[0]*1448, [0]*1448
numspam = 0
numgood = 0

with open('SPARSE.TRAIN.1400') as f:
#get labels, calculate frequency sums
 for i, line in enumerate(f):
  label = line.split('  ')[0]
  text = line.split('  ')[1]
  for cell in text.strip().split(' '):
   word, freq = cell.split(':')
   if int(label) == 1:
    spamsum[int(word)-1] += int(freq)
    numspam += 1
   if int(label) == -1:
    goodsum[int(word)-1] += int(freq)
    numgood += 1

 #get probability of good & spam
probspam = numspam / (numspam + numgood)
probgood = numgood / (numspam + numgood)

 #use frequency sums to get conditionals
for i in range(0, 1448):
 probwordgivspam[i] = (1 + spamsum[i])/(len(spamsum)+sum(spamsum))
 probwordgivgood[i] = (1 + goodsum[i])/(len(goodsum)+sum(goodsum))
 logprobs[i] = np.log(probwordgivspam[i]/probwordgivgood[i])

with open('SPARSE.TEST') as g:
 numdocs = 0
 misclass = 0
#get labels
 for i, line in enumerate(g):
  numdocs += 1
  guesslabel = 0
  logpost = np.log(probspam/probgood)
  testwords = []
  testlabel = line.split('  ')[0]
  testtext = line.split('  ')[1]
  for cell in testtext.strip().split(' '):
   testword, testfreq = cell.split(':')
   testwords.append(int(testword)-1)
  for test in testwords:
   logpost += np.log(probwordgivspam[test]/probwordgivgood[test])
  if (logpost > 0):
   guesslabel = 1
  elif (logpost < 0):
   guesslabel = -1
  if (int(testlabel) != guesslabel):
   misclass += 1

error = (misclass/numdocs)*100
print("error is")
print(error)
print("spammiest words are at")

spammiest = heapq.nlargest(5, range(len(logprobs)), logprobs.__getitem__)
print(spammiest + np.ones(len(spammiest)))
```

Training on the SPARSE.TRAIN.50 sample gives an error of 1.875%.
Training on the SPARSE.TRAIN.100 sample gives an error of 1.375%.
Training on the SPARSE.TRAIN.200 sample gives an error of 1.125%.
Training on the SPARSE.TRAIN.400 sample gives an error of 1.25%.
Training on the SPARSE.TRAIN.800 sample gives an error of 1.375%.
Training on the SPARSE.TRAIN.1400 sample gives an error of 1.25%.

b) Using the probability measures calculated, we find the words most indicative of spam. Training on the SPARSE.TRAIN.1400 dataset, we find that the ‘spammiest’ words are ‘httpaddr’, ‘spam’, ‘unsubscrib’, ‘ebai’, and ‘diploma’.