In [23]:
```
import os
import math
import numpy as np
```

Inputs

In [24]:
```
path_train = "assignment3_train\\train"
path_test = "assignment3_test\\test"
itr = 500
lam = 0.01
theta = 0.01
```

In [25]:
```
sizett = 0
size_spam = 0
size_ham = 0
x = os.listdir(path_train)
spamwc={}
hamwc = {}
totalwc = {}
for i in x:
    y = os.listdir(path_train+"\\" + i)
    if i=="spam":
        for j in y:
            sizett += 1
            size_spam += 1
            f = path_train+"\\"+ i + "\\" + j
            file=open(f,"r", errors = 'ignore')
            for word in file.read().split():
                if word not in spamwc and word.isalpha():
                    spamwc[word] = 1
                    totalwc[word] = 1
                elif word.isalpha():
                    spamwc[word] += 1
                    totalwc[word] += 1
    else:
        for j in y:
            sizett += 1
            size_ham += 1
            f = path_train+"\\"+ i + "\\" + j
            file=open(f,"r", errors = 'ignore')
            for word in file.read().split():
                if word not in hamwc and word.isalpha():
                    hamwc[word] = 1
                    totalwc[word] = 1
                elif word.isalpha():
                    hamwc[word] += 1
                    totalwc[word] += 1
print("Total Word Count:",len(totalwc))
```

```
Total Word Count: 9186
```

Naive Bayes

In [26]:
```
totalw_s = sum(spamwc.values())
totalw_h = sum(hamwc.values())
novoc = len(totalwc)
cs = 0
ch = 0
cst = 0
cht = 0
size_test = 0
# Naive Bayes
for i in x:
    y = os.listdir(path_test+"\\"+ i)
    for j in y:
        test_sh = {}
        size_test += 1
        f = path_test+"\\"+ i + "\\" + j
        file=open(f,"r", errors = 'ignore')
        for word in file.read().split():
            if word not in test_sh and word.isalpha():
                test_sh[word] = 1
            elif word.isalpha():
                test_sh[word] += 1
        prob_s = math.log(size_spam/sizett)
        prob_h = math.log(size_ham/sizett)
        for k in test_sh:
            if spamwc.get(k) != None:
                prob_s = prob_s + math.log((spamwc.get(k)+1)/((totalw_s)+(novoc)))
            else:
                prob_s = prob_s + math.log((1)/((totalw_s)+(novoc)))
            if hamwc.get(k) != None:
                prob_h = prob_h + math.log((hamwc.get(k)+1)/((totalw_h)+(novoc)))
            else:
                prob_h = prob_h + math.log((1)/((totalw_h)+(novoc)))

            if prob_s > prob_h:
                cs = cs + 1
                if i=="spam":
                    cst = cst + 1
            elif prob_h > prob_s:
                ch = ch + 1
                if i=="ham":
                    cht = cht + 1

print("Accuracy",(cst+cht)/(cs+ch))
```

```
Accuracy 0.9219214600635702
```

Logistic Regression

In [27]:
```
ltotalwc = list(totalwc.keys())
mat = np.zeros((sizett,len(ltotalwc)+1))
ind = 0
for i in x:
    y = os.listdir(path_train+"\\"+ i)
    for j in y:
        logwc = {}
        f = path_train+"\\"+ i + "\\" + j
        file=open(f,"r", errors = 'ignore')
        for word in file.read().split():
            if word not in logwc and word.isalpha():
                logwc[word] = 1
            elif word.isalpha():
                logwc[word] += 1
        for k in logwc:
            mat[ind][ltotalwc.index(k)] = logwc[k]
        if i=="spam":
            mat[ind][len(ltotalwc)] = 1
        ind = ind + 1
```

In [28]:

```
def prob(w,x):
    s = 0
    for i in range(len(x)):
        s = s + (w[i]*x[i])
    try:
        p = math.exp(w[0]+s)/(1 + math.exp(w[0]+s))
    except:
        p = 1
    return p
```

In [29]:

```
w_new = np.ones(len(totalwc)+1)
w = np.ones(len(totalwc)+1)
probab = np.ones(mat.shape[0])
for k in range(itr):
    w = w_new.copy()
    w_new = np.ones(len(totalwc)+1)
    for l in range(mat.shape[0]):
        probab[l] = prob(w,mat[l])
    for i in range(len(w)):
        temp = 0
        for j in range(mat.shape[0]):
            temp = temp + mat[j][i]*((mat[j][mat.shape[1]-1])-probab[j])
        w_new[i] = w[i]+ (lam * temp) - (lam*theta*w[i])
```

In [30]:

```
mat_test = np.zeros((size_test,len(ltotalwc)+1))
ind = 0
for i in x:
    y = os.listdir(path_test+"\\"+ i)
    for j in y:
        logwc = {}
        f = path_test+"\\"+ i + "\\" + j
        file=open(f,"r", errors = 'ignore')
        for word in file.read().split():
            if word not in logwc and word.isalpha():
                logwc[word] = 1
            elif word.isalpha():
                logwc[word] += 1
        for k in logwc:
            if k in ltotalwc:
                mat_test[ind][ltotalwc.index(k)] = logwc[k]
        if i=="spam":
            mat_test[ind][len(ltotalwc)] = 1
        ind = ind + 1
```

In [31]:

```
th = 0
```

```
ts = 0
tt = 0
for i in range(mat_test.shape[0]):
    s = 0
    for j in range(mat_test.shape[1]-1):
        s = s + (w_new[j]*mat_test[i][j])
    s = s + w[0]
    tt += 1
    if mat_test[i][len(ltotalwc)]==1 and s>0:
        ts += 1
    elif mat_test[i][len(ltotalwc)]==0 and s<0:
        th += 1
print("Accuracy:",(ts+th)/tt)
```

```
Accuracy: 0.8744769874476988
```

After Removing Stopwords

In [32]:

```
stopWords = ["a", "about", "above", "after", "again", "against", "all", "am", "an", "an
             "any", "are", "aren't", "as", "at", "be", "because", "been", "before", "be
             "between", "both", "but", "by", "can't", "cannot", "could", "couldn't", "d
             "do", "does", "doesn't", "doing", "don't", "down", "during", "each", "few"
             "further", "had", "hadn't", "has", "hasn't", "have", "haven't", "having",
             "he'll", "he's", "her", "here", "here's", "hers", "herself", "him", "himse
             "how's", "i", "i'd", "i'll", "i'm", "i've", "if", "in", "into", "is", "isn
             "itself", "let's", "me", "more", "most", "mustn't", "my", "myself", "no",
             "off", "on", "once", "only", "or", "other", "ought", "our", "ours", "ourse
             "own", "same", "shan't", "she", "she'd", "she'll", "she's", "should", "sho
             "such", "than", "that", "that's", "the", "their", "theirs", "them", "thems
             "there's", "these", "they", "they'd", "they'll", "they're", "they've", "th
             "to", "too", "under", "until", "up", "very", "was", "wasn't", "we", "we'd"
             "were", "weren't", "what", "what's", "when", "when's", "where", "where's",
             "who's", "whom", "why", "why's", "with", "won't", "would", "wouldn't", "yo
             "you're", "you've", "your", "yours", "yourself", "yourselves"]
```

In [39]:

```
x = os.listdir(path_train)
spamwc={}
hamwc = {}
totalwc = {}
for i in x:
    y = os.listdir(path_train+"\\"+ i)
    if i=="spam":
        for j in y:
            f = path_train+"\\"+ i + "\\" + j
            file=open(f,"r", errors = 'ignore')
            for word in file.read().split():
                if word not in stopWords:
                    if word not in spamwc and word.isalpha():
                        spamwc[word] = 1
                        totalwc[word] = 1
                    elif word.isalpha():
                        spamwc[word] += 1
                        totalwc[word] += 1
    else:
        for j in y:
            f = path_train+"\\"+ i + "\\" + j
            file=open(f,"r", errors = 'ignore')
            for word in file.read().split():
                if word not in stopWords:
                    if word not in hamwc and word.isalpha():
                        hamwc[word] = 1
                        totalwc[word] = 1
                    elif word.isalpha():
                        hamwc[word] += 1
                        totalwc[word] += 1

print("Total Word Count:",len(totalwc))
```

```
Total Word Count: 9068
```

Naive Bayes

In [34]:

```
totalw_s = sum(spamwc.values())
totalw_h = sum(hamwc.values())
novoc = len(totalwc)
cs = 0
ch = 0
cst = 0
cht = 0
for i in x:
    y = os.listdir(path_test+"\\"+ i)
    for j in y:
        test_sh = {}
        f = path_test+"\\"+ i + "\\" + j
        file=open(f,"r", errors = 'ignore')
        for word in file.read().split():
            if word not in stopWords:
                if word not in test_sh and word.isalpha():
                    test_sh[word] = 1
                elif word.isalpha():
                    test_sh[word] += 1
        prob_s = math.log(size_spam/sizett)
        prob_h = math.log(size_ham/sizett)
        # print(prob_s, prob_h)
        for k in test_sh:
            if spamwc.get(k) != None:
                prob_s = prob_s + math.log((spamwc.get(k)+1)/((totalw_s)+(novoc)))
            else:
                prob_s = prob_s + math.log((1)/((totalw_s)+(novoc)))
            if hamwc.get(k) != None:
                prob_h = prob_h + math.log((hamwc.get(k)+1)/((totalw_h)+(novoc)))
            else:
                prob_h = prob_h + math.log((1)/((totalw_h)+(novoc)))

            if prob_s > prob_h:
                cs = cs + 1
                if i=="spam":
                    cst = cst + 1
            elif prob_h > prob_s:
                ch = ch + 1
                if i=="ham":
                    cht = cht + 1

print("Accuracy",(cst+cht)/(cs+ch))
```

```
Accuracy 0.9231868643222761
```

Logistic Regression

In [35]:

```
ltotalwc = list(totalwc.keys())
mat = np.zeros((sizett,len(ltotalwc)+1))
ind = 0
for i in x:
    y = os.listdir(path_train+"\\"+ i)
    for j in y:
        logwc = {}
        f = path_train+"\\"+ i + "\\" + j
        file=open(f,"r", errors = 'ignore')
        for word in file.read().split():
            if word not in stopWords:
                if word not in logwc and word.isalpha():
                    logwc[word] = 1
                elif word.isalpha():
                    logwc[word] += 1
        for k in logwc:
            mat[ind][ltotalwc.index(k)] = logwc[k]
        if i=="spam":
            mat[ind][len(ltotalwc)] = 1
        ind = ind + 1
```

In [36]:

```
w_new = np.ones(len(totalwc)+1)
w = np.ones(len(totalwc)+1)
for k in range(itr):
    w = w_new.copy()
    w_new = np.ones(len(totalwc)+1)
    for l in range(mat.shape[0]):
        probab[l] = prob(w,mat[l])
    for i in range(len(w)):
        temp = 0
        for j in range(mat.shape[0]):
            temp = temp + mat[j][i]*((mat[j][mat.shape[1]-1])-probab[j])
        w_new[i] = w[i]+ (lam * temp) - (lam*theta*w[i])
```

In [37]:

```
mat_test = np.zeros((size_test,len(ltotalwc)+1))
ind = 0
for i in x:
    y = os.listdir(path_test+"\\"+ i)
    for j in y:
        logwc = {}
        f = path_test+"\\"+ i + "\\" + j
        file=open(f,"r", errors = 'ignore')
        for word in file.read().split():
            if word not in stopWords:
                if word not in logwc and word.isalpha():
                    logwc[word] = 1
                elif word.isalpha():
                    logwc[word] += 1
        for k in logwc:
            if k in ltotalwc:
                mat_test[ind][ltotalwc.index(k)] = logwc[k]
        if i=="spam":
            mat_test[ind][len(ltotalwc)] = 1
        ind = ind + 1
```

In [38]:

```
th = 0
ts = 0
tt = 0
for i in range(mat_test.shape[0]):
    s = 0
    for j in range(mat_test.shape[1]-1):
        s = s + (w_new[j]*mat_test[i][j])
    s = s + w[0]
    tt += 1
    if mat_test[i][len(ltotalwc)]==1 and s>0:
        ts += 1
    elif mat_test[i][len(ltotalwc)]==0 and s<0:
        th += 1
print("Accuracy:",(ts+th)/tt)
```

```
Accuracy: 0.8640167364016736
```