Important Note for question1 !

- Please **do not** change the default variable names in this problem, as we will use them in different parts.
- The default variables are initially set to "None".
- You only need to modify code in the "TODO" part. We added a lot of "assertions" to check your code. **Do not** modify them.

In [54]:

```
# load packages
import numpy as np
import pandas as pd
import time
from sklearn.naive_bayes import GaussianNB
```

P1. Load data and plot

TODO

- Load train and test data, and split them into inputs(trainX, testX) and labels(trainY, testY)

In [55]:

```
# Use pandas to load q1_train.csv and q1_test.csv
# Each data point has 200 features(X), followed by 1 label(Y)

#### TODO ####
train=pd.read_csv("C:/Radhika/CMU/SEM1/ML_AI/HW3/24787-hw3-handout/24787-hw3-handout/q1_train.csv")
test=pd.read_csv("C:/Radhika/CMU/SEM1/ML_AI/HW3/24787-hw3-handout/24787-hw3-handout/q1_test.csv")
trainX = train.iloc[:len(train),1:201]
trainY = train.iloc[:len(train),201]
testX = test.iloc[:len(test),1:201]
testY = test.iloc[:len(test),201]
##############

assert(len(trainX.shape) == 2)
assert(len(trainY.shape) == 1)
assert(trainX.shape[1] == 200)
```

P2. Write your Gaussian NB solver

TODO

- Finish the myNBSolver() function.
  - Compute P(y == 0) and P(y == 1), saved in "py0" and "py1"
  - Compute mean/variance of trainX for both y = 0 and y = 1, saved in "mean0", "var0", "mean1" and "var1"
    - Each of them should have shape (N_train, M), where N_train is number of train samples and M is number of features.
  - Compute P(xi | y == 0) and P(xi | y == 1), compare and save **binary** prediction in "train_pred" and "test_pred"
  - Compute train accuracy and test accuracy, saved in "train_acc" and "test_acc".
  - Return train accuracy and test accuracy.

In [56]:

```
def myNBSolver(trainX, trainY, testX, testY):

    N_train = trainX.shape[0]
    N_test = testX.shape[0]
    M = trainX.shape[1]

    #### TODO ####
    # Compute P(y == 0) and P(y == 1)
    NTR1=np.count_nonzero(trainY)
    NTR0=len(trainY)-NTR1

    py0 =NTR0/len(trainY)
    py1 = NTR1/len(trainY)

    ##############
    print("Total probablity is %.2f. Should be equal to 1." %(py0 + py1))

    #### TODO ####
    # Compute mean/var for each label
    trn=np.column_stack((trainX,trainY))
    trn_0 = trn[trn[:,200] == 0]
    trn_1 = trn[trn[:,200] == 1]
    trn_0x=trn_0[:,0:200]
    trn_1x=trn_1[:,0:200]
    mean0 = np.mean(trn_0x, axis = 0)
    mean1 = np.mean(trn_1x, axis = 0)
    var0 = np.var(trn_0x, axis = 0)
    var1 = np.var(trn_1x, axis = 0)

    ##############
    assert(mean0.shape[0] == M)

    #### TODO ####
    # Compute P(xi|y == 0) and P(xi|y == 1), compare and make prediction
    # This part may spend 5 - 10 minutes or even more if you use for loop, so feel free to
    # print something (like step number) to check the progress
    px_trt_y0=np.prod((1/np.sqrt(2*np.pi*var0))*np.exp(-(trainX-mean0)**2/(2*var0)),axis=1)*py0
    px_trt_y1=np.prod((1/np.sqrt(2*np.pi*var1))*np.exp(-(trainX-mean1)**2/(2*var1)),axis=1)*py1
    train_pred = []
    for i in range(len(trainY)):
        if(px_trt_y0[i]>=px_trt_y1[i]):
            train_pred.append(0)
        else:
            train_pred.append(1)
    px_tst_y0=np.prod((1/np.sqrt(2*np.pi*var0))*np.exp(-(testX-mean0)**2/(2*var0)),axis=1)*py0
    px_tst_y1=np.prod((1/np.sqrt(2*np.pi*var1))*np.exp(-(testX-mean1)**2/(2*var1)),axis=1)*py1
    test_pred=[]
    for j in range(len(testY)):
        if(px_tst_y0[j]>=px_tst_y1[j]):
            test_pred.append(0)
        else:
            test_pred.append(1)

    ##############
    assert(train_pred[0] == 0 or train_pred[0] == 1)
    assert(test_pred[0] == 0 or test_pred[0] == 1)

    #### TODO ####
    # Compute train accuracy and test accuracy
    correct_y0=0
    for k in range(len(trainY)):
        if (train_pred[k]==trainY[k]):
            correct_y0 +=1
    correct_y1=0
    for m in range(len(testY)):
        if (test_pred[m]==testY[m]):
            correct_y1 +=1

    train_acc = correct_y0/len(trainY)
    test_acc = correct_y1/len(testY)

    ##############

    return train_acc, test_acc
```

In [57]:

```
# driver to test your NB solver
train_acc, test_acc = myNBSolver(trainX, trainY, testX, testY)
print("Train accuracy is %.2f" %(train_acc * 100))
print("Test accuracy is %.2f" %(test_acc * 100))
```

```
Total probablity is 1.00. Should be equal to 1.
Train accuracy is 92.22
Test accuracy is 92.05
```

P3. Test your result using sklearn

TODO

- Finish the skNBSolver() function.
  - fit model, make prediction and return accuracy for train and test sets.

In [58]:

```
def skNBSolver(trainX, trainY, testX, testY):

    #### TODO ####
    # fit model
    # make prediction
    # compute accuracy
    NB=GaussianNB()
    NB.fit(trainX,trainY)
    sk_train_acc = NB.score(trainX,trainY)
    sk_test_acc = NB.score(testX,testY)

    ##############
    return sk_train_acc, sk_test_acc
```

In [59]:

```
# driver to test skNBSolver
sk_train_acc, sk_test_acc = skNBSolver(trainX, trainY, testX, testY)
print("Train accuracy is %.2f" %(sk_train_acc * 100))
print("Test accuracy is %.2f" %(sk_test_acc * 100))
```

```
Train accuracy is 92.22
Test accuracy is 92.05
```

In [ ]: