import numpy as np

import matplotlib.pyplot as plt

import random

"""

######### Homework\_2(a) ################

**2(a) Generate a 4 x 100 data matrix X, initially polulated with zeros.**

"""

numRows = 4

numCols = 100

X = np.zeros((numRows, numCols), dtype = float)

#print (X)

"""

######### Homework\_2(b) ################

2(b) Populate the First row with independently identically distributed (i.i.d.) random variables N(165; 25) (simulating height), and the second row with i.i.d. random variables N(137; 100) (simulating weight).

"""

height = X[0]

weight = X[1]

#whRatioList = [0.0] \* (numRows\*numRows\*numCols) # Generating a list of size numRows\*numRows

whRatioList = [0.0] \* (numCols) # Generating a list of size numRows\*numRows

whRatioArray = np.asarray(whRatioList) # Converting the list into array

print (whRatioArray.size)

for i in range(whRatioArray.size):

height[i] = np.random.normal(loc=165, scale=25, size=1)

weight[i] = np.random.normal(loc=137, scale=100, size=1)

# print("Before Cleaning Data: Weight[%d] / Height[%d] = %d / %d " % (i, i, weight[i], height[i]))

print("Before Cleaning Data, the whRatioArray contains:")

"""

######### Homework\_2(c) ################

**(c) Does this data make sense? Are all values reasonable? How would you clean/preprocess this data?**

"""

"""

Not really. Due to its large variance = 100 in the weight calculation with a mean =100 it may generate some negative or zero weights that are undesirable. Thus, I would like to preprocess the weights as follows, that will give all positive and pretty desirable weights.

"""

print("Data Cleaning/Preprocessing .... ")

for i in range(whRatioArray.size):

if (abs(weight[i]) < abs(height[i])):

weight[i] = abs(weight[i]) + abs(height[i])

# print("After Cleaning Data: Weight[%d]/Height[%d] = %d / %d " % (i, i, weight[i], height[i]))

"""

######### Homework\_2(d) ################

**2(d) Visualize/plot your preprocessed data (in two dimensions).**

"""

plt.plot(weight, color = 'blue')

plt.plot(height, color = 'red')

plt.title("Plot Visualization of Processed Samples")

plt.xlabel("Height (in cm)")

plt.ylabel("Weight (in lb)")

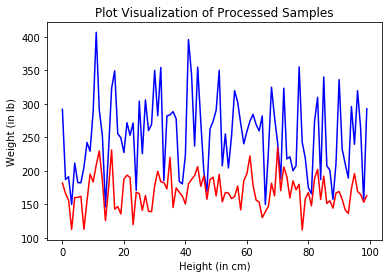
plt.show()

"""

Conclusion:

The scatter plot shows there is a pretty positive correlation between heights and weights

"""



"""

######### Homework\_2(e) ################

**2(e) Compute the weight/height ratio of each sample, and show its histogram.**

"""

for i in range(whRatioArray.size):

whRatioArray[i] = float(weight[i] / height[i])

# print("After Cleaning Data: Weight[%d]/Height[%d] = %d/%d = %f" % (i, i, weight[i], height[i], whRatioArray[i]))

#print (whRatioArray)

plt.hist(whRatioArray, bins=10) # arguments are passed to np.histogram

plt.title("Weight/Height ratio Histogram with Preprossed Data in 10 bins")

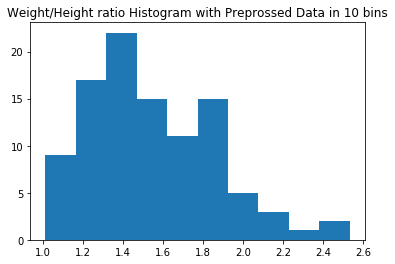
plt.show()

"""

**Conclusion:**

A histogram plot showing that most of the samples have Weight/Height ratio around 1.5. But it varies, run to run.

"""



"""

######### Homework\_2(f) ################

**2(f) Model/simulate the glucose level of each individual as a noisy version of its weight/height ratio (ratio + noise), and store this value in the third row of your data matrix X. Let the noise be i.i.d. N(0, SigmaSquared).**

"""

glucoseLevel = X[2]

sigmaValue = 0.25

glucoseLevelList = [0.0] \* (numCols) # Generating a list of size numRows\*numRows

glucoseLevelArray = np.asarray(glucoseLevelList) # Converting the list into array

print (glucoseLevelArray.size)

for i in range(glucoseLevelArray.size):

glucoseLevelArray[i] = float(whRatioArray[i] + np.random.normal(loc=0, scale=sigmaValue, size=1))

# print("Weight[%d]/Height[%d] = %f vs. Glucose[%d] = %f" % (i, i, whRatioArray[i], i, glucoseLevelArray[i] ))

#print("For the noisy version of the, the glucoseLevelArray contains:")

#print (glucoseLevelArray)

"""

######### Homework\_2(g) ################

**2(g) We will model an individual as healthy (label = 0) if its glucose level (as defined above) is below a threshold TauValue, and diabetic otherwise (label = 1). Store these labels in the fourth row of your data matrix X.**

"""

healthStatus = X[3]

healthyList = []

diabeticList = []

sigmaValueList = [0.2, 0.3, 0.4, 0.5]

tauValueList = [1.0, 1.25, 1.5, 1.75]

sigmaValueArray = np.asarray(sigmaValueList)

tauValueArray = np.asarray(tauValueList)

for i in range(healthStatus.size):

glucoseLevel[i] = float(whRatioArray[i] + np.random.normal(loc=0, scale=random.choice(sigmaValueArray), size=1))

if (abs(glucoseLevel[i]) <= random.choice(tauValueArray)):

healthStatus[i] = 0

healthyList.append(healthStatus[i])

else:

healthStatus[i] = 1

diabeticList.append(healthStatus[i])

# print("glucoseLevel[%d] = %f vs. healthStatus[%d] = %d" % (i, glucoseLevel[i], i, healthStatus[i]))

print ("# of Healthy Samples = ", len(healthyList))

print ("# of Diabetic Samples = ",len(diabeticList))

"""

######### Homework\_2(h) ################

**2(h) Visualize/plot the clustered data (in two dimensions) for different values of Sigma and Tau.**

"""

plt.plot(healthyList, color = 'blue') ## here the plot has to be transparent so we need to pic low alpha value

plt.plot(diabeticList, color = 'red') ## here the plot has to be transparent so we need to pic low alpha value

plt.title("Healthy and Diabetic Sample Distribution")

plt.xlabel("Sample Count")

plt.ylabel("Healthy(0) --- Diabetic(1)")

plt.show()

print ("# of Healthy Samples = ", len(healthyList))

print ("# of Diabetic Samples = ",len(diabeticList))

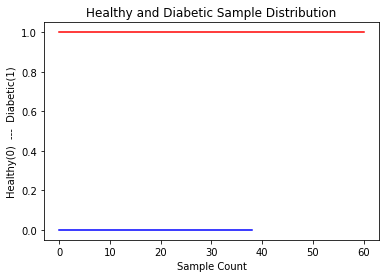
"""

**Conclusion:** People with higher glucose level more likely have diabetic. thus unhealthy

"""

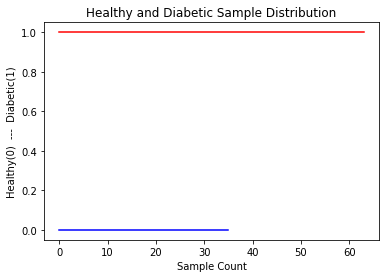
**sigmaValueList = [0.2, 0.3, 0.4, 0.5]**

**tauValueList = [1.0, 1.25, 1.5, 1.75]**



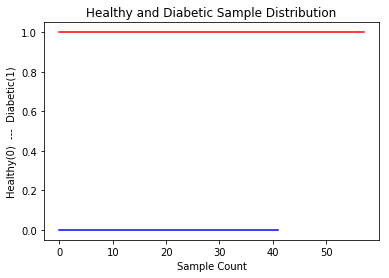
# of Healthy Samples = 39

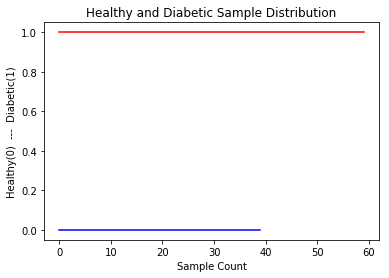
# of Diabetic Samples = 61

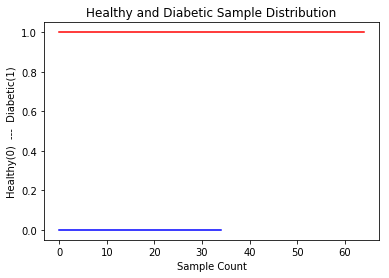


# of Healthy Samples = 36

# of Diabetic Samples = 64







"""

######### Homework\_2(i) ################

**2(i) How do Sigma and Tau affect the data?**

"""

"""

**Conclusion**: As Sigma value increases glucose level increases. So the higher sigma value has slight effect on glucose level.

On the other hand, as Tau value increases with same glucose level less samples will fall into having diabetic group.

I have run the plots a number of times, however, did not find any hard and solid relation between the Sigma and Tau values for randomly generated samples.

Some plots output are included above, along with the Sigma and Tau value lists. However, I this model I have randomly selected Sigma and Tau values from this list.

"""

#