# for parts a and b. Only need to change the file name, it can achieve all requirments

import math

from plotDecBoundaries import \*

xaxistemp = list()

yaxistemp = list()

labeltemp = list()

xaxis = list()

yaxis = list()

label = list()

xaxistemp2 = list()

yaxistemp2 = list()

labeltemp2 = list()

xaxis2 = list()

yaxis2 = list()

label2 = list()

average = list()

tdata = list()

tlabeltemp = list()

temp = list()

tlabel = list()

distance1 = list()

distance2 = list()

output = [0] \* 100

count = 0

trainingdata = list()

trainingdata2 = list()

def distance\_function(test\_val1, average\_val1, test\_val2, average\_val2):

square = (test\_val1 - average\_val1) \*\* 2 + ((test\_val2 - average\_val2) \*\* 2)

return math.sqrt(square)

with open('synthetic2\_train.csv') as train:

tfile = train.readlines()

for line in tfile:

line = line.split(',') # cut to piece

xaxistemp.append(line[0])

yaxistemp.append(line[1])

labeltemp.append(line[2])

for x in xaxistemp: # to float

xaxis.append(float(x))

for y in yaxistemp: # to float

yaxis.append(float(y))

for l in labeltemp: # to float

label.append(int(l))

xaverage = sum(xaxis[0: 50]) / 50

xaverage2 = sum(xaxis[50: 100]) / 50

yaverage = sum(yaxis[0: 50]) / 50

yaverage2 = sum(yaxis[50: 100]) / 50

average = [[xaverage, yaverage], [xaverage2, yaverage2]]

trainingdata.append(xaxis)

trainingdata.append(yaxis)

trainingdata = np.array(trainingdata) #input

trainingdata = trainingdata.T

label = np.array(label)

with open('synthetic2\_test.csv') as testdata:

tfile2 = testdata.readlines()

for line2 in tfile2:

temp.append(line2.split(','))

for data in temp:

tdata.append(data[0:2])

for i in range(100):

tdata[i] = [float(data) for data in tdata[i]]

for data in temp:

tlabeltemp.append(data[2])

for temp in tlabeltemp:

tlabel.append(int(temp))

for dis in tdata:

distance1.append(distance\_function(dis[0], average[0][0], dis[1], average[0][1]))

for dis2 in tdata:

distance2.append(distance\_function(dis2[0], average[1][0], dis2[1], average[1][1]))

for i in range(100):

if distance1[i] < distance2[i]:

output[i] = 1

else:

output[i] = 2

for i in range(len(tdata)):

if output[i] != tlabel[i]:

count = count + 1

error = count / 100

print(average)

print(output)

print(tlabel)

print(error)

average = np.array(average)

with open('synthetic2\_test.csv') as train2:

tfile2 = train2.readlines()

for line2 in tfile2:

line2 = line2.split(',') # cut to piece

xaxistemp2.append(line2[0])

yaxistemp2.append(line2[1])

labeltemp2.append(line2[2])

for x2 in xaxistemp2: # to float

xaxis2.append(float(x2))

for y2 in yaxistemp2: # to float

yaxis2.append(float(y2))

for l2 in labeltemp2: # to float

label2.append(int(l2))

trainingdata2.append(xaxis2)

trainingdata2.append(yaxis2)

trainingdata2 = np.array(trainingdata2) #input

trainingdata2 = trainingdata2.T

label2 = np.array(label2)

plotDecBoundaries(trainingdata2, label2, average)

# for parts c d e. Only need to change the file name, it can achieve all requirments

import math

from plotDecBoundaries import \* # import all functions

def dispose(file, choice, choice2):

xaxis = list()

yaxis = list()

label = list()

xaxistemp = list()

yaxistemp = list()

labeltemp = list()

average = list()

temp = list()

temp2 = list()

temp3 = list()

with open(file) as train:

tfile = train.read().splitlines()

for line in tfile:

line = line.split(',') # cut to piece

xaxistemp.append(line[choice])

yaxistemp.append(line[choice2])

labeltemp.append(line[13])

for x in xaxistemp: # to float

xaxis.append(float(x))

for y in yaxistemp: # to float

yaxis.append(float(y))

for l in labeltemp: # to float

label.append(int(l))

for j in range(len(label)):

if label[j] == 1:

temp.append(j)

if label[j] == 2:

temp2.append(j)

if label[j] == 3:

temp3.append(j)

xaverage = sum(xaxis[min(temp): max(temp) + 1]) / len(temp)

yaverage = sum(yaxis[min(temp): max(temp) + 1]) / len(temp)

xaverage2 = sum(xaxis[min(temp2): max(temp2) + 1]) / len(temp2)

yaverage2 = sum(yaxis[min(temp2): max(temp2) + 1]) / len(temp2)

xaverage3 = sum(xaxis[min(temp3): max(temp3) + 1]) / len(temp3)

yaverage3 = sum(yaxis[min(temp3): max(temp3) + 1]) / len(temp3)

average = [[xaverage, yaverage], [xaverage2, yaverage2], [xaverage3, yaverage3]]

return xaxis, yaxis, label, average

def distance\_function(test\_val1, average\_val1, test\_val2, average\_val2):

square = (test\_val1 - average\_val1) \*\* 2 + ((test\_val2 - average\_val2) \*\* 2)

return math.sqrt(square)

def choi(tdata, tmean):

dis = [[], [], []]

outcome = [0] \* len(tdata)

for point in tdata:

dis[0].append(distance\_function(point[0], tmean[0][0], point[1], tmean[0][1]))

for point2 in tdata:

dis[1].append(distance\_function(point2[0], tmean[1][0], point2[1], tmean[1][1]))

for point3 in tdata:

dis[2].append(distance\_function(point3[0], tmean[2][0], point3[1], tmean[2][1]))

for i in range(len(tdata)):

if dis[0][i] < dis[1][i]:

if dis[1][i] < dis[2][i]:

outcome[i] = 1

else:

if dis[0][i] < dis[2][i]:

outcome[i] = 1

else:

outcome[i] = 3

else:

if dis[0][i] < dis[2][i]:

outcome[i] = 2

else:

if dis[1][i] < dis[2][i]:

outcome[i] = 2

else:

outcome[i] = 3

return outcome

min\_error = float("inf")

det\_k = 100

det\_j = 100

errormean = 0

errorall = list()

var = 0

temp5 = []

for k in range(13):

for j in range(k+1, 13):

choice = k

choice2 = j

x, y, l, mdata = dispose('wine\_train.csv', choice, choice2)

with open('wine\_train.csv') as testdata:

tfile2 = testdata.readlines()

for line2 in tfile2:

temp5.append(line2.split(','))

testdata = [data[choice: choice2 + 1: choice2 - choice] for data in temp5]

tlabel = [data[13] for data in temp5]

tlabel = [int(test) for test in tlabel]

for m in range(len(testdata)):

testdata[m] = [float(data) for data in testdata[m]]

output = choi(testdata, mdata)

count = 0

for n in range(len(testdata)):

if output[n] != tlabel[n]:

count = count + 1

error = count / len(testdata)

errorall.append(error)

if error < min\_error:

min\_error = error

det\_k = choice

det\_j = choice2

for errormean2 in errorall:

errormean = errormean + errormean2

errormean = errormean / 78

for errormean3 in errorall:

var = var + ((errormean3 - errormean)\*\*2)

var = var / (78 - 1)

deviation = math.sqrt(var)

print(errormean)

print(var)

print(deviation)

det\_k2 = 0

det\_j2 = 1

x2, y2, l2, mdata2 = dispose('wine\_train.csv', det\_k2, det\_j2)

training2 = []

training2.append(x)

training2.append(y2)

training2 = np.array(training2)

mdata2 = np.array(mdata2)

l2 = np.array(l2)

x, y, l, mdata = dispose('wine\_train.csv', det\_k, det\_j)

# this is for calculate error

with open('wine\_test.csv') as testdata:

#with open('wine\_train.csv') as testdata:

temp4 = list()

tfile2 = testdata.readlines()

for line2 in tfile2:

temp4.append(line2.split(','))

testdata = [data[det\_k: det\_j + 1: det\_j - det\_k] for data in temp4]

# test\_data = [data[det\_k2: det\_j2 + 1: det\_j2 - det\_k2] for data in temp4]

print(det\_k2)

print(det\_j2)

tlabel = [data[13] for data in temp4]

tlabel = [int(test) for test in tlabel]

for m in range(len(testdata)):

testdata[m] = [float(data) for data in testdata[m]]

output = choi(testdata, mdata)

# output = choi(test\_data, mean\_data2)

print(mdata)

print(testdata)

print(output)

print(tlabel)

count = 0

for n in range(len(testdata)):

if output[n] != tlabel[n]:

count = count + 1

error = count / len(testdata)

print(count)

print(error)

x, y, l, mean\_data3 = dispose('wine\_test.csv', det\_k, det\_j)

training1 = []

training1.append(x)

training1.append(y)

training1 = np.array(training1)

mdata = np.array(mdata)

l = np.array(l)

# plotDecBoundaries(train\_dataset2.T, label\_data2, mean\_data2)

plotDecBoundaries(training1.T, l, mdata)