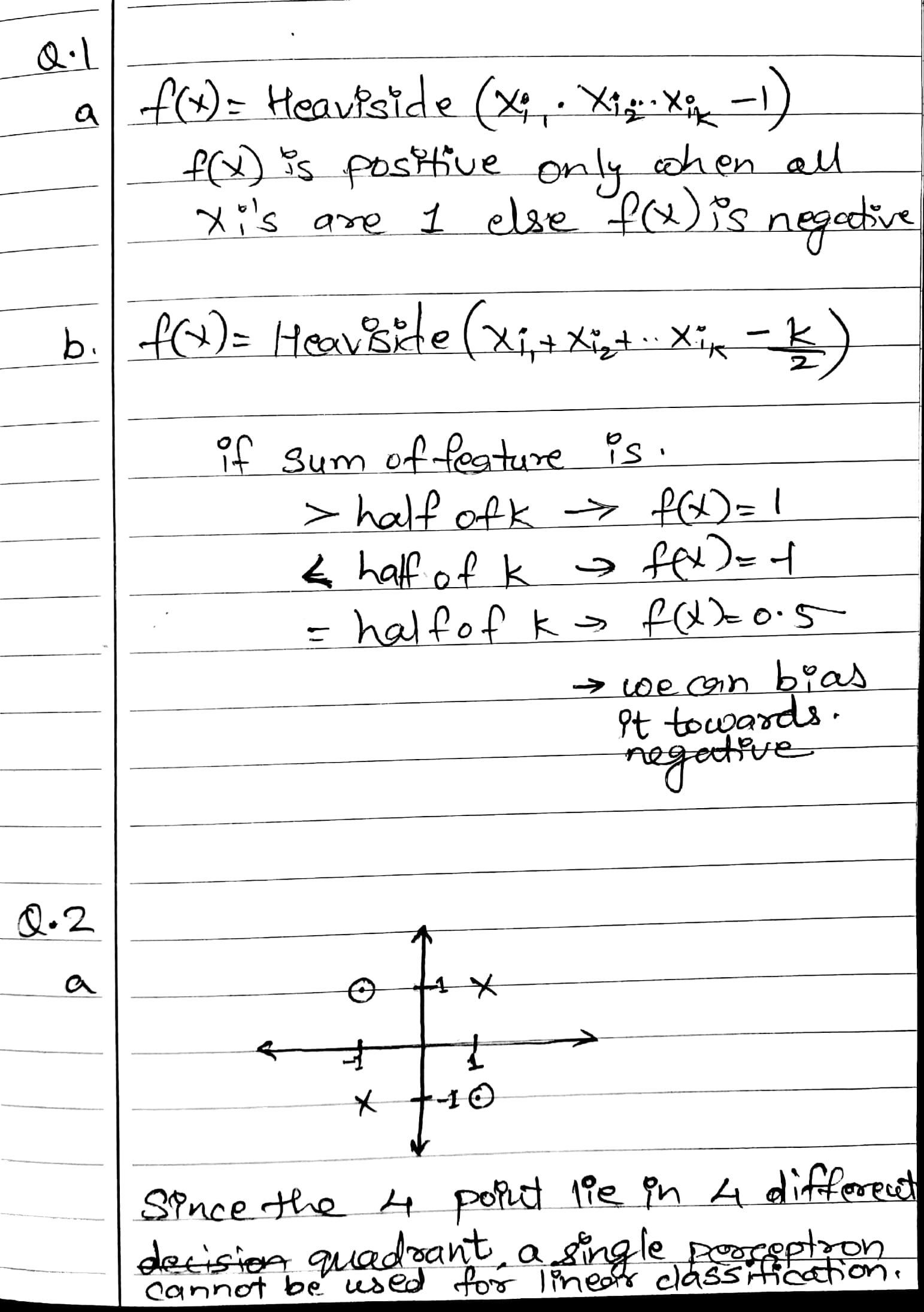
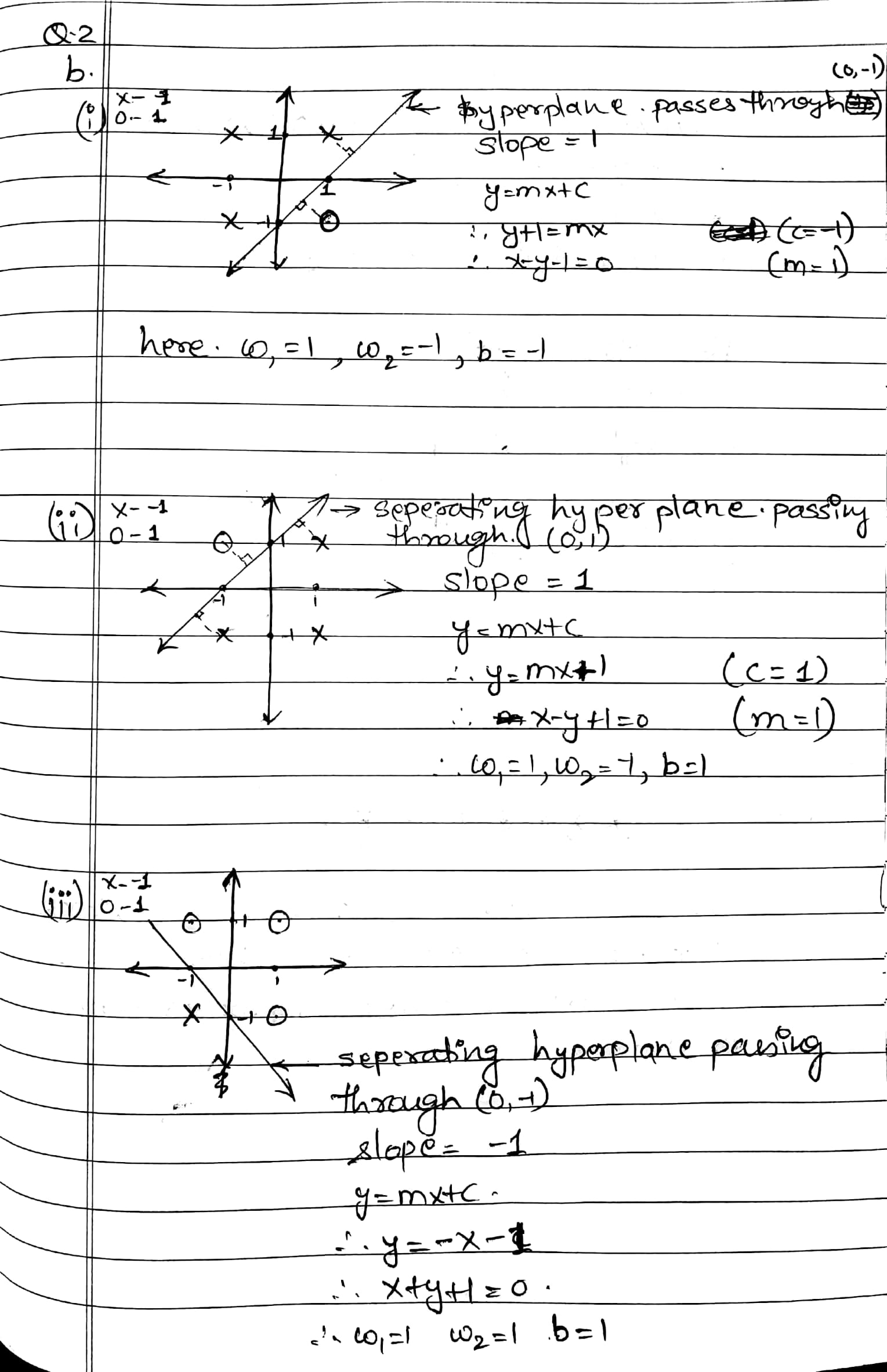
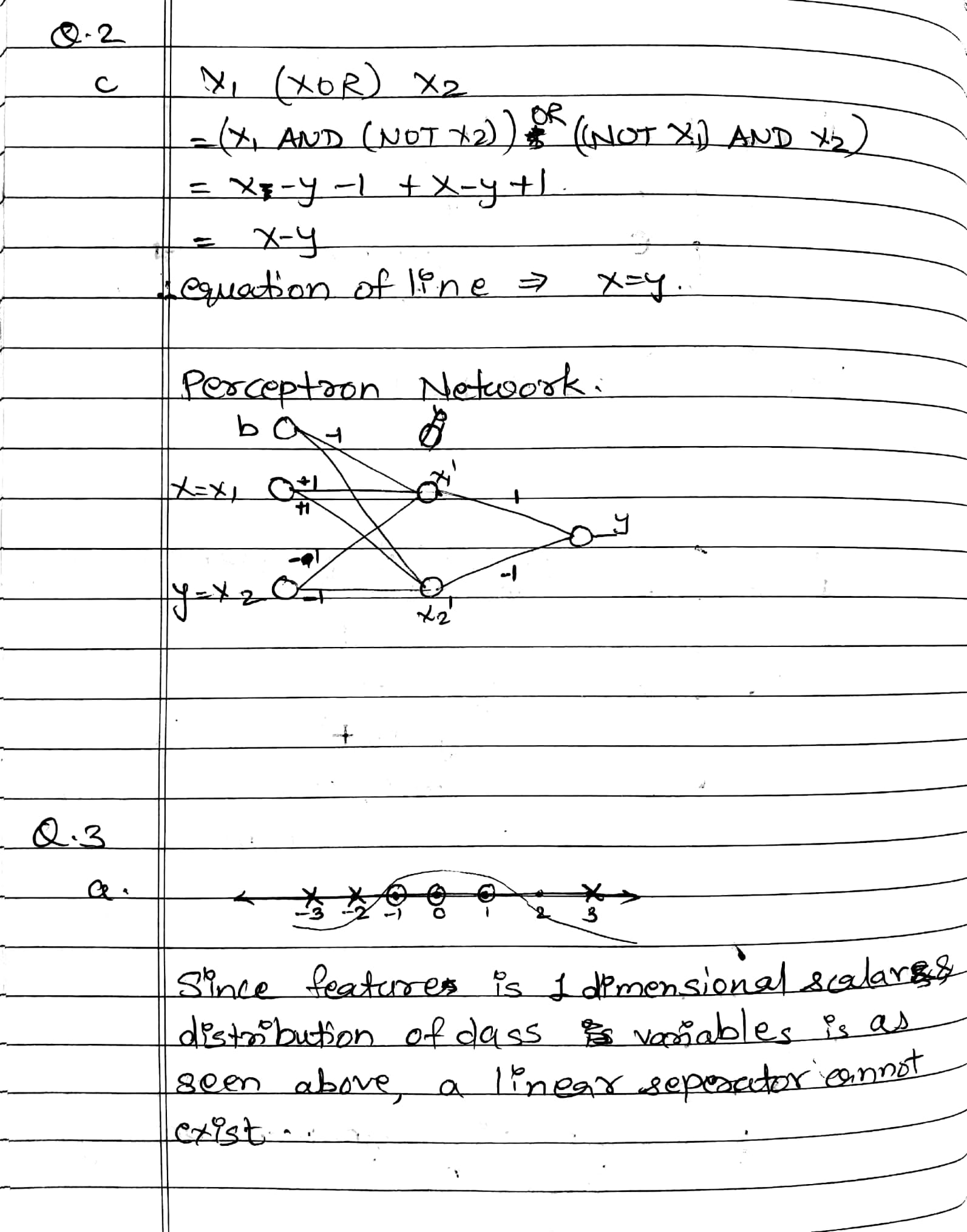
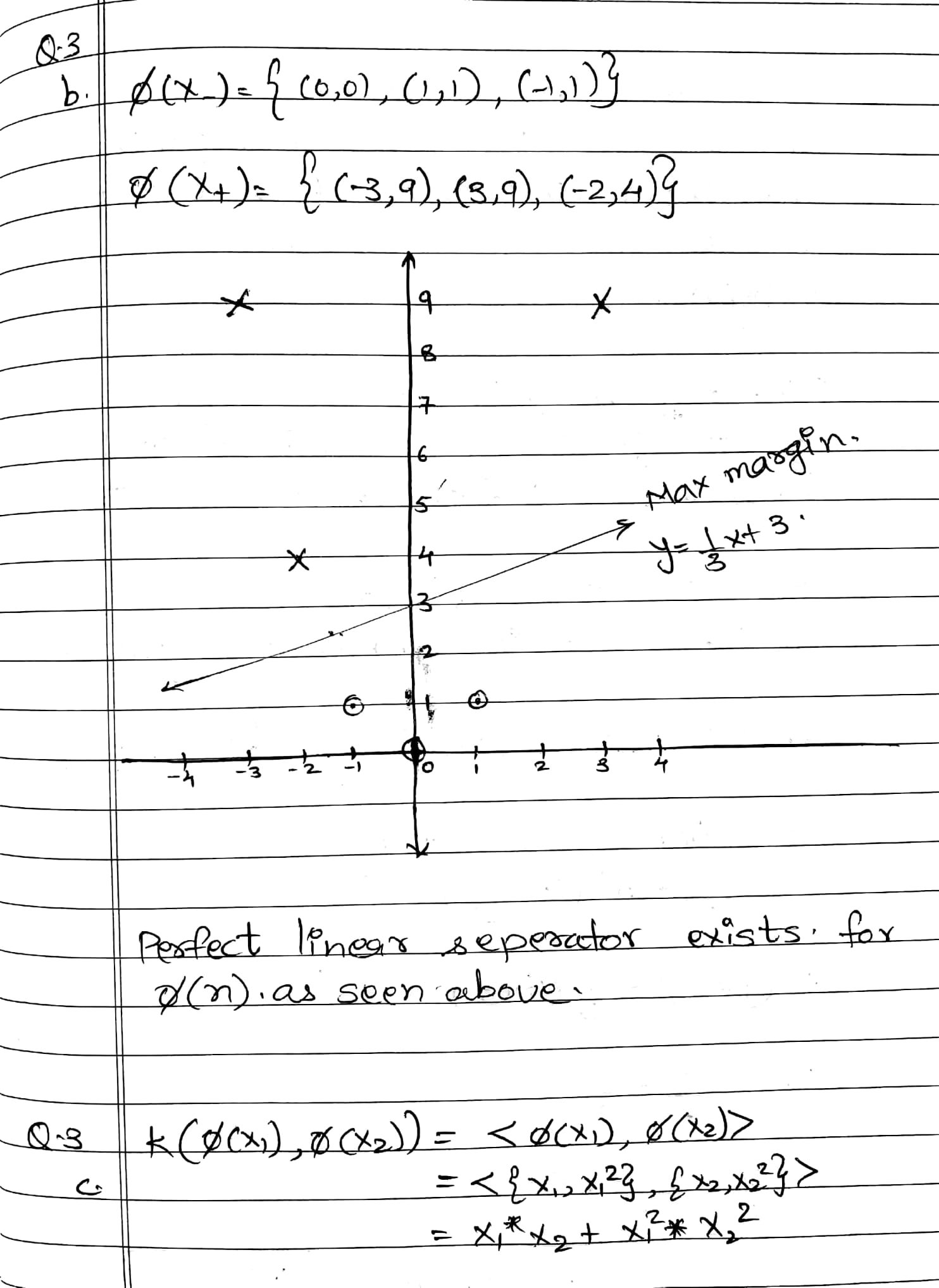
**EE 525 – Assignment 2**

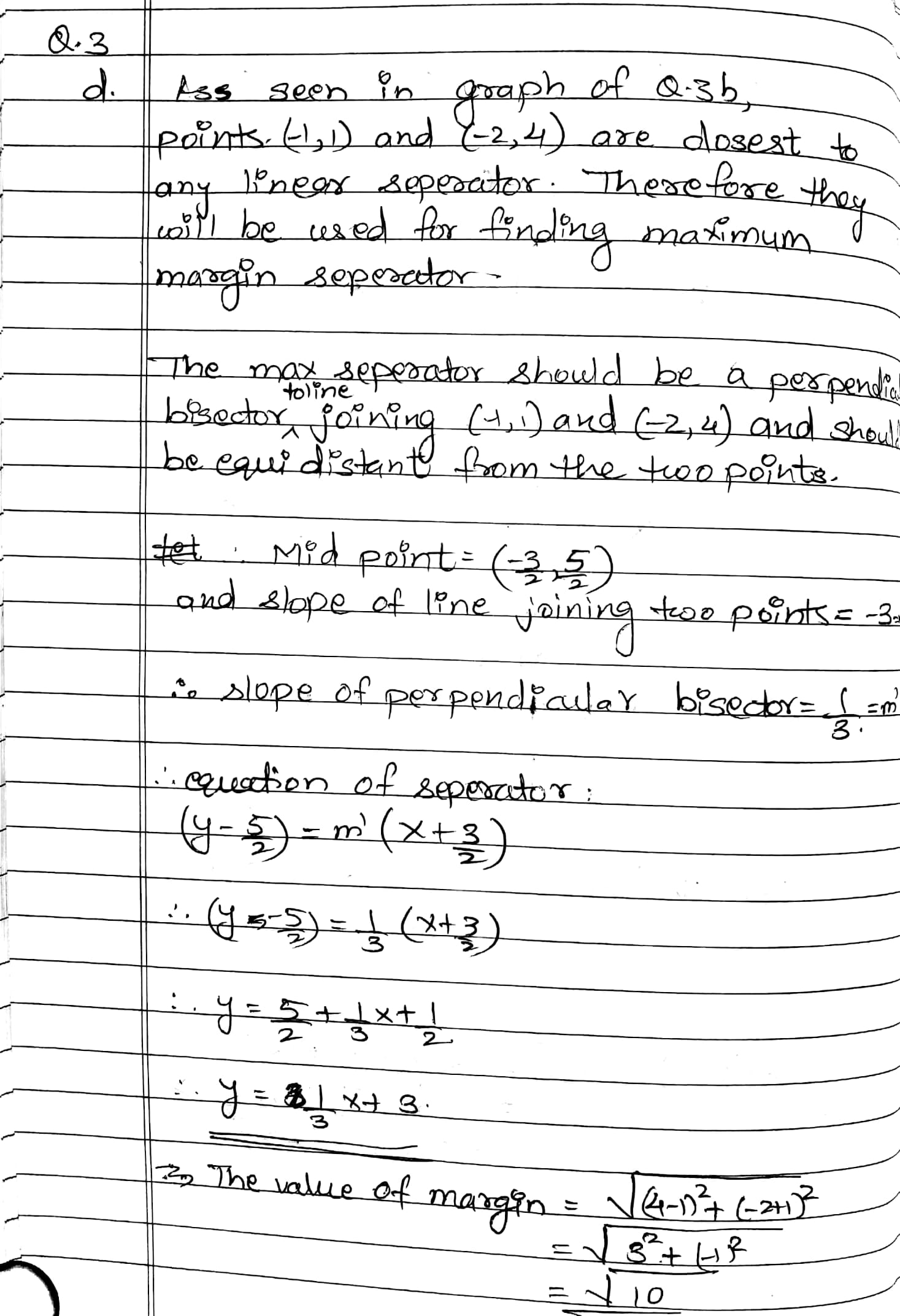
**Tanmay Gore -** [**Tgore03@iastate.edu**](mailto:Tgore03@iastate.edu)

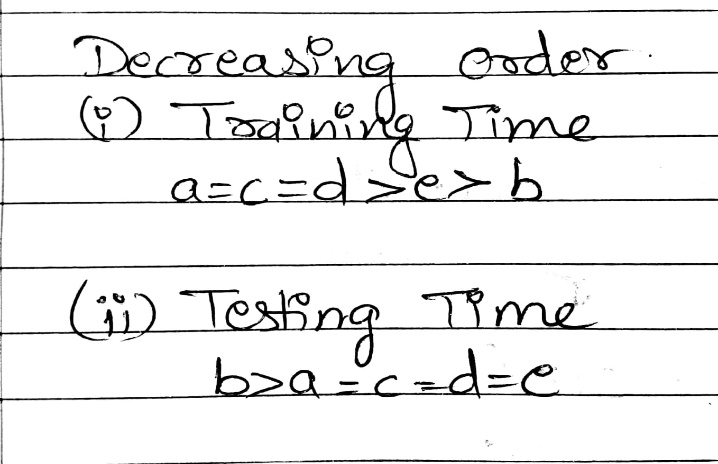
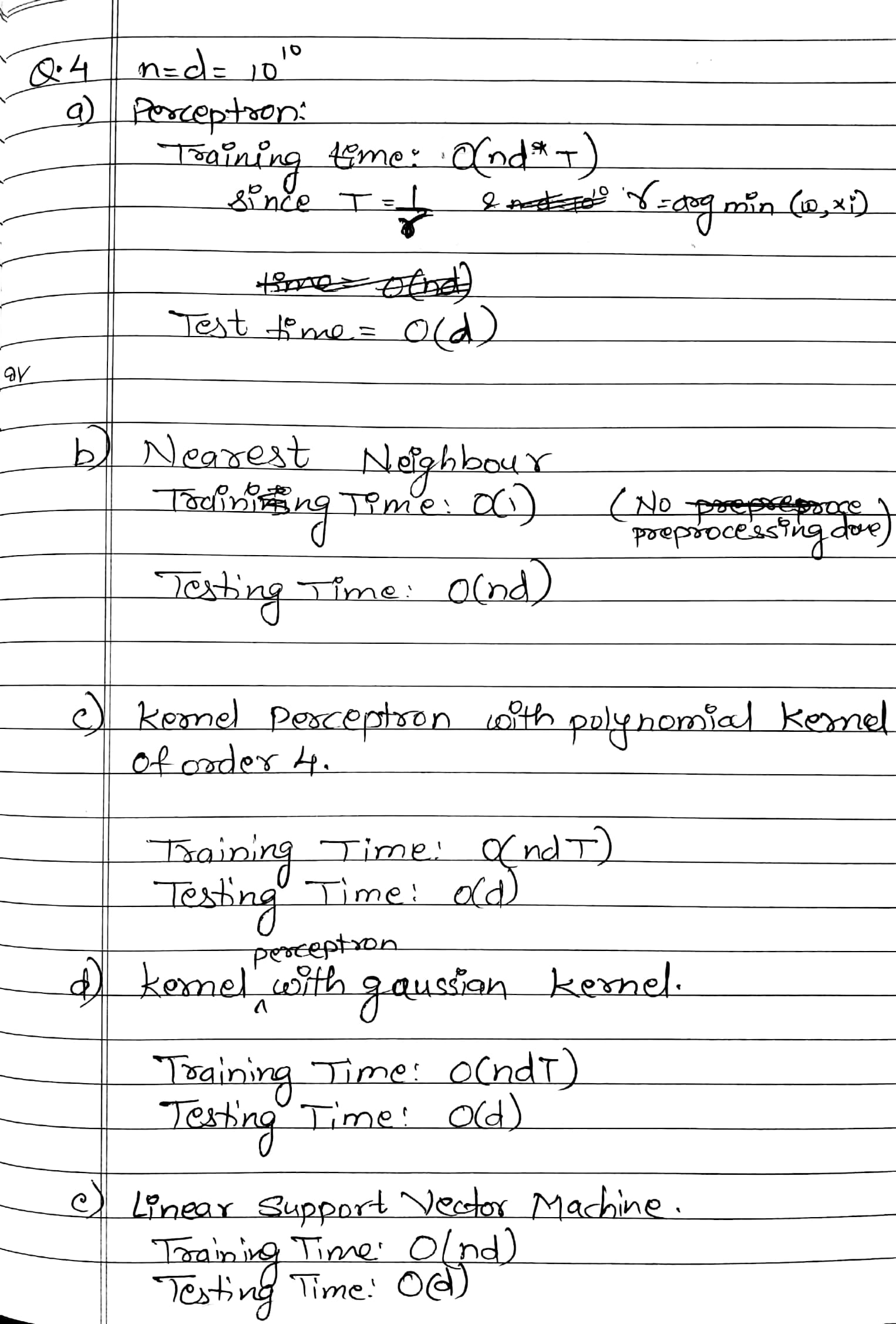






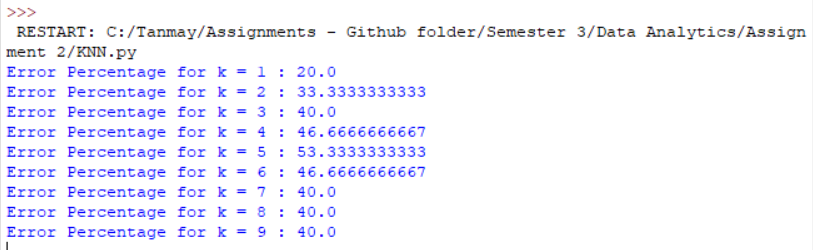


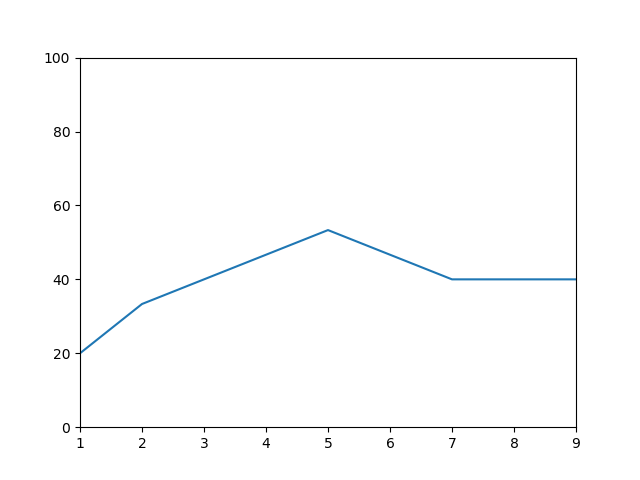




Q5)

KNN – Without Normalization





**Code:**

import csv

import numpy as np

import random

from sklearn import neighbors

import sys

import matplotlib.pyplot as plt

#Variable definition

CLASS1 = 59

CLASS2 = 71

CLASS3 = 48

totalrecords = 178

train\_rows = (CLASS1-5)+(CLASS2-5)+(CLASS3-5)

train\_cols = 13

global train\_data, train\_label, test\_data, test\_label, label, data

global knn, wrong\_class, neighbors, error

def main():

#readcsv data

data = np.genfromtxt('wine.data.csv', delimiter=',')

label = data[:, 0]

data = data[:,1:]

#train data

global wrong\_class, neighbors, error

error=[0]\*10

splitdata(data, label)

for i in range(1,10):

train\_model(i)

#predict data label

iter\_no = 100

wrong\_class=0

predict\_label(i, iter\_no)

error\_prob(i, 15, iter\_no)

plot\_error()

def plot\_error():

plt.plot(error[:])

plt.axis([1, 9, 0, 100])

plt.show()

def error\_prob(k, total\_data, iter\_no):

global wrong\_class, error

error[k] = (wrong\_class / (total\_data\*iter\_no)) \* 100.0

print "Error Percentage for k =",k,":", error[k]

def predict\_label(k, iter\_no):

global test\_data, test\_label

global knn, wrong\_class

for i in range(iter\_no):

#result = KNN(k)

result = knn.predict(test\_data)

for i in range(0, len(result)):

if(result[i] != test\_label[i]):

wrong\_class+=1.0

def train\_model(k):

global train\_data, train\_label

global knn

#Train Model

knn = neighbors.KNeighborsClassifier(n\_neighbors = k, algorithm='kd\_tree')

knn.fit(train\_data, train\_label)

def splitdata(data, label):

global train\_data, train\_label, test\_data, test\_label

train\_data = data

train\_label = label

test\_data = np.empty([15, train\_cols])

test\_label = np.empty([15, 1])

no1 = random.sample(range(0, 59-1), 5)

no2 = random.sample(range(59, 130-1), 5)

no3 = random.sample(range(130, totalrecords-1), 5)

no4 = no1 + no2 + no3

train\_data = np.delete(train\_data, (no4), axis=0)

train\_label = np.delete(train\_label, (no4), axis=0)

i=0

for no in no4:

test\_data[i] = data[no]

test\_label[i] = label[no]

i+=1

main()

def calc\_EHdist(train\_row, test\_row):

dist=0.0

for i in range(0, len(train\_row)):

dist += pow((train\_row[i]-test\_row[i]),2)

#print dist

return dist

def kneighbors(min\_val, dist, row\_index, k):

largest=0.0

#print min\_val

index=0

for i in range(0,k):

if min\_val[i][0] > largest:

largest = min\_val[i][0]

index = i

if min\_val[index][0] > dist:

min\_val[index][0] = dist

min\_val[index][1] = row\_index

return min\_val

def KNN(k):

#compute manhattan distance between test data and training data

global test\_data, train\_data

result = []

#For each test record

for test\_row in test\_data:

#argmin of manhattan distance

min\_val=[]

for i in range(0,k):

if(i==0):

min\_val=[[sys.float\_info.max,0]]

continue

min\_val.append([sys.float\_info.max, 0])

row\_index=0

for train\_row in train\_data:

dist = calc\_EHdist(train\_row, test\_row)

min\_val = kneighbors(min\_val, dist, row\_index,k)

row\_index+=1

#return max label

a, b, c = 0,0,0

for minv in min\_val:

label = train\_label[minv[1]]

if label == 1:

a+=1

elif label == 2:

b+=1

else:

c+=1

print a,b,c

if (a>=b & a>=c):

result.append(1)

elif (b>a & b>=c):

result.append(2)

else:

result.append(3)

return result

Q.7)

Time spent = 12 hours

**Acknowledgement**: Discussed with Nitesh Gupta.