2(a) source code and graph

%load data according to the question

d = load( 'detroit.mat' );

%process data

s = d.data(:, 9:10);

%definition

v = [1;1;1;1;1;1;1;1;1;1;1;1;1];

e = [] ;

%variables and factors

HOM = d.data(:,10);

LIC = d.data(:,4);

FTP = d.data(:,1);

WE = d.data(:,9);

matrix = [v, FTP, WE];

%procedures

i = 2

while(i < 9)

store = d.data(:,i);

new = [matrix, store];

%formula

b = (((new')\*new)^(-1))\*(new')\*HOM;

y = new \* b ;

sub = y - HOM;

sub2 = sub.^2;

e1 = sum(sub2);

l\_e = e1/(2\*13);

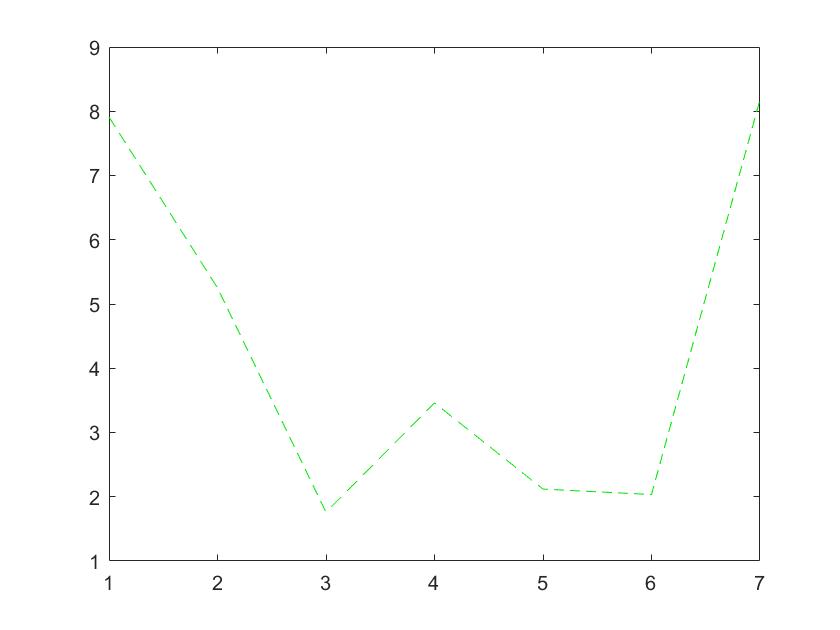
e = [e; l\_e];

i = i + 1 ;

end

result = e

plot(result,'--','color',[0 0.9 0]);



2(b) Since I can’t figure out how to use Matlab to process lenses and CA data, I use Python to work on this question. To process the data, I use Panda Package.

1. I replaced all unknow first features. Then I calculate median of possible values of the missing features that are not numbers. I also replace the attribute by the mode of all attributes. Finally, I use label conditioned mean for real-valued with plus label. The formula is

As for minus data set, the formula is

The detailed procedures are below and in process.py.

#Use Panda to process data

from sys import argv

import pandas as pd

import numpy as np

#command run

script, a, b = argv

#function below

def process(data):

data = data.replace('?', np.NaN)

r = [0,3,4,5,6]

#missing features

column = [1,2,7,10,13,14]

for i in r:

#replace by mode

data[i] = data[i].fillna(data[i].mode()[0])

r = [1,13]

#plus, minus label

lab = ['+', '-']

for m in r:

data[m] = data[m].apply(float)

for n in lab:

#get real-value missing ones

data.loc[ (data[m].isnull()) & ( data[15]==n ), m ] = data[m][data[15] == n].mean()

for c in column:

data[c] = (data[c] - data[c].mean())/data[c].std()

return data

#Use panda to process TrD which stands for training Data and TeD which stands for testing data

TrD = pd.read\_csv(a, header=None)

#process the data

TrD = process(TrD)

TrD.to\_csv('crx.training.processed', header=False, index=False)

#Same Usage as above

TeD= pd.read\_csv(b, header=None)

TeD = process(TeD)

TeD.to\_csv('crx.testing.processed', header=False, index=False)

1. KNN theory and its knowledge are cited from \*https://en.wikipedia.org/wiki/K-nearest\_neighbors\_algorithm \*https://www.analyticsvidhya.com/blog/2018/03/introduction-k-neighbours-algorithm-clustering/

In this question, I am required to write a k-NN algorithm with L2 distance which is DL2(a,b)=

To run the program under command lines, I still use sys import argv in Python. The command line in this case should have 3 parameters. There are 2 scripts needed to run with the Python to make the command work. The code is below:

import math

from sys import argv

import pandas as pd

#command run, parameters

script, KNN, Tr, Te = argv

#Use panda to read

TrD = pd.read\_csv(Tr, header=None)

TeD = pd.read\_csv(Te, header=None)

LAB = []

R1, C1 = TrD.shape

R2, C2 = TeD.shape

nei = []

#find real label to its testing data

def label(list1, TrD):

for i in range(len(list1)):

label = TrD.iloc[list1[i]][ C1-1 ]

LAB.append(label)

#need the label with most times

return max(set(LAB), key=LAB.count)

#distance is calculated below

def dist(m,n):

res = 0

list = []

for i in range(R1):

for j in range(C1 - 1 ):

if(isinstance(m.iloc[i][j], str ) == True):

if(m.iloc[i][j] != n[j]):

res = res + 1

else:

res = res

else:

#DL2(m,n) = sqrt(res(m , n)^2))

diff = math.pow((m.iloc[i][j] - n[j]), 2)

res = res + diff

list.append(math.sqrt(res))

res = 0

return list

TeD[C2] = TeD[C2-1]

a = 0

#run through the data

for index,row in TeD.iterrows():

dis = dist(TrD,row)

ordered = sorted(range(len(dis)), key=lambda k: dis[k])

for i in range(int(KNN)):

nei.append(ordered[i])

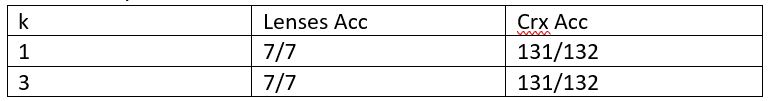
res = label(nei, TrD)

TeD.loc[a, C2] = res

a = a + 1

#use panda to output

TeD.to\_csv(Te, header=False, index=False)

1. The accuracy result is below:
2. Citation:

Theory and knowledge of KNN are referenced from \*https://en.wikipedia.org/wiki/K-nearest\_neighbors\_algorithm \*https://www.analyticsvidhya.com/blog/2018/03/introduction-k-neighbours-algorithm-clustering/