**IE-5300** Praveen Suguru

**Homework-1**  1001099696

1. **Input:**

filename='iris.data';

Data=fopen(filename,'r');

Data=textscan(Data,'%f%f%f%f%s','delimiter',',','collectoutput',1);

Feat=Data{1};

Label=[];

for i=1:150;

if strcmp(Data{2}(i),'Iris-setosa');

Label=[Label;1];

elseif strcmp(Data{2}(i),'Iris-versicolor');

Label=[Label;2];

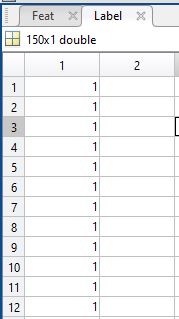
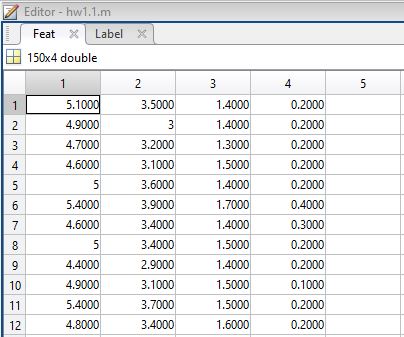
elseif strcmp(Data{2}(i),'Iris-virginica');

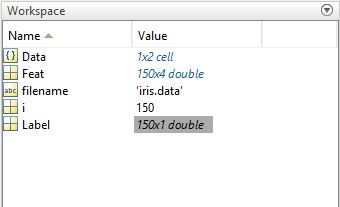
Label=[Label;3];

end

end

**Output:**

****

****

**Interpretation:** Iris data is having three different types of flowers with four attributes. The last column is the label column which is been replaced by 1 for Setosa, 2 for versicolor and 3 for virginica.

**2)**

**2-1)**

**2-D Scatter Plot:**

**Input:**

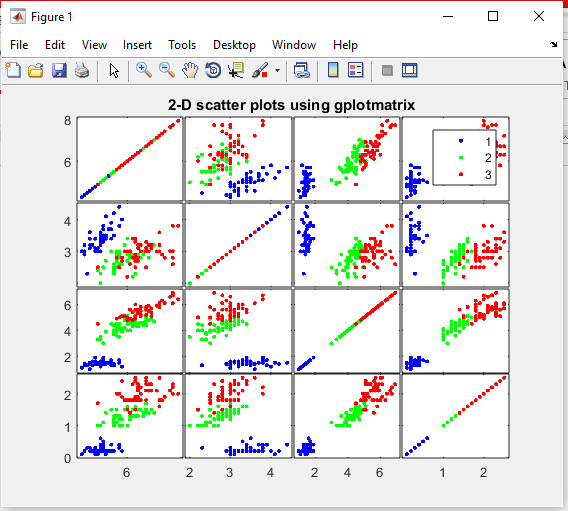
Feat(:,5) = Label(:,1);

attributes=Feat(:,1:4);

gplotmatrix(attributes,attributes,Label);

title('2-D scatter plots using gplotmatrix');

**Output:**

**Interpretation: 2-D plot helps us to identify the different attributes of different flowers but in few instances it is difficult as we can see the points overlap and it is hard to differentiate.**

**2-2)** **3-D Scatter Plot:**

**Input:**

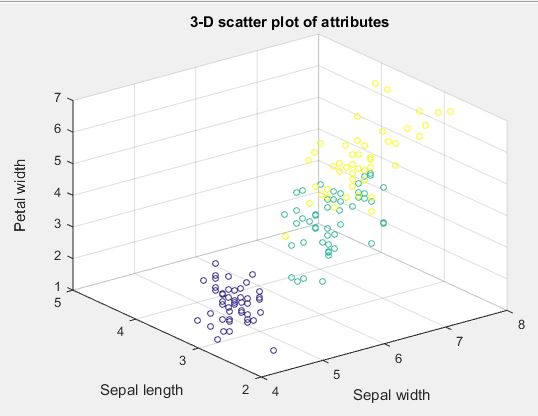
sepalwidth=Feat(:,1);

sepallength=Feat(:,2);

petalwidth=Feat(:,3);

scatter3(sepalwidth,sepallength,petalwidth,20,Label(:,1);

**Output:**

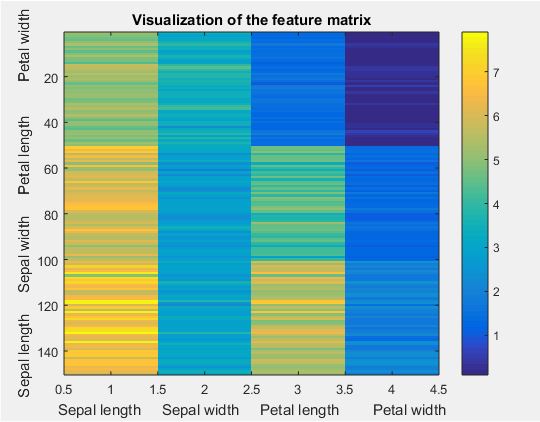


**Interpretation: 3-D plot is the improvisation of 2-D plot which gives a clear identification up to three attributes of any label.**

2-3**) Visualization of the feature matrix:**

**Input:** imagesc(attributes);

**Output:**



**Interpretation: This gives us the probability with respect to its attributes and classes.**

**2-4)** **Histogram of four attributes:**

**Input:**

idx1=find(Label==1);

x1= attribute(idx1,1);

y1= attribute(idx1,2);

z1= attribute(idx1,3);

v1= attribute(idx1,4);

idx2=find(Label==2);

x2= attribute(idx2,1);

y2= attribute(idx2,2);

z2= attribute(idx2,3);

v2= attribute(idx2,4);

idx3=find(Label==3);

x3= attribute(idx3,1);

y3= attribute(idx3,2);

z3= attribute(idx3,3);

v3= attribute(idx3,4);

figure(5)

histogram(x1);

hold on

histogram (x2);

hold on

histogram(x3)

clc

figure(2)

histogram(y1);

hold on

histogram (y2);

hold on

histogram(y3)

clc

figure(3)

histogram(z1);

hold on

histogram (z2);

hold on

histogram(z3)

clc

figure(4)

histogram(v1);

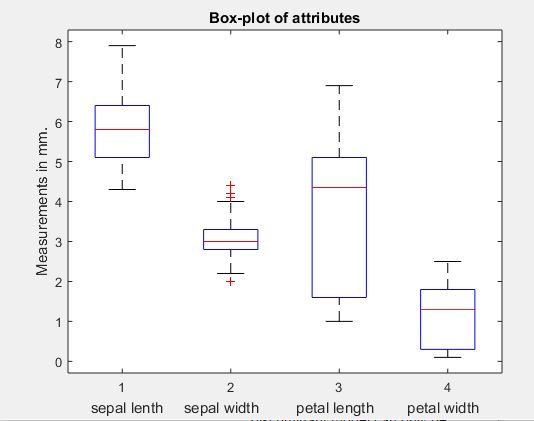
hold on

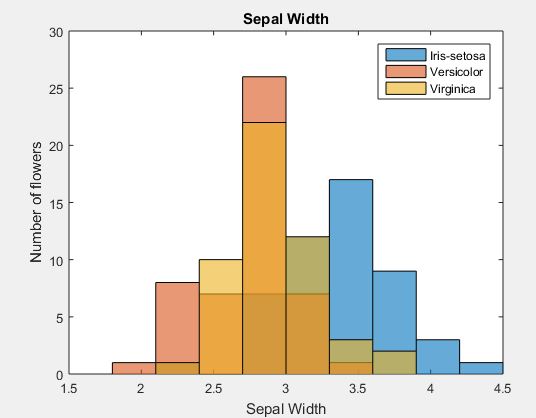
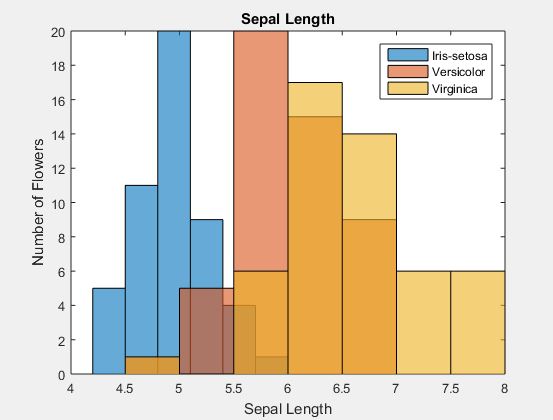
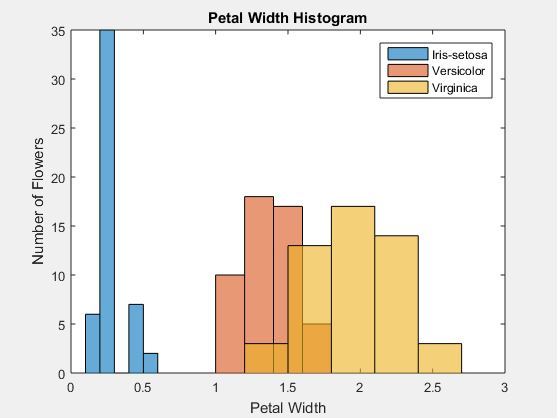
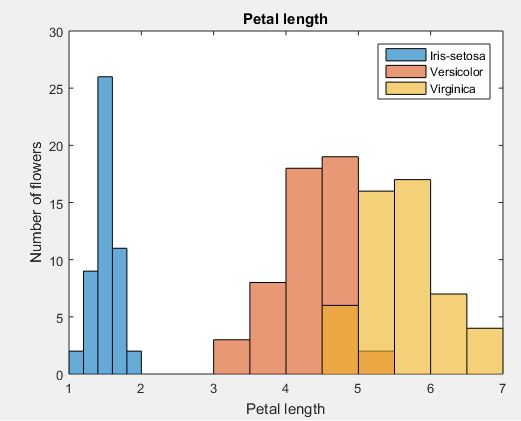
histogram (v2);

hold on

histogram(v3)

**Output:**



****

**2-5) Box-plots of four attributes:**

**Input:** boxplot(attribute)

**Output:**

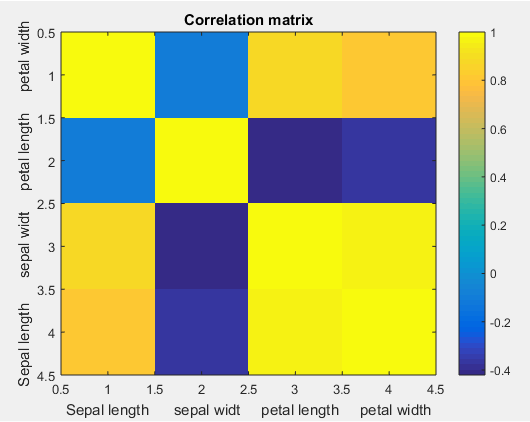
**Interpretation: Box plot gives the mean and limits of each and every attribute in the data. Red line represents the mean of the data.**

**2-6) Correlation matrix of attributes and Visualization:**

**Input:** correlation=corr(attribute);

imagesc(correlation)

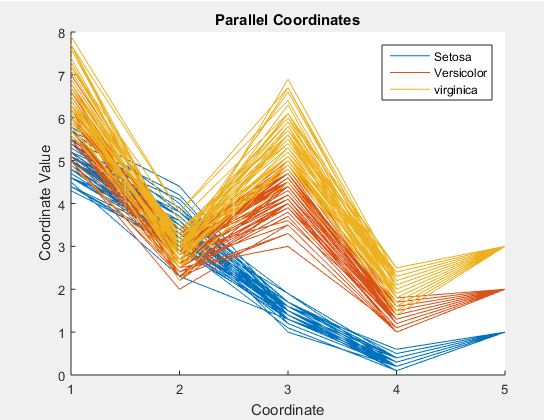
**Output:**



**2-7) Parallel coordinates of attributes:**

**Input:** parallelcoords(Feat, 'Group', Label)

**Output:**



**3-1) Minkowski distance:**

**Input:**

function d = Minkowski\_distance(x,y,p)

d = sum(abs(x-y).^p).^(1/p);

end

**Output:**

d = Minkowski\_distance([2 3 4],[3 4 5],2)

d = 1.7321

**3-2)** **T-Statistics distance:**

**Input:**

function s = T\_distance(vec1,vec2)

s = (abs(mean(vec1)-mean(vec2)))/(std(vec1-vec2));

end

**Output**:

f=T\_distance([44 67 76],[40 65 23])

f =0.6809

**3-3)** **Mahalanobis distance:**

**Input:**

function m\_dist = mahal\_distance(vec1,vec2,P)

m\_dist = (vec1-vec2) .\* inv(P) .\* (vec1-vec2)';

end

**Output:**

m=mahal\_distance([1 4;2 5],[5 6; 3 4])

m =

4.8980 -0.1224

-0.1224 0.6122

**3-4) Minkowski Distance:**

**Input:**

load iris.mat

feat\_S=[5,3.50,1.460,0.2540];

Nsp = length(label);

promt=('enter value of r = ');

r = input(promt);

Dlist=[];

for i = 1:Nsp

feat\_i = feat(i, :);

m\_dist = Minkowski\_distance(feat\_S, feat\_i,r);

Dlist(i) = m\_dist;

end

% plot the distance vector

figure;

plot(Dlist);

xlabel('Samples in each class')

ylabel('Euclidean Distance');

title('Distance between Sample S and the Samples in Iris Dataset');

% plot the distance vector of each class

figure;

c = unique(label);

for ic = c'

idx= find(label==ic);

Dvec = Dlist(idx);

if ic==1; plot(Dvec, 'r'); end

if ic==2; plot(Dvec, 'k'); end

if ic==3; plot(Dvec, 'g'); end

hold on;

end

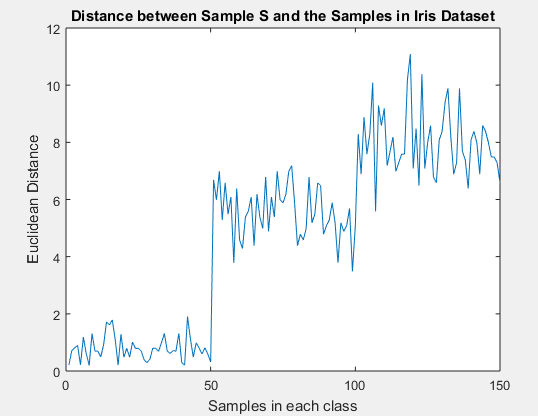
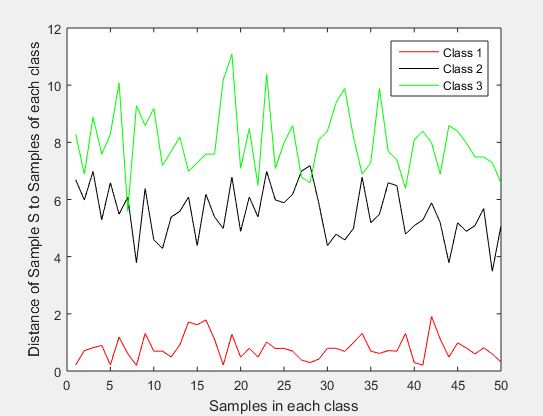
xlabel('Samples in each class');

ylabel('Distance of Sample S to Samples of each class');

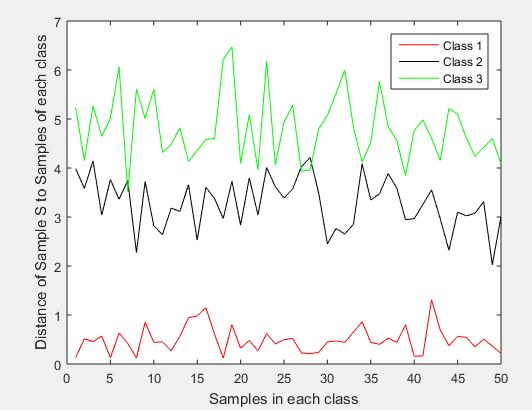
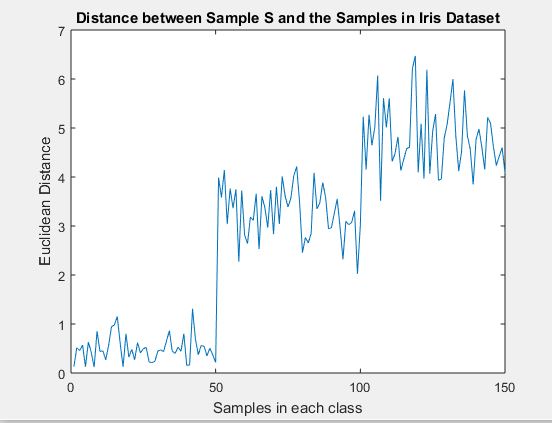
legend('Class 1', 'Class 2', 'Class 3');

**Output:**

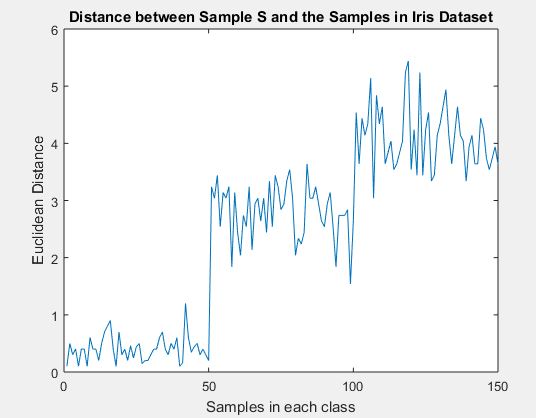
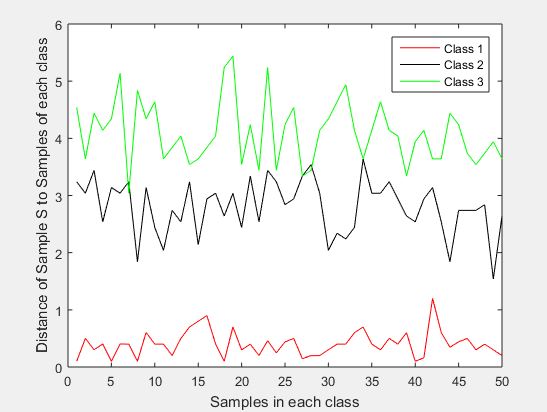
**R=1**

****

**R=2**

****

**R=100**

****

**3-5) Malahanobis Distance:**

**Input:**

load iris.mat

feat\_S=[5,3.50,1.460,0.2540];

Nsp = length(label);

P =cov(feat);

Dlist=[];

for i = 1:Nsp

feat\_i = feat(i, :);

% Here apply the Euclidean distance function we made in EU\_distance.m

m\_dist = mahal\_distance(feat\_S, feat\_i,P);

Dlist(i) = m\_dist;

end

% plot the distance vector

figure;

plot(Dlist);

xlabel('Samples in each class')

ylabel('Euclidean Distance');

title('Distance between Sample S and the Samples in Iris Dataset');

% plot the distance vector of each class

figure;

c = unique(label);

for ic = c'

idx= find(label==ic);

Dvec = Dlist(idx);

if ic==1; plot(Dvec, 'r'); end

if ic==2; plot(Dvec, 'k'); end

if ic==3; plot(Dvec, 'g'); end

hold on;

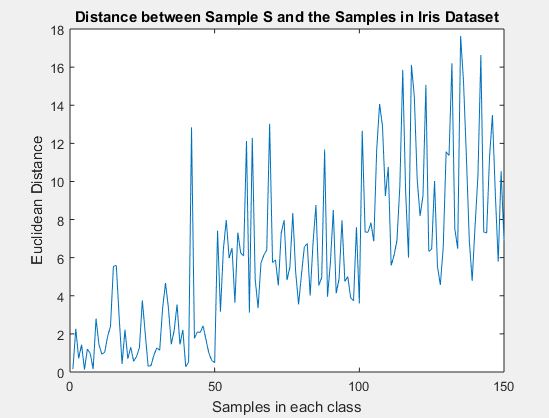
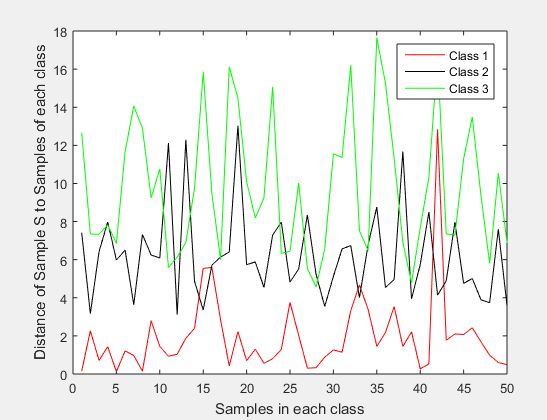
end

xlabel('Samples in each class');

ylabel('Distance of Sample S to Samples of each class');

legend('Class 1', 'Class 2', 'Class 3');

**Output:**



**3-6) Plot Two time series:**

**Input:**

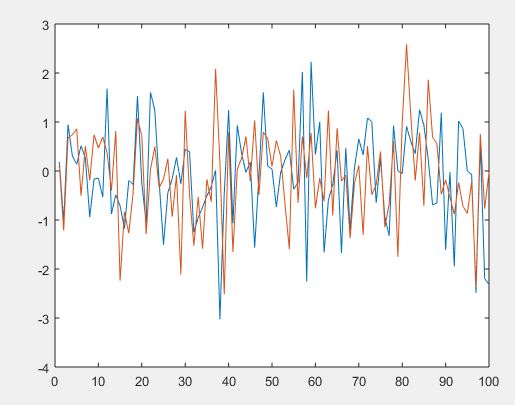
mu = [0;0];

sigma=[1 .3;.3 1];

S = mvnrnd(mu,sigma,100);

plot(S)

**Output:**



**3-7) T-statistics distance between two time series:**

**Input:**

A= S(:,1);

B=S(:,2);

function s = T\_distance(vec1,vec2)

s = (abs(mean(vec1)-mean(vec2)))/(std(vec1-vec2));

end

**Output:**

s= T\_distance(A,B)

s = 0.0141

**3-8) Correlation of time series:**

**corr(A,B)**

**ans = 0.3549**

**3-9) Normalization:**

**Input:** H=zscore(attributes)

**Output:**

k=H(:,1)

i=std(k)

i = 1.0000

j=h(:,2);

o=std(j)

o = 1.0000

d=H(:,3);

e=std(d)

e =1.0000

t=std(x)

t =1.0000

u=mean(k)

u =-1.4572e-15

i=mean(j)

i = -1.7225e-15

f=mean(d)

f = -2.0436e-15

c=mean(x)

c = -9.8440e-17

**Interpretation: As the standard deviation is equal to one and means is trending towards zero, hence the data is normalized.**