INDIAN INSTITUTE OF SPACE SCIENCE AND TECHNOLOGY THIRUVANANTHAPURAM

Assignment-III

Assignment #3 : MATLAB Implementation of Logistic Regression, Gaussian Discriminant Analsis, Naïve Bayes Classification & Short Notes on ANOVA, Students t distribution & its application

ASIF SALIM
SC14M063

MODELING OF DATA1

1. Logistic Regression

The program has done using Holdout technique by dividing the data into training and testing data set in the ratio 7:3

The MATLAB code used for the program

```
%%%%%%%% ASSIGNMENT #3 => MODELLING OF DATA1   %%%%%%%

%%%%%%%%%% LOGISTIC REGRESSION   %%%%%%%%%%%%%%%%%


%%%%%%%%%%%%%%% DATA UPLOAD AND INITIALISING VARIABLES AND VECTORS%%%%%%


data = load('D:\the world of machine learning\assignments\data mining\assignment_3\Data1.csv', 'v1');display(data);
[x, z]=size(data);
w=[0 1 0];
wnew=[0 0 0];
c_no=2;
sigmoid = zeros(1,100);
n = 10;     %initialising norm value to start while loop (arbitrary value > 1e-5)


%%%%%%%%%%%%%%%% LOOP FOR FINDING THE FITTEST PARAMETER VALUES %%%%%%%%%

C = cvpartition(data(:,3),'Holdout',0.3);
display(C);
holdoutdata = test(C);
display(holdoutdata);
inpdata = [ones(100,1) data(1:100,1) data(1:100,2)];
y= data (:,3);

while (n>1e-5)

[wnew]  = gradient( data,holdoutdata,w,inpdata,0.002); %function to update w values
n = norm ((wnew-w));    %calculating norm of previous and curent w values
w=wnew;

end
display('Value of the parameters');
display(w);
```

```
%%%%%%%%%%%%%%% PLOTTING THE LOGISTIC REGRESSION CURVE %%%%%%%%%%%%%%

[mat]=sigmoid_fn( inpdata,w );
figure(1);
plot(mat(:,1),mat(:,2) ,'-');     %plotting of cost function values

%%%%%%%%%%%%%%%%%%%% TESTING OF DATA STARTS%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

testdata = [ones(100,1) data(1:100,1) data(1:100,2)];
[result,pred,probability]=testing( data,holdoutdata,w,testdata );    %function to test the data

%%%%%%%%%%%%%%%%%% ASSESING THE PERFORMANCE OF THE SYSTEM %%%%%%%%%%%%%%

performance (result);   %function to assess performance
```

```
%%%%%%%%%%%%%%%%%%%%% TESTING OF DATA STARTS%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

testdata = [ones(100,1) data(1:100,1) data(1:100,2)];
[result,pred,probability]=testing( data,holdoutdata,w,testdata );    %function to test the data

%%%%%%%%%%%%%%%%%%% ASSESING THE PERFORMANCE OF THE SYSTEM %%%%%%%%%%%%%%%

performance (result);   %function to assess performance

%%%%%%% PLOTING THE DECISION BOUNDARY %%%%%%%%

[ positive,negative ] = classification(data);   %function to classif the data into positive and negative class
figure(2);
plotting(positive,negative );  %function to plot the positive and negative class members
x1 = data(:,1);
plot (-x1,-(w(1,1)+(w(1,2)*x1))/w(1,3));

%%%%%%%%%%% PLOTTING ROC %%%%%%%%%%%%%

figure(3);
rocp(pred,probability);   %function to plot ROC curve
```

Functions used in the program are:-

- gradient :- used to find gradient of the maximum likelihood function each time the parameter values are updated

```
%%%%%%%%%% CALCULATING THE GRADIENT FOR LOGISTIC REGRESSION %%%%%%%%%%
%%%%%%% Input arguments = data matrix, holdout data(matrix containing cv partition information), parameter
%%%%%%% matrix, input data with row of ones appended & learning parameter
%%%%%%% value
%%%%%%% Output arguments = updated parameter matrix


function [ wnew ] = gradient( data,trainingdata,w,inputdata,alpha)
[m,n]=size(data);
wnew=zeros(1,n);
for b = 1:n

    grad=0;     %initialising the variable to find gradient of cost function
for a=1:m  %loop for finding the gradient of cost function

    if(trainingdata(a,1)==0)
    data1=(inputdata(a,:));
    f = w*(transpose (data1));

    sigma = 1/(1+exp(-f));  %calculating the value of the sigmoid function
    grad= grad + (data(a,n)-sigma);
    end
end
   grad = grad*data1(1,b)/m;
   wnew(1,b)=w(1,b)+alpha*(grad);       %updating w values
end
end
```

- sigmoid_fn:- used to find sigmoid function values with the parameters

```
%%%%%%%%%%CALCULATION OF SIGMOID FUNCTION IN LOGISTIC REGRESSION %%%%%%%%%

%%%%%%% Input arguments = data matrix & parameter matrix
%%%%%%% Output arguments = logistic function values

function [mat ] = sigmoid_fn( ipdata,w )
[m,n]=size(ipdata);
 sigmoid=zeros(m,-1);
 att=zeros(m,1);
for a=1:m       %loop for finding the sigmoid function values for each row in the data set

    data1=(ipdata(a,:));   %taking the row of a particular data set
    b=(w*(transpose (data1)));
    att(a,1)=b;
    f = 1/(1+(exp(-b))); %calculating of f(x)
   sigmoid(a,1)=f;
end
mat=[att sigmoid];
mat= sortrows(mat,-1);
end
```

- testing :- used to test the data in logistic regression

```
%%%%%% TESTING OF LOGISTIC REGRESSION %%%%%%%%%%

%%%%%%% Input arguments = data matrix,  holdout data(matrix containing cv
%%%%%%% partition information), parameter matrix & data matrix to be tested
%%%%%%% Output arguments = result matrix, output matrix &its probabilities


function [result,testdataout,probability] = testing( data,holdoutdata,w,testdata )

[m,n]=size(data);

i=1;f=1;l=0;
for c=1:m
if holdoutdata(c,1)==1
     l=l+1;
end
end
 pred=zeros(l,1); %initialising the vector to store prdicted values
 probability = zeros(l,1);         %vector to store probabilities to plot roc
 testdataout = zeros(l,1);


for c=1:m
  flag=0;
  %variable for calculating the predicted values (w(transpose)*f(x)
if holdoutdata(c,1)==1
      testdataout(i,1)=data(c,n);

    for d=1:n
       flag = flag + (w(1,d)*(testdata(c,d)));
    end
       probability(f,1)= (1/(1+exp(-flag)));
       f=f+1;
            if (flag>0)
                 pred(i,1)=1;   %storing the predicted value
            else
                 pred(i,1)=0;
            end
                i=i+1;
end
end
display ('prediction');
result = [ testdataout pred ];
end
```

- performance :- function to find performance parameters of the classifiers

```
%%%%%%%%% ASSESING THE PERFORMANCE OF THE CLASSIFIER %%%%%%%%
%%%%%%% Input arguments = result matrix
%%%%%%% Output arguments = none

function [ ] = performance( data )
display(data);
confusion_matrix=zeros(2,2);
[m,n]=size(data);
P=0;N=0;
beta=2;

for i=1:m               % loop to find confusion mtrix elements
        if data(i,1)==1
            P=P+1;
        else
            N=N+1;
        end
        if data(i,1)==1 && data(i,1)==data(i,2)
        confusion_matrix(1,1)=confusion_matrix(1,1)+1;
        end
        if data(i,1)==1 && data(i,1)~=data(i,2)
        confusion_matrix(1,2)=confusion_matrix(1,2)+1;
        end

        if data(i,1)==0 && data(i,1)==data(i,2)
        confusion_matrix(2,2)=confusion_matrix(2,2)+1;
        end
        if data(i,1)==0 && data(i,1)~=data(i,2)
        confusion_matrix(2,1)=confusion_matrix(2,1)+1;
        end
end

TP=confusion_matrix(1,1);
FN=confusion_matrix(1,2);
FP=confusion_matrix(2,1);
TN=confusion_matrix(2,2);
```

```
accuracy = (TP+TN)/(P+N);
error_rate= (FP+FN)/(P+N);
tpr = TP/P;
tnr = TN/N;
precision = TP/(TP+FP);
f_score = (2*precision*tpr)/(precision+tpr);
f_beta = (precision*tpr)/(precision+tpr)*(1+(beta^2))/(beta^2);


display(accuracy);
display(error_rate);
display(tpr);
display(tnr);
display(precision);
display(f_score);
display(f_beta);

end
```

- classification:- function to divide the data set into positive and negative class

```
%%%% CLASSIFYING THE DATA INTO POSITIVE AND NAGATIVE CLASS %%%%%

%%%%%%% Input arguments = data(data)
%%%%%%% Output arguments = matrices classified into positive and negative
%%%%%%% classes

function [ positive,negative ] = classification(data)

[m,n]=size(data);
p=1;q=1;

for i=1:m
if data(i,n)==1
```

```
    for j=1:n
        positive (p,j)=data(i,j);     %creating the positive class
    end
    p=p+1;

else

    for j=1:n
        negative (q,j)=data(i,j);     %creating the negative class
    end
    q=q+1;

end
end
end
```

- plotting:- to plot positive and negative classes

```
%%%%%% PLOTTING THE VALUES %%%%%%
%%%%%%% Input arguments = matrices to be plotted
%%%%%%% Output arguments = none

function [] = plotting(positive,negative )

grid on
plot( positive(:,1), positive(:,2),'x');            %ploting of positive class
hold on
plot( negative(:,1), negative(:,2), 'or');          %ploting of negative class
hold on

end
```

- rocp:- to plot reciver operating characteristics (roc)

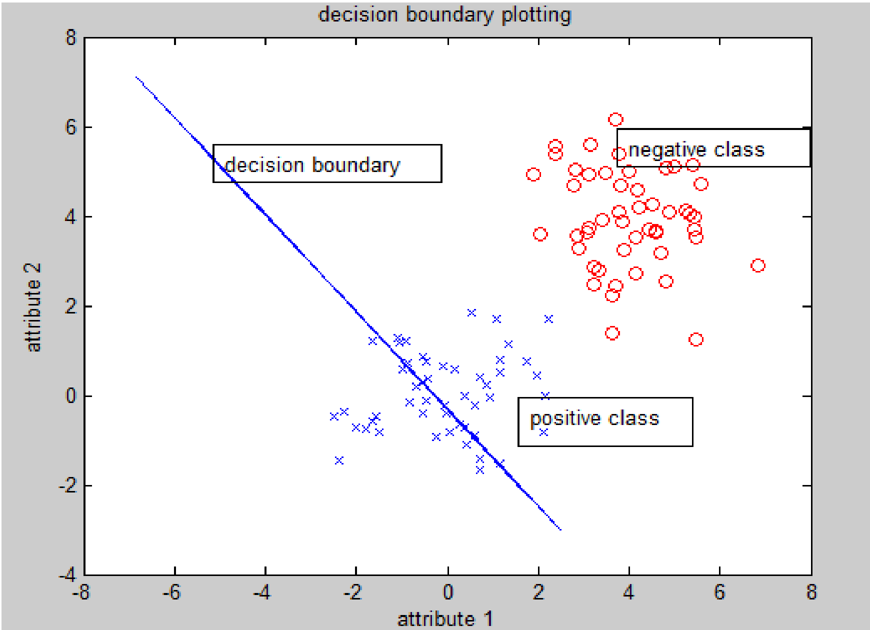
```
%%%%%%%% FUNCTION TO PLOT ROC CURVE %%%%%%%%%

%%%%%%%% Input arguments= actual output values of the test data set & their
%%%%%%%% predicted probabilities
%%%%%%%% Output arguments = none

function [ ] = rocp( predictedvalues, probability )

mat=[probability predictedvalues];
display(mat);
mat= sortrows(mat,1);  % sorting in ascending order
display(mat);

plotroc (mat(:,2)', mat(:,1)');
end
```

Results

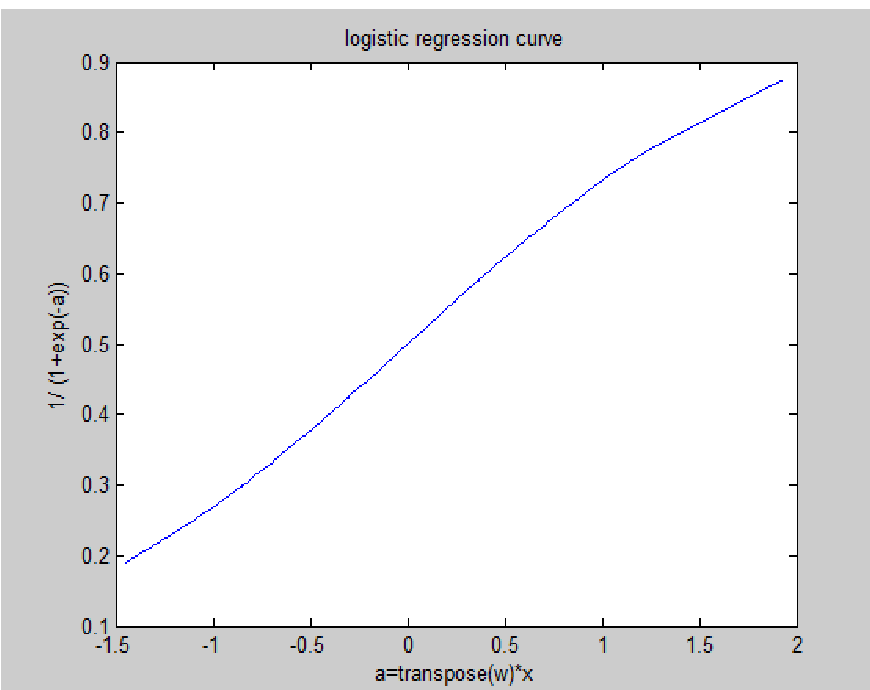
Values of parameters of the model

W1= -0.1343   W2= 0.4755   W3= -0.4380

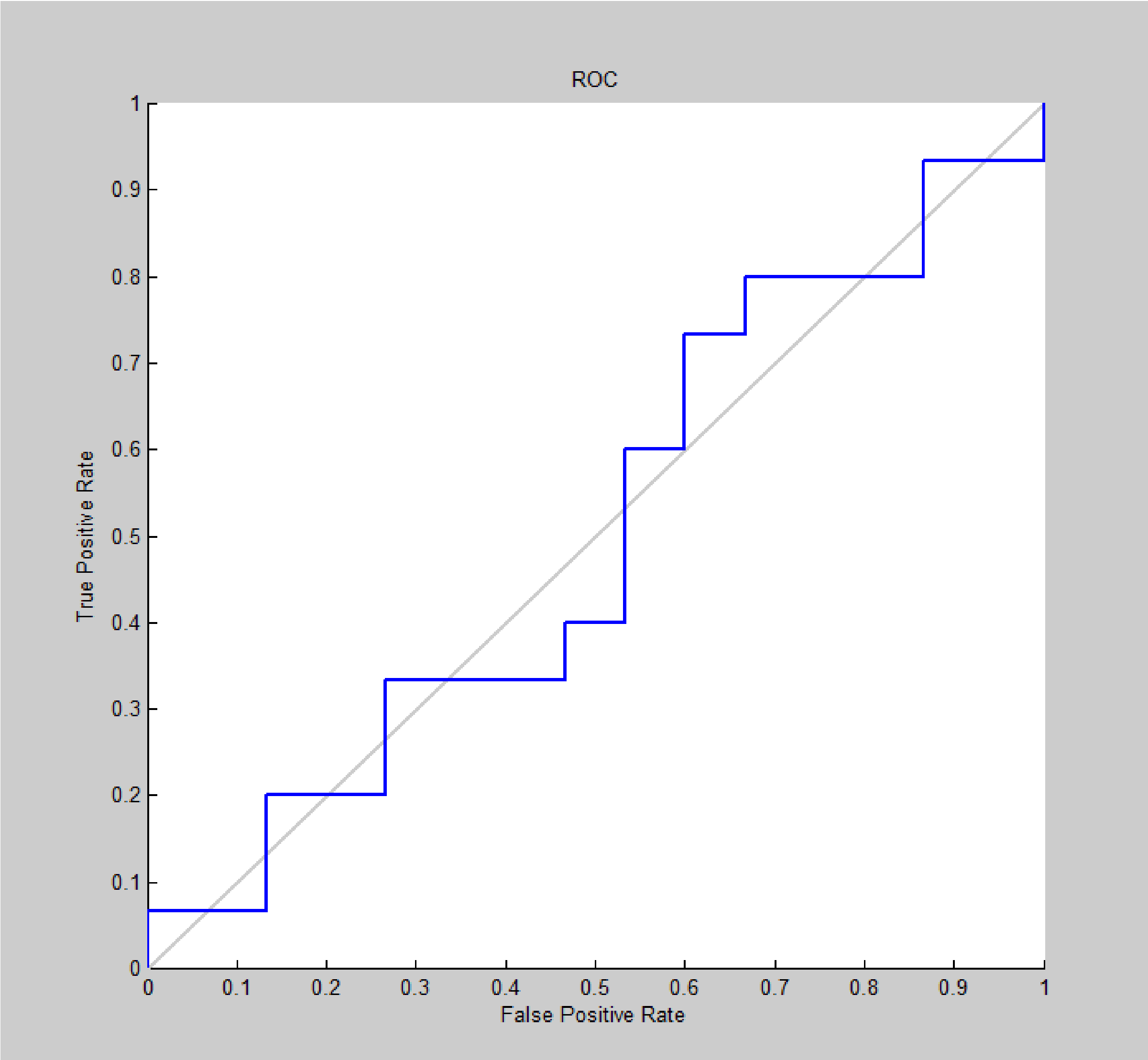
Decision Boundary Plot



Logistic Regression Curve



Receiver Operating Characteristic (ROC)



Performance Parameters

accuracy = 0.4333

error rate = 0.5667

true positive rate = 0.4000

true negative rate = 0.4667

precision = 0.4286

f_score = 0.4138

f_beta = 0.2586

2. Gaussian disrciminant analysis

The MATLAB code used for the program

```
%%%%%%%     ASSIGNMENT -->#3     %%%%%%%%%%%%%%%

%%%%%%    GAUSSIAN DISCRIMINANT ANALYSIS ON DATA1   %%%%%%%


%%%%%%%%%%%%%%% DATA UPLOAD AND INITIALISING VARIABLES AND VECTORS%%%%%%
clc;
data = load ('D:\the world of machine learning\assignments\data mining\assignment_3\Data1.csv');
c_no=2;
x=zeros(100,2);

%%%%%%   CLASSIFYING INTO POSITIVE AND NEGATIVE CLASS %%%%%%%

[ positive,negative] = classification(data);

%%%%%% PLOTTING THE VALUES %%%%%%
figure(1);
plotting(positive,negative );

%%%%%% FINDING MU1 AND MU2 AND COVARIANCE ON TEST DATA%%%%%%

mu1 = mean_vector( positive );
mu2 = mean_vector( negative );
display (mu1);
display (mu2);
covariance=cov([ data(:,1) data(:,2) ]);
sigma = std([ data(:,1) data(:,2) ]);
display(covariance);

%%%%%% TESTING THE MODEL  %%%%%%

testdata = [positive(36:50,1) positive(36:50,2);negative(36:50,1) negative(36:50,2)];
result1=zeros(30,1);
z=zeros(30,1);
for i = 1:30
y= (exp(((testdata(i,:)'-mu1)')/(covariance)*(testdata(i,:)'-mu1)))/exp(((testdata(i,:)'-mu2)')/(covariance)*(testdata(i,:)'-mu2));
z1=1/(1+y);
```

```
z(i,1)=z1;              %to store probability values for plotting roc
if z1>0.5
   result1(i,1)=1;
else
    result1(i,1)=0;
end
end
actual = [data(36:50,3); data(86:100,3)];
result = [data(36:50,3) result1(1:15,1) ; data(86:100,3) result1(16:30,1)];
display (result);

%%%%%%%%%%%%%%%%%% ASSESING THE PERFORMANCE OF THE SYSTEM %%%%%%%%%%%%%%
performance(result);

%%%%%%% PLOTING THE DECISION BOUNDARY %%%%%%%%

covin=inv(covariance);
a = (mu2'*covin*mu2) - (mu1'*covin*mu1);
b = covin(1,1)*(mu1(1,1)-mu2(1,1)) + covin(1,2)*(mu1(2,1)-mu2(2,1));
c = covin(2,1)*(mu1(1,1)-mu2(1,1)) + covin(2,2)*(mu1(2,1)-mu2(2,1));

for i=1:100
    x(i,1)=data(i,1);
    x(i,2)= (-a-(2*x(i,1)*b))/(2*c);
end
hold on; grid on
plot(x(:,1),x(:,2),'--y');

%%%%%%%%%%% PLOTTING ROC %%%%%%%%%%%%%
figure(2);
rocp(actual,z);
```

Assignment-III

```
%%%%%% DISPLAYING THE GAUSSIAN DISTRIBUTIONS %%%%%%%

figure(3);
gaussian_plot(mu1,sigma,-3,3);
gaussian_plot(mu2,sigma,-10,10);

%%%%%% DISPLAYING THE GAUSSIAN CONTOURS %%%%%%%

figure(4);
gaussian_contour(mu1,sigma,-13,13);
gaussian_contour(mu2,sigma,-13,13);
```

Functions used in the program are:-

- mean_vector :- to find mean of attribute columns separately

```
%%%%%% FINDING MU1 AND MU2 ON TEST DATA%%%%%%
%%%%%%% Input arguments = data matrix
%%%%%%% Output arguments = mean vector

function [ mu ] = mean_vector( data_matrix )
[m,n]=size(data_matrix);
mu=zeros(n-1,1);              %initialising mean vector
sum=0;
for j=1:n-1
for i=1:m
    sum=sum+data_matrix(i,j);   %finding the sum
end
mu(j,1)=sum/m;                %finding the mean
sum=0;
end
end
```

- gaussian_plot:- to plot multivariate gaussian density function

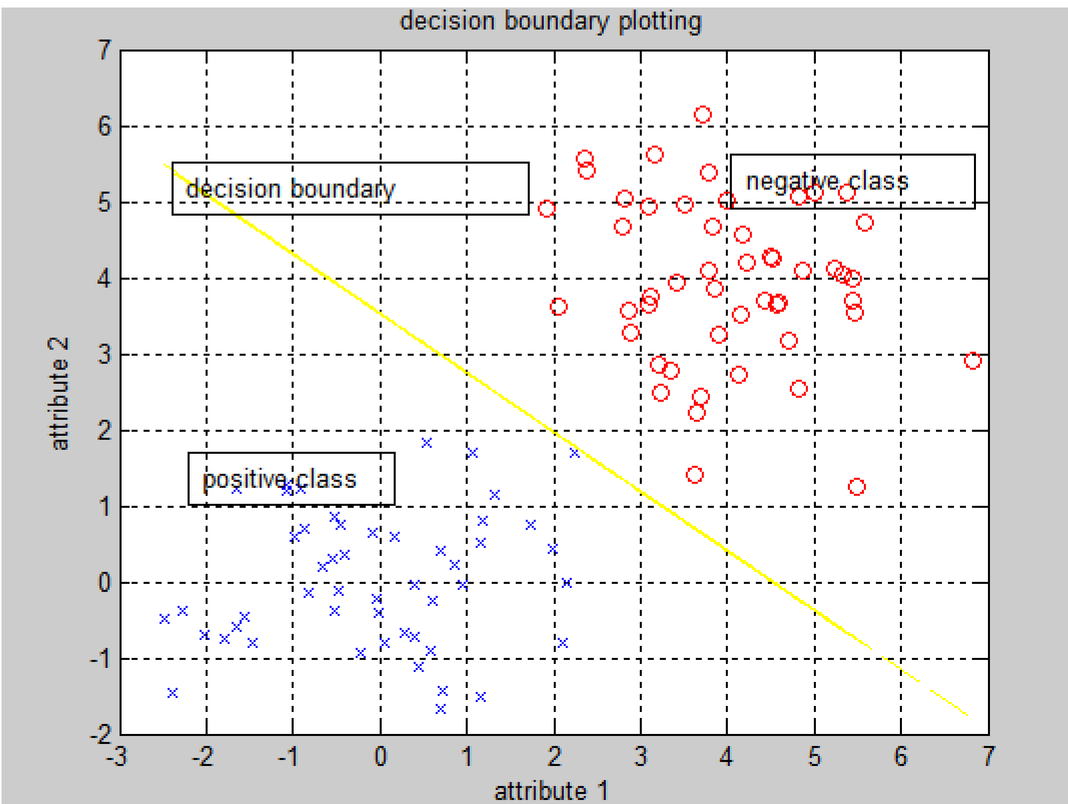
```
%%%%%% DISPLAYING THE GAUSSIAN DISTRIBUTION %%%%%%%
%%%%%%% Input arguments = mean vector, variance, range of values to be
%%%%%%% plotted (x,y)
%%%%%%% Output arguments = none

function [] = gaussian_plot(mu,sigma,x,y)

x1 = x:.2:y; x2 = x:.2:y;
[X1,X2] = meshgrid(x1,x2);
F = mvnpdf([X1(:) X2(:)],mu',sigma);  %to find pdf values
F = reshape(F,length(x2),length(x1)); %to reshape the array
surf(x1,x2,F);hold on                  %to make the surface plot
caxis([min(F(:))-.5*range(F(:)),max(F(:))]);   %seudocolor axis scaling
axis([-3 3 -3 3 0 .1])  %defining the range of axis for the plot
xlabel('x1'); ylabel('x2'); zlabel('Probability Density');
colorbar

end
```

- gaussian_contour:- to plot the gaussian contour

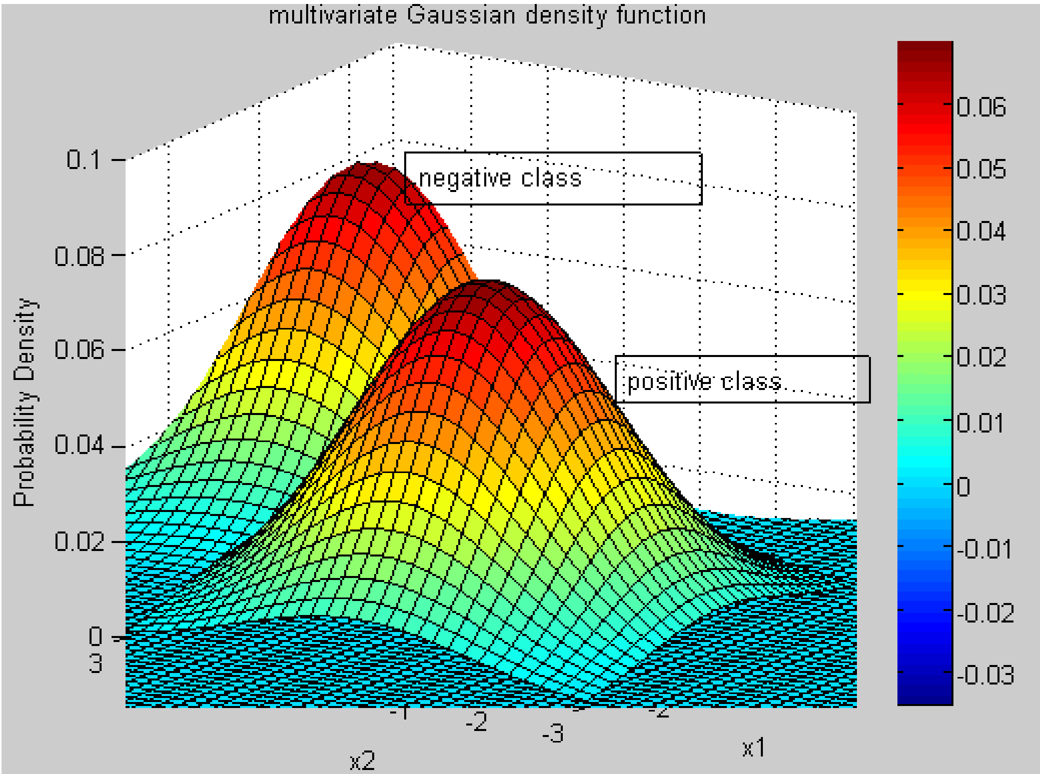
```
%%%%%% DISPLAYING THE GAUSSIAN CONTOURS %%%%%%%
%%%%%%% Input arguments = mean vector, variance, range of values to be
%%%%%%% plotted (x,y)
%%%%%%% Output arguments = none

function [] = gaussian_contour( mu,sigma,x,y )

x1 = x:.2:y; x2 = x:.2:y;
[X1,X2] = meshgrid(x1,x2);
F = mvnpdf([X1(:) X2(:)],mu',sigma);     %to find pdf values
F = reshape(F,length(x2),length(x1));     %to reshape the array
mvncdf([0 0],[1 1],mu',sigma);          %to find cdf values
contour(x1,x2,F,[.0001 .001 .01 .05:.1:.95 .99 .999 .9999]);   %to make contour plot
hold on
xlabel('x'); ylabel('y');
line([0 0 1 1 0],[1 0 0 1 1],'linestyle','--','color','k');     %to create a line


end
```

Results

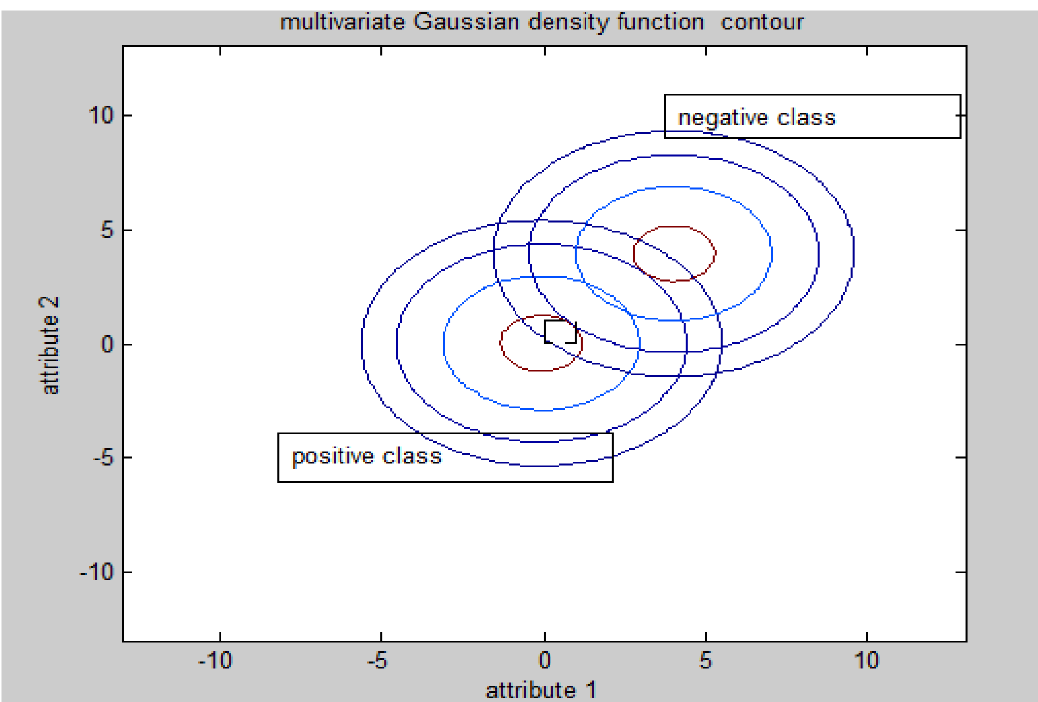
<u>**Decision Boundary Plot**</u>



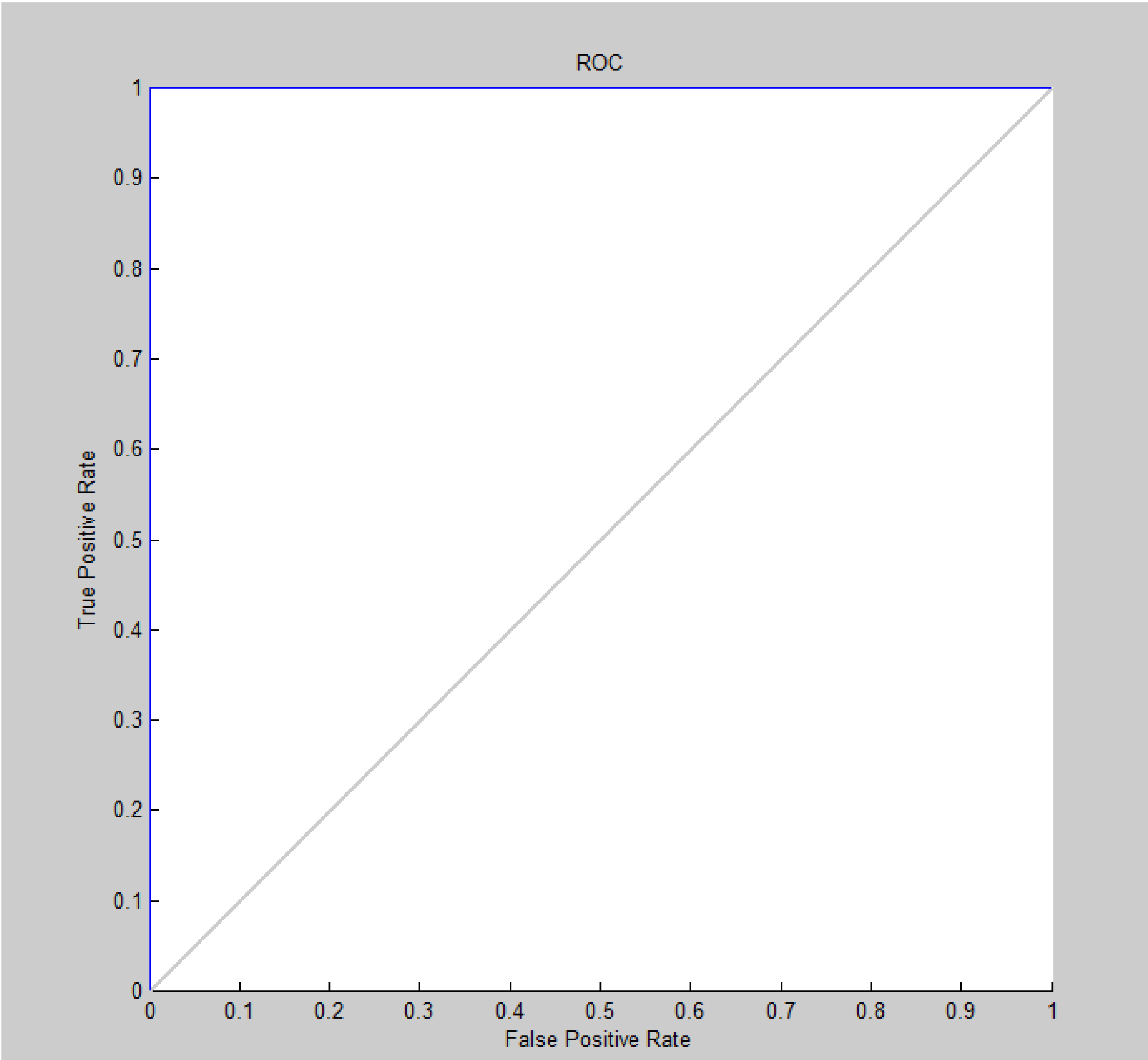
Plot of multivariate Gaussian density function



Plot of multivariate Gaussian density function contour



Receiver Operating Characteristic (ROC)



Performance Parameters

accuracy = 1

error rate = 0

true positive rate = 1

true negative rate = 1

precision = 1

f_score = 1

f_beta = 0.6250

3. Naïve Bayes Analysis

The program has done using Holdout technique by dividing the data into training and testing data set in the ratio 7:3

The MATLAB code used for the program

```
%%%%%%%     ASSIGNMENT -->#3     %%%%%%%%%%%%%%%

%%%%%%    NAIVE BAYES ANALYSIS ON DATA1   %%%%%%%


%%%%%%%%%%%%%%%% DATA UPLOAD AND INITIALISING VARIABLES AND VECTORS%%%%%%
%clc;
data = load ('D:\the world of machine learning\assignments\data mining\assignment_3\Data1.csv');
[m,n]=size(data);
c_no=5;    %no of discretising levels
%%%%%%%%%%   DISCRETISING THE DATA SET %%%%%%%%%

for i=1:n-1
    data(:,i)=discretizing(data(:,i),c_no);
end
display(data);

%%%%%%   CLASSIFYING INTO POSITIVE AND NEGATIVE CLASS %%%%%%%

[ positive,negative]=classification(data);


%%%%%%% CLASSIFYING INTO TRAINING DATA AND TESTING DATA %%%%%%%%%%%

[positive_tr,positive_tst] = nba_training( positive );
[a, b]=size(positive_tr);
[negative_tr,negative_tst] = nba_training( negative );
[c ,d]=size(negative_tr);
display(positive_tst);

%%%%%%% FINDING THE CONDITIONAL PROBABILITIES FOR POSITIVE & NEGATIVE CLASS AND THEIR TOTAL PROBABILITY %%%%%%%%%

[prob_pos]=cond_prob(positive_tr,5);
[prob_neg]=cond_prob(negative_tr,5);
ppos=a/a+c;
pneg=c/a+c;
```

```
%%%%%%%     LAPLACE SMOOTHING       %%%%%%%%%

prob_pos=laplace_smooth(prob_pos,a);
prob_neg=laplace_smooth(prob_neg,c);

display(prob_pos);
display(prob_neg);

%%%%%%%%%% TESTING OF THE DATA STARTS %%%%%%%%%%

p1=nba_testing( positive_tst,prob_pos,ppos,c_no);
p2=nba_testing( positive_tst,prob_neg,pneg,c_no);
p3=nba_testing( negative_tst,prob_pos,ppos,c_no);
p4=nba_testing( negative_tst,prob_neg,pneg,c_no);
[g, f]=size(p1);
pred1 = zeros(g,1);
pred2 = zeros(g,1);
p5=p1;
prob1= zeros(g,1);
prob2= zeros(g,1);
for i=1:g       %to take the decision corresponding to test data in positive class

        if p5(i,1)>=p2(i,1)
            pred1(i,1)=1;
            prob1(i,1)=p5(i,1);         %storing probability values for plotting roc
        else
            pred1(i,1)=0;
            prob1(i,1)=p2(i,1);         %storing probability values for plotting roc
        end
end

for i=1:g           %to take the decision corresponding to test data in negative class

        if(p3(i,1)>=p4(i,1))
            pred2(i,1)=1;
            prob2(i,1)=p3(i,1);         %storing probability values for plotting roc
        else
```

```
            pred2(i,1)=0;
            prob2(i,1)=p4(i,1);          %storing probability values for plotting roc
        end
end
probability=[prob1' prob2'];
prediction = [pred1' pred2'];
display('prediction...');
actual=[positive_tst(:,3)' negative_tst(:,3)'];
result=[actual;prediction ];
display(result);

%%%%%%%%%%%%%%%%%% ASSESING THE PERFORMANCE OF THE SYSTEM %%%%%%%%%%%%%%
performance(result');


%%%%%%%%%% PLOTTING ROC %%%%%%%%%%%%

rocp(actual',probability');
```

Functions used in the program are:-

- discretizing :- used to discretise the real valued functions in the data set

```
%%%%%%%%%   DISCRETISING THE DATA SET %%%%%%%%%

%%%%%%% Input arguments = individual attribute columns of the data set and
%%%%%%% no:of levels they want to be discretized.
%%%%%%% Output arguments = discretized data


function [ data ] = discretizing( data,c_no )

[m,n]=size(data);
maximum = max(max(data));
minimum = min(min(data));

maxi = round(maximum);                 %rounding the maximum value to the nearest integer
mini = round(minimum);                 %rounding the minimum value to the nearest integer
stepsize = abs(maxi-mini)/c_no;        %finding the stepsize between discrete levels

for i=1:m                              %discretizing process starts
if data(i,1)>=minimum && data(i,1)<(minimum+stepsize)
data(i,1)=0;
end

if data(i,1)>=(minimum+stepsize) && data(i,1)<(minimum+(2*stepsize))
data(i,1)=1;
end

if data(i,1)>=(minimum+(2*stepsize)) && data(i,1)<(minimum+(3*stepsize))
data(i,1)=2;
end

if data(i,1)>=(minimum+(3*stepsize)) && data(i,1)<(minimum+(4*stepsize))
data(i,1)=3;
end

if data(i,1)>=(minimum+(4*stepsize)) && data(i,1)<=maximum
data(i,1)=4;
end
end
end
```

- nba_training:- to classify the given data set into training and testing data set.

```
%%%%% CREATING THE TRAINING DATA SET AND TESTING DATA SET FOR NAIVE BAYES ANALYSIS %%%%%%%%
%%%%%%% Input arguments = classified data set
%%%%%%% Output arguments = training data set

function [ data_tr,data_tst ] = nba_training( data )
[m,n]=size(data);
j=1;k=1;
C = cvpartition(data(:,n-1),'Holdout',0.3);
holdoutdata = test(C);

for i=1:m     %loop to classify into training and testing data set
    if(holdoutdata(i,1)==0)
        data_tr(j,:)= data(i,:);
        j=j+1;
    else
        data_tst(k,:)= data(i,:);
        k=k+1;
    end
end
```

- cond_prob :- to find the conditional probabilities of attributes corresponding to positive and negative class

```
%%%%%%% FINDING THE CONDITIONAL PROBABILITIES FOR CLASSES %%%%%%%%%

%%%%%%% Input arguments = classified data sets into different classes and
%%%%%%% no of discrete output levels
%%%%%%% Output arguments = probability matrix of different attributes

function [ prob ] = cond_prob( data,c_no )

[m,n]=size(data);
sum=zeros(1,c_no);
prob=zeros(n-1,c_no);

for j=1:n-1

    for i=1:m              %loop for finding number of occurences of individual attribute members

        if data(i,j)==0
            sum(1,1)=sum(1,1)+1;
        elseif data(i,j)==1
            sum(1,2)=sum(1,2)+1;
        elseif data(i,j)==2
            sum(1,3)=sum(1,3)+1;
        elseif data(i,j)==3
            sum(1,4)=sum(1,4)+1;
        else data(i,j)==4;
            sum(1,5)=sum(1,5)+1;
        end
    end
    for l=1:c_no
    prob(j,l)=sum(1,l)/m;   %probability corresponding to individual attribute members
    end
    sum=zeros(1,5);
end
end
```

Assignment-III

- laplace_smooth:- to perform laplace smoothing on the data set

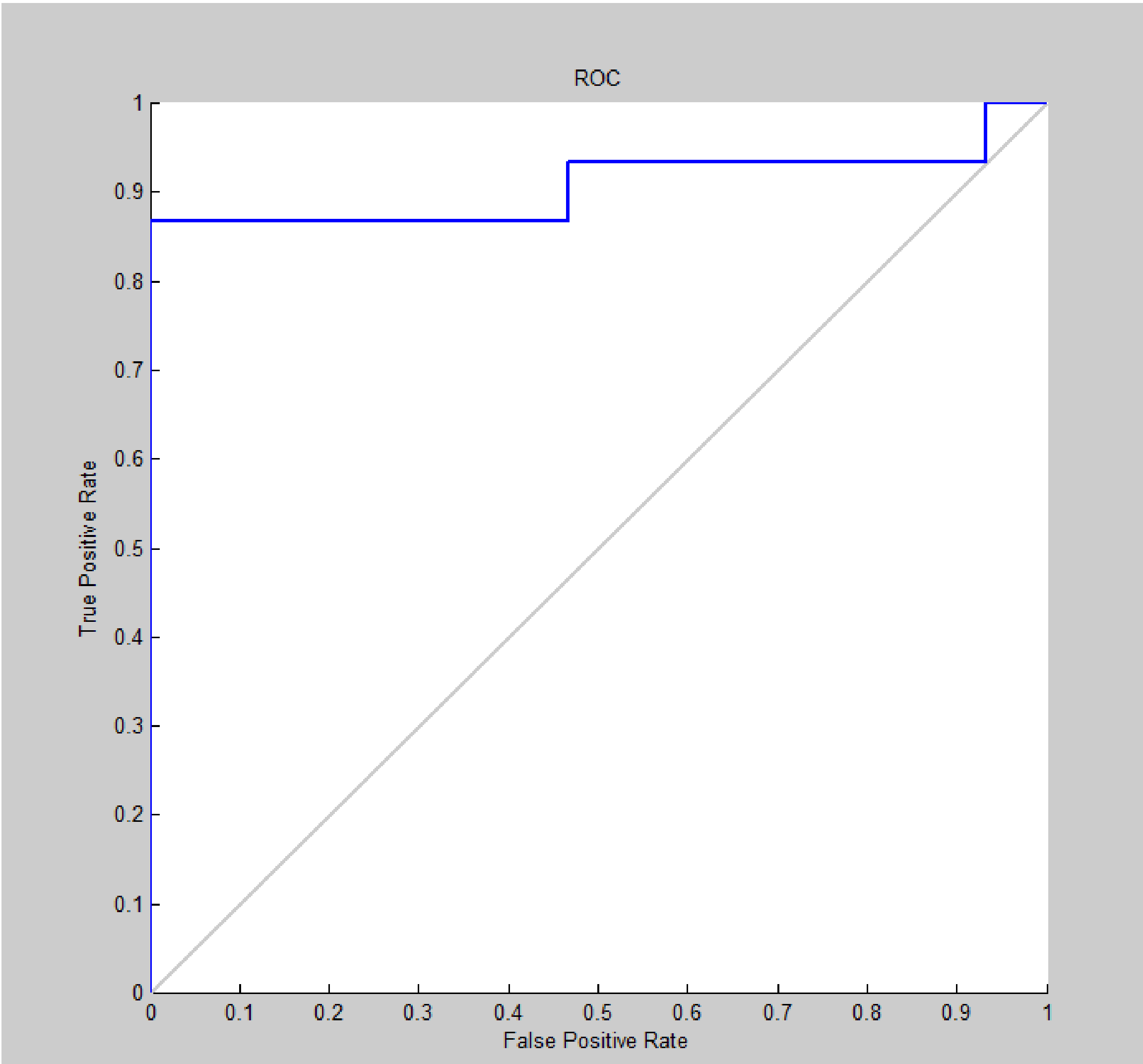
```
%%%%%%%     LAPLACE SMOOTHING       %%%%%%%%%
%%%%%%% Input arguments = probability matrix and size of the corresponding
%%%%%%% classified matrix
%%%%%%% Output argument = laplace smoothed probability matrix

function [ data ] = laplace_smooth( data,a )
z=0;
[m,n]=size(data);
 for i=1:n                  %to identify the zero elements in the matrix
        for j=1:m
            if data(j,i)==0
                z=z+1;
            end
        end
 end
 for i=1:n                  %smoothing operation starts
        for j=1:m
            if data(j,i)==0
                data(j,i)= 1/(a+z);
            else
             data(j,i)=  data(j,i)*a/(a+z);
            end
        end
 end
end
```

- nba_testing:- to perform testing in naïve bayes analysis

```
%%%%%%%%%% NAIVE BAYES ANALYSIS TESTING %%%%%%%%%%%%%%%%%%
%%%%%%% Input arguments = data matrix to be tested,probability
%%%%%%% matrix,corresponding class probability and no:of disrete levels
%%%%%%% Output arguments = calculated probability


function [ p ] = nba_testing( testdata,prob_matrix,prob,c_no)

[m,n]=size(testdata);
p=ones(m,1);
for i=1:m  %loop to find the product of the conditional probabilities corresponding to each data set
    for j=1:n-1
        for k=1:c_no
            if (testdata(i,j)==(k-1) && j==1)
            p(i,1)=p(i,1)*prob_matrix(1,k);
            end
            if (testdata(i,j)==(k-1) && j==2)
            p(i,1)=p(i,1)*prob_matrix(2,k);
            end
        end
    end
     p(i,1)=p(i,1)*prob;
end
display(p);
end
```

Results

Receiver Operating Characteristic (ROC)



Performance Parameters

accuracy = 0.9333

error rate = 0.0667

true positive rate = 0.9333

true negative rate = 0.9333

precision = 0.9333

f_score = 0.9333

f_beta = 0.5833

MODELING OF HABERMAN'S SURVIVAL DATA SET

1. Logistic Regression

The program has done using Holdout technique by dividing the data into training and testing data set in the ratio 7:3

The MATLAB code used for the program

```
%%%%%%%% ASSIGNMENT #3 => MODELLING OF HABERMAN'S SURVIVAL DATA   %%%%%%%

%%%%%%%%%% LOGISTIC REGRESSION   %%%%%%%%%%%%%%%%%


%%%%%%%%%%%%%%% DATA UPLOAD AND INITIALISING VARIABLES AND VECTORS%%%%%%


data = load('D:\the world of machine learning\assignments\data mining\assignment_3\haberman.txt', 'v1');display(data);
[x, z]=size(data);
w=[0 1 1 0];
wnew=[0 0 0 0];
parameters=zeros(6,4);
c_no=3;
sigmoid = zeros(1,306);
m=zeros(1,300);
p=1;
n = 1000;     %initialising norm value to start while loop (arbitrary value > 1e-5)

for i=1:x
    if data(i,4)==2
        data(i,4)=0;
    end
end

%%%%%%%%%%%%%%%% MINIMIZING PROCESS OF COST FUNCTION STARTS %%%%%%%%%

C = cvpartition(data(:,4),'Holdout',0.3);
display(C);
kfolddata = test(C);
inpdata = [ones(306,1) data(1:306,1) data(1:306,2) data(1:306,3)];
y= data (:,4);
while (n>1e-5)

wnew  = gradient( data,kfolddata,w,inpdata,0.001);
n = norm ((wnew-w));    %calculating norm of previous and curent w values
w=wnew;
end

%%%%%%%%%%%%%%% PLOTTING THE LOGISTIC REGRESSION CURVE %%%%%%%%%%%%%%

[mat]=sigmoid_fn( inpdata,w );
figure(1);
plot(mat(:,1),mat(:,2) ,'-');     %plotting of cost function values

%%%%%%%%%%%%%%%%%%%% TESTING OF DATA STARTS%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

testdata = [ones(306,1) data(1:306,1) data(1:306,2) data(1:306,3)];
[result,pred,probability]=testing( data,kfolddata,w,testdata );
display(result);

%%%%%%%%%%%%%%%%%% ASSESING THE PERFORMANCE OF THE SYSTEM %%%%%%%%%%%%%%
performance(result);

%%%%%%% PLOTING THE DECISION BOUNDARY %%%%%%%%

[ positive,negative ] = classification(data);
x1=[54 67 46];          %an extreme point in the data set to draw decision boundary (plane)
y1=[83 58 2];           %an extreme point in the data set to draw decision boundary (plane)
figure(2);
scatter3(positive(:,1),positive(:,2),positive(:,3),10);
hold on
scatter3(negative(:,1),negative(:,2),negative(:,3),20);
grid on
```

```
plane_log (w,x1',y1');
display(w);

%%%%%%%%%% PLOTTING ROC %%%%%%%%%%%%

figure(3);
rocp(pred,probability);
```

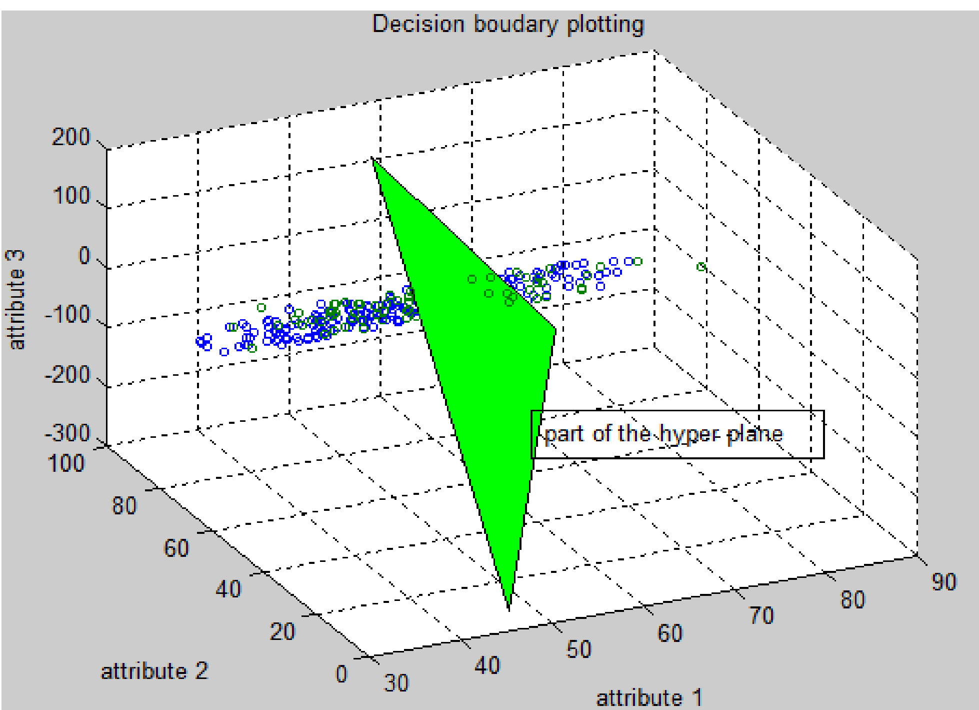
Functions used in the program are:-

- plane_log:- to plot the decision boundary for logistic regression(plotting only a part of the hyper plane)

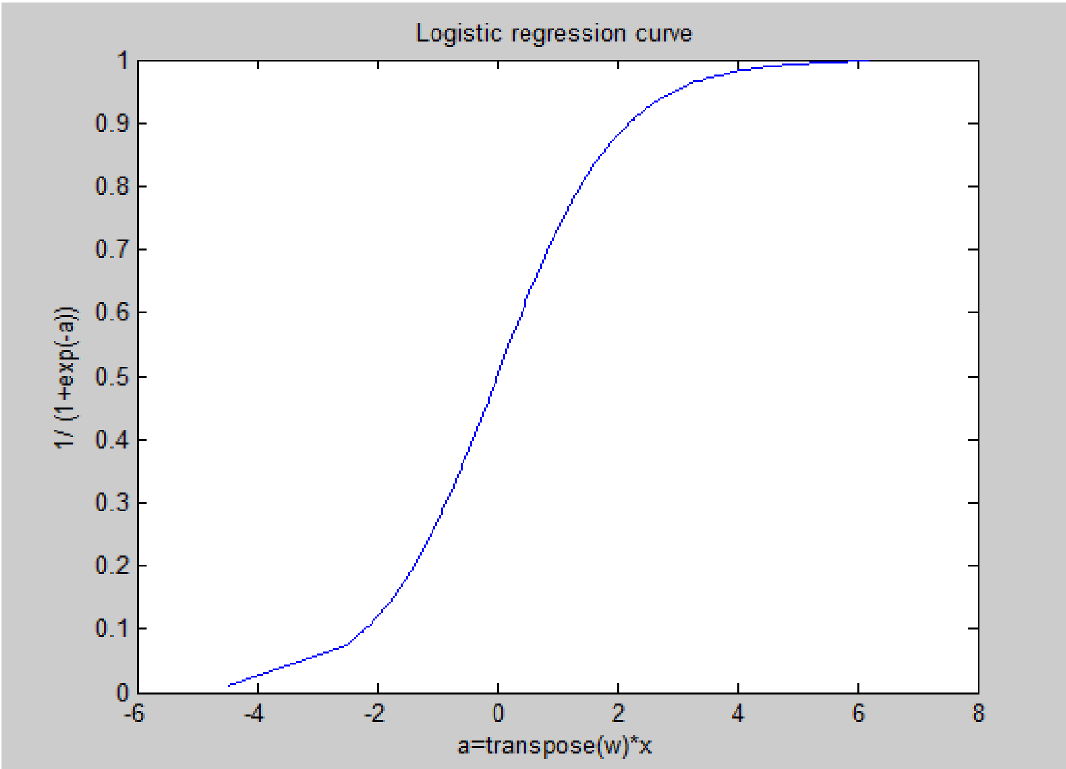
```
%%%%%%%%%%%%% function to plot the decision boundary (a portion of the plane)
%%%%%%%%%%%%% Input arguments= parameter matrix & 2 points in the data set
%%%%%%%%%%%%% Output arguments=none


function [] = plane_log( w,x,y )
z=zeros(3,1);
for i=1:3  %to find the third point to plot the plane
    z(i,1)=-(w(1,1)+(w(1,2)*x(i,1))+(w(1,3)*y(i,1)))/w(1,4);
end

patch(x',y',z','green');%%%%% plotting the plane
view(3);
end
```

Results

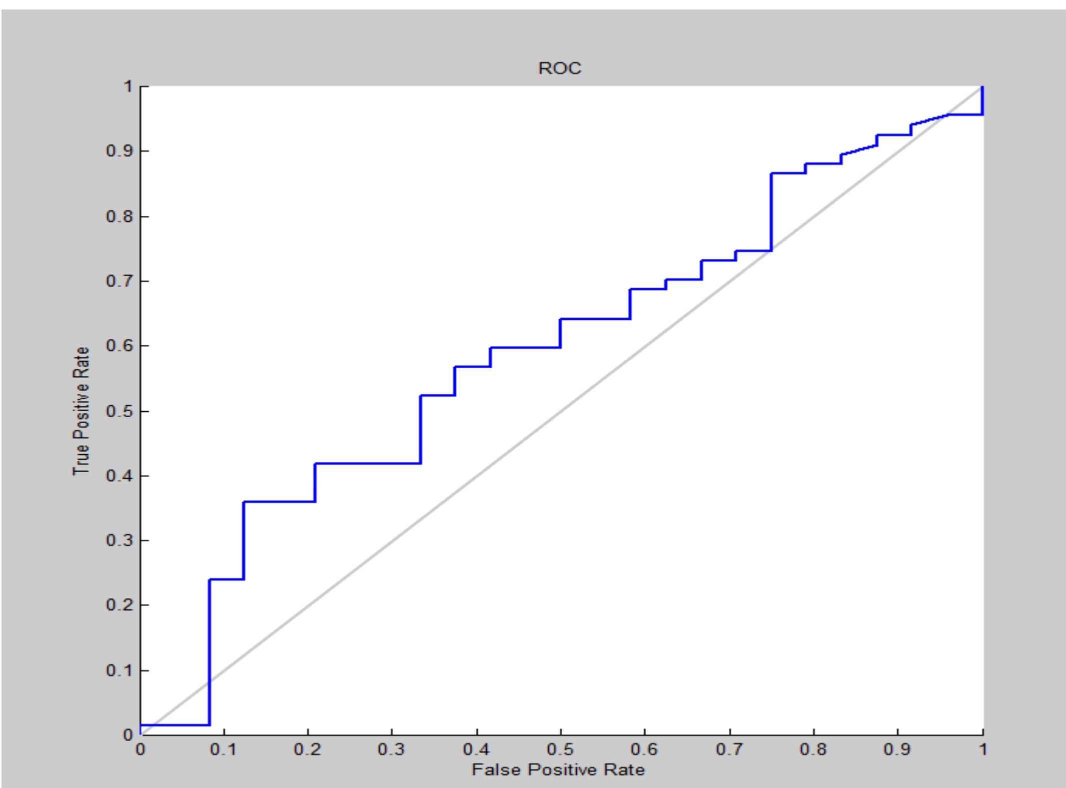
Decision Boundary Plot



Logistic Regression Curve



Receiver Operating Characteristic (ROC)



Values of parameters of the model

W1= -0.0142   W2= -0.1774   W3= 0.1772 W4= -0.0284

Performance Parameters

accuracy = 0.6703

error rate = 0.3297

true positive rate = 0.8209

true negative rate = 0.2500

precision = 0.7534

f_score = 0.7857

f_beta = 0.4911

2. Gaussian disrciminant analysis

The MATLAB code used for the program

```
%%%%%%%     ASSIGNMENT -->#3     %%%%%%%%%%%%%%%

%%%%%%    GAUSSIAN DISCRIMINANT ANALYSIS ON HABERMAN'S SURVIVAL DATA   %%%%%%%


%%%%%%%%%%%%%%% DATA UPLOAD AND INITIALISING VARIABLES AND VECTORS%%%%%%
clc;
data = load('D:\the world of machine learning\assignments\data mining\assignment_3\haberman.txt', 'v1');
[m,n]=size(data);
c_no=n-1;
x=zeros(m,3);

for i=1:m
    if data(i,4)==2
        data(i,4)=0;
    end
end
%%%%%%   CLASSIFYING INTO POSITIVE AND NEGATIVE CLASS %%%%%%%

[positive,negative] = classification(data);
[m1,n1]=size(positive);

%%%%%% FINDING MU1 AND MU2 AND COVARIANCE ON TEST DATA%%%%%%

mu1 = mean_vector( positive );
mu2 = mean_vector( negative );

display (mu1);
display (mu2);
covariance=cov([ data(:,1) data(:,2) data(:,3)]);
sigma = std([ data(:,1) data(:,2) data(:,3)]);
display(covariance);
```

```
%%%%%% TESTING THE MODEL  %%%%%%

testdata = [positive(158:225,1) positive(158:225,2) positive(158:225,3);negative(57:81,1) negative(57:81,2) negative(57:81,3)];
[m2,n2]=size(testdata);
display(m2);
result1=zeros(m2,1);
z=zeros(m2,1);
for i = 1:m2
y= (exp(((testdata(i,:)'-mu1)')/(covariance)*(testdata(i,:)'-mu1)))/exp(((testdata(i,:)'-mu2)')/(covariance)*(testdata(i,:)'-mu2));
z1=1/(1+y);
z(i,1)=z1;              %to store probability values for plotting roc
if z1>0.5
   result1(i,1)=1;
else
    result1(i,1)=0;
end
end
actual = [positive(158:225,4); negative(57:81,4)];
result = [positive(158:225,4) result1(1:68,1) ; negative(57:81,4) result1(69:93,1)];
display (result);


%%%%%%%%%%%%%%%%%% ASSESING THE PERFORMANCE OF THE SYSTEM %%%%%%%%%%%%%%
performance(result);

%%%%%%% PLOTING THE DECISION BOUNDARY %%%%%%%%

covin=inv(covariance);
figure(1);
scatter3(positive(:,1),positive(:,2),positive(:,3),10);
hold on
scatter3(negative(:,1),negative(:,2),negative(:,3),20);
grid on
u=[54 67 46]; v=[46 62 5];
plane_gda (covariance,mu1,mu2,u',v')
```

```
%%%%%%%%%% PLOTTING ROC %%%%%%%%%%%%
figure(2);
rocp(actual,z);
```

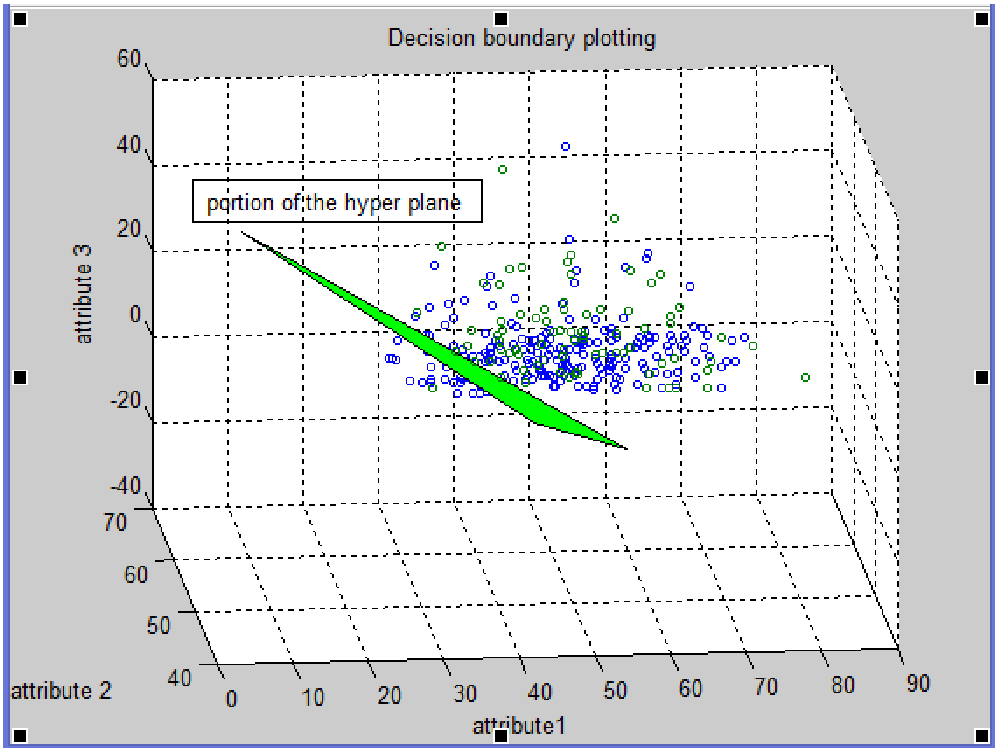
Functions used in the program are:-

- plane_gda:- to plot the decision boundary for logistic regression(plotting only a part of the hyper plane)

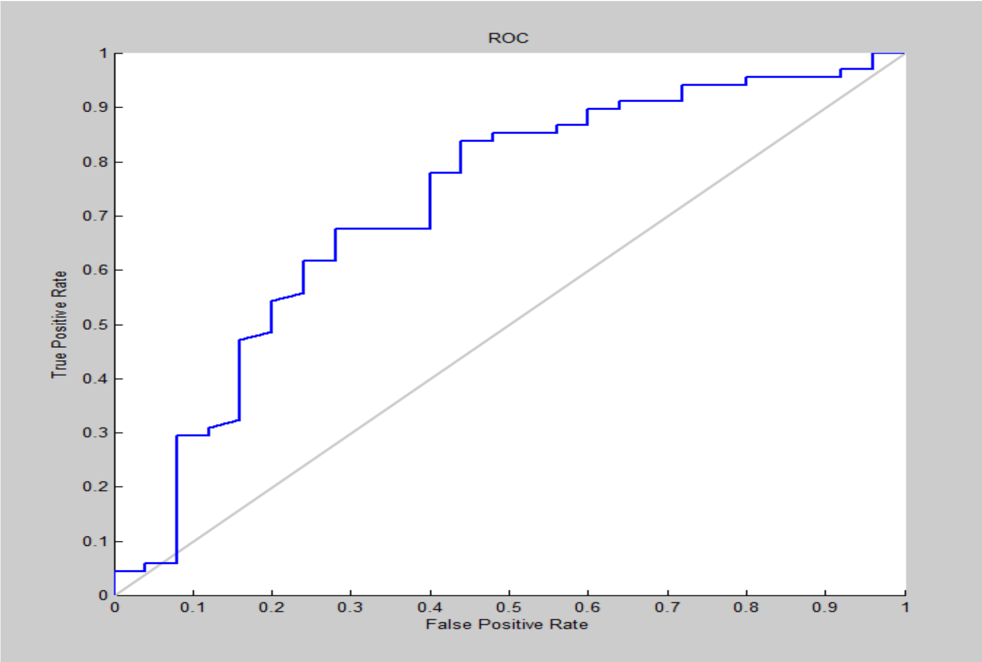
```
%%%%%%%%%%%%% function to plot the decision boundary (a portion of the plane)
%%%%%%%%%%%%% Input arguments= parameter matrix & 2 points in the data set
%%%%%%%%%%%%% Output arguments=none

function [] = plane_gda( var,m1,m2,u,v )
z=zeros(3,1);
m=m1-m2;
A=(m2'*var*m2)-(m1'*var*m1);

a=var(1,1);b=var(1,2);c=var(1,3);d=var(2,1);e=var(2,2);f=var(2,3);g=var(3,1);h=var(3,2);i=var(3,3);
j=m(1,1);k=m(2,1);l=m(3,1);

for o=1:3     %to find the third point to plot the plane
    z(o,1)= -( (2*u(o,1)*((j*d)+(k*e)+(l*f))) + (2*v(o,1)*((j*g)+(k*h)+(l*i))) + A) / (2*((j*a)+(k*b)+(l*c)));
end
patch(v',u',z','green');%%%%% plotting the plane
view(3);
end
```

Results

Decision Boundary Plot



Receiver Operating Characteristic (ROC)



Performance Parameters

accuracy = 0.7312

error rate = 0.2688

true positive rate = 0.7941

true negative rate = 0.5600

precision = 0.8308

f_score = 0.8120

f_beta = 0.5075

3. Naïve Bayes Analysis

The program has done using Holdout technique by dividing the data into training and testing data set in the ratio 7:3

The MATLAB code used for the program

```
%%%%%%%     ASSIGNMENT -->#3     %%%%%%%%%%%%%%%

%%%%%%    NAIVE BAYES ANALYSIS ON HABERMAN'S SURVIVAL DATA   %%%%%%%


%%%%%%%%%%%%%%% DATA UPLOAD AND INITIALISING VARIABLES AND VECTORS%%%%%%
%clc;
data = load ('D:\the world of machine learning\assignments\data mining\assignment_3\haberman.txt');
[m,n]=size(data);
c_no=5;  %no of disretising levels

for i=1:m
    if data(i,4)==2
        data(i,4)=0;
    end
end
%%%%%%%%%   DISCRETISING THE DATA SET %%%%%%%%%

for i=1:n-1
    data(:,i)=discretizing(data(:,i),c_no);
end

%%%%%%   CLASSIFYING INTO POSITIVE AND NEGATIVE CLASS %%%%%%%

[ positive,negative]=classification(data);

%%%%%%% CLASSIFYING INTO TRAINING DATA AND TESTING DATA %%%%%%%%%%

[positive_tr,positive_tst] = nba_training( positive );
[a, b]=size(positive_tr);
[negative_tr,negative_tst] = nba_training( negative );
[c ,d]=size(negative_tr);
display(positive_tst);
```

```
%%%%%%% FINDING THE CONDITIONAL PROBABILITIES FOR POSITIVE & NEGATIVE CLASS AND THEIR TOTAL PROBABILITY %%%%%%%%%

[prob_pos]=cond_prob(positive_tr,5);
[prob_neg]=cond_prob(negative_tr,5);
display(prob_pos);
ppos=a/(a+c);
pneg=c/(a+c);
```

```
%%%%%%%     LAPLACE SMOOTHING       %%%%%%%%%

prob_pos=laplace_smooth(prob_pos,a);
prob_neg=laplace_smooth(prob_neg,c);

display(prob_pos);
display(prob_neg);

%%%%%%%%% TESTING OF THE DATA STARTS %%%%%%%%%%

p1=nba_testing( positive_tst,prob_pos,ppos,c_no);
p2=nba_testing( positive_tst,prob_neg,pneg,c_no);
p3=nba_testing( negative_tst,prob_pos,ppos,c_no);
p4=nba_testing( negative_tst,prob_neg,pneg,c_no);

[g, f]=size(p1);
[g1, f1]=size(p3);
pred1 = zeros(g,1);
pred2 = zeros(g1,1);
p5=p1;
prob1= zeros(g,1);
prob2= zeros(g1,1);
for i=1:g

        if p5(i,1)>=p2(i,1)             %testing of the positive class
            pred1(i,1)=1;
            prob1(i,1)=p5(i,1);         %storing probability values for plotting roc
        else
            pred1(i,1)=0;
            prob1(i,1)=p2(i,1);         %storing probability values for plotting roc
        end
end

for i=1:g1                              %testing of the positive class
```

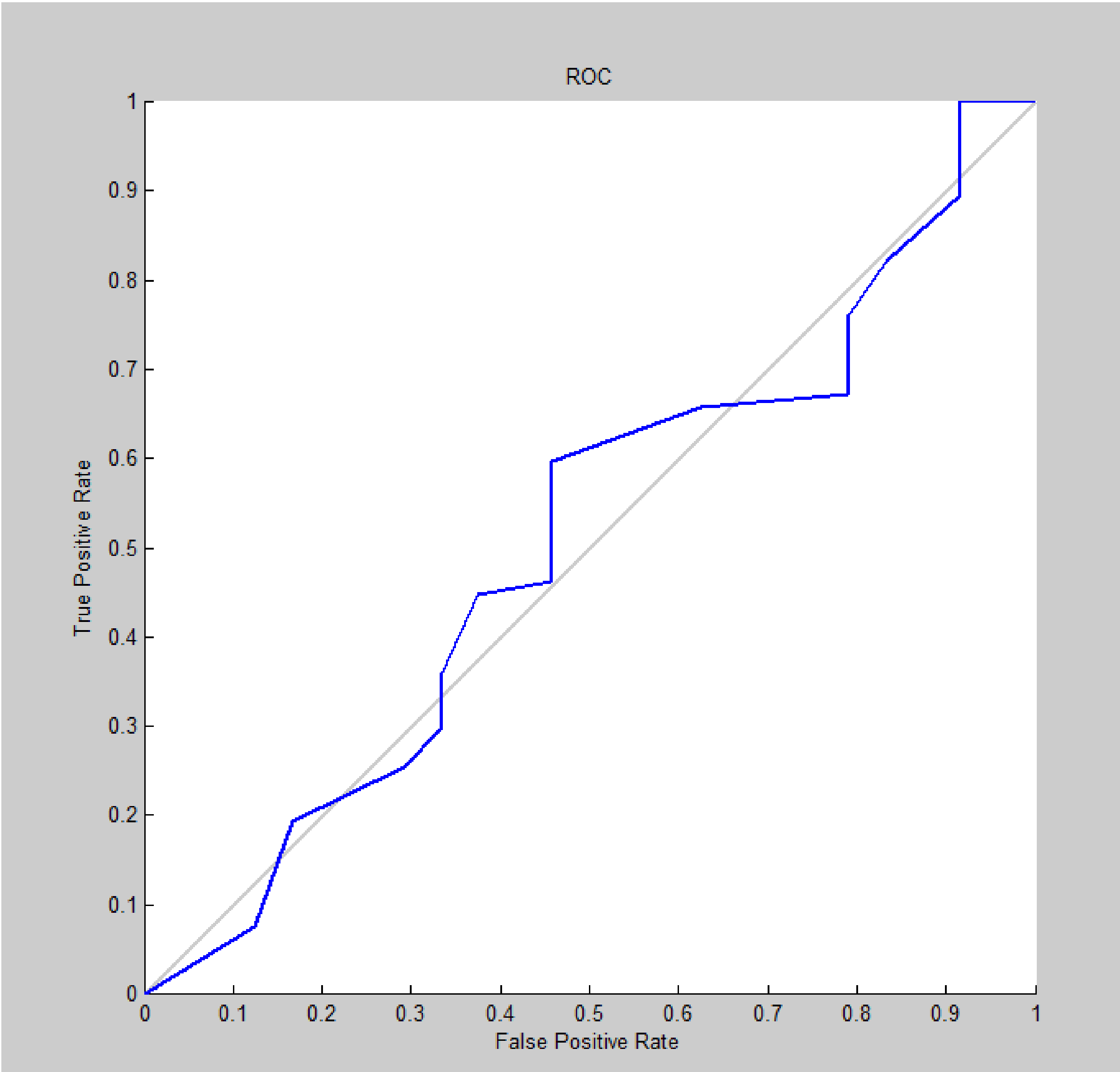
```
        if(p3(i,1)>=p4(i,1))
            pred2(i,1)=1;
            prob2(i,1)=p3(i,1);         %storing probability values for plotting roc
        else
            pred2(i,1)=0;
            prob2(i,1)=p4(i,1);         %storing probability values for plotting roc
        end
end
probability=[prob1' prob2'];
prediction = [pred1' pred2'];
display('prediction...');
actual=[positive_tst(:,4)' negative_tst(:,4)'];
result=[actual;prediction ];
display(result);

%%%%%%%%%%%%%%%%%% ASSESING THE PERFORMANCE OF THE SYSTEM %%%%%%%%%%%%%%%
performance(result');

%%%%%%%%%% PLOTTING ROC %%%%%%%%%%%%%

rocp(actual',probability');
```

Results

Performance Parameters

accuracy = 0.7363
error rate = 0.2637
true positive rate = 1
true negative rate = 0
precision = 0.7363
f_score = 0.8481
f_beta = 0.5301

Receiver Operating Characteristic (ROC)



EVALUATION OF THE PERFORMANCE OF THE CLASSIFIERS

Data1	Logistic Regression	Gaussian Discriminant Analys.	Naïve Bayes Analysis
Accuracy	0.4333	1	0.9333
Error rate	0.5667	0	0.0667
True Positive rate	0.4000	1	0.9333
True Negative rate	0.4667	1	0.9333
Precision	0.4286	1	0.9333
F score	0.4138	1	0.9333
F beta	0.2586	0.6250	0.5833

In terms of accuracy Gaussian discriminant analysis and Naïve bayes analysis produced better results respectively. The result is same in terms of F beta parameter.

Haberman's Survival Data	Logistic Regression	Gaussian Discriminant Analys.	Naïve Bayes Analysis
Accuracy	0.6703	0.7312	0.7363
Error rate	0.3297	0.2688	0.2637
True Positive rate	0.8209	0.7941	1
True Negative rate	0.2500	0.5600	0
Precision	0.7534	0.8308	0.7363
F score	0.7857	0.8120	0.8481
F beta	0.4911	0.5075	0.5301

In terms of accuracy Naïve bayes analysis and Gaussian discriminant analysis produced better results respectively. The result is same in terms of F beta parameter also.

STUDENT'S t DISTRIBUTION

In probability and statistics, **Student's *t*-distribution** (or simply the ***t*-distribution**) is a family of continuous probability distributions that arise when estimating the mean of a normally distributed population in situations where the sample size is small and population standard deviation is unknown. Whereas a normal distribution describes a full population, *t*-distributions describe samples drawn from a full population; accordingly, the *t*-distribution for each sample size is different, and the larger the sample, the more the distribution resembles a normal distribution.

The *t*-distribution plays a role in a number of widely used statistical analyses, including the Student's *t*-test for assessing the statistical significance of the difference between two sample means, the construction of confidence intervals for the difference between two population means, and in linear regression analysis. The Student's *t*-distribution also arises in the Bayesian analysis of data from a normal family.

If we take a sample of $n = v+1$ observations from a normal distribution (the black curve on the figure on the right of this page, representing a very large v), compute the sample mean and plot it, and repeat this process infinitely many times (for the same$n$), we get the probability density function for that n, as shown in the image on the right.

If we also compute the sample variance for these n observations, then the *t*-distribution (for n-*1*) c an be defined as the distribution of the location of the true mean, relative to the sample mean and divided by the sample standard deviation, after multiplying by the normalizing term $\sqrt{n}$, where n is the sample size. In this way, the *t*-distribution can be used to estimate how likely it is that the true mean lies in any given range.

The *t*-distribution is symmetric and bell-shaped, like the normal distribution, but has heavier tails, meaning that it is more prone to producing values that fall far from its mean. This makes it useful for understanding the statistical behavior of certain types of ratios of random quantities, in which variation in the denominator is amplified and may produce outlying values when the denominator of the ratio falls close to zero. The Student's *t*-distribution is a special case of the generalised hyperbolic distribution.

<u>Implementation</u>

Suppose a simple random sample of size n is taken from a population. If the population from which the sample is drawn forms a normal distribution, the distribution of follows Student's t-distribution with n – 1 degrees of freedom.

$$t = \frac{\overline{x} - \square}{\frac{s}{\sqrt{n}}}$$

where $\overline{x}$ is the mean , $\square$is the sample mean, s is the standard deviation and v=n-1 is the degree of freedom.

A Student's t-Distribution shares some characteristics of the normal distribution and differs from it on others.

Implementation of t-distribution in the classification algorithms used above

Consider the Data1 data set and implementation using logistic Regression Analsis. We take 5 individual observations to make a sample or in our case classification results. We are going to apply t distribution in terms of accuracy. The data under consideration is as follows.

Obseravations	**Accuracy**
1	0.4667
2	0.2667
3	0.4000

H_0: there is no significant difference between test values and predicted values.
H_1: there is significant difference between test values and predicted values.

Here the population mean, $\overline{x}$ = 1

Sample mean, ⍰ = 0.3778

Standard deviation,s = 0.1018

n= 2

t = 12.2240

here t > $t_{0.05}$ (the value of t at 5% level of significance) = 4.30

So H_0 is rejected and hence we conclude that there is significant difference between test values and predicted values or there is significant difference between test values and predicted values.

ANALYSIS OF VARIANCE (ANOVA)

Purpose

The reason for doing an ANOVA is to see if there is any difference between groups on some variable.

For example, you might have data on student performance in non-assessed tutorial exercises as well as their final grading. You are interested in seeing if tutorial performance is related to final grade. ANOVA allows you to break up the group according to the grade and then see if performance is different across these grades.

In general, one way anova techniques can be used to study the effect of multiple levels of a single factor. To determine if different levels of the factor affect measured observations differently, the following hypothesis are tested.

H_0: $\mu_i = \mu$ for all i=1,2,………..k

H_1: $\mu_i \neq \mu$ for some i=1,2,………..k

where μ_i is the population mean of individual level i.

Assumptions

When applying one way analysis of variance there are 3 key assumptions that should be satisfied.

1. The observations are obtained independently and randomly from the populations defined by the factor levels.
2. The population at each factor level is (approximately) normally distributed.
3. These normal populations have a common variance, σ^2

Thus for factor level I, the population is assumed to have a distribution which is N (μ_i, σ^2).

Two-way ANOVA

A two-way between groups ANOVA is used to look at complex groupings.

For example, the grades by tutorial analysis could be extended to see if overseas students performed differently to local students. What you would have from this form of ANOVA is:

The effect of final grade
The effect of overseas versus local
The interaction between final grade and overseas/local

Each of the **main effects** are one-way tests. The **interaction effect** is simply asking "is there any significant difference in performance when you take final grade and overseas/local acting together".

Non-parametric and Parametric

ANOVA is available for score or interval data as **parametric ANOVA.** This is the type of ANOVA you do from the standard menu options in a statistical package.

The **non-parametric version** is usually found under the heading "Nonparametric test". It is used when you have rank or ordered data.

You cannot use **parametric ANOVA** when you data is below interval measurement.

Where you have ***categorical*** data you do not have an ANOVA method - you would have to use Chi-square which is about interaction rather than about differences between groups.

Implementation of ANOVA (through F test) in the classification algorithms used above

Consider the Data1 data set and implementation using Logistic Regression Analysis (LRA) and Naïve bayes Analysis (NBA). We take 3 individual samples or in our case classification results. We are going to apply F distribution test in terms of accuracy. The data under consideration is as follows.

Obseravations	Accuracy (in LRA)	Accuracy (in NBA)
1	0.4667	0.9667
2	0.2667	0.9333
3	0.4000	0.9333

Aim

Our aim is to find whether these samples are taken from the same data set. For this we apply F test. The hypothesis are taken as follows.

H_0: the samples are taken from the same data set
H_1: the samples are taken from different data set

Obseravations	**Accuracy (in LRA)**	**Accuracy (in NBA)**
1	0.4667	0.9667
2	0.2667	0.9333
3	0.4000	0.9333
Mean ($\overline{x}$)	0.3778	0.9444
$\sum(x_i - \overline{x}_i)^2$	0.02073926	0.00074371

Sample size = 3

$s1^2 = \sum(x_1 - \overline{x}_1)^2$ / (sample size-1)

= 0.02073926 / 2

= 0.01036963

$s2^2 = \sum(x_2 - \overline{x}_2)^2$ / (sample size-1)

= 0.00074371 / 2

= 0.000371855

F = $\frac{s2^2}{s1^2}$

= 27.88

Conclusion

Since F < $F_{0.01}$ (= 99.0 (at 1 % level of significance)) , our hypothesis H_0 is accepted or the sample is taken from the same data set.

CHECKING WHETHER THE MAIL IS SPAM OR NOT

Here the algorithm used is Naïve Bayes Analysis. We have six training data set. Each word appearing in these test data are taken as an attribute and each training and test data are coded as shown in the table.

The prescence of a word in a particular member is taken as 1 and absence as 0. Similarly positive class is given 1 and negative class as 0.

	send	us	your	internet	banking	password	review	account	details	now	class
Training data 1	1	1	1	1	1	1	0	0	0	0	1
Training data 2	1	1	1	0	0	0	1	0	0	0	0
Training data 3	0	0	1	1	1	1	1	0	0	0	0
Training data 4	0	1	0	0	0	0	1	0	0	0	1
Training data 5	1	0	1	1	1	1	0	0	0	0	1
Training data 6	1	1	1	0	0	0	0	1	1	0	1
Test data 1	0	0	1	0	0	0	1	1	0	0	?
Test data 2	0	1	0	0	0	0	1	0	0	1	?

The test data is modeled using above data and tested on testing data

The MATLAB code used for the program

```
%%%%%%%     ASSIGNMENT -->#3     %%%%%%%%%%%%%%%

%%%%%%    NAIVE BAYES ANALYSIS ON SPAM CHECKING DATA   %%%%%%%


%%%%%%%%%%%%%%% DATA UPLOAD AND INITIALISING VARIABLES AND VECTORS%%%%%%
%clc;
data = load ('D:\the world of machine learning\assignments\data mining\assignment_3\spam.txt');
[m,n]=size(data);
c_no=2;

%%%%%%   CLASSIFYING INTO POSITIVE AND NEGATIVE CLASS %%%%%%%

[ positive,negative]=classification(data);
[a,b]=size(positive);
[c,d]=size(negative);

%%%%%%% FINDING THE CONDITIONAL PROBABILITIES FOR POSITIVE & NEGATIVE CLASS AND THEIR TOTAL PROBABILITY %%%%%%%%%

[prob_pos]=cond_prob(positive,2);
[prob_neg]=cond_prob(negative,2);
ppos=a/(a+c);
pneg=c/(a+c);
```

```
%%%%%%%     LAPLACE SMOOTHING       %%%%%%%%%

prob_pos=laplace_smooth(prob_pos,a);
prob_neg=laplace_smooth(prob_neg,c);

display(prob_pos);
display(prob_neg);

%%%%%%%%%% TESTING OF THE DATA STARTS %%%%%%%%%%%

testdata = [0 0 1 0 0 0 1 1 0 0 0; 0 1 0 0 0 0 1 0 0 1 0];
p1=nba_testing( testdata,prob_pos,ppos,c_no);
p2=nba_testing( testdata,prob_neg,pneg,c_no);

[g, f]=size(p1);
pred1 = zeros(g,1);
p5=p1;

for i=1:g

        if p5(i,1)>=p2(i,1)
            pred1(i,1)=1;
                 %storing probability values for plotting roc
        else
            pred1(i,1)=0;
                  %storing probability values for plotting roc
        end
end
display(pred1');
```

Result

Both the data has been marked as spam.