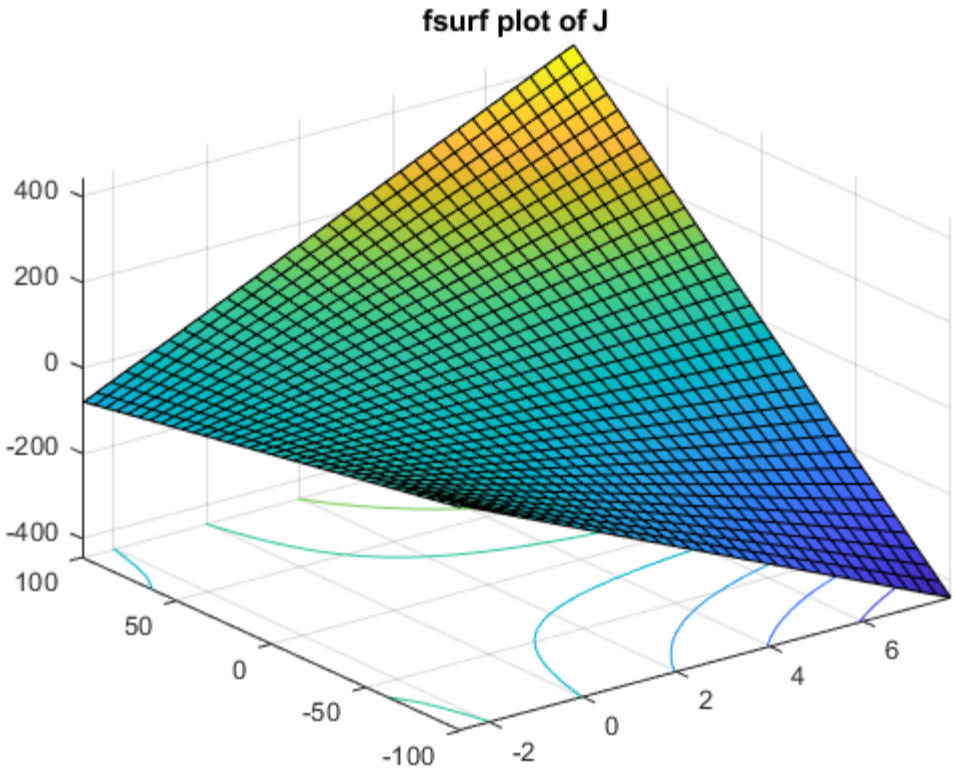
Contents Problem 1

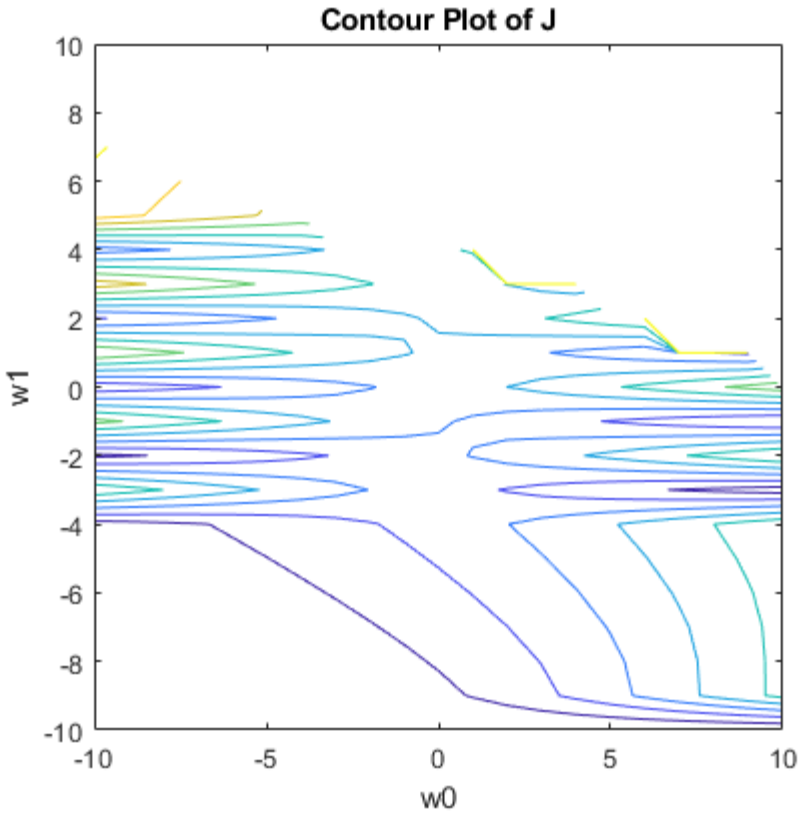

```
close all
clear all

warning('off', 'all')

%Data
id = 1:1:20;

hrs = [0.50, 0.75, 1.00, 1.25, 1.50, 1.75, 1.75, 2.00, 2.25, 2.50, 2.75, 3.00,...
    3.25, 3.50, 4.00, 4.25, 4.50, 4.75, 5.00, 5.50];
pss = [0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 1, 1, 1, 1, 1];

x = hrs;
y = pss;
```

Part A

```
%minimize J
% -1/20 * sum(pss[i] * ln(1/(1 + exp(-(w0 + w1*hrs[i])) + (1- pss[i]) * ln(1 - 1/(1 + exp(-(w0 +
w1*hrs[i]))))
w0 = -4.077;
w1 = 1.5046;

for i = 1:length(hrs)
  Jtemp = -1/20 * sum(pss(i) * log(1/(1 + exp(-(w0 + w1*hrs(i))))) + (1- pss(i)) * log(1 - 1/(1 +
exp(-(w0 + w1*hrs(i))))));
  J(i) = Jtemp;
end
for ii = 1:20
    t = -10;
    g = -10;
    fsurf(@(x,y) -1/20 * sum(y * log(1/(1 + exp(-(t + g*x)))) + (1- y) * log(1 - 1/(1 + exp(-(t +
g*x))))),[-100 100 -100 100], 'ShowContours','on')
    hold on
    t = t+1;
    g = g+1;
end
title('fsurf plot of J')


w_m = -10:10;
w2_m = -10:10;

[X1, X2] = meshgrid(w_m, w2_m);

for i = 1:length(hrs)
    Jtemp2 = -1/20 * sum(pss(i) * log(1./(1 + exp(-(X1 + X2.*hrs(i))))) + (1- pss(i)) * log(1 -
1./(1 + exp(-(X1 + X2.*hrs(i))))));
    Jtemp3(i+1,:) = Jtemp2;
end
```

```
figure;
contour(X1,X2,Jtemp3)
axis([-10 10 -10 10])
axis square
xlabel('w0')
ylabel('w1')
title('Contour Plot of J')
```

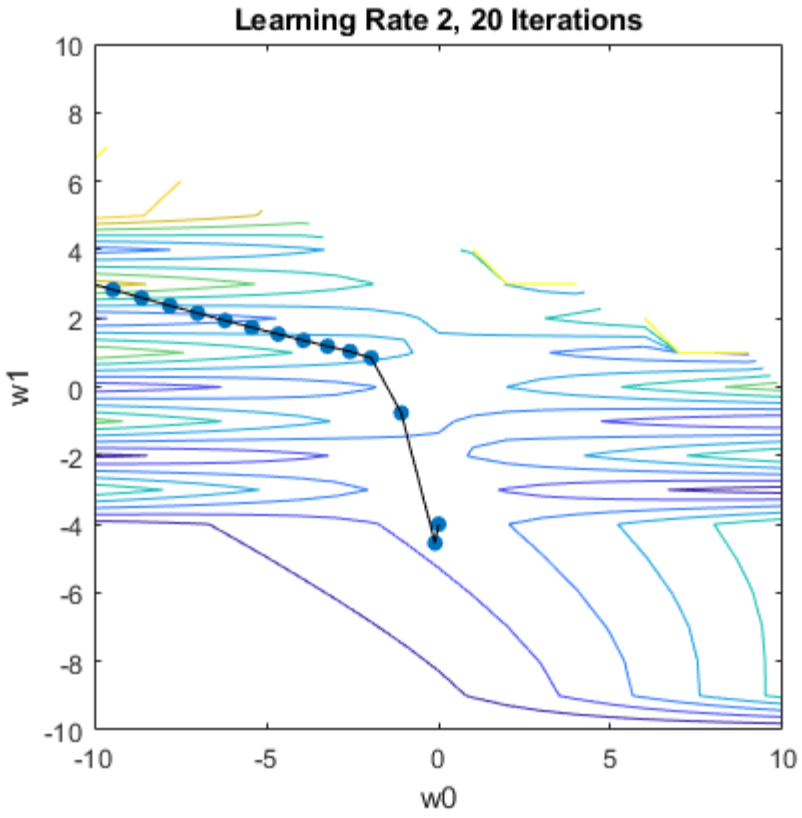




part B

```
w0 = 0;
w1 = -4;
alpha = 2;

for i = 1:length(hrs)
    dw0 = 1/length(hrs)*sum((-1 + y(i) + y(i) + y(i) * exp(-w0 - w1*x(i)))/(exp(-w0-w1*x(i)) + 1));

    dw1 = 1/length(hrs)*sum((-x(i) * (-y(i) + 1) + y(i)*x(i)*exp(-w0-w1*(x(i))))/ (exp(-w0-w1*x(i))
+ 1));

    w0new = w0 - alpha*dw0;
    w1new = w1 - alpha*dw1;
end

w0tot = [w0 w0new];
w1tot = [w1 w1new];

for kk = 1:19
       for i = 1:length(hrs)
            dw0 = 1/length(hrs)*sum((-1 + y(i) + y(i) + y(i) * exp(-w0new - w1new*x(i)))/(exp(-
w0new-w1new*x(i)) + 1));

            dw1 = 1/length(hrs)*sum(-(-x(i) * (-y(i) + 1) + y(i)*x(i)*exp(-w0new-w1new*(x(i))))/
(exp(-w0new-w1new*x(i)) + 1));

            w0new = w0new - alpha*dw0;
            w1new = w1new - alpha*dw1;
       end
    w0tot = [w0tot w0new];
    w1tot = [w1tot w1new];
end
```

```
figure;
contour(X1,X2,Jtemp3)
axis([-10 10 -10 10])
axis square
hold on
scatter(w0tot,w1tot,'filled')
hold on
plot(w0tot,w1tot, '-k')
title('Learning Rate 2, 20 Iterations')
xlabel('w0')
ylabel('w1')
```

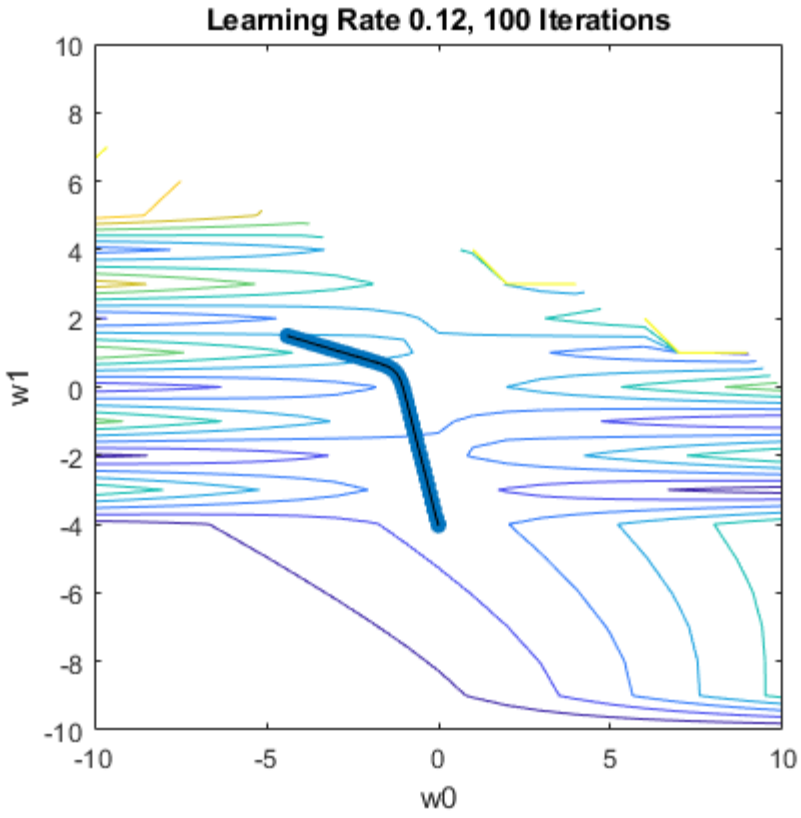


Part C

```
w0C = 0;
w1C = -4;
alpha = 0.12;

for i = 1:length(hrs)
    dw0 = 1/length(hrs)*sum((-1 + y(i) + y(i) + y(i) * exp(-w0 - w1*x(i)))/(exp(-w0-w1*x(i)) + 1));

    dw1 = 1/length(hrs)*sum((-x(i) * (-y(i) + 1) + y(i)*x(i)*exp(-w0-w1*(x(i))))/ (exp(-w0-w1*x(i))
+ 1));

    w0newC = w0 - alpha*dw0;
    w1newC = w1 - alpha*dw1;
end

w0totC = [w0C w0newC];
w1totC = [w1C w1newC];
```

```
for kk = 1:100
       for i = 1:length(hrs)
            dw0 = 1/length(hrs)*sum((-1 + y(i) + y(i) + y(i) * exp(-w0newC - w1newC*x(i)))/(exp(-
w0newC-w1newC*x(i)) + 1));

            dw1 = 1/length(hrs)*sum(-(-x(i) * (-y(i) + 1) + y(i)*x(i)*exp(-w0newC-w1newC*(x(i))))/
(exp(-w0newC-w1newC*x(i)) + 1));

            w0newC = w0newC - alpha*dw0;
            w1newC = w1newC - alpha*dw1;
       end
    w0totC = [w0totC w0newC];
    w1totC = [w1totC w1newC];
end



figure;
contour(X1,X2,Jtemp3)
axis([-10 10 -10 10])
axis square
hold on
scatter(w0totC,w1totC,'filled')
hold on
plot(w0totC,w1totC, '-k')

title('Learning Rate 0.12, 100 Iterations')
xlabel('w0')
ylabel('w1')

disp('I think there is something wrong with my equation, but even so I got close to the optimal
values')
disp(w0newC)
disp(w1newC)
```

```
I think there is something wrong with my equation, but even so I got close to the optimal values
   -4.4030

    1.4920
```



Published with MATLAB® R2018b

Contents Problem 2

prob2

```
clear all
close all
```

Part A

```
opts = detectImportOptions('biopsy_data_missing_values.csv', 'NumHeaderLines', 1');
preview('biopsy_data_missing_values.csv', opts)

A = readtable('biopsy_data_missing_values.csv', 'HeaderLines', 1);

for i = 1:height(A)
    if isempty(A.Var2{i})
        A.Var2{i} = 'Irregular';
    end
end



nanind = find(isnan(A.Var1));
var1tmp = A.Var1;
var1tmp(nanind(1)) = 6;
var1tmp(nanind(2)) = 10;
A.Var1 = var1tmp;

A
```

```
ans =

  8×7 table

    Var1      Var2        Var3        Var4         Var5        Var6          Var7
    ____    ________    _______    _________    ________    _________    ___________

      1     'Circle'    'Large'    'Convex'     'Smooth'    'Dark'       'Malignant'
      2     'Circle'    'Large'    'Concave'    'Smooth'    'Dark'       'Malignant'
      3     'Circle'    'Large'    'Flat'       'Smooth'    'Red'        'Malignant'
      4     ''          'Small'    'Concave'    'Rough'     'Dark'       'Malignant'
      5     'Circle'    'Large'    'Flat'       'Rough'     'Neutral'    'Malignant'
    NaN     'Circle'    'Large'    'Concave'    'Rough'     'Dark'       'Malignant'
      7     'Circle'    'Large'    'Convex'     'Smooth'    'Neutral'    'Benign'
      8     ''          'Large'    'Concave'    'Smooth'    'Red'        'Benign'


A =

  12×7 table
```

Var1	Var2	Var3	Var4	Var5	Var6	Var7
1	'Circle'	'Large'	'Convex'	'Smooth'	'Dark'	'Malignant'
2	'Circle'	'Large'	'Concave'	'Smooth'	'Dark'	'Malignant'
3	'Circle'	'Large'	'Flat'	'Smooth'	'Red'	'Malignant'
4	'Irregular'	'Small'	'Concave'	'Rough'	'Dark'	'Malignant'
5	'Circle'	'Large'	'Flat'	'Rough'	'Neutral'	'Malignant'
6	'Circle'	'Large'	'Concave'	'Rough'	'Dark'	'Malignant'
7	'Circle'	'Large'	'Convex'	'Smooth'	'Neutral'	'Benign'
8	'Irregular'	'Large'	'Concave'	'Smooth'	'Red'	'Benign'
9	'Triangle'	'Small'	'Convex'	'Rough'	'Dark'	'Benign'
10	'Circle'	'Large'	'Flat'	'Smooth'	'Neutral'	'Benign'
11	'Irregular'	'Large'	'Concave'	'Smooth'	'Dark'	'Benign'
12	'Irregular'	'Large'	'Concave'	'Smooth'	'Red'	'Benign'

Part B

```
ns1 = {'Irregular', 'Large', 'Convex', 'Rough', 'Neutral'};
ns2 = {'Irregular', 'Small', 'Flat', 'Rough', 'Red'};
ns3 = {'Circle', 'Large', 'Concave', 'Smooth', 'Neutral'};
ns4 = {'Cirlce', 'Large', 'Convex', 'Smooth', 'Dark'};
ns5 = {'Triangle', 'Large', 'Concave', 'Smooth', 'Neutral'};

nstot = {ns1 ns2 ns3 ns4 ns5};

%for ns1 ex


A1 = A(1:6,:); %malignant table (pos)
A2 = A(7:12,:); %benign table (neg)
pc1 = 0.5;
pc2 = 0.5;


for i = 1:length(nstot)
    pshape1 = length(find(strcmp(A1.Var2, nstot{i}{1})))/height(A1);
    prad1 = length(find(strcmp(A1.Var3, nstot{i}{2})))/height(A1);
    pconcav1 = length(find(strcmp(A1.Var4, nstot{i}{3})))/height(A1);
    ptext1 = length(find(strcmp(A1.Var5, nstot{i}{4})))/height(A1);
    pcol1 = length(find(strcmp(A1.Var6, nstot{i}{5})))/height(A1);

    posprob(i) = log(pshape1 * prad1 * pconcav1 * ptext1 * pcol1 * pc1);


    nshape1 = length(find(strcmp(A2.Var2, nstot{i}{1})))/height(A2);
    nrad1 = length(find(strcmp(A2.Var3, nstot{i}{2})))/height(A2);
    nconcav1 = length(find(strcmp(A2.Var4, nstot{i}{3})))/height(A2);
    ntext1 = length(find(strcmp(A2.Var5, nstot{i}{4})))/height(A2);
    ncol1 = length(find(strcmp(A2.Var6, nstot{i}{5})))/height(A2);

    negprob(i) = log(nshape1 * nrad1 * nconcav1 * ntext1 * ncol1 * pc2);

end


for ii = 1:5
    if negprob(ii) > posprob(ii)
```

```
        class{ii} = 'benign';
    else
        class{ii} = 'malignant';
        %err on this side, to prompt pt to another test
    end
end


samplenames = 1:5;
LogNegProb = negprob;
LogPosProb = posprob;
varnames = {'Sample', 'LogNegProb', 'LogPosProb', 'Classification'};
ResultTable = table(samplenames', LogNegProb', LogPosProb', class', 'VariableNames', varnames)


disp('Sample 5 had was one of two triangle shapes, which was not contained in the Malignant group.')
```

```
ResultTable =

  5×4 table

    Sample    LogNegProb    LogPosProb    Classification
    ______    __________    __________    ______________

      1        -5.5576       -6.9439       'benign'
      2        -7.8602       -7.8602       'malignant'
      3        -3.9482       -4.2358       'benign'
      4           -Inf          -Inf       'malignant'
      5        -4.6413          -Inf       'benign'

Sample 5 had was one of two triangle shapes, which was not contained in the Malignant group.
```

Published with MATLAB® R2018b

Contents

Problem 3

prob 3

```
clear all
close all

opts = detectImportOptions('iris_dataset.csv', 'NumHeaderLines', 1);
preview('iris_dataset.csv', opts)


A = readtable('iris_dataset.csv', 'HeaderLines', 1);
```

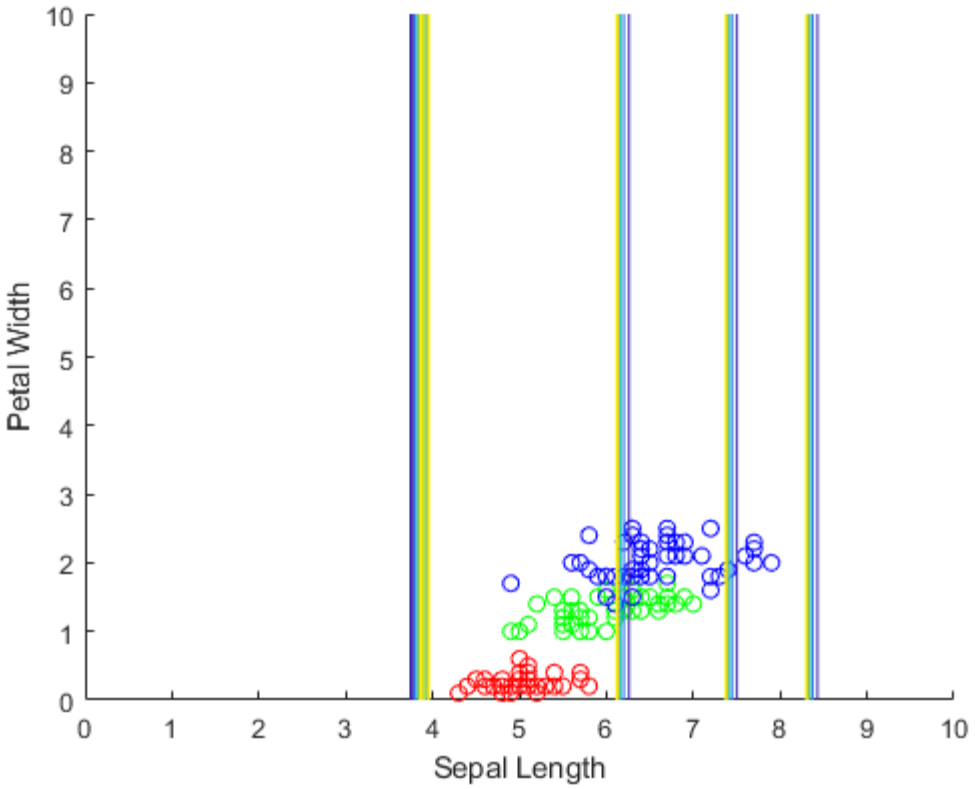
```
ans =

  8×5 table

    Var1    Var2    Var3    Var4         Var5
    ____    ____    ____    ____    _____________

    5.1     3.5     1.4     0.2     'Iris-setosa'
    4.9       3     1.4     0.2     'Iris-setosa'
    4.7     3.2     1.3     0.2     'Iris-setosa'
    4.6     3.1     1.5     0.2     'Iris-setosa'
      5     3.6     1.4     0.2     'Iris-setosa'
    5.4     3.9     1.7     0.4     'Iris-setosa'
    4.6     3.4     1.4     0.3     'Iris-setosa'
      5     3.4     1.5     0.2     'Iris-setosa'
```

Part B

```
%split A into 3
indis = find(strcmp(A.Var5, 'Iris-setosa'));
indver = find(strcmp(A.Var5, 'Iris-versicolor'));
indvir = find(strcmp(A.Var5, 'Iris-virginica'));

A1 = A(1:50,:);
A2 = A(51:100,:);
A3 = A(101:150,:);


scatter(A1.Var1, A1.Var4, 'r')
hold on
scatter(A2.Var1, A2.Var4, 'g')
hold on
scatter(A3.Var1, A3.Var4, 'b')
axis([0 10 0 10])
hold on
```

```
xlabel('Sepal Length')
ylabel('Petal Width')

x1 = 0:0.05:10;
x4 = 0:0.05:10;
[X1, X2] = meshgrid(x1,x4);

sigma1 = 0.2;
sigma4 = sigma1;
amesh1 = 0;
amesh1_x4 = 0;
m = 50;
pc1 = 50/150;
pc2 = 50/150;
pc3 = 50/150;

for i = 1:50
   am1 =  exp(-(X1 - A1.Var1(i)).^2./(2*sigma1^2));
   amesh1 = amesh1+am1;
end
for ii = 1:50
   am4 =  exp(-(X2 - A1.Var4(ii)).^2./(2*sigma4^2));
   amesh1_x4 = amesh1+am4;
end

meshtotx1a1 = 1/m*1/(sigma1*sqrt(2*pi)) .* amesh1;
meshtotx4a1 = 1/m*1/(sigma1*sqrt(2*pi)) .* amesh1_x4;
totmesha1 = meshtotx1a1*meshtotx4a1 *pc1;


contour(X1, X2, totmesha1, 0:0.03:0.15)
%abc.levels = [0:0.03:0.15]

%repeat for other groups

for i = 1:50
   am1 =  exp(-(X1 - A2.Var1(i)).^2./(2*sigma1^2));
   amesh1 = amesh1+am1;
end
for ii = 1:50
   am4 =  exp(-(X2 - A2.Var4(ii)).^2./(2*sigma4^2));
   amesh1_x4 = amesh1+am4;
end
meshtotx1a1 = 1/m*1/(sigma1*sqrt(2*pi)) .* amesh1;
meshtotx4a1 = 1/m*1/(sigma1*sqrt(2*pi)) .* amesh1_x4;
totmesha1 = meshtotx1a1*meshtotx4a1 *pc1;
contour(X1, X2, totmesha1, 0:0.03:0.15)



for i = 1:50
   am1 =  exp(-(X1 - A3.Var1(i)).^2./(2*sigma1^2));
   amesh1 = amesh1+am1;
end
for ii = 1:50
   am4 =  exp(-(X2 - A3.Var4(ii)).^2./(2*sigma4^2));
   amesh1_x4 = amesh1+am4;
end
meshtotx1a1 = 1/m*1/(sigma1*sqrt(2*pi)) .* amesh1;
```

```
meshtotx4a1 = 1/m*1/(sigma1*sqrt(2*pi)) .* amesh1_x4;
totmesha1 = meshtotx1a1*meshtotx4a1 *pc1;
contour(X1, X2, totmesha1, 0:0.03:0.15)
```

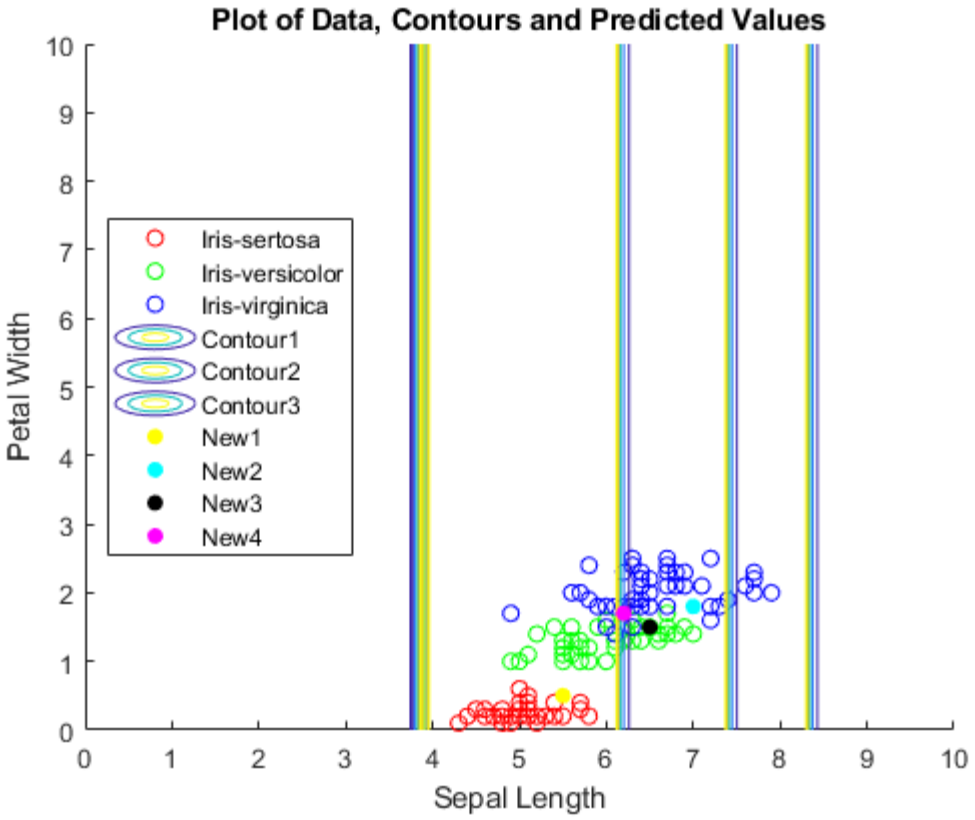


Part C

```
x1new = [5.5, 7, 6.5, 6.2];
x4new = [0.5, 1.8, 1.5, 1.7];
bm2 = 50*[];
bm4 = 50*[];
bm5 = 5*[];
bm6 = 5*[];
%c1
for i = 1:length(x1new)
    for kk = 1:50
        bm1 =  exp(-(x1new(i) - A1.Var1(kk))^2/(2*sigma1^2));
        bm2(kk) = bm1;

    end
    bm5(i) = sum(bm2);
end
for i = 1:length(x1new)
    for kk = 1:50
        bm3 =  exp(-(x4new(i) - A1.Var4(kk))^2/(2*sigma1^2));
        bm4(kk) = bm3;
    end
    bm6(i) = sum(bm4);
end
%now have the sum of gauss
%mult k, attributes and P(c1)

class1Q = (1/m*1/(sigma1*sqrt(2*pi)) * bm5) .*  (1/m*1/(sigma1*sqrt(2*pi)) * bm6) .* pc1;
```

```
%c2
for i = 1:length(x1new)
    for kk = 1:50
        bm1 =  exp(-(x1new(i) - A2.Var1(kk))^2/(2*sigma1^2));
        bm2(kk) = bm1;

    end
    bm5(i) = sum(bm2);
end
for i = 1:length(x1new)
    for kk = 1:50
        bm3 =  exp(-(x4new(i) - A2.Var4(kk))^2/(2*sigma1^2));
        bm4(kk) = bm3;
    end
    bm6(i) = sum(bm4);
end
%now have the sum of gauss

class2Q = (1/m*1/(sigma1*sqrt(2*pi)) * bm5) .*  (1/m*1/(sigma1*sqrt(2*pi)) * bm6) .* pc2;


%c3
for i = 1:length(x1new)
    for kk = 1:50
        bm1 =  exp(-(x1new(i) - A3.Var1(kk))^2/(2*sigma1^2));
        bm2(kk) = bm1;

    end
    bm5(i) = sum(bm2);
end
for i = 1:length(x1new)
    for kk = 1:50
        bm3 =  exp(-(x4new(i) - A3.Var4(kk))^2/(2*sigma1^2));
        bm4(kk) = bm3;
    end
    bm6(i) = sum(bm4);
end
%now have the sum of gauss


class3Q = (1/m*1/(sigma1*sqrt(2*pi)) * bm5) .*  (1/m*1/(sigma1*sqrt(2*pi)) * bm6) .* pc3;


for i = 1:length(x1new)
    if class1Q(i) > class2Q(i) && class3Q(i)
        classification_tot{i} = 'Iris-sertosa';
    elseif class2Q(i) > class3Q(i) && class1Q(i)
        classification_tot{i} = 'Iris-versicolor';
    elseif class3Q(i) > class2Q(i) && class1Q(i)
        classification_tot{i} = 'Iris-virginica';
    end
end

SampleNumber = 1:4;

varNames = {'SampleNumber', 'Class1QProb', 'Class2QProb', 'Class3QProb', 'classification'};
ResultTable = table(SampleNumber', class1Q', class2Q', class3Q', classification_tot',
```

```
'VariableNames', varNames);
ae = {'y', 'c', 'k', 'm'};
for i = 1:length(x1new)
    scatter(x1new(i), x4new(i), ae{i}, 'filled')
    hold on
end

legend('Iris-sertosa', 'Iris-versicolor', 'Iris-virginica', 'Contour1', 'Contour2', 'Contour3',
'New1', 'New2', 'New3', 'New4', 'Location', 'west')
xlabel('Sepal Length')
ylabel('Petal Width')
title('Plot of Data, Contours and Predicted Values')
```



Echoing Values

```
diary vjprob3.txt
echo on
ResultTable

disp('I recognize that my contours are incorrect, but I am not sure how to fix it')

echo off
```

```
ResultTable

ResultTable =

  4×5 table

    SampleNumber    Class1QProb    Class2QProb    Class3QProb    classification
```

1	0.12985	0.0029045	6.9147e-08	'Iris-sertosa'
2	1.4121e-19	0.017232	0.11379	'Iris-virginica'
3	6.9678e-11	0.16015	0.081672	'Iris-versicolor'
4	3.5286e-11	0.10984	0.15887	'Iris-virginica'

```
disp('I recognize that my contours are incorrect, but I am not sure how to fix it')
```

```
I recognize that my contours are incorrect, but I am not sure how to fix it
```

```
echo off
```

Published with MATLAB® R2018b

HW2_P4_Jha_Vibhav

March 29, 2021

0.1 HW2 Problem 4

0.2 Name: Vibhav Jha

0.2.1 Imports

[1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import confusion_matrix
from sklearn.metrics import auc
```

0.2.2 1. Logistic regression

a.

[2]:
```
df = pd.read_csv ('data.csv')

x = df[['radius_mean']].to_numpy()

y1 = df[['diagnosis']]
y1 = y1.to_numpy()

df['diagnosis'] = df['diagnosis'].replace(['B'], 0)
df['diagnosis'] = df['diagnosis'].replace(['M'], 1)

y = df[['diagnosis']].to_numpy()
y = y.ravel()
log_reg = LogisticRegression()
log_reg.fit(x,y)

X_new = np.linspace(0, 30, 569).reshape(-1,1)
y_proba = log_reg.predict_proba(X_new)

intercept = np.squeeze(log_reg.intercept_)
coef = np.squeeze(log_reg.coef_)
```

b.

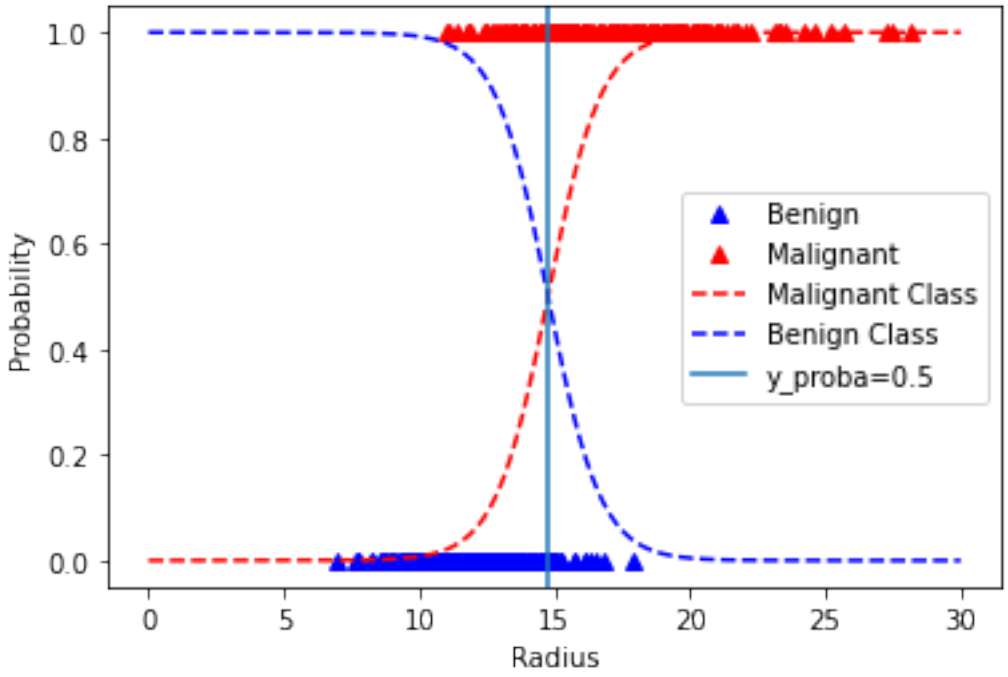
[3]:
```
print('The 50% Classification Boundary is at y_proba = 0.5 and at Radius Mean = ␣
 ↪ ', -intercept/coef)
#determine 50% Classification Boundary and corresponding radius value
```

```
The 50% Classification Boundary is at y_proba = 0.5 and at Radius Mean =
14.755610824791166
```

c.

[4]:
```
plt.figure()


plt.plot(x[y==0], y[y==0], "b^")
plt.plot(x[y==1], y[y==1], "r^")

plt.plot(X_new, y_proba[:,1], "r--")
plt.plot(X_new, y_proba[:,0], "b--")

plt.axvline(-intercept/coef)
 #this plots the line in which the y_proba = 0.5
 #this correlates with the decision boundary? not sure if this
 #is the correct way of thnking about it


plt.xlabel('Radius')
plt.ylabel('Probability')
plt.legend(['Benign', 'Malignant', 'Malignant Class', 'Benign Class', 'y_proba=0.
 ↪5'])
```

[4]:
```
<matplotlib.legend.Legend at 0x1ef0ccf2730>
```



0.2.3 2. Cost function plot

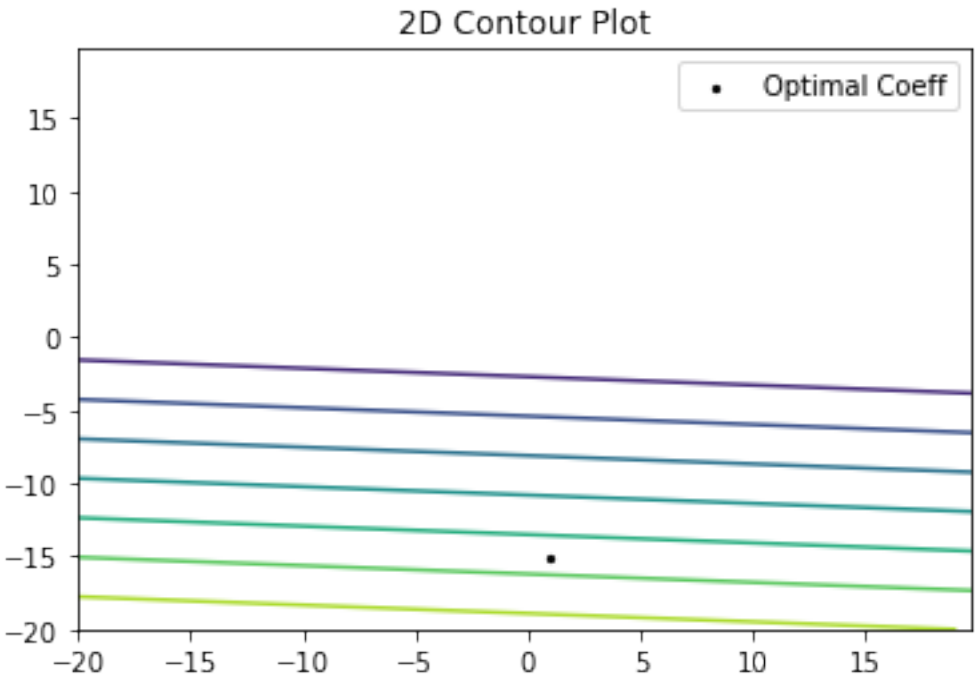
a.

[5]:
```
XX = np.arange(-20, 20, 0.25)
YY = np.arange(-20, 20, 0.25)
X1, X2 = np.meshgrid(XX, YY)
cost2 = []
cost2 = np.zeros((160, 160))
plt.figure()

for i in range(569):
    cost = -y.item(i)*np.log(1/(1 + np.exp(-X1 - X2*x.item(i)))-(1-y.item(i))*np.
 ↪log(1-1/np.exp(-X1-X2*x.item(i))))
    cost2 = cost2 + cost


plt.contour(X1, X2, cost2)

plt.scatter(coef,intercept, 5, 'k')

plt.legend(['Optimal Coeff'])
plt.title('2D Contour Plot')
```

```
<ipython-input-5-d8def7bd6dc8>:9: RuntimeWarning: divide by zero encountered in
log
```

```
  cost = -y.item(i)*np.log(1/(1 + np.exp(-X1 -
X2*x.item(i)))-(1-y.item(i))*np.log(1-1/np.exp(-X1-X2*x.item(i))))
<ipython-input-5-d8def7bd6dc8>:9: RuntimeWarning: invalid value encountered in
log
  cost = -y.item(i)*np.log(1/(1 + np.exp(-X1 -
X2*x.item(i)))-(1-y.item(i))*np.log(1-1/np.exp(-X1-X2*x.item(i))))
<ipython-input-5-d8def7bd6dc8>:9: RuntimeWarning: invalid value encountered in
multiply
  cost = -y.item(i)*np.log(1/(1 + np.exp(-X1 -
X2*x.item(i)))-(1-y.item(i))*np.log(1-1/np.exp(-X1-X2*x.item(i))))
```

[5]: `Text(0.5, 1.0, '2D Contour Plot')`

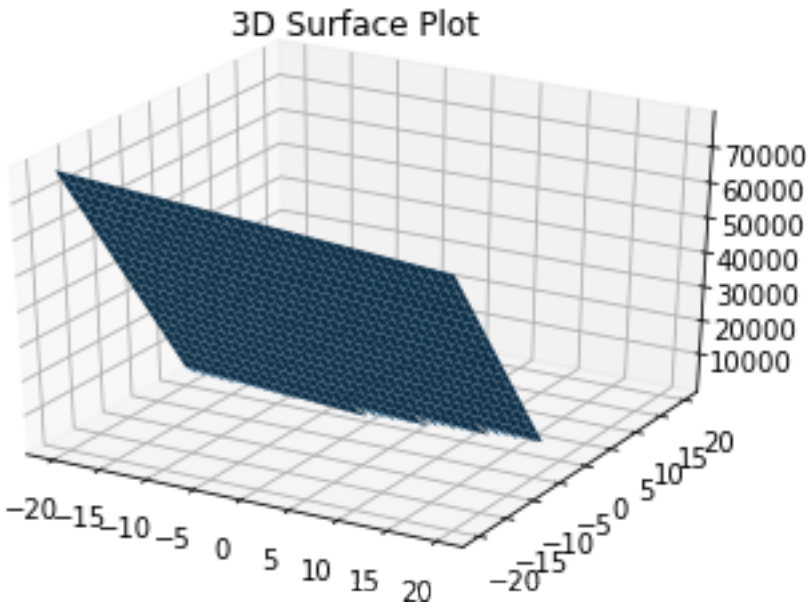


b.

[6]:
```
fig = plt.figure()
ax = plt.axes(projection='3d')
surf = ax.plot_surface(X1, X2, cost2)
plt.title('3D Surface Plot')
```

```
<ipython-input-6-893db1185b00>:3: UserWarning: Z contains NaN values. This may
result in rendering artifacts.
  surf = ax.plot_surface(X1, X2, cost2)
```

[6]: `Text(0.5, 0.92, '3D Surface Plot')`



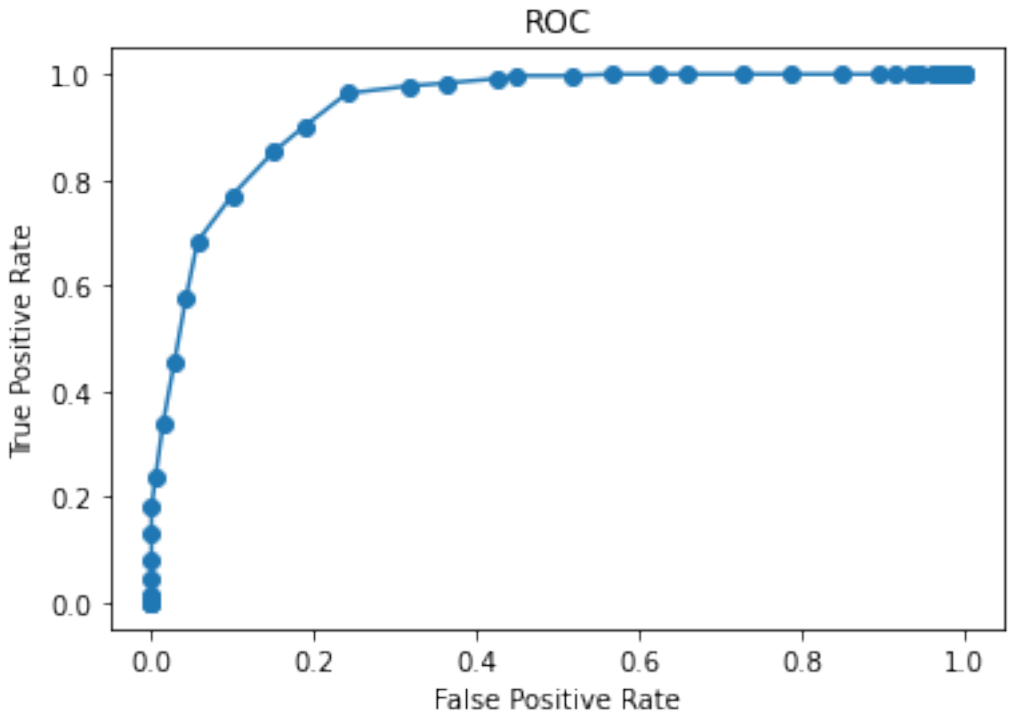
0.2.4 3. ROC

a.

[7]:
```
z3 = [2]*569
tpr = [2]*51
fpr = [2]*51
spec = [2]*51
varybound = 5

for kk in range(51):
    for j in range(569):
        for i in range(569):
            if x[i]<varybound:
                z3[i]=0

            if x[i]>varybound:
                z3[i] = 1


        cma =  confusion_matrix(y, z3)
        tp = cma[0,0]
        fn = cma[0,1]
        tn = cma[1,1]
        fp = cma[1,0]
        tpr[kk] = tp/(tp+fn)
        fpr[kk] = fp/(fp+tn)
        spec[kk] = tn/(tp+fn)
```

```
    varybound = varybound + 0.5
    # print(varybound)


plt.figure()
plt.scatter(fpr,tpr)
plt.plot(fpr,tpr)
plt.title('ROC')
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
```

[7]: Text(0, 0.5, 'True Positive Rate')



b.

[8]:
```
aucFPRTPR = auc(fpr, tpr)
print('AUC for FPR and TPR ROC: ', aucFPRTPR)
```

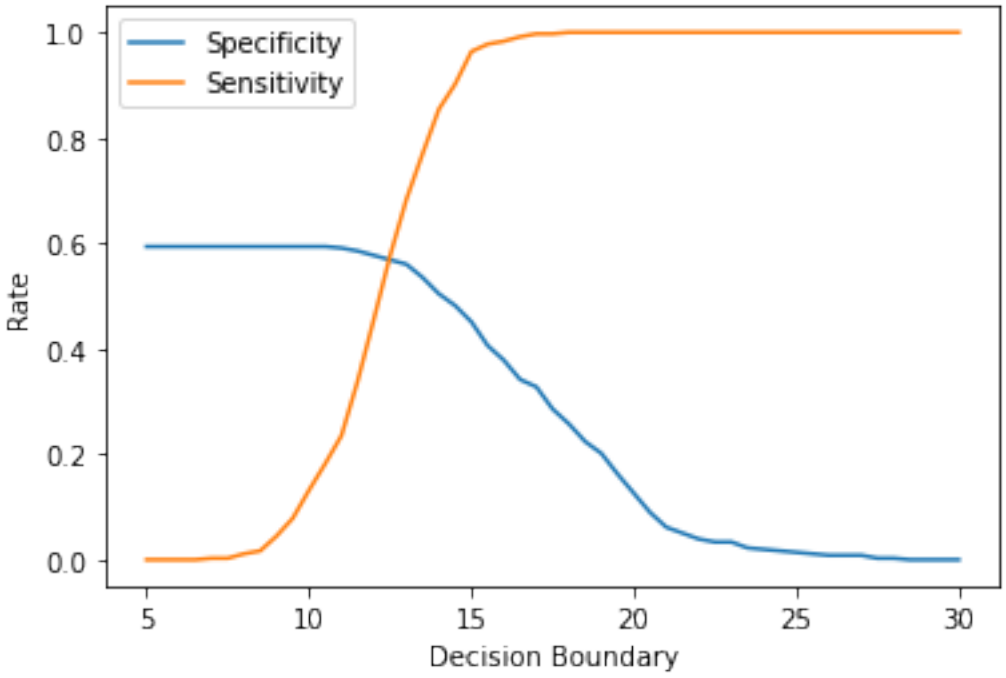
```
AUC for FPR and TPR ROC:  0.935217483219703
```

c.

[9]:
```
plt.figure()
vbound = np.linspace(5,30,51)
plt.plot(vbound, spec)
```

```
plt.plot(vbound, tpr)
plt.xlabel('Decision Boundary')
plt.ylabel('Rate')
plt.legend(['Specificity', 'Sensitivity'])
```

[9]: <matplotlib.legend.Legend at 0x1ef0d278be0>



0.2.5 4. Confusion matrix

a.

[10]:
```
z2 = [2]*569
for i in range(569):
    if x[i]<14.7:
       z2[i]=0
       # print('did it')
    if x[i]>14.7:
        z2[i] = 1
        #print(i)

cm = confusion_matrix(y, z2)
#sens or tpr = TP/(TP+FN)
#spec or tnr = TN/(TP+FN)

print('Confusion Matrix: ')
print(cm)
```

```
Confusion Matrix:
[[332  25]
 [ 44 168]]
```

b.

[11]:
```
tp = cm[0,0]
fn = cm[0,1]
tn = cm[1,1]
optimal_sensitivity = tp/(tp+fn)
optimal_specificity = tn/(tp+fn)

print('Optimal fit Sensitivity: ', optimal_sensitivity)
print('Optimal fit Specificiy: ', optimal_specificity)
```

```
Optimal fit Sensitivity:  0.9299719887955182
Optimal fit Specificiy:  0.47058823529411764
```