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.

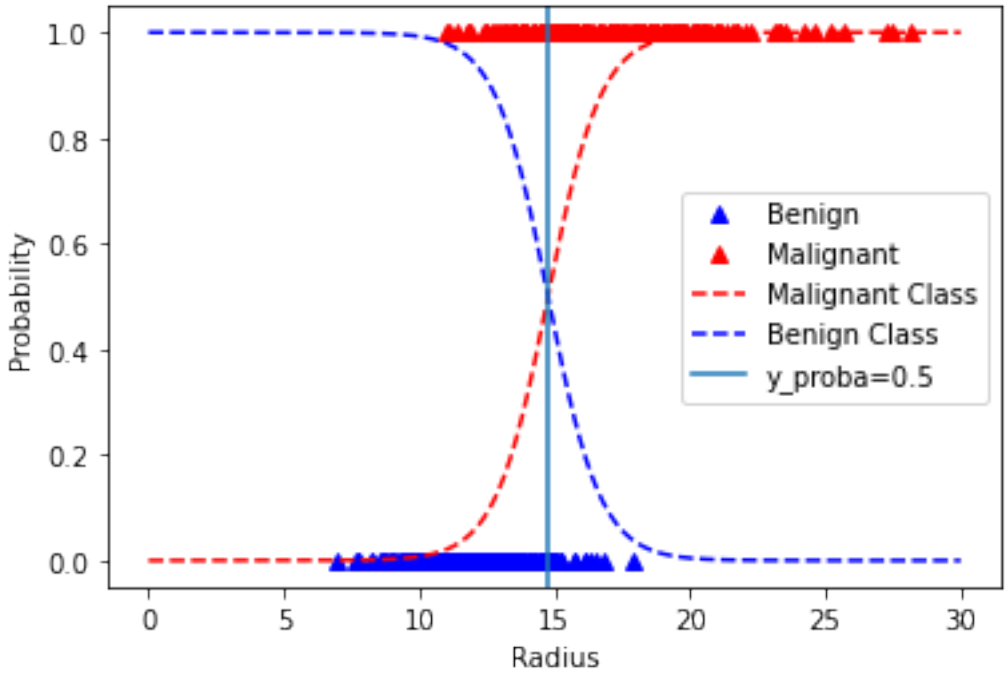
[3]:
```
print('The 50% Classification Boundary is at y_proba = 0.5 and at Radius Mean = ␣
 ↪ ', -intercept/coef)
#determine 50% Classification Boundary and corresponding radius value
```

```
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

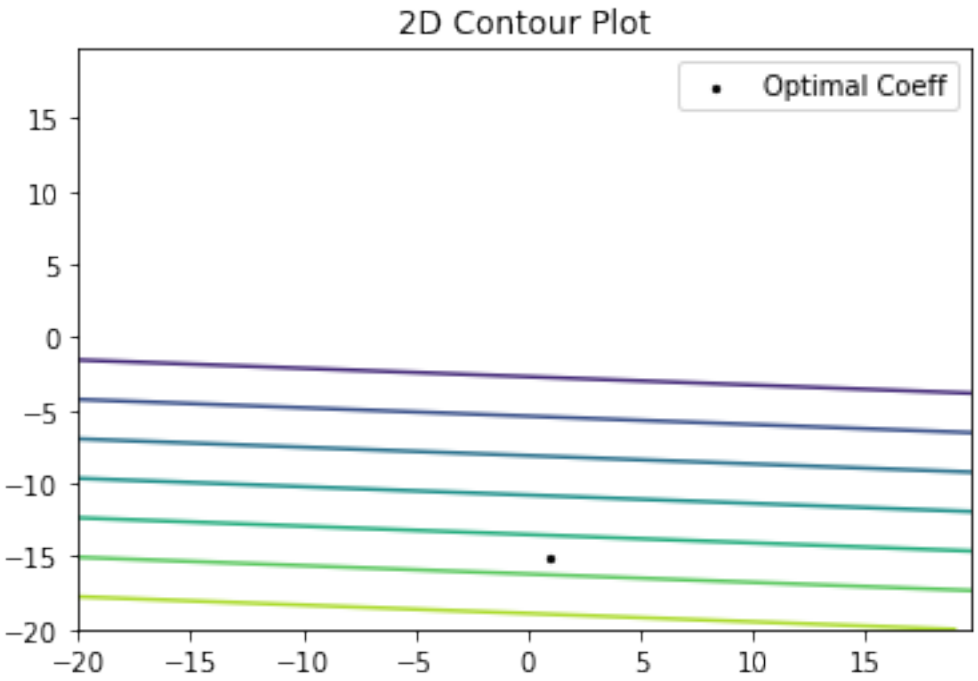
a.

[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')`

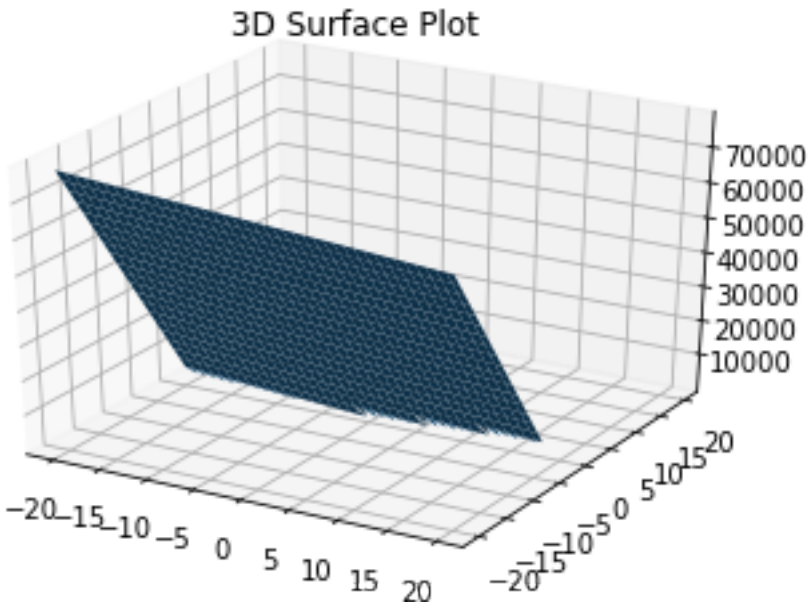


b.

[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')`



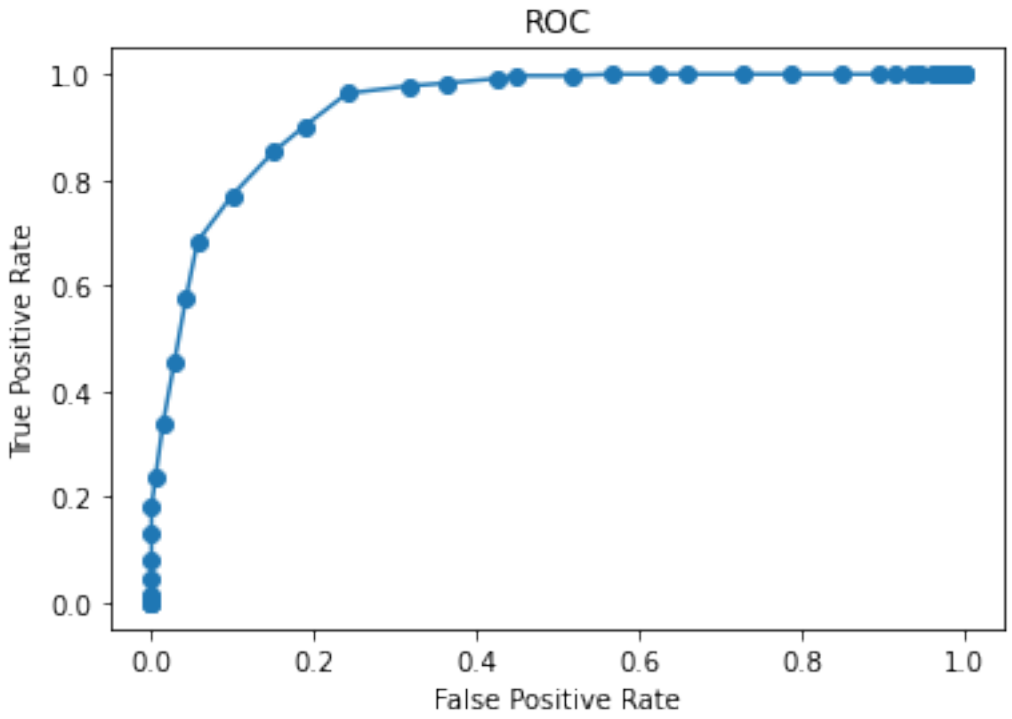
0.2.4 3. ROC

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)
```

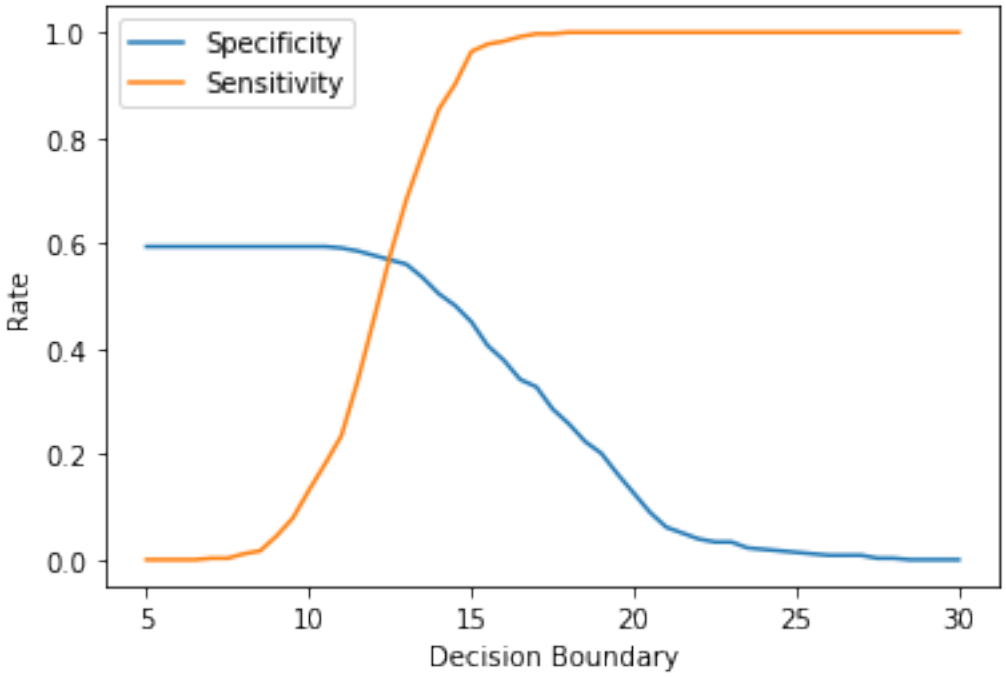
```
AUC for FPR and TPR ROC:  0.935217483219703
```

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
```