Lab 6

Ido Israeli (ID - 212432439)
Jonathan Derhy (ID - 315856377)

In [ ]:
```
import numpy as np
import pandas as pd
import warnings
warnings.filterwarnings('ignore')
import matplotlib.pyplot as plt
```

Loading Train Data

In [ ]:
```
dfTrain = pd.read_csv('.\External\Data\TrainData.csv')
dfTrain
```

Out[ ]:

	diagnosis	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_mean	cc
0	1	20.260	23.03	132.40	1264.0	0.09078	
1	0	13.300	21.57	85.24	546.1	0.08582	
2	0	12.220	20.04	79.47	453.1	0.10960	
3	0	9.847	15.68	63.00	293.2	0.09492	
4	1	21.100	20.52	138.10	1384.0	0.09684	
...	...	...	...	...	...	...	
450	0	13.680	16.33	87.76	575.5	0.09277	
451	0	11.290	13.04	72.23	388.0	0.09834	
452	0	13.490	22.30	86.91	561.0	0.08752	
453	1	20.160	19.66	131.10	1274.0	0.08020	
454	0	10.490	19.29	67.41	336.1	0.09989	

455 rows × 31 columns

Extracting Train Classifications into Ytrain

In [ ]:
```
Ytrain = dfTrain["diagnosis"].values
Ytrain
```

Out[ ]:
```
array([1, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 1, 0, 1, 1,
       0, 0, 1, 1, 1, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 1, 0, 1, 0, 1,
       1, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 1, 0, 0, 0, 0,
       0, 1, 1, 0, 1, 1, 0, 0, 1, 0, 0, 1, 1, 0, 1, 0, 0, 0, 0, 1, 0, 0,
       0, 0, 1, 1, 0, 0, 0, 1, 0, 1, 1, 0, 1, 1, 0, 1, 0, 0, 0, 1, 0, 1,
       1, 1, 1, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0,
       0, 1, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 1, 0, 0, 0,
       0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 1, 1, 0,
       0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 1,
       0, 1, 0, 1, 0, 1, 1, 0, 0, 1, 0, 1, 1, 0, 1, 0, 0, 1, 0, 1, 0, 0,
       1, 1, 1, 0, 0, 1, 0, 0, 1, 1, 1, 0, 1, 1, 1, 1, 0, 0, 1, 0, 0, 0,
       0, 1, 0, 1, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 1, 1, 0, 0, 0, 1, 1,
       1, 0, 0, 1, 1, 0, 0, 0, 1, 0, 0, 0, 1, 0, 1, 0, 1, 0, 1, 0, 0, 0,
       1, 1, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 1, 0, 0, 1, 1, 1, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 1, 1, 0, 0, 0, 1, 1, 0, 1, 1,
       0, 0, 1, 0, 1, 0, 0, 1, 1, 0, 0, 1, 0, 0, 1, 1, 0, 1, 0, 0, 0, 1,
       0, 1, 0, 1, 1, 0, 0, 1, 0, 0, 1, 1, 0, 0, 1, 0, 1, 0, 0, 0, 1, 0,
       0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 1, 1, 0, 0, 0, 0, 1,
       0, 0, 1, 1, 1, 1, 1, 0, 0, 0, 0, 1, 0, 0, 1, 1, 0, 1, 0, 0, 0, 0,
       1, 0, 1, 0, 0, 1, 0, 1, 0, 1, 1, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0,
       0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0], dtype=int64)
```

Extracting Train Core Data into Xtrain

In [ ]:
```
Xtrain = dfTrain[dfTrain.columns[1:]].values
Xtrain
```

Out[ ]:
```
array([[2.026e+01, 2.303e+01, 1.324e+02, ..., 1.573e-01, 3.689e-01,
        8.368e-02],
       [1.330e+01, 2.157e+01, 8.524e+01, ..., 5.614e-02, 2.637e-01,
        6.658e-02],
       [1.222e+01, 2.004e+01, 7.947e+01, ..., 8.088e-02, 2.709e-01,
        8.839e-02],
       ...,
       [1.349e+01, 2.230e+01, 8.691e+01, ..., 1.282e-01, 2.871e-01,
        6.917e-02],
       [2.016e+01, 1.966e+01, 1.311e+02, ..., 1.425e-01, 3.055e-01,
        5.933e-02],
       [1.049e+01, 1.929e+01, 6.741e+01, ..., 3.203e-02, 2.826e-01,
        7.552e-02]])
```

Loading Test Data

In [ ]:
```
dfTest = pd.read_csv('.\External\Data\TestData.csv')
dfTest
```

Out[ ]:

	diagnosis	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_mean	cc
0	1	16.74	21.59	110.10	869.5	0.09610	
1	1	18.25	19.98	119.60	1040.0	0.09463	
2	1	20.34	21.51	135.90	1264.0	0.11700	
3	0	11.08	14.71	70.21	372.7	0.10060	
4	0	12.46	12.83	78.83	477.3	0.07372	
...	...	...	...	...	...	...	
109	1	28.11	18.47	188.50	2499.0	0.11420	
110	0	12.54	18.07	79.42	491.9	0.07436	
111	1	17.35	23.06	111.00	933.1	0.08662	
112	1	20.20	26.83	133.70	1234.0	0.09905	
113	0	14.11	12.88	90.03	616.5	0.09309	

114 rows × 31 columns

Extracting Test Classifications into Ytest

In [ ]:
```
Ytest = dfTest["diagnosis"].values
Ytest
```

Out[ ]:
```
array([1, 1, 1, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 1, 1,
       1, 0, 1, 0, 0, 1, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0,
       1, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 1, 0, 1, 1,
       0, 1, 1, 0, 1, 1, 0, 1, 0, 1, 0, 0, 1, 1, 1, 1, 0, 1, 1, 0, 1, 0,
       0, 0, 1, 1, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 1, 1, 0, 0, 0, 1,
       0, 1, 1, 0], dtype=int64)
```

Extracting Test Core Data into Xtest

In [ ]:
```
Xtest = dfTest[dfTest.columns[1:]].values
Xtest
```

Out[ ]:
```
array([[1.674e+01, 2.159e+01, 1.101e+02, ..., 1.813e-01, 4.863e-01,
        8.633e-02],
       [1.825e+01, 1.998e+01, 1.196e+02, ..., 1.932e-01, 3.063e-01,
        8.368e-02],
       [2.034e+01, 2.151e+01, 1.359e+02, ..., 2.685e-01, 5.558e-01,
        1.024e-01],
       ...,
       [1.735e+01, 2.306e+01, 1.110e+02, ..., 8.235e-02, 2.452e-01,
        6.515e-02],
       [2.020e+01, 2.683e+01, 1.337e+02, ..., 2.152e-01, 3.271e-01,
        7.632e-02],
       [1.411e+01, 1.288e+01, 9.003e+01, ..., 5.890e-02, 2.100e-01,
        7.083e-02]])
```

Loading The Coefficients

Loading Coefficients1 into W_1

In [ ]:
```
W_1 = pd.read_csv('.\External\Coefficients\Coefficients1.csv', header = None).va
print("W_1:\n" + str(W_1))
```

```
W_1:
[-0.24051694 -1.28880728 -0.36962803 -0.12141193  0.01189173  0.04758857
  0.21983976  0.32611402  0.13494155  0.08678048  0.01065719 -0.07070465
 -0.44975324 -0.27664928  0.10552783  0.00375172  0.04655467  0.06966227
  0.0171825   0.01575658  0.00419185 -1.51903808  0.46308372  0.21284544
  0.01968184  0.08338499  0.63301259  0.85841348  0.25692222  0.21309401
  0.06045903]
```

Loading Coefficients2 into W_2

In [ ]:
```
W_2 = pd.read_csv('.\External\Coefficients\Coefficients2.csv', header = None).va
print("W_2:\n" + str(W_2))
```

```
W_2:
[-0.24905703 -1.28051636 -0.35943074 -0.11699259  0.00980262  0.04166949
  0.23260228  0.33917977  0.13511828  0.07738392  0.0069529  -0.06492399
 -0.44184093 -0.27944314  0.08919083  0.00773273  0.03781426  0.08210222
  0.01682068  0.01893739 -0.01471771 -1.51735691  0.46048118  0.21775737
  0.01619036  0.08014986  0.611179    0.83697098  0.25988656  0.21811395
  0.04032636]
```

Functions

In [ ]:
```
def probabilisticLogRegClassifier(W, X):
    """
    This Function
    """
    return 1/(1+np.exp(-(X@W[1:] + W[0])))
```

In [ ]:
```
def probabilisticLogRegClassifierForMatrix(W, X):
    """
    This Function
    """
    Y = []
    for x in X:
        Y.append(probabilisticLogRegClassifier(W, x))
    return np.array(Y)
```

In [ ]:
```
def finalClassification(prb_Ypredicted_equals_one, th):
    if th > 1 or th < 0:
        print("th should be 0<=th<=1")
        return
    Y = []
    for prb_y_is_one in prb_Ypredicted_equals_one:
        Y.append(1.0 if prb_y_is_one >= th else 0.0)
    return np.array(Y)
```

In [ ]:
```
def accuracy(actualY, predictedY):
    """
    This Function
    """
    return 100*(predictedY == actualY).mean()
```

In [ ]:
```
def printAccuracy(actualY, predictedY):
    """
    This Function
    """
    print(f'{accuracy(actualY, predictedY)}%')
    return
```

In [ ]:
```
def confusionMatrix(actualY, predictedY):
    confusion_matrix = np.zeros([2, 2])
    values = [0, 1]
    for actual in values:
        for pred in values:
            confusion_matrix[actual, pred] = ((actualY == actual)*(predictedY ==
    return confusion_matrix
```

In [ ]:
```
def confusionMatrixWithThreshold(W, X, Y, th):
    final_classification = finalClassification(probabilisticLogRegClassifier(W,
    return confusionMatrix(Y, final_classification)
```

In [ ]:
```
def printConfusionMatrix(W, X, Y, th):
    confusion_matrix = confusionMatrixWithThreshold(W, X, Y, th)
    indexes = [0, 1]
    confusion_matrix = pd.DataFrame(data = confusion_matrix, columns = indexes,
    print(f'\nConfusion Matrix with th={th}\n{confusion_matrix}')
    return
```

In [ ]:
```
from sklearn.metrics import roc_curve, auc

def drawROC(actualY, prob_y_is_one, title):
    # Compute the false positive rate and true positive rate
    fpr, tpr, _ = roc_curve(actualY,  prob_y_is_one)
    # Compute the area under the curve
    roc_auc = auc(fpr, tpr)
    # Plot the ROC curve
    plt.plot(fpr, tpr, color='darkorange', label='ROC curve\n(area = %0.2f)' % r
    plt.xlim([-0.05, 1.0])
    plt.ylim([0.0, 1.05])
    plt.xlabel('FPR')
    plt.ylabel('TPR')
    plt.title(title)
    plt.legend(loc="lower right")
    plt.show()
    return
```

Metrics For Train with Coefficients1

In [ ]:
```
Y_probability_of_one = probabilisticLogRegClassifierForMatrix(W_1, Xtrain)
# print(Y_probability_of_one)
```

In [ ]:
```
th = 0.5
Ypredict_train = finalClassification(Y_probability_of_one, th)
# print(Ypredict_train)
```

Accuracy

In [ ]:
```
printAccuracy(Ytrain, Ypredict_train)
```

```
95.38461538461539%
```

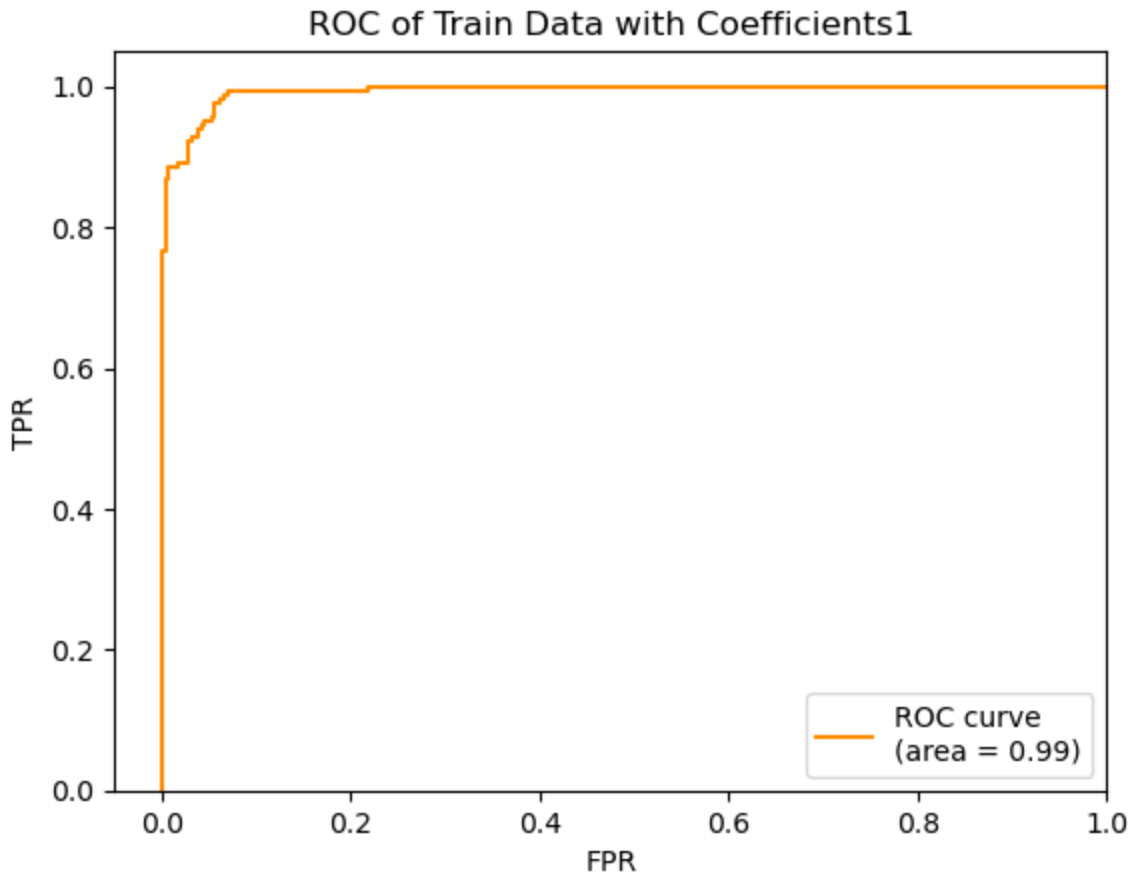
Confusion Matrix

In [ ]:
```
printConfusionMatrix(W_1, Xtrain, Ytrain, th)
```

```
Confusion Matrix with th=0.5
       0      1
0  281.0    8.0
1   13.0  153.0
```

ROC

In [ ]:
```
drawROC(Ytrain, Y_probability_of_one, 'ROC of Train Data with Coefficients1')
```



Metrics For Test with Coefficients1

In [ ]:
```
Y_probability_of_one = probabilisticLogRegClassifierForMatrix(W_1, Xtest)
# print(Y_probability_of_one)
```

In [ ]:
```
th = 0.5
Ypredict_test = finalClassification(Y_probability_of_one, th)
# print(Ypredict_test)
```

Accuracy

In [ ]:
```
printAccuracy(Ytest, Ypredict_test)
```

```
93.85964912280701%
```

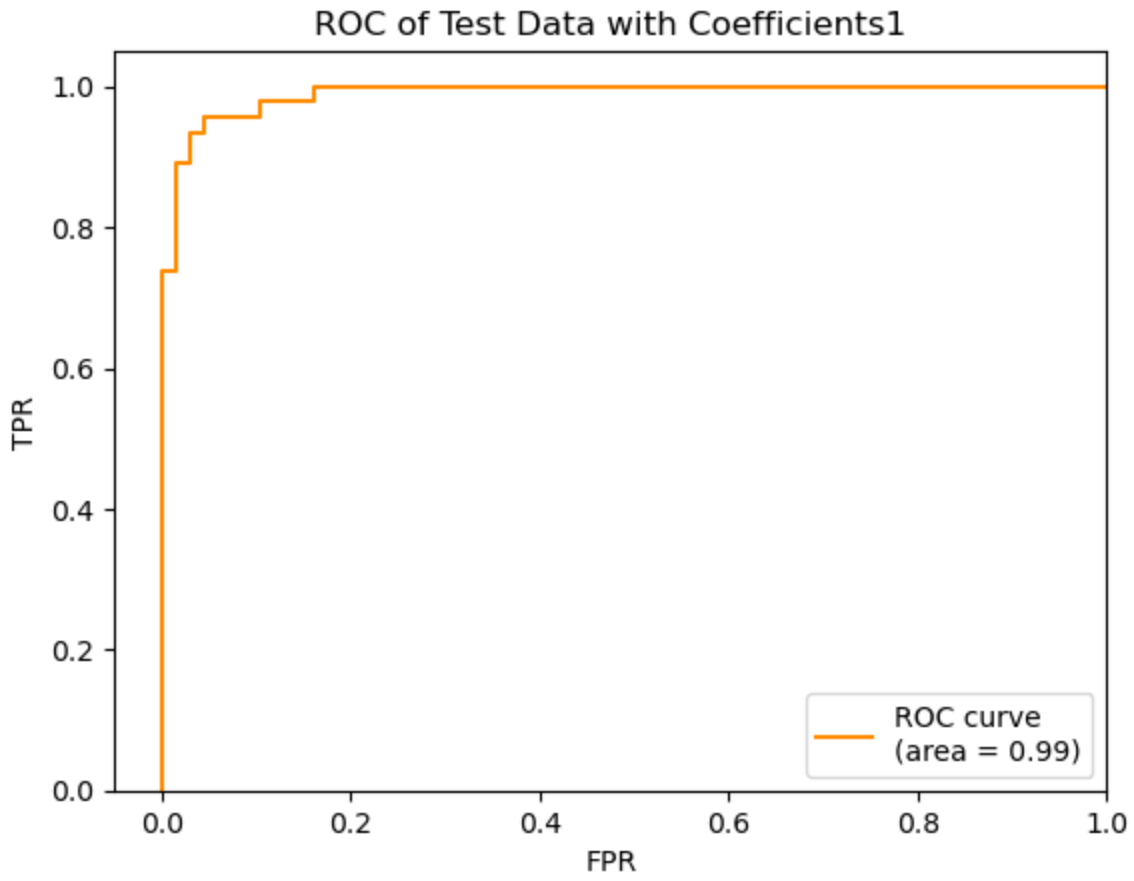
Confusion Matrix

In [ ]:
```
printConfusionMatrix(W_1, Xtest, Ytest, th)
```

```
Confusion Matrix with th=0.5
      0     1
0  66.0   2.0
1   5.0  41.0
```

ROC

In [ ]:
```
drawROC(Ytest, Y_probability_of_one, 'ROC of Test Data with Coefficients1')
```



Metrics For Train with Coefficients2

In [ ]:
```
Y_probability_of_one = probabilisticLogRegClassifierForMatrix(W_2, Xtrain)
# print(Y_probability_of_one)
```

In [ ]:
```
th = 0.5
Ypredict_train = finalClassification(Y_probability_of_one, th)
# print(Ypredict_train)
```

Accuracy

In [ ]:
```
printAccuracy(Ytrain, Ypredict_train)
```

```
89.01098901098901%
```

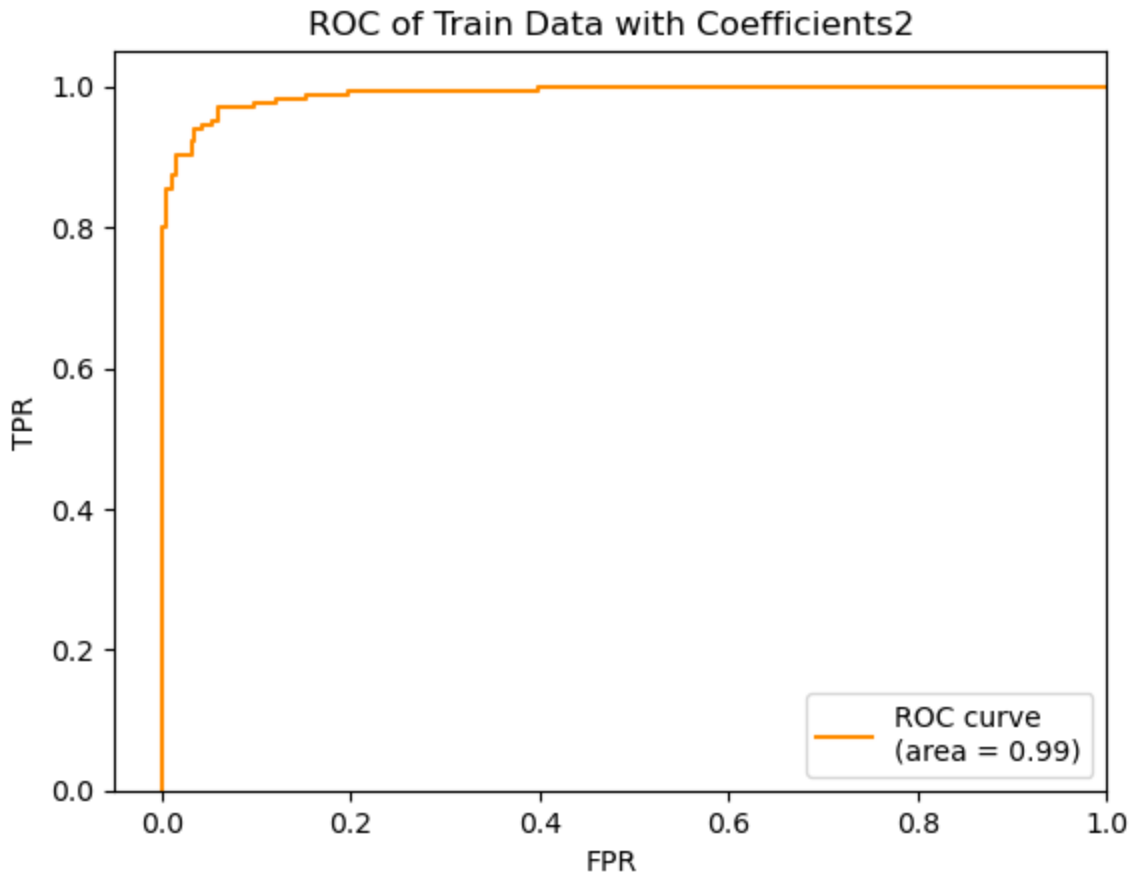
Confusion Matrix

In [ ]:
```
printConfusionMatrix(W_2, Xtrain, Ytrain, th)
```

```
Confusion Matrix with th=0.5
       0      1
0  289.0    0.0
1   50.0  116.0
```

ROC

In [ ]:
```
drawROC(Ytrain, Y_probability_of_one, 'ROC of Train Data with Coefficients2')
```



Metrics For Test with Coefficients2

In [ ]:
```
Y_probability_of_one = probabilisticLogRegClassifierForMatrix(W_2, Xtest)
# print(Y_probability_of_one)
```

In [ ]:
```
th = 0.5
Ypredict_test = finalClassification(Y_probability_of_one, th)
# print(Ypredict_test)
```

Accuracy

In [ ]:
```
printAccuracy(Ytest, Ypredict_test)
```

```
86.8421052631579%
```

Confusion Matrix

In [ ]:
```
printConfusionMatrix(W_2, Xtest, Ytest, th)
```

```
Confusion Matrix with th=0.5
      0     1
0  68.0   0.0
1  15.0  31.0
```

ROC

In [ ]:
```
drawROC(Ytest, Y_probability_of_one, 'ROC of Test Data with Coefficients2')
```