0.0.1 Q2. Role of parameters in a logistic function [10 pts]

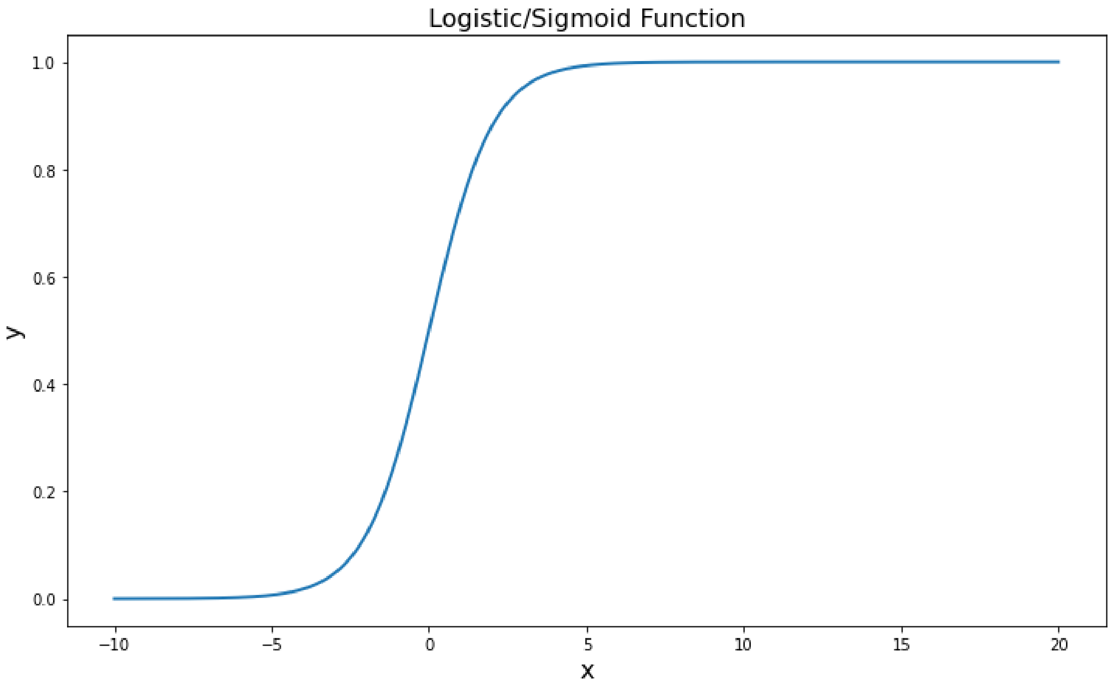
2-a) [5 points]: Generate a vector x of length N with values lying between limits Xa and Xb (for this you will have to choose your own limits; play around with different values) and apply the `gen_logistic` function to this vector. Proceed to plot the output and verify the shape of the output. If your decision boundary value is about the center of your x range, you will see an S-shape. Your final plot should show the S-curve.

```
In [35]: # TODO: change the values of N, a and b below to check how the output of your function works
         # Use a value for N greater than 1 and any limits a and b so that an S-shape graph is generate

         N = 1000
         Xa = -10
         Xb = 20
         w = 1
         b = 0


         x = np.expand_dims(np.linspace(Xa,Xb,N), axis=1)
         y = gen_logistic(x, w, b)

         fig, ax = plt.subplots(nrows=1,ncols=1,figsize=(12,7))
         ax.plot(x,y, lw=2)
         ax.set_xlabel("x", fontsize=16)
         ax.set_ylabel("y", fontsize=16)
         ax.set_title("Logistic/Sigmoid Function", fontsize=16);
```



3-c. Print out prediction probability and prediction labels from the above model (from the sklearn library) using test data. [5 pts] Explain 1) why there are two columns in the prediction probability output, and 2) how you can manually optain prediction label from the predictio probability output.

1) There are two columns because the dataset has two classes. 2) STILL NEED TO ANSWER is it pp[:,1] or pp[:,2]???

```
In [41]: pp = LogReg.predict_proba(data.x_test)
         y_pred = LogReg.predict(data.x_test)
         # print("Prediction Label: \n:", yp )

         print("\n", data.y_test)

         pd.DataFrame(confusion_matrix(data.y_test,yp,labels=[0,1]))
```

```
 [0 0 0 0 1 1 1 1 0 1 1 0 1 1 1 1 1 0 1 1 1 1 1 1 1 1 0 1 1 1 1 0 0 0 0 1 0
 1 1 0 1 1 1 0 0 1 0 1 0 1 1 1 0 1 1 1 1 1 0 1 0 0 0 0 1 1 0 1 0 0 1 0 0 1
 0 0 0 1 1 0 1 0 0 1 1 0 1 1 0 0 1 1 0 1 0 1 1 0 0 1 1 0 0 1 1 1 1 1 1 1 0
 1 1 0 0 1 1 0 1 0 1 1 1 1 1 1 1 1 0 1 1 1 1 0 1 1 1 1 1 0 1 1 1]
```

```
Out[41]:     0   1
         0  18  34
         1  30  61
```

1 Part C. Understanding classification performance metrics

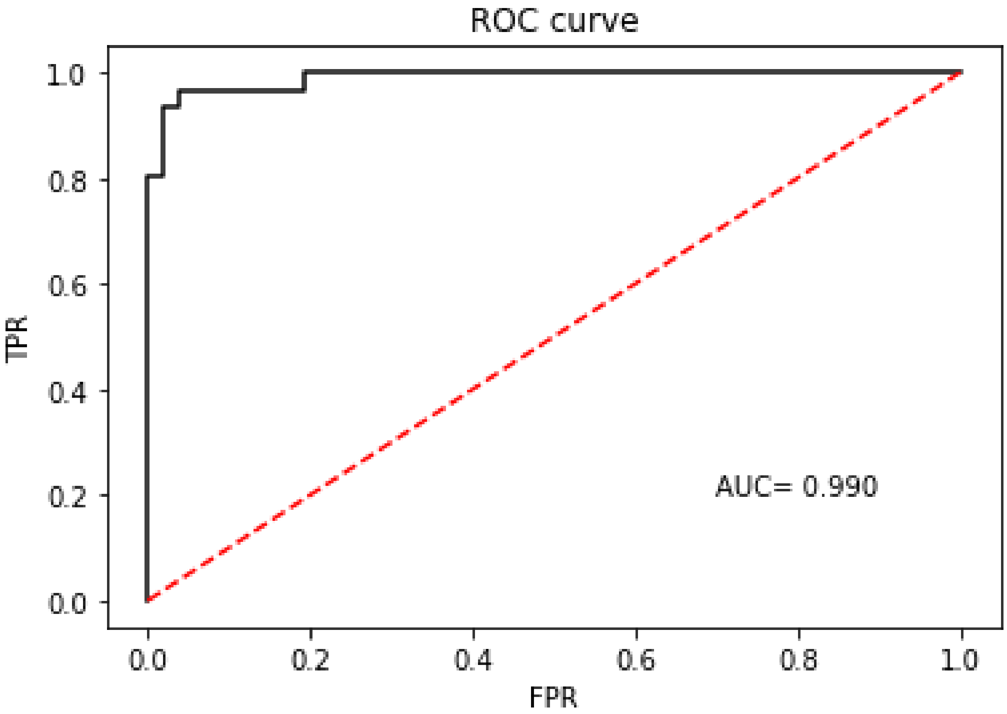
1.0.1 Q4. ROC curve [10 pts]

In the next cell, compute the ROC curve and the area under the curve and plot the ROC curve. Your ROC curve plot also should display area under the curve.
Hint: Use relevant functions in sklearn.metrics. Feel free to refer to the sklearn documentation's examples.

```
In [42]: # TODO: compute the area under the curve and plot ROC curve
         # Plot the ROC curve ( True positive rate v/s False positive rate) and indicate the AUC on the


         #fpr = false-positive rate (Fall out)
         #tpr = true positive rate (Recall, Sensitivity)
         #th = threshold to calculates spots
         #auc - area underneath the curve

         fpr,tpr,th = roc_curve(data.y_test, pp[:,1])
         auc = roc_auc_score(data.y_test, pp[:,1])
         plt.plot(fpr,tpr, 'k-')
         plt.plot(np.arange(0,1.1,0.1), np.arange(0,1.1,0.1), 'r--')
         plt.title('ROC curve')
         plt.xlabel('FPR')
         plt.ylabel('TPR')
         plt.text(0.7,0.2, 'AUC= ' + "{:.3f}".format(auc));
```


ROC curve
TPR
FPR
1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0
AUC= 0.990

1.0.2 Q6. Putting things together [10 pts]

In the next cell you will generate the predictions for the test data `data.x_test` and compute prediction and recall metrics by calling the functions you built above.

STEP1. Get weight and bias from your fitted model (sklearn model)
STEP2. Plug weight and bias and test data into your `gen_logistic` function to get prediction probability.
STEP3. From the predicion probability output from STEP2, calculate prediction label (y_pred).

```
In [47]: data.x_test[:,1].shape
```

```
Out[47]: (143,)
```

```
In [48]: weight = LogReg.coef_
         bias = LogReg.intercept_

         pp = gen_logistic(data.x_test, weight, bias)

         print(data.x_test)
         print(data.x_test.shape)


         # print(data.x_test.shape)
         # print(weights)
         # print(biase)
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
<ipython-input-48-901e6a549243> in <module>
      2 bias = LogReg.intercept_
      3
----> 4 pp = gen_logistic(data.x_test, weight, bias)
      5
      6 print(data.x_test)

<ipython-input-33-c0117f49b66e> in gen_logistic(x, w, b)
     28     y = 1 / (1 + np.exp(-z))
     29
---> 30     return y.reshape(y.shape[0],)  #y.shape[0] gets how many rows -> (row,)
     31

ValueError: cannot reshape array of size 4290 into shape (143,)
```

```
In [49]: # TO-DO : Generate predicted y values using coefficients of the fit logistic regression model
         # Then compute and print the precision and recall metrics
         # Checking your results. Do not modify codes below.
         print(y_pred.shape)
         precision = calculate_precision(data.y_test, y_pred)
         recall = calculate_recall(data.y_test, y_pred)



         print('Model Precision : %0.2f' % precision)
         print('Model Recall : %0.2f' % recall)
```

```
(143,)
Model Precision : 0.98
Model Recall : 4.00
```