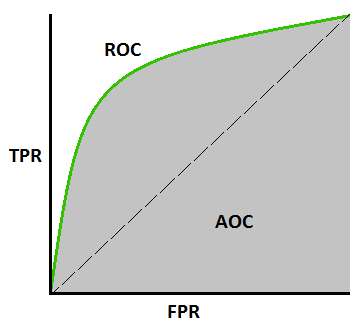
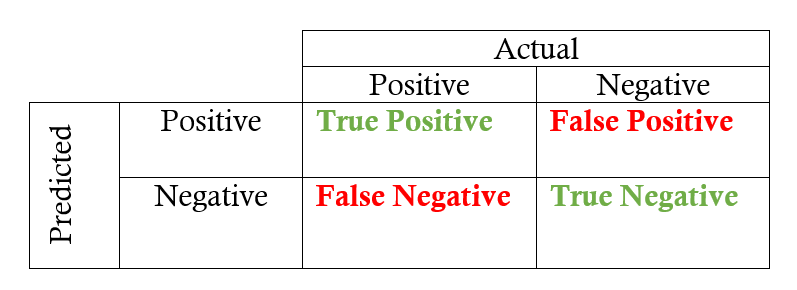
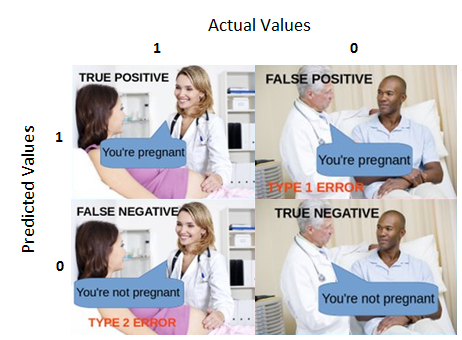
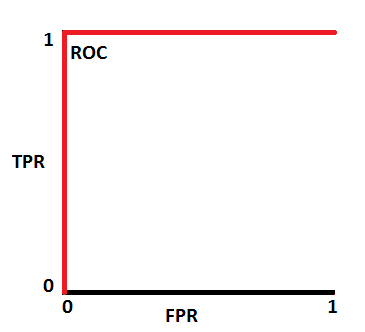
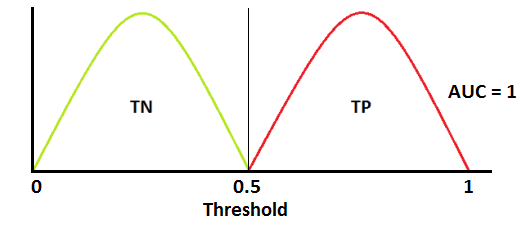
**AUC ROC Curve:**

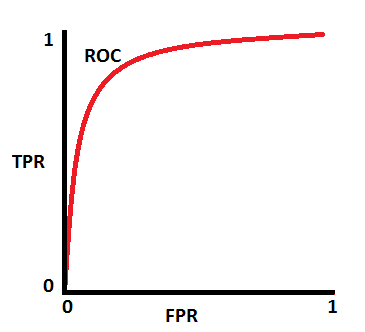
1. Used to check or visualize the performance of the multi-classification problem.
2. AUC(Area Under the Curve) ROC(Receiver Operating Characteristics) curve
3. Also written as AUROC (Area Under the Receiver Operating Characteristics)
4. ROC is a probability curve and AUC represents the degree or measure of separability
5. Higher the AUC, the better the model is at predicting 0 classes as 0 and 1 classes as 1
6. The ROC curve is plotted with TPR against the FPR where TPR is on the y-axis and FPR is on the x-axis.  
     
     
     
     
     
     
     
     
     
     
     
     
     
     
   

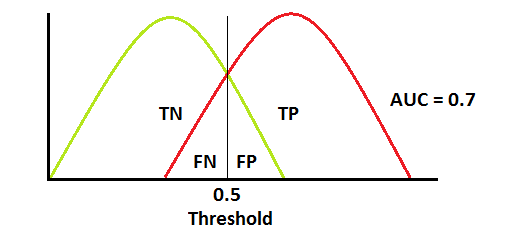
1. TPR (True Positive Rate/ Recall/ Sensitivity):
   1. TPR = TP/ (TP + FN)
2. Specificity:
   1. Specificity = TN/ (TN + FP)
3. FPR:
   1. FPR = 1 - Specificity OR FP/ (TN + FP)

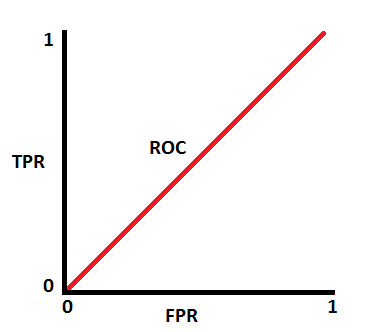


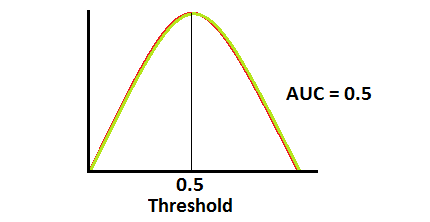
1. AUC near to 1 means better model
2. Red distribution curve is of the positive class (patients with disease) and the green distribution curve is of the negative class(patients with no disease).











* We have **roc\_curve()** from sklearn.metrics to get the roc curves.
* The AUC score can be computed using the **roc\_auc\_score()** method of sklearn:  
  auc\_score = roc\_auc\_score(y\_test, y\_pred[:,1]

**Evaluation Metrics:**

1. **Accuracy**:
   1. quintessential classification metric
   2. Accuracy = (TP+TN)/(TP+FP+FN+TN)
   3. proportion of true results among the total number of cases examined.
   4. valid choice of evaluation for classification problems which are well balanced and not skewed or No class imbalance
2. **Precision**:
   1. Answers the following question: what proportion of predicted Positives is truly Positive?
   2. Precision = (TP)/(TP+FP)
   3. valid choice of evaluation metric when we want to be very sure of our prediction
3. **Recall**:
   1. Answers a different question: what proportion of actual Positives is correctly classified?
   2. Recall = (TP)/(TP+FN)
   3. valid choice of evaluation metric when we want to capture as many positives as possible
   4. For example: If we are building a system to predict if a person has cancer or not, we want to capture the disease even if we are not very sure.
4. **F1 Score**:
   1. It is a number between 0 and 1 and is the harmonic mean of precision and recall.
   2. F1 = 2 \* (precision \* recall / precision + recall)
   3. It maintains a balance between the precision and recall for your classifier
   4. *If you are a police inspector and you want to catch criminals, you want to be sure that the person you catch is a criminal (Precision) and you also want to capture as many criminals (Recall) as possible. The F1 score manages this tradeoff.*
   5. How to use:
      1. from sklearn.metrics import f1\_score  
         y\_true = [0, 1, 1, 0, 1, 1]  
         y\_pred = [0, 0, 1, 0, 0, 1]  
         f1\_score(y\_true, y\_pred)
   6. Here is one function to get the best threshold for maximizing F1 score for binary predictions. The below function iterates through possible threshold values to find the one that gives the best F1 score.
      1. # y\_pred is an array of predictions  
         def bestThresshold(y\_true,y\_pred):  
          best\_thresh = None  
          best\_score = 0  
          for thresh in np.arange(0.1, 0.501, 0.01):  
          score = f1\_score(y\_true, np.array(y\_pred)>thresh)  
          if score > best\_score:  
          best\_thresh = thresh  
          best\_score = score  
          return best\_score , best\_thresh
   7. F1 score gives equal weight to precision and recall. Sometimes we want to have more recall or more precision. We can resolve this by creating a weighted F1 metric by giving beta value which manages tradeoff between precision and recall
      1. from sklearn.metrics import fbeta\_score  
         y\_true = [0, 1, 1, 0, 1, 1]  
         y\_pred = [0, 0, 1, 0, 0, 1]  
         fbeta\_score(y\_true, y\_pred,beta=0.5)
5. **Log Loss/Binary Cross Entropy**:
   1. Binary cross entropy compares each of the predicted probabilities to actual class output which can be either 0 or 1.
   2. It then calculates the score that penalizes the probabilities based on the distance from the expected value. That means how close or far from the actual value.
   3. Definition: Binary Cross Entropy is the negative average of the log of corrected predicted probabilities.
   4. When to use: When the output of a classifier is prediction probabilities. Log Loss takes into account the uncertainty of your prediction based on how much it varies from the actual label