**Expt-4**

**Aim: Generate a Confusion Matrix and compute true positive, true negative, false positive, and false negative.**

**Program to generate confusion Matrix and classification report**

**from sklearn.metrics import confusion\_matrix**

**from sklearn.metrics import classification\_report**

**from sklearn import metrics**

**import matplotlib.pyplot as plt**

**from sklearn.metrics import accuracy\_score**

**# actual values**

**#A=1= Positive Class , B=0=Negative Class**

**actual = [1,0,0,1,0,1,1,1,0,1,0]**

**# predicted values**

**predicted = [1,0,0,1,0,0,0,1,0,0,1]**

**# confusion matrix**

**matrix = confusion\_matrix(actual,predicted, labels=[1,0])**

**print('Confusion matrix : \n',matrix)**

**acc=accuracy\_score(actual,predicted)**

**print('Accuracy = ',acc)**

**matrix = classification\_report(actual,predicted,labels=[1,0])**

**print('Classification Report \n')**

**print(matrix)**

**fpr, tpr , \_= metrics.roc\_curve(actual, predicted) #create ROC curve**

**print('fpr = ',fpr)**

**print('tpr = ',tpr)**

**plt.plot(fpr,tpr)**

**plt.ylabel('True Positive Rate')**

**plt.xlabel('False Positive Rate')**

**plt.show()**

**OUTPUT**

**Confusion matrix :**

**[[3 3]**

**[1 4]]**

**Accuracy = 0.6363636363636364**

**Classification Report**

**precision recall f1-score support**

**1 0.75 0.50 0.60 6**

**0 0.57 0.80 0.67 5**

**accuracy 0.64 11**

**macro avg 0.66 0.65 0.63 11**

**weighted avg 0.67 0.64 0.63 11**

**fpr = [0. 0.2 1. ]**

**tpr = [0. 0.5 1. ]**

A blue line graph with numbers

Description automatically generated

**Assignment:**

1. **Verify theoretically the entries of the classification report.**

**Note:**

1. **Experiment with the following actual and predicted samples and verify the entries of the classification report.**

**# actual values**

**#A=1=Positive Class , B=0=Negative Class**

**actual = [1,0,0,1,0,1,1,1,0,1,0,1,1,1,1,0,0,1]**

**# predicted values**

**predicted = [1,0,0,1,0,0,0,1,0,0,1,0,0,0,0,0,1,0]**