Compute performance metrics for the given Y and Y_score without sklearn

In [71]:

```
import numpy as np
import pandas as pd
# other than these two you should not import any other packages
```

A. Compute performance metrics for the given data **5_a.csv**
Note 1: in this data you can see number of positive points >> number of negatives points
Note 2: use pandas or numpy to read the data from **5_a.csv**
Note 3: you need to derive the class labels from given score

$$y^{pred} = [0 \text{ if y\_score} < 0.5 \text{ else } 1]$$

1. Compute Confusion Matrix

2. Compute F1 Score

3. Compute AUC Score, you need to compute different thresholds and for each threshold compute tpr,fpr and then use numpy.trapz(tpr_array, fpr_array) https://stackoverflow.com/q/53603376/4084039 (https://stackoverflow.com/q/53603376/4084039), https://stackoverflow.com/a/39678975/4084039 (https://stackoverflow.com/a/39678975/4084039) Note: it should be numpy.trapz(tpr_array, fpr_array) not numpy.trapz(fpr_array, tpr_array)

4. Compute Accuracy Score

In [2]:

```
# write your code here
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt

def compute(y_act,y_pred):

    """
    True Positive - actual = 1, predicted = 1
    False Positive - actual =1, predicted = 0
    False Negative - actual =0,predicted = 1
    True Negative  - actual = 0, predicted = 0
    """
    tp = sum((y_act == 1) & (y_pred == 1)) #Calculate True Positive
    tn = sum((y_act == 0) & (y_pred == 0)) #Calculate True Negative
    fn = sum((y_act == 1) & (y_pred == 0)) #Calculate False Negative
    fp = sum((y_act == 0) & (y_pred == 1)) #Calculate False Positive
    return tp,tn,fn,fp


def compute_accuracy(tp,tn,fn,fp):
    """
    Accuracy
    """
    return ((tp + tn) * 100) / float(tp+tn+fn+fp) #Calculate Accuracy

def compute_precision(tp,fp):
    """
    Precision
    """
    return (tp * 100)/ float(tp+fp)  #Calculate Precision

def compute_recall(tp,fn):
    """
    Recall
    """
    return (tp * 100)/ float(tp+fn)  #Calculate Recall

def tptn(y_act,y_prob):
    """
    This function returns TPR and FPR
    """
    threshold = np.random.uniform(0,1,size =10000)  #Randomly Selecting values between
 [0,1]
    reverse_order = np.sort(threshold)[::-1]  #Sorted values by descending order
    tpr = []
    fpr = []
    for i in reverse_order :                      #Iterating the loop
        df[i]= ( y_prob >= i ).astype('int')    #Based on threshold value to classify
        tp,tn,fn,fp = compute(y_act,df[i])       #Call the compute function
        tpr.append((compute_recall(tp,fn))/100)  #Append all True Positive rates to lis
t
        fpr.append(compute_falsepositive(fp,tn))  #Append all False Positive rates to l
ist
    return tpr,fpr

def compute_falsepositive(fp,tn):
    """
    False Positive Rate
    """
    return (fp)/ float(tn+fp)  #Calculate False Positie rate

def compute_f1_score(y_act,y_pred):
    """
    Calcualte F1 Score
    """
    tp,tn,fn,fp = compute(y_act,y_pred)         #Call the compute function
    precision = compute_precision(tp,fp)/100    #Call the precision function
    recall = compute_recall(tp,fn)/100         #Call the recall function
    f1_score = (2*precision*recall)/(recall+precision) #Calculate f1 score
    return f1_score

df = pd.read_csv("5_a.csv")                    #Read the CSV file stored into df
df['y_predicted']= (df['proba'] >= 0.5).astype('int')   #Based on threshold value to cl
assify
df_confusion = pd.crosstab(df['y'], df['y_predicted'],margins=True)  #computing confusi
on matrix
print("Confusion Matrix :")
print(df_confusion)
y_prob = df['proba']
y_act = df['y']
y_pred = df['y_predicted']
tp,tn,fn,fp = compute(y_act,y_pred)
Accuracy = compute_accuracy(tp,tn,fn,fp)  #computing accuracy
precision = compute_precision(tp,fp)  #computing precision
recall = compute_recall(tp,fn)        #Computing recall
f1_score = compute_f1_score(y_act,y_pred)  #Calling F1 Score
print("f1 score is " + str(f1_score))
tparray,fparray = tptn(y_act,y_prob)   #calculating tpr,fpr
a= np.asarray(tparray, dtype = float)
b= np.asarray(fparray, dtype = float)
AUC = np.trapz(a, b)            #Calculating Area under Curve
print("AUC Score is " + str(AUC))
print("Accuracy score  is " + str(Accuracy))
```

```
Confusion Matrix :
y_predicted      1    All
y
0.0            100    100
1.0          10000  10000
All          10100  10100
f1 score is 0.9950248756218906
AUC Score is 0.488306
Accuracy score  is 99.00990099009901
```

B. Compute performance metrics for the given data **5_b.csv**

Note 1: in this data you can see number of positive points << number of negatives points

Note 2: use pandas or numpy to read the data from **5_b.csv**

Note 3: you need to derive the class labels from given score

$$y^{pred} = [\text{0 if y\_score < 0.5 else 1}]$$

1. Compute Confusion Matrix

2. Compute F1 Score

3. Compute AUC Score, you need to compute different thresholds and for each threshold compute tpr,fpr and then use numpy.trapz(tpr_array, fpr_array) https://stackoverflow.com/q/53603376/4084039 (https://stackoverflow.com/q/53603376/4084039), https://stackoverflow.com/a/39678975/4084039 (https://stackoverflow.com/a/39678975/4084039)

4. Compute Accuracy Score

In [3]:

```
# write your code# write your code here
import pandas as pd
import numpy as np

def compute(y_act,y_pred):

    """
    True Positive - actual = 1, predicted = 1
    False Positive - actual =1, predicted = 0
    False Negative - actual =0,predicted = 1
    True Negative  - actual = 0, predicted = 0
    """
    tp = sum((y_act == 1) & (y_pred == 1)) #Calculate True Positive
    tn = sum((y_act == 0) & (y_pred == 0)) #Calculate True Negative
    fn = sum((y_act == 1) & (y_pred == 0)) #Calculate False Negative
    fp = sum((y_act == 0) & (y_pred == 1)) #Calculate False Positive
    return tp,tn,fn,fp

def compute_accuracy(tp,tn,fn,fp):
    """
    Accuracy
    """
    return ((tp + tn) * 100) / float(tp+tn+fn+fp)  #Calculate Accuracy

def compute_precision(tp,fp):
    """
    Precision
    """
    return (tp * 100)/ float(tp+fp) #Calculate precision

def compute_falsepositive(fp,tn):
    """
    False Positive Rate
    """
    return (fp)/ float(tn+fp) #Calculate False positive

def compute_recall(tp,fn):
    """
    Recall
    """
    return (tp * 100)/ float(tp+fn)  #Calculate Recall

def compute_f1_score(y_act,y_pred):
    """
    Calcualte F1 Score
    """
    tp,tn,fn,fp = compute(y_act,y_pred)         #Call the compute function
    precision = compute_precision(tp,fp)/100    #Call the precision function
    recall = compute_recall(tp,fn)/100         #Call the recall function
    f1_score = (2*precision*recall)/(recall+precision) #Calculate f1 score
    return f1_score

def tptn(y_act,y_prob):
    """
    This function returns TPR and FPR
    """
    threshold = np.random.uniform(0,1,size =10000)  #Randomly Selecting values between
 [0,1]
    reverse_order = np.sort(threshold)[::-1]  #Sorted values by descending order
    tpr = []
    fpr = []
    for i in reverse_order :                        #Iterating the loop
        df[i]= ( y_prob >= i ).astype('int')    #Based on threshold value to classify
        tp,tn,fn,fp = compute(y_act,df[i])       #Call the compute function
        tpr.append((compute_recall(tp,fn))/100)  #Append all True Positive rates to lis
t
        fpr.append(compute_falsepositive(fp,tn))  #Append all False Positive rates to l
ist
    return tpr,fpr


df = pd.read_csv("5_b.csv")   #Read the CSV file stored into df
df['y_predicted']= (df['proba'] >= 0.5).astype('int')  #Based on threshold value to cla
ssify
df_confusion = pd.crosstab(df['y'], df['y_predicted'],margins=True) #computing confusio
n matrix
print("Confusion Matrix :")
print(df_confusion)
y_prob = df['proba']
y_act = df['y']
y_pred = df['y_predicted']
tp,tn,fn,fp = compute(y_act,y_pred)    #Computing tp,tn,fp,fn
Accuracy = compute_accuracy(tp,tn,fn,fp)    #Computing Accuracy
precision = compute_precision(tp,fp)  #Computing precision
recall = compute_recall(tp,fn)   #Compute recall
f1_score = compute_f1_score(y_act,y_pred)  #Compute f1 score
print("f1 score is " + str(f1_score))
tparray,fparray = tptn(y_act,y_prob)  #Calculate tpr,fpr
a= np.asarray(tparray, dtype = float)
b= np.asarray(fparray, dtype = float)
AUC = np.trapz(a, b)     #Calculate Area under Curve
print("AUC Score is " + str(AUC))
print("Accuracy score  is " + str(Accuracy))
```

```
Confusion Matrix :
y_predicted     0    1    All
y
0.0          9761  239  10000
1.0            45   55    100
All          9806  294  10100
f1 score is 0.27918781725888325
AUC Score is 0.9377635
Accuracy score  is 97.18811881188118
```

C. Compute the best threshold (similarly to ROC curve computation) of probability which gives lowest values of metric **A** for the given data **5_c.csv**

you will be predicting label of a data points like this: $y^{pred} = [0 \text{ if y\_score} < \text{threshold else } 1]$

$$A = 500 \times \text{number of false positives} + 100 \times \text{numebr of false negatives}$$

```
    Note 1: in this data you can see number of positive points < number of positi
ve points
    Note 2: use pandas or numpy to read the data from 5_c.csv
```

In [4]:

```
# write your code
def compute(y_act,y_pred):

    """
    True Positive - actual = 1, predicted = 1
    False Positive - actual =1, predicted = 0
    False Negative - actual =0,predicted = 1
    True Negative  - actual = 0, predicted = 0
    """
    tp = sum((y_act == 1) & (y_pred == 1)) #Calculate True Positive
    tn = sum((y_act == 0) & (y_pred == 0)) #Calculate True Negative
    fn = sum((y_act == 1) & (y_pred == 0)) #Calculate False Negative
    fp = sum((y_act == 0) & (y_pred == 1)) #Calculate False Positive
    return tp,tn,fn,fp

df = pd.read_csv("5_c.csv")   #Read the CSV file stored into df
y_prob = df['prob']
y_act = df['y']
threshold = np.random.uniform(0,1,size =10000) #Randomly Selecting threshold values
setin = {}
for i in threshold:
    df[i]= ( y_prob >= i ).astype('int')   #Based on threshold to predict
    tp,tn,fn,fp = compute(y_act,df[i])   #Compute tp,tn,fn,fp
    A = 500*fp + 100 * fn      # Calcualting function A = 500 ×number of false positives
+100×numebr of false negatives
    setin[i] = A
sorted_x = sorted(setin.items(), key=lambda kv: kv[1])  #Sorting ascending order based
 on Value A
print(sorted_x[0][0])  #Printing best threshod value with corresponding lowest values o
f metrics
```

```
0.6432476012102228
```

D. Compute performance metrics(for regression) for the given data **5_d.csv**

Note 2: use pandas or numpy to read the data from **5_d.csv**

Note 1: **5_d.csv** will having two columns Y and predicted_Y both are real valued features

1. Compute Mean Square Error

2. Compute MAPE: https://www.youtube.com/watch?v=ly6ztgIkUxk

3. Compute R^2 error: https://en.wikipedia.org/wiki/Coefficient_of_determination#Definitions

In [5]:

```
def mse(y, y_pred):
    """
    This function returns Mean Square Error
    """
    return np.square(y - y_pred).mean()  # Mean of the Square of the difference of Pred
icted and Actual value

def mape(y, y_pred):
    """
    This  function return mape
    """
    return np.sum(np.abs(y-y_pred))/np.sum(y)  #Sum of absolute values of predicted and
actual value/sum of actual values

def rse(y, y_pred):
    """
    This function return r square value
    """
    ymean = np.mean(y)                         #Calculating the mean of actual value
    ss_tot = np.sum(np.square(y-ymean))        #Calculating ss_total
    ss_res = np.sum(np.square(y-y_pred))  #  Refer this document for calcualation  http
s://en.wikipedia.org/wiki/Coefficient_of_determination#Definitions
    return 1-(ss_res/ss_tot)              #Calculating Coefficent of Determination

df = pd.read_csv("5_d.csv")
mse = mse(df['y'],df['pred'])     #Calling mean square error function
mmape = mape(df['y'],df['pred'])   #Calling mape function
r2 = rse(df['y'],df['pred'])      #Calling R square function
print("Mean Square Error is " + str(mse))
print("MAPE is " + str(mmape))
print("R2 Square Error is "  + str(r2))
```

```
Mean Square Error is 177.16569974554707
MAPE is 0.1291202994009687
R2 Square Error is 0.9563582786990937
```

Observation :

1. For imbalanced datasets,accuracy does not matter only it depends on AUC Score to find better classification techhnique
2. As observed in 5a_csv, no of positive points is much greater than negative points,AUC Score getting 0.488306
3. As observed in 5b_csv, no of negative points is much greater than positive points,AUC Score getting 0.9377635
4. Compared to 2 & 3, 5b_csv(3) is good for classification
5. A Perfect classifier has 100% true positive rate and 0% false positive rate (0 false positive).
6. R2 Square error must be in between 0 to 1 .For mean Square Error there is no limit