Compute performance metrics for the given Y and Y_score without sklearn

In [1]:

```
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]:

```
## Loading data

data=pd.read_csv("5_a.csv")
data.head(5)


## defining a function to calculate confusion matrix, F1 score and accuracy

def performance(data):

    ## mapping the probability values to class labels

    data['y_predicted'] = [0.0 if x <0.5 else 1.0 for x in data['proba']]

    ## finding confusion matrix
    ## initializing confusion matrix elements and running loop to get all values(

    TP,TN,FN,FP=0,0,0,0
    for i in range(len(data['y'])):
        if data['y'][i]==1 and data['y_predicted'][i]==1:
            TP+=1
        elif data['y'][i]==0 and data['y_predicted'][i]==0:
            TN+=1

        elif data['y'][i]==0 and data['y_predicted'][i]==1:
            FP+=1
        elif data['y'][i]==1 and data['y_predicted'][i]==0:
            FN+=1

    confusion_matrix=[[TN,FN],[FP,TP]]

## Finding precision and recall

    total_positive,total_negative=data['y'].value_counts()
    precision=TP/(TP+FP)
    recall=TP/(TP+FN)

## Finding F1 score

    F_1_score=2*((precision*recall)/(precision+recall))

## finding accuracy

    accuracy=(TP + TN)/(TP + TN + FP + FN)

## printing the outputs

    print('Accurarcy : ',accuracy,'\n\n'+'F1 score : ',F_1_score,'\n\n'+'confusi
          confusion_matrix[0],'\n',confusion_matrix[1])



## calling the above defined function to printing the output

performance(data)
```

```
Accurarcy :  0.9900990099009901

F1 score :  0.9950248756218906

confusion matrix
 [0, 0]
 [100, 10000]
```

In [3]:

```
## calculating AUC Score

from tqdm import tqdm
unique_probability=(data['proba'].round(decimals=2)).unique()
list(unique_probability)
unique_probability.sort()
n_thresholds=list(unique_probability)
n_thresholds.reverse()
n_thresholds=n_thresholds


## comparing with different values of thresholds
TPR,FPR=[],[]
for i in tqdm(range(len(n_thresholds))):
    threshold=n_thresholds[i]

    data['y_predicted'] = [0.0 if x <threshold else 1.0 for x in data['proba']]




    TP,TN,FN,FP=0,0,0,0
    for i in range(len(data['y'])):
        if data['y'][i]==1 and data['y_predicted'][i]==1:
            TP+=1
        elif data['y'][i]==0 and data['y_predicted'][i]==0:
            TN+=1

        elif data['y'][i]==0 and data['y_predicted'][i]==1:
            FP+=1
        elif data['y'][i]==1 and data['y_predicted'][i]==0:
            FN+=1
    tpr=TP/(FN+TP)
    fpr=FP/(TN+FP)
    TPR.append(tpr)
    FPR.append(fpr)

## Finding the value of AUC

tpr_array=np.array(TPR)
fpr_array=np.array(FPR)

AUC_Score=np.trapz(tpr_array, fpr_array)

print('AUC score : ',AUC_Score)
```

```
100%|██████████| 41/41 [01:05<00:00,  1.60s/it]

AUC score :  0.4875514999999999
```

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

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

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)

4. Compute Accuracy Score

In [4]:

```
## Loading data

data=pd.read_csv("5_b.csv")
data.head(5)


## calling the above defined function to printing the output

performance(data)
```

```
Accurarcy :  0.9718811881188119

F1 score :  0.2791878172588833

confusion matrix
 [9761, 45]
 [239, 55]
```

In [5]:

```
## calculating AUC Score

from tqdm import tqdm
unique_probability=(data['proba'].round(decimals=2)).unique()
list(unique_probability)
unique_probability.sort()
n_thresholds=list(unique_probability)
n_thresholds.reverse()
n_thresholds=n_thresholds


## comparing with different values of thresholds
TPR,FPR=[],[]
for i in tqdm(range(len(n_thresholds))):
    threshold=n_thresholds[i]

    data['y_predicted'] = [0.0 if x <threshold else 1.0 for x in data['proba']]




    TP,TN,FN,FP=0,0,0,0
    for i in range(len(data['y'])):
        if data['y'][i]==1 and data['y_predicted'][i]==1:
            TP+=1
        elif data['y'][i]==0 and data['y_predicted'][i]==0:
            TN+=1

        elif data['y'][i]==0 and data['y_predicted'][i]==1:
            FP+=1
        elif data['y'][i]==1 and data['y_predicted'][i]==0:
            FN+=1
    tpr=TP/(FN+TP)
    fpr=FP/(TN+FP)
    TPR.append(tpr)
    FPR.append(fpr)

## Finding the value of AUC

tpr_array=np.array(TPR)
fpr_array=np.array(FPR)

AUC_Score=np.trapz(tpr_array, fpr_array)

print('AUC score : ',AUC_Score)
```

```
100%|██████████| 51/51 [01:16<00:00,  1.49s/it]

AUC score :  0.9372849999999999
```

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

$$A = 500 \times \text{number of false negative} + 100 \times \text{numebr of false positive}$$

Note 1: in this data you can see number of negative points > number of positive points

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

In [6]:

```
## Loading data
data=pd.read_csv('5_c.csv')

thresholds=(data['prob'].round(decimals=3)).unique()
list1=list(thresholds)
list1.sort()



dict={}
for i in tqdm(range(len(list1))):

    data['y_predicted'] = [0.0 if x <list1[i] else 1.0 for x in data['prob']]

    FN,FP=0,0
    for j in range(len(data['y'])):
        if data['y'][j]==0 and data['y_predicted'][j]==1:
            FP+=1
        elif data['y'][j]==1 and data['y_predicted'][j]==0:
            FN+=1
    A=(500*FN)+(100*FP)
    dict[list1[i]]=A


min_A = min(dict.values())
min_threshold = [key for key in dict if dict[key] == min_A]
print("Threshold for minimum value of A : " + str(min_threshold))
```

```
100%|██████████| 782/782 [04:08<00:00,  3.15it/s]

Threshold for minimum value of A : [0.23]
```

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 [7]:

```
## Loading data
data=pd.read_csv('5_d.csv')

## calculating mean square error

MSE=0
for i in range(len(data['y'])):
    diff=data['y'][i]-data['pred'][i]
    diff=diff**2
    MSE+=diff

MSE=MSE/len(data['y'])
print('Mean square error : ',MSE)


## calculating mean of actual values and putting for zero values and calculating
MAPE=0


for i in range(len(data['y'])):
    diff=(data['y'][i]-data['pred'][i])
    MAPE+=abs(diff)

MAPE=MAPE/data['y'].sum()
print('Mean absolute percentage error : ',MAPE)

## calculating Total sum of square for calculating R^2
avg=data['y'].mean()
TSS=0
for i in range(len(data['y'])):
    diff=(data['y'][i]-avg)
    TSS+=diff**2



## using above calculated values and putting in formula of R^2 value

R_squared=1-(MSE/TSS)

print('R^2 value is : ',R_squared)
```

```
Mean square error :  177.16569974554707
Mean absolute percentage error :  0.1291202994009687
R^2 value is :  0.9999997223809077
```