Compute performance metrics for the given Y and Y_score without sklearn

In [2]:
```
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/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 [155…
```
data = pd.read_csv('5_a.csv')
dataAUC = pd.read_csv('5_a.csv')

data.sort_values(by='proba', ascending=True , inplace = True) #sort data by proba in de

data['proba'] = np.where(data['proba'] >= 0.5, 1, data['proba'])    #replacing proba wi
data['proba'] = np.where(data['proba'] < 0.5, 0, data['proba'])     #replacing proba wi

data['y'] = data['y'].astype(int)                    #convering to int
data['proba'] = data['proba'].astype(int)            #convering to int


data_TN = data.query('y == 0 & proba == 0')          #filtering TN
data_FN = data.query('y == 1 & proba == 0')          #filtering FN
data_FP = data.query('y == 0 & proba == 1')          #filtering FP
data_TP = data.query('y == 1 & proba == 1')          #filtering TP

TN = data_TN.shape[0]                                #Getting number of records
FN = data_FN.shape[0]
```

```
FP = data_FP.shape[0]
TP = data_TP.shape[0]

print("Confusion Matrix : TN = {},FN = {}, FP = {} TP = {} ".format(TN , FN , FP , TP))

pr = TP/(TP+FP)   #Calculating Precision
re = TP/(TP+FN)   #Calculating Recall

f1 = (2*pr*re)/(pr+re)   #Calculating F1 Score

acc = (TN+TP)/(TP+FP+FN+TN)  #Calculating Accuracy Score

print("F1 Score = {}".format(round(f1,3)))   #printing F1 Score
print("Accuracy = {}".format(round(acc,3)))  #printing Accuracy Score



### For AUC Score
dataAUC.sort_values(by='proba', ascending=False , inplace = True) #sorting in ascending

arr_proba = np.array(dataAUC.proba.unique()) # array of unique probabilities

TPR_Final =  []
FPR_Final  = []


for i in arr_proba:
    data_new_ite = dataAUC.copy()   #used copy to point to different memory location ,

    data_new_ite['proba'] = np.where(data_new_ite['proba'] >= i, 1.0, data_new_ite['pro
    data_new_ite['proba'] = np.where(data_new_ite['proba'] < i, 0.0, data_new_ite['prob

    data_P = data_new_ite.query('y == 1.0')
    data_N = data_new_ite.query('y == 0.0')
    data_FP = data_new_ite.query('y == 0.0 & proba == 1.0')
    data_TP = data_new_ite.query('y == 1.0 & proba == 1.0')

    P = data_P.shape[0]
    N = data_N.shape[0]
    FP = data_FP.shape[0]
    TP = data_TP.shape[0]

    TPR = TP/P
    FPR = FP/N

    TPR_Final.append(TPR)   #creating list of TPR for every threshold
    FPR_Final.append(FPR)   #creating list of FPR for every threshold

AUCscore = np.trapz(TPR_Final, FPR_Final)  #calculation AUC Score
print("AUC score = {}".format(round(AUCscore,3)))
```

```
Confusion Matrix : TN = 0,FN = 0, FP = 100 TP = 10000
F1 Score = 0.995
Accuracy = 0.99
AUC score = 0.488
```

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/a/39678975/4084039

4. Compute Accuracy Score

In [156…

```
data = pd.read_csv('5_b.csv')
dataAUC = pd.read_csv('5_b.csv')

data.sort_values(by='proba', ascending=True , inplace = True) #sort data by proba in as

data['proba'] = np.where(data['proba'] >= 0.5, 1, data['proba'])    #replacing proba wi
data['proba'] = np.where(data['proba'] < 0.5, 0, data['proba'])     #replacing proba wi

data['y'] = data['y'].astype(int)                    #convering to int
data['proba'] = data['proba'].astype(int)            #convering to int


data_TN = data.query('y == 0 & proba == 0')          #filtering TN
data_FN = data.query('y == 1 & proba == 0')          #filtering FN
data_FP = data.query('y == 0 & proba == 1')          #filtering FP
data_TP = data.query('y == 1 & proba == 1')          #filtering TP

TN = data_TN.shape[0]                                #Getting number of records
FN = data_FN.shape[0]
FP = data_FP.shape[0]
TP = data_TP.shape[0]

print("Confusion Matrix : TN = {},FN = {}, FP = {} TP = {} ".format(TN , FN , FP , TP))

pr = TP/(TP+FP)   #Calculating Precision
re = TP/(TP+FN)   #Calculating Recall

f1 = (2*pr*re)/(pr+re)   #Calculating F1 Score

acc = (TN+TP)/(TP+FP+FN+TN)  #Calculating Accuracy Score

print("F1 Score = {}".format(round(f1,3)))   #printing F1 Score
print("Accuracy = {}".format(round(acc,3)))  #printing Accuracy Score



### For AUC Score
```

```
dataAUC.sort_values(by='proba', ascending=False , inplace = True) #sorting in descendin

arr_proba = np.array(dataAUC.proba.unique()) # array of unique probabilities

TPR_Final =  []
FPR_Final  = []


for i in arr_proba:
    data_new_ite = dataAUC.copy()   #used copy to point to different memory location ,

    data_new_ite['proba'] = np.where(data_new_ite['proba'] >= i, 1.0, data_new_ite['pro
    data_new_ite['proba'] = np.where(data_new_ite['proba'] < i, 0.0, data_new_ite['prob

    data_P = data_new_ite.query('y == 1.0')
    data_N = data_new_ite.query('y == 0.0')
    data_FP = data_new_ite.query('y == 0.0 & proba == 1.0')
    data_TP = data_new_ite.query('y == 1.0 & proba == 1.0')

    P = data_P.shape[0]
    N = data_N.shape[0]
    FP = data_FP.shape[0]
    TP = data_TP.shape[0]

    TPR = TP/P
    FPR = FP/N

    TPR_Final.append(TPR)   #creating list of TPR for every threshold
    FPR_Final.append(FPR)   #creating list of FPR for every threshold

AUCscore = np.trapz(TPR_Final, FPR_Final)  #calculation AUC Score
print("AUC score = {}".format(round(AUCscore,3)))
```

```
Confusion Matrix : TN = 9761,FN = 45, FP = 239 TP = 55
F1 Score = 0.279
Accuracy = 0.972
AUC score = 0.938
```

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

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

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 [16]:
```
dataAUC = pd.read_csv('5_c.csv')
dataAUC.sort_values(by='prob', ascending=False , inplace = True) #sorting in descending

arr_proba = np.array(dataAUC.prob.unique()) # array of unique probabilities

Final_A =  []  #Final values of metric A
thresh = []  # All unique threshold values



for i in arr_proba:
```

```
    data_new_ite = dataAUC.copy()   #used copy to point to different memory location ,

    data_new_ite['prob'] = np.where(data_new_ite['prob'] >= i, 1.0, data_new_ite['prob'
    data_new_ite['prob'] = np.where(data_new_ite['prob'] < i, 0.0, data_new_ite['prob']

    data_FP = data_new_ite.query('y == 0.0 & prob == 1.0')
    data_FN = data_new_ite.query('y == 1.0 & prob == 0.0')


    FP = data_FP.shape[0]
    FN = data_FN.shape[0]

    A = ((500*FN) + (100*FP)) #calculating metric
    Final_A.append(A)
    thresh.append(i)

best_thresh = thresh[Final_A.index(min(Final_A))]  #Finding best thresh by getting loc

print("Best threshold which gives minimum value of A {}".format(round(best_thresh,2)))
```

```
Best threshold which gives 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 [199…

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

###MSE
n = data.shape[0] #total data points
summ = 0

for i in data.index:
    summ += ( (data['y'][i] - data['pred'][i])  *  (data['y'][i] - data['pred'][i]) )

MSE = summ/n  #calculating MSE

print("Mean Square Error : {}".format(round(MSE,3)))

###MAPE
summ_err_mape = 0
a_bar = data['pred'].sum()  #taking sum of all y

for i in data.index:
    summ_err_mape += abs( (data['y'][i] - data['pred'][i])  )   #sum of absolute value
```

```
MAPE = (summ_err_mape/a_bar)*100   #calculating  MAPE

print("Mean Absolute percentage error : {}".format(round(MAPE,3)))

###R^2 Error

y_bar = data['y'].mean()          #calculating mean of all points y

ss_sum = 0
for i in data.index:
    ss_sum += ( (data['y'][i] - y_bar)  *  (data['y'][i] - y_bar) )   # sum of squares

r2Error = 1- (summ/ss_sum)   # using summ calculated for MSE , calculating R2Error

print("R Squared Error : {}".format(round(r2Error,3)))
```

```
Mean Square Error : 177.166
Mean Absolute percentage error : 12.927
R Squared Error : 0.956
```