# Compute performance metrics for the given Y and Y_score without sklearn

In [21]:
```
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)
   Note- Make sure that you arrange your probability scores in descending order while calculating AUC

4. Compute Accuracy Score

In [22]:

```
df_a=pd.read_csv('5_a.csv')
df_a.head
```

Out[22]:

```
<bound method NDFrame.head of           y     proba
0      1.0  0.637387
1      1.0  0.635165
2      1.0  0.766586
3      1.0  0.724564
4      1.0  0.889199
...    ...       ...
10095  1.0  0.665371
10096  1.0  0.607961
10097  1.0  0.777724
10098  1.0  0.846036
10099  1.0  0.679507

[10100 rows x 2 columns]>
```

In [23]:

```
def predict(df,y,thresh_hold):
    y_prediction=[]
    for value in df[y]:
        if value<thresh_hold:
            y_prediction.append(0)
        else:
            y_prediction.append(1)
    return y_prediction


# confusion matrix
def calculate_vals(df):
    tp=0
    tn=0
    fn=0
    fp=0
    for val1,val2 in enumerate(df['y']):
        if(df.y_prediction[val1]==1) and df.y[val1]==1:
            tp=tp+1
        if(df.y_prediction[val1]==0) and df.y[val1]==0:
            tn=tn+1
        if(df.y_prediction[val1]==1) and df.y[val1]==0:
            fp=fp+1
        if(df.y_prediction[val1]==0) and df.y[val1]==1:
            fn=fn+1
    return {'tn':tn,'tp':tp,'fn':fn,'fp':fp}
```

In [24]:

```
thresh_hold=0.5
df_a['y_prediction']=predict(df_a,'proba',thresh_hold)
```

In [25]:

```
df_a.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10100 entries, 0 to 10099
Data columns (total 3 columns):
 #   Column        Non-Null Count  Dtype  
---  ------        --------------  -----  
 0   y             10100 non-null  float64
 1   proba         10100 non-null  float64
 2   y_prediction  10100 non-null  int64  
dtypes: float64(2), int64(1)
memory usage: 236.8 KB
```

In [26]:

```
confusion_matrix=calculate_vals(df_a)
```

In [27]:

```
print("The Confusion Matrix is ",confusion_matrix)
```

```
The Confusion Matrix is  {'tn': 0, 'tp': 10000, 'fn': 0, 'fp': 100}
```

In [28]:

```
#calculating F1 score
x=df_a.y.value_counts()
P=x[1]

precision=confusion_matrix['tp']/(confusion_matrix['tp']+confusion_matrix['fp'])
recall=confusion_matrix['tp']/P

F1=2*precision*recall/(precision+recall)
print('the F1 score is: ',F1)
```

```
the F1 score is:  0.9950248756218906
```

In [29]:

```
print(x)
```

```
1.0    10000
0.0      100
Name: y, dtype: int64
```

In [30]:

```
# Calculating Accuracy
Acc=(confusion_matrix['tp']+confusion_matrix['tn'])/df_a.shape[0]
print('the accuracy is: ',Acc)
```

```
the accuracy is:  0.9900990099009901
```

In [31]:

```
# AUC score funtion
from tqdm import tqdm_notebook      # purpose of import is to just see progress
def auc(df):
    s = df['y'].value_counts()
    P = s[1]
    N = s[0]
    tpr = []
    fpr = []
    for elem in tqdm_notebook(df['proba']):
        df['y_prediction']=predict(df,'proba',elem)
        confusion_matrix=calculate_vals(df)
        tpr.append(confusion_matrix['tp']/P)
        fpr.append(confusion_matrix['fp']/N)
        df.drop(columns=['y_prediction'])
    return np.trapz(tpr,fpr)
```

In [32]:

```
data=df_a.sort_values(by='proba',ascending=False)
df_a.drop(columns=['y_prediction'])
```

Out[32]:

```
          y     proba
    0   1.0  0.637387
    1   1.0  0.635165
    2   1.0  0.766586
    3   1.0  0.724564
    4   1.0  0.889199
  ...   ...       ...
10095   1.0  0.665371
10096   1.0  0.607961
10097   1.0  0.777724
10098   1.0  0.846036
10099   1.0  0.679507

10100 rows × 2 columns
```

```
In [33]: data
```

Out[33]:

| | y | proba | y_prediction |
|---|---|---|---|
| **1664** | 1.0 | 0.899965 | 1 |
| **2099** | 1.0 | 0.899828 | 1 |
| **1028** | 1.0 | 0.899825 | 1 |
| **9592** | 1.0 | 0.899812 | 1 |
| **8324** | 1.0 | 0.899768 | 1 |
| **...** | ... | ... | ... |
| **8294** | 1.0 | 0.500081 | 1 |
| **1630** | 1.0 | 0.500058 | 1 |
| **7421** | 1.0 | 0.500058 | 1 |
| **805** | 1.0 | 0.500047 | 1 |
| **5012** | 1.0 | 0.500019 | 1 |

10100 rows × 3 columns

```
In [34]: AUC_score=auc(data)
         print ('the AUC Score is :',AUC_score)
```

```
C:\Users\honey\AppData\Local\Temp/ipykernel_17736/4015618333.py:9: TqdmDeprecat
ionWarning: This function will be removed in tqdm==5.0.0
Please use `tqdm.notebook.tqdm` instead of `tqdm.tqdm_notebook`
  for elem in tqdm_notebook(df['proba']):

100%                                        10100/10100 [1:13:17<00:00, 2.23it/s]

the AUC Score is : 0.48829900000000004
```

## 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)
   Note- Make sure that you arrange your probability scores in descending order while calculating AUC

4. Compute Accuracy Score

In [35]:
```
df_b=pd.read_csv('5_b.csv')
df_b.head()
```

Out[35]:

|   | y | proba |
|---|---|---|
| **0** | 0.0 | 0.281035 |
| **1** | 0.0 | 0.465152 |
| **2** | 0.0 | 0.352793 |
| **3** | 0.0 | 0.157818 |
| **4** | 0.0 | 0.276648 |

In [36]:
```
thresh_hold=0.5
df_b['y_prediction']=predict(df_b,'proba',thresh_hold)
confusion_matrix_B=calculate_vals(df_b)
```

In [37]:
```
#confusion matrix values
print('the confusion matrix is :', confusion_matrix_B)
```

```
the confusion matrix is : {'tn': 9761, 'tp': 55, 'fn': 45, 'fp': 239}
```

In [38]:
```
# F1 score
x=df_b.y.value_counts()
P=x[1]

precision_B=confusion_matrix_B['tp']/(confusion_matrix_B['tp']+confusion_matrix_B
recall_B=confusion_matrix_B['tp']/P

F1_B=2*precision_B*recall_B/(precision_B+recall_B)
print('the F1 Score is : ',F1_B)
```

```
the F1 Score is :  0.2791878172588833
```

In [39]:
```
# Accuracy
Acc_B=(confusion_matrix_B['tp']+confusion_matrix_B['tn'])/df_b.shape[0]
print('the Accuracy is :',Acc_B)
```

```
the Accuracy is : 0.9718811881188119
```

In [40]:
```
#AUC score
data_B=df_b.sort_values(by='proba',ascending=False)
data_B.drop(columns=['y_prediction'])
AUC_score_B=auc(data_B)
print('the AUC Score is: ',AUC_score_B)
```

```
C:\Users\honey\AppData\Local\Temp/ipykernel_17736/4015618333.py:9: TqdmDeprecat
ionWarning: This function will be removed in tqdm==5.0.0
Please use `tqdm.notebook.tqdm` instead of `tqdm.tqdm_notebook`
  for elem in tqdm_notebook(df['proba']):
```

100% 10100/10100 [1:18:51<00:00, 2.36it/s]

```
the AUC Score is:  0.9377570000000001
```

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

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 [50]:

```
# min metric function

def min_metric(data):
    s = data['y'].value_counts()
    P = s[1]
    N = s[0]
    tpr = []
    fpr = []
    metric={}
    for elem in tqdm_notebook(data['prob']):
        data['y_prediction']=predict(data,'prob',elem)
        confusion_matrix=calculate_vals(data)
        metric_val=(500*confusion_matrix['fn'])+(100*confusion_matrix['fp'])
        metric[elem]=metric_val
        data.drop(columns=['y_prediction'])
    return(metric)
```

In [51]:

```
df_c=pd.read_csv('5_c.csv')
df_c.head()
```

Out[51]:

|   | y | prob |
|---|---|---|
| **0** | 0 | 0.458521 |
| **1** | 0 | 0.505037 |
| **2** | 0 | 0.418652 |
| **3** | 0 | 0.412057 |
| **4** | 0 | 0.375579 |

In [52]:

```
data=pd.read_csv('5_c.csv')
print(data.head())
print(data.shape)
data=data.sort_values(by='prob',ascending=False)
result=min_metric(data)
```

```
   y      prob
0  0  0.458521
1  0  0.505037
2  0  0.418652
3  0  0.412057
4  0  0.375579
(2852, 2)

C:\Users\honey\AppData\Local\Temp/ipykernel_17736/3767384905.py:10: TqdmDeprecationWarning: This function will be removed in tqdm==5.0.0
Please use `tqdm.notebook.tqdm` instead of `tqdm.tqdm_notebook`
  for elem in tqdm_notebook(data['prob']):
```

100% 2852/2852 [06:35<00:00, 8.13it/s]

In [53]:

```
temp = min(result.values())
res = [key for key in result if result[key] == temp]
print('The KEY and VALUE pair for minimum value for the specified metric-',res,te
```

```
The KEY and VALUE pair for minimum value for the specified metric- [0.230039027
8970873] 141000
```

# 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 [54]:

```
df_d=pd.read_csv('5_d.csv')
df_d.head()
```

Out[54]:

|   | y | pred |
|---|---|---|
| **0** | 101.0 | 100.0 |
| **1** | 120.0 | 100.0 |
| **2** | 131.0 | 113.0 |
| **3** | 164.0 | 125.0 |
| **4** | 154.0 | 152.0 |

In [67]:

```
def ss_res(df,col):
    val=0
    for index,value in enumerate(df[col]):
        val=val+(value*value)
    return val

def error(df,col1,col2):
    val=[]
    for index, (value1, value2) in enumerate(zip(df[col1], df[col2])):
        val.append(value1-value2)
    return val
def ss_tot(df,col):
    val=0
    mean_val=df_d['y'].mean()
    for index,value in enumerate(df[col]):
        val=val+ (value-mean_val)*(value-mean_val)
    return val

def mean_sq_error(df,col):
    return ss_res(df,col)/len(df[col])

def mape(df,col1,col2):
    val=sum(df[col1])/sum(df[col2])
    return val

def absolute_error(df,col):
    val=[]
    for index,value in enumerate(df[col]):
        val.append(abs(value))
    return val
```

In [68]:

```
df_d['error']=error(df_d,'y','pred')
df_d['abs_error']=absolute_error(df_d,'error')
```

In [69]:

```
MSE=mean_sq_error(df_d,'error')
print("the Mean squared error is : ", MSE)
```

```
the Mean squared error is :  177.16569974554707
```

In [70]:

```
MAPE=mape(df_d,'abs_error','y')
print('the MAPE value is :', MAPE)
```

```
the MAPE value is : 0.1291202994009687
```

In [71]:

```
SS_RES=ss_res(df_d,'error')
SS_TOT=ss_tot(df_d,'y')
R_square= 1- (SS_RES/SS_TOT)
print('The Co-efficient of determination value is: ',R_square)
```

```
The Co-efficient of determination value is:  0.9563582786990964
```

In [ ]: