Compute performance metrics for the given Y and Y_score without sklearn

In [26]:
```
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 [27]:
```
def conf_matrix(y,y_hat):
    TP = 0
    TN = 0
    FP = 0
    FN = 0
    for y_i,y_hat_i in zip(y,y_hat):
        if y_i == y_hat_i:
            if y_hat_i == 1:
                TP+=1
            else:
                TN+=1
        else:
            if y_i == 1 and y_hat_i == 0:
                FN+=1
            else:
                FP+=1
    return [TN,FN,FP,TP]
```

In [28]:
```
df_a=pd.read_csv('5_a.csv')
df_a.head()
print(df_a.head())
df_a['y'].value_counts()
```

```
     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
```

Out[28]:
```
1.0    10000
0.0      100
Name: y, dtype: int64
```

In [29]:
```
y_hat = np.select([df_a['proba'] < 0.5,df_a['proba'] >= 0.5],[0,1])
confusion_matrix = conf_matrix(list(df_a['y']),list(y_hat))
TN = confusion_matrix[0]
FN = confusion_matrix[1]
FP = confusion_matrix[2]
TP = confusion_matrix[3]
print(confusion_matrix)
```

```
[0, 0, 100, 10000]
```

- Precision = TP/(FP + TP). It means that out of all the points our model predicted to be positive, what percentage of them are actually positive

In [30]:
```
precision = TP/(FP+TP)
print(precision)
```

```
0.9900990099009901
```

- Recall = TPR = TP/(FN + TP). It means that out of all positive points that were present in the dataset, how many of them were predicted successfully.

In [31]:
```
recall = TP/(TP+FN)
print(recall)
```

```
1.0
```

- F1-Score is Harmonic mean of precision and recall.
- F1-Score = 2*(pricision*recall) / (pricision+recall)

In [32]:
```
f1_score = 2*(precision*recall)/(precision+recall)
print(f1_score)
```

```
0.9950248756218906
```

- Accuracy can be defined as the ratio of the correctly classified and the total number of points

In [33]:
```
accuracy = TP/(sum(confusion_matrix))
print(accuracy)
```

```
0.9900990099009901
```

In [34]:
```
threshold = sorted(list(df_a['proba']),reverse=True)
TPR = []
FPR = []

for thr in threshold:
    y_hat = np.select([df_a['proba'] < thr,df_a['proba'] >= thr],[0,1])
    confusion_matrix = conf_matrix(list(df_a['y']),list(y_hat))
    TN = confusion_matrix[0]
    FN = confusion_matrix[1]
    FP = confusion_matrix[2]
    TP = confusion_matrix[3]
    tpr = TP/(TP+FN)
    fpr = FP/(TN+FP)
    TPR.append(tpr)
    FPR.append(fpr)

np.trapz(TPR,FPR)
```

Out[34]:
```
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')
print(df_b.head())
df_b['y'].value_counts()
```

```
     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
```

Out[35]:
```
0.0    10000
1.0      100
Name: y, dtype: int64
```

In [36]:
```
y_hat = np.select([df_b['proba'] < 0.5,df_b['proba'] >= 0.5],[0,1])
print(y_hat)
confusion_matrix = conf_matrix(list(df_b['y']),list(y_hat))
TN = confusion_matrix[0]
FN = confusion_matrix[1]
FP = confusion_matrix[2]
TP = confusion_matrix[3]
print(confusion_matrix)
```

```
[0 0 0 ... 0 0 0]
[9761, 45, 239, 55]
```

In [37]:
```
precision = TP/(FP+TP)
print(precision)
```

```
0.1870748299319728
```

In [38]:
```
recall = TP/(TP+FN)
print(recall)
```

```
0.55
```

In [39]:
```
f1_score = 2*(precision*recall)/(precision+recall)
print(f1_score)
```

```
0.2791878172588833
```

In [40]:
```
accuracy = TP/(sum(confusion_matrix))
print(accuracy)
```

```
0.005445544554455445
```

In [41]:
```
threshold = sorted(list(df_b['proba']),reverse=True)
TPR = []
FPR = []

for thr in threshold:
    y_hat = np.select([df_b['proba'] < thr,df_b['proba'] >= thr],[0,1])
    confusion_matrix = conf_matrix(list(df_b['y']),list(y_hat))
    TN = confusion_matrix[0]
    FN = confusion_matrix[1]
    FP = confusion_matrix[2]
    TP = confusion_matrix[3]
    tpr = TP/(TP+FN)
    fpr = FP/(TN+FP)
    TPR.append(tpr)
    FPR.append(fpr)

np.trapz(TPR,FPR)
```

Out[41]: 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 [42]:
```
df_c=pd.read_csv('5_c.csv')
print(df_c.head())
df_c['y'].value_counts()
```

```
   y      prob
0  0  0.458521
1  0  0.505037
2  0  0.418652
3  0  0.412057
4  0  0.375579
```

Out[42]:
```
0    1805
1    1047
Name: y, dtype: int64
```

In [43]:
```
threshold = sorted(list(df_c['prob']),reverse=True)
min_A = float("inf")
for thr in threshold:
    y_hat = np.select([df_c['prob'] < thr,df_c['prob'] >= thr],[0,1])
    confusion_matrix = conf_matrix(list(df_c['y']),list(y_hat))
    TN = confusion_matrix[0]
    FN = confusion_matrix[1]
    FP = confusion_matrix[2]
    TP = confusion_matrix[3]
    A = 500*FN+100*FP
    #print(thr,A)
    if A < min_A:
        best_threshold = thr
        min_A = A

print(best_threshold,min_A)
```

```
0.2300390278970873 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 [44]:
```
df_d=pd.read_csv('5_d.csv')
df_d.head()
```

Out[44]:

	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 [45]:
```
y = np.array(df_d['y'])
y_hat = np.array(df_d['pred'])
diff_sqr = np.square(y-y_hat)
MSE = np.sum(diff_sqr)/len(y)
print(MSE)
```

```
177.16569974554707
```

In [46]:
```
M_MAPE = np.sum(np.absolute(y-y_hat))/np.sum(y)
print(M_MAPE)
```

```
0.1291202994009687
```

In [47]:
```
y_mean = np.mean(y)
diff_sqr = np.square(y-y_mean)
ss_total = np.sum(diff_sqr)


diff_sqr = np.square(y-y_hat)
ss_res = np.sum(diff_sqr)

R_sqr = 1 - (ss_res/ss_total)
print(R_sqr)
```

```
0.9563582786990937
```