Compute performance metrics for the given Y and Y_score without sklearn

In [4]:
```
import numpy as np
import pandas as pd
# other than these two you should not import any other packages
```

In [25]:
```
data = pd.read_csv('5_a.csv')
data=data.sort_values(by= ["proba"], ascending = False)
data.head(10)
```

Out[25]:

	y	proba
1664	1.0	0.899965
2099	1.0	0.899828
1028	1.0	0.899825
9592	1.0	0.899812
8324	1.0	0.899768
2396	1.0	0.899751
3789	1.0	0.899467
2822	1.0	0.899444
2370	1.0	0.899429
7636	1.0	0.899415

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 [26]:
```
true_y = data['y']
pred_y = [0 if j['proba'] < 0.5 else 1 for i,j in data.iterrows()]
```

In [27]:
```
def confusion_matrix(true_y,pred_y):
  classes = sorted(list(set(true_y)))
  size = len(list(set(true_y)))
  c = np.zeros([size, size],dtype=int)
  for i in range(size):
    for j in range(size):
      for k in range(len(true_y)):
        if true_y[k] == classes[i] and pred_y[k]  == classes[j]:
          c[i][j] += 1

  return c

c = confusion_matrix(true_y,pred_y)
print(f'the confusion matrix is :')
print(c)
```

```
the confusion matrix is :
[[    0   100]
 [    0 10000]]
```

In [28]:
```
precision = c[1][1]/(c[1][0]+c[1][1])
recall = c[1][1]/(c[1][1]+c[0][1])
f1_score = 2*((precision*recall)/(precision+recall))
print(f'f1 score is {f1_score}')
```

```
f1 score is 0.9950248756218906
```

In [31]:
```
%%time
def auc_score(data):
  #TTP - Total True Positive

  TPR = []
  FPR = []
  for threshold in data['proba'].unique():
    TP, FP, TN, FN, tpr, fpr = 0,0,0,0,0,0
    data['y_pred'] =np.where( data['proba'] >=  threshold, 1,0)
    TP = ((data['y']==1.0) & (data['y_pred'] == 1.0)).sum()
    FP = ((data['y']==0.0) & (data['y_pred'] == 1.0)).sum()
    TN = ((data['y']==0.0) & (data['y_pred'] == 0.0)).sum()
    FN = ((data['y']==1.0) & (data['y_pred'] == 0.0)).sum()
    tpr = TP/(TP+FN)
    fpr = FP/(FP+TN)
    TPR.append(tpr)
    FPR.append(fpr)
  AUC_Score = np.trapz(TPR,FPR)
  return AUC_Score
```

```
CPU times: user 6 µs, sys: 0 ns, total: 6 µs
Wall time: 9.06 µs
```

In [33]:
```
print(f'the AUC SCORE is {auc_score(data)}')
```

```
the AUC SCORE is 0.48829900000000004
```

In [34]:
```
Accuracy_Score = ((c[0][0]+c[1][1])/data.shape[0])
print(f'the Accuracy Score is {Accuracy_Score}')
```

```
the Accuracy Score is 0.9900990099009901
```

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

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

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 [64]:
```
data1 = pd.read_csv('5_b.csv')
data1.head(10)
```

Out[64]:

	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
5	0.0	0.190260
6	0.0	0.320328
7	0.0	0.435013
8	0.0	0.284849
9	0.0	0.427919

In [65]:
```
true_y = data1['y']
pred_y = [0 if j['proba'] < 0.5 else 1 for i,j in data1.iterrows()]
```

In [66]:
```
def confusion_matrix(true_y,pred_y):
  classes = sorted(list(set(true_y)))
  size = len(list(set(true_y)))
  c = np.zeros([size, size],dtype=int)
  for i in range(size):
    for j in range(size):
      for k in range(len(true_y)):
        if true_y[k] == classes[i] and pred_y[k]  == classes[j]:
          c[i][j] += 1

  return c

c = confusion_matrix(true_y,pred_y)
print(f'the confusion matrix is :')
print(c)
```

```
the confusion matrix is :
[[9761  239]
 [  45   55]]
```

In [67]:
```
precision = c[1][1]/(c[1][0]+c[1][1])
recall = c[1][1]/(c[1][1]+c[0][1])

f1_score = 2*((precision*recall)/(precision+recall))
print(f'f1 score is {f1_score}')
```

```
f1 score is 0.2791878172588833
```

In [69]:
```
data1 = pd.read_csv('5_b.csv')
data1=data1.sort_values(by= ["proba"], ascending = False)
```

In [70]:
```
print(f'the AUC SCORE is {auc_score(data1)}')
```

```
the AUC SCORE is 0.9377570000000001
```

In [71]:
```
Accuracy_Score = ((c[0][0]+c[1][1])/data1.shape[0])
print(f'Accuracy Score is {Accuracy_Score}')
```

```
Accuracy Score is 0.9718811881188119
```

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 [83]:
```
data2 = pd.read_csv('5_c.csv')
data2.head(10)
```

Out[83]:

	y	prob
0	0	0.458521
1	0	0.505037

	y	prob
2	0	0.418652
3	0	0.412057
4	0	0.375579
5	0	0.595387
6	0	0.370288
7	0	0.299273
8	0	0.297000
9	0	0.266479

In [84]:
```
true_y = data2['y']
pred_y = [0 if j['prob'] < 0.5 else 1 for i,j in data2.iterrows()]
```

In [85]:
```
thresholds = [i for i in sorted(list(set(data2['prob'])))]
thresholds = sorted(list(set(thresholds)))
print(f'Total length of thresholds {len(thresholds)}')
```

```
Total length of thresholds 2791
```

In [86]:
```
%%time
A = np.zeros(len(thresholds), dtype = float)
for i, k in enumerate(thresholds):
  predicted_y = [0 if j['prob'] < k else 1 for i,j in data2.iterrows()]
  z = confusion_matrix(true_y,predicted_y)
  A[i] = 500*z[1][0] + 100*z[0][1]
```

```
CPU times: user 9min 26s, sys: 90.9 ms, total: 9min 27s
Wall time: 9min 27s
```

In [87]:
```
A_list = list(A)
b = min(A_list)
print(f'A is {b}')
```

```
A is 141000.0
```

In [88]:
```
best_threshold = thresholds[A_list.index(b)]
print(f'The best threshold is {best_threshold}')
```

```
The best threshold is 0.2300390278970873
```

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 [ ]:
```
data3 = pd.read_csv('5_d.csv')
data3.head(10)
```

Out[ ]:

	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
5	133.0	153.0
6	148.0	139.0
7	172.0	145.0
8	153.0	162.0
9	162.0	154.0

In [ ]:
```
sum_Square_Error = 0
for i,j in data3.iterrows():
  sum_Square_Error = sum_Square_Error + pow((j['y']-j['pred']),2)

Mean_Square_Error = (sum_Square_Error/data3.shape[0])

print(f'Mean Square Error is {Mean_Square_Error}')
```

```
Mean Square Error is 177.16569974554707
```

In [ ]:
```
error = 0
for i in range(len(data3)):
  error = error + np.absolute((data3['y'][i]-data3['pred'][i]))
sum= 0
for i in range(len(data3)):
  sum = sum + data3['y'][i]
mape = error/sum
print(f'Mean Square percentage Error is {mape}')
```

```
Mean Square percentage Error is 0.1291202994009687
```

In [ ]:
```
def coefficient_of_determination(data3):
  sum_true_y = 0
  for i,j in data3.iterrows():
    sum_true_y = sum_true_y + j['y']
  mean =  sum_true_y/data3.shape[0]

  total_sum_of_square = 0
  for i,j in data3.iterrows():
    total_sum_of_square = total_sum_of_square + pow((j['y']-mean),2)
```

```
  sum_of_square_residue = 0
  for i,j in data3.iterrows():
    sum_of_square_residue = sum_of_square_residue + pow((j['y']-j['pred']),2)

  coefficient_of_determination  = (1-(sum_of_square_residue/total_sum_of_square))

  return coefficient_of_determination

R2_error = coefficient_of_determination(data3)
print(f'R2_error is {R2_error}')
```

```
R2_error is 0.9563582786990964
```