Compute performance metrics for the given Y and Y_score without sklearn

```
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 negat

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 f

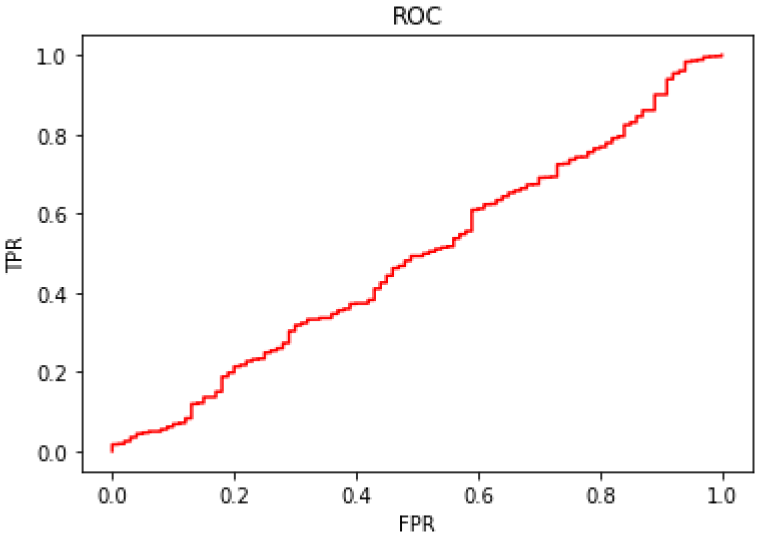
4. Compute Accuracy Score

```
df=pd.read_csv('5_a.csv') #Reading a Dataframe

df['op']=np.round(df["proba"])
df.head()
```

y proba op

```
import matplotlib.pyplot as plt
from tqdm import tqdm

def confusion_matrix(df): #Function to print Confusion Matrix
    TP=0;FP=0;TN=0;FN=0
    for index, row in df.iterrows(): #Iteration through each row of Dataframe
      if row['op']==row['y']:
        if row['op']==1.0:
          TP+=1
        else:
          TN+=1
      else:
        if row['op']==1.0:
          FP+=1
        else:
          FN+=1

    print("TP=",TP,"   FP=",FP);print()
    print("FN=",FN,"   TN=",TN)

    return TP,FP,TN,FN

def F1_score(TP,FP,TN,FN): #FUnction to find F1 SCore
  recall=TP/(TP+FN)
  precision=FP/(FP+TN)

  print("\nPrecision=",precision,"Recall=",recall)
  F1 = 2*precision*recall/(precision + recall)
  print("F1-Score=",F1)

def accuracy(df): #Function to print accuracy
  p=0
  for index, row in df.iterrows(): #Iterating and checking y==ypred?
    if row['op']==row['y']:
      p+=1
  print("\nAccuracy=",(p/len(df))*100)

def get_tpr_fpr(df,sorted_thresholds):

  tpr=np.array([]);fpr=np.array([])
  for threshold in sorted_thresholds: #Itereating through numpy array of sorted th
    TP=0;FP=0;TN=0;FN=0
    for i in range(len(df)): #Iterating through dataframe converted to numpy array
      if df[i][1]>=threshold:
        if df[i][0]==1.0:
          TP+=1
        else:FP+=1
      else:
        if df[i][0]==1.0:
          FN+=1
        else:TN+=1
    tpr=np.append(tpr,TP/(TP+FN))
    fpr=np.append(fpr,FP/(FP+TN))
  return tpr,fpr

def AUC(df,sorted_thresholds): #Function to draw graph and give AUC
  tpr,fpr=get_tpr_fpr(df,sorted_thresholds) #this function gives tpr and fpr as ar
  plt.title("ROC")
  plt.xlabel("FPR")
  plt.ylabel("TPR")
  plt.plot(fpr, tpr, color ="red")
  plt.show()

  print("AUC=",np.trapz(np.flipud(tpr) , np.flipud(fpr))) #CAlculation AUC score


TP,FP,TN,FN=confusion_matrix(df)
F1_score(TP,FP,TN,FN)
accuracy(df)
AUC(df.to_numpy(),np.sort(df.proba.unique()))
```

```
TP= 10000    FP= 100

FN= 0    TN= 0

Precision= 1.0 Recall= 1.0
F1-Score= 1.0

Accuracy= 99.00990099009901
```



```
AUC= 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 negat

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 f

4. Compute Accuracy Score

```
# write your code
df2=pd.read_csv('5_b.csv')

df2['op']=np.round(df2["proba"])
df2.head()
```

	y	proba	op
0	0.0	0.281035	0.0
1	0.0	0.465152	0.0
2	0.0	0.352793	0.0
3	0.0	0.157818	0.0
4	0.0	0.276648	0.0

```
TP2,FP2,TN2,FN2=confusion_matrix(df2)
F1_score(TP2,FP2,TN2,FN2)
accuracy(df2)
AUC(df2.to_numpy(),np.sort(df2.proba.unique()))
```

```
TP= 55     FP= 239

FN= 45     TN= 9761

Precision= 0.0239 Recall= 0.55
F1-Score= 0.04580937445548005
```

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

```
# write your code
df3=pd.read_csv('5_c.csv')

df3['op']=np.round(df3["prob"])
df3.head()
```

	y	prob	op
0	0	0.458521	0.0
1	0	0.505037	1.0
2	0	0.418652	0.0
3	0	0.412057	0.0
4	0	0.375579	0.0

```
def get_best_threshold(df,sorted_thresholds):
  best_threshold=0;A=np.array([1000000000]) #appended some large value to avoid mu
  tpr=np.array([]);fpr=np.array([])
  for threshold in sorted_thresholds: #iterating through all thresholds
    FP=0;FN=0
    for i in range(len(df)):  #Iterating to dataframe convertedd to numpy array
      if df[i][1]>=threshold and df[i][0]==0.0: #Calculating only FP and FN
        FP+=1
      elif df[i][1]<threshold and df[i][0]==1.0:
        FN+=1
    A_value=(500*FN)+(100*FP) # A for current iteration
    if A_value < A.min(): # CHecking if A value is minimum or not
      best_threshold=threshold
    A=np.append(A,A_value)

  return best_threshold
```

```
uniques=np.sort(df3.prob.unique())
best_threshold=get_best_threshold(df3.to_numpy(),uniques)
print("Best Threshold Value",best_threshold)
```

```
Best Threshold Value 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 valu

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_d

```
df4=pd.read_csv('5_d.csv')
```

```
df4.head()
```

	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

```
def MSE(df):
  Error=0
  for i in range(len(df)): #Iterating through each row of dataframe
    Error+=np.square(df[i][0]-df[i][1])
  return Error/(len(df))

def avoid_zero(x):
  return x if x else 1

def MAPE(df):
  Error=0
  Avg=np.mean(df,axis=0)[0]
  for i in range(len(df)):
    Error+=abs((df[i][0] - df[i][1])) #Function to avoid zero because around 5000 v
  return Error/(len(df)*Avg)

def R_squared(df):
  SS_total=0;SS_residual=0
  mean=np.mean(df,axis=0)[1]
  for i in range(len(df)):
    SS_residual+=np.square(df[i][0]-df[i][1])
    SS_total+=np.square(df[i][0]-mean)

  return 1-SS_residual/SS_total


npdf=df4.to_numpy()
print("Mean Squared Error =",MSE(npdf))
print("Mean Absolute Percentage Error =",MAPE(npdf))
print("R-Squared =",R_squared(npdf))
```

```
Mean Squared Error = 177.16569974554707
Mean Absolute Percentage Error = 0.1291202994009687
R-Squared = 0.9563583447288628
```

✓ 1s completed at 6:42 PM ● ×