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 negatives poi
   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 eacl

4.  Compute Accuracy Score
```

```
# write your code here

from google.colab import drive
drive.mount('/gdrive')
```

```
Go to this URL in a browser: https://accounts.google.com/o/oauth2/auth?client_id=947

Enter your authorization code:
··········
Mounted at /gdrive
```

```
task_one = pd.read_csv('/content/5_a.csv')
#ypred=[0 if y_score < 0.5 else 1]
```

```
task_one['y'] = task_one['y'].astype(int)
task_one['P'] = (task_one['proba'] >= 0.5).astype(int)
task_one.head()
```

	y	proba	P
0	1	0.637387	1
1	1	0.635165	1
2	1	0.766586	1
3	1	0.724564	1
4	1	0.889199	1

```
def compute_All(Actual,Predicted):

  TP = np.sum((Actual==1) & (Predicted==1))
  TN = np.sum((Actual==0) & (Predicted==0))
  FN = np.sum((Actual==1) & (Predicted==0))
  FP = np.sum((Actual==0) & (Predicted==1))

  Accuracy = ((TP+TN)/float(TP+TN+FP+FN))*100

  Precision = (TP/(FP+TP))*100
  Recall = (TP/(TP+FN))*100

  F1_Score = 2 * ((Precision*Recall)/(Precision+Recall))

  return TP,TN,FN,FP,Accuracy,F1_Score

TP,TN,FN,FP,Accuracy,F1_Score = compute_All(task_one['y'], task_one['P'])

print('Confusion Matrix:')
print('True Positive',TP)
print('True Negative',TN)
print('False Positive',FP)
print('False Negative',FN)
print('***************************')
print('Accuarcy of task A:',Accuracy)
print('F1_Score of Task A:',F1_Score)
```

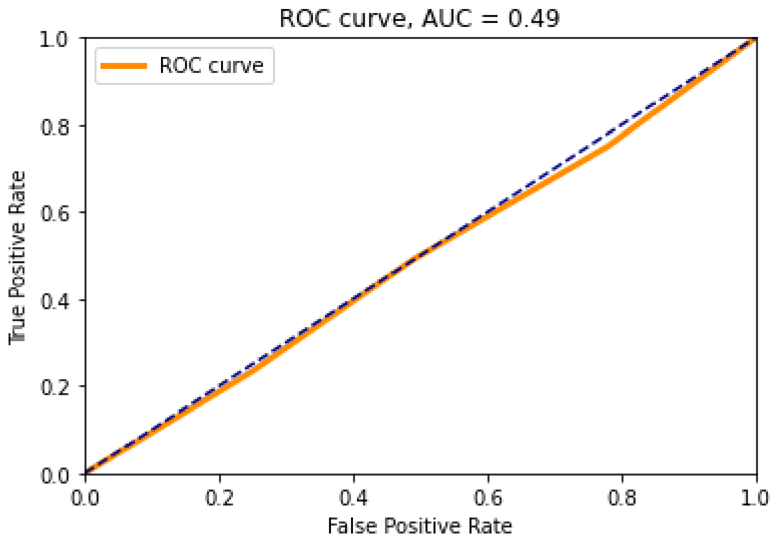
```
Confusion Matrix:
True Positive 10000
True Negative 0
False Positive 100
False Negative 0
***************************
Accuarcy of task A: 99.00990099009901
F1_Score of Task A: 99.50248756218905
```

```
def roc_curve(actual,probability,thresholds):
    FPR = []
    TPR = []
```

```
    for threshold in thresholds:
        threshold = round(threshold,2)
        predicted = np.where(probability >= threshold, 1, 0) #for each threshold value we

        #Computing Confusion Matrix
        tp = np.sum((predicted == 1) & (actual == 1))
        tn = np.sum((predicted == 0) & (actual == 0))
        fp = np.sum((predicted == 1) & (actual == 0))
        fn = np.sum((predicted == 0) & (actual == 1))

        #Computing TPR & FPR based on the formulae
        FPR.append(fp / (fp + tn))
        TPR.append(tp / (tp + fn))

    return [FPR,TPR]

FPR, TPR= roc_curve(task_one['y'],task_one['proba'],thresholds = np.sort(np.arange(0.0,1.0
FPR_array = np.asarray(FPR)
TPR_array = np.asarray(TPR)

AUC_A = np.trapz(TPR_array,FPR_array)
print('Area Under the Curve:', AUC_A)
```

```
Area Under the Curve: 0.48897750000000006
```

```
#Plotting the below ROC Curve just for reference
import matplotlib.pyplot as plt
plt.plot(FPR, TPR,color='darkorange',lw= 3,label='ROC curve')
plt.plot([0, 1], [0, 1], color='navy', linestyle='--')
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.0])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('ROC curve, AUC = %.2f'%AUC_A)
plt.legend()
plt.show()
```



```
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 poi
   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 eacl

4. Compute Accuracy Score
```

```
# write your code
task_two = pd.read_csv('/content/5_b.csv')

#ypred=[0 if y_score < 0.5 else 1]
task_two['y'] = task_two['y'].astype(int)
task_two['P'] = (task_two['proba'] >= 0.5).astype(int)
task_two.head()
```

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

```
def compute_All(Actual,Predicted):

  TP = np.sum((Actual==1) & (Predicted==1))
  TN = np.sum((Actual==0) & (Predicted==0))
  FN = np.sum((Actual==1) & (Predicted==0))
  FP = np.sum((Actual==0) & (Predicted==1))

  Accuracy = ((TP+TN)/float(TP+TN+FP+FN))*100
```

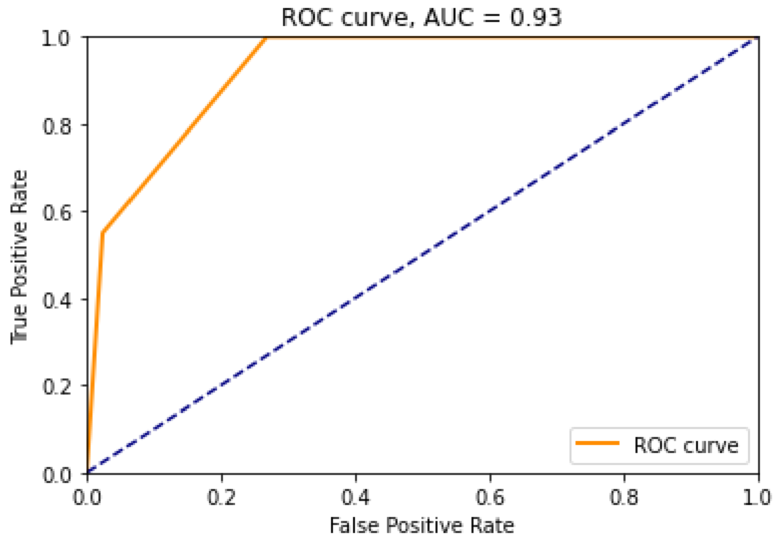
```
  Precision = (TP/(FP+TP))*100
  Recall = (TP/(TP+FN))*100

  F1_Score = 2 * ((Precision*Recall)/(Precision+Recall))

  return TP,TN,FN,FP,Accuracy,F1_Score

TP,TN,FN,FP,Accuracy,F1_Score = compute_All(task_two['y'], task_two['P'])

print('Confusion Matrix:')
print('True Positive',TP)
print('True Negative',TN)
print('False Positive',FP)
print('False Negative',FN)
print('***************************')
print('Accuarcy of task B:',Accuracy)
print('F1_Score of Task B:',F1_Score)
```

```
Confusion Matrix:
True Positive 55
True Negative 9761
False Positive 239
False Negative 45
***************************
Accuarcy of task B: 97.18811881188118
F1_Score of Task B: 27.918781725888326
```

```
def roc_curve(actual,probability,thresholds):
    FPR = []
    TPR = []

    for threshold in thresholds:
        threshold = round(threshold,2)
        predicted = np.where(probability >= threshold, 1, 0) #for each threshold value we

        #Computing Confusion Matrix
        tp = np.sum((predicted == 1) & (actual == 1))
        tn = np.sum((predicted == 0) & (actual == 0))
        fp = np.sum((predicted == 1) & (actual == 0))
        fn = np.sum((predicted == 0) & (actual == 1))

        #Computing TPR & FPR based on the formulae
        FPR.append(fp / (fp + tn))
        TPR.append(tp / (tp + fn))

    return [FPR,TPR]

FPR_2, TPR_2= roc_curve(task_two['y'],task_two['proba'],thresholds=np.sort(np.arange(0.0,1
FPR_arr = np.asarray(FPR_2)
TPR_arr = np.asarray(TPR_2)
AUC = np.trapz(TPR_arr, FPR_arr)
print('Area Under the Curve:', AUC)
```

```
Area Under the Curve: 0.9276825
```

```
#Plotting the below ROC Curve just for reference
import matplotlib.pyplot as plt
plt.plot(FPR_2, TPR_2, color='darkorange', lw = 2, label='ROC curve')
plt.plot([0, 1], [0, 1], color='navy', linestyle='--')
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.0])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('ROC curve, AUC = %.2f'%AUC)
plt.legend()
plt.show()
```



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

```
task_three = pd.read_csv('/content/5_c.csv')
task_three['P'] = (task_three['prob'] >= 0.5).astype(int)
task_three.head()
```

	y	prob	P
0	0	0.458521	0
1	0	0.505037	1

```
def roc_curve(actual,probability,thresholds):
    A = []
    for threshold in thresholds:
        threshold = round(threshold,2)
        predicted = np.where(probability >= threshold, 1, 0) #for each threshold value we

        #Computing FP and FN and calculating value a based on formula
        fp = np.sum((predicted == 1) & (actual == 0))
        fn = np.sum((predicted == 0) & (actual == 1))
        a = 500 * fn + 100 * fp
        A.append([a, threshold])
        Low_thres_Value_A = min(A)
    return A,Low_thres_Value_A

A,Low_thres_Value_A = roc_curve(task_three['y'],task_three['prob'],thresholds=np.sort(np.a
print(A)
print('Lowest Value of A = {0} and best theshold = {1}'.format(Low_thres_Value_A[(0)],Low_
```

```
[[523500, 1.0], [523500, 0.99], [523500, 0.98], [523500, 0.97], [523500, 0.96], [5225
Lowest Value of A = 141000 and best theshold = 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 featu

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_determi

```
task_four = pd.read_csv('/content/5_d.csv')
task_four.head()
```

	y	pred
0	101.0	100.0
1	120.0	100.0
2	131.0	113.0

```
def compute_All_task_4(actual,predicted):
  MSE = np.square(np.subtract(actual,predicted)).mean()

  error = abs(np.subtract(actual,predicted))
  MAPE = ((error/np.mean(actual)).mean())*100

  SST = np.sum(np.square((np.subtract(actual,(np.mean(actual))))))
  SSR = np.sum(np.square(np.subtract(actual,predicted)))
  COD = 1-(SSR/SST)

  return MSE,MAPE,COD

MSE,MAPE,COD = compute_All_task_4(task_four['y'], task_four['pred'])

print('Mean Square Error:', MSE)
print('************************')
print('Mean Absolute Percentage Error:',MAPE)
print('************************')
print('COD or R^2:',COD)
```