Compute performance metrics for the given Y and Y_score without sklearn

In [1]:

```
import numpy as np
import pandas as pd
from tqdm import tqdm
# 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)

4. Compute Accuracy Score

In [2]:

```
# write your code here
data = pd.read_csv('5_a.csv')
# print(data.shape)
data['output'] = np.where(data['proba']<0.5, 0, 1)
# print(data.head())
# data.head(10000)
TP = len(data[(data['output']==1) & (data['y']==1)])
TN = len(data[(data['output']==0) & (data['y']==0)])
FP = len(data[(data['output']==1) & (data['y']==0)])
FN = len(data[(data['output']==0) & (data['y']==1)])
# print(TP)
print('TP: {}; TN:{}; FP:{}; FN:{}'.format(TP,TN,FP,FN))

precision = TP/(TP + FP)
recall = TP/(TP + FN)
print("Precision : {} Recall:{}".format(precision,recall))

f1_score = 2*precision*recall/(precision + recall)
print("F1 Score: ",f1_score)

accuracy = (TP+TN)/(TP+TN+FP+FN)
print("Accuracy: ",accuracy)
```

```
TP: 10000; TN:0; FP:100; FN:0
Precision : 0.9900990099009901 Recall:1.0
F1 Score:  0.9950248756218906
Accuracy:  0.9900990099009901
```

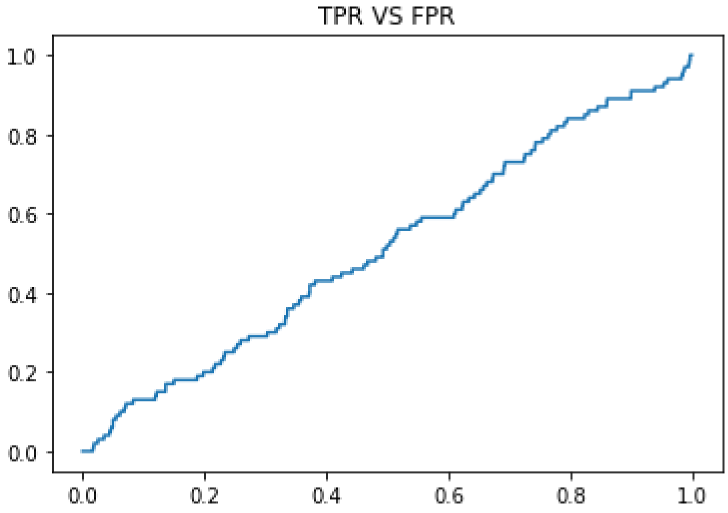
In [11]:

```
# Computing AUC Score
n = 10
Size = data.shape[0]
Threshold_values = set(data['proba'].tolist())
Threshold_values = sorted(Threshold_values)

tpr = []
fpr = []

for thres in tqdm(Threshold_values):
    TP = len(data[(np.where(data['proba']<thres, 0, 1)==1) & (data['y']==1)])
    TN = len(data[(np.where(data['proba']<thres, 0, 1)==0) & (data['y']==0)])
    FP = len(data[(np.where(data['proba']<thres, 0, 1)==1) & (data['y']==0)])
    FN = len(data[(np.where(data['proba']<thres, 0, 1)==0) & (data['y']==1)])

    tpr.append(TP/(TP + FN))
    fpr.append(FP/(FP + TN))
```

```
100%|██████████| 10100/10100 [01:17<00:00, 130.50it/s]
```

In [12]:

```
import matplotlib.pyplot as plt

# Plotting TPR and FPR
plt.plot(tpr,fpr)
plt.title('TPR VS FPR')
plt.show()
```



In [15]:

```
from sklearn import metrics
print(metrics.auc(fpr, tpr))
```

```
0.48829900000000004
```

In [16]:

```
tpr.sort()
fpr.sort()
print('AUC Score: ',np.trapz(tpr,fpr))
```

```
AUC Score:  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)

4. Compute Accuracy Score

In [17]:

```
# write your code
data = pd.read_csv('5_b.csv')
data['output'] = np.where(data['proba']<0.5, 0, 1)

TP = len(data[(data['output']==1) & (data['y']==1)])
TN = len(data[(data['output']==0) & (data['y']==0)])
FP = len(data[(data['output']==1) & (data['y']==0)])
FN = len(data[(data['output']==0) & (data['y']==1)])

print('TP: {}; TN:{}; FP:{}; FN:{}'.format(TP,TN,FP,FN))

precision = TP/(TP + FP)
recall = TP/(TP + FN)
print("Precision : {} Recall:{}".format(precision,recall))

f1_score = 2*precision*recall/(precision + recall)
print("F1 Score: ",f1_score)

accuracy = (TP+TN)/(TP+TN+FP+FN)
print("Accuracy: ",accuracy)
```

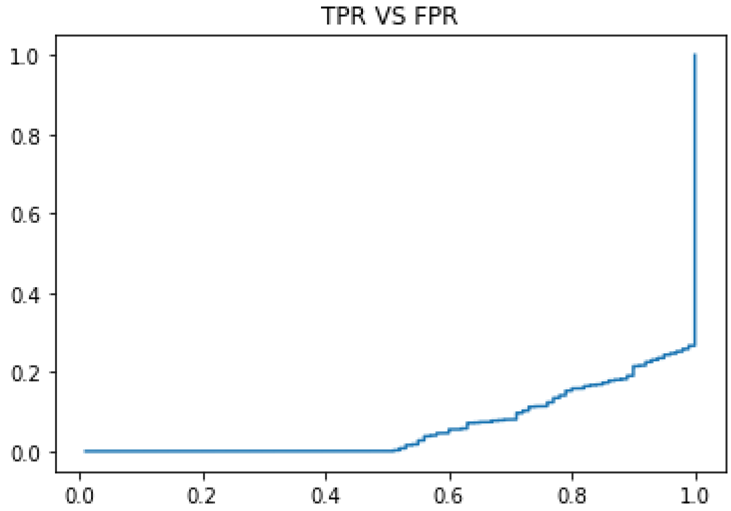
```
TP: 55; TN:9761; FP:239; FN:45
Precision : 0.1870748299319728 Recall:0.55
F1 Score:  0.2791878172588833
Accuracy:  0.9718811881188119
```

In [18]:

```
n = 10
Size = data.shape[0]
Threshold_values = set(data['proba'].tolist())
Threshold_values = sorted(Threshold_values)


tpr = []
fpr = []

for thres in tqdm(Threshold_values):
    TP = len(data[(np.where(data['proba']<thres, 0, 1)==1) & (data['y']==1)])
    TN = len(data[(np.where(data['proba']<thres, 0, 1)==0) & (data['y']==0)])
    FP = len(data[(np.where(data['proba']<thres, 0, 1)==1) & (data['y']==0)])
    FN = len(data[(np.where(data['proba']<thres, 0, 1)==0) & (data['y']==1)])

    tpr.append(TP/(TP + FN))
    fpr.append(FP/(FP + TN))
```

```
100%|██████████| 10100/10100 [01:22<00:00, 121.81it/s]
```

```
In [19]: import matplotlib.pyplot as plt

         # Plotting TPR and FPR
         plt.plot(tpr,fpr)
         plt.title('TPR VS FPR')
         plt.show()
```



```
In [20]: from sklearn import metrics
         print(metrics.auc(fpr, tpr))

         0.937757
```

```
In [21]: tpr.sort()
         fpr.sort()
         print(np.trapz(tpr,fpr))

         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 **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 points
> **Note 2:** use pandas or numpy to read the data from **5_c.csv**

```
In [22]: # write your code
         data = pd.read_csv('5_c.csv')
         data.shape

Out[22]: (2852, 2)
```

In [23]:

```
n = 10

Threshold_values = set(data['prob'].tolist())
Threshold_values = sorted(Threshold_values)

FP = []
FN = []

for thres in tqdm(Threshold_values):
    FP.append(len(data[(np.where(data['prob']<thres, 0, 1)==1) & (data['y']==0
)]))
    FN.append(len(data[(np.where(data['prob']<thres, 0, 1)==0) & (data['y']==1
)]))
```

```
100%|██████████| 2791/2791 [00:10<00:00, 260.80it/s]
```

In [24]:

```
for thres in tqdm(Threshold_values):
    new_roc = 500*FN[Threshold_values.index(thres)] + 100*FP[Threshold_values.
index(thres)]

    if Threshold_values.index(thres) == 0:
        roc = new_roc
    elif new_roc < roc:
        roc = new_roc
        best_thres = thres
print('ROC: {} for Thres:{}'.format(roc,best_thres))
```

```
100%|██████████| 2791/2791 [00:00<00:00, 7778.69it/s]

ROC: 141000 for Thres: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 [26]:

```
from math import pow
from tqdm import tqdm
```

In [27]:

```
data = pd.read_csv('5_d.csv')
N = len(data)
mse = 0
mape = 0
SS_res = 0
SS_tot = 0
y_mean = np.mean(data['y'])
# print(y_mean)
# len(data)
data.head()
# print(data.shape)
for i in tqdm(range(0,N)):
#     print(data.iloc[i,0])
#     print(data.iloc[i,1])
#     print(i)
    SS_res += pow((data.iloc[i,0]-data.iloc[i,1]), 2)
    SS_tot += pow((data.iloc[i,0]-y_mean), 2)
    mape += abs(data.iloc[i,0]-data.iloc[i,1])*100/(N*np.mean(data['y']))

RR_2 = 1- SS_res/SS_tot
mse = SS_res/N
print('SS_res: {} SS_tot:{}'.format(SS_res,SS_tot))
print('MSE: {} :  MAPE:{} : RR_2:{}'.format(mse,mape,RR_2))
```

```
100%|██████████| 157200/157200 [01:09<00:00, 2268.55it/s]

SS_res: 27850448.0 SS_tot:638161080.035662
MSE: 177.16569974554707 :  MAPE:12.912029940108486 : RR_2:0.9563582786990964
```