Compute performance metrics for the given Y and Y_score without sklearn

In [1]:

```
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)

4. Compute Accuracy Score

In [2]:

```
df = pd.read_csv("5_a.csv")
df.head()
```

Out[2]:

	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

In [3]:

```
df['y_pred']= df.proba.map(lambda x: 0 if x<0.5 else 1)
df['y'] = df.y.astype('int')
df.rename(columns = {'y':'y_act'},inplace=True)
df.head()
```

Out[3]:

	y_act	proba	y_pred
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

In [4]:

```
df.shape
```

Out[4]:

```
(10100, 3)
```

In [5]:

```
def compute_confusion_matrix(y_act,y_pred):
    """returns TP, FP, FN, TN"""
    TP = sum((y_act == 1) & (y_pred == 1))
    FP = sum((y_act == 0) & (y_pred == 1))
    FN = sum((y_act == 1) & (y_pred == 0))
    TN = sum((y_act == 0) & (y_pred == 0))
    return TP, FP, FN, TN

def compute_precision(TP, FP):
    """ Precision = TP/(TP+FP) """
    if TP!= 0:
        return TP/(TP+FP)
    else :
        return 0

def compute_recall(TP, FN):
    """ Recall = TP/(TP+FN) """
    if TP!= 0:
        return TP/(TP+FN)
    else :
        return 0

def compute_FPR(FP, TN):
    """FPR = FP /(FP + TN)"""
    if  FP!= 0:
        return FP /(FP + TN)
    else :
        return 0

def compute_F1_score(y_act,y_pred):
    """F1_score = (2*precision*recall)/(precision + recall)"""
    TP, FP, FN, TN = compute_confusion_matrix(y_act,y_pred)
    precision = compute_precision(TP, FP)
    recall = compute_recall(TP,FN)
    F1_score = (2*precision*recall)/(precision + recall)
    return F1_score

def compute_accuracy(TP, FP, FN, TN):
    """Accuracy = TP + TN / (TP + FP + FN + TN)"""
    return (TP + TN)/(TP + FP + FN + TN)

def compute_AUC_score(TPR, FPR):
    """Computes AUC scores for different Thresholds"""
    return np.trapz(TPR, FPR)
```

In [6]:

```
TP, FP, FN, TN = compute_confusion_matrix(df.y_act,df.y_pred)
print("TP, FP, FN, TN = ",TP, FP, FN, TN)
print("Precision =" , compute_precision(TP,FP))
print("Recall =", compute_recall(TP,FN))
print("F1 Score =", compute_F1_score(df.y_act,df.y_pred))
print("Accuracy = ", compute_accuracy(TP, FP, FN, TN))
```

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

In [7]:

```
threshold =  sorted(np.unique(df.proba),reverse=True)
```

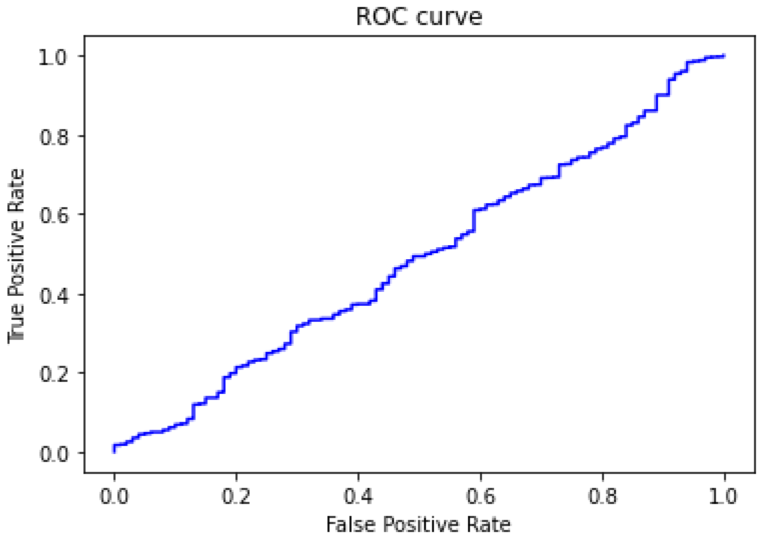
In [8]:

```
from tqdm import tqdm

TPR =[]
FPR =[]
for i in tqdm(threshold):
    df['threshold_' + str(i)] = df.proba.map(lambda x: 0 if x<i else 1)
    TP, FP, FN, TN = compute_confusion_matrix(df.y_act,df['threshold_' + str(i
)])
    tpr_ = compute_recall(TP, FN)
    fpr_ = compute_FPR(FP, TN)
    TPR.append(tpr_)
    FPR.append(fpr_)

print("AUC_score =",compute_AUC_score(np.array(TPR), np.array(FPR)))
```

```
100%|██████████| 10100/10100 [18:58<00:00,  8.87it/s]

AUC_score = 0.48829900000000004
```

In [9]:

```
import matplotlib.pyplot as plt
plt.plot(FPR, TPR, color='b')
plt.title('ROC curve')
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
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 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/q/53603376/4084039), [https://stackoverflow.com/a/39678975/4084039 (https://stackoverflow.com/a/39678975/4084039)](https://stackoverflow.com/a/39678975/4084039)

4. Compute Accuracy Score

In [10]:

```
df_b = pd.read_csv("5_b.csv")
df_b.head()
```

Out[10]:

	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

In [11]:

```
df_b['y_pred']= df_b.proba.map(lambda x: 0 if x<0.5 else 1)
df_b['y'] = df_b.y.astype('int')
df_b.rename(columns = {'y':'y_act'},inplace=True)
df_b.head()
```

Out[11]:

	y_act	proba	y_pred
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

In [12]:

```
df_b.shape
```

Out[12]:

```
(10100, 3)
```

In [13]:

```
TP, FP, FN, TN = compute_confusion_matrix(df_b.y_act,df_b.y_pred)
print("TP, FP, FN, TN = ",TP, FP, FN, TN)
print("Precision =" , compute_precision(TP,FP))
print("Recall =", compute_recall(TP,FN))
print("F1 Score =", compute_F1_score(df_b.y_act,df_b.y_pred))
print("Accuracy = ", compute_accuracy(TP, FP, FN, TN))
```

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

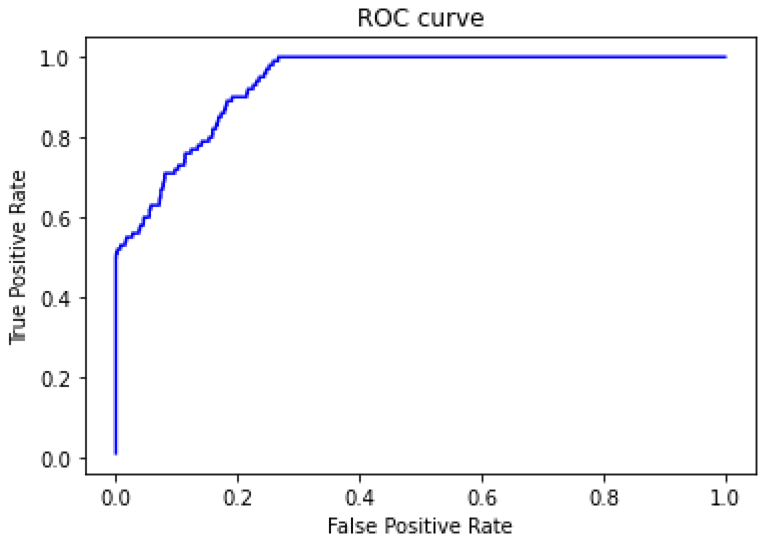
In [14]:

```
from tqdm import tqdm

threshold =  sorted(np.unique(df_b.proba),reverse=True)
TPR =[]
FPR =[]
for i in tqdm(threshold):
    df_b['threshold_' + str(i)] = df_b.proba.map(lambda x: 0 if x<i else 1)
    TP, FP, FN, TN = compute_confusion_matrix(df_b.y_act,df_b['threshold_' + str
(i)])
    tpr_ = compute_recall(TP, FN)
    fpr_ = compute_FPR(FP, TN)
    TPR.append(tpr_)
    FPR.append(fpr_)

print("AUC_score =",compute_AUC_score(np.array(TPR), np.array(FPR)))
```

```
100%|██████████| 10100/10100 [20:15<00:00,  8.31it/s]


AUC_score = 0.9377570000000001
```

In [15]:

```
import matplotlib.pyplot as plt
plt.plot(FPR, TPR, color='b')
plt.title('ROC curve')
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
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 points
> **Note 2:** use pandas or numpy to read the data from **5_c.csv**

In [16]:

```
df_c = pd.read_csv('5_c.csv')
df_c.head()
```

Out[16]:

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

In [17]:

```
from tqdm import tqdm

threshold =  sorted(np.unique(df_c.prob),reverse=True)
A=[]
for i in tqdm(threshold):
    df_c['threshold_' + str(i)] = df_c.prob.map(lambda x: 0 if x<i else 1)
    TP, FP, FN, TN = compute_confusion_matrix(df_c.y,df_c['threshold_' + str(i
)])
    a_ = (500 * FN) + (100 * FP)
    A.append(a_)

best_threshold = threshold[A.index(min(A))]
print("optimal threshold is : ", best_threshold)
```

```
100%|██████████| 2791/2791 [02:18<00:00, 20.15it/s]

optimal 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 [18]:

```python
df_d = pd.read_csv('5_d.csv')
df_d.head()
```

Out[18]:

	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 [19]:

```python
#https://www.geeksforgeeks.org/python-mean-squared-error/
def MSE(y_act, y_pred):
    return np.square(np.subtract(y_act,y_pred)).mean()

def MAPE(y_act, y_pred):
    y_act, y_pred = np.array(y_act), np.array(y_pred)
    return np.mean(np.abs(y_act - y_pred))/ np.mean(y_act) *100

def R2(y_act, y_pred):
    ss_res = np.square(np.subtract(y_act, y_pred)).sum()
    ss_tot = np.square(np.subtract(y_act, np.mean(y_act))).sum()
    return 1-(ss_res/ss_tot)

print("MSE : ", MSE(df_d.y, df_d.pred))
print("MAPE : ", MAPE(df_d.y, df_d.pred))
print("R2 : ", R2(df_d.y, df_d.pred))
```

```
MSE :  177.16569974554707
MAPE :  12.91202994009687
R2 :  0.9563582786990937
```