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

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

1. Compute Confusion Matrix

2. Compute F1 Score

3. Compute AUC Score, you need to compute different thresholds and for each threshold compute the 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 [2]:

```
# write your code here
data = pd.read_csv('5_a.csv')
print(data.shape)
data.head(3)
```

```
(10100, 2)
```

Out[2]:

	y	proba
0	1.0	0.637387
1	1.0	0.635165
2	1.0	0.766586

In [3]:

```
data['y_pred'] = (data.proba>=0.5).map({True:1,False:0})
print(data['y_pred'].value_counts())
data.head(3)
```

```
1    10100
Name: y_pred, dtype: int64
```

Out[3]:

Out[3]:

	y	proba	y_pred
0	1.0	0.637387	1
1	1.0	0.635165	1
2	1.0	0.766586	1

In [4]:

```
# 1. Confusion Matrix calculation
tn = ((data['y']==0)&(data['y_pred']==0)).sum()
fn = ((data['y']==1)&(data['y_pred']==0)).sum()
fp = ((data['y']==0)&(data['y_pred']==1)).sum()
tp = ((data['y']==1)&(data['y_pred']==1)).sum()

confusion_matrix = np.array([[tn,fn],[fp,tp]])
print(confusion_matrix)
```

```
[[    0     0]
 [  100 10000]]
```

In [5]:

```
# 2. F1-Score calcuation
precision = tp/(tp+fp)
recall = tp/(tp+fn)
f1_score = 2*(precision*recall/(precision+recall))
print(f1_score)
```

```
0.9950248756218906
```

In [6]:

```
# 4. Accuracy score calculation
accuracy_score = (tn+tp)/(tn+fn+fp+tp)
print(accuracy_score)
```

```
0.9900990099009901
```

In [7]:

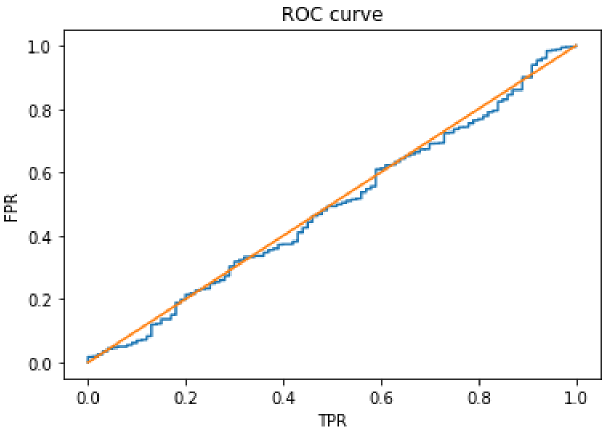
```
def calc_auc(y_true, y_score):
    desc_score_indices = np.argsort(y_score)[::-1]
    y_score = y_score[desc_score_indices]
    y_true = y_true[desc_score_indices]

    distinct_indices = np.where(np.diff(y_score))[0]
    end = np.array([y_true.size - 1])
    threshold_indices = np.hstack((distinct_indices, end))

    thresholds = y_score[threshold_indices]
    tps = np.cumsum(y_true)[threshold_indices]

    fps = (1 + threshold_indices) - tps

    return(tps,fps,thresholds)
```

In [9]:

```
import matplotlib.pyplot as plt
tps, fps, thresholds = calc_auc(np.array(data['y']),np.array(data['proba']))
tpr = np.hstack((0,tps/tps[-1]))
fpr = np.hstack((0,fps/fps[-1]))
```

```
auc_score = np.trapz(tpr,fpr)
print(auc_score)
plt.plot(fpr, tpr)
plt.plot((0.0,1.0),(0.0,1.0))
plt.xlabel('TPR')
plt.ylabel('FPR')
plt.title('ROC curve')
plt.show()
```

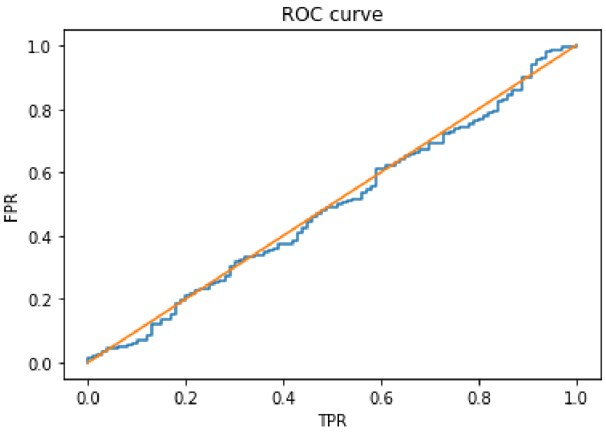
```
0.48829900000000004
```



In [10]:

```
#Comparison with sklearn
from sklearn import metrics
fpr, tpr, thresholds = metrics.roc_curve(data['y'], data['proba'])
auc_score = metrics.roc_auc_score(data['y'], data['proba'])
print(auc_score)

plt.plot(fpr, tpr)
plt.plot((0.0,1.0),(0.0,1.0))
plt.xlabel('TPR')
plt.ylabel('FPR')
plt.title('ROC curve')
plt.show()
```

```
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

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

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 [11]:

```
# write your code here
data = pd.read_csv('5_b.csv')
print(data.shape)
data.head(3)
```

```
(10100, 2)
```

Out[11]:

	y	proba
0	0.0	0.281035
1	0.0	0.465152
2	0.0	0.352793

In [12]:

```
data['y_pred'] = (data.proba>=0.5).map({True:1,False:0})
print(data['y_pred'].value_counts())
data.head(3)
```

```
0    9806
1     294
Name: y_pred, dtype: int64
```

Out[12]:

	y	proba	y_pred
0	0.0	0.281035	0
1	0.0	0.465152	0
2	0.0	0.352793	0

In [13]:

```
# 1. Confusion Matrix calculation
tn = ((data['y']==0)&(data['y_pred']==0)).sum()
fn = ((data['y']==1)&(data['y_pred']==0)).sum()
fp = ((data['y']==0)&(data['y_pred']==1)).sum()
tp = ((data['y']==1)&(data['y_pred']==1)).sum()

confusion_matrix = np.array([[tn,fn],[fp,tp]])
print(confusion_matrix)
```

```
[[9761   45]
 [ 239   55]]
```

In [14]:

```
# 2. F1-Score calcuation
precision = tp/(tp+fp)
recall = tp/(tp+fn)
f1_score = 2*(precision*recall/(precision+recall))
print(f1_score)
```

```
0.2791878172588833
```

In [15]:

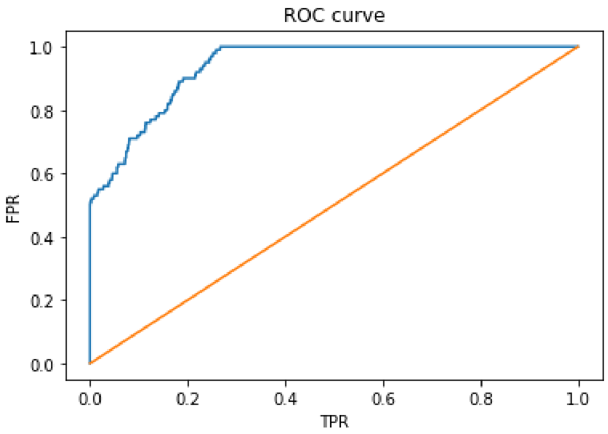
```
# 4. Accuracy score calculation
accuracy_score = (tn+tp)/(tn+fn+fp+tp)
print(accuracy_score)
```

```
0.9718811881188119
```

In [16]:

```
tps,fps,threshold = calc_auc(np.array(data['y']),np.array(data['proba']))
fpr = np.hstack((0,fps/fps[-1]))
tpr = np.hstack((0,tps/tps[-1]))
auc_score = np.trapz(tpr,fpr)
print(auc_score)
plt.plot(fpr, tpr)
plt.plot((0.0,1.0),(0.0,1.0))
plt.xlabel('TPR')
plt.ylabel('FPR')
plt.title('ROC curve')
plt.show()
```

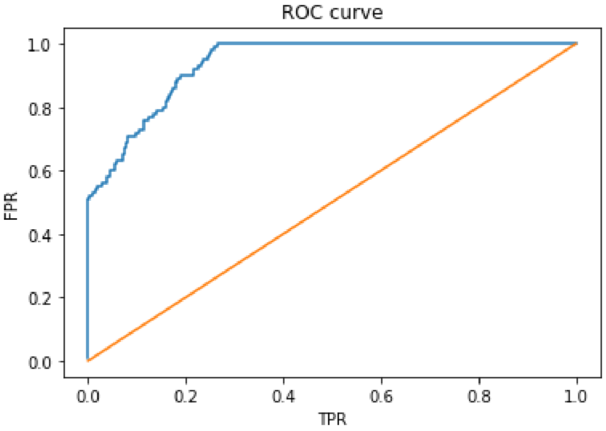
```
0.9377570000000001
```



In [17]:

```
#Comparison with sklearn
fpr, tpr, thresholds = metrics.roc_curve(data['y'], data['proba'])
auc_score = metrics.roc_auc_score(data['y'], data['proba'])
print(auc_score)

plt.plot(fpr, tpr)
plt.plot((0.0,1.0),(0.0,1.0))
plt.xlabel('TPR')
plt.ylabel('FPR')
plt.title('ROC curve')
plt.show()
```

```
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: $y^{pred}= \text{[0 if y_score < threshold else 1]}$

$ 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 [18]:

```
data = pd.read_csv('5_c.csv')
print(data.shape)
data.head(3)
```

```
(2852, 2)
```

Out[18]:

	y	prob
0	0	0.458521
1	0	0.505037
2	0	0.418652

In [19]:

```
data['y_pred'] = (data.prob>=0.5).map({True:1,False:0})
print(data['y_pred'].value_counts())
data.head(3)
```

```
0    2099
1     753
Name: y_pred, dtype: int64
```

Out[19]:

	y	prob	y_pred

	y	prob	y_pred
0	0	0.458521	0
1	0	0.505037	1
2	0	0.418652	0

In [20]:

```
tn = ((data['y']==0)&(data['y_pred']==0)).sum()
fn = ((data['y']==1)&(data['y_pred']==0)).sum()
fp = ((data['y']==0)&(data['y_pred']==1)).sum()
tp = ((data['y']==1)&(data['y_pred']==1)).sum()

confusion_matrix = np.array([[tn,fn],[fp,tp]])
A = 500*fn+100*fp
print(confusion_matrix)
print('A = ',A)
```

```
[[1637  462]
 [ 168  585]]
A =  247800
```

In [21]:

```
def calc_A(y_true, y_score):
    min_A = 1000000
    desc_score_indices = np.argsort(y_score)[::-1]
    y_score = y_score[desc_score_indices]
    y_true = y_true[desc_score_indices]

    distinct_indices = np.where(np.diff(y_score))[0]
    end = np.array([y_true.size - 1])
    threshold_indices = np.hstack((distinct_indices, end))

    thresholds = y_score[threshold_indices]
    for threshold in thresholds:
        tn = ((y_true==0) & (y_score<threshold)).sum()
        fn = ((y_true==1) & (y_score<threshold)).sum()
        fp = ((y_true==0) & (y_score>=threshold)).sum()
        tp = ((y_true==1) & (y_score>=threshold)).sum()

        A = 500*fn + 100*fp
        if A<min_A:
            min_A = A
            min_threshold = threshold

    return(min_threshold)
```

In [22]:

```
threshold = calc_A(np.array(data['y']),np.array(data['prob']))
print('Required Threshold = ',threshold)
```

```
Required Threshold =  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#Definition

s

In [23]:

```
data = pd.read_csv('5_d.csv')
print(data.shape)
data.head(3)
```

```
(157200, 2)
```

Out[23]:

	y	pred
0	101.0	100.0
1	120.0	100.0
2	131.0	113.0

In [24]:

```
#Mean Square Error Calculation
mse = ((data['y']-data['pred'])**2).sum()
print('Mean Square Error: ',mse/data.shape[0])
```

```
Mean Square Error:  177.16569974554707
```

In [25]:

```
#Mean absolute percentage error calculation
mape = abs(data['y']-data['pred']).sum()/data['y'].sum()
print('Mean Absolute Percentage Error: ',mape)
```

```
Mean Absolute Percentage Error:  0.1291202994009687
```

In [26]:

```
#R2-Score calculation
SS_tot = ((data['y']-data['y'].mean())**2).sum()
SS_res = ((data['y']-data['pred'])**2).sum()
R2_Score = 1 -(SS_res/SS_tot)
print('R2-Score = ',R2_Score)
```

```
R2-Score =  0.9563582786990937
```