1 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
```

In [2]:
```
# Reference---> https://github.com/vennela28/AppliedAI/blob/ad6961fce6d0d9b3d28a9ce55bcee5ab37fc5380/Assignments/
```

1.0.1 To compute Confusion Matrix

In [3]:
```
def confusion_matrix(data):
    #returns TN, FP, FN, TP
    count_tn = len(data[(data['y'] == 0) & (data['y_pred'] == 0)])
    count_fp = len(data[(data['y'] == 0) & (data['y_pred'] == 1)])
    count_fn = len(data[(data['y'] == 1) & (data['y_pred'] == 0)])
    count_tp = len(data[(data['y'] == 1) & (data['y_pred'] == 1)])
    return count_tn, count_fp, count_fn, count_tp
```

1.0.2 To compute F1 Score

In [4]:
```
def f1_score(data):
    tn,fp,fn,tp = confusion_matrix(data)
    precision = tp / (tp+fp)
    recall = tp / (tp+fn)
    f1 = 2*((precision*recall) / (precision+recall))
    return f1
```

1.0.3 To compute Accuracy

In [5]:
```
def compute_accuracy(data):
    tn,fp,fn,tp = confusion_matrix(data)
    accuracy = ((tp+tn) / (tp+tn+fp+fn))
    return accuracy
```

1.0.4 To compute AUC Score

In [6]:
```
def auc_score(data):
    #Computing AUC Scores for different thresholds
    tpr_array = []
    fpr_array = []
    sort = data.sort_values('proba', ascending=False)
    for i in range(0, len(sort)):
        sort['y_pred'] = np.where(sort['proba'] >= sort.iloc[i]['proba'],1,0)
        tn,fp,fn,tp = confusion_matrix(sort)
        tpr = tp / (tp+fn)
        fpr = fp / (tn+fp)
        tpr_array.append(tpr)
        fpr_array.append(fpr)
    c = np.trapz(tpr_array,fpr_array)
    return c
```

1.1 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)
   Note- Make sure that you arrange your probability scores in descending order while calculating AUC

4. Compute Accuracy Score

In [7]:
```
df_a=pd.read_csv('5_a.csv')
df_a.head()
```

Out[7]:

	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 [8]:
```
df_a['y_pred'] = df_a['proba'].apply(lambda x: 1 if x>=0.5 else 0)
```

In [9]:
```
df_a.head(10)
```

Out[9]:

	y	proba	y_pred
0	1.0	0.637387	1
1	1.0	0.635165	1
2	1.0	0.766586	1
3	1.0	0.724564	1
4	1.0	0.889199	1
5	1.0	0.601600	1
6	1.0	0.666323	1
7	1.0	0.567012	1
8	1.0	0.650230	1
9	1.0	0.829346	1

1.1.1 Confusion Matrix

In [10]:
```
tn,fp,fn,tp = confusion_matrix(df_a)
print("True Negative  :",tn)
print("False Positive :",fp)
print("False Negative :",fn)
print("True Positive  :",tp)
```

```
True Negative  : 0
False Positive : 100
False Negative : 0
True Positive  : 10000
```

1.1.2 F1 Score

In [11]:
```
f1 = f1_score(df_a)
print('F1 Score :',f1)
```

```
F1 Score : 0.9950248756218906
```

1.1.3 AUC Score

In [12]:
```
auc = auc_score(df_a)
print('AUC Score:',auc)
```

```
AUC Score: 0.48829900000000004
```

1.1.4 Accuracy Score

In [13]:
```
accuracy = compute_accuracy(df_a)
print('Accuracy Score :',accuracy)
```

```
Accuracy Score : 0.9900990099009901
```

1.2 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)
   Note- Make sure that you arrange your probability scores in descending order while calculating AUC

4. Compute Accuracy Score

In [14]:
```
df_b = pd.read_csv('5_b.csv')
df_b.head()
```

Out[14]:

	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

1.2.1 Converting probability value to output class label 0 Or 1

In [15]:
```
df_b['y_pred'] = df_b['proba'].apply(lambda x: 1 if x>=0.5 else 0)
```

1.2.2 Confusion Matrix

In [16]:
```
tn,fp,fn,tp = confusion_matrix(df_b)
print("True Negative  :",tn)
print("False Positive :",fp)
print("False Negative :",fn)
print("True Positive  :",tp)
```

```
True Negative  : 9761
False Positive : 239
False Negative : 45
True Positive  : 55
```

1.2.3 F1-Score

In [17]:
```
f1 = f1_score(df_b)
print("F1 Score :",f1)
```

```
F1 Score : 0.2791878172588833
```

1.2.4 AUC Score

In [18]:
```
auc = auc_score(df_b)
print('AUC Score :',auc)
```

```
AUC Score : 0.9377570000000001
```

1.2.5 Accuracy Score

In [19]:
```
accuracy = compute_accuracy(df_b)
print('Accuracy Score :',accuracy)
```

```
Accuracy Score : 0.9718811881188119
```

1.2.6 C. Compute the best threshold (similarly to ROC curve computation) of probability which gives lowest values of metric A for the given data

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 [20]:
```
df_c=pd.read_csv('5_c.csv')
df_c.head()
```

Out[20]:

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

In [21]:
```
def best_threshold(data):
    check = 0
    thresh = []
    a = []
    _sorted = data.sort_values('prob',ascending=False)
    for i in range(0,len(_sorted)):
        if check==(_sorted.iloc[i]['prob']):
            continue
        check = _sorted.iloc[i]['prob']
        thresh.append(check)
        _sorted['y_pred'] = np.where(_sorted['prob']>=_sorted.iloc[i]['prob'],1,0)
        tn,fp,fn,tp = confusion_matrix(_sorted)
        value = (500*fn) + (100*fp)
        a.append(value)
    index = a.index(min(a))
    return thresh[index]
```

In [22]:
```
best = best_threshold(df_c)
print('Best Threshold value:',best)
```

```
Best Threshold value: 0.2300390278970873
```

1.3 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 [23]:
```
df_d=pd.read_csv('5_d.csv')
df_d.head()
```

Out[23]:

	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 [24]:
```
#Code to calculate MSE, MAPE and R-Squared Error
def regress_metrics(data):
    n = len(data)
    data['ei'] = data.apply(lambda x: abs(x['y'] - x['pred']),axis=1)
    data['mse'] = data['ei'].apply(lambda x: x*x)
    total = data['mse'].sum()
    mse = total/n
    mape = ((data['ei'].sum())/(data['y'].sum()))*100
    mean = (data['y'].sum())/n
    ssRes = data['mse'].sum()
    data['sstotal'] = data.apply(lambda x: (x['y'] - mean), axis=1)
    data['sstotal'] = data['sstotal'].apply(lambda x: x*x)
    ssTotal = data['sstotal'].sum()
    rSquared = 1 - (ssRes/ssTotal)
    return mse, mape, rSquared
mse, mape, rSquared = regress_metrics(df_d)
```

1.3.1 Mean Squared Error

In [25]:
```
print('Mean Squared Error :',mse)
```

```
Mean Squared Error : 177.16569974554707
```

1.3.2 MAPE

In [26]:
```
print('Mean Absolute Percentage Error :',mape)
```

```
Mean Absolute Percentage Error : 12.91202994009687
```

1.3.3 R-Squared Error

In [27]:
```
print('R-Squared Error :',rSquared)
```

```
R-Squared Error : 0.9563582786990937
```