Compute performance metrics for the given Y and Y_score without sklearn

A. Compute performance metrics for the given data 5_a.csv

```
Note 1. In this data you can see number of positive points >> nu
mber of negatives points
Note 2. Use pandas or numpy to read the data from <b>5_a.csv
Note 3. You need to derive the class labels from given score
```

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

a. Compute Confusion Matrix

b. Compute F1 Score

c. 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)

d. Compute Accuracy Score

In [1]:
```
from google.colab import drive
drive.mount('/content/gdrive')
```

```
Drive already mounted at /content/gdrive; to attempt to forcibly remoun
t, call drive.mount("/content/gdrive", force_remount=True).
```

In [2]:
```
def confusion_matrix(df):
  tp,tn,fp,fn = [],[],[],[]
  for indx,row in (df.iterrows()):
    if row['y_score']==1 and row['y']==1:
```

```
      tp.append(df.iloc[indx])
    elif row['y_score']==0 and row['y']==0:
      tn.append(df.iloc[indx])
    elif row['y_score']==1 and row['y']==0:
      fp.append(df.iloc[indx])
    else: #row['y_score']==0 and row['y']==1:
      fn.append(df.iloc[indx])
  #print(len(tn),len(fn),len(fp),len(tp))
  #conf_matrix = [len(tn),len(fn),len(fp),len(tp)]
  conf_matrix = np.reshape([len(tn),len(fn),len(fp),len(tp)],(2,2))
  acc_score = (len(tn)+len(tp))/(len(tn)+len(fn)+len(fp)+len(tp))
  return conf_matrix,acc_score

def f1_score(df):
  precision = df[df['y_score']==1]
  pr = precision[precision['y']==1].shape[0]/precision.shape[0]
  print('precision :',pr,'\n')

  recall = df[df['y']==1]
  rc = recall[recall['y_score']==1].shape[0]/recall.shape[0]
  print('recall :',rc,'\n')

  f1_sc = 2*((pr*rc)/(pr+rc))
  #print('f1_score :',f1_sc,'\n')
  return f1_sc

def AUC_score(df):
  x = df.sort_values(by=['proba'],ascending=False,ignore_index=True)
  x['y_score'] = 0
  D_tpr = dict()
  D_fpr = dict()
  for i in range(len(x)):
    x['y_score'].iloc[i] = 1
    d  = dict(x['y_score'].value_counts())

    d1 = dict(x['y'].iloc[:i].value_counts())
    #print('fp',d1.get(0,0))
    #print('tp',d1.get(1,0))
```

```
    d2 = dict(x['y'].iloc[i:].value_counts())
    #print('tn',d2.get(0,0))
    #print('fn',d2.get(1,0))

    P = d1.get(1,0) + d2.get(1,0)
    N = d2.get(0,0) + d1.get(0,0)

    tpr = (d1.get(1,0))/P
    fpr = (d1.get(0,0))/N
    D_tpr[x['proba'].iloc[i]]=tpr
    D_fpr[x['proba'].iloc[i]]=fpr
  #print(D_tpr,'\n',len(D_tpr),'\n',df.shape[0],'\n',D_fpr,'\n',len(D_f
pr))
  fpr_array = np.array(list(D_fpr.values()))
  tpr_array = np.array(list(D_tpr.values()))
  auc_score = np.trapz(tpr_array, fpr_array)
  print(len(D_tpr),len(D_fpr))
  return auc_score
```

In [3]:

```
import warnings
warnings.filterwarnings("ignore")
import numpy as np
import pandas as pd


df1 = pd.read_csv('/content/gdrive/My Drive/AAIC/Assignments/7.Compute
 Performance metrics without Sklearn/Copy of 5_a.csv')
df1['y_score']=0
for indx,row in df1.iterrows():
  if row['proba']>0.5:
    df1['y_score'].iloc[indx] = 1
cm,acc_score = confusion_matrix(df1)
print('Confusion-Matrix :\n',cm,'\n')
print('f1_score : ',f1_score(df1),'\n')
print('AUC_Score :', AUC_score(df1),'\n')
print('Accuraccy_Score :',acc_score,'\n')
```

```
Confusion-Matrix :
 [[    0     0]
```

```
 [  100 10000]]

precision : 0.9900990099009901

recall : 1.0

f1_score :  0.9950248756218906

10100 10100
AUC_Score : 0.48829900000000004

Accuraccy_Score : 0.9900990099009901
```

In [ ]:

In [ ]:

B. Compute performance metrics for the given data 5_b.csv

```
Note 1: in this data you can see number of positive points << nu
mber 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
```

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

a. Compute Confusion Matrix

b. Compute F1 Score

c. 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)

```
href='https://stackoverflow.com/q/53603376/4084039 https://stack
overflow.com/q/53603376/4084039
href='https://stackoverflow.com/a/39678975/4084039 https://stack
```

overflow.com/a/39678975/4084039

d. Compute Accuracy Score

In [4]:
```
import warnings
warnings.filterwarnings("ignore")
import numpy as np
import pandas as pd


df2=pd.read_csv('/content/gdrive/My Drive/AAIC/Assignments/7.Compute Performance metrics without Sklearn/Copy of 5_b.csv')
df2['y_score']=0
for indx,row in df2.iterrows():
  if row['proba']>0.5:
    df2['y_score'].iloc[indx] = 1
cm,acc_score = confusion_matrix(df2)
print('Confusion-Matrix :\n',cm,'\n')
print('f1_score : ',f1_score(df2),'\n')
print('AUC_Score :', AUC_score(df2),'\n')
print('Accuraccy_Score :',acc_score,'\n')
```

```
Confusion-Matrix :
 [[9761   45]
 [ 239   55]]

precision : 0.1870748299319728

recall : 0.55

f1_score :  0.2791878172588833

10100 10100
AUC_Score : 0.9376570000000001

Accuraccy_Score : 0.9718811881188119
```

In [ ]:

In [ ]:

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 [5]:
```
df3 = pd.read_csv('/content/gdrive/My Drive/AAIC/Assignments/7.Compute
 Performance metrics without Sklearn/Copy of 5_c.csv')
```

In [6]:
```
def Metric_A(df):
  x = df.sort_values(by=['prob'],ascending=False,ignore_index=True)
  x['y_score'] = 0
  #D_fn = dict()
  #D_fp = dict()
  D_A  = dict()
  for i in range(len(x)):
    x['y_score'].iloc[i] = 1
    d  = dict(x['y_score'].value_counts())

    d1 = dict(x['y'].iloc[:i].value_counts())
    #print('fp',d1.get(0,0))
    #print('tp',d1.get(1,0))

    d2 = dict(x['y'].iloc[i:].value_counts())
```

```
    #print('tn',d2.get(0,0))
    #print('fn',d2.get(1,0))

    A = (500*(d2.get(1,0))) + (100*(d1.get(0,0)))

    #D_fn[x['prob'].iloc[i]]= d2.get(1,0)
    #D_fp[x['prob'].iloc[i]]= d1.get(0,0)
    D_A[x['prob'].iloc[i]]= A
  best_threshold = min(D_A,key=D_A.get) # https://stackoverflow.com/questions/3282823/get-the-key-corresponding-to-the-minimum-value-within-a-dictionary

  return best_threshold
```

In [7]:
```
best_threshold = Metric_A(df3)
df3['y_score']=0
for indx,row in df3.iterrows():
  if row['prob']>best_threshold:
    df3['y_score'].iloc[indx] = 1
df3 = pd.DataFrame(df3)
print('best_threshold :',best_threshold)
```

```
best_threshold : 0.22987164436159915
```

In [69]:

In [ ]:

In [ ]:

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 a
re real valued features
```

Compute Mean Square Error

Compute MAPE: https://www.youtube.com/watch?v=ly6ztgIkUxk

Compute R^2 error: https://en.wikipedia.org/wiki/Coefficient_of_determination#Definitions

In [8]:
```
df4 = pd.read_csv('/content/gdrive/My Drive/AAIC/Assignments/7.Compute
 Performance metrics without Sklearn/Copy of 5_d.csv')
```

In [9]:
```
mse = np.mean(((df4['y']-df4['pred'])**2))
mape = (abs(df4['pred']-df4['y']).sum(axis=0))/(df4['y'].sum(axis=0))
ss_tot = ((df4['y']-(np.mean(df4['y'])))**2).sum(axis=0)
ss_res = ((df4['y']-df4['pred'])**2).sum(axis=0)
r2_err = 1-(ss_res/ss_tot)

print('Mean Square Error :', mse,'\n')
print('Mean Absolute % Error :', mape,'\n')
print('R-square Error :', r2_err,'\n')
```

```
Mean Square Error : 177.16569974554707

Mean Absolute % Error : 0.1291202994009687

R-square Error : 0.9563582786990937
```

In [28]:

In [45]:

In [46]:

In [9]:

In [47]:

In [ ]: