Compute performance metrics for the given Y and Y_score without sklearn

In [1]:
```python
import numpy as np
import pandas as pd
from tqdm import tqdm
# other than these two you should not import any other packages
```

In [107]:
```python
#Ref
#https://www.nbshare.io/notebook/626706996/Learn-And-Code-Confusion-Matrix-With-Python/
#https://www.askpython.com/python-modules/pandas/update-the-value-of-a-row-dataframe
#https://www.youtube.com/watch?v=HJkInJV29t8&t=12s
#https://datagy.io/mean-squared-error-python/



def confusion_mat(df):
  #df['predicted'] = df['predicted'].apply(lambda x:1 if x >= 0.5 else 0 )
  #df should have 2 cols name actual and predicted

  TN = ((df['actual'] == 0) & (df['predicted'] == 0)).sum()
  TP = ((df['actual'] == 1) & (df['predicted'] == 1)).sum()
  FN = ((df['actual'] == 1) & (df['predicted'] == 0)).sum()
  FP = ((df['actual'] == 0) & (df['predicted'] == 1)).sum()
  return TN,TP,FN,FP


def f1_score(df):
  TN,TP,FN,FP = confusion_mat(df)
  precision = TP/(TP + FP)
  recall = TP/(FN + TP)
  f1 = 2*((precision*recall)/(precision + recall))
  return f1


def accuracy_mat(df):
  TN = ((df['actual'] == 0) & (df['predicted'] == 0)).sum()
  TP = ((df['actual'] == 1) & (df['predicted'] == 1)).sum()
  return (TN+TP)/len(df)


def auc(df):
  tpr = []
  fpr = []
  df = df.sort_values(by = ['predicted'], ascending = False)
  #print(df,' This is df')


  for i in tqdm(range(0,len(df))):
    df_c = df.copy()
    df_c['predicted'] = np.where(df_c['predicted'] >= df_c.iloc[i]['predicted'],1,0)
    #print(df_c)

    TN,TP,FN,FP = confusion_mat(df_c)

    fpr.append(FP/(TN + FP))
    tpr.append(TP/(TP + FN))
    #print(tpr)
    #print(fpr,'Fpr')

  auc = np.trapz(tpr,fpr)
  return auc
```

In [ ]:
```python
# Sir I tried diffearent way to update thresholds
# df_copy['predicted'] = df_copy['predicted'].apply(lambda x:1 if x >= i else 0 )
#df_copy.loc[df_copy['predicted'] < i] = 0
#df_copy.loc[df_copy['predicted'] >= i] = 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

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

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 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 [8]:
```python
df_a=pd.read_csv('5_a.csv')

df_a.columns = ['actual','predicted']
print(df_a.head())
df_a['predicted'] = df_a['predicted'].apply((lambda x : 1 if x > 0.5 else 1))
print(df_a.head())
```

```
   actual  predicted
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
   actual  predicted
0     1.0          1
1     1.0          1
2     1.0          1
3     1.0          1
4     1.0          1
```

In [13]:
```python
TN,TP,FN,FP = confusion_mat(df_a)
print('True Negative' ,TN)
print('True Positive', TP)
print('False Positive', FP)
print('False Negative',FN)

print('\n')

accuracy = accuracy_mat(df_a)
print('Accuracy:',accuracy)

print('\n')

f1_score = f1_score(df_a)
print('F1 score ',f1_score)
```

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


Accuracy: 0.9900990099009901


F1 score  0.9950248756218906
```

In [15]:
```python
df_a=pd.read_csv('5_a.csv')

df_a.columns = ['actual','predicted']

auc_val = auc(df_a)
print(auc_val)
```

```
100%|██████████| 10100/10100 [00:31<00:00, 315.64it/s]
0.48829900000000004
```

In [ ]:
```python
# write your code here for task A
```

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
   Note- Make sure that you arrange your probability scores in descending order while calculating AUC
4. Compute Accuracy Score

In [103]:
```python
df_b=pd.read_csv('5_b.csv')
df_b.head()

df_b.columns = ['actual','predicted']
print(df_b.head())
df_b['predicted'] = df_b['predicted'].apply((lambda x : 0 if x < 0.5 else 1))
print(df_b.head())
```

```
   actual  predicted
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
   actual  predicted
0     0.0          0
1     0.0          0
2     0.0          0
3     0.0          0
4     0.0          0
```

In [104]:
```python
df_b['predicted'].value_counts()
```

Out[104]:
```
0    9806
1     294
Name: predicted, dtype: int64
```

In [108]:
```python
# write your code here for task B

TN,TP,FN,FP = confusion_mat(df_b)
print('True Negative' ,TN)
print('True Positive', TP)
print('False Positive', FP)
print('False Negative',FN)

print('\n')

accuracy = accuracy_mat(df_b)
print('Accuracy:',accuracy)

print('\n')

f1_score = f1_score(df_b)
print('F1 score ',f1_score)
```

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


Accuracy: 0.9718811881188119


F1 score  0.2791878172588833
```

In [109]:
```python
df_b=pd.read_csv('5_b.csv')

df_b.columns = ['actual','predicted']

auc_val = auc(df_b)
print(auc_val)
```

```
100%|██████████| 10100/10100 [00:26<00:00, 385.48it/s]
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

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 [110]:
```python
df_c=pd.read_csv('5_c.csv')

df_c.columns = ['actual','predicted']
print(df_c.head())
```

```
   actual  predicted
0       0   0.458521
1       0   0.505037
2       0   0.418652
3       0   0.412057
4       0   0.375579
```

In [111]:
```python
 # write your code for task C

def best_threshold(df):

  threshold = []
  A_val = []
  df = df.sort_values(by = ['predicted'], ascending = False)
  #print(df,' This is df')


  for i in tqdm(range(0,len(df))):
    df_c = df.copy()
    threshold.append(df_c.iloc[i]['predicted'])
    #print(df_c.iloc[i]['predicted'], ' I am thres')
    df_c['predicted'] = np.where(df_c['predicted'] >= df_c.iloc[i]['predicted'],1,0)
    #print(df_c)


    TN,TP,FN,FP = confusion_mat(df_c)
    A_val.append(500*FN + 100* FP)


  return threshold[np.argmin(A_val)]

A_min = best_threshold(df_c)
print(A_min)
```

```
100%|██████████| 2852/2852 [00:08<00:00, 354.25it/s]
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 [113]:
```python
df_d=pd.read_csv('5_d.csv')
df_d.head()
df_d.columns = ['actual', 'predicted']
```

In [114]:
```python
 # write your code for task 5d

def metrics(df):
  y = np.array(df['actual'])
  y_hat = np.array(df['predicted'])

  mse = np.sum(np.power(y - y_hat, 2))/len(y)

  mpe = np.sum(np.abs(y - y_hat)) / np.sum(y)

  y_mean = np.mean(np.abs(y))
  residual_square = np.sum(np.power(y-y_hat, 2))
  total_sum = np.sum(np.power(y- y_mean,2))
  r2 = 1 - (residual_square/total_sum)

  return mse,mpe,r2



mse, mpe,r2 = metrics(df_d)

print('MSE', mse)
print('MPE', mpe)
print('r_square', r2)
```

```
MSE 177.16569974554707
MPE 0.1291202994009687
r_square 0.9563582786990937
```