Question 1

In [457]:

```
import numpy as np
import pandas as pd
```

In [458]:

```
data=pd.read_csv("C:/Users/91888/Desktop/Assignment/PerformanceMetric Assignment/5_a.csv")
```

In [459]:

```
data.head()
```

Out[459]:

	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 [460]:

```
data.shape
```

Out[460]:

```
(10100, 2)
```

In [461]:

```
def predict(data,proba,y_score):
    y_pred=[]
    for i in data['proba']:
        if i<y_score:
            y_pred.append(0)
        else:
            y_pred.append(1)
    return y_pred
```

In [462]:
```
y_score=0.5
data['y_pred']=predict(data,'proba',y_score)
```

In [463]:
```
data.head()
```

Out[463]:

	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

In [464]:
```
def confusion_matrix(y_pred,y):

  fp = np.sum((y_pred == 1) & (y == 0))
  tp = np.sum((y_pred == 1) & (y == 1))

  fn = np.sum((y_pred == 0) & (y == 1))
  tn = np.sum((y_pred == 0) & (y == 0))
  return {'tn':tn,'tp':tp,'fn':fn,'fp':fp}
cm=confusion_matrix(data.y_pred,data.y)
cm
```

Out[464]:
```
{'tn': 0, 'tp': 10000, 'fn': 0, 'fp': 100}
```

In [465]:
```
def f1_score(TP,FP):
    P=(data['y']== 1).sum()
    precision=TP/(TP+FP)
    Recall=TP/P
    f1=2*(precision*Recall/(precision+Recall))
    return f1
f1=f1_score(cm['tp'],cm['fp'])
```

In [466]:

```
print("f1 score : ",f1)
```

```
f1 score :  0.9950248756218906
```

In [467]:

```
def AUC_score(data):
    P=(data['y']== 1).sum()
    N=(data['y']== 0).sum()
    tpr=[]
    fpr=[]
    thresholds = np.arange(1, 0, -0.1)
    for y_score in thresholds:
        y_score = round(y_score,2)
        data['y_pred']=predict(data,'proba',y_score)
        cm=confusion_matrix(data.y_pred,data.y)
        tpr.append(cm['tp']/P)
        fpr.append(cm['fp']/N)
        data.drop(columns=['y_pred'])
    return np.trapz(tpr,fpr)
AUC=AUC_score(data)
```

In [468]:

```
print("AUC Score is : ",AUC)
```

```
AUC Score is :  0.48897750000000006
```

In [469]:

```
def Accuracy_score(data,tp,tn):
    total_pts=data['y'].count()
    acc=(tp+tn)/total_pts
    return acc
Accuracy=Accuracy_score(data,cm['tp'],cm['tn'])
```

In [470]:

```
print("Accuracy is : ", Accuracy)
```

```
Accuracy is :  0.9900990099009901
```

Question 2

In [176]:
```
data=pd.read_csv("C:/Users/91888/Desktop/Assignment/PerformanceMetric Assignment/5_b.csv")
```

In [177]:
```
data.shape
```

Out[177]:
```
(10100, 2)
```

In [178]:
```
data.head()
```

Out[178]:

	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 [179]:
```
def predict(data,proba,y_score):
    y_pred=[]
    for i in data['proba']:
        if i<y_score:
            y_pred.append(0)
        else:
            y_pred.append(1)
    return y_pred
```

In [180]:
```
y_score=0.5
data['y_pred']=predict(data,'proba',y_score)
```

In [95]:

```
data.head()
```

Out[95]:

	y	proba	y_pred
0	0.0	0.281035	0
1	0.0	0.465152	0
2	0.0	0.352793	0
3	0.0	0.157818	0
4	0.0	0.276648	0

In [181]:

```
def confusion_matrix(y_pred,y):

  fp = np.sum((y_pred == 1) & (y == 0))
  tp = np.sum((y_pred == 1) & (y == 1))

  fn = np.sum((y_pred == 0) & (y == 1))
  tn = np.sum((y_pred == 0) & (y == 0))
  return {'tn':tn,'tp':tp,'fn':fn,'fp':fp}
cm=confusion_matrix(data.y_pred,data.y)
cm
```

Out[181]:

```
{'tn': 9761, 'tp': 55, 'fn': 45, 'fp': 239}
```

In [182]:

```
def f1_score(TP,FP):
    P=(data['y']== 1).sum()
    precision=TP/(TP+FP)
    Recall=TP/P
    f1=2*(precision*Recall/(precision+Recall))
    return f1
f1=f1_score(cm['tp'],cm['fp'])
print("f1 score : ",f1)
```

```
f1 score :  0.2791878172588833
```

In [183]:

```
def AUC_score(data):
    P=(data['y']== 1).sum()
    N=(data['y']== 0).sum()
    tpr=[]
    fpr=[]
    thresholds = np.arange(1,0,-0.1)
    for y_score in thresholds:
        y_score = round(y_score,2)
        data['y_pred']=predict(data,'proba',y_score)
        cm=confusion_matrix(data.y_pred,data.y)
        tpr.append(cm['tp']/P)
        fpr.append(cm['fp']/N)
        data.drop(columns=['y_pred'])
    return np.trapz(tpr,fpr)
AUC=AUC_score(data)
```

In [184]:

```
print("AUC score is : ", AUC)
```

```
AUC score is :  0.9276825
```

In [185]:

```
def Accuracy_score(data,tp,tn):
    total_pts=data['y'].count()
    acc=(tp+tn)/total_pts
    return acc
Accuracy=Accuracy_score(data,cm['tp'],cm['tn'])
```

In [186]:

```
print("Accuracy is : ",Accuracy)
```

```
Accuracy is :  0.9718811881188119
```

Question 3

In [471]:

```
data=pd.read_csv("C:/Users/91888/Desktop/Assignment/PerformanceMetric Assignment/5_c.csv")
```

In [136]:

```
data.shape
```

Out[136]:

```
(2852, 2)
```

In [188]:

```
data.head()
```

Out[188]:

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

In [472]:

```
def predict(data,proba,y_score):
    y_pred=[]
    for i in data['prob']:
        if i<y_score:
            y_pred.append(0)
        else:
            y_pred.append(1)
    return y_pred
data['y_pred']=predict(data,'prob',y_score)
```

In [473]:

```
def confusion_matrix(y_pred,y):

  fp = np.sum((y_pred == 1) & (y == 0))
  tp = np.sum((y_pred == 1) & (y == 1))

  fn = np.sum((y_pred == 0) & (y == 1))
  tn = np.sum((y_pred == 0) & (y == 0))
  return {'tn':tn,'tp':tp,'fn':fn,'fp':fp}
```

In [474]:

```
def metric_value(data):
    A={}
    thresholds = np.arange(1,0,-0.01)
    for y_score in thresholds:
        y_score = round(y_score,2)
        data['y_pred']=predict(data,'prob',y_score)
        cm=confusion_matrix(data.y_pred,data.y)
        metric_val=(500*cm['fn'])+(100*cm['fp'])
        A[y_score]=metric_val

        data.drop(columns=['y_pred'])
    return A
metric=metric_value(data)
```

In [475]:

```
minv = min(metric.values())
key = [k for k, v in metric.items() if v==minv]
```

In [476]:

```
print("Threshold giving lowest values of metric A",key)
print("Minimum value of metric A is ",minv)
```

```
Threshold giving lowest values of metric A [0.23]
Minimum value of metric A is  141000
```

Question 4

In [477]:

```
data=pd.read_csv("C:/Users/91888/Desktop/Assignment/PerformanceMetric Assignment/5_d.csv")
```

In [478]:

```
data.shape
```

Out[478]:

```
(157200, 2)
```

In [340]:
```
data.head()
```

Out[340]:

	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 [479]:
```
def error(y,pred):
    val=[]
    for i in range(len(y)):
        e=y[i]-pred[i]
        val.append(e)
    return val
err=error(data['y'],data['pred'])
```

In [480]:
```
data['error']=err
```

In [481]:
```
#converting errors to absolute errors
A=data.error.abs()
data['abs2']=A
```

In [482]:

```
data.head()
```

Out[482]:

	y	pred	error	abs2
0	101.0	100.0	1.0	1.0
1	120.0	100.0	20.0	20.0
2	131.0	113.0	18.0	18.0
3	164.0	125.0	39.0	39.0
4	154.0	152.0	2.0	2.0

In [483]:

```
def MSE(error):
    sum=0
    for i in range(len(error)):
        sum=sum+(np.square(error[i]))
    mse=sum/len(error)

    return mse
MSE=MSE(data['error'])
```

In [484]:

```
print("Mean Square Error is : ",MSE)
```

```
Mean Square Error is :  177.16569974554707
```

In [485]:

```
def MAPE(absolute,y):
    mape=np.sum(data.abs2)/np.sum(data.y)
    return mape
```

In [486]:

```
M=MAPE(data.abs2,data.y)
print("MAPE is : ",M)
```

```
MAPE is :  0.1291202994009687
```

In [487]:

```
def st(y):
    s=0
    mean=data.y.mean()
    for i in range(len(y)):
        s=s+(y[i]-mean)*(y[i]-mean)


    return s
stotal=st(data.y)
```

In [488]:

```
def s_res(error):
    sum1=0
    for i in range(len(error)):
        sum1=sum1+(np.square(error[i]))


    return sum1
s_res=s_res(data['error'])
```

In [489]:

```
RSQUARE=1-(s_res/stotal)
```

In [490]:

```
print(" R square error is : ",RSQUARE)
```

```
 R square error is :  0.9563582786990964
```