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 $ypred$=[0 if y_score < 0.5 else 1]

Compute Confusion Matrix

Compute F1 Score

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

Compute Accuracy Score

```
import numpy as np
import pandas as pd
from tqdm.notebook import tqdm
```

```
from google.colab import files
files = files.upload()
```

Choose Files 5_a.csv

- **5_a.csv**(text/csv) - 241203 bytes, last modified: 8/4/2022 - 100% done

```
Saving 5_a.csv to 5_a.csv
```

```
df_a = pd.read_csv("5_a.csv")
```

```
df_a.shape
```

```
(10100, 2)
```

in below code snippet i have declared the function predict for set threshold value between 0 and 1

```
def predict(df,y,thresh_hold):
    y_pred=[]
    for label in df[y]:
        if label<thresh_hold:
            y_pred.append(0)
        else:
            y_pred.append(1)
    return y_pred


# so now here we will i m creating a function confusion matrix for calclulate the value of co
def cal_the_val(df,y,y_pred):
    true_positive=0
    true_negative=0
    false_negative=0
    false_positive=0
    for val1,val2 in enumerate(df['y']):
        if(df.y_pred[val1]==1) and df.y[val1]==1:     # here if our predicted value 1 actual
            true_positive=true_positive+1
        if(df.y_pred[val1]==0) and df.y[val1]==0:  # here if predicted and actual value both
            true_negative=true_negative+1
        if(df.y_pred[val1]==0) and df.y[val1]==1:  # here if predicted value 0 and actual val
            false_negative=false_negative+1
        if(df.y_pred[val1]==1) and df.y[val1]==0:  # here if predicted value is 1 and actual
            false_positive=false_positive+1

            # now we will return are variable
    return {'true_negative':true_negative,'true_positive':true_positive,'false_negative':fals
```

in below code snippet we have set the throshold value as 0.5 if less than 0.5 then predicted value is 0 greater than 0.5 then predicted value is

```
thresh_hold=0.5
df_a['y_pred']=predict(df_a,'proba',thresh_hold)
confusion_matrix=cal_the_val(df_a,'y','y_pred')
```

confusion matrix

```
print("the confusion matrix is: ",confusion_matrix)
```

```
the confusion matrix is:  {'true_negative': 0, 'true_positive': 10000, 'false_negative'
```

f1 score

refer **bold text : https://www.kaggle.com/code/paulrohan2020/performance-metrics-without-sklearn/notebook the formula of f1 score is

f1 = 2*precision*recall/(precision + recall)**

```
x=df_a.y.value_counts()
A=x[1]

precision=confusion_matrix['true_positive']/(confusion_matrix['true_positive']+confusion_matr
recall=confusion_matrix['true_positive']/A

F1=2*precision*recall/(precision+recall)
print('the F1 score is: ',F1)
```

```
     the F1 score is:  0.9950248756218906
```

Accuracy

formula of accuracy = true positive + true negative / total value of dataset

```
# here simple execution of formula


Acc=(confusion_matrix['true_positive']+confusion_matrix['true_negative'])/df_a.shape[0]
print('the accuracy is: ',Acc)
```

```
     the accuracy is:  0.9900990099009901
```

```
from tqdm import tqdm_notebook      # tqdm library i have imported just because of see the pr
def auc(df):   # here created the function auc with parameterof df
    s = df['y'].value_counts()
    P = s[1]
    N = s[0]
    tpr = []  # here created the emplty variable for true positive rate
    fpr = []  # here created the emplty variable for false positive rate


    for elem in tqdm(df['proba']):   # here will check every element from proba value from da
        df['y_pred']=predict(df,'proba',elem)
        confusion_matrix=cal_the_val(df,'y','y_pred')
        tpr.append(confusion_matrix['true_positive']/P)   # TPR IS TRUE POSITIVE RATE
        fpr.append(confusion_matrix['false_positive']/N)  # FPR IS FALSE POSITIVE RATE
        df.drop(columns=['y_pred'])
    return np.trapz(tpr,fpr)
```

Double-click (or enter) to edit

```
df_a=df_a.sort_values(by='proba',ascending=False)
df_a.drop(columns=['y_pred'])
```

	y	proba
1664	1.0	0.899965
2099	1.0	0.899828
1028	1.0	0.899825
9592	1.0	0.899812
8324	1.0	0.899768
...	...	...
8294	1.0	0.500081
1630	1.0	0.500058
7421	1.0	0.500058
805	1.0	0.500047
5012	1.0	0.500019

10100 rows × 2 columns

here we calculate the auc score of whole dataset

refer : https://stackoverflow.com/questions/65748968/how-to-compute-auc-score-manually-without-using-sklearn

```
AUC_score=auc(df_a)
print ('the AUC Score is :',AUC_score)
```

```
the AUC Score is : 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
```

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

```
1.  Compute Confusion Matrix

2.  Compute F1 Score

3.  Compute AUC Score, you need to compute different thresholds and for each t
   Note- Make sure that you arrange your probability scores in descending orde

4.  Compute Accuracy Score
```

```
from google.colab import files
files = files.upload()
```

Choose Files 5_b.csv

- **5_b.csv**(text/csv) - 247322 bytes, last modified: 8/4/2022 - 100% done

```
Saving 5_b.csv to 5_b.csv
```

```
data_B=pd.read_csv('5_b.csv')
data_B.shape
```

```
(10100, 2)
```

again setting the threhsold value

```
thresh_hold=0.5
data_B['y_pred']=predict(data_B,'proba',thresh_hold)
confusion_matrix_B=cal_the_val(data_B,'y','y_pred')
```

confusion matrix

```
print('the confusion matrix is :', confusion_matrix_B)
```

```
the confusion matrix is : {'true_negative': 9761, 'true_positive': 55, 'false_negative'
```

F1 score

as per formula of f1 score after calculating the precision and recall

```
x=data_B.y.value_counts()
P=x[1]

precision_B=confusion_matrix_B['true_positive']/(confusion_matrix_B['true_positive']+confusio
recall_B=confusion_matrix_B['true_positive']/P

F1_B=2*precision_B*recall_B/(precision_B+recall_B)
print('the F1 Score is : ',F1_B)
```

```
     the F1 Score is :  0.2791878172588833
```

calculating the accuracy of dataset B

```
Acc_B=(confusion_matrix_B['true_positive']+confusion_matrix_B['true_negative'])/data_B.shape[
print('the Accuracy is :',Acc_B)
```

```
     the Accuracy is : 0.9718811881188119
```

Double-click (or enter) to edit

```
data_B=data_B.sort_values(by='proba',ascending=False)
data_B.drop(columns=['y_pred'])
```

	y	proba
8446	1.0	0.595294
1978	1.0	0.594808
1657	1.0	0.592198

again like dataset A here also we are calcuting the auc score of B

```
AUC_score_B=auc(data_B)
print('the AUC Score is: ',AUC_score_B)
```

```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:8: TqdmDeprecationWarning:
Please use `tqdm.notebook.tqdm` instead of `tqdm.tqdm_notebook`

the AUC Score is:  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:

$ypred$=[0 if y_score < threshold else 1]

A=500×number of false negative+100×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

```
from google.colab import files
files = files.upload()
```

Choose Files 5_c.csv

- **5_c.csv**(text/csv) - 63471 bytes, last modified: 8/4/2022 - 100% done

```
Saving 5_c.csv to 5_c.csv
```

```
df_c = pd.read_csv("5_c.csv")
```

```
def min_metric(df_b):
    s = df_a['y'].value_counts()
    A = s[1]
    B = s[0]
    tpr = []   # TRUE POSITIVE RATE
    fpr = []   # FALSE POSITIVE RATE
    metric = {}
    for ele in tqdm_notebook(df_c['prob']):
        df_c['y_pred']= predict(df_c, 'prob' , ele)
        confusion_matrix =cal_the_val( df_c,'y','y_pred')
        metric_value=(500*confusion_matrix['false_negative'])+(100*confusion_matrix['false_po
        metric[ele]=metric_value
        df_c.drop(columns=['y_pred'])
    return(metric)
```

Double-click (or enter) to edit

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

df_c=df_c.sort_values(by='prob',ascending=False)
result=min_metric(df_c)
```

```
   y      prob
0  0  0.458521
1  0  0.505037
2  0  0.418652
3  0  0.412057
4  0  0.375579
(2852, 2)
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:8: TqdmDeprecationWarning:
Please use `tqdm.notebook.tqdm` instead of `tqdm.tqdm_notebook`
```

```
df_c=df_c.sort_values(by='prob',ascending=False)
result=min_metric(df_c)
```

```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:8: TqdmDeprecationWarning:
Please use `tqdm.notebook.tqdm` instead of `tqdm.tqdm_notebook`
```

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
```

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

```
from google.colab import files
files = files.upload()
```

Choose Files 5_d.csv

- **5_d.csv**(text/csv) - 1742949 bytes, last modified: 8/4/2022 - 100% done

```
Saving 5_d.csv to 5_d.csv
```

```
df_d=pd.read_csv('5_d.csv')
df_d.head()
```

	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

```
df_d.shape
```

```
(157200, 2)
```

refer : https://stats.stackexchange.com/questions/58391/mean-absolute-percentage-error-mape-in-scikit-learn

```
# refer : https://en.wikipedia.org/wiki/Coefficient_of_determination#Definitions
# refer : https://stats.stackexchange.com/questions/58391/mean-absolute-percentage-error-mape
```

```
def error(df,col1,col2):  # here we are calculting the error .
    val=[]
    for index, (value1, value2) in enumerate(zip(df[col1], df[col2])):  # error is actual val
        val.append(value1-value2)
    return val

def absolute_error(df,col):
    val=[]
    for index,value in enumerate(df[col]):
        val.append(abs(value))
    return val

# define the function to calculate the mean sqaured error
def mean_squared_error(df,col):
    return sum_res(df,col)/len(df[col])



# define a function the for mean absolute percentag error as MAPE
def mape(df,col1,col2):
    val=sum(df[col1])/sum(df[col2])
    return val



# here we are calculating the sum of error and its also callsed as sum of residules  refer ab
def sum_res(df,col):
    val=0
    for index,value in enumerate(df[col]):
        val=val+(value*value)
    return val

# refer : In some cases, as in simple linear regression, the total sum of squares equals the




def sum_tot(df,col):
    val=0
    mean_val=df_d['y'].mean()
    for index,value in enumerate(df[col]):
        val=val+ (value-mean_val)*(value-mean_val)
    return val
```

here we are calculating the error by substrating predicting value from actual value

```
df_d['error']=error(df_d,'y','pred')
df_d['abs_error']=absolute_error(df_d,'error')
```

Double-click (or enter) to edit

mean sqaured error

```
MSE=mean_squared_error(df_d,'error')
print("the Mean squared error is : ", MSE)

     the Mean squared error is :  177.16569974554707
```

mean absolute percentage error

```
MAPE=mape(df_d,'abs_error','y')
print('the mean absolute percentage error  value is :', MAPE)

     the mean absolute percentage error  value is : 0.1291202994009687
```

calculate the sum of resuduals

sum of total errors

```
sum_RES=sum_res(df_d,'error')
sum_TOT=sum_tot(df_d,'y')


# refer : https://www.geeksforgeeks.org/python-coefficient-of-determination-r2-score/
# refer above link for calulating r^2

R_square= 1- (sum_RES/sum_TOT)
print('the Co-efficient of determination value is: ',R_square)

     the Co-efficient of determination value is:  0.9563582786990964
```

●

Could not connect to the reCAPTCHA service. Please check your internet connection and reload to get a reCAPTCHA challenge.