```
import numpy as np
import pandas as pd

PART A

df_a=pd.read_csv('5_a.csv')
df_a.head

df_a = df_a.sort_values( by= ["proba"], ascending = False)

# write your code here for task A
#convert probability to class labels
df_a['y_pred']=df_a['proba'].apply(lambda x1: 1 if x1>=0.5 else 0)

#creating a confusion matrix (cm)

def cm(data):

  tn_cnt = int(df_a[(data.y == 0) & (data.y_pred == 0)].count()[0])
  fn_cnt = int(df_a[(data.y == 1) & (data.y_pred == 0)].count()[0])
  fp_cnt = int(df_a[(data.y == 0) & (data.y_pred == 1)].count()[0])
  tp_cnt = int(df_a[(data.y == 1) & (data.y_pred == 1)].count()[0])
  return tn_cnt,fn_cnt,fp_cnt,tp_cnt


tn,fn,fp,tp = cm(df_a)

print("false negative",fn)
print("false positive",fp)
print("true negative ",tn)
print("true positive ",tp )

#accuracy value(av)
def accuracy(data):
    fn,fp,tn,tp=cm(data)
    accur = (tn + tp) / (tn + fn + fp + tp)
    return accur

accur = accuracy(df_a)
print("\n")
print("accuracy value", accur)


# Compute AUC score
def accur_score(data):
    tpr_arr=[]
    fpr_arr=[]
    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)
        fn,fp,tn,tp=cm(sort)
        ntp = ((data['y']==1.0) & (sort['y_pred'] == 1)).sum()
        nfn = ((data['y']==1.0) & (sort['y_pred'] == 0)).sum()
        nfp = ((data['y']==0.0) & (sort['y_pred'] == 1)).sum()
        ntn = ((data['y']==0.0) & (sort['y_pred'] == 0)).sum()
        fpr_rate=nfp/(ntn+nfp)
        tpr_rate=ntp/(ntp+nfn)
        tpr_arr.append(tpr_rate)
        fpr_arr.append(fpr_rate)
    d = np.trapz(tpr_arr, fpr_arr)
    return d

final = accur_score(df_a)
print("\n")
print('AUC value :',auc)


#calculating f1_score(fc)
def fc(data):
  precision = tp / (tp + fp)
  recall = tp / (fn + tp)
  f1_score = 2 * precision * recall / (precision + recall)
  return f1_score

f1_score= fc(df_a)
print("\n")
print("f1_score",f1_score)
```

```
false negative 0
false positive 100
true negative  0
true positive  10000


accuracy value 1.0


AUC value : 0.48829900000000004


f1_score 0.9950248756218906
```

PART B

```
df_b=pd.read_csv('5_b.csv')
df_b.head()
```

```
     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
```

```
import numpy as np
import pandas as pd

df_b=pd.read_csv('5_b.csv')
df_b.head()
#convert probability to class labels

df_b['y_pred']=df_b['proba'].apply(lambda x2: 1 if x2>=.5 else 0)

#creating a confusion matrix (cm)
def cm(data):

  tn_cnt = int(df_b[(df_b.y == 0) & (df_b.y_pred == 0)].count()[0])
  fn_cnt = int(df_b[(df_b.y == 1) & (df_b.y_pred == 0)].count()[0])
  fp_cnt = int(df_b[(df_b.y == 0) & (df_b.y_pred == 1)].count()[0])
  tp_cnt = int(df_b[(df_b.y == 1) & (df_b.y_pred == 1)].count()[0])
  return tn_cnt,fn_cnt,fp_cnt,tp_cnt

tn,fn,fp,tp = cm(df_b)
print("true postive", tp)
print("true negative", tn)#
print("false negative", fn)#
print("false postive",fp)


#calculating f1_score(fc)
def fc(data):
  tn,tp,fp,fn = cm(data)
  precision = tp/(tp+fp)
  recall = tp/(tp+fn)
  f1_score = 2*precision*recall/(precision+recall)
  return f1_score


f1_score= fc(df_b)
print("\n")
print("f1_score",f1_score)
```

```
# Compute AUC score
def accur_score(data):
    tpr_arr=[]
    fpr_arr=[]
    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)
        fn,fp,tn,tp=cm(sort)
        ntp = ((data['y']==1.0) & (sort['y_pred'] == 1.0)).sum()
        nfn = ((data['y']==1.0) & (sort['y_pred'] == 0.0)).sum()
        nfp = ((data['y']==0.0) & (sort['y_pred'] == 1.0)).sum()
        ntn = ((data['y']==0.0) & (sort['y_pred'] == 0.0)).sum()
        fpr_rate=nfp/(ntn+nfp)
        tpr_rate=ntp/(ntp+nfn)
        tpr_arr.append(tpr_rate)
        fpr_arr.append(fpr_rate)
    d=np.trapz(tpr_arr, fpr_arr)
    return d

auc=accur_score(df_b)
print("\n")
print('AUC value :',auc)


#accuracy value(av)
def accuracy(data):
  tn,tp,fp,fn = cm(data)
  accur = (tn+tp)/(tn+tp+fn+fp)
  return accur

accur = accuracy(df_b)
print("\n")
print("accuracy value", accur)
```

```
true postive 55
true negative 9761
false negative 45
false postive 239


f1_score 0.23437499999999997


AUC value : 0.9377570000000001
```

```
accuracy value 0.9708910891089109
```

PART C

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

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

```
 # write your code for task C

df_c['y_prd'] = df_a['proba'].apply(lambda x1: 1 if x1>0.5 else 0)


from tqdm import tqdm
uni = list(df_c.prob) #unique value of df_c
uni.sort()
x3 = {}

for i in tqdm(uni):

    df_c.loc[df_c['prob'] > i, 'y_pred'] = 1
    df_c.loc[df_c['prob'] < i, 'y_pred'] = 0

    fp = int(df_c[(df_c.y == 0) & (df_c.y_pred == 1)].count()[0])
    fn = int(df_c[(df_c.y == 1) & (df_c.y_pred == 0)].count()[0])

    final = (500 * fn) + (100 * fp)
    x3[i] = final

val = min(x3.values()) # to calculate minimum value

for thres in x3: #to calculate threshold
    if x3[thres] == val:
        print(f"The minimum value  {val} and  threshold is {thres}")
```

```
100%|██████████| 2852/2852 [00:15<00:00, 185.20it/s]

The minimum value  141000 and  threshold is 0.2300390278970873
```

PART D

```
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
```

```
 # write your code for task 5d
df_d = np.loadtxt('5_d.csv', delimiter=',', skiprows=1)
z_pred = df_d[:, 1]
z = df_d[:, 0]


#  Mean Square Error(mse)
mean_sqr_err = np.sum(np.power(z - z_pred, 2)) / len(df_d)
print("Mean Square Error: ", mean_sqr_err)
print("\n")

# Mean Absolute Percentage Error(mape)
mean_abs_per = np.sum(np.absolute(z - z_pred)) / np.sum(z)
print("Mean Absolute Percentage Error: ", mean_abs_per)
print("\n")

# R^2 error
mean_z = np.mean(np.absolute(z)) #mean

total = np.sum(np.power(z - mean_z, 2)) # total sum of squares

residues = np.sum(np.power(z - z_pred, 2)) #sum_squares_of_residues

coeffecient_determination = 1 - (residues / total)
print("R^2 error: ", coeffecient_determination)
```

```
Mean Square Error:  177.16569974554707


Mean Absolute Percentage Error:  0.1291202994009687


R^2 error:  0.9563582786990937
```