**Data Mining Lab**

**Assignment 04**

**Group** – C

**B.Tech 6th Semester**

**Name** – Neelangshu Roy

**Reg. No.** – 20214060

Note: Feature Selection- Select k useful features out of n futures in a dataset, where k<n.

Objective 1:

Load page\_block.csv numeric (Beginner's Guide Page-block Classification (kaggle.com)) dataset (make appropriate preprocessing if required). Use Fisher’s Score (f) ranking method for assigning an appropriate rank to each feature and select the top k ranked features.

μ1, μ2, σ1, σ2 are means and standard deviations for negative and positive class respectively.

Write the equivalent function in python for the following:

1. Compute the mean of attribute values against both class labels.

Python Code :

import pandas as pd

import numpy as np

data = [

    ['A',21,76,23],['B',80,70,65],['C',96,31,43],['A',29,25,11],

    ['A',38,10,45],['B',45,91,97],['C',66,44,60],['B',99,61,32]

]

names = ['Class', 'Attr1', 'Attr2', 'Attr3']

df=pd.DataFrame(data,columns=names)

print(df)

print('Attribute means:')

attr\_mean=df.iloc[:,1:].mean(axis=0)

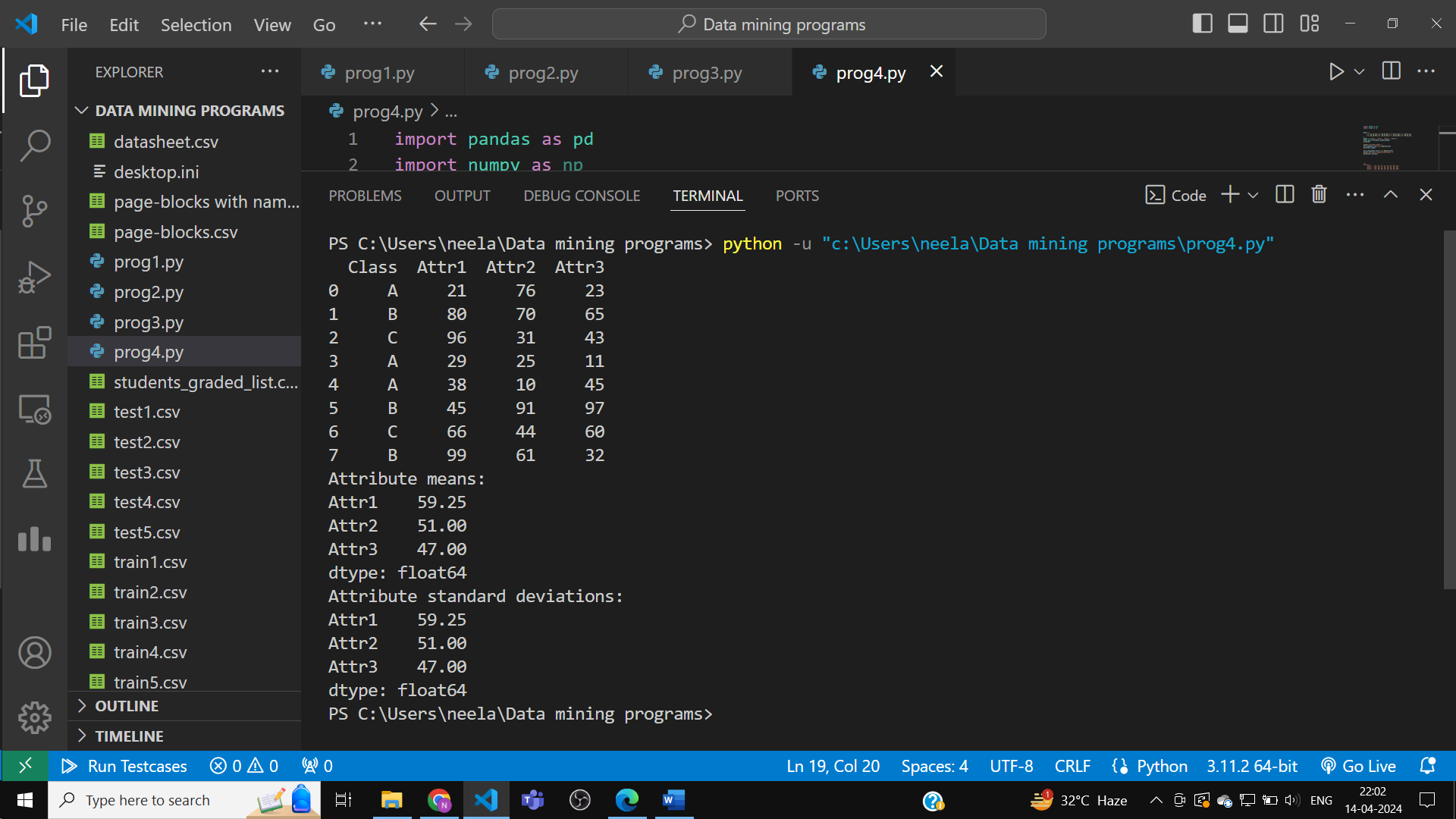
print(attr\_mean)

attr\_std\_dev=df.iloc[:,1:].mean(axis=0)

print('Attribute standard deviations:')

print(attr\_std\_dev)

Output:



2. Compute the standard deviation of attribute values against both class labels.

Python Code :

import pandas as pd

import numpy as np

data = [

    ['A',21,76,23],['B',80,70,65],['C',96,31,43],['A',29,25,11],

    ['A',38,10,45],['B',45,91,97],['C',66,44,60],['B',99,61,32]

]

names = ['Class', 'Attr1', 'Attr2', 'Attr3']

df=pd.DataFrame(data,columns=names)

print(df)

print('Attribute means:')

attr\_mean=df.iloc[:,1:].mean(axis=0)

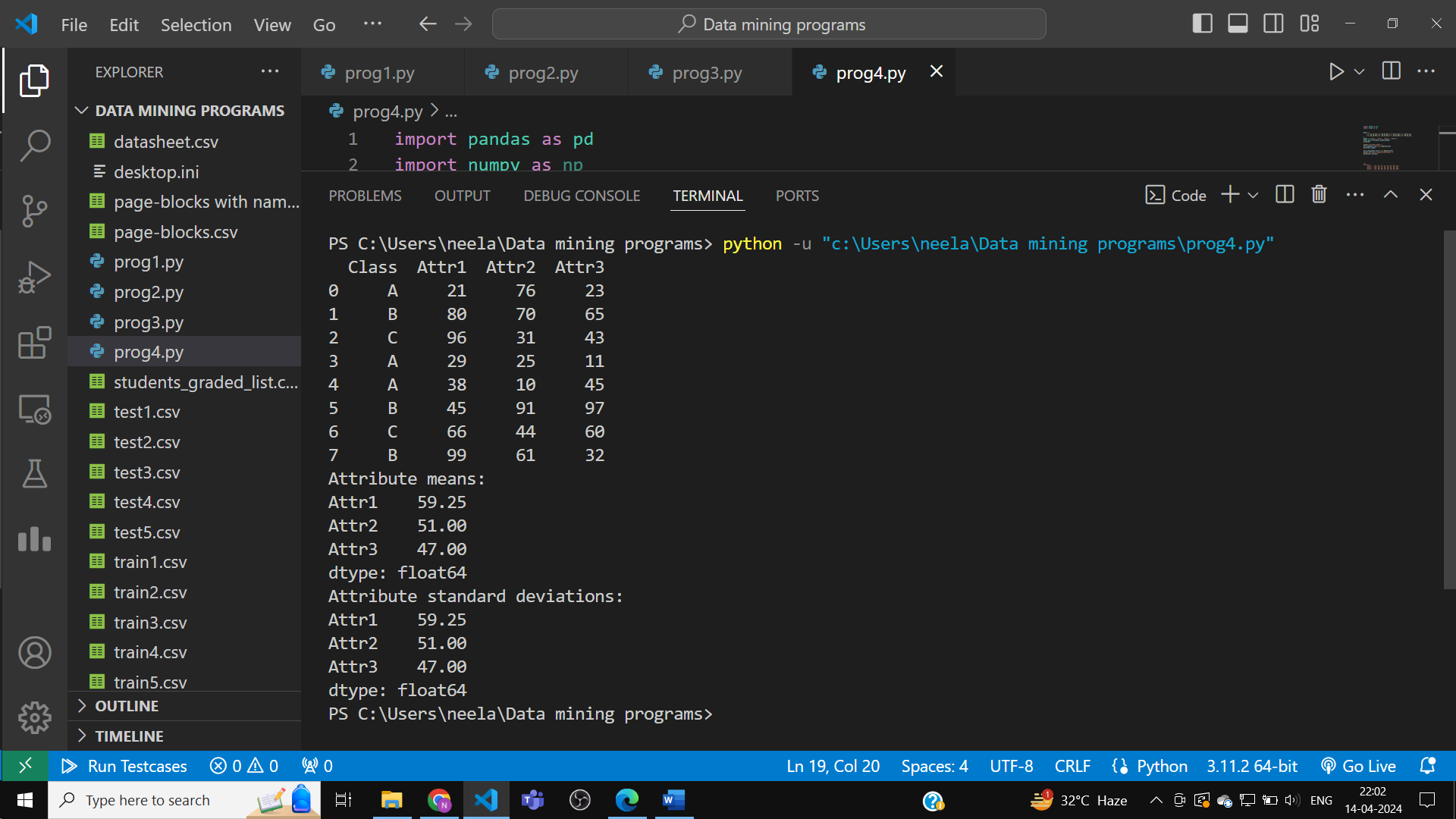
print(attr\_mean)

attr\_std\_dev=df.iloc[:,1:].mean(axis=0)

print('Attribute standard deviations:')

print(attr\_std\_dev)

Output:



3. ComputeFisher’s score (f) the for each attribute.

Python Code :

import pandas as pd

import numpy as np

data = {

    'Attr1' : [23, 65, 43, 11, 45, 97, 60, 32],

    'Attr2' : [76, 70, 31, 25, 10, 91, 44, 61],

    'Attr3' : [21, 80, 96, 29, 38, 45, 66, 99],

    'Class' : ['A', 'B', 'C', 'A', 'A', 'B', 'C', 'B']

}

df=pd.DataFrame(data)

print(df)

X,Y=df[['Attr1','Attr2','Attr3']],df['Class']

print(X)

print(Y)

classes\_mean=X.groupby(Y).mean()

classes\_variance=X.groupby(Y).var()

print('Class mean:')

print(classes\_mean)

print('Class variance:')

print(classes\_variance)

inter\_class\_variance=(classes\_mean.diff().dropna()\*\*2).sum()

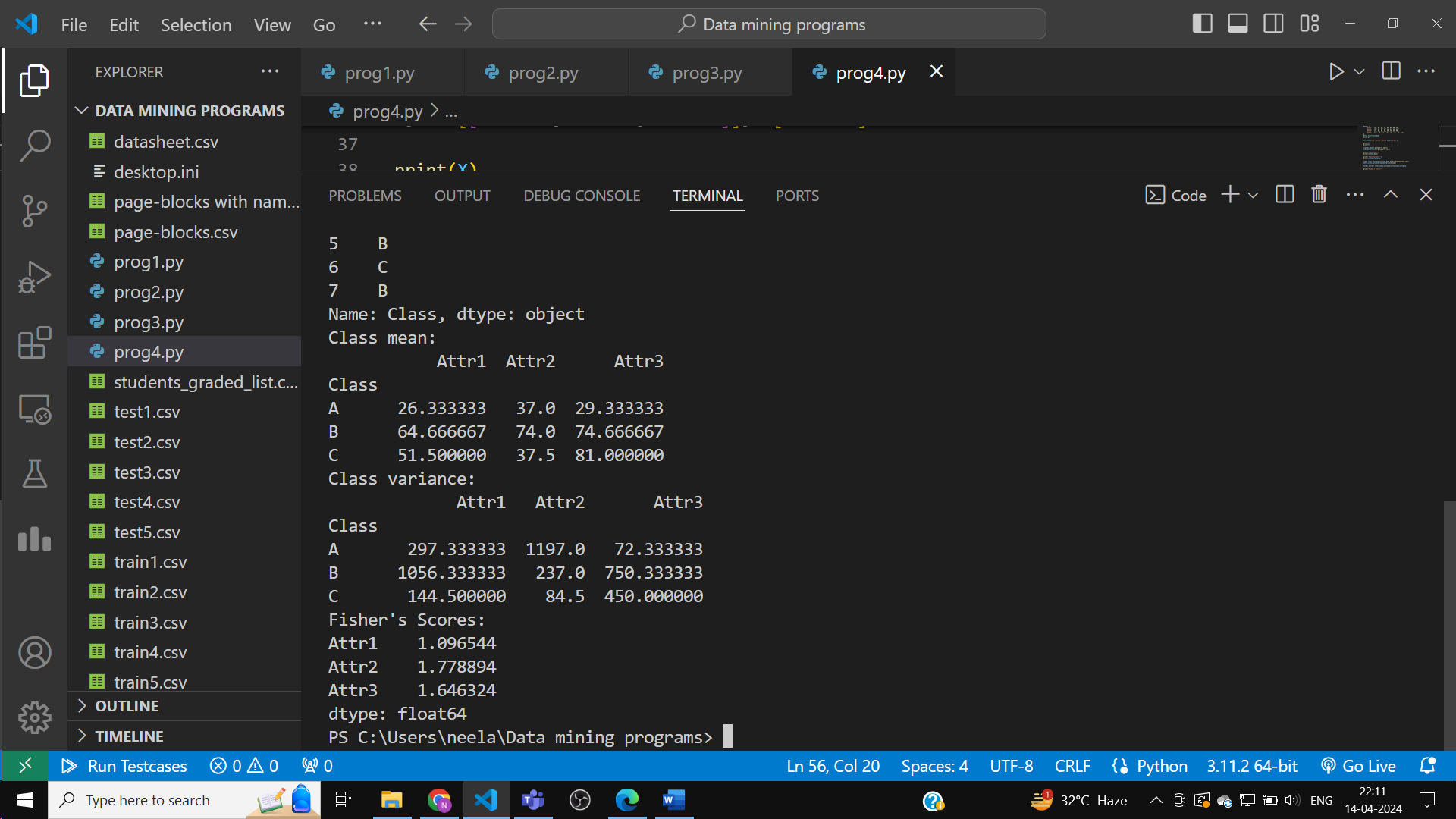
intra\_class\_variance=classes\_variance.sum()

fisher\_score = inter\_class\_variance/intra\_class\_variance

print("Fisher's Scores:")

print(fisher\_score)

Output:



4. Assigned rank for each attribute (High f value has a high ranking).

Python Code :

import pandas as pd

import numpy as np

data = {

    'Attr1' : [23, 65, 43, 11, 45, 97, 60, 32],

    'Attr2' : [76, 70, 31, 25, 10, 91, 44, 61],

    'Attr3' : [21, 80, 96, 29, 38, 45, 66, 99],

    'Class' : ['A', 'B', 'C', 'A', 'A', 'B', 'C', 'B']

}

df=pd.DataFrame(data)

print(df)

X,Y=df[['Attr1','Attr2','Attr3']],df['Class']

print(X)

print(Y)

classes\_mean=X.groupby(Y).mean()

classes\_variance=X.groupby(Y).var()

print('Class mean:')

print(classes\_mean)

print('Class variance:')

print(classes\_variance)

inter\_class\_variance=(classes\_mean.diff().dropna()\*\*2).sum()

intra\_class\_variance=classes\_variance.sum()

fisher\_score = inter\_class\_variance/intra\_class\_variance

print("Fisher's Scores:")

print(fisher\_score)

liss=[]

for i,j in fisher\_score.items():

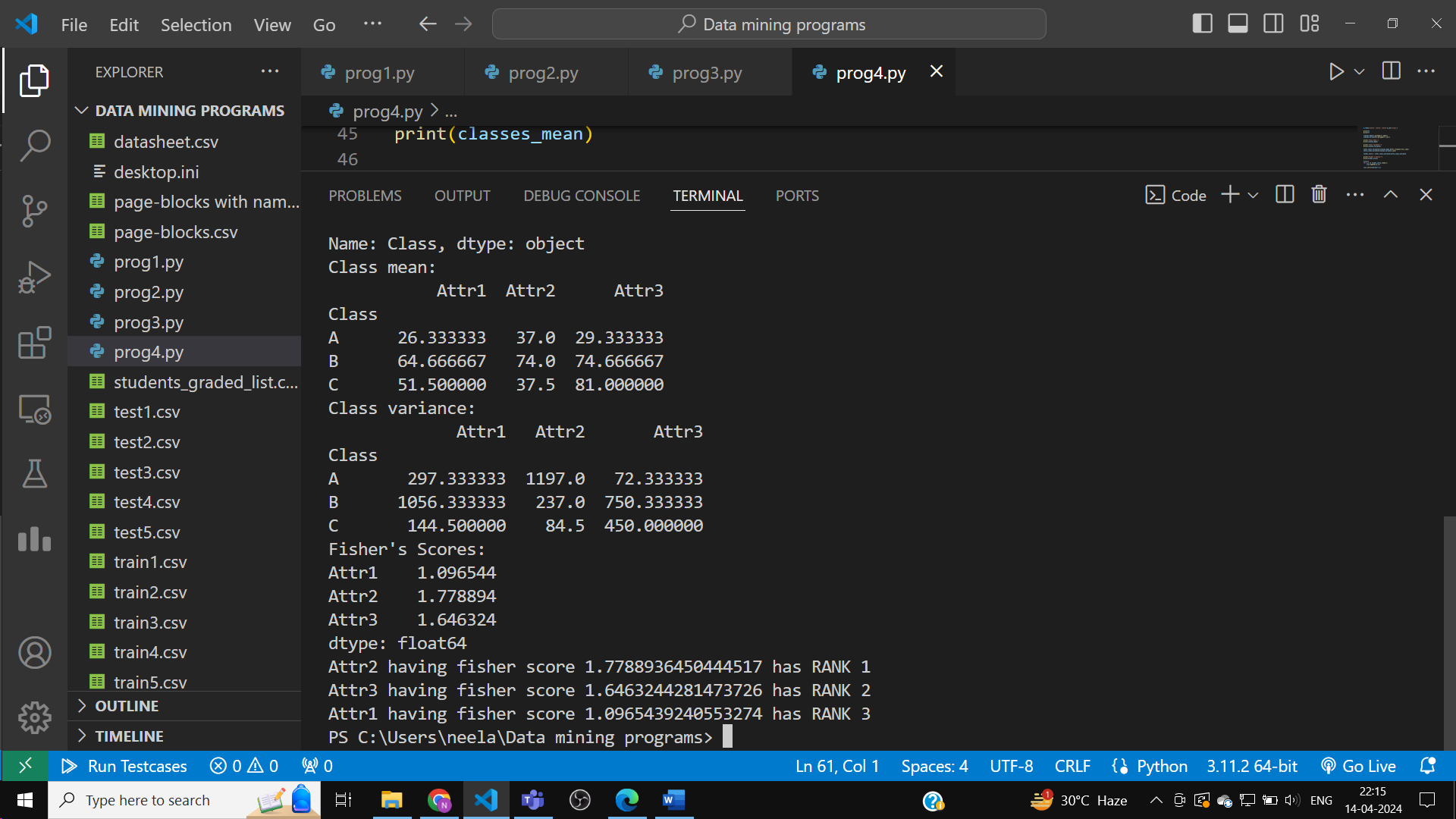
    liss.append([j,i])

liss.sort(reverse=True)

for i in range(len(liss)):

    print(str(liss[i][1])+' having fisher score '+str(liss[i][0])+' has RANK '+str(i+1))

Output:



Objective 2:

Load buys\_computer.csv nominal (Buy Computer (kaggle.com)) dataset (make appropriate preprocessing if required). Use the information gain formula.

Let dataset D with two class labels c1 and c2 then expected information (entropy) info (D) can be computed by the formula as given below:

Where pi is the probability of class ci and label is the number of classes(2 in the case of the binary class dataset).

The information gain of attribute A is given by the formula as follows:

Gain (A)= Info(D) – InfoA (D)

Write the equivalent function in python for the following:

1. Compute the entropy (info (dataset)) of the entire dataset.

2. Compute InfoA(Dataset) for each attribute A in the dataset.

3. Compute Information Gain (A) for each attribute A in the dataset.

4. Select k attributes with the highest information gain.

Python Code :

import math

import pandas as pd

import numpy as np

def calculateEntropy(i,j):

    prob1,prob2=i/(i+j),j/(i+j)

    return -prob1\*math.log2(prob1)-prob2\*math.log2(prob2)

data=[

    ['high','fair','Yes'],['medium','fair','No'],['medium','excellent','Yes'],

    ['low','fair','No'],['low','excellent','No'],['high','excellent','No'],

    ['low','fair','No'],['low','excellent','Yes'],['high','fair','No']

]

names=['Income','credit\_rating','Buy Computer']

df=pd.DataFrame(data,columns=names)

print(df)

positive=0

for i in data:

    positive += i[-1]=='Yes'

infoD = calculateEntropy(positive,len(data)-positive)

info\_A={}

for i in range(len(names)-1):

    tmp={}

    for j in range(len(data)):

        if data[j][i] not in tmp:

            tmp[data[j][i]]=[j]

        else:

            tmp[data[j][i]].append(j)

    calc=0

    for key,liss in tmp.items():

        cnt=0

        for idx in liss:

            cnt += data[idx][-1]=='Yes'

        calc += (len(liss)/len(data))\*calculateEntropy(cnt,len(liss)-cnt)

    info\_A[names[i]]=calc

gain\_A={}

for i in range(-1+len(names)):

    gain\_A[names[i]]=infoD-info\_A[names[i]]

# 1. Compute the entropy (info (dataset)) of the entire dataset.

print('Entropy of entire dataset: %.5f'%infoD)

# 2. Compute InfoA(Dataset) for each attribute A in the dataset.

print('\nInfo\_A for each attribute in the dataset:')

for key,val in info\_A.items():

    print('Attribute',key,": %.5f"%val)

# 3. Compute Information Gain (A) for each attribute A in the dataset.

print('\nInformation gain for each attribute in the dataset:')

for key,val in gain\_A.items():

    print('Attribute',key,": %.5f"%val)

liss=[]

for key,val in gain\_A.items():

    liss.append([val,key])

liss.sort(reverse=True)

# 4. Select k attributes with the highest information gain.

print('\nAttributes sorted in descending order by their information gain:')

for i in liss:

    print('Attribute',i[1],': %.5f'%i[0])

Output:

