**Data Mining Lab**

**Assignment 04**

**Group** – C

**B.Tech 6th Semester**

**Name** – Neelangshu Roy

**Reg. No.** – 20214060

Objective 1:

Working with the data dictionary, list, and DataFrame:

1. Prepare dataFrame with the following lists\_marks detail in different subjects:

Columns\_list =[‘Reg\_no’,’Name’,’Subject1’, ’Subject2’ ,’Subject3’ ,’Subject4’]

Rows\_list=

[

[2022001,’Abhijeet’,65,65,69,81], [2022002,’Ajeet’,75,75,90,81],

[2022003,’Amit’,75,05,69,87], [2022004,’Ranjeet’,55,65,79,91],

[2022005,’Santosh’,85,85,60,61], [2022006,’Satyam’,73,75,68,51],

[2022007,’Shivam’,85,85,50,40], [2022009,’Shyam’,75,65,69,81] ,

[2022010,’Yash’,85,75,89,61]

]

Python Code :

import numpy as np

import pandas as pd

attribute\_names=["Reg\_no","Name","Subject1", "Subject2" ,"Subject3" ,"Subject4"]

tuple\_data=[

[2022001,"Abhijeet",65,65,69,81], [2022002,"Ajeet",75,75,90,81],

[2022003,"Amit",75,5,69,87], [2022004,"Ranjeet",55,65,79,91],

[2022005,"Santosh",85,85,60,61], [2022006,"Satyam",73,75,68,51],

[2022007,"Shivam",85,85,50,40], [2022009,"Shyam",75,65,69,81] ,

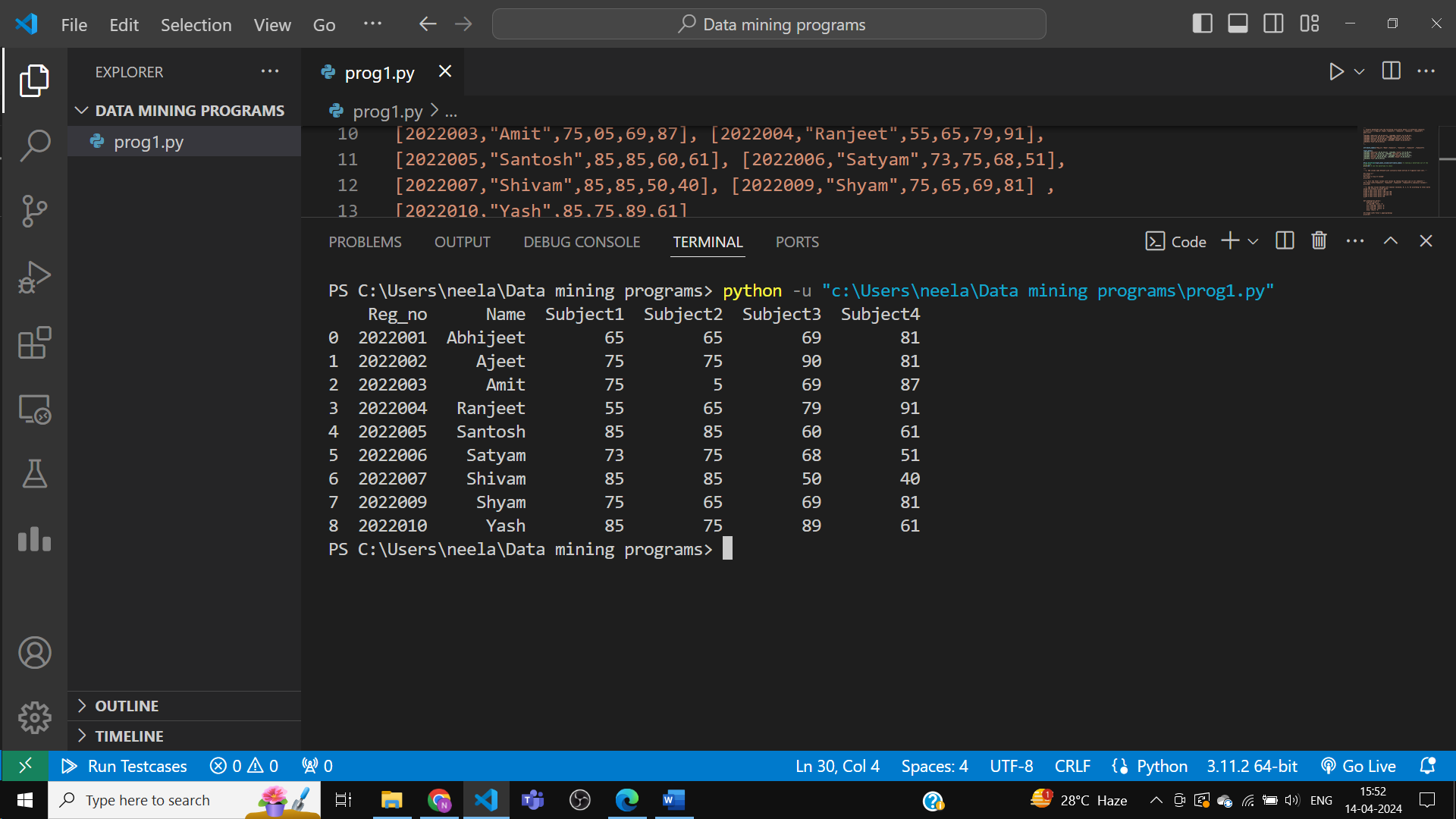
[2022010,"Yash",85,75,89,61]

]

df=pd.DataFrame(tuple\_data,columns=attribute\_names) # creating a dataframe out of the given data

print(df) # see the dataframe to check

Output:



2. Add column name ‘Total’ with initially blank entries ‘ ’ against each cell.

Python Code:

import numpy as np

import pandas as pd

attribute\_names=["Reg\_no","Name","Subject1", "Subject2" ,"Subject3" ,"Subject4"]

tuple\_data=[

[2022001,"Abhijeet",65,65,69,81], [2022002,"Ajeet",75,75,90,81],

[2022003,"Amit",75,5,69,87], [2022004,"Ranjeet",55,65,79,91],

[2022005,"Santosh",85,85,60,61], [2022006,"Satyam",73,75,68,51],

[2022007,"Shivam",85,85,50,40], [2022009,"Shyam",75,65,69,81] ,

[2022010,"Yash",85,75,89,61]

]

df=pd.DataFrame(tuple\_data,columns=attribute\_names) # creating a dataframe out of the given data

print(df) # see the dataframe to check

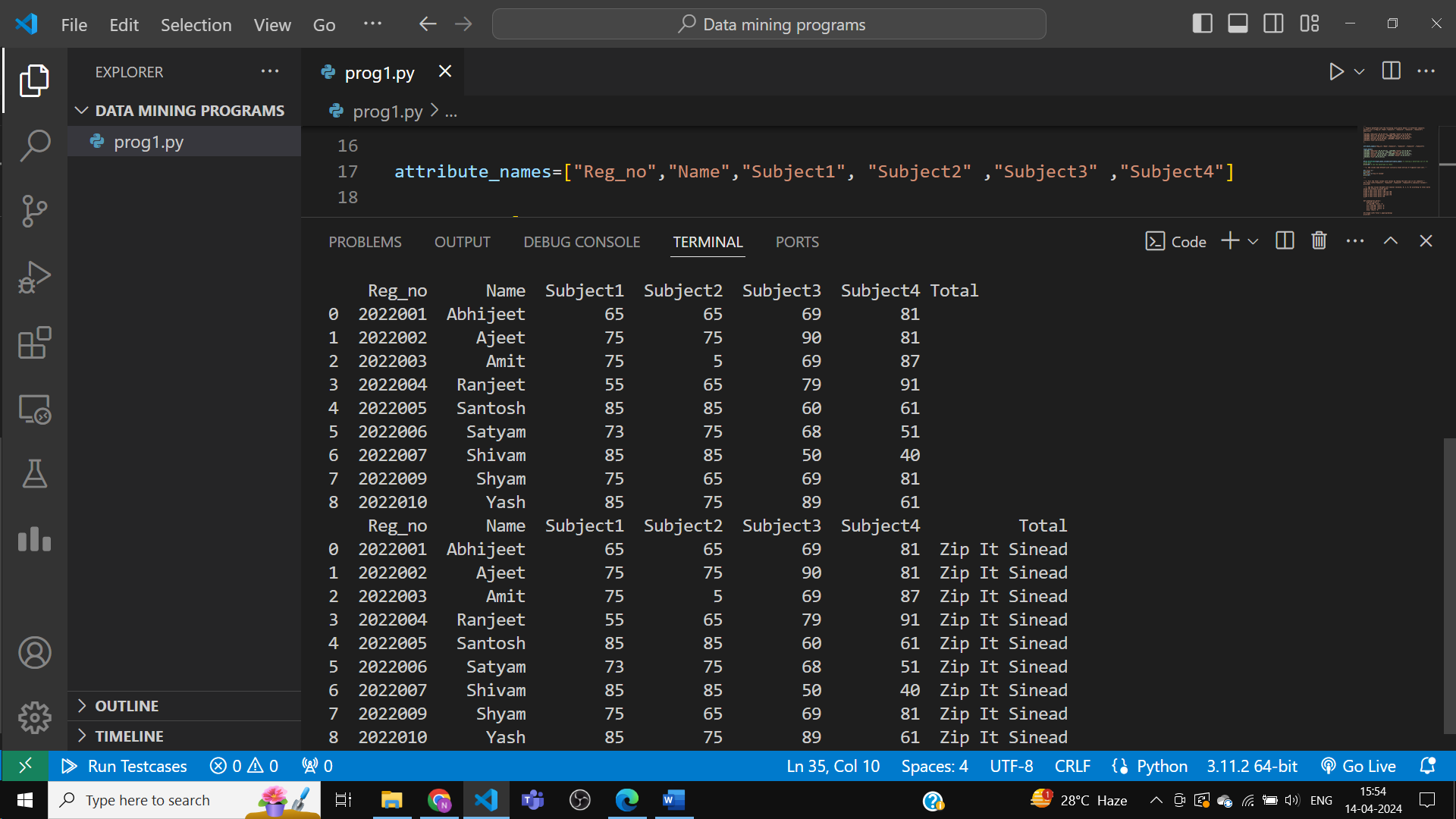
df['Total']=''

print(df)

df['Total']='Zip It Sinead'

print(df)

Output:



3. Fill the Total column with values by taking the mark sum in all subjects.

Python Code:

import numpy as np

import pandas as pd

attribute\_names=["Reg\_no","Name","Subject1", "Subject2" ,"Subject3" ,"Subject4"]

tuple\_data=[

[2022001,"Abhijeet",65,65,69,81], [2022002,"Ajeet",75,75,90,81],

[2022003,"Amit",75,5,69,87], [2022004,"Ranjeet",55,65,79,91],

[2022005,"Santosh",85,85,60,61], [2022006,"Satyam",73,75,68,51],

[2022007,"Shivam",85,85,50,40], [2022009,"Shyam",75,65,69,81] ,

[2022010,"Yash",85,75,89,61]

]

df=pd.DataFrame(tuple\_data,columns=attribute\_names) # creating a dataframe out of the given data

print(df) # see the dataframe to check

df['Total']=''

print(df)

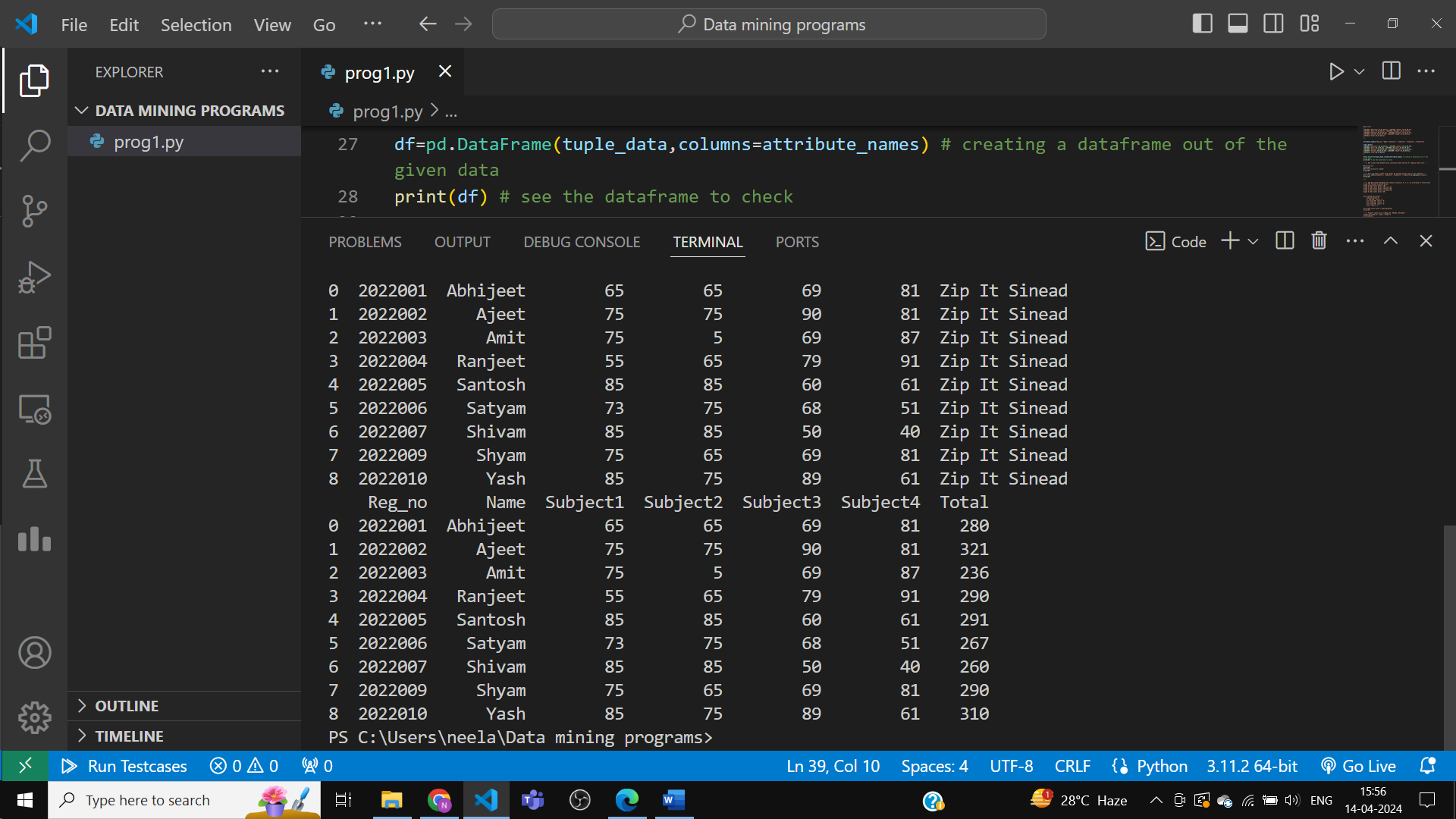
df['Total']='Zip It Sinead'

print(df)

df['Total']=df[["Subject1", "Subject2" ,"Subject3" ,"Subject4"]].sum(axis='columns')

print(df)

Output:



4. Add New Column ‘Grade’ with nominal values {A, B, C, D, E} according to total marks using

the formula as given below:

Grade A when total marks >=90

Grade B when total marks >=80 and <90

Grade C when total marks >=70 and <80

Grade D when total marks >=50 and <70

Grade E when total marks <50

Python Code:

import numpy as np

import pandas as pd

attribute\_names=["Reg\_no","Name","Subject1", "Subject2" ,"Subject3" ,"Subject4"]

tuple\_data=[

[2022001,"Abhijeet",65,65,69,81], [2022002,"Ajeet",75,75,90,81],

[2022003,"Amit",75,5,69,87], [2022004,"Ranjeet",55,65,79,91],

[2022005,"Santosh",85,85,60,61], [2022006,"Satyam",73,75,68,51],

[2022007,"Shivam",85,85,50,40], [2022009,"Shyam",75,65,69,81] ,

[2022010,"Yash",85,75,89,61]

]

df=pd.DataFrame(tuple\_data,columns=attribute\_names) # creating a dataframe out of the given data

df['Total']=df[["Subject1", "Subject2" ,"Subject3" ,"Subject4"]].sum(axis='columns')

print(df)

def grading(tot\_marks):

    avg=tot\_marks//4

    if avg>=90: return 'A'

    elif avg>=80: return 'B'

    elif avg>=70: return 'C'

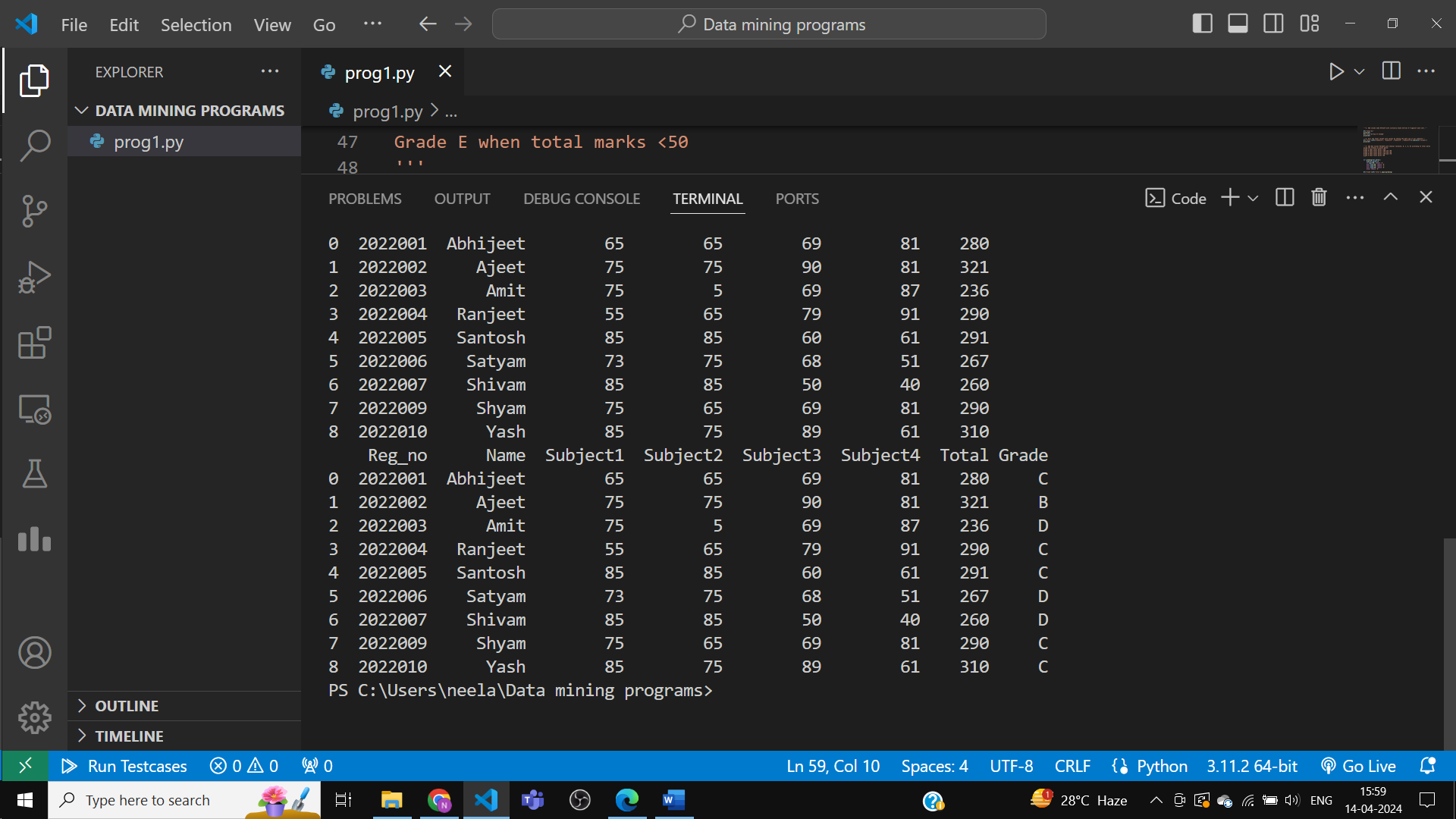
    elif avg>=50: return 'D'

    else: return 'E'

df['Grade']=df['Total'].apply(grading)

print(df)

Output:



5. Prepare subset with [[‘Reg\_no’, ’Name’, ’Grade’]

Python Code:

import numpy as np

import pandas as pd

attribute\_names=["Reg\_no","Name","Subject1", "Subject2" ,"Subject3" ,"Subject4"]

tuple\_data=[

[2022001,"Abhijeet",65,65,69,81], [2022002,"Ajeet",75,75,90,81],

[2022003,"Amit",75,5,69,87], [2022004,"Ranjeet",55,65,79,91],

[2022005,"Santosh",85,85,60,61], [2022006,"Satyam",73,75,68,51],

[2022007,"Shivam",85,85,50,40], [2022009,"Shyam",75,65,69,81] ,

[2022010,"Yash",85,75,89,61]

]

df=pd.DataFrame(tuple\_data,columns=attribute\_names) # creating a dataframe out of the given data

df['Total']=df[["Subject1", "Subject2" ,"Subject3" ,"Subject4"]].sum(axis='columns')

print(df)

def grading(tot\_marks):

    avg=tot\_marks//4

    if avg>=90: return 'A'

    elif avg>=80: return 'B'

    elif avg>=70: return 'C'

    elif avg>=50: return 'D'

    else: return 'E'

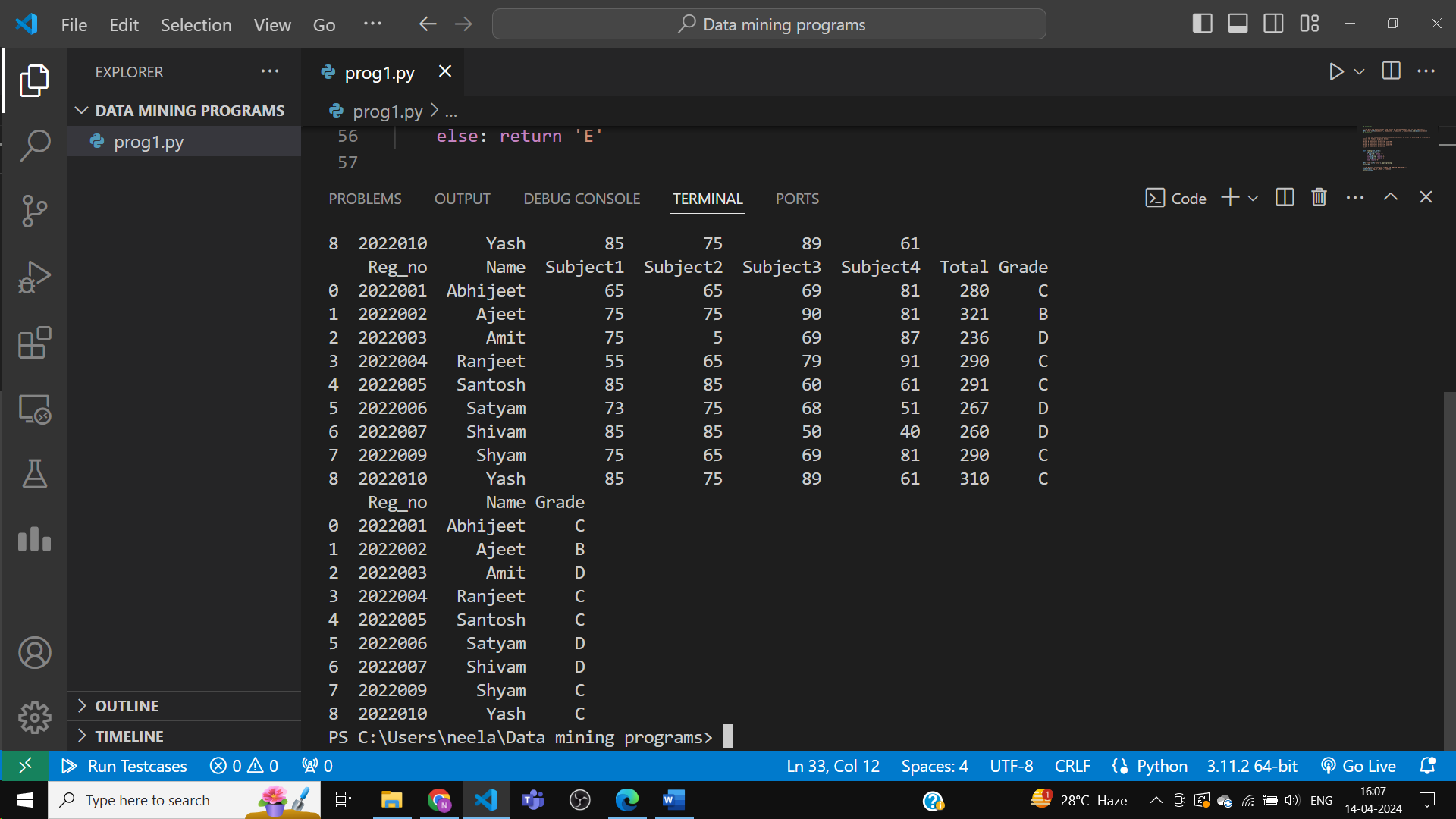
df['Grade']=df['Total'].apply(grading)

print(df)

subset=df[['Reg\_no','Name','Grade']]

print(subset)

Output:



6. Prepare a list of students according to grades in the separate data file.

Python Code:

import numpy as np

import pandas as pd

attribute\_names=["Reg\_no","Name","Subject1", "Subject2" ,"Subject3" ,"Subject4"]

tuple\_data=[

[2022001,"Abhijeet",65,65,69,81], [2022002,"Ajeet",75,75,90,81],

[2022003,"Amit",75,5,69,87], [2022004,"Ranjeet",55,65,79,91],

[2022005,"Santosh",85,85,60,61], [2022006,"Satyam",73,75,68,51],

[2022007,"Shivam",85,85,50,40], [2022009,"Shyam",75,65,69,81] ,

[2022010,"Yash",85,75,89,61]

]

df=pd.DataFrame(tuple\_data,columns=attribute\_names) # creating a dataframe out of the given data

df['Total']=df[["Subject1", "Subject2" ,"Subject3" ,"Subject4"]].sum(axis='columns')

print(df)

def grading(tot\_marks):

    avg=tot\_marks//4

    if avg>=90: return 'A'

    elif avg>=80: return 'B'

    elif avg>=70: return 'C'

    elif avg>=50: return 'D'

    else: return 'E'

df['Grade']=df['Total'].apply(grading)

newListByGrade=df.copy(deep=True)

newListByGrade=newListByGrade.sort\_values(by='Grade',ascending=True)

print('Original data:')

print(df)

print('Data sorted by grades:')

print(newListByGrade)

newListByGrade=newListByGrade.reset\_index(drop=True)

print('Sorted data and indices reset:')

print(newListByGrade)

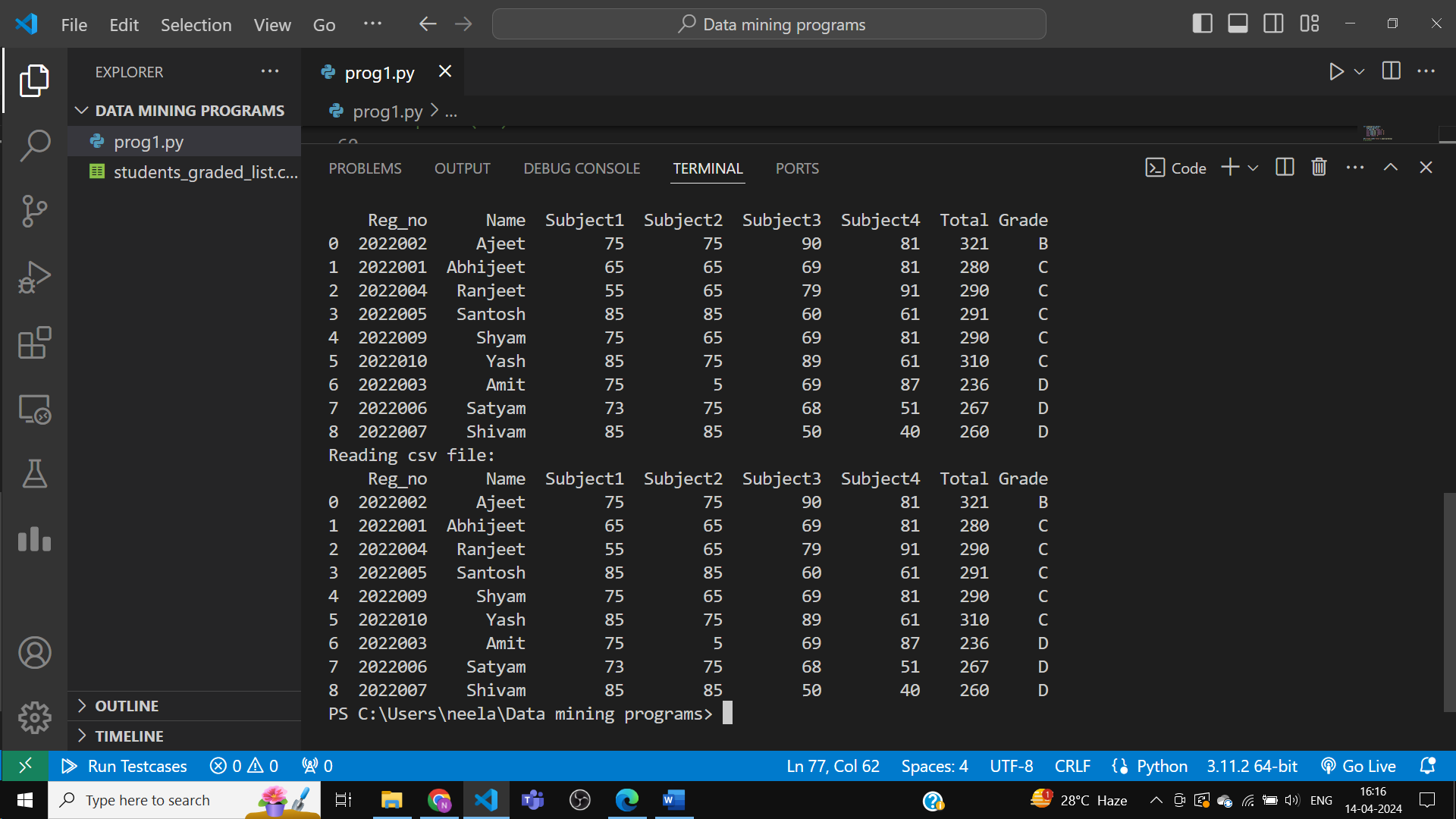
newListByGrade.to\_csv('students\_graded\_list.csv' ,index=False)

dff=pd.read\_csv('students\_graded\_list.csv')

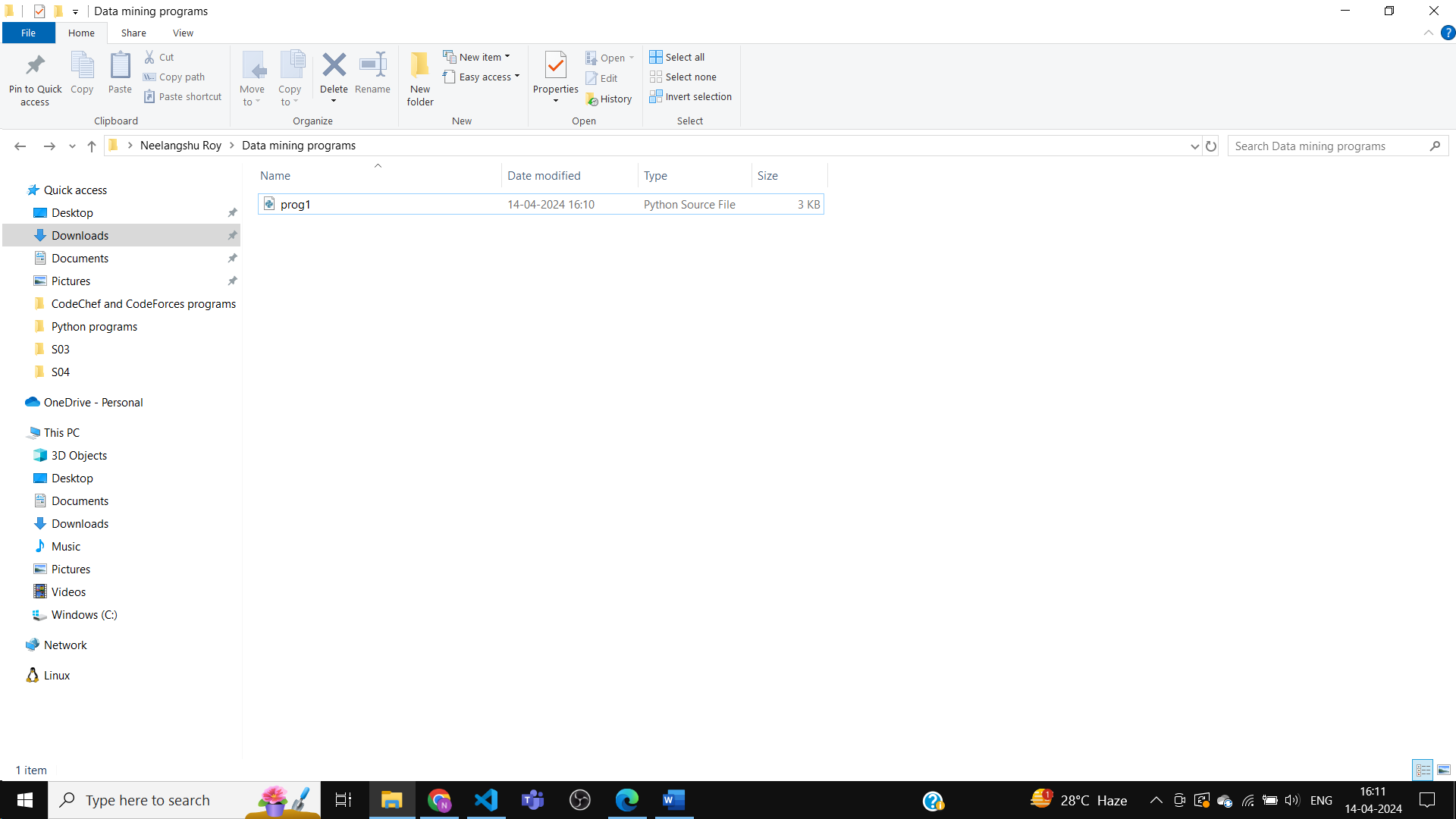
print('Reading csv file:')

print(dff)

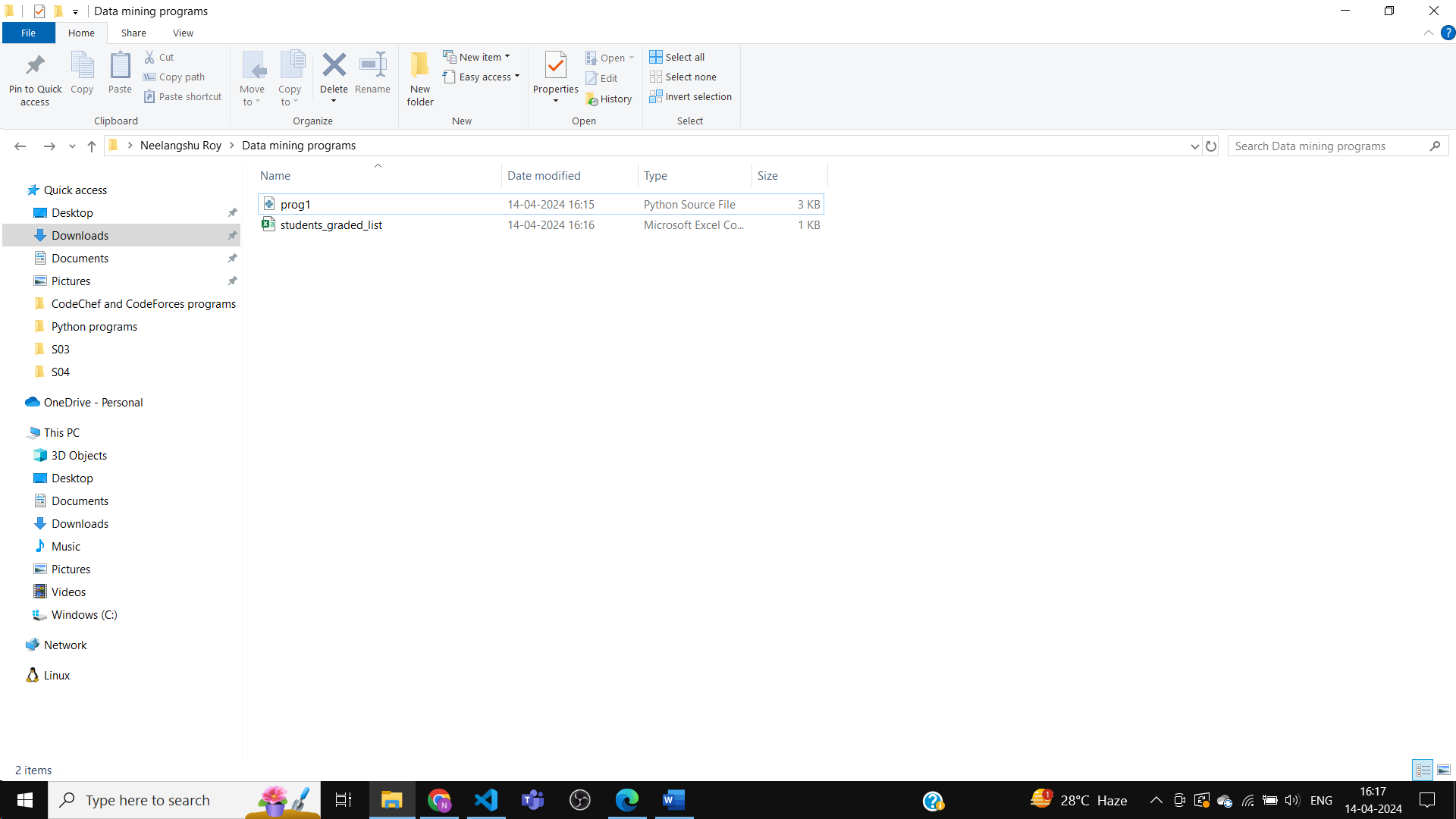
Output:



Before:



After:



Objective 2:

Working with Pandas CSV reading\writing and preparing training\testing dataset:

1. Read weatherNumeric.csv file and assigned it to object df.

Python Code :

import pandas as pd

import numpy as np

from scipy.io import arff

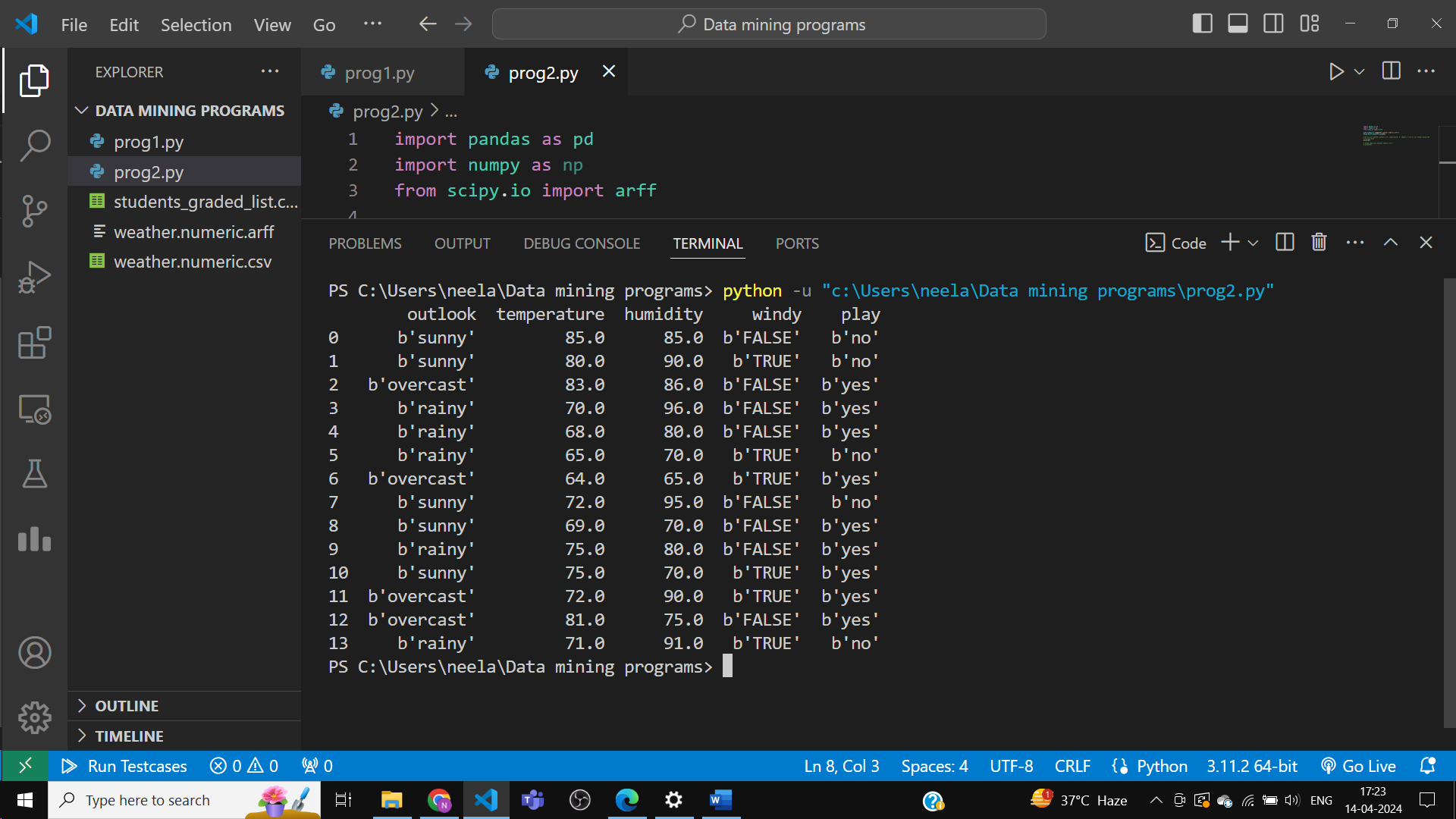
arff\_file=arff.loadarff('weather.numeric.arff')

df=pd.DataFrame(arff\_file[0])

df.to\_csv('weather.numeric.csv',index=False) # creates a file in csv format using the arff file data

print(df)

Output:



2. Select the last column as a class and assign it to object Y

Python Code :

import pandas as pd

import numpy as np

from scipy.io import arff

arff\_file=arff.loadarff('weather.numeric.arff')

df=pd.DataFrame(arff\_file[0])

# df.to\_csv('weather.numeric.csv',index=False) # creates a file in csv format using the arff file data

print(df)

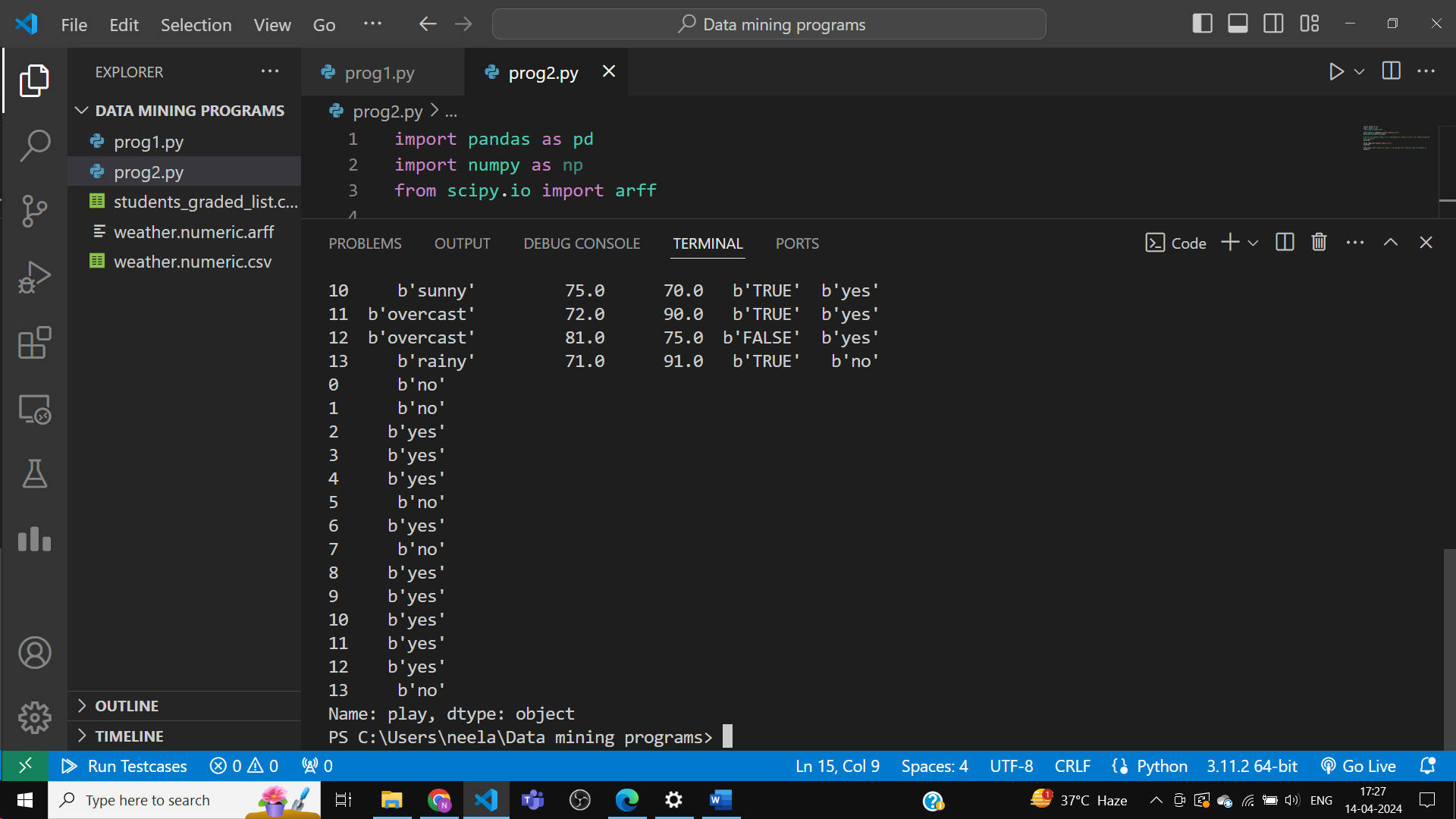
df=pd.read\_csv('weather.numeric.csv')

print(df)

Y=df.iloc[:,-1] # select all rows(:), and assign last column as class attribute(-1)

print(Y)

Output :



3. Select all remaining columns other than the last and assigned them to object X.

Python Code:

import pandas as pd

import numpy as np

from scipy.io import arff

arff\_file=arff.loadarff('weather.numeric.arff')

df=pd.DataFrame(arff\_file[0])

# df.to\_csv('weather.numeric.csv',index=False) # creates a file in csv format using the arff file data

print(df)

df=pd.read\_csv('weather.numeric.csv')

print(df)

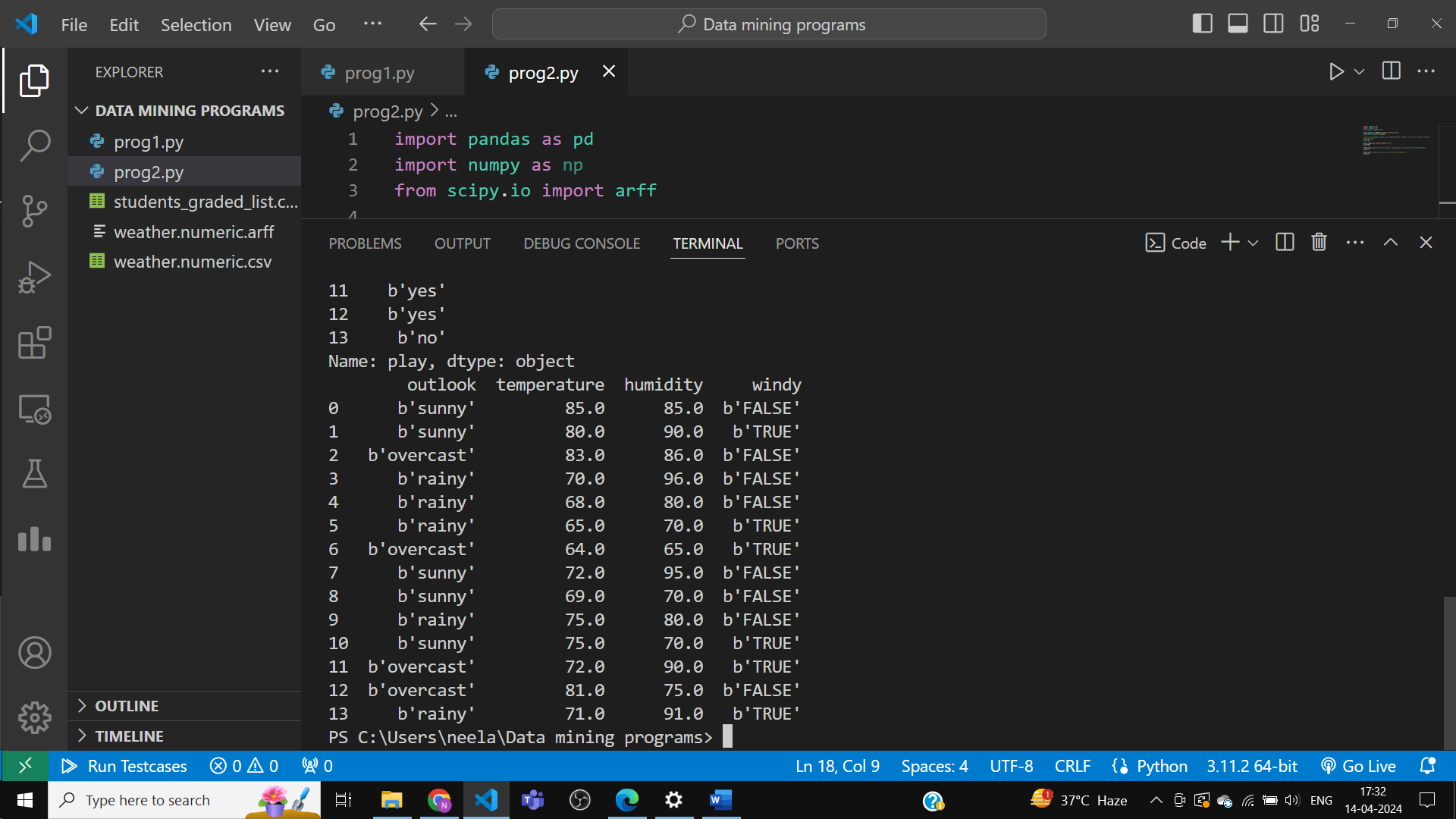
Y=df.iloc[:,-1] # select all rows(:), and assign last column as class attribute(-1)

print(Y)

X=df.iloc[:,:-1] # all rows(:), and omit last column(:-1)

print(X)

Output:



4. Split entire both X and Y into training: 80%, testing:20% parts and assigned it into X\_train, X\_test, Y\_train, and Y\_test respectively.

Python Code :

import pandas as pd

import numpy as np

import random as rand

from sklearn.model\_selection import train\_test\_split

df=pd.read\_csv('weather.numeric.csv')

print(df)

Y=df.iloc[:,-1] # select all rows(:), and assign last column as class attribute(-1)

print(Y)

X=df.iloc[:,:-1] # all rows(:), and omit last column(:-1)

print(X)

X\_train,X\_test,Y\_train,Y\_test=train\_test\_split(X,Y,test\_size=0.2,random\_state=rand.randint(41,47))

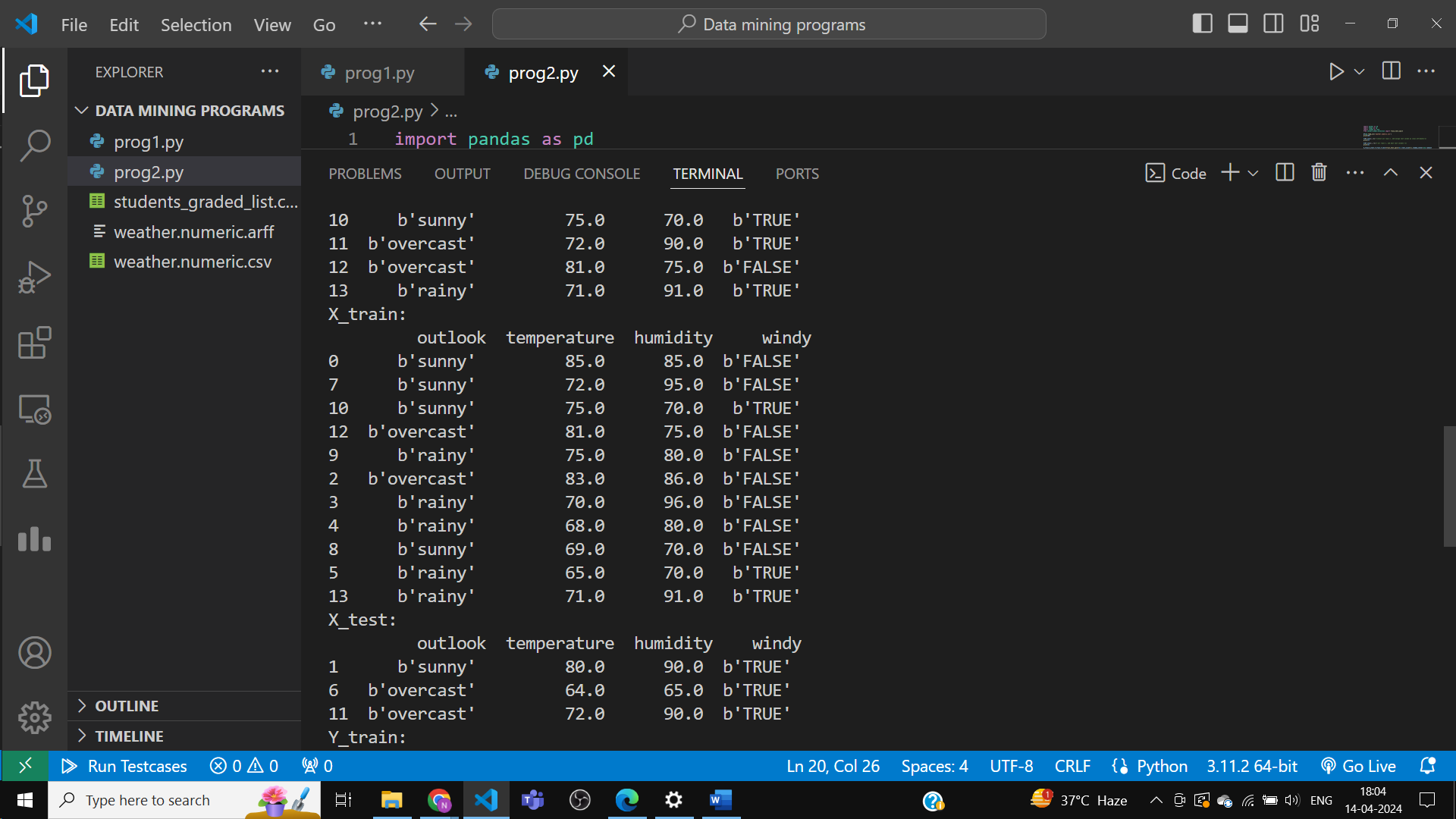
print('X\_train:\n',X\_train)

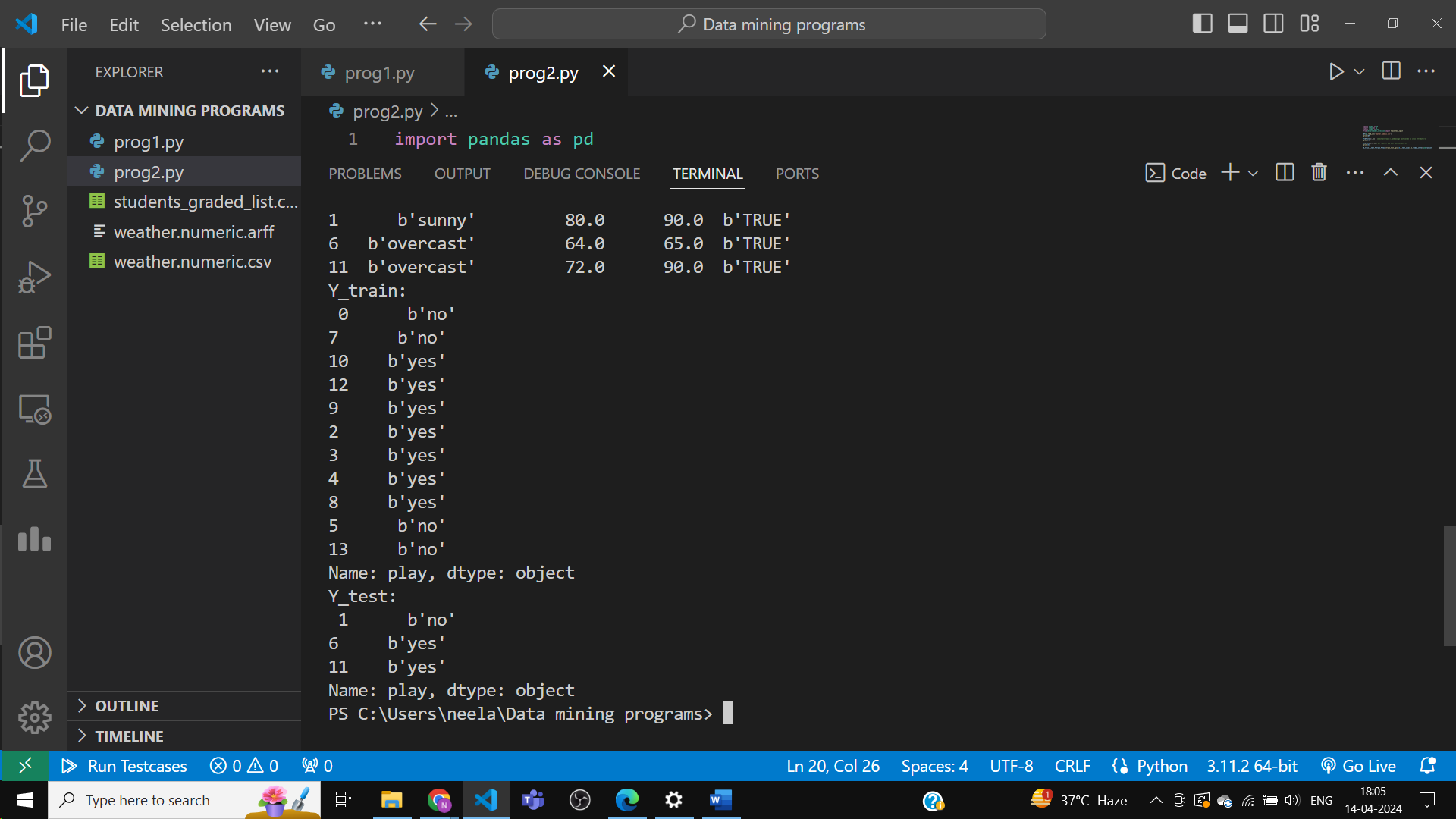
print('X\_test:\n',X\_test)

print('Y\_train:\n',Y\_train)

print('Y\_test:\n',Y\_test)

Output:





5. Prepare five different training\testing pairs and use pandas to\_csv() to save these into file

names: train1,test1, train2,test2, train3,test3, train4,test4, train5,test5.

Python Code :

import pandas as pd

import numpy as np

import random as rand

from sklearn.model\_selection import train\_test\_split

df=pd.read\_csv('weather.numeric.csv')

print(df)

X=df.copy(deep=True)

for i in range(5):

    X\_train,X\_test=train\_test\_split(X,test\_size=0.2,random\_state=rand.randint(41,47))

    X\_train=X\_train.reset\_index(drop=True)

    X\_test=X\_test.reset\_index(drop=True)

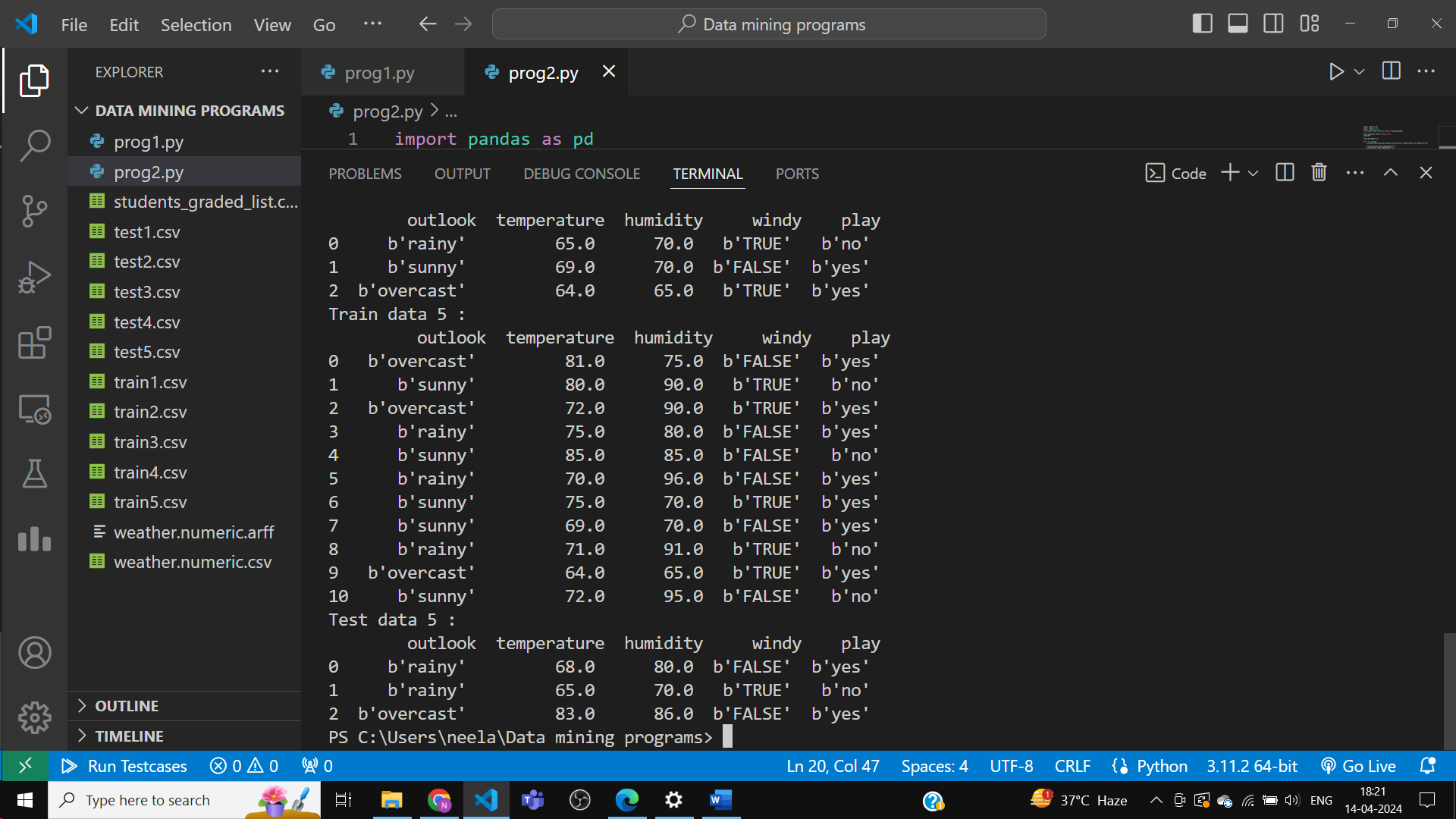
    X\_train.to\_csv('train'+str(i+1)+'.csv',index=False)

    X\_test.to\_csv('test'+str(i+1)+'.csv',index=False)

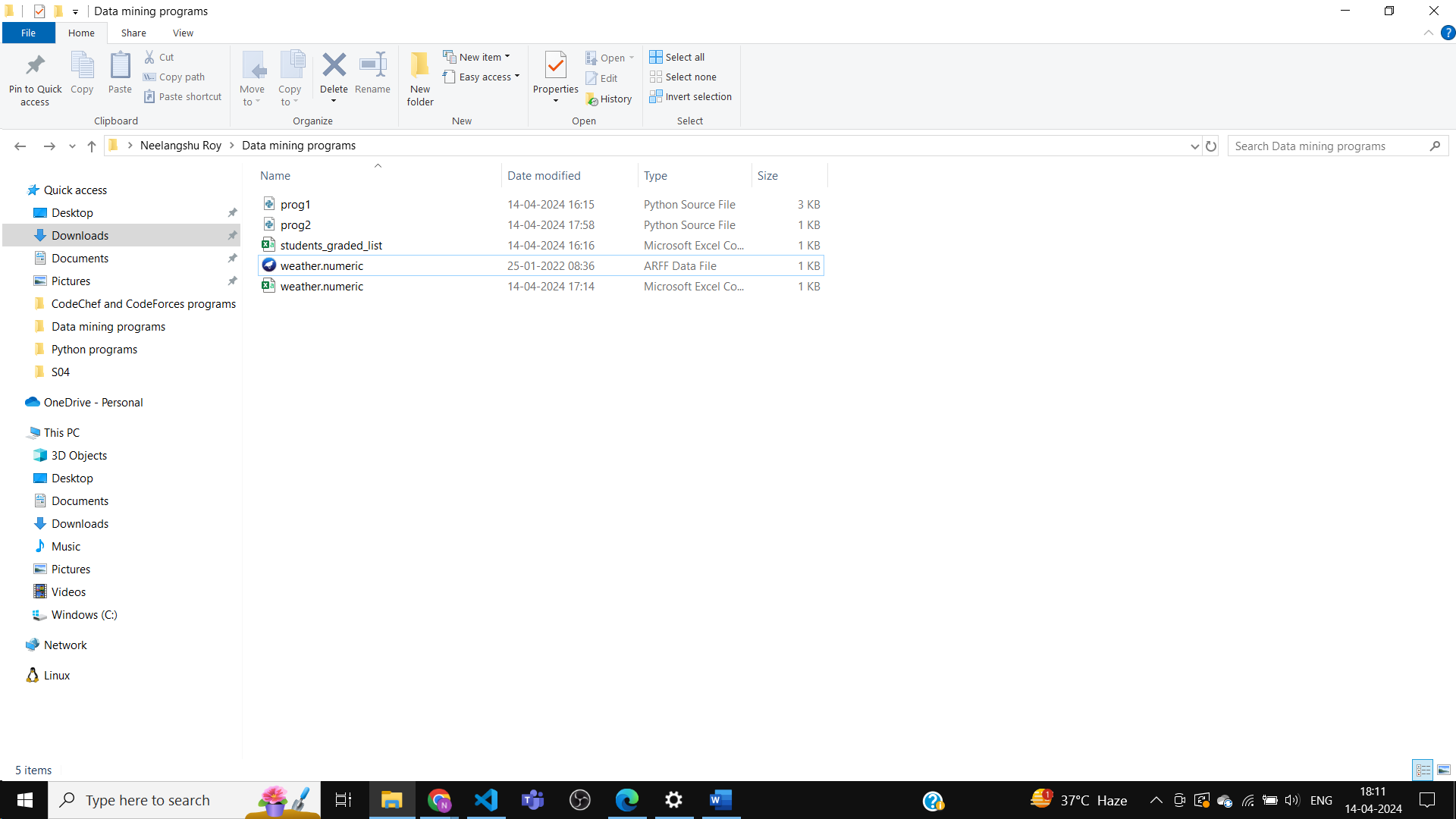
    print('Train data',str(i+1),':\n',X\_train)

    print('Test data',str(i+1),':\n',X\_test)

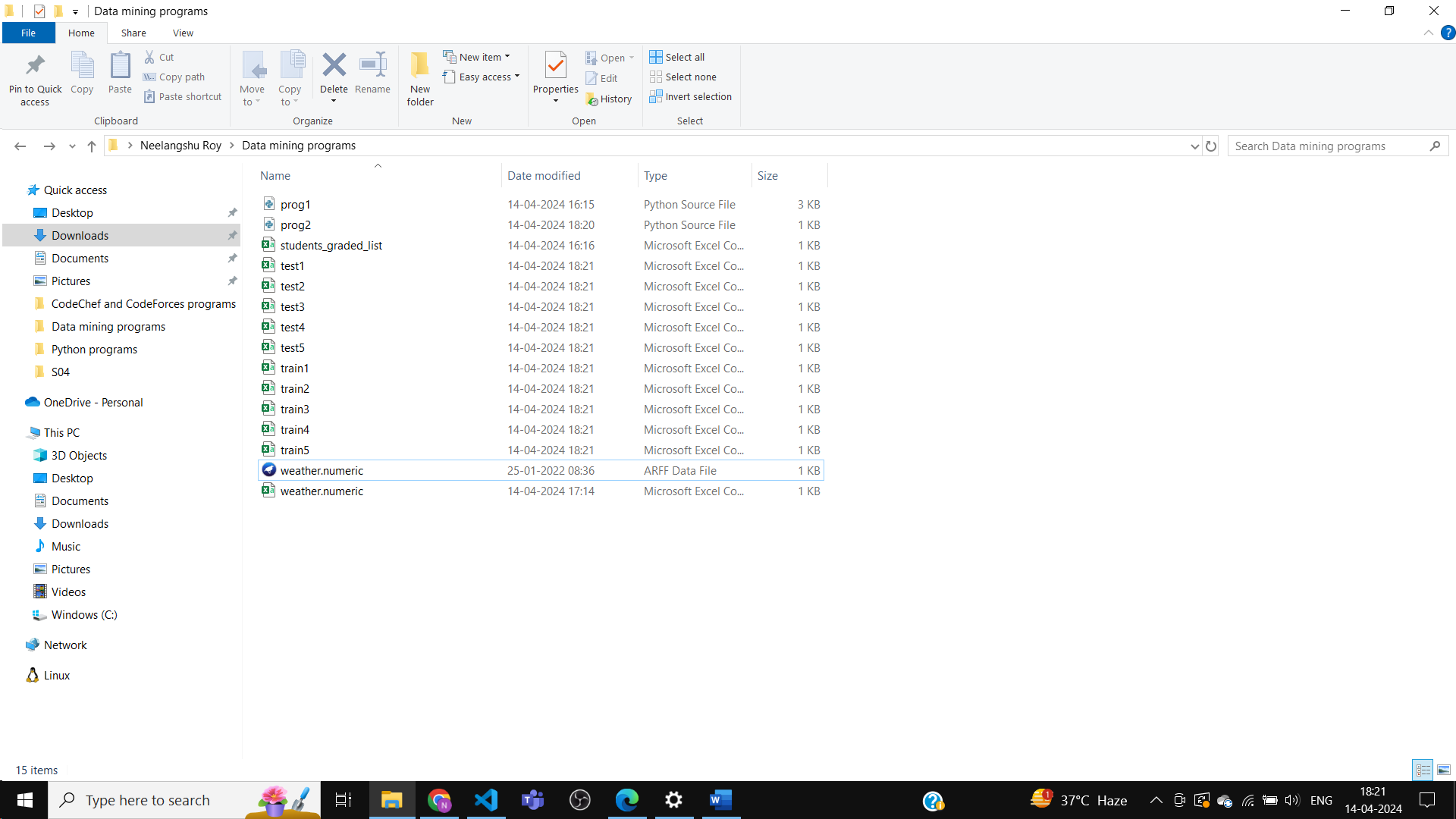
Output:



Before :



After :



Objective 3:

Analysis with result dataset:

The datasheet.csv file contains sensitivity score results generated by Random Forest Tree (RFT) classifiers on 24 equivalent re-samples of a dataset by 18 different resampling methods. The first column of datasheet.csv represents the dataset and all other remaining columns represent sensitivity value by RFT on different resampling methods. Read datasheet.csv file from the directory and do the following analysis.

First, create the csv file.

Python code :

import pandas as pd

import numpy as np

glass\_data = {

    'Dataset': ['Pima', 'Glass', 'Wisconsin'],

    'ENN': [0.9552, 0.9773, 0.7864],

    'AllKNN': [0.9452, 0.9773, 0.7864],

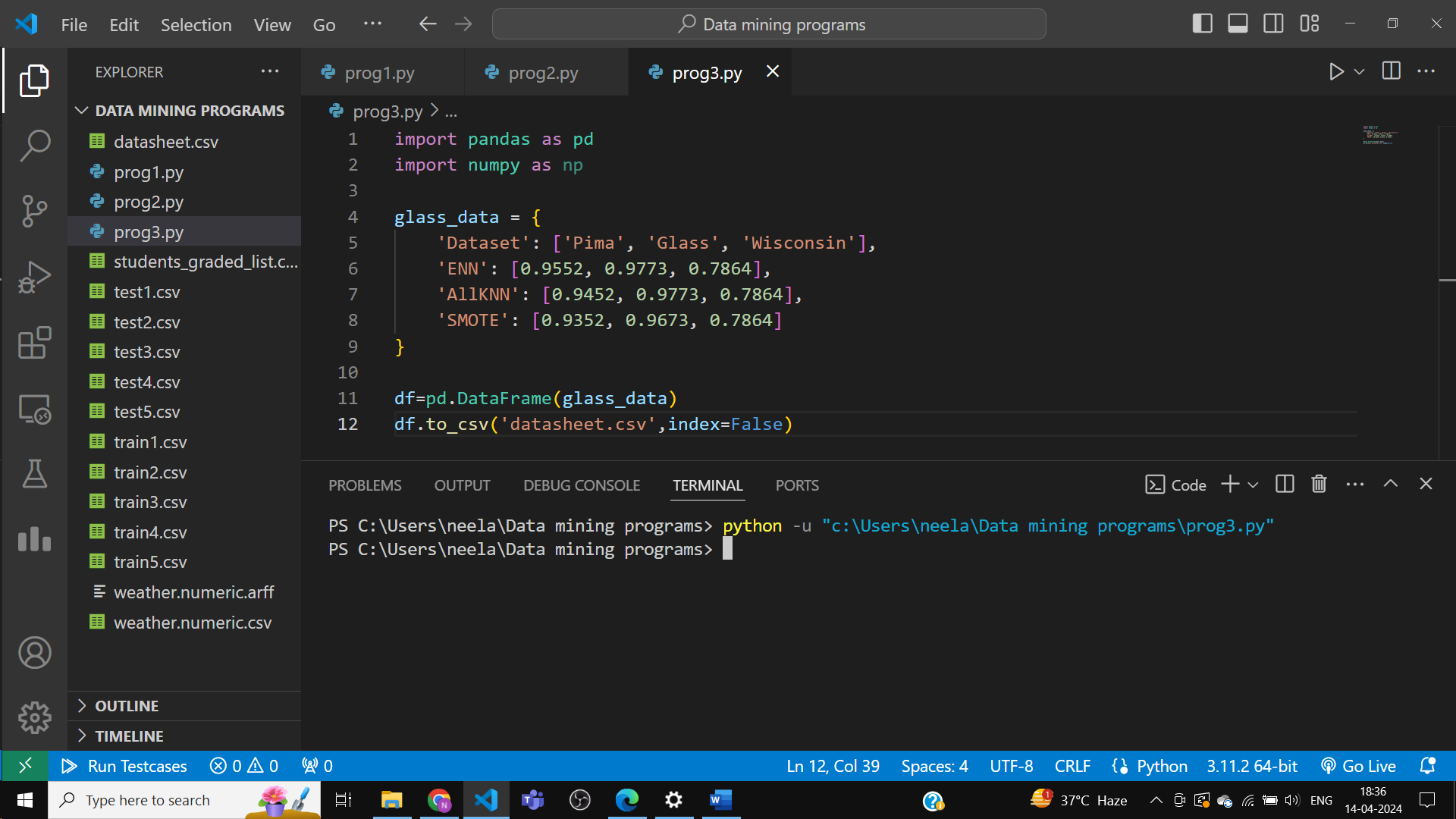
    'SMOTE': [0.9352, 0.9673, 0.7864]

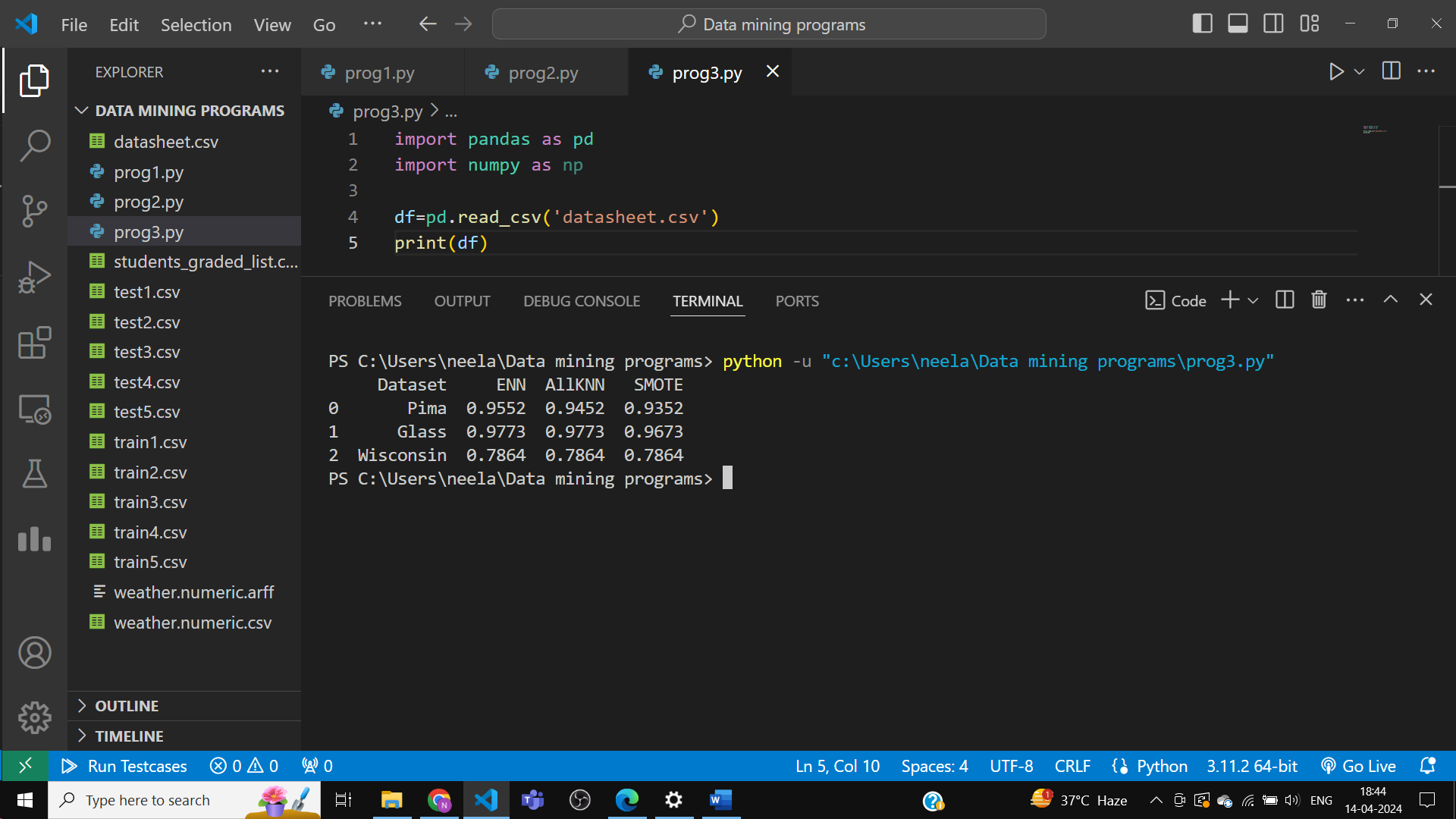
}

df=pd.DataFrame(glass\_data)

df.to\_csv('datasheet.csv',index=False)

Output:





1. Assigned rank to each resampling method, corresponding to each sensitivity score on each dataset row by using the following ranking strategy.

a. Rank(1:Higher sensitivity score and so on)

b. Assign the same rank for the two or more similar sensitivity scores

c. Rank range(1 to 18 in case all sensitivity values in a row are distinct)

Python Code :

import pandas as pd

import numpy as np

df=pd.read\_csv('datasheet.csv')

print(df)

def ranking(rows):

    sorted\_rows=rows.sort\_values(ascending=False)

    print(sorted\_rows)

    ranked\_dict={}

    rank=0

    prev=None

    for method,score in sorted\_rows.items():

        if score!=prev:

            rank+=1

            prev=score

        ranked\_dict[method]=rank

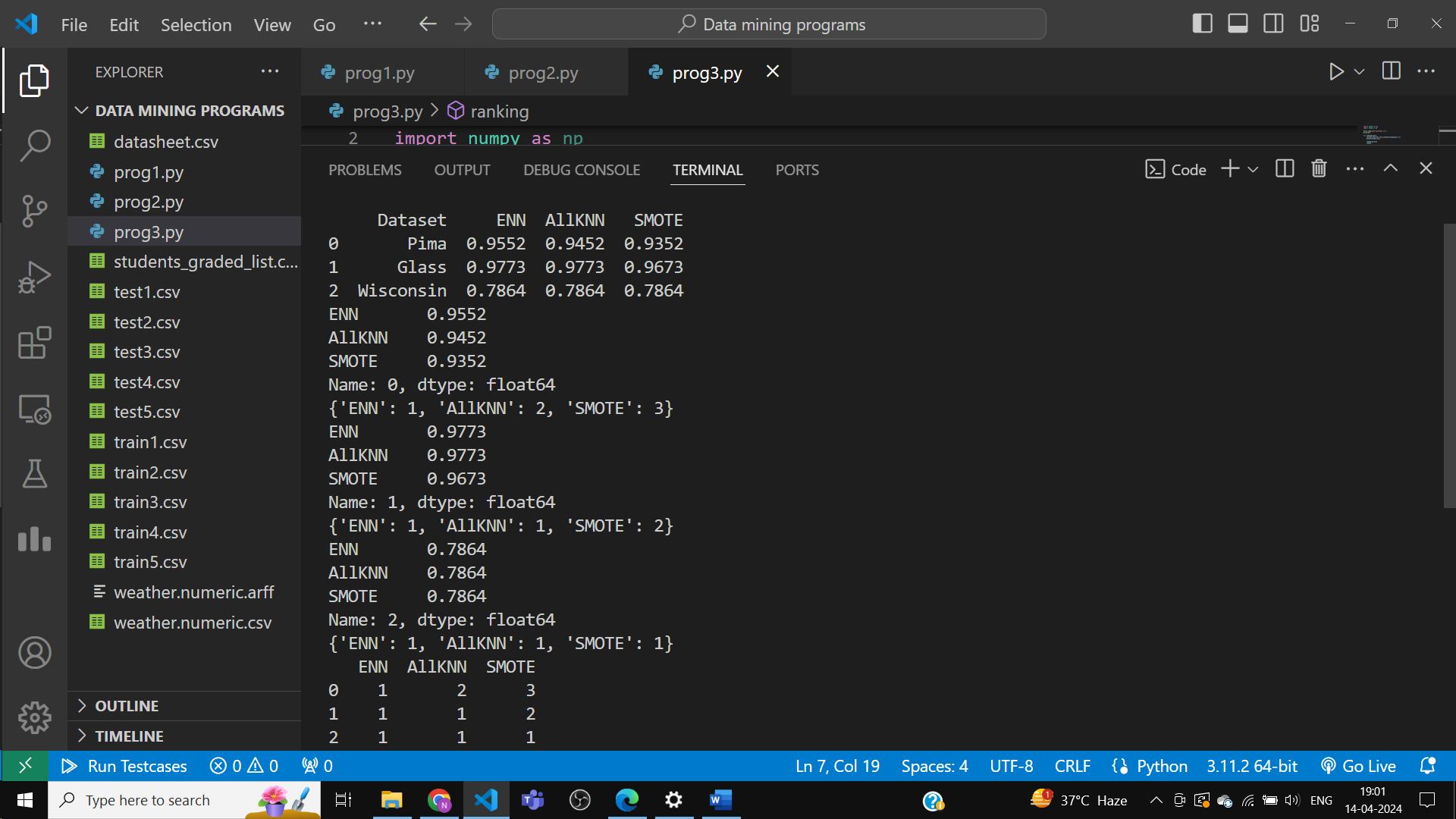
    print(ranked\_dict)

    return pd.Series(ranked\_dict)

ranked\_df=df.drop('Dataset',axis=1).apply(ranking,axis=1)

print(ranked\_df)

Output:



2. Compute the average sensitivity rank of each resampling method on the result against all datasets.

Python Code :

import pandas as pd

import numpy as np

df=pd.read\_csv('datasheet.csv')

# print(df)

def ranking(rows):

    sorted\_rows=rows.sort\_values(ascending=False)

    # print(sorted\_rows)

    ranked\_dict={}

    rank=0

    prev=None

    for method,score in sorted\_rows.items():

        if score!=prev:

            rank+=1

            prev=score

        ranked\_dict[method]=rank

    # print(ranked\_dict)

    return pd.Series(ranked\_dict)

ranked\_df=df.drop('Dataset',axis=1).apply(ranking,axis=1)

print(ranked\_df)

avg\_ranks={}

# for item in ranked\_df.items():

    # print(item)

for idx,val in ranked\_df.items():

    sum=0

    for i in val:sum+=i

    avg\_ranks[idx]=sum/len(val)

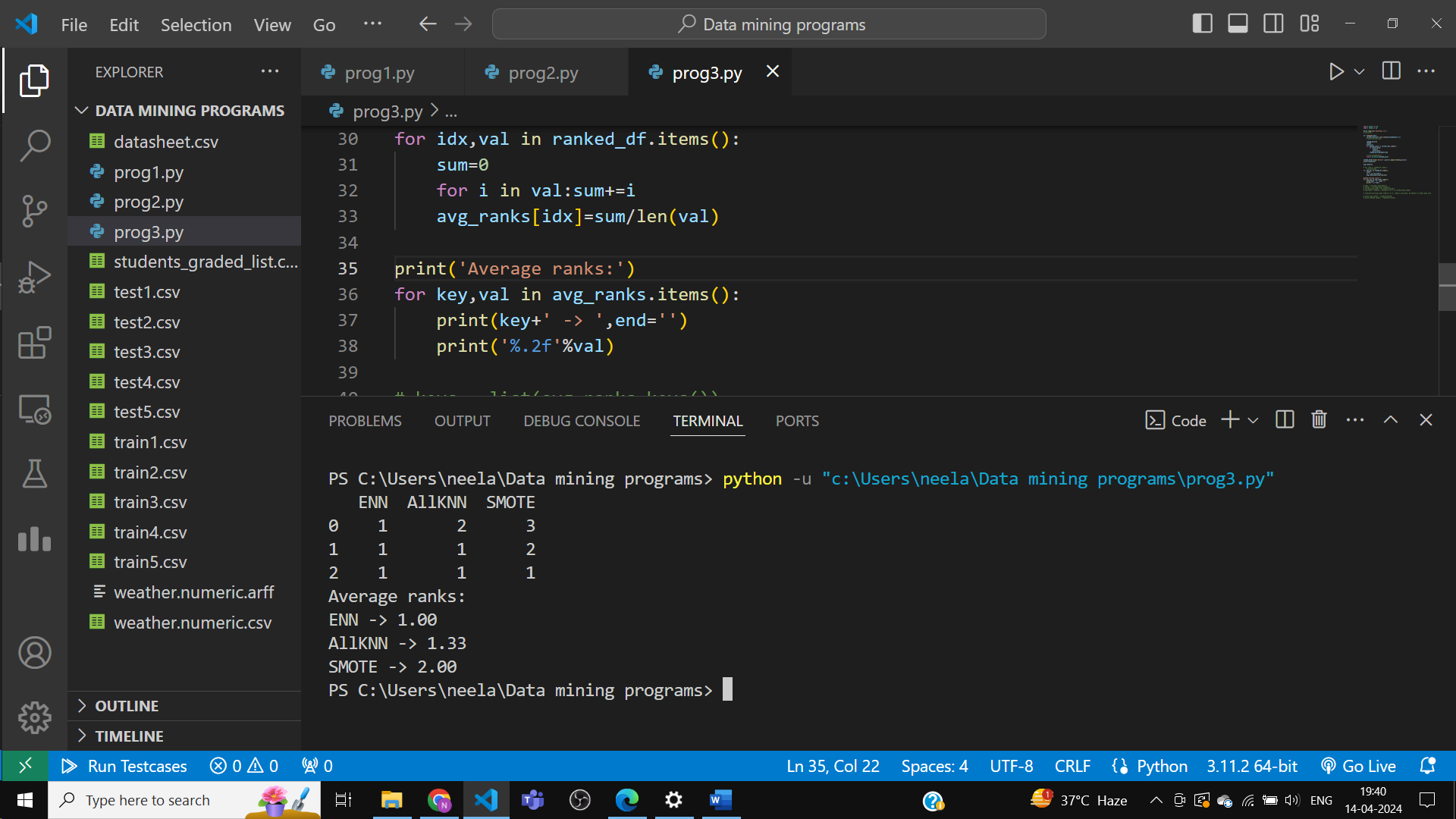
print('Average ranks:')

for key,val in avg\_ranks.items():

    print(key+' -> ',end='')

    print('%.2f'%val)

Output:



3. Identify lowest and highest performing methods (average rank high for the higher performer)

Python Code :

import pandas as pd

import numpy as np

df=pd.read\_csv('datasheet.csv')

# print(df)

def ranking(rows):

    sorted\_rows=rows.sort\_values(ascending=False)

    # print(sorted\_rows)

    ranked\_dict={}

    rank=0

    prev=None

    for method,score in sorted\_rows.items():

        if score!=prev:

            rank+=1

            prev=score

        ranked\_dict[method]=rank

    # print(ranked\_dict)

    return pd.Series(ranked\_dict)

ranked\_df=df.drop('Dataset',axis=1).apply(ranking,axis=1)

print(ranked\_df)

avg\_ranks={}

# for item in ranked\_df.items():

    # print(item)

for idx,val in ranked\_df.items():

    sum=0

    for i in val:sum+=i

    avg\_ranks[idx]=sum/len(val)

print('Average ranks:')

for key,val in avg\_ranks.items():

    print(key+' -> ',end='')

    print('%.2f'%val)

keys = list(avg\_ranks.keys())

values = list(avg\_ranks.values())

sorted\_value\_index = np.argsort(values)

avg\_ranks = {keys[i]: values[i] for i in sorted\_value\_index}

ranklist=list(avg\_ranks.items()) # if .items() not given, by default it means keys only

print('Top ranker: '+ranklist[0][0])

print('Bottom ranker: '+ranklist[-1][0])

Output :

