****

**ANL252**

**Python For Data Analytics**

**Group-based Assignment**

**July 2021 Presentation**

|  |  |
| --- | --- |
| **Name** | **PI No.** |
| **NEO HUI YING JOEY** | **B1882288** |
| **LOO CHENKAI** | **Y2071132** |
| **AMIR HAMZAH BIN ABDUL KHALID** | **B2110864** |

|  |  |  |
| --- | --- | --- |
| **Name** | **Contribution** | **Signature** |
| **LOO CHENKAI** | **Q1** |  |
| **AMIR HAMZAH BIN ABDUL KHALID** | **Q2** |  |
| **NEO HUI YING JOEY** | **Q3** | **A black spider on a white surface  Description automatically generated with low confidence** |

**Submission Date:** 29 August 2021, 2355 hrs

**Question 1a**

# #Q1 A)

# Construct a Python program to store the above data in a NumPy array.

# Import numpy & matplotlib.pyplot

import numpy as np

import matplotlib.pyplot as plt

data\_table = np.array([[4 , 0.2 , 1.16],

[6 , 0.1 , 0.06],

[8 , 0.3 , -1.79],

[4 , 0.6 , 1.55],

[10 , 0.1 , -4.88],

[1 , 0.4 , 1.37],

[9 , 0.6 , -1.25],

[5 , 0.3 , -1.1],

[2 , 0.5 , 3.23],

[7 , 0.5 , -2.71],

[8 , 0.1 , -0.99],

[2 , 0.9 , 3.23],

[2 , 0.8 , 4.55],

[8 , 1 , 2.7],

[7, 0.9 , -1.13],

[9 , 0.1 , -0.88],

[1 , 0.2 , 2.08],

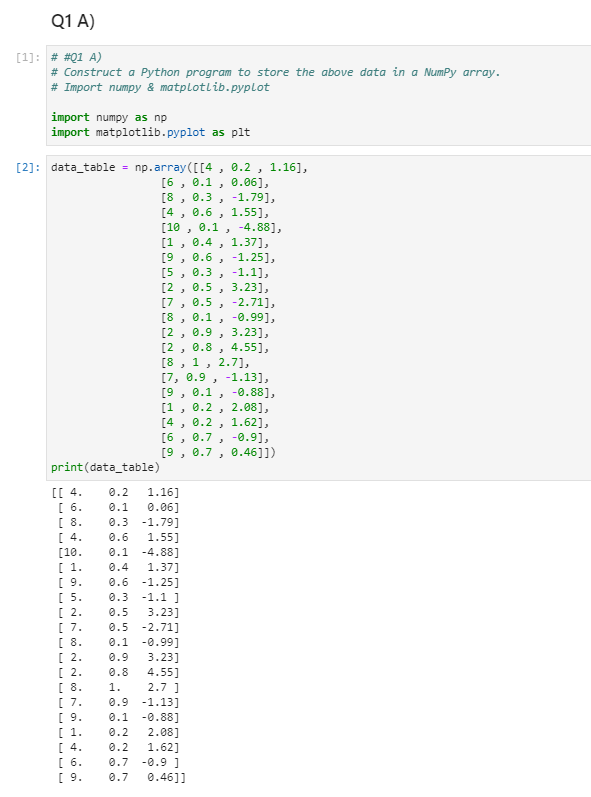
[4 , 0.2 , 1.62],

[6 , 0.7 , -0.9],

[9 , 0.7 , 0.46]])

print(data\_table)

**Screenshot:**



**Question 1b**

# Q1 B)

# Design a Python program to compute Ŷ for every row of the

# array and store the results in a separate NumPy array as well.

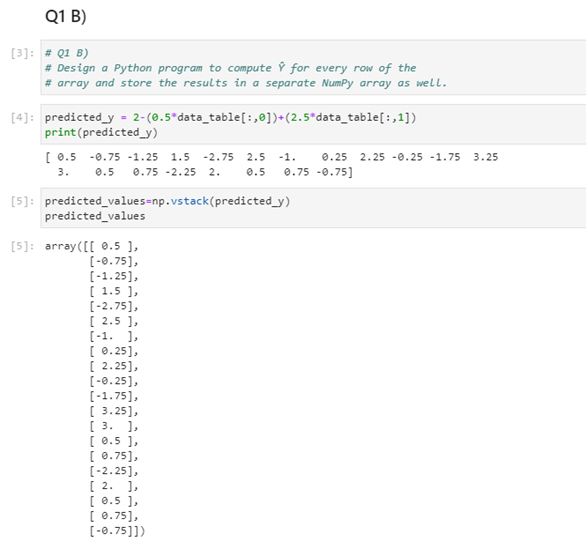
predicted\_y = 2-(0.5\*data\_table[:,0])+(2.5\*data\_table[:,1])

print(predicted\_y)

predicted\_values=np.vstack(predicted\_y)

predicted\_values

**Screenshot:**



**Question 1c**

#Q1 C)

# Use a Python program to compute ê for every row of the

# array and store the results in a separate NumPy array.

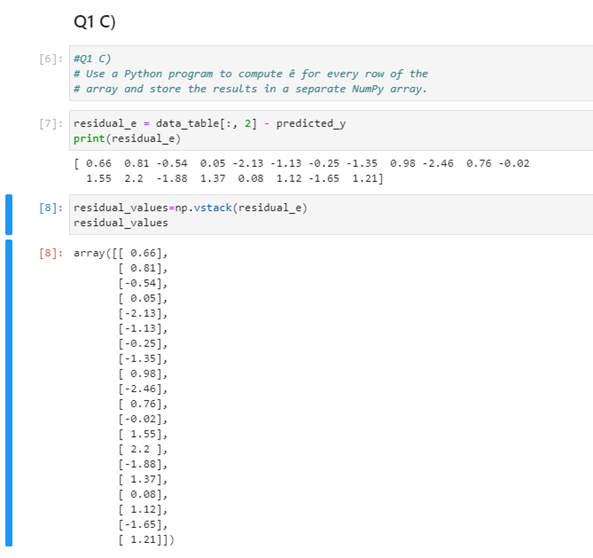
residual\_e = data\_table[:, 2] - predicted\_y

print(residual\_e)

residual\_values=np.vstack(residual\_e)

residual\_values

**Screenshot of Q1C:**



**Question 1d**

# Q1 D)

# Create a histogram of the residuals calculated in (c) by using the matplotlib package.

# Adjust the parameters of the chart so the ticks on the x-axis can be read clearly,

# a title is given to the chart, and both the axes are labelled.

plt.hist(residual\_values, bins=(-4,-3,-2,-1,0,1,2,3,4), edgecolor='white', color='black', linewidth=1)

plt.xlabel('Residuals')

plt.ylabel('Frequency')

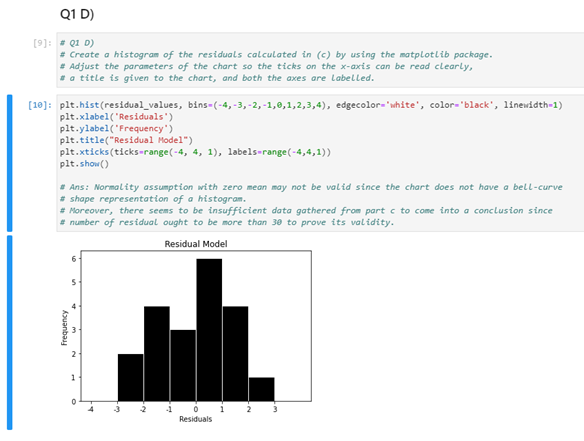
plt.title("Residual Model")

plt.xticks(ticks=range(-4, 4, 1), labels=range(-4,4,1))

plt.show()

# Ans: Normality assumption with zero mean may not be valid since the chart does not have a bell-curve shape representation of a histogram. Moreover, there seems to be insufficient data gathered from part c to come into a conclusion since the number of residual ought to be more than 30 to prove its validity.

**Screenshot:**



**Question 1e**

#Q1 E)

# Write a Python program to create such a scatter plot for

# checking the constant variance assumption.

# Eventually, discuss whether you agree that the constant variance

# assumption is valid based on this scatter plot.

plt.scatter(predicted\_values, residual\_values, color = "black", marker = "o", edgecolor = "white")

plt.axhline(y=0, color = "blue", linestyle="-")

plt.xlabel("Predicted Values")

plt.ylabel("Residual")

plt.xticks(ticks=range(-3,5,1), labels = range(-3,5,1))

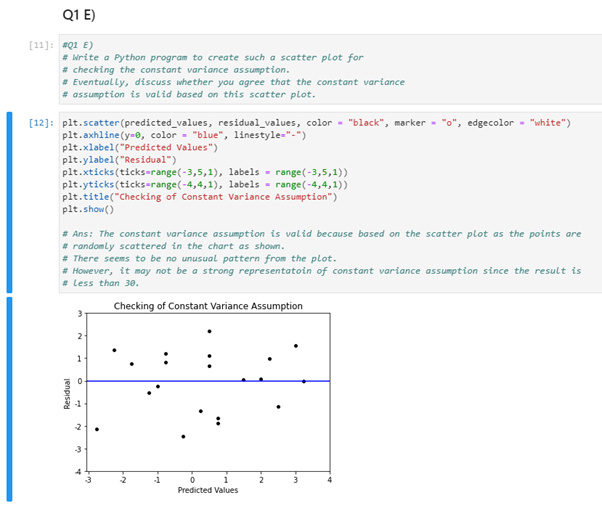
plt.yticks(ticks=range(-4,4,1), labels = range(-4,4,1))

plt.title("Checking of Constant Variance Assumption")

plt.show()

# Ans: The constant variance assumption is valid because based on the scatter plot as the points are randomly scattered in the chart as shown. There seems to be no unusual pattern from the plot. However, it may not be a strong representation of constant variance assumption since the result is less than 30.

**Screenshot:**



**Question 2a**

#A) Prepare a Python program to read in and to convert the data from a .csv text file into a pandas DataFrame.

#Check the existing missing data in the dataset and adjust the reader accordingly.

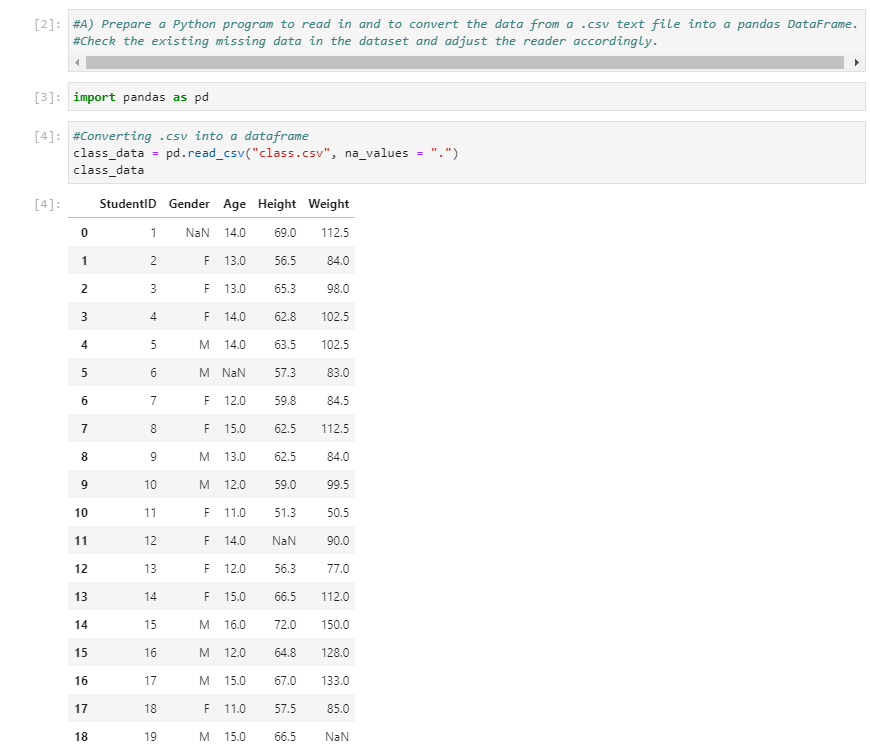
import pandas as pd

#Converting .csv into a dataframe

class\_data = pd.read\_csv("class.csv", na\_values = ".")

class\_data

**Screenshot:**



**Question 2b**

# B) The data should be sorted by the age of the students in the descending order

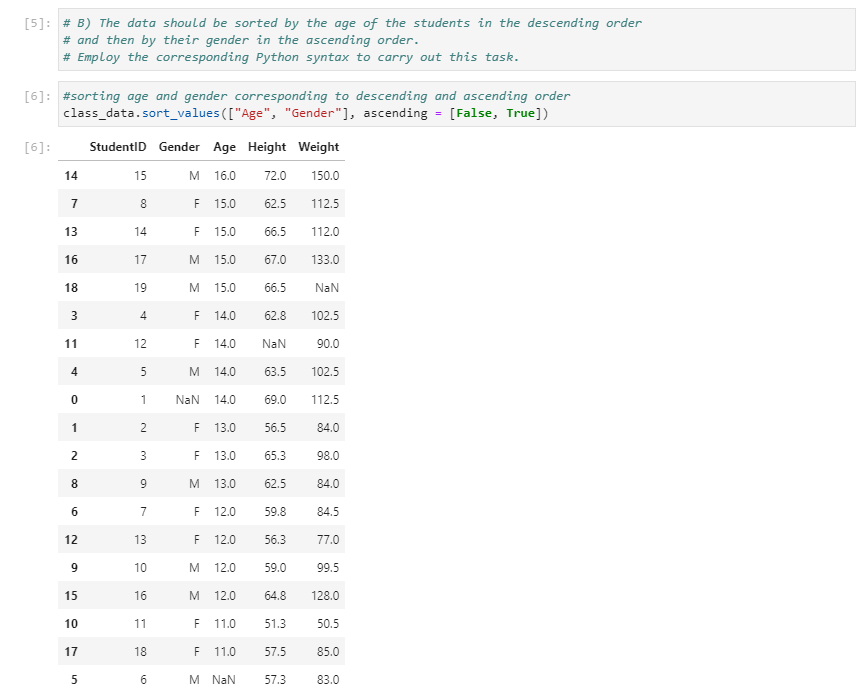
# and then by their gender in the ascending order.

# Employ the corresponding Python syntax to carry out this task.

#sorting age and gender corresponding to descending and ascending order

class\_data.sort\_values(["Age", "Gender"], ascending = [False, True])

**Screenshot:**



**Question 2c**

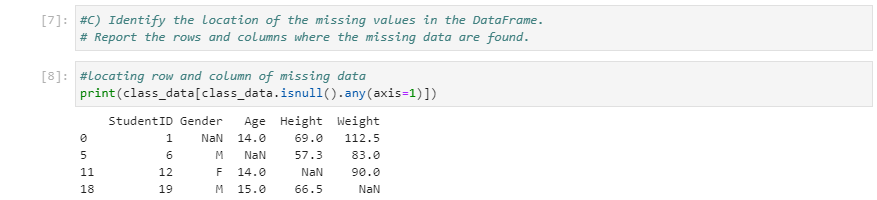
#C) Identify the location of the missing values in the DataFrame.

# Report the rows and columns where the missing data are found.

#locating row and column of missing data

print(class\_data[class\_data.isnull().any(axis=1)])

**Screenshot:**

****

**Question 2d**

#D) If missing values are detected in the DataFrame,

# they have to be treated according to the columns they belong to.

# Here are the instructions of how we should deal with the missing data in each column:

#Gender – replace missing values by the gender with the highest frequency

#Age – replace missing values by the median age

#Height – replace missing values by the mean height

#Weight – replace missing values by the mean weight

# Design your own Python program to determine the corresponding statistics for each column to

# replace the missing values in it.

#assigning mode of Gender to missing value

#assigning median of Age to missing value

#assigning mean of Height and weight to missing values

class\_data["Gender"].fillna(value=class\_data["Gender"].mode(), inplace=True)

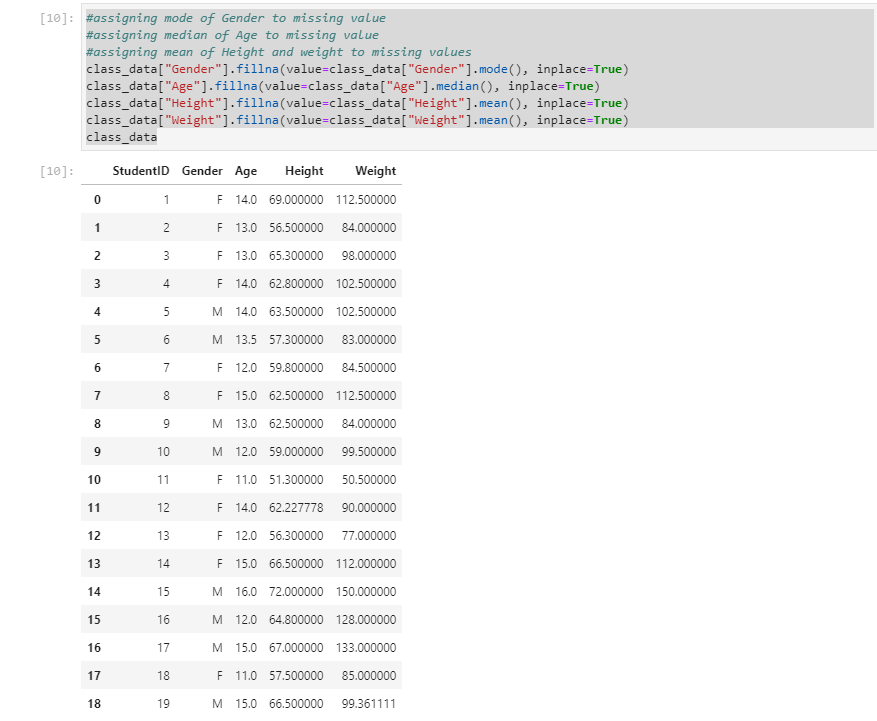
class\_data["Age"].fillna(value=class\_data["Age"].median(), inplace=True)

class\_data["Height"].fillna(value=class\_data["Height"].mean(), inplace=True)

class\_data["Weight"].fillna(value=class\_data["Weight"].mean(), inplace=True)

class\_data

**Screenshot:**

****

**Question 2e**

#E) Use Python code to detect outliers in the DataFrame and delete the corresponding rows if they exist.

#Get interquartile ranges (Height) and retrieve outlier

height\_q1 = class\_data["Height"].quantile(q=0.25)

height\_q3 = class\_data["Height"].quantile(q=0.75)

height\_iqr = height\_q3 - height\_q1

print(height\_q1)

print(height\_q3)

height\_iqr

class\_data[~((class\_data["Height"]<height\_q1-1.5\*height\_iqr)|(class\_data["Height"]>height\_q3-1.5\*height\_iqr))]

# Outlier detected - index 10

#Get interquartile ranges (Weight) and retrieve outlier

weight\_q1 = class\_data["Weight"].quantile(q=0.25)

weight\_q3 = class\_data["Weight"].quantile(q=0.75)

weight\_iqr = weight\_q3 - weight\_q1

print(weight\_q1)

print(weight\_q3)

weight\_iqr

class\_data[~((class\_data["Weight"]<weight\_q1-1.5\*weight\_iqr)|(class\_data["Weight"]>weight\_q3-1.5\*weight\_iqr))]

# Outlier detected - index 10

#Get interquartile ranges (Age) and retrieve outlier

age\_q1 = class\_data["Age"].quantile(q=0.25)

age\_q3 = class\_data["Age"].quantile(q=0.75)

age\_iqr = age\_q3 - age\_q1

print(age\_q1)

print(age\_q3)

age\_iqr

class\_data[~((class\_data["Age"]<age\_q1-1.5\*age\_iqr)|(class\_data["Age"]>age\_q3-1.5\*age\_iqr))]

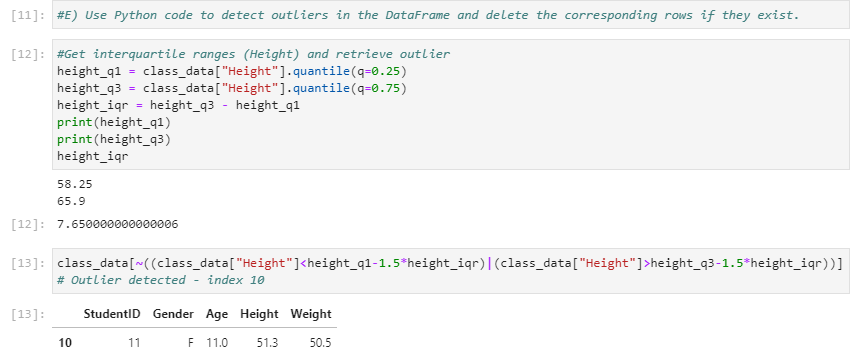
# No Outlier detected

#remove outlier (in index 10) and display final table

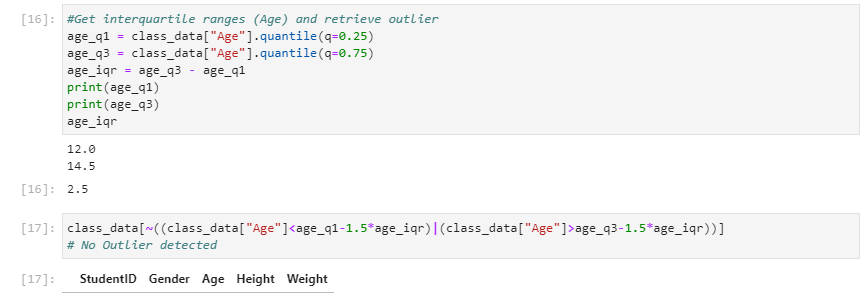
class\_data.drop(10, axis = 0)

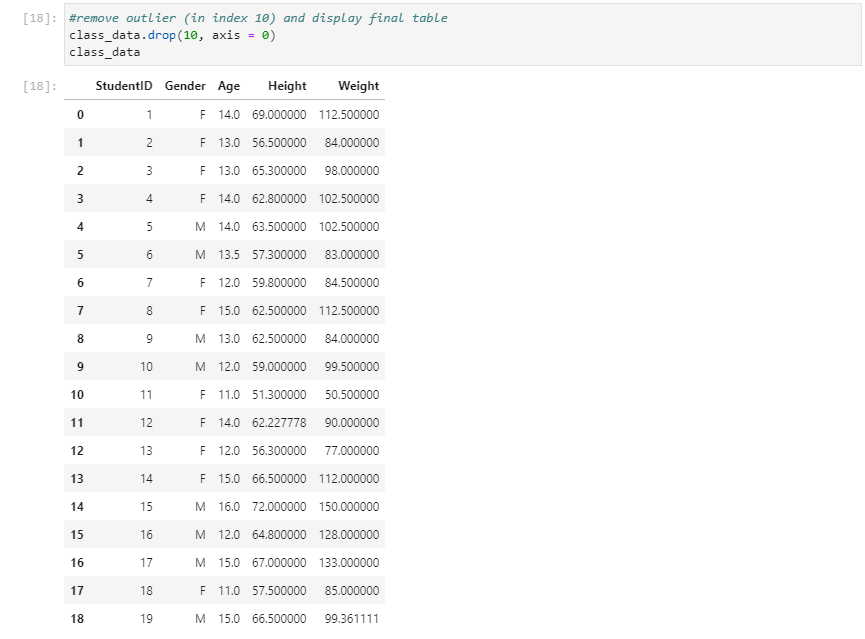
class\_data

**Screenshot:**









**Question 3**

Inner join selects and displays the common data of the DataFrames merged. Outer join selects and displays the combination of all the data, regardless of their commonality. However, the results returned vary from the types of outer join.

Both the concat() and merge() functions can be adopted. For concat() function, if the axis stated is 0, the common columns will be searched for inner joins, while all the columns will be included for outer joins, and the rows will be concatenated. If the stated axis is 1, the function will search for the common rows for inner joins, while all the rows will be included for outer joins, and the columns will be concatenated. Merge() functions will take the DataFrames as arguments and the stated column for the join to be performed on. For inner joins, only rows that have matching column values will be returned. Left joins will return every row in the left DataFrame, right joins will return every row in the right DataFrame and every row in both DataFrames will be returned for full joins. When performing outer joins using either function, if the rows or columns do not have corresponding values, null values will be assigned.