Logo, company name

Description automatically generated

|  |  |
| --- | --- |
| **Course:** | ANL252 |
| **Assignment:** | GBA01 |
| **Group Members:** | Tan Jing Jie (Q1882541) |
| Ang Chung Siong Derek (N2072594) |
| **Group:** | 03 |
| **Tutorial Group:** | T09 |
| **Submission Date:** | 15 August 2021 |

**\*\* Inactive Member: Tan Zhi Hao (Q1711184) – missing in action. Do not have any contribution to the group project. Informed via email on 23 August 2021 subject GBA01 – Inactive group member.**

**Question 1:**

**a)**

Code:

#import of numPy into Python

import numpy as np

#Storing dataset as numpy array

numbers = 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,-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]])

numbers

Output:

A picture containing shape

Description automatically generated

**b)**

Code:

#import pandas into python

import pandas as pd

#Assigning variables name to columns

X1 = numbers[...,0]

X2 = numbers[...,1]

#Defining Ŷ

Ŷ = [2 - 0.5\*X1[loop1] + 2.5\*X2[loop1]

for loop1 in range(len(X1))]

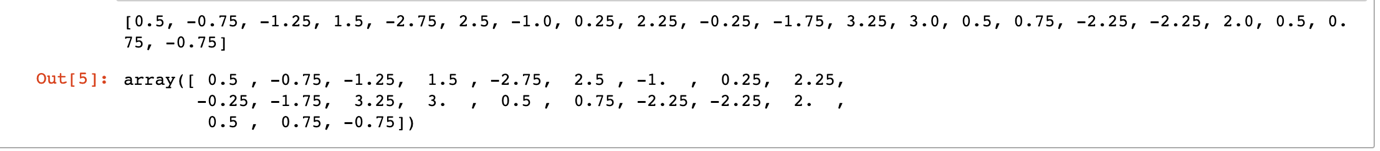
print(Ŷ)

#Converting list into array

Ŷ\_arr = np.array(Ŷ)

Ŷ\_arr

Output:



**c)**

Code:

#Defining variable Y in column 3.

Y = numbers[...,2]

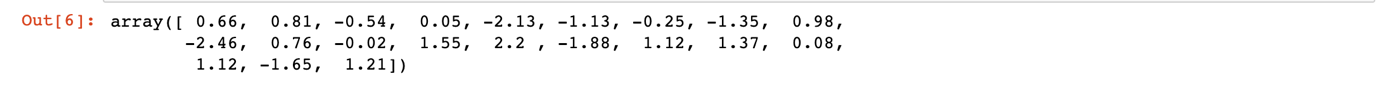
model\_e = [Y[loop1] - Ŷ[loop1] for loop1 in range(len(X1))]

#Converting list of numbers computed in model\_e to array

model\_e\_arr = np.array(model\_e)

model\_e\_arr

Output:



**d)**

Code:

#import of matplotlib

import matplotlib.pyplot as plt

#Plotting of histogram

fs = 20

plt.hist(model\_e\_arr, bins= "auto")

plt.title("Histogram of the residuals calculated", fontsize = fs)

plt.xlabel("Residual", fontsize = fs)

plt.ylabel("Frequency", fontsize = fs)

plt.xticks([-3,-2,-1,0,1,2,3])

Output:

Chart, histogram

Description automatically generated

Discussion:

Based on the shape of the histogram, we do not agree with the normality assumption with zero mean. This is because most of the data is more than zero. This means that the mean of the data is more than zero. Based on the shape reflected in the histogram, it is skew to the right and a normal distribution would have reflected a bell shape with the peak in the middle. Therefore, the data points is not normally distribution.

**e)**

Code:

#Plotting of scatter plot

plt.scatter(Y, model\_e\_arr)

plt.title("Scatter plot of Residuals vs predicted Y")

plt.xlabel("Predicted value of Y")

plt.ylabel("Residuals")

plt.xticks([-5,-4,-3,-2,-1,0,1,2,3,4,5])

plt.yticks([-3,-2,-1,0,1,2,3])

plt.show()

Output:

Chart, scatter chart

Description automatically generated

Discussion:

Based on the scatterplot, we do not agree that the constant variance assumption is valid. A constant variance would reflect a horizontal line as the value of predicted Y increases. However, the scatterplot reflect an linear upward pattern as the value of Y increases. This shows that the variance are increasing. Besides the upwards trending, when the predicted value of Y is at -1, it shows a vertical line, reflecting a very difference variance across multiple predicted value of Y. These observations reflected that the constant variance assumption is invalid.

**Question 2:**

**a)**

Code:

# import of pandas into jupyter notebook

import pandas as pd

import numpy as np

#import of class.csv into notebook

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

class\_data

Output:

Table

Description automatically generated

Code:

#Error in the label of column head. (Derived from assumption that the height column is supposed to be for weight and the weight column is supposed to be for height base on observation.

class\_data.rename(columns = {"Height" : "Weight", "Weight":"Height"}, inplace="True")

class\_data.head(1)

Output:



**b)**

Code:

#sorting of age in decending order then Gender by ascending order.

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

class\_data = class\_data.reset\_index(drop=True)

class\_data

Output:

Table

Description automatically generated

**c)**

Code:

#Detecting and reporting missing values.

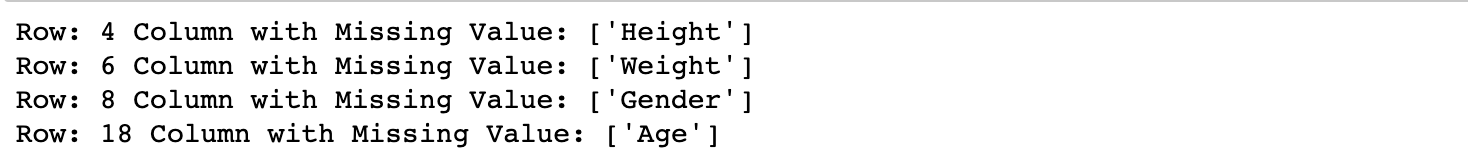
null\_data = class\_data[class\_data.isnull().any(axis=1)]

for i in range(len(null\_data.index)) :

a = null\_data.iloc[[i]]

print("Row:", a.index[0], "Column with Missing Value:", a.columns[a.isna().any()].tolist())

Output:



**d)**

Code:

gender\_counts = class\_data['Gender'].value\_counts().to\_dict()

default\_gender = 'M'

if (gender\_counts.get('F') > gender\_counts.get('M')):

default\_gender = 'F'

class\_data['Gender'].fillna(default\_gender, inplace=True)

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

# class\_data['Age'] = class\_data['Age'].round(2)

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

class\_data['Height'] = class\_data['Height'].round(1)

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

class\_data['Weight'] = class\_data['Weight'].round(1)

class\_data

Output:

Table

Description automatically generated

**e)**

Code:

#Determine the lower and upper bound for Age

q1 = class\_data['Age'].quantile(q = 0.25)

q3 = class\_data['Age'].quantile(q = 0.75)

iqr = q3-q1

age1 = q1 - 1.5\*iqr

age2 = q3 + 1.5\*iqr

print(f"Lower Bound:\n {age1}")

print(f"Upper Bound:\n {age2}")

#Detecting outliers in Age column and removing it

age\_outlier = class\_data[~(

(class\_data["Age"]< age1) | (class\_data["Age"]> age2)

)]

age\_outlier

Output:

Table

Description automatically generated

Code:

#Determine the lower and upper bound for Weight

q1 = class\_data['Weight'].quantile(q = 0.25)

q3 = class\_data['Weight'].quantile(q = 0.75)

iqr = q3-q1

weight1 = q1 - 1.5\*iqr

weight2 = q3 + 1.5\*iqr

print(f"Lower Bound:\n {weight1}")

print(f"Upper Bound:\n {weight2}")

#Detecting outliers in Weight column and removing it

weight\_outlier = age\_outlier[~(

(age\_outlier["Weight"]< weight1) | (age\_outlier["Weight"]> weight2)

)]

weight\_outlier

Output:

Table

Description automatically generated

Code:

#Determine the lower and upper bound for Height

q1 = class\_data['Height'].quantile(q = 0.25)

q3 = class\_data['Height'].quantile(q = 0.75)

iqr = q3-q1

height1 = q1 - 1.5\*iqr

height2 = q3 + 1.5\*iqr

print(f"Lower Bound:\n {height1}")

print(f"Upper Bound:\n {height2}")

#Detecting outliers in Height column and removing it

height\_outlier = weight\_outlier[~(

(weight\_outlier["Height"]< height1) | (weight\_outlier["Height"]> height2)

)]

height\_outlier

Output:

Table

Description automatically generated

**Question 3:**

Both inner and outer joins are used to merge two or more dataframe in python. First, we must ensure that we import pandas into python and then import both dataset via the csv via df = pd.read\_csv(file\_name.csv) or converting array into dataframe (df = pd.DataFrame(array\_name, colums = [‘Col\_1’, ‘Col\_2’, ‘Col\_3’]).

Once there is 2 or more dataframe in python, we can use inner join or outer join to merge both dataset together by using these code:

#When we peform join operation using pandas

#We used 2 data frames sets as example, namely EXP1 and EXP2, both sets of data have common column

#Executing the Inner join:

#inner join

df = pd.merge(exp1, exp2, on='id', how='inner')

#Executing left outer join:

# left outer join

df = pd.merge(exp1, exp2, on='id', how='left')

#Executing the right outer join:

#right outer join

df = pd.merge(exp1, exp2, on='id', how='right')

#Executing the full outer join:

# full outer join

df = pd.merge(exp1, exp2, on='id', how='outer')

The main differences between an inner join and an outer join is that an inner join can only be done if both dataset has similar record id in the dataset or similar variables. For example, if one dataset has variable 1 and 2 while the other has variable 1 & 3, the execution of inner join will only merge all the records in variable 1 while removing variable 2 and 3. For example:

Dataset 1:

|  |  |  |
| --- | --- | --- |
| **ID** | **Variable 1** | **Variable\_2** |
| 1 | 1 | 0 |
| 2 | 1 | 1 |

Dataset 1:

|  |  |  |
| --- | --- | --- |
| **ID** | **Variable 1** | **Variable\_3** |
| 3 | 1 | 0 |
| 4 | 1 | 1 |

Output of an inner join:

|  |  |
| --- | --- |
| **ID** | **Variable 1** |
| 1 | 1 |
| 2 | 1 |
| 3 | 1 |
| 4 | 1 |

While for an outer join, the differences in the variables or record id will not be an issue. All the records in the two dataset will be merge and any missing records will return NAN in the corresponding rows. For example, if dataset 1 has variable 1 and 2 while dataset 2 has variable 1 and 3, the output of the join will be as such:

Dataset 1:

|  |  |  |
| --- | --- | --- |
| **ID** | **Variable 1** | **Variable\_2** |
| 1 | 1 | 0 |
| 2 | 1 | 1 |

Dataset 1:

|  |  |  |
| --- | --- | --- |
| **ID** | **Variable 1** | **Variable\_3** |
| 3 | 1 | 0 |
| 4 | 1 | 1 |

Output after outer join:

|  |  |  |  |
| --- | --- | --- | --- |
| **ID** | **Variable 1** | **Variable\_2** | **Variable\_3** |
| 1 | 1 | 0 | NAN |
| 2 | 1 | 1 | NAN |
| 3 | 1 | NAN | 0 |
| 4 | 1 | NAN | 1 |

Hence, no matter the differences of the two datasets, an outer join can be executed.

Another differences between inner join and outer is the purpose of the merge. If the analyst is required to look at similar variables or records of the datasets an inner join should be executed as the inner join will ignore incomplete datas when both datasets are joined. But if the analyst required to merge both datasets regardless of the missing records, an outer join should be use.

Lastly, in outer joins, there is also full, right and left outer join while inner join only has one function. Full outer join allows all data to merge without restriction. As for right out join, all the data in the right data frame will be included while only the common data in the left data frame will be merge into the new data frame. Similarly for left outer join, all data in the left dataset will be included while only common data in the right dataset will be merged.