

**ANL252**

**Python for Data Analytics**

# **Group-Based Assignment**

**July 2021 Presentation**

**Submitted by:**

|  |  |
| --- | --- |
| **Name** | **PI No.** |
| **Peh Hong Wei Dyan** | **H1870791** |
| **Chiu Li Wen Renee** | **K1981620** |
| **Tan Yi Hao** | **H2070659** |
| **Geekger Ee** | **W2071283** |
| **Elijah Song Ming Fu** | **K2070438** |

**Tutorial Group: ­­­­­­­­­­­­ T09 - GBA 1**

**Instructor’s Name: Mr. MUNISH KUMAR**

**Submission Date: 26/08/2021**

**Question 1**

(a)

# Import required python packages

import numpy as np

import matplotlib.pyplot as plt

# Storing the data as numpy arrays

x1 = np.array([4, 6, 8, 4, 10, 1, 9, 5, 2, 7, 8, 2, 2, 8, 7, 9, 1, 4, 6, 9])

x2 = np.array([0.2, 0.1, 0.3, 0.6, 0.1, 0.4, 0.6, 0.3, 0.5, 0.5, 0.1, 0.9, 0.8, 1, 0.9, 0.1, 0.2, 0.2, 0.7, 0.7])

y = np.array([1.16, 0.06, -1.79, 1.55, -4.88, 1.37, -1.25, -1.1, 3.23, -2.71, -0.99, 3.23, 4.55, 2.7, -1.13, -0.88, 2.08, 1.62, -0.9, 0.46])

(b)

# Create a defined function (DF) to store regression formula

def y\_prime(x1,x2):

yp = 2 - 0.5 \* x1 + 2.5 \* x2

return yp

# Create new array to store y-prime answers

array\_y\_prime = []

# Create a row counter starting at 0, as array starts counting at 0

i = 0

# While loop to pull out array row by row then sub into DF to calculate y prime, up till i = 19, or 20 rows

while i <= 19:

x1\_amt = x1[i]

x2\_amt = x2[i]

y\_prime\_formula = y\_prime(x1\_amt,x2\_amt)

print("Y-prime for ", "X1 = ", x1[i], " and X2 = ", x2[i], "is :", y\_prime\_formula, "\n")

#Store in new array

array\_y\_prime.append(y\_prime\_formula)

#Add 1 to the row counter

i += 1

else:

print("Stored all the y prime values into an array:", array\_y\_prime)

(c)

# Create another defined function (DF) to store model e\_prime formula

def e\_prime(y\_amt,y\_prime\_formula):

ep = y\_amt - y\_prime\_formula

return ep

# Create new array to store e-prime answers

array\_e\_prime = []

# Create a row counter starting at 0, as array starts counting at 0

i = 0

# While loop to pull out array row by row then sub into DF to calculate y prime, up till i = 19, or 20 rows

while i <= 19:

x1\_amt = x1[i]

x2\_amt = x2[i]

y\_amt = y[i]

y\_prime\_formula = y\_prime(x1\_amt,x2\_amt)

e\_prime\_formula = e\_prime(y\_amt,y\_prime\_formula)

#Store in new array

array\_e\_prime.append(e\_prime\_formula)

#Add 1 to the row counter

i += 1

else:

print("Stored all the e prime values into an array:", array\_e\_prime)

(d)

# creating the histogram of residuals

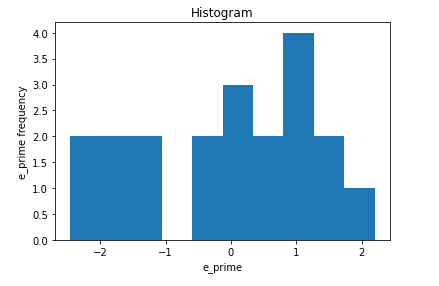
plt.hist(array\_e\_prime)

plt.title('Histogram')

plt.xlabel('e\_prime')

plt.ylabel('e\_prime frequency')

plt.show()



From the histogram it is evident that the distribution is not really normal as there’s an outlier.

(e)

# creating the scatter plot

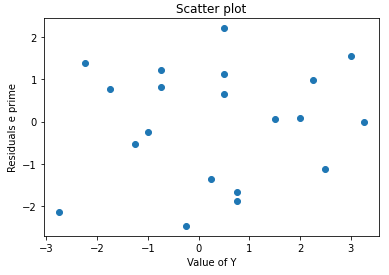
plt.scatter(array\_y\_prime, array\_e\_prime)

plt.title("Scatter plot")

plt.xlabel("Value of Y")

plt.ylabel("Residuals e prime")

plt.show()



As seen from the scatter plot, there is no distinct pattern to show.

**Question 2**

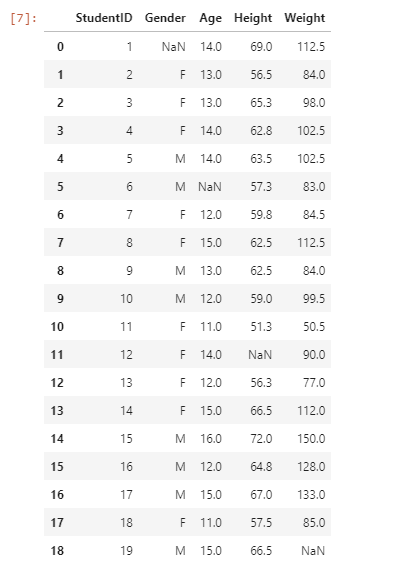
import pandas as pd

import math

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

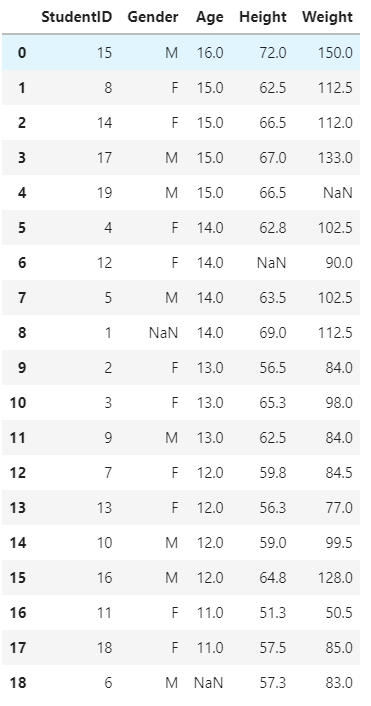
#can remove na\_filter as there are no white spaces

#originally the missing values were just “.”



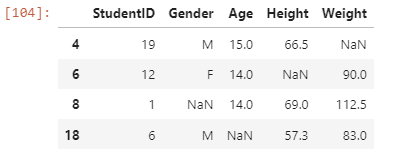
dfclass.sort\_values(by = ["Age","Gender"], ascending = [False,True], inplace = True, ignore\_index=True)

dfclass



missing = dfclass.isnull().any(axis=1)

dfclass.loc[missing[missing==True].index]



#Gender  
missingGender = dfclass["Gender"].fillna(dfclass["Gender"].mode(), inplace = True)  
  
#Age  
missingAge = dfclass["Age"].fillna(dfclass["Age"].median(), inplace = True)

dfclass["Age"] =dfclass['Age'].apply(lambda x: math.floor(x))  
  
#Height  
missingHeight = dfclass["Height"].fillna(dfclass["Height"].mean(), inplace = True)

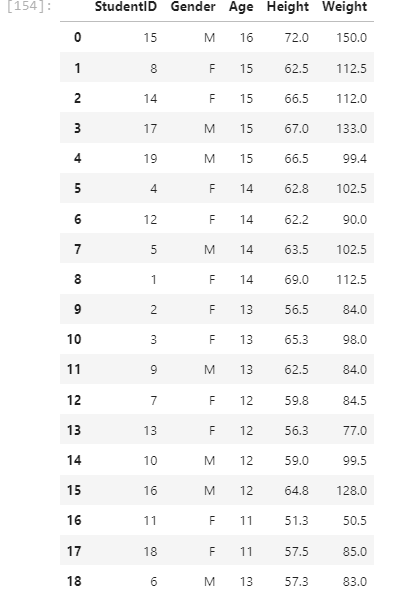
decimals = 2

dfclass["Height"] =dfclass['Height'].apply(lambda x: round(x, decimals))  
  
#Weight

missingHeight = dfclass["Weight"].fillna(dfclass["Weight"].mean(), inplace = True)

dfclass["Weight"] =dfclass['Weight'].apply(lambda x: round(x, decimals))

dfclass



Reason:

Even though it is supposed to be grouped by age and gender the value at row 18 does not match because it was updated after grouping. Therefore, it is not grouped perfectly.

1. #will not remove anything as there is no outlier

def quantilesRange(field):

quantilevalue = field.quantile([0.25, 0.75]).values

IQR = quantilevalue[1] - quantilevalue[0]

upper = quantilevalue[1] +1.5\* IQR

lower = quantilevalue[0] -1.5\* IQR

return(lower,upper)

for x in ['Age','Height','Weight']:

upper,lower = quantilesRange(dfclass[x])

dropoutliers = (dfclass[x] < lower) | (dfclass[x] > upper)

if len(dfclass[x].values ==0):

print(f"No outliers found in {x}")

else:

print(f"Outlier found in {x} is {dfclass[x].values}")

dfclass

**Question 3**

The difference between the inner and outer join when merging is that both joins will pull different data into the joined table. Both joins look for similar variants or objects from tables to merge. But when the outer join joins, it will not only bring the requested variant or object over but also the other variants and objects. As for the missing data, “Nan” will be shown to indicate that no data is in those columns or rows. For inner join, it will only look for the requested variant or object to display in the new table as for the other non-required variant or object, they will be left out. Outer join also works like an “OR” logical operator and inner join works like an “AND” logical operator.

To carry out the mentioned in the pandas package, the 1st step is to import the pandas in. It will then be followed by using either the pandas.concat function or the pandas.DataFrame.merge function. For the pandas.concat function, users will have to indicate the variant or object’s name, the axis to present in left to right or up to down format, and the join type. As for pandas.DataFrame.merge function, the user will have to indicate the dataframe name, “how” the type of join, and on the variant or object to be pulled over.

*Example:*

*inner\_join = pd.concat([df1, df2], axis=1, join="inner")* or … *join=”outer”* to decide either inner or outer join.

**References**

Unpingco J. (2021) Pandas. In: Python Programming for Data Analysis. Springer, Cham.

https://doi.org/10.1007/978-3-030-68952-0\_5

Wu, K. Y. (2021). ANL252 Python for data analytics (study guide). *Singapore University of*

*Social Sciences.*