****

**ANL252**

**Python for Data Analytics**

**End-of-Course Assessment**

**July 2021 Presentation**

Name: Chiu Li Wen Renee

PI Number: K1981620

Tutorial Group: T09

***Declaration:***

“*I declare that this assignment is my own work, unless otherwise acknowledged or credited by appropriate referencing. I have read and abide by the SUSS Honour Code and I am aware of the penalties associated with plagiarism and collusion listed in the SUSS Student Handbook*.”

**Embed Full Code:**



**Question 1: Task(a)(i) - Code**

# ANL252 ECA - K1981620 - Chiu Li Wen Renee

# Import necessary packages

import math

import numpy as np

import pandas as pd

import sqlite3

# Q1 Task (a)(i)

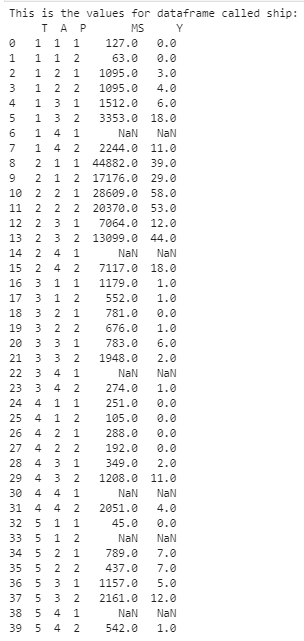
# Read new csv file "ship.csv" and convert to pandas DataFrame

# Recognise missing values namely "." values, and declare them as missing values using "na\_values"

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

print("This is the values for dataframe called ship:", "\n", ship)

**Question 1: Task(a)(i) - Output**



**Question 1: Task(a)(ii) - Code**

# Q1 Task (a)(ii)

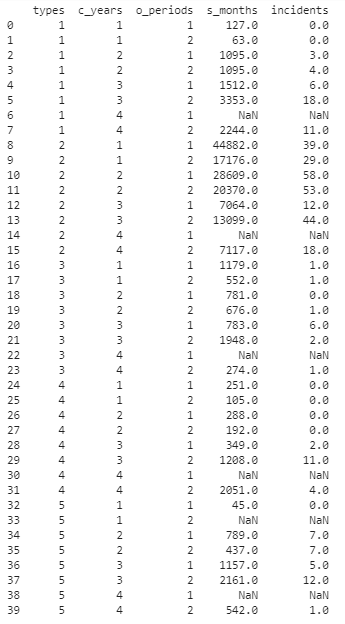
# If use list to rename: DataFrame\_Ship.columns = ['types', 'c\_years', 'o\_periods', 's\_months','incidents']

# Replace the old dataframe columns' names using list

ship = ship.rename(columns={'T':'types', 'A':'c\_years', 'P':'o\_periods', 'MS': 's\_months', 'Y': 'incidents'})

print(ship)

**Question 1: Task(a)(ii) – Output**



**Question 1: Task(a)(iii) - Code**

# Q1 Task (a)(iii)

# Compute avg s\_months and avg incidents for cross products of every category in types and operation periods

# Averages shld be rounded to nearest int

# Store resulting table in an object named "shipgroup"

# Group types and o\_periods together, then find the average of s\_months and incidents using mean()

#DataFrame\_Ship = DataFrame\_Ship.fillna(method="ffill")

#print(DataFrame\_Ship)

ship = ship.groupby(['types', 'o\_periods']).mean(['s\_months','incidents'])

# Round only 's\_months' and 'incidents' values in the dataframe to integer, and also store table in an object called 'shipgroup'

# Source 1: https://www.geeksforgeeks.org/convert-floats-to-integers-in-a-pandas-dataframe/

# Source 2: https://www.geeksforgeeks.org/python-pandas-dataframe-round/#:~:text=Pandas%20is%20one%20of%20those%20packages%20and%20makes,by%20different%20places.%20Syntax%3A%20DataFrame.round%20%28decimals%3D0%2C%20%2Aargs%2C%20%2A%2Akwargs%29

shipgroup = ship.round({'s\_months':0,'incidents':0}).astype({'s\_months':int,'incidents':int})

print(shipgroup)

**Question 1: Task(a)(iii) – Output**



**Question 1: Task(a)(iv) - Code**

# Q1 Task (a)(iv)

# Re-read and rename "ship.csv" as the dataframe "ship", to get "ship" back to its unmerged state for data manupulation in part (a)(iv)

# Recognise missing values namely "." values, and declare them as missing values using "na\_values"

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

ship = ship.rename(columns={'T':'types', 'A':'c\_years', 'P':'o\_periods', 'MS': 's\_months', 'Y': 'incidents'})

# Fill only NaN values from DataFrame\_Ship's 's\_months' and 'incidents', with the means of 's\_months' and 'incidents', after grouping the dataframe according to 'types' and 'o\_periods' using .groupby()

# Source: https://towardsdatascience.com/when-to-use-pandas-transform-function-df8861aa0dcf

# Code Explanation:

# Select DataFrame\_Ship's 's\_months' and 'incidents' columns by using DataFrame\_Ship['specific column name'], then fill the NaN values of these columnes with .fillna()

# To find the average, use .transform('mean'), instead of .mean to enable the missing NaN values to be replaced later

# For DataFrame\_Ship's 's\_months' column

ship['s\_months'].fillna(ship.groupby(['types', 'o\_periods'])['s\_months'].transform('mean'), inplace=True)

# For DataFrame\_Ship's 'incidents' column

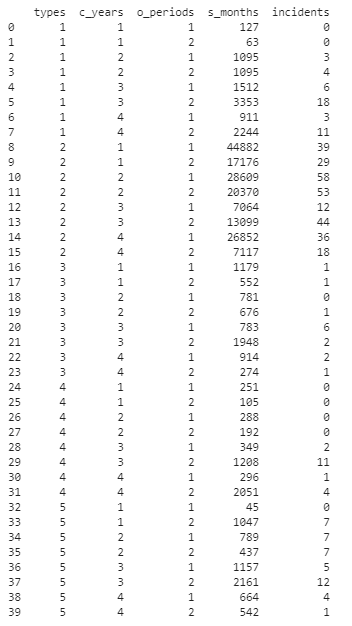
ship['incidents'].fillna(ship.groupby(['types', 'o\_periods'])['incidents'].transform('mean'), inplace=True)

# Store the newly updated dataframe back into "ship", and round 's\_months' and 'incidents' values in the dataframe to integer

ship = ship.round({'s\_months':0,'incidents':0}).astype({'s\_months':int,'incidents':int})

print(ship)

**Question 1: Task(a)(iv) – Output**



**Question 1: Task(a)(v) – Code**

# Q1 Task (a)(v)

# Source: https://stackoverflow.com/questions/34682828/extracting-specific-selected-columns-to-new-dataframe-as-a-copy

# Save incidents as a new dataframe called 'Y'

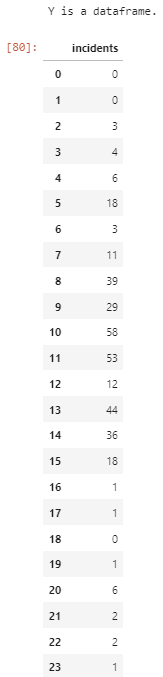
Y = ship[['incidents']]

# To check if Y is a dataframe type

if isinstance(Y, pd.DataFrame): print("Y is a dataframe.", "\n")

Y

**Question 1: Task(a)(v) - Output**



**Question 1(b) - Code**

# Q1 (b)

# (b)(i) Convert dataframe NewShipGroup's 'types', 'c\_years', and 'o\_periods' columns into categorical type variables.

# Source: https://www.datasciencemadesimple.com/convert-column-to-categorical-pandas-python-2/

ship = ship.astype({'types':'category', 'c\_years':'category','o\_periods':'category'})

# To check if 'types', 'c\_years', and 'o\_periods' have been converted into categorical type variables

print("Ship's values' datatypes are as follows: ", "\n", ship.dtypes, "\n")

# (b)(ii) Convert all categorial variables to dummy variables and save result as a pandas DataFrame X

X = (pd.get\_dummies(ship, columns=['types', 'c\_years', 'o\_periods'])).copy()

#To check if X is a dataframe type

if isinstance(X, pd.DataFrame): print("X is a dataframe.", "\n")

# (b)(iii)

# Pull out s\_months from "X" or "ship" dataframe using iloc for log transformation (s\_months in both datafranes are the same)

s\_months\_log = X.iloc[:,:1]

# Carry out a log transformation for aggregated months of service "s\_months", to be attached to both "ship" and "X"

# Source: https://stats.stackexchange.com/questions/402470/how-can-i-use-scaling-and-log-transforming-together

s\_months\_log = np.log(s\_months\_log)

# Insert into dataframe source: https://www.geeksforgeeks.org/python-pandas-dataframe-insert/

# Insert the newly log transformed "s\_months" into the front of "ship" dataframe

ship.insert(5, "log\_s\_months", s\_months\_log)

print("This is the current dataframe for ship with the transformed s\_months:", "\n", ship)

# Drop the "s\_months" and "incidents" columns

# Source: https://pynative.com/pandas-drop-columns/

X = X.drop(columns = ['s\_months','incidents'])

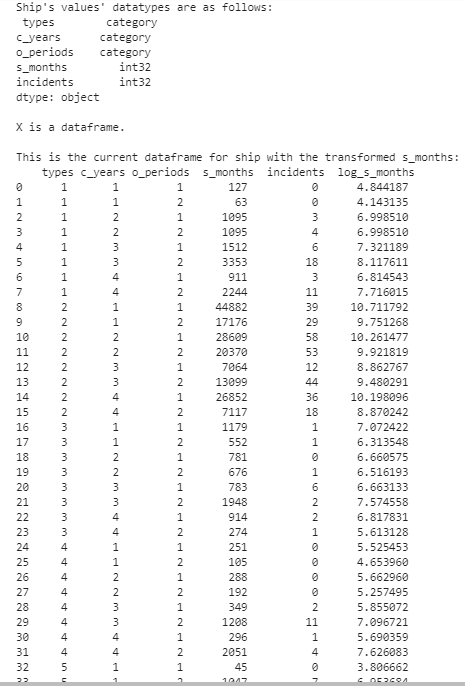
# Insert the newly log transformed "s\_months" into the front of "X" dataframe

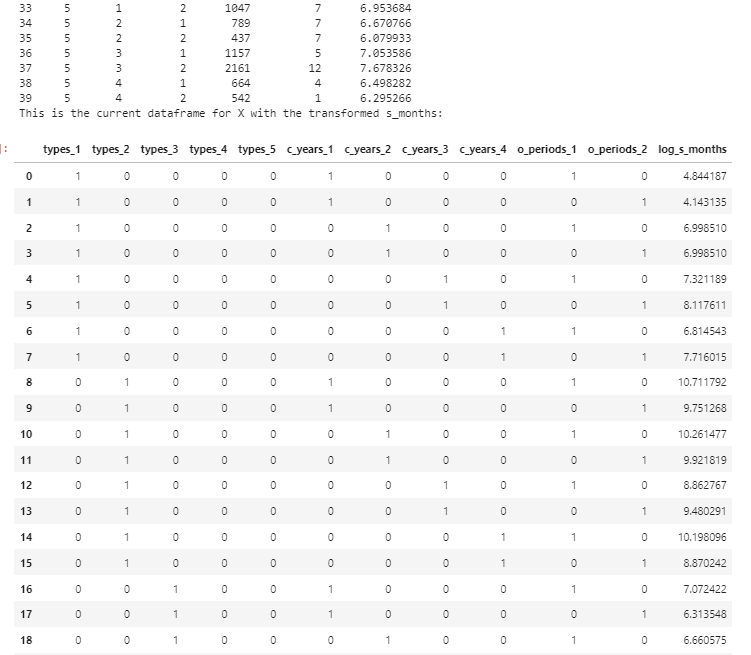
X.insert(11, "log\_s\_months", s\_months\_log)

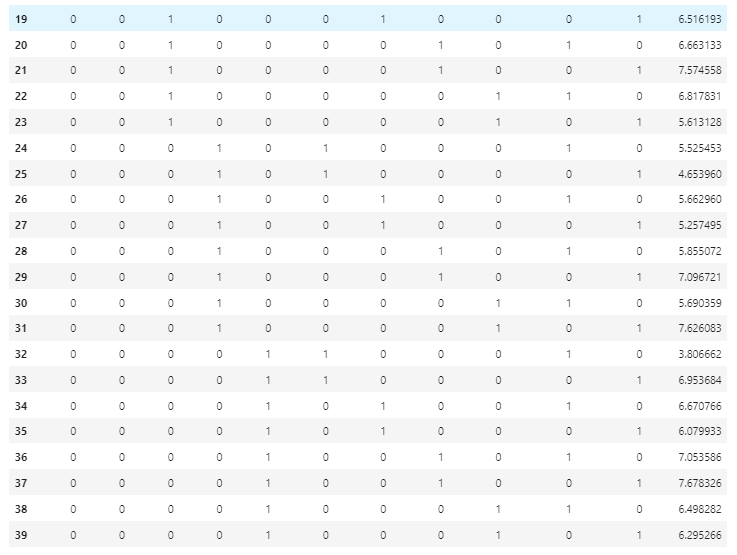
print("This is the current dataframe for X with the transformed s\_months:", "\n")

X

**Question 1(b) – Output**







**Question 1(c):**

There is no need for splitting the dataset into training and testing sets, as there are no underfitting and overfitting instances. According to AskPython (2021), whenever one trains a predictive model, they are doing so by finding a pattern representing the whole dataset while minimizing errors.

However, this leads to two common problems, namely underfitting and overfitting of data, which can be solved by splitting datasets into a training and testing sets (AskPython, 2021).

Underfitting refers to when a model’s low accuracy attributable to the fact it was too simple to be able to track the dataset’s complexities, hence rendering it unable to represent the dataset (AskPython, 2021). This is not the case for this question, as both ship and X have relatively simple datasets with only a few different variables, or dummy variables in X which only varies between 1 and 0 for the presence of data. This allows the model to be able capture the data with more accuracy.

Overfitting on the other hand, refers to when a model is too complex and accurate that it greatly represents the dataset (AskPython, 2021). This is not the case for this question, as some of the NaN values under s\_months and incidents were estimated with the means of their respective variables, therefore reducing the potential possibility for overfitting.

**Question 1(d) - Code**

# Q1 (d)

# We shall now save the prepared DataFrame "ship" as a new csv text file called "ship\_prepared.csv"

# Source: https://www.askpython.com/python-modules/pandas/save-dataframe-as-csv-file

ship.to\_csv('ship\_prepared.csv')

print("The dataframe ship has been exported as 'ship\_prepared.csv' excel folder.")

# Furthermore, we shall also create a database called "ship.db" and export the DataFrame to the database as tables. Write a Python program to carry out these two tasks.

# Create a new database called "ship.db" using sqlite3

conn = sqlite3.connect('ship.db')

# Create a cursor object

cur = conn.cursor()

# Read the prepared ship dataframe from ship\_prepared.csv

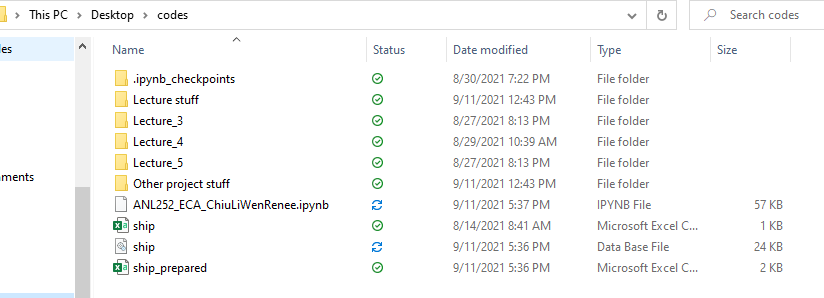
ship\_prepared = pd.read\_csv("ship\_prepared.csv")

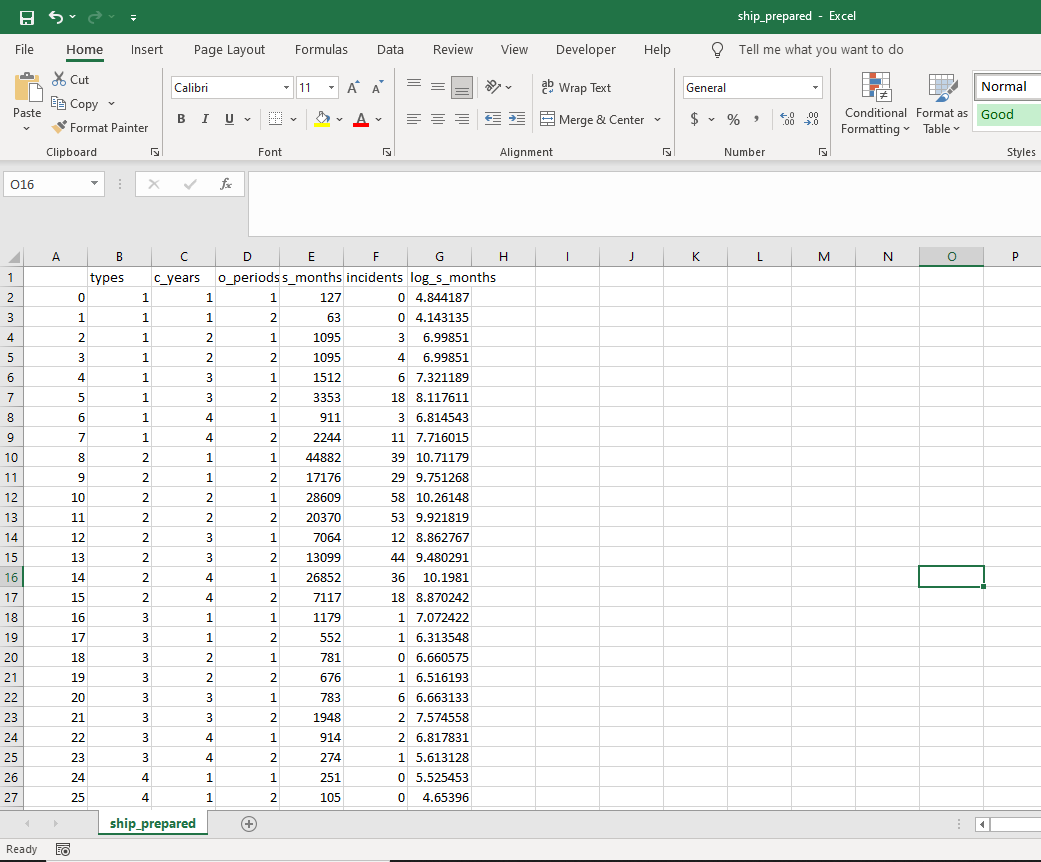
# Push ship dataframe to sqlite3 database table

ship\_prepared.to\_sql("Ship Prepared", conn, if\_exists="replace")

**Question 1(d) - Output**







**Question 2(a):**

According to scikit learn’s official website (2020), the corresponding scikit-learn module for Poisson Regression in this case is the “PoissonRegressor” model, which is also known as “sklearn.linear\_model.PoissonRegressor”.

Discuss the corresponding module:

Before using PoissonRegressor module for regressing linear models, users will have to import the linear model component from sklearn, which can be accomplished using “from sklearn.linear\_model”, or “from sklearn import linear\_model” (scikit learn, 2020). The poisson regression model can then be created by adding “.PoissonRegressor()”, behind the newly imported “linear\_model” (scikit learn, 2020). The PoissonRegressor model also uses the ‘log’ function to be able carry out its dataset predictions.

Estimator:

In PoissonRegressor module, the estimator is a list, set or tuple, which will be cloned for the parameters.

Fit:

According to scikit learn (2020), the fit function of the PoissonRegressor module, usually typed as “fit(X, y, sample\_weight)”, is mainly used for fitting the datasets from the linear model into the poisson regression model. The “X” in the fit function is the training dataset, or in this case the X dataframe from Q1 (b), which must be in the array or sparse matrix format. The “y” in the fit function is the target values to be predicted, or in this case the Y dataframe from Q1 (a)(v), which must be in the one-dimensional array format. The fit function does not return any other output other than itself.

Predict:

The predict function of the PoissonRegressor module, usually typed as “predict(X)”, is mainly used to predict the outcome from the training dataset, which is the X dataframe in this case, using poisson regression (scikit learn, 2020). The predict function returns the predicted y values, that was fitted into the model using the fit function under “y”.

Other parameter functions:

There parameters functions in the PoissonRegressor module, “get\_params(deep=True)” and “set\_params(\*\*params)” (scikit learn, 2020). If deep is set equals to True, “get\_params(deep=True)” retrieves the parameters of the estimator and any subjects that were recognised to be estimators as well. For “set\_params(\*\*params)”, this function allows users to set the parameters of their estimators.

**Question 2(b) – Code**

# Q2 (b)

# Source: https://scikit-learn.org/stable/modules/generated/sklearn.linear\_model.PoissonRegressor.html#examples-using-sklearn-linear-model-poissonregressor

from sklearn import linear\_model

clf = linear\_model.PoissonRegressor()

# Convert X dataset into array or sparse matrix format using .to\_numpy from numpy library

# Source: https://www.marsja.se/how-to-convert-a-pandas-dataframe-to-a-numpy-array/#:~:text=How%20do%20you%20convert%20a%20DataFrame%20to%20an,can%20add%20this%20code%3A%20np\_array%20%3D%20df.to\_numpy%20%28%29.

X\_arr = X.to\_numpy()

# To check the shape of array transformed X. X shld be 40 x 12

print("The shape of array X is: ", X\_arr.shape, "\n")

# Convert Y into a 'one dimensional array' format using .values.flatten from numpy library

# Source: https://datascience.stackexchange.com/questions/18904/how-do-i-convert-a-pandas-dataframe-to-a-1d-array#:~:text=You%20can%20first%20convert%20the%20DataFrame%20to%20NumPy,run.flatten%20%28%29%20to%20collapse%20it%20into%20one%20dimension.

Y\_arr = Y.values.flatten()

# To check the shape of array transformed X. X shld be 40 x 1 (1-D array)

print("The shape of array Y is: ", Y\_arr.shape,"\n")

# Report the parameters of the estimated model

print("Here are the parameters of the model: ")

display(clf.get\_params(deep=True))

# Fit the X and Y datasets into the poisson regression model

clf.fit(X\_arr,Y\_arr)

# Run the predicition function of the regression model to get the predicted values

coefficients = clf.coef\_

# Generate a table or data frame of the predictedvalues with corresponding labels

coefficients\_dataframe = pd.DataFrame(coefficients)

# Rename the columns of coefficients dataframe, according to the column names in dataframe X, after transposing the coefficients dataframe

# Source: https://stackoverflow.com/questions/45264141/convert-array-into-dataframe-in-python

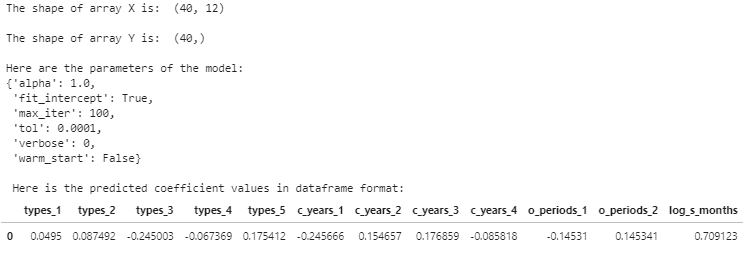
transpose\_coeff = coefficients\_dataframe.transpose()

transpose\_coeff = transpose\_coeff.rename(index=str, columns={0:'types\_1', 1:'types\_2', 2:'types\_3', 3:'types\_4', 4: 'types\_5', 5:'c\_years\_1', 6:'c\_years\_2', 7:'c\_years\_3', 8:'c\_years\_4', 9:'o\_periods\_1', 10:'o\_periods\_2',11:'log\_s\_months'})

print("\n", "Here is the predicted coefficient values in dataframe format: ")

display(transpose\_coeff)

**Question 2(b) - Output**



**Question 2(c) – Code**

# Q2 (c)

# D's formula: D = 2∑ni=1 {Ylog[YE(Y)] − [Y−E(Y)]}

EY = clf.predict(X)

display(EY)

Y = Y.values.flatten()

display(Y)

index = 0

SummationDev = 0

for x in Y:

if x == 0:

Y\_value = Y[index]

EY\_value = EY[index]

SummationDev = SummationDev + ((Y\_value \* 0) - (Y\_value - EY\_value))

index += 1

else:

Y\_value = Y[index]

EY\_value = EY[index]

SummationDev = SummationDev + ((Y\_value \* math.log(Y\_value/EY\_value)) - (Y\_value - EY\_value))

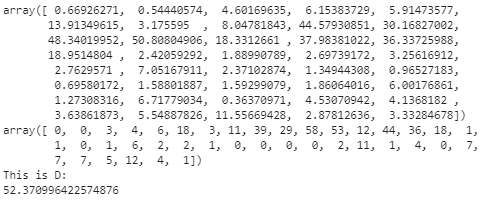
index += 1

D = 2 \* SummationDev

print("This is D: ")

display(D)

**Question 2(c) - Output**



**References**

AskPython. (2021). *How to Split Data into Training and Testing Sets in Python using*

*sklearn?* Retrieved from <https://www.askpython.com/python/examples/split-data-training-and-testing-set>

Scikit learn. (2020). *sklearn.linear\_model.PoissonRegressor.* Retrieved from

<https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.PoissonRegressor.html>

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

Social Sciences.