****

**ANL 252 Python For Data Analytics**

**End-of-Course Assignment**

**July 2021 Presentation**

**Done by:**

**Ong Guo Lun (M1072994)**

**The Script**

"""

Created on Sat Sep 11 12:36:23 2021

@author: alanonggl

"""

#Importing all the libraries required for questions 1 & 2

import numpy as np

import pandas as pd

import sqlite3

from sklearn import linear\_model

from collections import defaultdict

#Display the "ship.csv" and indicating the missing values as NaN

print("\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_1a(i)\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_")

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

print(ship)

#Renaming the requested variable names of the ship dataset

print("\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_1a(ii)\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_")

ship = ship.rename(columns={"T": "types", "A": "c\_years", "P": "o\_periods", "MS": "s\_months", "Y": "incidents"})

print(ship)

#Creating a new object "shipgroup" to keep the resulting table in 1a(iii)

print("\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_1a(iii)\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_")

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

print(shipgroup)

#Replacing missing values in "ship.csv" for "s\_month" & "incidents"

#Using the respective means of other ships' data that have the sames "o\_periods" & "types"

print("\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_1a(iv)\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_")

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

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

print(ship)

#Saving the variable from "incidents" to a DataFrame called "y"

print("\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_1a(v)\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_")

y = pd.DataFrame(ship['incidents'])

print(y)

#Converting to categorical variables for "types", "c\_years" & "o\_periods"

print("\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_1b(i)\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_")

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

print(ship)

#Converting the categorical variables to dummy variables

#Keeping the result in DataFrame named "x"

print("\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_1b(ii)\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_")

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

print(x)

#Scaling down the wide range of values

#Log-transforming the dataframe "x"

#Naming the transformed variable "log\_s\_months"

#Attaching to the dataframe "x" & "ship"

print("\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_1b(iii)\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_")

ship['log\_s\_months'] = np.log(ship['s\_months'])

print(ship)

x['log\_s\_months'] = np.log(x['s\_months'])

print(x)

#1c

#Using the current dataset from "ship.csv", there will not be enough data.

#Not having enough data, the training set will not get recognised enough and become a successful set.

#Similarly, the testing set is unable to check properly as the dataset is not enough.

#Another test instead can be the "k-overlay cross-approval".

#Saving the dataframe "ship" to a new csv "ship\_prepared.csv"

#Creating a database "ship.db"

#Exporting the dataframe to "ship.db" as tables

print("\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_1d\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_")

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

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

ship.to\_sql('ship\_table', conn, if\_exists='replace', index=False)

conn.commit()

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

c = conn.cursor()

c.execute('''

SELECT

\* from ship\_table

''')

shipdb\_df = pd.DataFrame(c.fetchall())

print (shipdb\_df)

#2a

#The corresponding scikit-learn mobile will be the “sklearn\_model”

#The parameters are “max\_iter: int, default=100” which determine the highest number of iterations for this regression to meet, “alphafloat, default = 0.0001” which is to stop overfitting and making regularization stronger via multiplying the constant and “fit\_intercept: bool, default=True” which is to count this regression’s intercept and the default is set to not true, the count will not include the intercept. (scikit-learn developers, 2007 - 2020)

#The Attributes are “intercept\_:float or array of shape (n\_targets,)” which will produce the β0’s intercept and “coef\_: array of shape (n\_features, ) or (n\_targets, n\_features)” which produces the coefficients’ arrangement.

#The methods are “score(X, y[,sample\_weight]) which calculate the deviance’s percentage, “get\_params([deep]) which produces the parameters of the coefficients, “predict(X) which will anticipate via using this given model and “fit(X,y[,sampe\_weight]) to fit into this regression.

#Fit the suggested Poisson regression with dataframes "x" & "y"

#Producing a dataframe to show the coefficients towards the corresponding labels

print("\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_2b\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_")

arr\_x=x.to\_numpy()

arr\_y=y.to\_numpy()

model = linear\_model.PoissonRegressor()

model.fit(x, y)

df\_xy=defaultdict(float)

df\_xy['β0']=model.intercept\_

i=1

for val in model.coef\_:

df\_xy['β'+str(i)]=val

i=i+1

print(df\_xy)

#Calculating devariance with using the score() function

print("\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_2c\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_")

model\_y=model.predict(arr\_x)

D = []

#j is used instead to represent Y in the equation D in this question

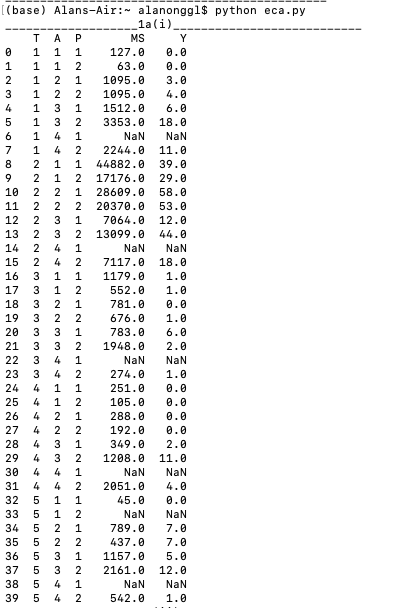
#This is needed as we already use y for question 1 and not to confuse the script

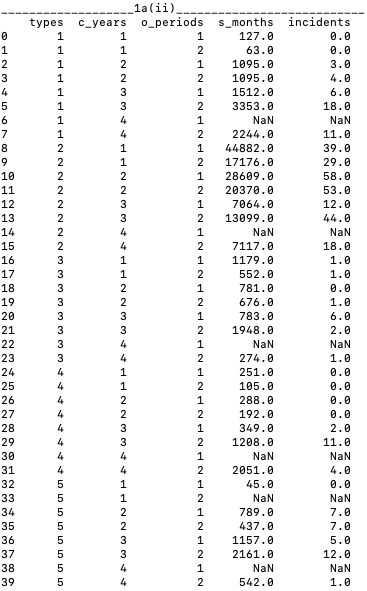
for j in arr\_y:

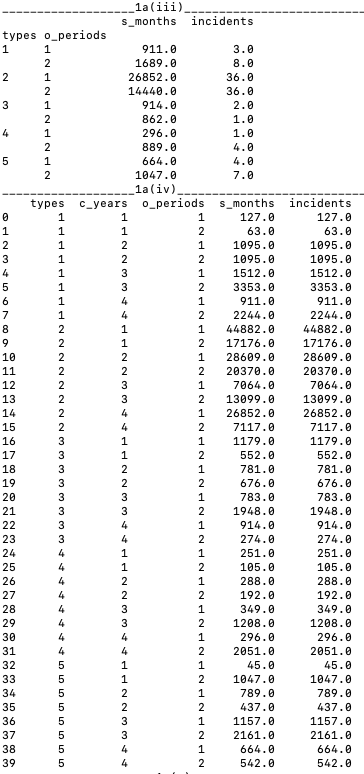
D.append(np.sign(j-np.mean(model\_y)\*(2\*y\*np.log(j/np.mean(model\_y))-(j-np.mean(model\_y)))))

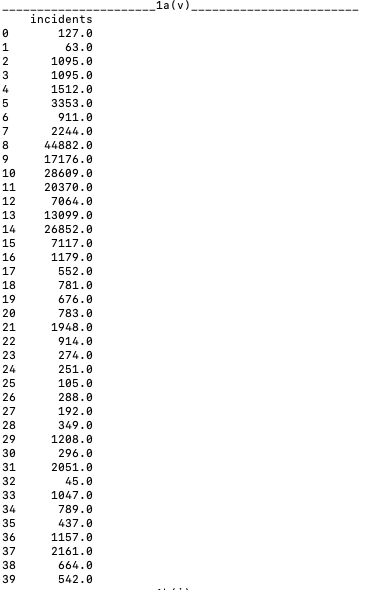
print("The resulting deviance will be ",np.mean(D))

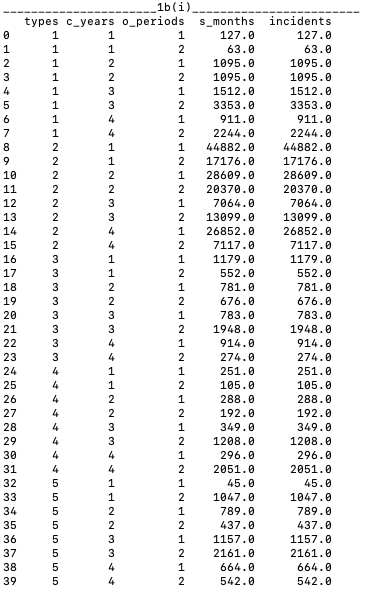
**The Output**

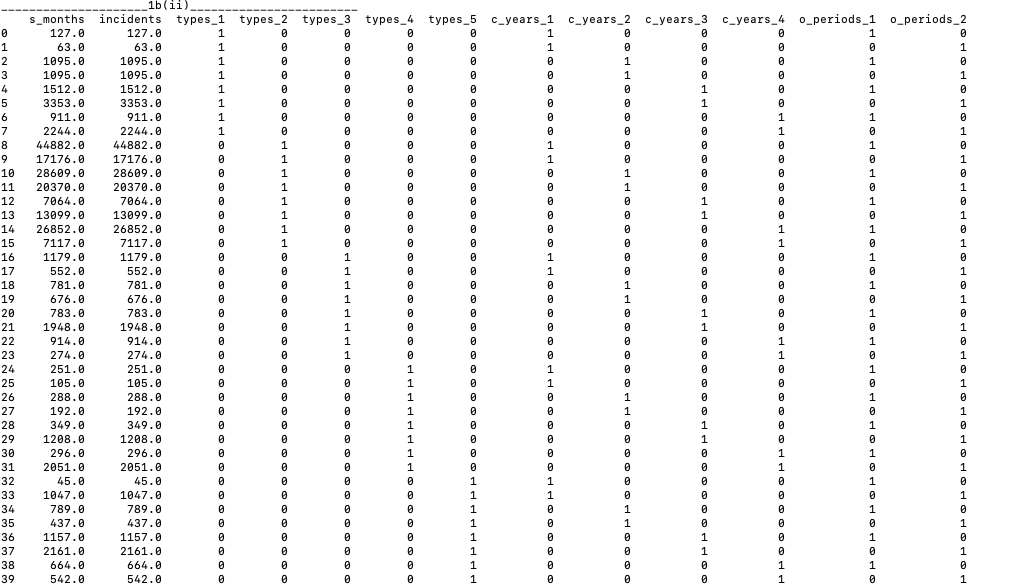
****

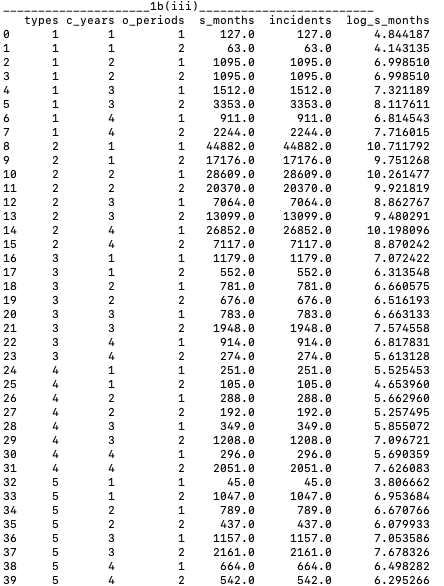
****

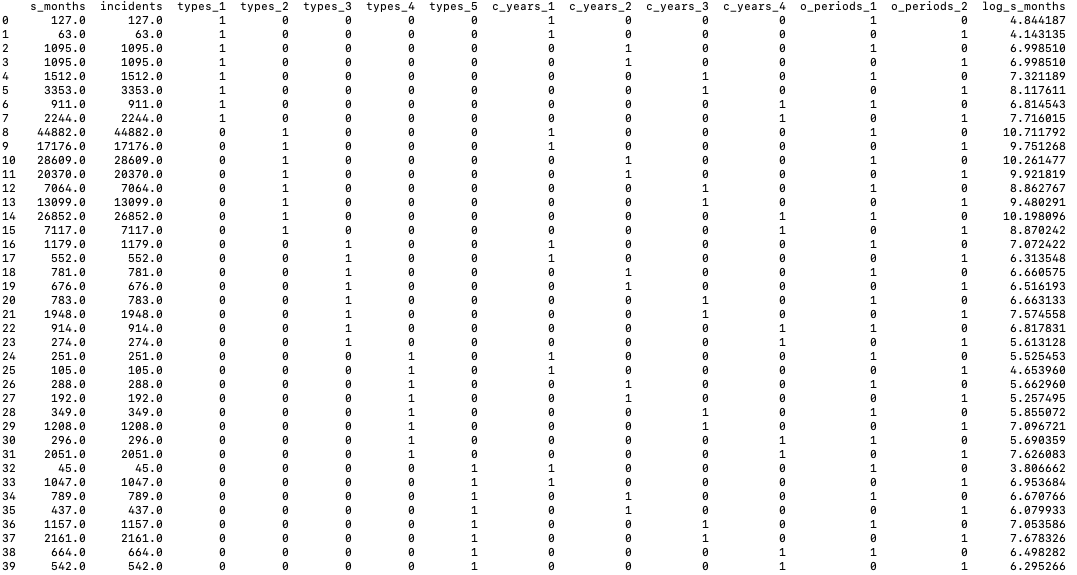
****

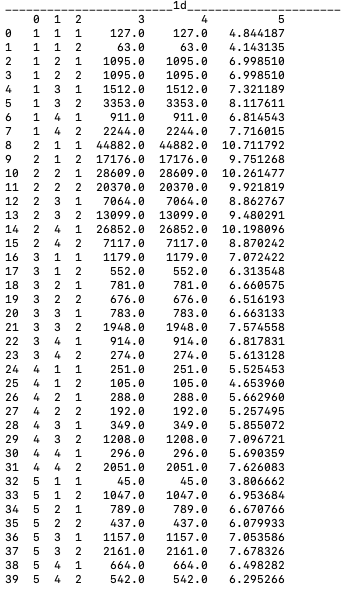
****

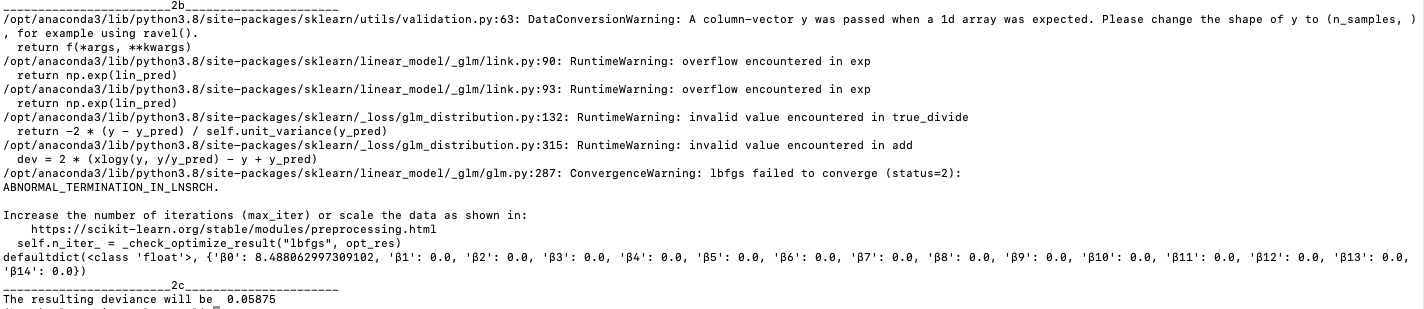
****

****

****

****

****

****

**Reference**

sklearn.linear\_model.LinearRegression (2007 - 2020). Retrieved from <https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LinearRegression.html>

sklearn.linear\_model.LogisticRegression (2007 - 2020). Retrieved from <https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html>

sklearn.linear\_model.Perceptron (2007 - 2020). Retrieved from <https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.Perceptron.html>