

**ANL252**

**PYTHON FOR DATA ANALYTICS**

**End Course Assignment**

Semester 1 Term 1

TG Group: 09

Name: Michael Cheng Lee Beng

SUSS PI Number: B1981656

Date of Submission: 12 September 2021

**Question 1a(i)**

import pandas as pd

import numpy as np

import sqlite3

from sqlite3 import Error

#Reading ship.csv as pandas dataframe ship

ship = pd.read\_csv(‘ship.csv’)

#using to\_numeric function to identify character “.” & converts

#to missing value NaN

ship['MS'] = pd.to\_numeric(ship['MS'], errors='coerce')

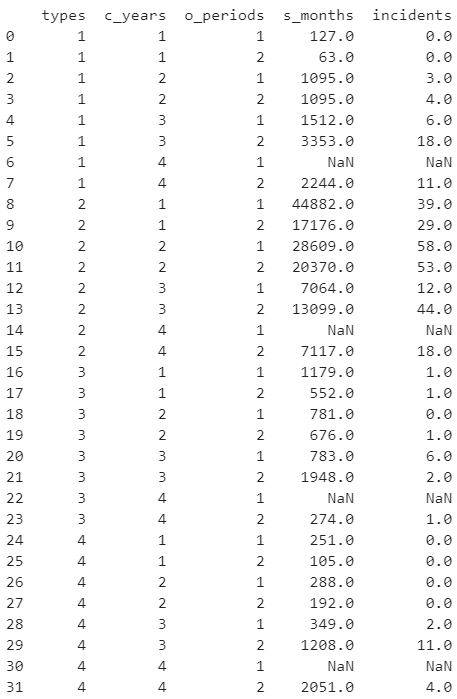
ship['Y'] = pd.to\_numeric(ship['Y'], errors='coerce')

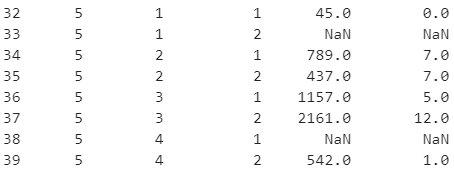
**Question1a(ii)**

# Renaming the columns

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

print(ship)





**Question1a(iii)**

#calculating averages of columns service months and incidents

avg\_service\_months = round(ship['s\_months'].mean())

avg\_incidents = round(ship['incidents'].mean())

print(avg\_service\_months)

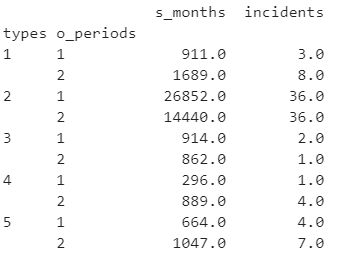
print(avg\_incidents)

|  |  |
| --- | --- |
| Output printed |  |

**Question1a(iii)**

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

print(shipgroup)



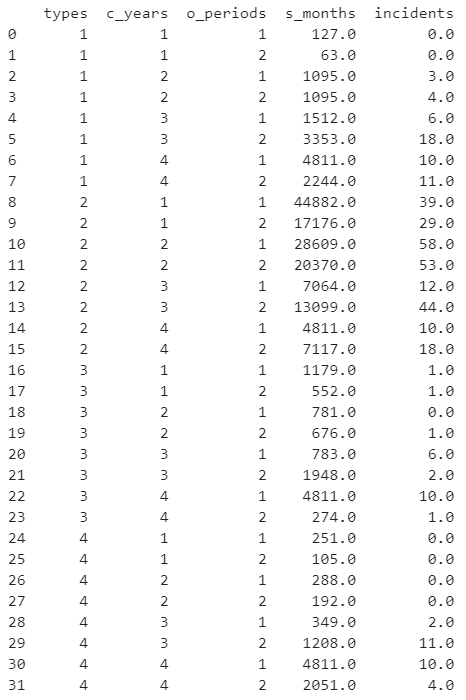
**Question1a(iv)**

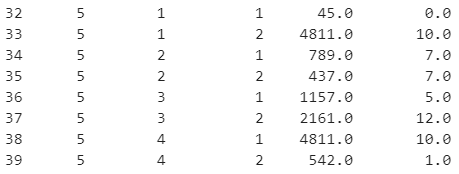
#Using .fillna function to fill NaN values/missing values

ship['s\_months'].fillna(float(avg\_service\_months))

ship['incidents'] = ship['incidents'].fillna(float(avg\_incidents))

print(ship)





**Question1a(v)**

# saving the target variable incidents in Y

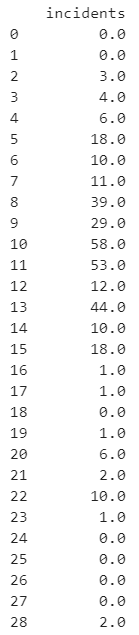
X = ship.iloc[:, :-1]

Y = ship.iloc[:, -1]

# selecting the column ‘incidents’ from ship df & convert to df.

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

print(Y)



**Question 1b(i)**

# 'types', c\_years, o\_periods as categorical variables

ship['types'] = ship['types'].astype('category')

ship['c\_years'] = ship['c\_years'].astype('category')

ship['o\_periods'] = ship['o\_periods'].astype('category')

**Question1b(ii)**

# categorical varables to dummy values

ship['types'] = ship['types'].cat.codes

ship['c\_years'] = ship['c\_years'].cat.codes

ship['o\_periods'] = ship['o\_periods'].cat.codes

print(ship)

**Question 1b(iii)**

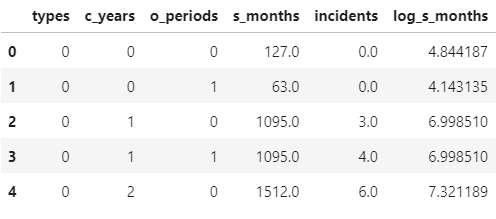
# log transformation of s\_months to log\_s\_months

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

X = ship

X = X.drop(columns=['incidents'])

y = ship['incidents']



**Question 1c**

#If datasets is splitted into train and test sets, there may be insufficient data in the training dataset for the model

#to execute the codes since insufficient data results in the programme being unfamliar with the codes.

#As such, it may be an issue to execute the code with the lack of data.

#With a lack of data, the k-overlay cross approval methodology may be suitable.

**Question 1d**

# Making a database and connecting to it

DB\_path = 'ship.db'

def connect\_to\_db(DB\_path):

sqlite3\_conn = None

try:

sqlite3\_conn = sqlite3.connect(DB\_path)

return sqlite3\_conn

except Error as error\_msg:1+1+1+1 = 4M

print(error\_msg)

if sqlite3\_conn is not None:

sqlite3\_conn.close()

# Creating database table and data from df to database

def insert\_values\_to\_table(t1):

conn = connect\_to\_db(DB\_path)

if conn is not None:

c = conn.cursor()

c.execute('CREATE TABLE IF NOT EXISTS ' + t1 +

'( INTEGER,'

'types INTEGER,'

'c\_years INTEGER,'

'o\_periods INTEGER,'

's\_months DECIMAL,'

'incidents DECIMAL,'

'log\_s\_months DECIMAL)')

ship.to\_sql(name=t1, con=conn, if\_exists='append', index=False)

conn.close()

print('SQL function completed')

else:

print('Unable to connect to database')

insert\_values\_to\_table('imdb\_temp')

|  |  |
| --- | --- |
| Output printed |  |

**Question 2a**

An estimator is an object that fits a model based on inputs and performs calculations that correspond to properties on new data.

#This would mean that an estimator is similar to a regressor. The estimator consists of parameters: alpha, fit\_intercept as well as max\_iter.

alpha is a constant that takes in any integer value and the default value is 1.

fit\_intercept specifies whether a constant are to be added to the predictor and takes in the value of ‘True’ or ‘False’.

max\_iter specifies the max number of iterations, which takes in any integer value.

Fit function accepts input for sample data (x) and accepts argument for labels. It takes in two parameters X and y.

The X is a training vectors and y is the target value. Fit function perform arithmetic such as numerous operations.

Fit function can also estimate attributes out of the input data and

return the fited estimator.

#Additionally, the predict() function is used to predict the target values.

#It only takes in one parameter (eg. X), and it passes the training samples data

#to the function to get the predictions.

#As such, the predicted values for the respect samples of data is returned

**Question 2b**

import numpy as np

X=X.drop(columns=['types','c\_years','o\_periods','s\_months'])



Y\_=Y.to\_numpy()

from sklearn import linear\_model

model = linear\_model.PoissonRegressor()

model.fit(X\_, Y\_)

from collections import defaultdict

table = defaultdict(float)

table['β0'] = model.intercept\_

i = 1

for val in model.coef\_:

table['β' + str(i)] = val

i = i + 1

print(table)

|  |  |
| --- | --- |
| Output printed |  |

**Question 2c**

Y\_prediction = model.predict(X)

Dev = []

# calculating deviance using the formula

for y in Y\_prediction:

if (y == 0):

Dev.append(-(y - (np.mean(Y\_))))

else:

Dev.append(y \* np.log(y / (np.mean(Y\_))) - (y - (np.mean(Y\_))))

print("Deviance is ", 2 \* np.mean(Dev))

|  |  |
| --- | --- |
| Output printed |  |