

**ANL252**

**Python for Data Analytics**

**End-of-Course Assessment**

**July 2021 Presentation**

**Submitted by:**

|  |  |
| --- | --- |
| **Name** | **PI No.** |
| **Fok Jiajun Samuel** | **J2111086** |

**Tutorial Group: ­­­­­­­­­­­­ T09**

**Instructor’s Name: Dr. Munish Kumar**

**Submission Date: 11/09/2021**

# Question 1

(a)

import pandas as pd

import numpy as np

import sklearn

import sqlite3

(i)

ship = pd.read\_csv("ship.csv",na\_values=".") # to indicate "." as missing values

ship.tail(10) # to display last 10 of df



(ii)

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

# rename T to types, A to c\_years, P to o\_periods, MS to s\_months and Y to incidents

ship.tail(10) # to display renaming of variables df

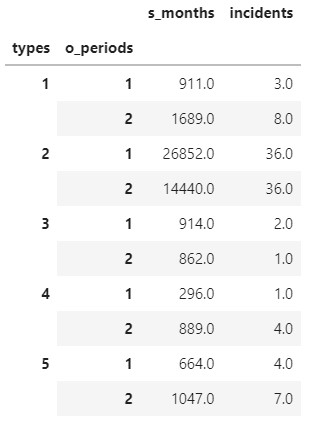


(iii)

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

# finding the mean of service months and incidents for each cross product of types and operation periods

shipgroup # to display df



(iv)

shipindex = ship.set\_index(['types', 'o\_periods']) # set the index to a multiindex to types and o\_periods to match shipgroup

shipindex['s\_months'].fillna(shipgroup['s\_months'], inplace = True)

# replace the NaN values in s\_months with the mean in s\_months from shipgroup

shipindex['incidents'].fillna(shipgroup['incidents'], inplace = True)

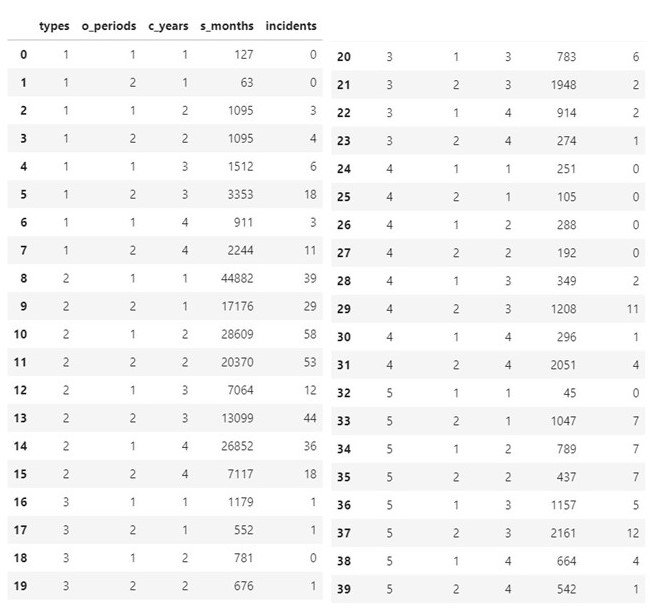
# replace the NaN values in incidents with the mean in incidents from shipgroup

ship = shipindex.reset\_index() # to reset the index

ship['s\_months'] = ship.s\_months.astype('int') # to change float to 'int'

ship['incidents'] = ship.incidents.astype('int') # to change float to 'int'

ship # to display df



(v)

Y = pd.DataFrame(ship['incidents']) # extracting the column incidents to create a new dataframe Y

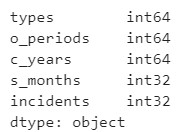
Y # to display df



(b)

(i)

ship.dtypes #to check the type before converting

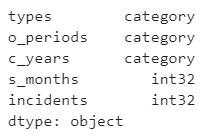


ship['types'] = ship.types.astype('category') #to change a numeric categorical variable (types) to 'category'

ship['o\_periods'] = ship.o\_periods.astype('category') #to change a numeric categorical variable (o\_periods) to 'category'

ship['c\_years'] = ship.c\_years.astype('category') #to change a numeric categorical variable (c\_years) to 'category'

ship.dtypes #to check the type after converting

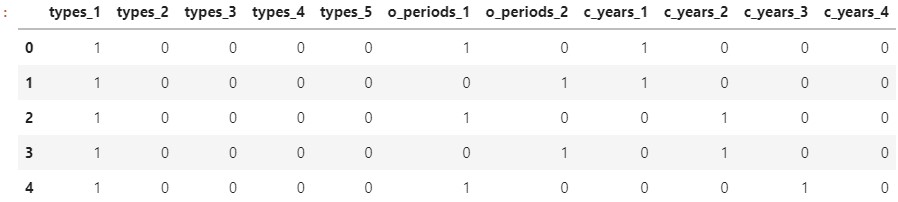


(ii)

X = pd.get\_dummies(ship, columns = ['types', 'o\_periods', 'c\_years']).drop(['s\_months','incidents'], axis = 1,)

#to convert the categorical variables (types, o\_periods and c\_years) to dummy variables.

X.head() # to display.top 5 of df

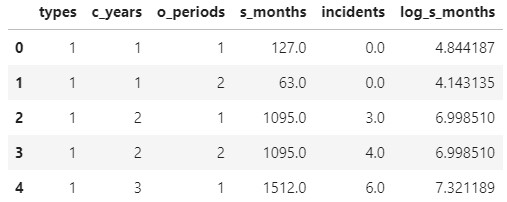


(iii)

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

#using numpy to do log-transformation of s\_months to log\_s\_months and adding to ship

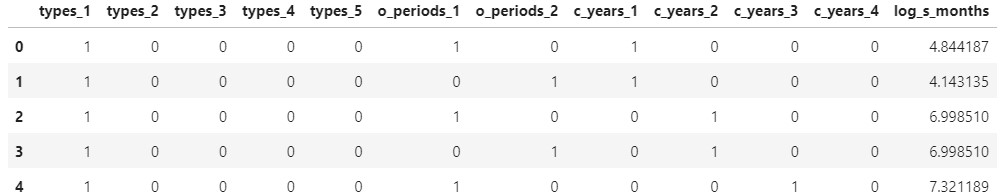
ship.head() # to display top 5 of df



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

#using numpy to do log-transformation of s\_months to log\_s\_months and adding to X

X # to display top 5 of df



(c)

Firstly, the data set is small, there will not be enough data in the training set to learn an effective model if the dataset is split into the train and test sets. There will also not be enough data in the test set to evaluate the model performance. Hence, the estimated performance of the model might be over optimistic or overly pessimistic.

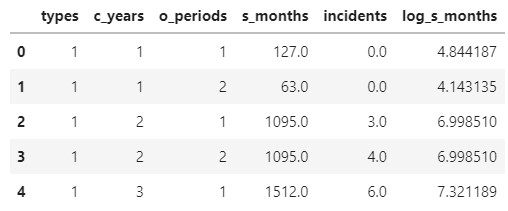
Secondly, the data set is also imbalanced. There is 8 counts of 0 incidents, 6 counts of 1 incident, 3 counts of 2, 4 and 7 incidents each, 2 counts of 3, 6, 11, 12, 18 incidents each and 1 count of 5, 29, 36, 39, 44 53, 58 incidents each. The number of counts of incidents is imbalanced.

(d)

ship.to\_csv('ship\_prepared.csv', index = False) #saving as a new csv file

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

ship\_prepared.head() # to display top 5 of df





%load\_ext sql

%sql sqlite://

#importing sql library

from sqlalchemy import create\_engine

cnx = create\_engine(r'sqlite:////BSFI/Compulsory/ANL252/ECA/ship.db',

echo = False)

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

# attach the data frame to the sql with a name of the table

ship\_prepared.to\_sql('ship\_prepared', con=cnx, index=False)



# Question 2

(a)

The sklearn.linear\_model.PoissonRegressor is a regressor that uses the Generalized Linear Model with a Poisson distribution.

**fit\_intercept: *bool, default=True***, tells sklearn to include β0 which is the intercept to the predictor

**max\_iter: *int, default=100****,* refers to the number of times it takes for the solvers to converge.

**fit (X, y, [sample\_weight]),** fits the x values into the estimator and y fits the count data to the estimator.

**predict(X),** gets the predicted values from the estimator with the matrix X. X refers to the samples.

(b)

# following the instruction in the official website

from sklearn import linear\_model

feature\_cols = ['types\_1', 'types\_2', 'types\_3', 'types\_4', 'types\_5', 'o\_periods\_1', 'o\_periods\_2', 'c\_years\_1', 'c\_years\_2', 'c\_years\_3', 'c\_years\_4', 'log\_s\_months']

# this will be the corresponding labels

Y = Y.values.ravel() # reshape the df

clf = linear\_model.PoissonRegressor()

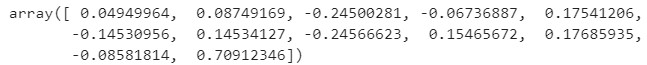
clf.fit(X, Y)



clf.intercept\_ # finding the intercept

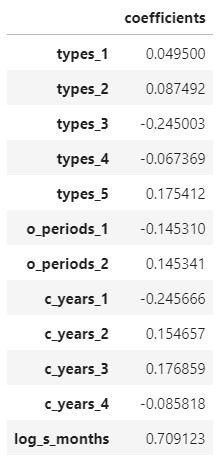


clf.coef\_ # finding the coefficients



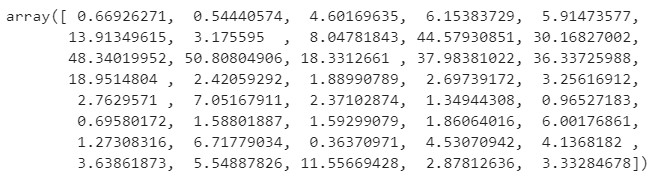
coef = pd.DataFrame(data = clf.coef\_, index = feature\_cols, columns = ['coefficients']) # to generate a dataframe to present the coefficients with the corresponding labels

coef # to display the coefficients



(c)

clf.predict(X)



# I split the formula into 2 parts

def first\_part(i):

result = (Y[i]\*np.log(Y[i]/Y\_pred[i]))

return result

def second\_part(i):

result = (-(Y[i]-(Y\_pred[i])))

return result

Y\_pred= clf.predict(X) # to predict the expected Y values

D=[] # defining a list

l=len(Y\_pred)

for i in range(l):

if(Y[i]==0):

D.append(second\_part(i)) # for Y = 0, as per question, the first part can be taken as 0

else:

D.append(first\_part(i)+second\_part(i)) # applying given formula for deviance

Deviance = 2\*(sum(D)) # as per the formula, taking the sum of D and multiplying by 2

print("The deviance is ", (Deviance).round(4))



# Appendix

