

**ANL 252**

**Python for Data Analytics**

**End-of-Course Assessment**

**July Semester 2021**

**Submitted by:**

|  |  |
| --- | --- |
| **Name** | **PI No.** |
| **Yeo Ker Woon, Jan** | **E1881560** |

**Tutorial Group: T09**

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

**Submission Date: 07/09/2021**

Question 1

a)

1. #Import of numpy and pandas

#Read in the ship.csv

import numpy as np

import pandas as pd

missing\_values = ['.']

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

ship

Output:

A picture containing table

Description automatically generated

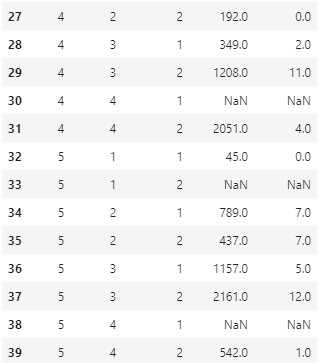
Calendar

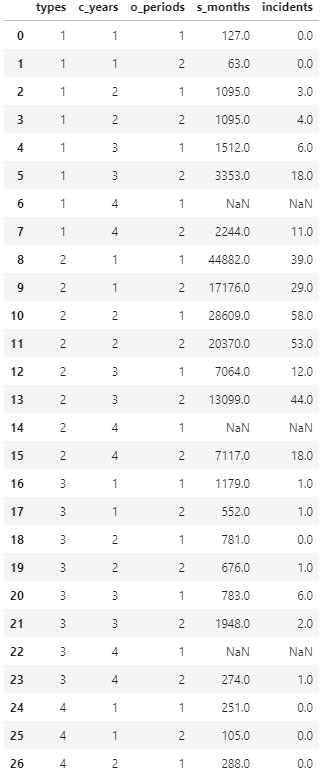
Description automatically generated with medium confidence

1. #Rename of variables names of the ship data set

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

ship

Output:



1. #Computation of the average for the cross-products of every category

shipgroup = pd.DataFrame()

for i in range(0, 5):

df1 = ship[ship['types'] == i + 1]

df1 = round(df1.groupby('o\_periods').mean())

shipgroup = shipgroup.append(df1, ignore\_index = False)

shipgroup['o\_periods'] = shipgroup.index

shipgroup

Output:

Table

Description automatically generated

1. #Replace the missing values with the mean

for i in range(len(ship)):

val = ship.loc[i, 's\_months']

val1 = ship.loc[i, 'incidents']

if np.isnan(val):

replace = shipgroup[(shipgroup['types'] == ship.loc[i, 'types']) & (shipgroup['o\_periods'] == ship.loc[i, 'o\_periods'])]ship.at[i, 's\_months'] = replace['s\_months']

if np.isnan(val1):

replace = shipgroup[(shipgroup['types'] == ship.loc[i, 'types']) & (shipgroup['o\_periods'] == ship.loc[i, 'o\_periods'])]ship.at[i, 'incidents'] = replace['incidents']

ship

Output:

Table

Description automatically generatedTable

Description automatically generated

1. # Construct a Python program to save the target variable "incidents" in a pandas DataFrame named "Y"

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

Y = ship.iloc[:, -1]

Y

Output:

DataFrame Y

Table

Description automatically generated

b)

1. #Categorical values

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

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

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

ship

Output:

Table

Description automatically generatedTable

Description automatically generated

1. #assign dummy values

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

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

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

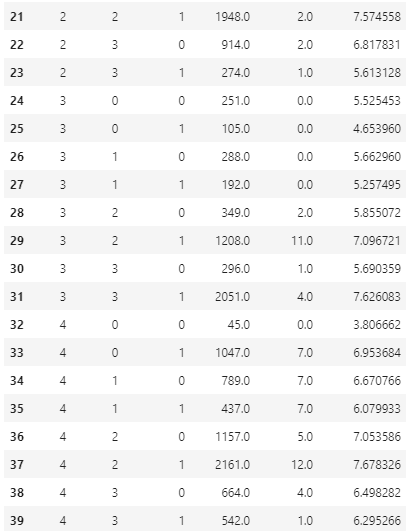
X = pd.DataFrame(ship)

X

Output:

Table

Description automatically generatedDataFrame X



1. #log transformation

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

X = ship

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

y = ship['incidents']

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

ship.head()

Output:

Table

Description automatically generated

1. Based on the dataset, there are only 39 data in the Data Frame, this is a relatively little amount of data for both training and testing. When the dataset is split into training and testing data sets, we will not have enough data in the training dataset for the model to learn an effective mapping which could cause the model to be inaccurate (Brownlee, 2020). Additionally, there will be insufficient data for the testing set to evaluate the model.

There is also much deviation in the data set so splitting the small sample model further into training and testing would be infective. This is because the estimated performance could be on the opposite side on the spectrum or either overly optimistic or pessimistic. Hence, using the entire dataset for the training would make the training set more accurate in determining the results.

import pandas as pd

import pyodbc

# Import CSV

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

df = pd.DataFrame(data, columns= ["types","c\_years", "o\_periods", "s\_months", "incidents", "log\_s\_months"])

# Connect to SQL Server

conn = pyodbc.connect('Driver={SQL Server};'

'Server=JAN\SQLEXPRESS;'

'Database=TestDB;'

'Trusted\_Connection=yes;')

cursor = conn.cursor()

# Create Table

cursor.execute('CREATE TABLE people\_info (types int, c\_years int, o\_periods int, s\_months float, incidents float, log\_s\_months float)')

# Insert DataFrame to Table

for row in df.itertuples():

cursor.execute('''

INSERT INTO TestDB.dbo.ships (types, c\_years, o\_periods, s\_months, incidents, log\_s\_months)

VALUES (?,?,?,?,?)

''',

row.types,

row.c\_years,

row.o\_periods,

row.s\_months,

incidents,

log\_s\_months

)

conn.commit()

Output:



Table

Description automatically generated

Question 2

1. The linear model named PoissonRegressor has multiple functions which we can use to perform different tasks.

The following functionalities are:

1. estimator.fit(X, Y, sample\_weight)

Where X and Y are train and test dataframes. It is used to fit a generalized linear model.

This is used to estimate.fit the model based on its training and testing set while inferring on the data. Depending on the nature, fit can also accept keywords arguments. For the attributes that was estimated from data must end with an underscore.

1. get\_params([deep])

This is used to get parameters for the estimator.

1. Predit(X)

This is used to predict the test data where X is the test date.

1. score(X,Y,[sample\_weight]

This is use to compute D^2, the percentage of deviance. Where X and Y are test and train data.

1. set\_params(\*\*params)

This is used to set the values of the params of the parameters.

1. #Fit Poisson regression

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

X\_ = X.to\_numpy()

Y\_ = Y.to\_numpy()

from sklearn import linear\_model

model = linear\_model.PoissonRegressor()

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

model.coef\_

model.intercept

frame = pd.DataFrame()

frame['Coefficients']= model.coef\_

frame['Intercept']= model.intercept\_

frame

Output:

A picture containing graphical user interface

Description automatically generated

1. #Import defaultdict and calculate Beta

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:

#Calculation of the deviance

Y\_pred = model.predict(X)

D = []

for y in Y\_pred:

if(y == 0):

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

else:

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

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

Output:

****

**Reference**

Brownlee, J. (2020, August 26). Train-Test split for Evaluating machine learning algorithms. Machine Learning Mastery. <https://machinelearningmastery.com/train-test-split-for-evaluating-machine-learning-algorithms/>.

Developing scikit-learn estimators¶. scikit. (n.d.). <https://scikit-learn.org/stable/developers/develop.html>.