Logo, company name

Description automatically generated

|  |  |
| --- | --- |
| Course: | ANL252 |
| Assignment: | ECA |
| Name: | Tan Jing Jie |
| PI Number: | Q1882541 |
| Tutorial Group: | T09 |
| Submission Date: | 12 Sep 2021 |

**Question 1**

a)

i)

#Import of pandas and numpy into python

import pandas as pd

import numpy as np

#import and read ship.csv into python

#na\_values ="." is to read all "." in the CSV file as missing values in the dataframe

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

ship.head()

Output:

Background pattern

Description automatically generated with low confidence

ii)

# rename of columns into actual name

ship.rename(columns = {"T" : "types",

"A":"c\_years",

"P" : "o\_periods",

"MS" : "s\_months",

"Y" : "incidents"

}, inplace="True")

# printing of dataframe head to verify the change

ship.head(2)

Output:



iii)

# getting the mean of s\_months and incidents column grouping by types and periods

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

shipgroup

Output:

A picture containing table

Description automatically generated

iv)

#replacing missing values with mean according to the type and period

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

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

ship.head(8)

Output:

A picture containing graphical user interface

Description automatically generated

v)

# saving incidents variable as a separate dataframe

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

Y.head(8)

Output:

Background pattern

Description automatically generated with low confidence

b)

i)

#changing of variable types for types, c\_years and o\_periods to category

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

ship.dtypes

Output:



ii)

#making a copy of ship dataframe

X = ship.copy()

#converting all categorical variables to dummy variables

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

#dropping of columns that are to categorical, leaving X with only categorical datas

X = X.drop(["s\_months", "incidents"], axis = 1)

X

Output:

Table

Description automatically generated

iii)

#transforming s\_month values with the np.log function

log\_s\_months = pd.DataFrame(ship["s\_months"].transform(np.log))

#changing variable name to log\_s\_months with rename function.

log\_s\_months.rename(columns = {"s\_months" : "log\_s\_months"}, inplace = True)

log\_s\_months

Output:

A picture containing background pattern

Description automatically generated

#merging dataframe X with log\_s\_months.

X = X.merge(log\_s\_months, right\_index = True, left\_index = True)

X

Output:

Table

Description automatically generated

# merging ship dataframe with log\_s\_months dataframe

ship = ship.merge(log\_s\_months, right\_index = True, left\_index = True)

ship

Output:

Table

Description automatically generated

c)

Based on the DataFrame provided, there is only 40 observations given. In the train-test technique, a portion of the DataFrame observations are used for the training which is used to determine the predictive model. Once the predictive model is set, the test set is used to determine the accuracy of the predictive model. As the sample size given was only 40, any splitting of data will result in the decrease in the number of observations used in the training of the predictive model. This would compromise on the accuracy of the predictive model if the number of samples used in the training set is further reduced due to the allocation of a portion of the sample to the testing set as there will not be sufficient data in the training set to map the input to the output and similarly to evaluate the model using the test set (Browniee, 2020). Since the dataset has a small sample, the number of data use to validate the usefulness of the predictive model will be meaningless as it will be insufficient.

Besides the limited number of observations in the Dataframe, the presence of missing values may result in the inaccuracy of the training and test technique. In the dataset provided, there are 6 missing values in the aggregated variable and the incident variable. Although the mean is used to replace the these missing data in the dataset, the cleaned data will not be able to provide an accurate training and testing of the model. Essentially, the number of useful data is 26 observations and this further reduce the competency of the model. Hence, if the dataset is further split into training and testing, the model generated in the training will not be accurate and when the model is using for the training set for cross validation, it will not be well validated due to the lack of dataset used for the test, making it meaningless.

d)

#export of ship dataframe to csv file name ship\_prepared.csv

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

# import of sqlalchemy create\_engine and sqlite3

from sqlalchemy import create\_engine

import sqlite3

#creating a new data base

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

#export ship dataframe into the ship.db dataframe

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

#create an engine in order to read the sql

engine = create\_engine('sqlite:///ship.db')

Screenshot of files created: ship\_prepared.csv & ship.db

Graphical user interface, application

Description automatically generated

#read of dataframe to verify that the data is exported to the database

pd. read\_sql('ship',engine).head(5)

Output:

Graphical user interface, application

Description automatically generated

**Question 2**

a) Poisson regression is part of the generalised linear models. In the generalised linear model, there are 3 underlying assumptions. The first assumption is that Y variable is an independent observation. The second assumption states that the each observation of the value of Y is from an exponential distribution. The third assumption is that the mean is not necessary linear with response. The generalised linear model allows all types of data.

In Poisson Regression, the distribution of y can be predicted by incorporating the log-linked function into the x variables and a Poisson regression is used if the predicted value of y is a count data or a positive frequencies distribution.

In order to use Poisson regression, linear model has to be imported into python via from sklearn import linear\_model. This code will allow us to use the module.

The estimators includes alpha which is a constant that determines the regularisation strength of the model where the default value is 1.

The fit\_intercept will determine if the constant should be included into the linear predictor, where the default value is True.

As for the max\_iter which is the maximal number of iteration for the solver where the default value is 100

The tol is the stopping criteria where the default value is 0.0001.

The fit function is used to fit or train a model by inputting the variables X and Y. The parameters include X variables as the training data in the form of array or dataframe, y variables as the target values in the form of array or dataframe, and the sample weight to define the sample size. The syntax is fit(X,Y). This will return an instance of self to the user.

The prediction functions is to predict the target value using x variables. The parameters include data of x variables in the form of an array or dataframe with a syntax of predict(X). This will return an array of predicted y values based on the inputs in x in a dataframe format.

b)

#import of sklearn linear model poissonregressor into python

from sklearn.linear\_model import PoissonRegressor

#running a poisson regression with X and y dataframes with the default settings

PR = PoissonRegressor(alpha= 1.0, fit\_intercept= True, max\_iter= 100, tol=0.0001, warm\_start= False, verbose= 0)

#using the fit function to fit a poisson regression x and y

PR.fit(X,Y, sample\_weight = None)

Output:



#creating labels for each coefficient and converting it into dataframe

index = ["𝛽1","𝛽2","𝛽3","𝛽4","𝛽5","𝛽6","𝛽7","𝛽8","𝛽9","𝛽10","𝛽11","𝛽12”]

index = pd.DataFrame(index, columns = [" "])

#getting the coefficient and put it in a dataframe

coefficient = pd.DataFrame(PR.coef\_, columns = ["coefficient"])

#merge both index and coefficient data frame together

coef\_index = index.merge(coefficient, right\_index = True, left\_index = True)

#setting index as the index and printing the table.

coef\_index.set\_index(" ")

Output:

Background pattern

Description automatically generated with low confidence

c)

#getting the expected value of Y using PR.predict(X)

expected\_y = PR.predict(X)

#converting DataFrame Y to array

Y = Y['incidents'].to\_numpy()

#program to compute D using for loop to run each iterations

#computing the sum in the equations

sum = 0

for i in range(len(Y)):

if Y[i] == 0:

sum = sum + (0-(Y[i]-expected\_y[i]))

else:

sum = sum + ((Y[i]\*(np.log(Y[i]/expected\_y[i])))-(Y[i]-expected\_y[i]))

#multiplying the sum with 2 to get the value of D

D = 2 \* sum

print(D)

Output:

Text

Description automatically generated

Reference

Browniee, J. (2020). Train-Test Split for Evaluating Machine Learning Algorithms. Retrieved from: https://machinelearningmastery.com/train-test-split-for-evaluating-machine-learning-algorithms/

Scikit Learn (n.d.). Sklearn.Linear\_model.PoissonRegressor. Retrieved from: <https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.PoissonRegressor.html#sklearn.linear_model.PoissonRegressor>

Appendix A: Files

Link: <https://www.dropbox.com/sh/0p68g9jpv0w6a5h/AAA_AKEhTJSiXQVNtEk2NWmEa?dl=0>