A picture containing logo

Description automatically generated

ANL252

Python for Data Analysis

End-of-Course Assessment

July 2021 Presentation

|  |  |
| --- | --- |
| **Name** | **PI No.** |
| **Neo Hui Ying Joey** | **B1882288** |

**Submission Date: 13 Sept 2021, 1200hrs**

**Question 1a**

1. # Import pandas and numpy

import pandas as pd

import numpy as np

# Indicate and declare “.” as missing values

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

display(ship)

Calendar

Description automatically generated

*Figure 1: Q1a(i)*

1. # Rename the variable names of the dataset

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

display(ship)

Table, calendar

Description automatically generated

*Figure 2: Q1a(ii)*

1. # Compute the average service months and the average number of incidents for the cross-products of every category in types and operation periods, rounded to the nearest integers

# Store the resulting table to “shipgroup”

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

display(shipgroup)

Table

Description automatically generated

*Figure 3: Q1a(iii)*

1. # Replace the missing values in the variable “s\_months” and “incidents” by the respective means of the other ships that share the same type and the same operation period

# Set “types” and “o\_periods” as indices, and put “inplace = True” to modify the existing DataFrame

ship.set\_index(["types","o\_periods"],inplace = True)

# Use .fillna() to fill in the missing values in “s\_months” and “incidents”

ship["s\_months"].fillna(value = shipgroup["s\_months"],inplace = True)

ship["incidents"].fillna(value = shipgroup["incidents"],inplace = True)

display(ship)

Table

Description automatically generated

*Figure 4: Q1a(iv)*

1. # Save “incidents” in a pandas DataFrame named “Y”

ship.reset\_index(inplace = True)

Y = ship[["incidents"]]

display(Y)



*Figure 5: Q1a(v)*

**Question 1b**

1. # Perform data conversion so that variables can be recognized 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")

ship.dtypes

A picture containing text

Description automatically generated

*Figure 6: Q1b(i)*

1. # Convert all categorical variables into dummy variables and save result as a pandas DataFrame named “X”

X = pd.get\_dummies(ship)

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

display(X)

Table

Description automatically generated

*Figure 7: Question 1b(ii)*

1. # Perform a log-transformation of “s\_months” and named the transformed variable “log\_s\_months”

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

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

display(ship)

display(X)

Table

Description automatically generated

*Figure 8a: Q1b(iii)*

Table

Description automatically generated

*Figure 8b: Question 1b(iii)*

**Question 1c**

A train-test split is appropriate when a sufficiently large dataset is available. This means that there are enough records in both the training and testing datasets to give a relatively accurate prediction of the model. The training dataset is used to construct the model while the testing dataset is used to evaluate the model. In a train-test split in sklearn, the default ratio of training to testing data is 0.75:0.25. In this case, this split is not appropriate due to the small dataset available. The number of training data is not enough for the model to effectively map out the inputs to outputs and the number of testing data is insufficient to evaluate the performance of the model. This will hence give an inaccurate prediction of the model. Therefore, all the data available should be used as training data as the samples in the training dataset will be able to highlight the important characteristics of the dataset hence improving the accuracy of the model.

**Question 1d**

# Save the prepared DataFrame “ship” as “ship\_prepared.csv”

import csv

ship.to\_csv("ship\_prepared.csv",index = False)

# Create a database called “ship.db” and export the DataFrame to the database as tables

import sqlite3

connection = sqlite3.connect("ship.db")

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

ship\_prepared.to\_sql("ship\_prepared",connection,if\_exists = "replace")

**Question 2a**

The corresponding scikit-learn module is the linear\_model module. In this case, since Poisson regression models are required, import the estimator PoissonRegressor from the linear\_model module. The estimator is the statistical method in the module to form a linear model. The fit function is to form a best-fit linear model into the DataFrames and the predict function is to determine the value of the dependent variable given the value of the independent variable based on the best-fit linear model. The parameters would be the conditions of the Poisson regression model have to satisfy.

**Question 2b**

# Import sklearn

import sklearn

# Import linear\_model module

from sklearn import linear\_model

clf = linear\_model.PoissonRegressor()

# Fit a Poission regression in DataFrames X and Y

clf.fit(X,Y)

# Get parameters of estimator

clf.get\_params()

# Find the coefficients

coef = clf.coef\_

# Generate a DataFrame to present the coefficients with the corresponding labels

Variable = np.array(X.columns.values)

PR\_model = pd.DataFrame({"Variable":Variable,"Coefficient":coef})

display(PR\_model)

Table

Description automatically generated

*Figure 9: Q2b*

**Question 2c**

# Return an array of Y values and their corresponding estimated Y values

Y1 = Y.to\_numpy()

Estimated\_Y = clf.predict(X)

# Create a list to store the values of the array

list = []

for y,ey in zip(Y1,Estimated\_Y):

if(y==0):

list.append(0 - (y - ey))

else:

list.append(y\*np.log10(y/ey) - (y - ey))

# Compute D

D = 2\*sum(list)

print(f"The value of D is {D}.")



*Figure 10: Q2c*