

**ANL252**

**Python for Data Analytics**

**End-of-Course Assignment**

PI No. Y2270831

Shaiful Bin Jafar

6th March 2023

Question 1

According to the output of the data, ‘Claim\_ID’, ‘Actual’ and “Terms” have 5, 1677 and 7 missing data respectively. Syntax can be found in Appendix.

Question 2

Missing values found in ‘Claim\_ID’ and ‘Terms’ will be ignored as it is not relevant nor will it affect the insights that will be provided. As for missing values in “Actual”, it will be treated after providing insights of Question 4 as unfiltered data is required to produce them.

Question 3

With pd.to\_datetime function, it will convert the dates to a pandas datetime object. With this new series of datetime object, the date format can be specified. A column created to calculate the difference in days between ‘Actual’ and ‘Planned’.

An object was identified in column ‘Amount’ which is ‘1762.OO’. This was replaced with ‘1762.00’ value and the column ‘Amount’ was converted to type float.

Question 4

Chart, pie chart

Description automatically generated

Based on the above pie chart, 93.4% of the claim amount has been paid by the insurance company.

Chart, box and whisker chart

Description automatically generated

We can infer that the means of claim amount in 2021 and 2022 is almost similar with mean amount 2675.63 and 2656.40 respectively. In year 2021 and 2022, 25th percentile and 75th percentile seems to be similar - 475.64 and 4229.64 in 2021 and 508.06 and 4138.24 in 2022 respectively.

Chart, scatter chart

Description automatically generated

From scatter plot above, the past claims are mostly settled with little to no delays. We can also observe that the delays in settling the claims are no more than 150 days (excluding 2 outliers with delays of 250 days)

Question 5

Firstly, import relevant libraries such as LinearRegression, preprocessing, StandardScaler, train\_test\_split to perform the linear regression model. Focusing on columns ‘Amount’ and ‘Date\_diff’ which was calculated earlier in the script (‘Actual’ – ‘Planned), cleaning the data is required by using dropna method to remove no data in column ‘Actual’. Before scaling the dataset, we will perform data normalization such that the mean and unit variance is 0. Afterwhich, scaling the data through preprocessing function. The scaled data will be used later on in train\_test\_split. Then, using a polynomial fit with higher degree will get us better fit and in this case, degree = 2.

Train\_test\_split where we separate the data into training and testing datasets. With this trained sets, we will perform linear regression model. Afterwards, we will do some predictions on the test set. We can also gather both predicted value and original value to determine any error in the prediction. This is done through mean\_squared\_error function.

Question 6

The linear regression equation is y = (0.04244516)\*X + -0.006544666175828274. We can observe that for every increase of $1 in amount, there will be an increase of 0.042 delay in days in processing the claim.

Appendix

Q1.

import pandas as pd

import numpy as py

from datetime import datetime

import matplotlib.pyplot as plt

# reading the ECA data and identifying missing values and declaring 'Unkn' and '???' as missing values

claim\_data = pd.read\_csv("ECA.csv", na\_values = ("Unkn", "???"), na\_filter = True)

# locating missing data by adding up all the missing values in a column

claim\_data.isnull().sum(axis = 0)

Q2.

# We will ignore the missing values in the Claim\_ID and Terms as it will not affect our analysis

# As for the missing values in Actual, we will only treat the missing values after

# we have provided 3 observations that is required in Question 4 as we will need the unfiltered data.

Q3.

# changing and standardise date format for the dates

claim\_data['Actual'] = pd.to\_datetime(claim\_data['Actual'], format='%d/%m/%Y %H:%M')

claim\_data['Actual'] = claim\_data['Actual'].dt.strftime('%Y-%m-%d')

claim\_data['Created'] = pd.to\_datetime(claim\_data['Created'], format='%Y%m%d')

claim\_data['Created'] = claim\_data['Created'].dt.strftime('%Y-%m-%d')

claim\_data['Planned'] = pd.to\_datetime(claim\_data['Planned'], format='%d/%m/%Y')

claim\_data['Planned'] = claim\_data['Planned'].dt.strftime('%Y-%m-%d')

claim\_data

claim\_data['Actual'] = pd.to\_datetime(claim\_data['Actual'])

claim\_data['Planned'] = pd.to\_datetime(claim\_data['Planned'])

claim\_data['Date\_diff'] = (claim\_data['Actual'] - claim\_data['Planned']).dt.days

claim\_data

# discovered an object in column 'Amount' and replaced the value to show as float

print(claim\_data['Amount'].dtype)

claim\_data.loc[3698, 'Amount'] = 1762.00

claim\_data['Amount'] = claim\_data['Amount'].astype(float)

print(claim\_data['Amount'])

Q4.

# Sum the amounts based on the status of payment

amount\_paid = claim\_data.loc[claim\_data['Paid'] == 'Yes', 'Amount'].sum()

amount\_unpaid = claim\_data.loc[claim\_data['Paid'] == 'No', 'Amount'].sum()

# Create a pie chart

labels = ['Paid', 'Unpaid']

amounts = [amount\_paid, amount\_unpaid]

colors = ['green', 'red']

plt.pie(amounts, labels=labels, colors=colors, autopct='%1.1f%%')

plt.title('Amounts Paid and Unpaid')

plt.show()

# removing outliers in 'Amount'

q1 = claim\_data['Amount'].quantile(q = 0.25)

q3 = claim\_data['Amount'].quantile(q = 0.75)

iqr = q3 - q1

print(q1)

print(q3)

print(iqr)

no\_outliers = claim\_data[~((claim\_data['Amount']<q1-1.5\*iqr) | (claim\_data['Amount']>q3+1.5\*iqr))]

# box plot to display total claims recorded by year basis

no\_outliers['Year'] = pd.to\_datetime(no\_outliers['Created']).dt.year

no\_outliers.plot(kind = 'box', figsize=(5, 3), column='Amount', by='Year')

plt.xlabel('Year')

plt.ylabel('Claim Amount')

plt.title('Box plot of Claim Amounts by Year')

plt.show()

# finding out mean in the various years and the 25th/75th percentile, afterwards analysing the trend

mean\_by\_year = no\_outliers.groupby('Year')['Amount'].mean()

percentiles\_by\_year = no\_outliers.groupby('Year')['Amount'].quantile([0.25, 0.75])

print(mean\_by\_year)

print(percentiles\_by\_year)

# removing delays that is Nan from the data set

date\_diff\_filtered = claim\_data[claim\_data['Date\_diff'].notnull()]

# plotting scatter plots for delay in days versus amount of the settled claims

no\_outliers.groupby('Year')['Amount'].mean()

plt.scatter(no\_outliers['Date\_diff'], no\_outliers['Amount'])

plt.xlabel('Delay in Days')

plt.ylabel('Settled Claim Amounts')

plt.title('Delay in Days vs Amount Scatter Plot')

plt.show()

Q5.

from sklearn.linear\_model import LinearRegression

from sklearn import preprocessing

from sklearn.preprocessing import StandardScaler

from sklearn.model\_selection import train\_test\_split

from sklearn import linear\_model

# dropping rows with missing values in column "Actual"

claim\_data.dropna(axis = 0, how = "any", subset = ['Actual'], inplace = True)

claim\_data

# perform data normalization for 'Amount' and 'Date\_diff' to have zero mean and unit variance

scaler = StandardScaler()

scaled\_data = scaler.fit\_transform(claim\_data[['Amount', 'Date\_diff']])

claim\_data[['Amount', 'Date\_diff']] = scaled\_data

# scaling the data 'Amount' and 'Date\_diff'

X\_scaled = preprocessing.scale(claim\_data[['Amount', 'Date\_diff']])

# generating polynomial features

pft = preprocessing.PolynomialFeatures(degree = 2)

X\_poly = pft.fit\_transform(X\_scaled)

#Create a copy

claim\_traintest\_set=claim\_data.copy()

Feature = claim\_traintest\_set[[

'Amount',

]]

X = Feature

y = claim\_traintest\_set['Date\_diff'].values

print(x.head())

print(y[0:5])

print(x.shape, y.shape)

# splitting the data set into training and testing dataset

random\_state = 0

test\_size = 0.3

X\_train, X\_test, y\_train, y\_test = train\_test\_split(

X, y, test\_size = test\_size, random\_state = random\_state

)

# conducting multiple linear regression

model = linear\_model.LinearRegression()

model.fit(X\_train, y\_train)

print(model.intercept\_,model.coef\_)

predictionTestSet = model.predict(X\_test)

from sklearn.metrics import mean\_squared\_error

# Error in predicted value

errorTestSet = mean\_squared\_error(y\_test, predictionTestSet)

errorTestSet

Q6.

print(f"y = {model.coef\_}\*X + {model.intercept\_}")

****

**References:**

Wu, K. Y. (2022). ANL252 Python for data analytics (study guide). Singapore University of Social Sciences.

Kumar, A. (2021, December 7). Make your own model to predict house prices in Python. Medium. <https://medium.com/@kumar.bits009/make-your-own-model-to-predict-house-prices-in-python-ad843aee1e>2