

**ANL252**

**Python for Data Analytics**

**End-of-Course Assessment**

**January 2023**



**Submitted by:**

|  |  |
| --- | --- |
| **Name** | **PI No.** |
| Nur Firdaus Bin Mohamed Riduan | B2170520 |

**Tutorial Group:**  **T05**

**Instructor’s Name:**  **Mr Munish Kumar**

**Submission Date:** **6th March 2023**

**Section A**

**Question 1**

Variables that contain missing values:  
Claim\_ID True  
Actual True  
Terms True

**Question 2**

I first dropped the rows that have missing data in the ‘Actual’ and ‘Terms’ columns as the missing data are not retrievable and can lead to a bias analysis. For example, there is no possible way to assign a value to the missing data in the ‘Actual’ column as the claim settlement has not been made and removing the rows will ensure the avoidance of biasness in our analysis involving those variables.

After which, I replaced the missing data in the 'Claim\_ID’ column with ‘0’ as I feel that these missing data are not crucial, as they are just unique identifiers of claim. As they are also unique identifiers, I did not replace the values with the mean, median or mode of the column. I also did not drop the rows with missing data in the ‘Claim\_ID’ column as there might be other data in the other columns that might be useful for analysis and improve the accuracy of the analysis.

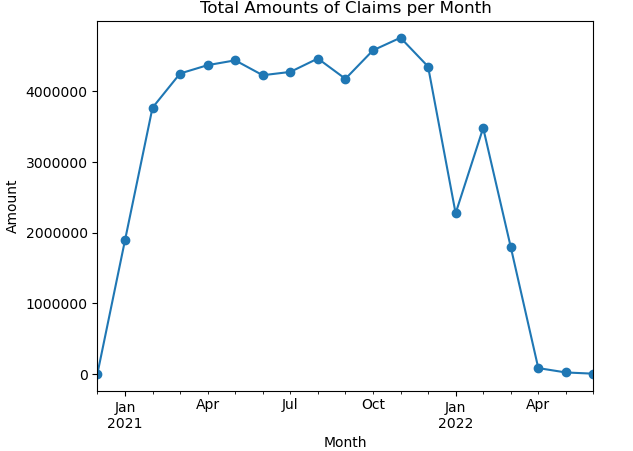
**Question 3**

The first task is to convert the data types in the ‘Planned’, ‘Actual’ and ‘Created’ columns to datetime type. For this case, the data types were converted from object to datetime format by using the ‘pd.to\_datetime’ function on Pandas. Doing so will ease the analysis process, where the data is needed to be in date format. For example, to calculate the delays between the planned and actual date in processing the claims.

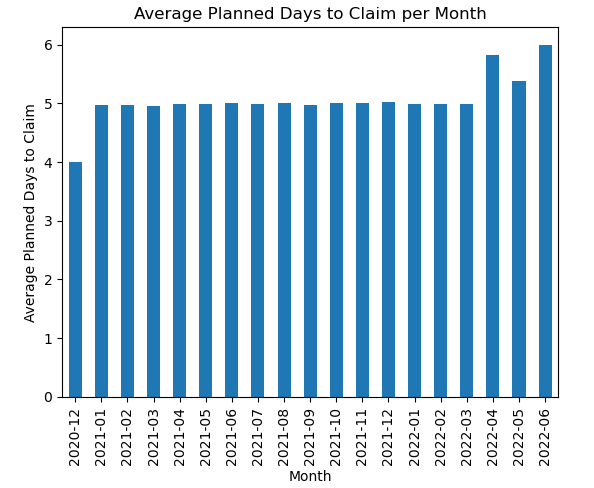
Subsequently, the other task is to replace the ‘1762.OO’ data in the ‘Amount’ column. The first step is to replace the ‘OO’ in the ‘1762.OO’ data with ‘ ‘ by using the ‘.str.replace’ function. Doing so will ensure that our next step will be successful, which is to convert the data type in the ‘Amount’ column to float. After which, I created two new columns: ‘Delay’ and ‘Planned\_Days’. The ‘Delay’ column refers to the number of days between actual and planned date of claims process, while the ‘Planned\_Days’ column refers to the number of intended days for the claims to be processed, from the date of creation.

The third and final task is the removal of outliers from the ‘Amount’ column. I did this by calculating the first quartile (q1) with the ‘.quantile(q = 0.25)’ function, followed by the third quartile (q3) with the ‘.quantile(q = 0.75)’ function. Next is to calculate the interquartile range (iqr) by subtracting q1 from q3. The last step is to determine the outliers in the dataframe and removing them to ensure the accuracy and reliability of the analysis as outliers are not an accurate representation of the data.

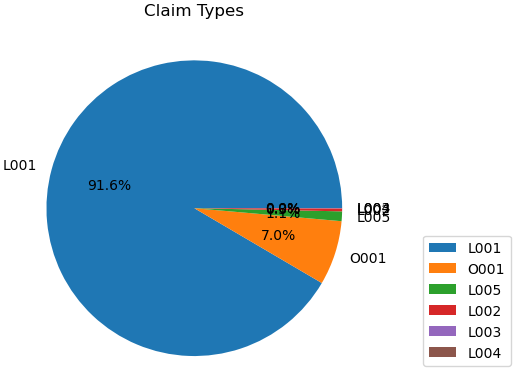
**Question 4**



The line plot above shows the total amount of claims per month, from December 2020 till June 2022. From the above, it can be observed that the company has a relatively high amount of constant claims in 2021, apart from January and February. It is however observed that the claim drops in 2022 onwards, as seen from December 2021 to January 2022 and plummets to a very low amount from April 2022 onwards.

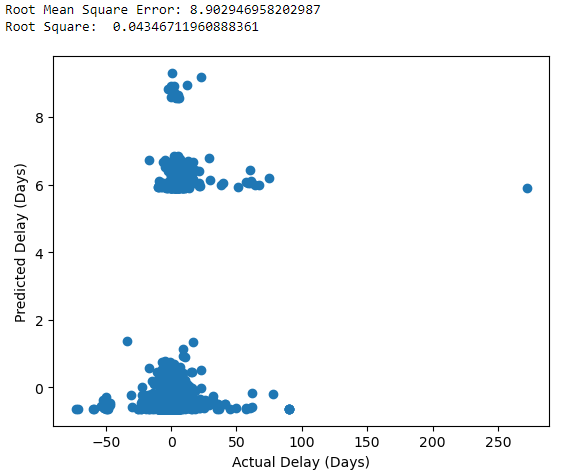


The bar graph above shows the average number of days that the insurance company intends to process the claims per month. It can be observed that the number of days are relatively constant, ranging from four to six days on average. From this, it can be inferred that the insurance company might have a policy where claims are supposed to be processed within four to six days.



The pie chart above shows the percentage of claim types that were processed in the insurance company. It can be observed that 91.6% of the claims were claim type L001, the second highest being O001 at 7% and the remaining less than 2% are made up of L002, L003, L004 and L005.

**Question 5**



The scatter graph above shows the linear regression modelling, predicting the number of delays between actual and planned date in claims processing.

The first step is to import math and the sklean libraries: train\_test\_split, LinearRegression, r2\_score and mean\_squared\_error.

The next step is data preprocessing which includes the selection of only relevant columns: ‘Amount’, ‘Type’ and ‘Delay’ columns. Afterwards is the encoding categorical variables for the ‘Type’ column, so there will be a total of 7 variables: ‘Amount’, ‘L001’, ‘L002’, ‘L003’, ‘L004’, ‘L005’ and ‘O001’.

Subsequently, I split the data into training and test sets, using 20% of the data for testing and the remaining 80% for training. I have also set the random state to 42.

This is followed by the creation, training and evaluation of linear regression model.

The final step is the plotting of the results.

**Question 6**

Root Mean Square Error (RMSE): 8.902946958202987  
  
R-Square: 0.04346711960888361  
  
Slope: [ 8.66142026e-05 -5.73330285e+00 3.32272058e+00 2.41591669e+00  
 3.87816275e+00 -4.67255004e+00 7.89052860e-01]  
  
Y-Intercept: 5.0904628122164945  
  
Linear Regression Equation: y = 0.00x1 + -5.73x2 + 3.32x3 + 2.42x4 + 3.88x5 + -4.67x6 + 0.79x7 + 5.09

Based on the R-squared value, it is a low value thus the model is not a good model.

Likewise, the relatively high RMSE value tells us that the modelling is not a good one.

The linear regression equation is

y = 0.00 Amount + -5.73 L001 + 3.32 L002 + 2.42 L003 + 3.88 L004 + -4.67 L005 + 0.79 O001 + 5.09.

**Appendix**

**Question 1**

import pandas as pd

df = pd.read\_csv('ECA.csv', na\_values=['Unkn', '???'])

variables\_missing = df.isna().any()

print('Variables that contain missing values:')

print(variables\_missing[variables\_missing == True].to\_string(header=False))

**Question 2**

#removing rows with missng data

df.dropna(subset=['Actual','Terms'], inplace=True)

df['Claim\_ID'] = df['Claim\_ID'].fillna(0)

**Question 3**

#converting data type to datetime format

df['Planned'] = pd.to\_datetime(df['Planned'], infer\_datetime\_format=True)

df['Actual'] = pd.to\_datetime(df['Actual'], infer\_datetime\_format=True)

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

df['Delay'] = (df['Actual'] - df['Planned']).dt.days

df['Planned\_Days'] = (df['Planned']-df['Created']).dt.days

#replacing non-numeric characters with blank

df['Amount'] = df['Amount'].str.replace('.OO',' ')

#coverting data type to float

df['Amount'] = df['Amount'].astype(float)

#determining and removing the outliers

q1 = df['Amount'].quantile(q = 0.25)

q3 = df['Amount'].quantile(q = 0.75)

iqr = q3 - q1

df = df[~((df['Amount']<q1-1.5\*iqr) | (df['Amount']>q3+1.5\*iqr))]

**Question 4**

import matplotlib.pyplot as plt

#plotting of line chart

monthly\_totals = df.groupby(df['Planned'].dt.to\_period('M'))['Amount'].sum()

line = monthly\_totals.plot(kind='line', marker='o')

line.set\_title('Total Amounts of Claims per Month (2020-2022)')

line.set\_xlabel('Month')

line.set\_ylabel('Amount')

plt.ticklabel\_format(style='plain', axis='y')

plt.show()

#plotting of bar chart

monthly\_planned\_days = df.groupby(df['Planned'].dt.to\_period('M'))['Planned\_Days'].mean()

monthly\_planned\_days.plot.bar()

plt.xlabel('Month')

plt.ylabel('Average Planned Days to Claim')

plt.title('Average Planned Days to Claim by Month')

plt.show()

#plotting of pie chart

claim\_counts = df['Type'].value\_counts()

plt.pie(claim\_counts, labels=claim\_counts.index, autopct='%1.1f%%')

plt.title('Claim Types')

plt.legend(loc='center left', bbox\_to\_anchor=(1.1, 0.25))

plt.show()

**Question 5**

import math

from sklearn.linear\_model import LinearRegression

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

from sklearn.metrics import r2\_score, mean\_squared\_error

#preprocessing the data

df.copy = df[['Delay', 'Amount', 'Type']]

df.copy = pd.get\_dummies(df.copy, columns=['Type'])

#splitting data into training and test sets

X = df.copy.drop(['Delay'], axis=1)

y = df.copy['Delay']

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)

#creating linear regression model

model = LinearRegression()

#training the model

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

#evaluating the model

y\_pred = model.predict(X\_test)

#plotting the results

plt.scatter(y\_test, y\_pred)

plt.xlabel('Actual Delay (Days)')

plt.ylabel('Predicted Delay (Days)')

plt.show()

**Question 6**

r2 = r2\_score(y\_test, y\_pred)

mse = mean\_squared\_error(y\_test, y\_pred)

rmse = math.sqrt(mse)

print('Root Mean Square Error (RMSE): ', rmse)

print('\nR-Square: ', r2)

print('\nSlope: ', model.coef\_)

print('\nY-Intercept: ', model.intercept\_)

print('\nLinear Regression Equation: y = {:.2f}x1 + {:.2f}x2 + {:.2f}x3 + {:.2f}x4 + {:.2f}x5 + {:.2f}x6 + {:.2f}x7 + {:.2f}'

.format(model.coef\_[0], model.coef\_[1], model.coef\_[2], model.coef\_[3], model.coef\_[4], model.coef\_[5], model.coef\_[6], model.intercept\_))