## 



## **ANL252**

**Python for Data Analytics**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**End-of-Course Assessment**

**January 2023 Presentation**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Name: Kerry Kung Wei Jie**

**Student PI: H2211486**

**Submission Date: 06 March 2023**

**Section A**

**Q1)**

Variables with missing values: Claim\_ID, Actual, Terms

**Q2)**

The columns with missing values identified in question 1 are ‘Claim\_ID’, ‘Terms’ and ‘Actual’.

The blanks in column ‘Actual’ were filtered out by selecting “Yes” in column ‘Paid’. This is because the claims have yet to settle and the data is still incomplete for meaningful comparison. The new dataframe for the filtered data is “paid\_yes”.

Next, every claim that is completed is significant to interpret the data. Hence, a unique temporary claim ID is assigned to the missing claim IDs that are completed, The unique temporary claim ID assigned is in 10 digits format, similar to the rest, starting from 0000000001. Each increment is 1 for every missing claim ID. Lastly, values ‘???’ and ‘Unkn’ are replaced by ‘XXXX’. This is to categorise claims with a missing term code. When checking if ‘Unkn’ and ‘???’ is replaced with XXXX, the below code is used.

print(paid\_yes.iloc[[6177,6329]])

**Q3)**

The datatype for column ‘Claim\_ID’ was float64 and should be formatted to integer as it represents a unique ID without a decimal point. In Question 2, the empty values under column ‘Claim\_ID’ were replaced with 0000000001 in increments of 1, hence column ‘Claim\_ID’ was also formatted to be 10 characters long to include the leading zeros.

Next, the format of columns ‘Planned’, ‘Actual’ and ‘Created’ is inconsistent in the CSV file showing dd/mm/yyyy, dd/mm/yyyy hh:mm, yyyy/mm/dd respectively. It had been formatted to be standardised at yyyy/mm/dd and the datatype had been changed to datetime64 from the previous int64 and object which are incorrect. Using a standardised date format can avoid confusion and makes it easier to compare the data.

Lastly, the format of column ‘Amount’ is changed to float in two decimal points to represent dollar value. However, there was a value error as one of the values was recorded as “1762.OO” instead of “1762.00”. To rectify the error, “O” was replaced with “0” as the data for column ‘Amount’ should only contain integers.

**Q4)**

**Insight 1:**

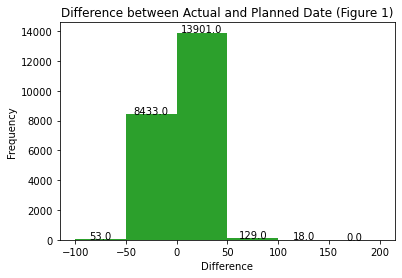


Figure 1 shows if the claims settlements are early, on time or late using the difference between the actual date of claim settlement and the planned date of claim settlement. From Figure 1, it is observed that the distribution is unimodal and normally distributed with the majority of the settlement claims done between on time and 50 days late. There are also a small number of outliers that have a settlement claim period of more than 100 days late. The insurance company could look into their claims settlement process to find out the issue of the delays.

**Insight 2:**

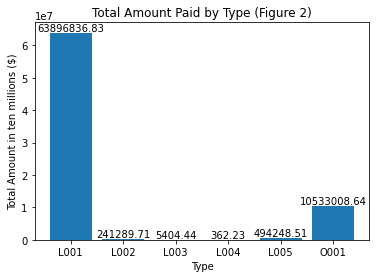


Figure 2 shows a bar chart of the total amount of claim settlement for each type that had been paid. The figure shows that L001 is considerably higher than the rest of the types. This may be a possibility for the insurance company to look into their internal type classification to have a more accurate classification.

**Insight 3:**

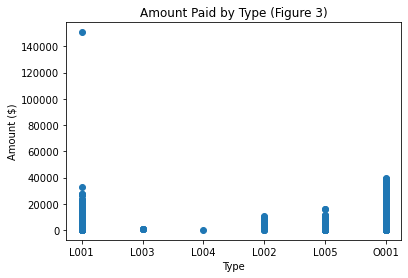


Figure 3 shows a scatter plot diagram of the amount earned by type. This is because from Figure 2, the total amount for type “L001” is considerably high. Hence, a scatter plot is chosen to show if there are any outliers for the amount paid. It is shown that there is indeed an outlier of more than $150,000 claimed for a single claim settlement. This could be a wrong entry keyed into the data. However, it does not cause the extremely high total amount for type “L001”. Hence, the insurance company should look into its internal type classification.

**Q5)**

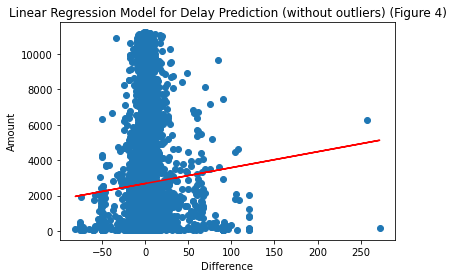


Figure 4 shows the linear regression model for delay prediction based on the amount of claim settlement without outliers. The scikit-learn library is imported using import sklearn. To get the difference between planned and actual, the column ‘Paid’ would have to be filtered to show only “Yes”. The formatting from the previous questions is also used. Next, the data types have been changed to float for both variables to be able to plot the linear regression model. Lastly, the outliers from the amount might affect the predictions, hence only the interquartile range was used. Thereafter, the linear regression line is plotted and predictions made using the model.

**Q6)**

The result from the linear regression model shows that the claim settlement would be earlier than expected for small amounts and the time taken for the claim process would be longer as the amount increases.

The linear regression equation is y = 8.99x + 2676.01

**Appendix 1 - Question 1 Code**

import pandas as pd

# Load the CSV file into dataframe

df = pd.read\_csv('ECA.csv')

#Identify and locate missing\_values and blanks and add results to missing\_column

missing\_values = ['Unkn', '???']

missing\_columns = []

for col in df.columns:

if df[col].isin(missing\_values).any() or df[col].isnull().any():

missing\_columns.append(col)

print(missing\_columns)

**Appendix 2 - Question 2 Code:**

import pandas as pd

# Load the CSV file into dataframe

df = pd.read\_csv('ECA.csv')

# Filter "Yes" in column 'Paid'

paid\_yes = df.loc[df['Paid'] == 'Yes'].copy()

#Assign value to column 'Claim\_ID'

claim\_id\_counter = 1

# Iterate over each row in the DataFrame

for index, row in paid\_yes.iterrows():

# Check if column 'Claim\_ID' is empty

if pd.isna(row['Claim\_ID']):

# If empty, assign value to it, formatted to 10 characters

paid\_yes.at[index, 'Claim\_ID'] = '{:010}'.format(claim\_id\_counter)

# Increment of 1 for next empty cell in column 'Claim\_ID'

claim\_id\_counter += 1

# Replace 'Unkn' and '???' with 'XXXX' in column 'Terms'

paid\_yes.loc[:, 'Terms'] = paid\_yes['Terms'].replace(['Unkn', '???'], 'XXXX')

# For Checking if Unkn and ??? replaced to XXXX

print(paid\_yes.iloc[[6177,6329]])

print(paid\_yes)

**Appendix 3 - Question 3 Code:**

import pandas as pd

# Load the CSV file into dataframe

df = pd.read\_csv('ECA.csv')

# Filter "Yes" in column 'Paid'

paid\_yes = df.loc[df['Paid'] == 'Yes'].copy()

#Assign value to column 'Claim\_ID'

claim\_id\_counter = 1

# Iterate over each row in the DataFrame

for index, row in paid\_yes.iterrows():

# Check if column 'Claim\_ID' is empty

if pd.isna(row['Claim\_ID']):

# If empty, assign value to it, formatted to 10 characters

paid\_yes.at[index, 'Claim\_ID'] = '{:010}'.format(claim\_id\_counter)

# Increment of 1 for next empty cell in column 'Claim\_ID'

claim\_id\_counter += 1

# Replace 'Unkn' and '???' with 'XXXX' in column 'Terms'

paid\_yes.loc[:, 'Terms'] = paid\_yes['Terms'].replace(['Unkn', '???'], 'XXXX')

#Format column 'Claim\_ID' to int, with leading zeros, 10 characters long

paid\_yes['Claim\_ID'] = paid\_yes['Claim\_ID'].apply(lambda x: '{:010}'.format(int(x)) if pd.notnull(x) else '')

#Format column 'Planned', 'Actual', 'Created' to datetime64 in same format yyyy/mm/dd

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

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

paid\_yes['Actual'] = paid\_yes['Actual'].dt.date

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

paid\_yes['Created'] = paid\_yes['Created'].dt.date

#Replace O with 0 and format column 'Amount' to float and 2 d.p

paid\_yes['Amount'] = paid\_yes['Amount'].str.replace('O', '0')

paid\_yes['Amount'] = paid\_yes['Amount'].astype('float').round(2)

print(paid\_yes)

**Appendix 4 - Question 4 Code (Insight 1):**

import pandas as pd

import matplotlib.pyplot as plt

# Load the CSV file into dataframe

df = pd.read\_csv('ECA.csv')

# Filter "Yes" in column 'Paid'

paid\_yes = df.loc[df['Paid'] == 'Yes'].copy()

#Assign value to column 'Claim\_ID'

claim\_id\_counter = 1

# Iterate over each row in the DataFrame

for index, row in paid\_yes.iterrows():

# Check if column 'Claim\_ID' is empty

if pd.isna(row['Claim\_ID']):

# If empty, assign value to it, formatted to 10 characters

paid\_yes.at[index, 'Claim\_ID'] = '{:010}'.format(claim\_id\_counter)

# Increment of 1 for next empty cell in column 'Claim\_ID'

claim\_id\_counter += 1

# Replace 'Unkn' and '???' with 'XXXX' in column 'Terms'

paid\_yes.loc[:, 'Terms'] = paid\_yes['Terms'].replace(['Unkn', '???'], 'XXXX')

#Format column 'Claim\_ID' to int, with leading zeros, 10 characters long

paid\_yes['Claim\_ID'] = paid\_yes['Claim\_ID'].apply(lambda x: '{:010}'.format(int(x)) if pd.notnull(x) else '')

#Format column 'Planned', 'Actual', 'Created' to yyyy/mm/dd, and datatype as object

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

paid\_yes['Planned'] = paid\_yes['Planned'].dt.date

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

paid\_yes['Actual'] = paid\_yes['Actual'].dt.date

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

paid\_yes['Created'] = paid\_yes['Created'].dt.date

#Replace O with 0 and format column 'Amount' to float and 2 d.p

paid\_yes['Amount'] = paid\_yes['Amount'].str.replace('O', '0')

paid\_yes['Amount'] = paid\_yes['Amount'].astype('float').round(2)

# Calculate the difference between the two columns

paid\_yes['Difference'] = paid\_yes['Actual'] - paid\_yes['Planned']

paid\_yes['Difference'] = paid\_yes['Difference'] / pd.Timedelta(days=1)

# Plot a histogram

bins = [-100, -50, 0, 50, 100, 150, 200]

plt.hist(paid\_yes['Difference'], bins=bins)

# Add labels and title to the plot

plt.xlabel('Difference')

plt.ylabel('Frequency')

plt.title('Difference between Actual and Planned Date (Figure 1)')

# Add labels to the bars

for i in range(len(bins)-1):

plt.text((bins[i]+bins[i+1])/2, plt.hist(paid\_yes['Difference'], bins=bins)[0][i]+5, str(plt.hist(paid\_yes['Difference'], bins=bins)[0][i]), ha='center')

plt.show()

**Appendix 5 - Question 4 Code (Insight 2):**

import pandas as pd

import matplotlib.pyplot as plt

# Load the CSV file into dataframe

df = pd.read\_csv('ECA.csv')

# Filter "Yes" in column 'Paid'

paid\_yes = df.loc[df['Paid'] == 'Yes'].copy()

#Assign value to column 'Claim\_ID'

claim\_id\_counter = 1

# Iterate over each row in the DataFrame

for index, row in paid\_yes.iterrows():

# Check if column 'Claim\_ID' is empty

if pd.isna(row['Claim\_ID']):

# If empty, assign value to it, formatted to 10 characters

paid\_yes.at[index, 'Claim\_ID'] = '{:010}'.format(claim\_id\_counter)

# Increment of 1 for next empty cell in column 'Claim\_ID'

claim\_id\_counter += 1

# Replace 'Unkn' and '???' with 'XXXX' in column 'Terms'

paid\_yes.loc[:, 'Terms'] = paid\_yes['Terms'].replace(['Unkn', '???'], 'XXXX')

#Format column 'Claim\_ID' to int, with leading zeros, 10 characters long

paid\_yes['Claim\_ID'] = paid\_yes['Claim\_ID'].apply(lambda x: '{:010}'.format(int(x)) if pd.notnull(x) else '')

#Format column 'Planned', 'Actual', 'Created' to yyyy/mm/dd, and datatype as object

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

paid\_yes['Planned'] = paid\_yes['Planned'].dt.date

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

paid\_yes['Actual'] = paid\_yes['Actual'].dt.date

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

paid\_yes['Created'] = paid\_yes['Created'].dt.date

#Replace O with 0 and format column 'Amount' to float and 2 d.p

paid\_yes['Amount'] = paid\_yes['Amount'].str.replace('O', '0')

paid\_yes['Amount'] = paid\_yes['Amount'].astype('float').round(2)

# Calculate the difference between the two columns

paid\_yes['Difference'] = paid\_yes['Actual'] - paid\_yes['Planned']

paid\_yes['Difference'] = paid\_yes['Difference'] / pd.Timedelta(days=1)

# Convert to float

paid\_yes['Difference'] = paid\_yes['Difference'].astype(float)

# Group by 'Type' and sum 'Amount'

type\_sum = paid\_yes.groupby('Type')['Amount'].sum()

# Plot the histogram

plt.bar(type\_sum.index, type\_sum.values)

for i, v in enumerate(type\_sum.values):

plt.text(i, v, str(v), ha='center', va='bottom')

plt.xlabel('Type')

plt.ylabel('Total Amount in ten millions ($)')

plt.title('Total Amount Paid by Type (Figure 2)')

plt.show()

**Appendix 6 - Question 4 Code (Insight 3):**

import pandas as pd

import matplotlib.pyplot as plt

# Load the CSV file into dataframe

df = pd.read\_csv('ECA.csv')

# Filter "Yes" in column 'Paid'

paid\_yes = df.loc[df['Paid'] == 'Yes'].copy()

#Assign value to column 'Claim\_ID'

claim\_id\_counter = 1

# Iterate over each row in the DataFrame

for index, row in paid\_yes.iterrows():

# Check if column 'Claim\_ID' is empty

if pd.isna(row['Claim\_ID']):

# If empty, assign value to it, formatted to 10 characters

paid\_yes.at[index, 'Claim\_ID'] = '{:010}'.format(claim\_id\_counter)

# Increment of 1 for next empty cell in column 'Claim\_ID'

claim\_id\_counter += 1

# Replace 'Unkn' and '???' with 'XXXX' in column 'Terms'

paid\_yes.loc[:, 'Terms'] = paid\_yes['Terms'].replace(['Unkn', '???'], 'XXXX')

#Format column 'Claim\_ID' to int, with leading zeros, 10 characters long

paid\_yes['Claim\_ID'] = paid\_yes['Claim\_ID'].apply(lambda x: '{:010}'.format(int(x)) if pd.notnull(x) else '')

#Format column 'Planned', 'Actual', 'Created' to yyyy/mm/dd, and datatype as object

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

paid\_yes['Planned'] = paid\_yes['Planned'].dt.date

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

paid\_yes['Actual'] = paid\_yes['Actual'].dt.date

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

paid\_yes['Created'] = paid\_yes['Created'].dt.date

#Replace O with 0 and format column 'Amount' to float and 2 d.p

paid\_yes['Amount'] = paid\_yes['Amount'].str.replace('O', '0')

paid\_yes['Amount'] = paid\_yes['Amount'].astype('float').round(2)

# Calculate the difference between the two columns

paid\_yes['Difference'] = paid\_yes['Actual'] - paid\_yes['Planned']

paid\_yes['Difference'] = paid\_yes['Difference'] / pd.Timedelta(days=1)

# Group by 'Type' and sum 'Amount'

type\_sum = paid\_yes.groupby('Type')['Amount'].sum()

# Create a scatter plot

plt.scatter(paid\_yes['Type'], paid\_yes['Amount'])

plt.xlabel('Type')

plt.ylabel('Amount ($)')

plt.title('Amount Paid by Type (Figure 3)')

plt.show()

**Appendix 7 - Question 5 and 6 Code:**

import pandas as pd

import matplotlib.pyplot as plt

from sklearn.linear\_model import LinearRegression

# Load the CSV file into dataframe

df = pd.read\_csv('ECA.csv')

# Filter "Yes" in column 'Paid'

paid\_yes = df.loc[df['Paid'] == 'Yes'].copy()

#Assign value to column 'Claim\_ID'

claim\_id\_counter = 1

# Iterate over each row in the DataFrame

for index, row in paid\_yes.iterrows():

# Check if column 'Claim\_ID' is empty

if pd.isna(row['Claim\_ID']):

# If empty, assign value to it, formatted to 10 characters

paid\_yes.at[index, 'Claim\_ID'] = '{:010}'.format(claim\_id\_counter)

# Increment of 1 for next empty cell in column 'Claim\_ID'

claim\_id\_counter += 1

# Replace 'Unkn' and '???' with 'XXXX' in column 'Terms'

paid\_yes.loc[:, 'Terms'] = paid\_yes['Terms'].replace(['Unkn', '???'], 'XXXX')

#Format column 'Claim\_ID' to int, with leading zeros, 10 characters long

paid\_yes['Claim\_ID'] = paid\_yes['Claim\_ID'].apply(lambda x: '{:010}'.format(int(x)) if pd.notnull(x) else '')

#Format column 'Planned', 'Actual', 'Created' to yyyy/mm/dd, and datatype as object

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

paid\_yes['Planned'] = paid\_yes['Planned'].dt.date

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

paid\_yes['Actual'] = paid\_yes['Actual'].dt.date

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

paid\_yes['Created'] = paid\_yes['Created'].dt.date

#Replace O with 0 and format column 'Amount' to float and 2 d.p

paid\_yes['Amount'] = paid\_yes['Amount'].str.replace('O', '0')

paid\_yes['Amount'] = paid\_yes['Amount'].astype('float').round(2)

# Calculate the difference between the two columns

paid\_yes['Difference'] = paid\_yes['Actual'] - paid\_yes['Planned']

paid\_yes['Difference'] = paid\_yes['Difference'] / pd.Timedelta(days=1)

# Convert to float

paid\_yes['Difference'] = paid\_yes['Difference'].astype(float)

# Calculate the IQR of 'Planned' (Question 5)

Q1 = paid\_yes['Amount'].quantile(0.25)

Q3 = paid\_yes['Amount'].quantile(0.75)

IQR = Q3 - Q1

# Define the outlier boundaries

lower\_bound = Q1 - 1.5 \* IQR

upper\_bound = Q3 + 1.5 \* IQR

# Remove outliers

paid\_yes = paid\_yes[(paid\_yes['Amount'] >= lower\_bound) & (paid\_yes['Amount'] <= upper\_bound)]

# Fit a linear regression model to the updated data

X = paid\_yes[['Difference']]

y = paid\_yes['Amount']

model = LinearRegression()

model.fit(X, y)

# Make predictions using the model

predictions = model.predict(X)

# Plot the actual data and the model's predictions

plt.scatter(X['Difference'], y)

plt.plot(X['Difference'], predictions, color='red')

plt.xlabel('Difference')

plt.ylabel('Amount')

plt.title('Linear Regression Model for Delay Prediction (without outliers) (Figure 4)')

plt.show()

# Get slope and y-intercept values (Question 6)

m = model.coef\_[0]

b = model.intercept\_

# Print equation of the line

print("y = {:.2f}x + {:.2f}".format(m, b))