

End-of-Course Assessment (ECA)

For Module:

ANL252, Python for Data Analytics

Submitted by:

Shamini D/O Eliaperumal (E2170152)

Date of Submission:

6 March 2023

**Question 1.**

import pandas as pd

# List that contains all the possible missing values

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

insurance\_data = pd.read\_csv('ECA.csv', na\_values= missing\_values)

## Question 1

## Print the list of columns with missing data = True

print()

print("Columns with missing values:")

print()

print(insurance\_data.isnull().any().to\_string())

print()

**Output >>>**

Columns with missing values:

Claim\_ID True

Policy\_No False

Name False

Planned False

Actual True

Created False

Amount False

Paid False

Category False

Terms True

Region False

Type False

**Question 2.**

import pandas as pd

# List that contains all the possible missing values

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

insurance\_data = pd.read\_csv('ECA.csv', na\_values= missing\_values)

## Question 2

## Print the number of missing rows per column

print()

print("Count rows with missing data by Column:")

print()

print(insurance\_data.isnull().sum().to\_string())

print()

## Drop rows with missing "Claim\_ID"

insurance\_data.dropna(subset=['Claim\_ID'], inplace=True)

## Replace missing values in column "Terms"

insurance\_data['Terms'].fillna(value="CUST", inplace=True)

print("Data prepared:")

print()

print(insurance\_data)

print()

**Output >>>**

Count rows with missing data by Column:

Claim\_ID 5

Policy\_No 0

Name 0

Planned 0

Actual 1677

Created 0

Amount 0

Paid 0

Category 0

Terms 7

Region 0

Type 0

Data prepared:

Claim\_ID Policy\_No Name Planned Actual Created Amount Paid Category Terms Region Type

0 2.928510e+09 300764795 Roger Torres 17/1/2021 18/1/2021 0:00 20210112 3072.349 Yes AT AD23 LOC L001

1 2.928511e+09 300434439 Jason Jones 5/2/2021 16/1/2021 0:00 20210130 910.944 Yes AT EC05 LOC L001

2 2.928517e+09 300769623 Robert Martin 18/1/2021 14/1/2021 0:00 20210113 567.936 Yes AT AB27 LOC L001

3 2.928517e+09 300794332 Stacy Anderson 15/1/2021 18/1/2021 0:00 20210110 181.651 Yes AT AE14 LOC L001

4 2.928518e+09 300792283 Mr. Adam Whitaker III 5/2/2021 8/2/2021 0:00 20210131 238.74 Yes AT EC05 LOC L001

... ... ... ... ... ... ... ... ... ... ... ... ...

24203 3.960633e+09 240104229 Tyler Hall 22/5/2022 NaN 20220517 561.516 No AT CB91 FVS O001

24204 3.960633e+09 240104340 Lucas Hill 21/5/2022 NaN 20220516 124.106 No AT CB91 FVS O001

24205 3.960633e+09 240105686 Nicole Gray 19/5/2022 NaN 20220514 2825.863 No AT CB91 FVS O001

24206 3.960633e+09 240105686 Mr. Robert Rivera 18/5/2022 NaN 20220513 3661.873 No AT CB91 FVS O001

24207 3.960634e+09 240104409 Nathan Kennedy 18/5/2022 NaN 20220513 1403.989 No AT CB91 FVS O001

[**24208 rows x 12 columns**]

To treat the missing data as part of data preparation, we need to decide whether to drop, fill or replace the missing values based on our analysis goal and the nature of the dataset. Some of the factors that I considered are:

\* The percentage of missing values in a row or column. If it is too high, I may want to drop it as it may not provide much information.

\* The pattern of missing values in a row or column. If it is random or not related to other variables, I may want to fill or replace it with some reasonable values. If it is systematic or related to other variables, you may want to drop it or use a more sophisticated imputation method.

\* The type of missing values in a row or column. If it is numeric, I may want to fill or replace it with some statistics like mean, median or mode. If it is categorical, I may want to fill or replace it with some labels like “Unknown”, “Other” or the most frequent category.

The rationale of the treatments I implemented are:

1) I dropped "Claim\_ID" rows with empty values as these entries may be caused by some system or input error. The amount is 2198 and the difference can be revised by some back office team.

2) I leaved "Actual" empty rows as these claims are not paid yet.

3) I replaced "Terms" values as the terms and conditions maybe didn't exist or were customized for this specific client and claim. For this reason, I will replace with value "CUST".

**Question 3.**

import pandas as pd

# List that contains all the possible missing values

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

insurance\_data = pd.read\_csv('ECA.csv', na\_values= missing\_values)

# Drop rows with missing "Claim\_ID"

insurance\_data.dropna(subset=['Claim\_ID'], inplace=True)

# Replace missing values in column "Terms"

insurance\_data['Terms'].fillna(value="CUST", inplace=True)

# Question 3

# 1- Remove duplicates

insurance\_data.drop\_duplicates(inplace=True)

# 2- Data type conversion

# Convert columns "Actual" and "Planned" to datetime format

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

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

# Extract columns "Actual" and "Planned" in 'YYYYMMDD' format

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

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

# Convert column "Amount" to float - There is a wrong value: "1762.OO" - The result will be NaN

insurance\_data['Amount'] = pd.to\_numeric(insurance\_data['Amount'], errors='coerce')

# Drop wrong values - NaN

insurance\_data.dropna(subset=['Amount'], inplace=True)

# 3 - Remove outliers

q1 = insurance\_data['Amount'].quantile(q = .25)

q3 = insurance\_data['Amount'].quantile(q = .75)

iqr = q3 - q1

low\_bound = q1-1.5\*iqr

upper\_bound = q3+1.5\*iqr

print()

print("IQR details:")

print(f"q1: {q1}\nq3: {q3}\ninterquartile range: {iqr}\nlower threshold: {low\_bound}\nupper threshold: {upper\_bound}")

print()

insurance\_data = insurance\_data[~((insurance\_data["Amount"]<low\_bound) | (insurance\_data["Amount"]>upper\_bound))]

print("Data prepared:")

print()

print(insurance\_data)

print()

**Output >>>**

IQR details:

q1: 525.02025

q3: 4803.76075

interquartile range: 4278.7405

lower threshold: -5893.0905

upper threshold: 11221.871500000001

Data prepared:

Claim\_ID Policy\_No Name Planned Actual Created Amount Paid Category Terms Region Type

0 2.928510e+09 300764795 Roger Torres 20210117 20210118 20210112 3072.349 Yes AT AD23 LOC L001

1 2.928511e+09 300434439 Jason Jones 20210205 20210116 20210130 910.944 Yes AT EC05 LOC L001

2 2.928517e+09 300769623 Robert Martin 20210118 20210114 20210113 567.936 Yes AT AB27 LOC L001

3 2.928517e+09 300794332 Stacy Anderson 20210115 20210118 20210110 181.651 Yes AT AE14 LOC L001

4 2.928518e+09 300792283 Mr. Adam Whitaker III 20210205 20210208 20210131 238.740 Yes AT EC05 LOC L001

... ... ... ... ... ... ... ... ... ... ... ... ...

24203 3.960633e+09 240104229 Tyler Hall 20220522 NaN 20220517 561.516 No AT CB91 FVS O001

24204 3.960633e+09 240104340 Lucas Hill 20220521 NaN 20220516 124.106 No AT CB91 FVS O001

24205 3.960633e+09 240105686 Nicole Gray 20220519 NaN 20220514 2825.863 No AT CB91 FVS O001

24206 3.960633e+09 240105686 Mr. Robert Rivera 20220518 NaN 20220513 3661.873 No AT CB91 FVS O001

24207 3.960634e+09 240104409 Nathan Kennedy 20220518 NaN 20220513 1403.989 No AT CB91 FVS O001

[**22962 rows x 12 columns**]

In this case, I implemented three more data preparation tasks for further analysis the data:

1. Remove duplicates: This task considers only the first value as unique.
2. Data conversion: I noticed the dates columns are not in the same format, and it is important to keep the same format to work with these columns. Other conversion I’m doing is in amount column as there is a value with letters ‘1762.OO’. This numeric conversion will return the value ‘NaN’ if the amount is not valid, which will help to filter wrong input.
3. Outliers: Using statistics such as the interquartile range (IQR) to detect the existence of outliers in a variable and remove from the dataset. The only column I think could be the target is “Amount”.

**Question 4 – Insight 1**

**Chart, line chart

Description automatically generated**

The average delay in days between planned and actual date of processing claims has always been below 2 but after April 2022, it went up to above 10. The insurance company will need to investigate the reason of the delay.

**Question 4 – Insight 2**

Chart

Description automatically generated

L001 has the highest amount claims paid out and the other types has minimal number of claims. The insurance company might consider reorganising the types or increase the granularity type to avoid a generic type that comprise of all the claims.

**Question 4 – Insight 3**

Chart, bar chart, histogram

Description automatically generated

Among all the customers, there are 11 that exceeds over 500 claims, which is much higher from the average customer claim. The insurance company might consider to investigate the reason of these high number of claims as it might be possible that there is some fraudulence or misuse of the insurance policy.

**Question 5.**

# Import libraries

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

from sklearn.linear\_model import LinearRegression

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

# List that contains all the possible missing values

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

insurance\_data = pd.read\_csv('ECA.csv', na\_values= missing\_values)

# Drop rows with missing "Claim\_ID"

insurance\_data.dropna(subset=['Claim\_ID'], inplace=True)

# Replace missing values in column "Terms"

insurance\_data['Terms'].fillna(value="CUST", inplace=True)

# 1- Remove duplicates

insurance\_data.drop\_duplicates(inplace=True)

# 2- Data type conversion

# Convert columns "Actual" and "Planned" to datetime format

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

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

# Convert column "Amount" to float - There is a wrong value: "1762.OO" - The result will be NaN

insurance\_data['Amount'] = pd.to\_numeric(insurance\_data['Amount'], errors='coerce')

# Drop wrong values - NaN

insurance\_data.dropna(subset=['Amount'], inplace=True)

# 3 - Remove outliers

q1 = insurance\_data['Amount'].quantile(q = .25)

q3 = insurance\_data['Amount'].quantile(q = .75)

iqr = q3 - q1

low\_bound = q1-1.5\*iqr

upper\_bound = q3+1.5\*iqr

# Filter by IQR

insurance\_data = insurance\_data[~((insurance\_data["Amount"]<low\_bound) | (insurance\_data["Amount"]>upper\_bound))]

# Calculate delay in days

insurance\_data["Delay"] = (insurance\_data["Actual"] - insurance\_data["Planned"]).dt.days

# Filter by Paid = Yes

insurance\_data = insurance\_data[insurance\_data['Paid'] == 'Yes']

# Select features and target

features = ["Policy\_No", "Amount", "Category", "Terms", "Region", "Type"]

target = ["Delay"]

# Convert categorical variables to dummy variables using pandas.get\_dummies()

X = pd.get\_dummies(insurance\_data[features])

y = insurance\_data[target]

# Create a linear regression model and fit it with data

model = LinearRegression()

model.fit(X, y)

# Print model coefficients and intercept

print("Model coefficients:", model.coef\_)

print("Model intercept:", model.intercept\_)

# Make predictions on the same data and evaluate model performance

y\_pred = model.predict(X)

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

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

rmse = np.sqrt(mse)

print("R-squared score:", r2)

print("Mean squared error:", mse)

print("Root mean squared error:", rmse)

# Plot actual vs predicted values using matplotlib.pyplot.scatter()

plt.scatter(y, y\_pred)

plt.xlabel("Actual Delay")

plt.ylabel("Predicted Delay")

plt.title("Linear Regression Model")

plt.show()

**Chart, scatter chart

Description automatically generated**

Output >>>

Model coefficients: [[ 2.09365945e-09 -1.17964262e-04 -2.11944451e-06 -1.37100067e+00

1.25791215e+01 -1.46804853e+00 -9.54134265e+00 5.58352554e+01

-4.15797748e+00 -8.12859887e-01 6.09548557e+01 -1.60042017e-01

-7.84124249e+00 -2.07196338e+00 -1.93970552e+01 6.67218438e+01

-1.28418495e+00 1.73009905e+01 -3.26062534e-01 2.50166815e+00

-1.61650718e+01 -7.29462421e-01 -5.38926478e+00 9.68010249e+00

-7.39643761e+00 -9.89478644e+00 -4.53238588e+00 -9.81783301e-01

-1.51725745e+00 -1.44021576e+00 -7.06123991e+00 8.31567303e+00

2.99391504e-01 -9.84522620e+00 1.93838133e+00 2.09582917e+00

-1.29280237e+00 -2.88408923e-01 -1.53834939e+00 -1.45374201e+00

6.66699592e+00 -6.66434661e+00 2.44741435e+01 2.22294433e+00

-2.95228579e+00 -4.88388672e+00 -3.84655641e+00 -4.16883733e+00

-4.38843008e+01 -1.40607510e+00 -9.30748890e+00 -1.16505949e+01

-5.57576788e+00 8.42780278e-01 -1.70180404e+01 9.93208496e+00

-1.57557691e+00 7.32508712e+00 1.17525784e+01 5.57521049e+00

-1.73457424e+01 -4.86901058e+00 4.40745273e+00 -1.91152627e+00

-8.81898789e+00 -3.20199243e+00 -3.91548279e+01 -5.22837763e+00

3.72887771e-01 -3.72891637e-01 -1.00347960e+00 -8.66681581e-01

-2.92497305e+00 1.30788760e+00 3.11435136e+00 3.72894751e-01]]

Model intercept: [3.28149981]

R-squared score: 0.1729148116423922

Mean squared error: 74.9299868907199

Root mean squared error: 8.65621088529617

**Question 6.**

A linear regression equation is a mathematical expression that describes the relationship between a dependent variable and one or more independent variables. The linear regression model from Question 5 has 77 predictor variables.

The R-squared score is a measure of how well the model fits the data. It ranges from 0 to 1, with higher values indicating better fit. My R-squared score is 0.1729148116423922 which means that the model explains about 17% of the variation in y.

The mean squared error (MSE) is a measure of how much error there is between the predicted values and the actual values of y. It is calculated by taking the average of the squared differences between y and y-hat (the predicted value). My MSE is 74.9299868907199 which means that on average, the predictions are off by about 75 units squared.

The root mean squared error (RMSE) is another measure of error that is calculated by taking the square root of MSE. It has the same units as y and gives an idea of how much error there is in terms of those units. My RMSE is 8.65621088529617 which means that on average, the predictions are off by about 8.66 units.

In general, a higher R-squared value is better than a lower one because it means that the model explains more of the variation in the dependent variable. However, there are some caveats to consider:

- A high R-squared does not necessarily mean that the model is correct or has a causal relationship. It only measures how well the model fits the data, not whether it makes sense theoretically or practically.

- A high R-squared can also be a sign of overfitting, which means that the model is too complex and captures the noise rather than the signal in the data. Overfitting can reduce the generalizability and predictive power of the model on new data. We can use adjusted R-squared or other methods to penalize adding too many variables to the model and avoid overfitting.

- A low R-squared does not necessarily mean that the model is bad or useless. It may be that the dependent variable is inherently hard to predict or has a lot of random variation that is not related to the independent variables. In some fields, such as social sciences or biology, a low R-squared can still be acceptable if it provides meaningful insights or hypotheses.

Therefore, we should not rely solely on R-squared to evaluate the model performance. We should also consider other criteria such as theoretical soundness, practical relevance, cross-validation results, and residual analysis.

# **Appendix**

**Question 4 – Insight 1**

import pandas as pd

import matplotlib.pyplot as plt

# List that contains all the possible missing values

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

insurance\_data = pd.read\_csv('ECA.csv', na\_values= missing\_values)

# Drop rows with missing "Claim\_ID"

insurance\_data.dropna(subset=['Claim\_ID'], inplace=True)

# Replace missing values in column "Terms"

insurance\_data['Terms'].fillna(value="CUST", inplace=True)

# 1- Remove duplicates

insurance\_data.drop\_duplicates(inplace=True)

# 2- Data type conversion

# Convert columns "Actual" and "Planned" to datetime format

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

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

# Convert column "Amount" to float - There is a wrong value: "1762.OO" - The result will be NaN

insurance\_data['Amount'] = pd.to\_numeric(insurance\_data['Amount'], errors='coerce')

# Drop wrong values - NaN

insurance\_data.dropna(subset=['Amount'], inplace=True)

# 3 - Remove outliers

q1 = insurance\_data['Amount'].quantile(q = .25)

q3 = insurance\_data['Amount'].quantile(q = .75)

iqr = q3 - q1

low\_bound = q1-1.5\*iqr

upper\_bound = q3+1.5\*iqr

# Filter by IQR

insurance\_data = insurance\_data[~((insurance\_data["Amount"]<low\_bound) | (insurance\_data["Amount"]>upper\_bound))]

#Filter only paid claims

insurance\_data = insurance\_data[insurance\_data['Paid'] == 'Yes']

# Question 4 - Insight 1

# Calculate delay in days between planned and actual date of processing

insurance\_data["Delay"] = (insurance\_data["Actual"] - insurance\_data["Planned"]).dt.days

# group data by month and calculate average delay

avg\_delay = insurance\_data.groupby(insurance\_data['Actual'].dt.strftime('%Y-%m'))['Delay'].mean()

# plot average delay over time

plt.plot(avg\_delay.index, avg\_delay.values, '-o')

plt.xlabel('Month')

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

plt.title('Average Delay in Days Between Planned and Actual Date of Processing Claims')

plt.show()

**Question 4 – Insight 2**

import pandas as pd

import matplotlib.pyplot as plt

# List that contains all the possible missing values

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

insurance\_data = pd.read\_csv('ECA.csv', na\_values= missing\_values)

# Drop rows with missing "Claim\_ID"

insurance\_data.dropna(subset=['Claim\_ID'], inplace=True)

# Replace missing values in column "Terms"

insurance\_data['Terms'].fillna(value="CUST", inplace=True)

# 1- Remove duplicates

insurance\_data.drop\_duplicates(inplace=True)

# 2- Data type conversion

# Convert columns "Actual" and "Planned" to datetime format

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

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

# Convert column "Amount" to float - There is a wrong value: "1762.OO" - The result will be NaN

insurance\_data['Amount'] = pd.to\_numeric(insurance\_data['Amount'], errors='coerce')

# Drop wrong values - NaN

insurance\_data.dropna(subset=['Amount'], inplace=True)

# 3 - Remove outliers

q1 = insurance\_data['Amount'].quantile(q = .25)

q3 = insurance\_data['Amount'].quantile(q = .75)

iqr = q3 - q1

low\_bound = q1-1.5\*iqr

upper\_bound = q3+1.5\*iqr

# Filter by IQR

insurance\_data = insurance\_data[~((insurance\_data["Amount"]<low\_bound) | (insurance\_data["Amount"]>upper\_bound))]

#Filter only paid claims

insurance\_data = insurance\_data[insurance\_data['Paid'] == 'Yes']

# Question 4 - Insight 2

# Group data by type and sum amount

total\_amount = insurance\_data.groupby('Type')['Amount'].sum()

# Plot total amount by type

plt.bar(total\_amount.index, total\_amount.values)

plt.xlabel('Type')

plt.ylabel('Total Amount Paid')

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

plt.show()

**Question 4 – Insight 3**

import pandas as pd

import matplotlib.pyplot as plt

# List that contains all the possible missing values

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

insurance\_data = pd.read\_csv('ECA.csv', na\_values= missing\_values)

# Drop rows with missing "Claim\_ID"

insurance\_data.dropna(subset=['Claim\_ID'], inplace=True)

# Replace missing values in column "Terms"

insurance\_data['Terms'].fillna(value="CUST", inplace=True)

# 1- Remove duplicates

insurance\_data.drop\_duplicates(inplace=True)

# 2- Data type conversion

# Convert columns "Actual" and "Planned" to datetime format

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

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

# Convert column "Amount" to float - There is a wrong value: "1762.OO" - The result will be NaN

insurance\_data['Amount'] = pd.to\_numeric(insurance\_data['Amount'], errors='coerce')

# Drop wrong values - NaN

insurance\_data.dropna(subset=['Amount'], inplace=True)

# 3 - Remove outliers

q1 = insurance\_data['Amount'].quantile(q = .25)

q3 = insurance\_data['Amount'].quantile(q = .75)

iqr = q3 - q1

low\_bound = q1-1.5\*iqr

upper\_bound = q3+1.5\*iqr

# Filter by IQR

insurance\_data = insurance\_data[~((insurance\_data["Amount"]<low\_bound) | (insurance\_data["Amount"]>upper\_bound))]

#Filter only paid claims

insurance\_data = insurance\_data[insurance\_data['Paid'] == 'Yes']

# Question 4 - Insight 3

# Group data by name and count number of claims

claims\_by\_policy = insurance\_data.groupby('Name')['Claim\_ID'].count()

# Get top 15 names with most claims

top\_15\_policies = claims\_by\_policy.nlargest(15)

# Plot top 50 policies by number of claims

plt.bar(top\_15\_policies.index, top\_15\_policies.values)

plt.xlabel('Name')

plt.ylabel('Number of Claims')

plt.title('Top 15 Claims by Name')

plt.xticks(rotation=50)

plt.show()