

**ANL252**

**Python for Data Analytics**

# **ECA**

**January 2023**

**Submitted by:**

|  |  |
| --- | --- |
| **Name** | **PI No.** |
| **Ng Kuo Xun** | **H2071354** |
| Click to enter NAME. | Click to enter PI NO. |
| Click to enter NAME. | Click to enter PI NO. |
| Click to enter NAME. | Click to enter PI NO. |

**Submission Date: 06/03/2023**

1)

By putting “Unkn” and “???” as na\_values in panda command, we will be able to see that the variables with missing values is “Claim\_ID”, “Actual” and “Terms”.

2)

The main method to treat missing data are deleting of data, replacing data and ignoring. Deleting of data are to delete the column/row that have missing values and act like it does not exist. Replacing data are to replace the missing value with data with a predefined value such as average. Ignoring missing data are to ignore the data as certain algorithm does not use the data if there a missing value by default. In our case, I will opt to delete the data that have missing values. This is due to the whole dataset have a total number of 24213 data and the number of data that have missing values are roughly 1700. Since the number of data that have missing data are less than 10% of a huge dataset, deleting those that data will not affect the result of the analysis.

3)

The 3 other steps for data preparation that I will do are Data grouping, handling of outliers, encoding of categorical variables.

Data grouping:

Depending on what kind of analysis we want to do, there are times whereby we will need to use data grouping to group certain data together. For example, in our dataset, there are a column of “Region”. It would be good if we are able to group them together and do separate analysis since location are factors that affect the culture of things happening. To group the data, we can use “DataFrame\_name.groupby(by = [List\_of\_Labels]).anymethod()” method.

Handling of outliers:

Outliers of data will greatly affect the result of an analysis if it is not handled properly. In our dataset, the numeric data of “Amount” have extreme outliers like the 150723.86 which are more than 100000 from the biggest amount before that. This will greatly affect the average despite it is the only data with such a huge amount. We can remove the outliers by using IQR and define the data by IQR.

Encoding of categorical variables:

The dataset contains categorical variables which might not be able to use when doing analysis. We will have to encode the variables to numerical values before certain algorithms can run them. For example, the “Region” which define the location of the data are not in numerical value and to encode it, we can use pd.get\_dummies.

4)

Chart, bar chart

Description automatically generated

Insight 1: Average Payout Amount by Region. The first insight I would like to provide is the average payout amount by region. This insight can help the insurance company understand the distribution of payouts across different regions and identify any regions where the payouts are higher or lower than the average.

Chart, histogram

Description automatically generated

Insight 2: Numbers of days of delays. The second insight I would like to provide is the overview of numbers of days delay from planned claim date to actual claim date. This data will allow the company to know if planned claim date is accurate. If it is accurate, that’s mean the algorithm that the company are using is good, if not they might have to do some adjustment.

Chart, bar chart, histogram

Description automatically generated

Insight 3: Distribution of claims type. The second insight I would like to provide is the distribution of claims type. This data will allow the company to know what the most common claims type are, and, in this case, it will be L001. From there companies can prepare and limit the number of claims for this type of insurance since it will most likely get claimed.

5)

Chart, scatter chart

Description automatically generated

We will do this step by step below:

1. We will have to load the dataset into panda.
2. Creation of new column “Delay” in the data frame to calculate the number of days between “Planned” and “Actual.”
3. Check for missing values and drop the data with missing values.
4. Plot the model to check for relationship between delay and other variable.
5. Data preparation for modeling. Encoding categorical variables. Ensuring there are numeric features in data.
6. Split the data into training and test set.
7. Train linear regression model on training set.
8. Evaluate the performance of the model from testing set.

Qn 6)

We got a RMSE of 8.825 and R square of 0.205 which tell us that our model is not able to explain significant amount of variance in the delay between planned and actual dates. This could be because of insufficient features due to underfitting the data as there are lack of informative features. It could also be caused by outliers as we did not handle outliers in our coding.

Appendix:

**Qn 1.**  
import pandas as pd

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

vars\_with\_missing = df.columns[df.isna().any()].tolist()

print("Variables that contain missing values:", vars\_with\_missing)

**Qn 4.**

**Insight 1:**

import pandas as pd

import matplotlib.pyplot as plt

df = pd.read\_csv('3.csv')

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

avg\_payout\_by\_region = df.groupby('Region')['Amount'].mean()

plt.bar(avg\_payout\_by\_region.index, avg\_payout\_by\_region.values)

plt.xlabel('Region')

plt.ylabel('Average Payout Amount')

plt.title('Average Payout Amount by Region')

plt.show()

**Insight 2:**

import pandas as pd

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

vars\_with\_missing = df.columns[df.isna().any()].tolist()

df = df.dropna(subset=['Actual'])

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

df['Planned'] = pd.to\_datetime(df['Planned'], format='%d/%m/%Y', errors='coerce')

df['Actual'] = pd.to\_datetime(df['Actual'], format='%d/%m/%Y', errors='coerce')

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

avg\_delay = df['DaysBetween'].mean()

plt.hist(df['DaysBetween'].dropna(), bins=30)

plt.axvline(avg\_delay, color='r', linestyle='--', label='Average delay')

plt.xlabel('Number of days of delay')

plt.ylabel('Number of claims')

plt.title('Histogram of settlement delay')

plt.legend()

plt.xlim([-100, 100])

plt.show()

**Insight 3:**

import pandas as pd

import matplotlib.pyplot as plt

# Load the data

import pandas as pd

import matplotlib.pyplot as plt

df = pd.read\_csv("3.csv")

plt.bar(df["Type"].unique(), df["Type"].value\_counts())

plt.xlabel("Claim Type")

plt.ylabel("Count")

plt.title("Distribution of Claim Types")

plt.show()

**Qn 5:**

import pandas as pd

import numpy as np

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

from sklearn.linear\_model import LinearRegression

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

from sklearn.preprocessing import StandardScaler, OneHotEncoder

from sklearn.compose import ColumnTransformer

import matplotlib.pyplot as plt

df = pd.read\_csv('3.csv')

print(df.head())

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

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

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

df['Delay'] = df['Actual'] - df['Planned']

df['Delay'] = df['Delay'].apply(lambda x: x.days)

print(df.head())

print(df.isnull().sum())

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

df = df.dropna()

plt.scatter(df['Amount'], df['Delay'])

plt.xlabel('Amount')

plt.ylabel('Delay')

plt.show()

print(df.corr())

numeric\_features = ['Amount']

numeric\_transformer = StandardScaler()

categorical\_features = ['Policy\_No', 'Name', 'Category', 'Terms', 'Region', 'Type']

categorical\_transformer = OneHotEncoder(handle\_unknown='ignore')

preprocessor = ColumnTransformer(

transformers=[

('num', numeric\_transformer, numeric\_features),

('cat', categorical\_transformer, categorical\_features)])

X = df.drop(['Delay', 'Claim\_ID', 'Created', 'Actual'], axis=1)

y = df['Delay']

if len(numeric\_features) > 0:

X\_prep = preprocessor.fit\_transform(X[numeric\_features + categorical\_features])

else:

X\_prep = preprocessor.fit\_transform(X[categorical\_features])

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

model = LinearRegression()

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

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

print('Root Mean Squared Error:', np.sqrt(mean\_squared\_error(y\_test, y\_pred)))

print('R^2 Score:', r2\_score(y\_test, y\_pred