****

**ANL 252**

**PYTHON FOR DATA ANALYTICS**

**End-of-Course Accessment**

**January 2023 Presentation**

**Submitted by:**

|  |  |
| --- | --- |
| **Name** | **PI Number** |
| XIONG XIAOFENG | Y2110266 |
|  |  |
|  |  |
|  |  |

**Tutorial Group: T05 / Group 07**

**Instructor’s Name: MUNISH KUMAR**

**Submission Date:   21 Feb 2023**

Q1.

Variables that contain missing values:

Claim\_ID, Actual & Terms

Q2.

The columns that contain missing values are “Claim\_ID”, “Actial” & “Terms”.

The usual ways to treat missing data are replace, ignore or drop them. As we cannot define the variable mean of each specific column, the most suitable way of data treatment is to delete all the missing value.

Q3.

1) Showing Basics Statistics

This function is to show the overview of the values that each column contains. By using the function “.describe():”, we can get the following information:

| **Name** | **Planned** | **Actual** | **Amount** | **Paid** | **Category** | **Terms** | **Region** | **Type** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **count** | 22524 | 22524 | 22524 | 22524 | 22524 | 22524 | 22524 | 22524 | 22524 |
| **unique** | 2979 | 478 | 376 | 20969 | 1 | 1 | 64 | 2 | 6 |
| **top** | Troy Phillips | 8/11/2021 | 12/11/2021 0:00 | 102.956 | Yes | AT | AD23 | LOC | L001 |
| **freq** | 535 | 103 | 176 | 52 | 22524 | 22524 | 9461 | 20736 | 20432 |

2) Showing how often specific values occur in a column

By using the “value\_counts()”, we can know the occurrence number of one specific value of a column. For example, we want to know the how often the value “LOC” appears in column “Region”:

LOC 20736

FVS 1788

Name: Region, dtype: int64

3) Covert string date time into Python date time

When importing a csv file and creating a data frame, the date time objects contained within the file are interpreted as string objects rather than date time objects. This can make it difficult to perform operations such as calculating time differences using strings instead of proper date time objects. By using “to\_datetime()” method, the string date time object can be converted into Python date time objects

(Using column “Actual” as example)

Output:

0 2021-08-20

1 2021-11-27

2 2021-03-27

3 2021-03-15

4 2021-09-17

5 2021-02-07

6 2021-07-11

7 2021-01-07

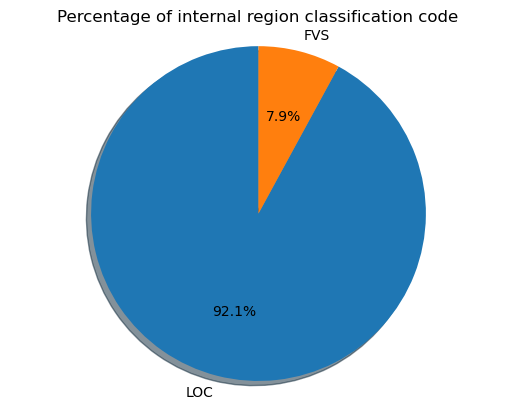
8 2021-01-10

9 2021-02-21

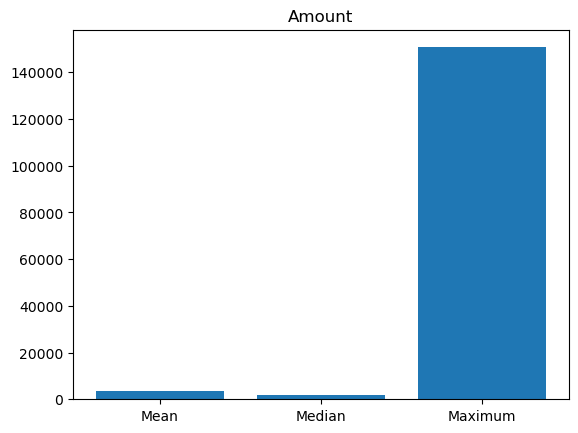
10 2021-01-08

Q4.

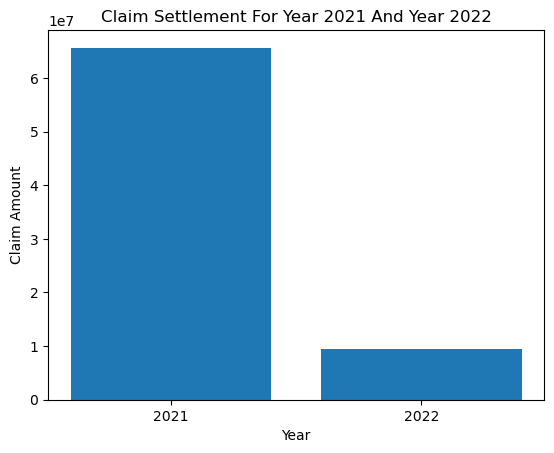
1) By analysing the dataset, we can find that most of the corporate claims are processed under internal regional code “LOC”, only a small part is proceeded under regional code “FVS”. The percentage of “LOC” is 92.1% while the percentage of “FVS” is 7.9%.



2) By analysing the data set, we can find the mean, median & maximum of claim amounts are $3336.71, $1633.28 & $150723.86 respectively. Below is the bar chart showing the comparison of these three figures.



3) By analysing the data set, we can find the total claim amount for Year 2021 and Year 2022 individually. The total claim amount for Year 2021 is $65621986.81 while the total claim amount for Year 2022 is $9535922.39. The total claim amount for Year 2021 is much higher than Year 2022.



Q5.

Linear regression is a statistical technique used to model the relationship between two variables. In this case, the two variables are “Planned” and “Actual” columns in the data set.

Before performing a linear regression, the data must be pre-processed to ensure it is suitable for modelling. Below are the steps that we can take to pre-process the data:

1. Convert the data columns to numerical data types so can be used in the regression model.

2. Data cleaning. This step is mainly about checking if got missing values in the data set. As missing values can cause issues in the linear regression model, we have to either remove or impute them.

3. Check for outliers. We need to check for outliers in the data, either remove them or transform them.

4. Check for multicollinearity. Multicollinearity occurs when two or more independent variables are highly correlated with each other. This can affect the accuracy of the regression model. We need to check for multicollinearity in the data, and either remove one of the correlated variables or transform them.

Once we have pre-processed the data, we can perform the following steps to perform linear regression:

1. Split the data into training and testing sets: We will use the training set to train the regression model, and the testing set to evaluate its performance.

2. Fit the linear regression model: We will fit a linear regression model to the training set, using the planned date as the independent variable and the delay as the dependent variable.

3. Evaluate the model: We will use the testing set to evaluate the performance of the model. We will use metrics such as the root mean squared error to evaluate the performance of the model.

4. Make predictions: Once we have a good model, we can use it to make predictions on new data.

The coding of applying linear regression is in Appendix for Q5

Q6.

The result obtained from the modelling is “Mean Squared Error: 8.071377136615683e-25”, indicates that the linear regression model fits the data very well as the error is very small. This means that the predicted values are very close to the actual values.

The linear regression equation obtained from the model is:

Delay = -2.842170943040401e-14 + 1.0000000000000004 \* Planned

This equation indicates the linear relationship between planned date and the actual date delay.

Appendix for Q1

# Import Library

import pandas as pd

import numpy as np

# Create a list of missing value types

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

# Display the columns with missing value

ECA = pd.read\_csv('C:\\Users\\Xiong Xiaofeng\\Desktop\\ANL252\\ECA.csv',na\_values = missing\_values)

ECA.isnull().any(axis = 0)

Output:

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

dtype: bool

Appendix for Q2

# Import Library

import pandas as pd

import numpy as np

# Create a list of missing value types

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

# Display the columns with missing value

ECA = pd.read\_csv('C:\\Users\\Xiong Xiaofeng\\Desktop\\ANL252\\ECA.csv',na\_values = missing\_values)

ECA.isnull().any(axis = 0)

# Remove rows where columns have missing value

columns\_with\_missing\_value = ['Claim\_ID','Actual','Terms']

ECA\_dropped = ECA.dropna(axis = 0, subset = columns\_with\_missing\_value)

ECA\_dropped.info()

Output:

<class 'pandas.core.frame.DataFrame'>

Int64Index: 22524 entries, 0 to 24212

Data columns (total 12 columns):

# Column Non-Null Count Dtype

--- ------ -------------- -----

0 Claim\_ID 22524 non-null float64

1 Policy\_No 22524 non-null int64

2 Name 22524 non-null object

3 Planned 22524 non-null object

4 Actual 22524 non-null object

5 Created 22524 non-null int64

6 Amount 22524 non-null object

7 Paid 22524 non-null object

8 Category 22524 non-null object

9 Terms 22524 non-null object

10 Region 22524 non-null object

11 Type 22524 non-null object

dtypes: float64(1), int64(2), object(9)

memory usage: 2.2+ MB

Appendix for Q3

1) Showing Basics Statistics

# Display the descriptive of the data set

ECA\_dropped.describe(include=object)

2) Showing how often specific values occur in a column

(Use column “Region” as example)

# Count the value of "Region" column

ECA\_dropped["Region"].value\_counts()

3) Convert the data column of the data frame from a string object to a data time object

(Use column “Actual” as example)

# Covert string date time into Python date time

ECA\_dropped["Actual"] = pd.to\_datetime(ECA\_dropped["Actual"])

ECA\_dropped["Actual"]

Appendix for Q4

1) Percentage of internal region classification code

# Import matplotlib

import matplotlib.pyplot as plt

# Input data size for "Region" column

sizes = [20736, 1788]

labels = ['LOC','FVS']

# Create pie chat

plt.pie(sizes, labels=labels, autopct='%1.1f%%', explode=[0,0], shadow=True, startangle=90)

plt.title('Percentage of internal region classification code')

plt.axis('equal')

plt.show()

2) Mean, Median and Maximum of “Amount” column

# Define values of x and y axis

x\_axis = ['Mean', 'Median','Maximum']

y\_axis = [3336.71, 1633.28, 150723.86]

# Create bar chart of "Amount"

plt.bar(x\_axis, y\_axis)

plt.title('Amount')

plt.show()

3) Claim settlement for Year 2021 and Year 2022

# Define values of x and y axis

x\_axis = ['2021', '2022']

y\_axis = [65621986.81, 9535922.394]

# Create bar chart of "Claim settlement"

plt.bar(x\_axis, y\_axis)

plt.title('Claim Settlement For Year 2021 And Year 2022')

plt.xlabel('Year')

plt.ylabel('Claim Amount')

plt.show()

Appendix for Q5

# Convert the dates to datetime format

ECA\_dropped['Planned'] = pd.to\_datetime(ECA\_dropped['Planned'], errors='coerce')

ECA\_dropped['Actual'] = pd.to\_datetime(ECA\_dropped['Actual'], errors='coerce')

# Check the data types

ECA\_dropped.info()

# Output

<class 'pandas.core.frame.DataFrame'>

Int64Index: 22524 entries, 0 to 24212

Data columns (total 12 columns):

# Column Non-Null Count Dtype

--- ------ -------------- -----

0 Claim\_ID 22524 non-null float64

1 Policy\_No 22524 non-null int64

2 Name 22524 non-null object

3 Planned 22524 non-null datetime64[ns]

4 Actual 22524 non-null datetime64[ns]

5 Created 22524 non-null int64

6 Amount 22524 non-null float64

7 Paid 22524 non-null object

8 Category 22524 non-null object

9 Terms 22524 non-null object

10 Region 22524 non-null object

11 Type 22524 non-null object

dtypes: datetime64[ns](2), float64(2), int64(2), object(6)

memory usage: 2.2+ MB

# Calculate the difference between two dates as a timedelta

Delay = ECA\_dropped['Actual'] - ECA\_dropped['Planned']

# Display the difference between dates

Delay.head()

# Output

0 90 days

1 -17 days

2 7 days

3 -4 days

4 90 days

Name: delay, dtype: timedelta64[ns]

# Import sklearn model

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

# Convert datetime to numerical data type

ECA\_dropped['Planned'] = (ECA\_dropped['Planned'] - pd.Timestamp('1970-01-01')) / pd.Timedelta('1 day')

ECA\_dropped['Actual'] = (ECA\_dropped['Actual'] - pd.Timestamp('1970-01-01')) / pd.Timedelta('1 day')

Delay = ECA\_dropped['Actual'] - ECA\_dropped['Planned']

# Perform linear regression

X = ECA\_dropped[['Planned', 'Actual']]

y = Delay

# Split the data into training and testing sets

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

# Choose linear regression model and train it on the traning data

lr\_model = LinearRegression()

lr\_model.fit(X\_train, y\_train)

# Test the model on the testing data and find the mean squared error

y\_pred = lr\_model.predict(X\_test)

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

print("Mean Squared Error:", mse)

# Output

Mean Squared Error: 8.071377136615683e-25