

**ANL252 (Online)**

**Python For Data Analytics**

# **End-of-Course Assessment**

**January 2023**

**Submitted by:**

|  |  |
| --- | --- |
| **Name** | **PI No.** |
| **Ng Zi Ya** | **M2272807** |

**Tutorial Group: ­­­­­­­­­­­­ T 05**

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

**Q1**

Claim\_ID and actual contains missing values which is represented by NaN. Which means Not A Number.

**Q2**

There are three ways to treat the missing data. To delete the entire row, replace them or ignore them. I chose to replace the missing data with 0 and not delete the entire row, as I felt that it would affect the accuracy if I were to count the amount of data.

First to filter out the data with missing values using “.isna()”. Once I got the variables that are missing, I then replace the missing values with “0” using“.fillna()”

*claims[claims.isna().any(axis=1)]*

*Then to replace the missing value with “0”*

*claims.fillna(value = 0)*

**Q3**

Besides handling missing data. First data preparation task is to select and renaming the variables. This is to prepare the selected variables that we will be using for our analysis. It will be easier to us to plot out the graph and charts later on. To do this we have to select rows and columns from the pandas DataFrame using index, Boolean masks, and localisation.

In this case I used:

*time\_taken= claims[["Created","Actual","Planned"]]*

*time\_taken*

to filter out the columns that I prepare to analyse.

I would get the following result.

|  | **Created** | **Actual** | **Planned** |
| --- | --- | --- | --- |
| **0** | 20210112 | 18/1/2021 0:00 | 17/1/2021 |
| **1** | 20210130 | 16/1/2021 0:00 | 5/2/2021 |
| **2** | 20210113 | 14/1/2021 0:00 | 18/1/2021 |
| **3** | 20210110 | 18/1/2021 0:00 | 15/1/2021 |
| **4** | 20210131 | 8/2/2021 0:00 | 5/2/2021 |
| **...** | ... | ... | ... |
| **24208** | 20210328 | 27/12/2021 0:00 | 30/3/2021 |
| **24209** | 20210315 | 31/5/2021 0:00 | 18/3/2021 |
| **24210** | 20210318 | 23/5/2021 0:00 | 21/3/2021 |
| **24211** | 20210329 | 22/5/2021 0:00 | 1/4/2021 |
| **24212** | 20210331 | 8/5/2021 0:00 | 4/4/2021 |

To retrieve the case that has been paid

*processed= claims[["Created","Paid"]]*

*processed*

| **Created** | **Paid** |
| --- | --- |
| **0** | 20210112 | Yes |
| **1** | 20210130 | Yes |
| **2** | 20210113 | Yes |
| **3** | 20210110 | Yes |
| **4** | 20210131 | Yes |
| **...** | ... | ... |
| **24208** | 20210328 | Yes |
| **24209** | 20210315 | Yes |
| **24210** | 20210318 | Yes |
| **24211** | 20210329 | Yes |
| **24212** | 20210331 | Yes |

Second, create dummy variables. The dummy variables is converted from the categorical variables before they can be evaluated and included in the computation of the scikit-learn algorithms. It only has 2 value, 0 and 1, if the observation belongs to one of the category it will be labelled as 1.

I use.

*claims\_dummy = pd.get\_dummies(filtered)*

*claims\_dummy*

to get dummy variables for the selected variables that I would be analyzing later.

I would get the following result.

|  | **Created** | **Paid\_No** | **Paid\_Yes** |
| --- | --- | --- | --- |
| **0** | 20210112 | 0 | 1 |
| **1** | 20210130 | 0 | 1 |
| **2** | 20210113 | 0 | 1 |
| **3** | 20210110 | 0 | 1 |
| **4** | 20210131 | 0 | 1 |
| **...** | ... | ... | ... |
| **24208** | 20210328 | 0 | 1 |
| **24209** | 20210315 | 0 | 1 |
| **24210** | 20210318 | 0 | 1 |
| **24211** | 20210329 | 0 | 1 |
| **24212** | 20210331 | 0 | 1 |

Lastly, to Change String to Date using pandas.to\_datetime so that the format would be easier for us to use later. In this case I changed the “Created”, “Actual” and “Planned” column format individually.

Firstly, convert Planned to datetime format with the following code:

*claims["Planned"] = pd.to\_datetime(time\_taken["Planned"], format="%d/%m/%Y")*

*claims*

|  | **Claim\_ID** | **Policy\_No** | **Name** | **Planned** | **Actual** | **Created** | **Amount** | **Paid** | **Category** | **Terms** | **Region** | **Type** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **0** | 2.928510e+09 | 300764795 | Roger Torres | 2021-01-17 | 18/1/2021 0:00 | 20210112 | 3072.349 | Yes | AT | AD23 | LOC | L001 |
| **1** | 2.928511e+09 | 300434439 | Jason Jones | 2021-02-05 | 16/1/2021 0:00 | 20210130 | 910.944 | Yes | AT | EC05 | LOC | L001 |
| **2** | 2.928517e+09 | 300769623 | Robert Martin | 2021-01-18 | 14/1/2021 0:00 | 20210113 | 567.936 | Yes | AT | AB27 | LOC | L001 |
| **3** | 2.928517e+09 | 300794332 | Stacy Anderson | 2021-01-15 | 18/1/2021 0:00 | 20210110 | 181.651 | Yes | AT | AE14 | LOC | L001 |
| **4** | 2.928518e+09 | 300792283 | Mr. Adam Whitaker III | 2021-02-05 | 8/2/2021 0:00 | 20210131 | 238.74 | Yes | AT | EC05 | LOC | L001 |
| **...** | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... |
| **24208** | NaN | 200030194 | Daniel Davis | 2021-03-30 | 27/12/2021 0:00 | 20210328 | 188.4 | Yes | XT | CB91 | FVS | O001 |
| **24209** | NaN | 200054349 | Jennifer Thomas | 2021-03-18 | 31/5/2021 0:00 | 20210315 | 460.8 | Yes | XT | CB91 | FVS | O001 |
| **24210** | NaN | 200030194 | Amber Newton | 2021-03-21 | 23/5/2021 0:00 | 20210318 | 120.735 | Yes | XT | CB91 | FVS | O001 |
| **24211** | NaN | 240104429 | Marcus Hernandez | 2021-04-01 | 22/5/2021 0:00 | 20210329 | 591.12 | Yes | XT | CB91 | FVS | O001 |
| **24212** | NaN | 240106379 | James Fernandez | 2021-04-04 | 8/5/2021 0:00 | 20210331 | 837 | Yes | XT | CB91 | FVS | O001 |

*claims.dtypes*

Claim\_ID float64

Policy\_No int64

Name object

Planned datetime64[ns]

Actual datetime64[ns]

Created datetime64[ns]

Amount object

Paid object

Category object

Terms object

Region object

Type object

dtype: object

Second, convert Actual to datetime format

*claims["Actual"] = pd.to\_datetime(time\_taken["Actual"])*

*claims.dtypes*

Claim\_ID float64

Policy\_No int64

Name object

Planned datetime64[ns]

Actual datetime64[ns]

Created datetime64[ns]

Amount object

Paid object

Category object

Terms object

Region object

Type object

dtype: object

Lastly, convert Created to datetime format

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

*claims.dtypes*

Claim\_ID float64

Policy\_No int64

Name object

Planned datetime64[ns]

Actual datetime64[ns]

Created datetime64[ns]

Amount object

Paid object

Category object

Terms object

Region object

Type object

dtype: object

**Q4**

**Graph (Pie Chart):** To calculate how many percent of claims has been paid.

*sums2 = claims.groupby(claims["Paid"])["Paid"].count()*

*pie(sums2, labels=sums2.index, autopct='%0.01f%%')*

*labels = [r'Paid', r'Not Paid']*

*colors=['orange','blue']*

*sizes = [93.1, 6.9]*

*patches, texts = plt.pie(sizes, colors=colors, startangle=90)*

*plt.legend(patches, labels, loc="best")*

*show()*

Chart, pie chart

Description automatically generated

*print(sums2)*

Paid

No 1677

Yes 22536

Name: Type, dtype: int64

*sums3 = claims.groupby(claims["Type"])["Type"].count()*

*pie(sums3, labels=sums3.index, autopct='%0.01f%%')*

*labels = [r'L001', r'L002',r'L003',r'L004',r'L005',r'O001']*

*colors=['lightblue','purple','red','brown','yellow','pink']*

*sizes = [21977, 69,7,1,247,1912]*

*patches, texts = plt.pie(sizes, colors=colors, startangle=90)*

*plt.legend(patches, labels, loc="best")*

*show()*

Chart, pie chart

Description automatically generated

*print(sums3)*

Type

L001 21977

L002 69

L003 7

L004 1

L005 247

O001 1912

Name: Type, dtype: int64

*claims.plot(x='Planned', y='Actual', style='o')*

*plt.title('Planned VS Actual')*

*plt.xlabel('Planned')*

*plt.ylabel('Actual')*

*plt.show()*

**Chart, scatter chart

Description automatically generated**

**Insight:**

First insight, the insurance company is efficient in processing their claims as they had paid 93.1% of the claims, which is equivalent to 22536 claims out of the 24214 claims. At the same time, the remaining 6.9% is equal to 1677 claims that have not been processed and paid. In the dataset, there are 21977 claims for the L001 type of claims, which makes up 90.8% of the claims cases. The second insight, the rest of the types are L002, L003, L004, L005, and O001, which have 69, 7, 1, 247, and 1912 claims cases, respectively. Third insight, there is a strong relationship between “Planned Date” and “Actual Date” as the pattern of the graph is inclining towards the top right. In the scatter plot, a claim case plans to pay on 2021-12, but the actual pay date was 2022-04. The scatter plot for this case means it took four months to process this claim. This claim case took the longest to process compared to those planned in 2021-12.

**Q5**

*#to lessen the data used to 2021-01*

*claims['year-month'] = pd.to\_datetime('2021-' + '1-' + claims['Planned'].dt.strftime('%d'))*

*print(claims['year-month'])*

*#to plot the scatter plot*

*ax = claims.plot(kind='scatter', x='Actual', y='year-month')*

*#title*

*plt.title('Planned VS Actual')*

*#x label*

*plt.xlabel('Actual')*

*#y label*

A picture containing chart

Description automatically generated*plt.ylabel('Planned')*

*claims['year'] = pd.to\_datetime('2021-' + '1-' + claims['Planned'].dt.strftime('%d'))*

*claims['count']=claims.year.value\_counts().astype(float)*

*claims.year.value\_counts().plot(kind='bar')*

*plt.title('Number of Planned cases')*

*plt.xlabel('Planned')*

*plt.ylabel('No.of Cases')*

Chart

Description automatically generated

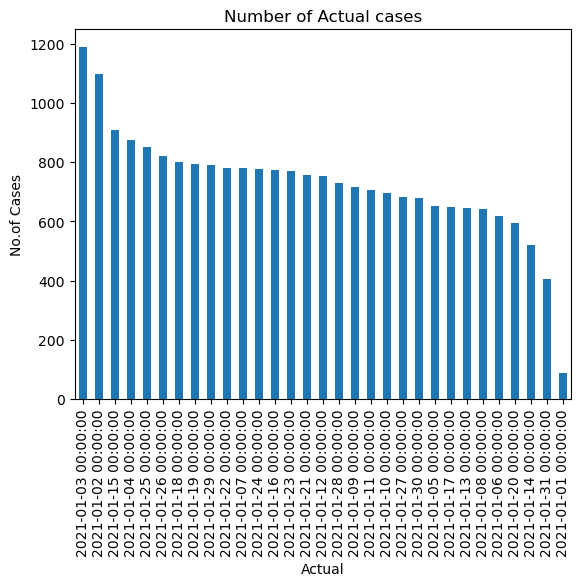
*claims['year1'] = pd.to\_datetime('2021-' + '1-' + claims['Actual'].dt.strftime('%d'))*

*claims.year1.value\_counts().plot(kind='bar')*

*plt.title('Number of Actual cases')*

*plt.xlabel('Actual')*

*plt.ylabel('No.of Cases')*



Firstly, as this is big data file, I would filter out some of the data for easy plotting of regression model. I decided to use 2021 January as the data to perform the linear regression modelling. To filter the data, I used pd.to\_datetime to filter 2021 January, and named it year-month. After which we had to plot the scatter plot by using year-month as my y-axis. Lastly, would be to label my graph with title, y and x-axis. The reason for using bar graph for my model is to have a clearer picture of the delays in for each date.

**Q6**

From the first graph, we can tell there is no correlation between the Planned and Actual data as the chart is spread out evenly, with no incline to the left or right side. For example, on 2021-01-26, the insurance company paid the claim case on 2022-12. Which shows that it took more than ten months to process the claims. Another example is that the claims to be paid on 2021-01-23 got paid on 2022-12 instead. This takes about ten months to process the claim. However, some cases got paid on the planned date, like 2021-01-01. The second and third graph shows a decrease in the number of actual cases paid compared to the number of cases planned. For example, in 2021-01-01, the number of cases planned to pay was about 700. However, the actual number of cases paid is only about 100. This means 600 claims cases still need to be paid. Another example is

2021-01-27, where the planned date is 800 cases, but on the actual date, only 700 claims cases have been paid, delaying 100 claims cases not processed. To sum it up, the graphs show no relationship between the planned and actual date, as the processing duration might be based on the type of claims, sums, category and region.