Logo, company name

Description automatically generated

**ANL252**

**Python for Data Analytics**

**ECA**

**January 2023 Presentation**

|  |  |
| --- | --- |
| Name | Chan Huan Hueng, Brandon |
| Class | T05 |
| PI no. | J1981618 |
| Date | 6 March 2023 |

**Question 1**

After reading in dataset as a Pandas dataframe, the variables that contain missing values are “Claim\_ID”, “Actual”, and “Terms”.

**Question 2**

As part of data preparation, I decided to delete the missing data in this dataset. My rationale of this treatment is because the missing data might be fundamental to the corporate claims processing of this insurance company. As we do not have enough information whether each different variable can affect the claims processing, I decided that deleting the rows that had missing data was the safe solution and replacing or filling in the missing data was not ideal. For example, “Terms” was one of the variables that contained missing values. According to the data dictionary in Appendix 1, since it is an internal terms and conditions code, different terms might mean different claims processes. This can be said the same for “Claim\_ID” as well. Having null value for “Claim\_ID” is not desirable when dealing with insurance, as it is a unique identifier of the claim itself.

For “Actual”, it is even more appropriate to delete these rows because it represents the actual date of the settlement. When further investigated, the actual date of claim settlement has a null value most possibly due to the fact that the claims have not been settled yet. This can be evidenced by the “Paid” column, which shows “No” for the rows that had null values in the “Actual” column. Therefore, since it is likely the claims are not settled yet, it will not be correct to add these rows in. It will also not be wise to fill in the missing values with “fillna()” as this might create a biased analysis since it uses preceding or succeeding values in the same column.

**Question 3**

The first data preparation task is to convert columns into their expected data types. In this particular dataset, the columns “Planned”, “Actual” and “Created” needs to be converted into a datetime format. This is to ensure that all 3 columns are running the same formatting.

The second data preparation task is to check for unusual or in the amount for the claims. Since the dataset is with regards to insurance claims, the amount involved is of utmost importance in this data preparation. With this preparation, there was a row which had a value of “1762.OO” in the “Amount” column, which needed correction, to change it to “1762.00”.

This second data preparation task leads us to the third preparation task. Since the “Amount” column involves numeric values, if “1762.OO” was being able to be typed in, there must be an error in the data variable type. Therefore, checking the variable data type is the last data preparation task. After checking, it was indeed true that the “Amount” column was wrong and it was changed.

**Question 4**

**Analysis 1**

Chart, line chart

Description automatically generated

As seen in Analysis 1, we can observe the following insights:

* From the graph above, when comparing the total amount of Actual claims per month of 2021 and 2022, we can observe that for the year 2022 there were significantly a lot less claims as compared to the year 2021.
* As this dataset is with regards to corporate claims processing of an insurance company, it is likely that the claimants are from employees of a company. We can deduce that there were significantly a lot more claims in the year 2021 due to COVID-19 since 2021 was the year where it hit hard.
* It is highly likely that these employees were more prone to falling sick and seeing the doctors more often which explains the vast difference in actual claims between the 2 years.

**Analysis 2**

**Chart, bar chart

Description automatically generated**

As seen in analysis 2, we can observe the following insights:

* From the bar chart above, we are able to observe the top 10 claimants of insurance per frequency count. It shows 10 persons who had the highest number of claims throughout.
* This insight might be valuable to the insurance company as it is able to show the specific persons who are claiming insurance so frequently and can consider looking into it to ensure the validity of the claims.

**Analysis 3**

Chart, bar chart

Description automatically generated

As seen in analysis 3, we can observe the following insights:

* From the bar graph above, we can observe the top 10 claimants by the mean amount. This graph shows the mean amount of money claimed derived from the total amount claimed divided by the frequency/number of claims.
* We can deduce that the individual “Daniel Brown” had claimed a substantial amount which was more than $140,000.
* In comparison to other mean amounts by other claimants, this is a relatively large amount as the second mean amount was just over $20,000.
* With such an immense difference in amount, it might be beneficial to look into this particular individual, to ensure and check the validity of his claim.

**Question 5**

To perform a linear regression model to predict the delay in processing the claims, there are a few steps needed to be done.

First will be to import all the relevant libraries that are needed to perform this model. For this situation in particular, the libraries that will be needed will be pandas, matplotlib, seaborn and sklearn.linear\_model. Checking the data types of the variables that you are using is fundamental as well, as it is important to convert it to another data type if necessary. In this example, since we will be using dates, we need to convert it to a datetime format. Converting the data type is also essential because we are trying to compute a delay between the actual date and planned date to derive the “delay” in days taken to process these insurance claims. To do this, we create a new column “Delay” by subtracting “Planned” column from “Actual”column.

Since linear regression requires numerical inputs, the usage of dummy variables are applied here. This way, we are able to evaluate the data even though the variables that are used are not in numerical inputs.

The next step is the splitting of data into training and testing sets, which is to evaluate the performance on untouched data. This will help to prevent overfitting while being able to evaluate our data. The most common ratio that are used is 70% of the data for training dataset while 30% for testing**Chart, scatter chart

Description automatically generated**

As seen in the above figure, this is the scatter plot that will be created after the performing the linear regression model. We are able to observe that the x-axis represents the actual delay values, while the y-axis represents the predicted delay values.

**Question 6**

As seen in Question 5, my linear regression model was not able to deduce or obtain a result that was useful to predict the delay in days in processing the claims.

**Appendix 1**

**Remarks:** This is the code for Question 1 but import of libraries applies to all other appendix as well

#import of all the libraries

import pandas as pd

import statsmodels.api as sm

from sklearn import linear\_model

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

data= "ECA.csv"

import seaborn as sns

import matplotlib.pyplot as plt

import numpy as np

from sklearn.linear\_model import LinearRegression

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

from sklearn.model\_selection import cross\_val\_score, cross\_val\_predict

#loading dataset into a pandas dataframe  
claims=pd.read\_csv(data)

#to find out which variables have missing data

for i,x in claims.iterrows():

for g in x:

if (g=='???') | (g=='Unkn'):

print(x,g)

# Checking for null values  
claims.isnull().any()

**Appendix 2**

**Remarks:** This is the code for Question 2

#To see rows that has null values in the “Claim\_ID” column  
claims[claims.Claim\_ID.isnull()]

#To see the rows that has null values in the “Actual” column  
claims[claims.Actual.isnull()]

#To drop and delete the rows that contain null values  
claims.dropna(inplace=True)

#To drop and delete the rows that contain “???” and “Unkn” values  
claims.drop(claims[(claims.Terms == "???") | (claims.Terms == "Unkn")].index, inplace=True)

**Appendix 3**

**Remarks:** This is the code for question 3

#Convert the dates columns to datetime format for the first task  
claims.Planned = pd.to\_datetime(claims.Planned)

claims.Actual = pd.to\_datetime(claims.Actual)

claims.Created = pd.to\_datetime(claims.Created, format="%Y%m%d")

#This is to check the unusual numbers in the Amount column for the second task  
for x in claims.Amount:

try:

x=float(x)

except:

print(x)

#Locate the row with the error  
claims[claims.Amount == "1762.OO"]

#Change the amount to a float  
claims.loc[3698, "Amount"]=1762.00

#To check the data type of the columns  
claims.info()

#To convert Amount column to float  
claims.Amount=claims.Amount.astype(float)

**Appendix 4**

**Remarks:** This is the code for Question 4

#Check how many years in the dataset  
claims.Actual.dt.year.value\_counts()

#Add a new column called “delay” by subtracting the “Planned” column from “Actual” column  
claims['delay']=claims.Actual-claims.Planned

#Convert delay datetime to days in float/int for easier use  
claims.delay=claims.delay.dt.days

#Create a copy to avoid modifying original  
seasonal\_claims=claims.copy()

#Get month out of the datetime  
seasonal\_claims['Actual\_month']=seasonal\_claims.Actual.dt.month

seasonal\_claims['Planned\_month']=seasonal\_claims.Planned.dt.month

#Splitting it into the different years  
seasonal\_claims2021=seasonal\_claims[seasonal\_claims.Actual.dt.year==2021]

seasonal\_claims2022=seasonal\_claims[seasonal\_claims.Actual.dt.year==2022]

#Plotting of the graph for Analysis 1  
fig, ax = plt.subplots(figsize = (15,7))

ax.set\_ylabel('Total Number of Claims', fontsize=15)

ax.set\_xlabel('Month', fontsize=15)

ax.set\_title('Total number of Actual claims per month per year', fontsize=22)

seasonal\_claims2021.groupby('Actual\_month').count()["Planned"].plot(ax=ax)

seasonal\_claims2022.groupby('Actual\_month').count()["Planned"].plot(ax=ax)

#Plotting of bar graph for Analysis 2  
fig, ax = plt.subplots(figsize = (15,7))

ax.set\_ylabel('Number of claims', fontsize=15)

ax.set\_xlabel('Name', fontsize=15)

ax.set\_title('Top 10 Claimants per count', fontsize=22)

claims.groupby('Name').count().sort\_values('Amount').tail(10).Amount.plot(ax=ax, kind='bar')

#Create a copy to avoid modifying original  
name\_claims=claims.copy()

#Plotting of bar graph for Analysis 3

fig, ax = plt.subplots(figsize = (15,7))

ax.set\_ylabel('Mean Amount', fontsize=15)

ax.set\_xlabel('Name', fontsize=15)

ax.set\_title('Top 10 Claimants by mean Amount', fontsize=22)

name\_claims.groupby("Name").mean("Amount").sort\_values("Amount").Amount.tail(10).plot(ax=ax, kind='bar')

**Appendix 5**

**Remarks:** This is the code for Question 5

#Create new column ‘year’  
claims['year']=claims.Created.dt.year

#Choosing selection of columns   
X2=claims[['Amount','Region','Type','year']]

#Create dummy variables  
X3=pd.get\_dummies(data=X2,drop\_first=True)

#Splitting of training and testing set  
X\_train=X3[X3.year==2021]

Y\_train=claims[claims.year==2021]['delay']

X\_test=X3[X3.year==2022]

Y\_test=claims[claims.year==2022]['delay']

#Performing of linear regression model  
lm=LinearRegression()

model=lm.fit(X\_train,Y\_train)

Y=model.predict(X\_test)

print(model.score(X\_test,Y\_test))

#plotting of the linear regression model  
sns.jointplot(Y\_test, Y)