

**ANL252 Python for Data Analytics**

**End Course Assignment (ECA01)**

**January 2023 Presentation**

TEY JIN HEAN K1810665

1)

In[5]

import pandas as pd

# Read in the file from the CSV located in Downloads

df = pd.read\_csv(r'C:\Users\Jin Hean\Downloads\ECA.csv')

# search for invalid values denoted by 'Unkn' or '???'

invalid\_values = ['Unkn', '???']

df.replace(invalid\_values, pd.NA, inplace=True)

# Check for invalid values in each column

invalid\_cols = df.isna().any()

print(invalid\_cols)

# Print out the specific columns with invalid values

invalid\_vars = invalid\_cols.loc[invalid\_cols]

print(invalid\_vars)

Out[5]

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

Claim\_ID True

Actual True

Terms True

dtype: bool

2)

To clean the dataset, we just eliminate all rows with invalid inputs (i.e. with unkn, ???) as they are considered incomplete data.

In[6]

# Drop rows with any missing values

df.dropna(inplace=True)

3)

First step will be to transform both “Created” and “Actual” to Datetime objects as they are currently in different formats. We will also convert “Amount” to Numeric objects, and “Region” to categorical Objects. We are doing these Data type conversions to make future manipulation of the data simpler.

In[7]

# Convert 'Created', “Actual’ columns to datetime objects

df['Created'] = pd.to\_datetime(df['Created'])

df['Actual'] = pd.to\_datetime(df['Actual'])

# Convert 'Amount' column to numeric objects

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

# Convert 'Region' column to categorical objects

df['Region'] = df['Region'].astype('category')

Second step is to create a new variable “time\_diff” to identify the delay between the planned and actual date of settlement, for future data manipulation:

In[16]

# Delay between 'Planned' and 'Actual', as a new variable

df['Time\_Diff'] = (df['Actual'] - df['Planned']).dt.days

df['Category\_First\_Letter'] = df['Category'].str[0]

4)

The payout amount is a significant variable to analyse, as it give us a view into how severe the claims are, that was processed by the Insurance firm.

We can see a distribution of the payout amount by plotting a histogram between the payout amount and the Count:

In[31]  
  
import matplotlib.pyplot as plt

plt.hist(df['Amount'], bins=20)

plt.xlabel('Payout Amount')

plt.ylabel('Count')

plt.title('Distribution of Payout Amount')

plt.show()  
  
Chart

Description automatically generated

Classifying claim counts by region would be useful for budgeting purposes, utilising the region classification code.  
  
# using the plot(kind=’bar’) to plot a bar chart

df['Region'].value\_counts().plot(kind='bar')

plt.xlabel('Region')

plt.ylabel('Claim Count')

plt.title('Claim Counts by Region')

plt.show()

Chart, bar chart

Description automatically generated

5)

In[1]

# importing all necessary libraries

import os

import pandas as pd

import matplotlib.pyplot as plt

from sklearn.linear\_model import LinearRegression

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

# *sklearn.linear\_model.LinearRegression*. (n.d.).

In[2]

# redefine data\_filepath

data\_filepath = os.path.join("data", r'C:\Users\Jin Hean\Downloads\ECA.csv')

In[3]

df = pd.read\_csv(data\_filepath)

In[4]

df.shape

# identifying number of rows and columns

Out[4]

(24213, 12)

In[5]

df.head

# inspecting top of csv

Out[5]

Claim\_IDPolicy\_NoNamePlannedActualCreatedAmountPaidCategoryTermsRegionType02.928510e+09300764795Roger Torres17/1/202118/1/2021 0:00202101123072.349YesATAD23LOCL00112.928511e+09300434439Jason Jones5/2/202116/1/2021 0:0020210130910.944YesATEC05LOCL00122.928517e+09300769623Robert Martin18/1/202114/1/2021 0:0020210113567.936YesATAB27LOCL00132.928517e+09300794332Stacy Anderson15/1/202118/1/2021 0:0020210110181.651YesATAE14LOCL00142.928518e+09300792283Mr. Adam Whitaker III5/2/20218/2/2021 0:0020210131238.74YesATEC05LOCL001

In[6]

df.tail()

# Inspecting bottom of csv

Out[6]

laim\_IDPolicy\_NoNamePlannedActualCreatedAmountPaidCategoryTermsRegionType24208NaN200030194Daniel Davis30/3/202127/12/2021 0:0020210328188.4YesXTCB91FVSO00124209NaN200054349Jennifer Thomas18/3/202131/5/2021 0:0020210315460.8YesXTCB91FVSO00124210NaN200030194Amber Newton21/3/202123/5/2021 0:0020210318120.735YesXTCB91FVSO00124211NaN240104429Marcus Hernandez1/4/202122/5/2021 0:0020210329591.12YesXTCB91FVSO00124212NaN240106379James Fernandez4/4/20218/5/2021 0:0020210331837YesXTCB91FVSO001

In[7]

# identifying nature of variables in csv

df.dtypes

Out[7]

Claim\_ID float64

Policy\_No int64

Name object

Planned object

Actual object

Created int64

Amount object

Paid object

Category object

Terms object

Region object

Type object

dtype: object

In[10]

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

df = df[['Amount', 'Delay']]

df = df[df['Amount'] <= 40000]

# Creating a new variable, known as Delay, which is the difference between Planned and Actual

# proceed to show df

Df

Out[10]

| **Amount** | **Delay** |
| --- | --- |
| **0** | 3072.349 | 1.0 |
| **1** | 910.944 | 106.0 |
| **2** | 567.936 | 4.0 |
| **3** | 181.651 | 3.0 |
| **4** | 238.740 | 92.0 |
| **...** | ... | ... |
| **24208** | 188.400 | 272.0 |
| **24209** | 460.800 | 74.0 |
| **24210** | 120.735 | 63.0 |
| **24211** | 591.120 | 138.0 |
| **24212** | 837.000 | 123.0 |

24211 rows × 2 columns

In[14]

df = df.dropna()

# drops invalid fields

df.shape

#number of columns and rows after dropping invalid fields

Out[14]

(22534, 2)

In[16]

# specifying parameters to be trained using SciKit\_learn

X\_train, X\_test, y\_train, y\_test = train\_test\_split(df['Amount'], df['Delay'], test\_size=0.2, random\_state=42)  
  
X\_train = pd.DataFrame(X\_train)

X\_test = pd.DataFrame(X\_test)

In[17]

print(f"Train rows: {X\_train.shape[0]}")

print(f"Test rows: {X\_test.shape[0]}")

# checking for number of train vs test rows, after specifying and splitting the test and train datasets

Out[18]

Train rows: 18027

Test rows: 4507

In[19]

plt.scatter(X\_train, y\_train)

plt.show()  
  
# Show scatter plot of training against testing data

Out[19]

Chart, scatter chart

Description automatically generated

In[20]

model = LinearRegression()

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

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

model.predict([[460.800]])

#Predict the time delay according to Linear regression model

Out[25]

array([44.44199229])

Citations:

*sklearn.linear\_model.LinearRegression*. (n.d.). Scikit-learn. https://scikit-learn.org/stable/modules/generated/sklearn.linear\_model.LinearRegression.html