****

**ANL 252**

**PYTHON FOR DATA ANALYTICS**

**End-of-Course Assessment**

**January 2023 Presentation**

**Submitted by:**

|  |  |
| --- | --- |
| **Name** | **PI Number** |
| Chong Xin Hui Sandy | W2110447 |

**Tutorial Group: T05**

**Instructor’s Name: MUNISH KUMAR**

**Submission Date: 06 March 2023**

**Question 1**

Jupyter Python code:-

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

import seaborn as sns

# Question 1

# Load the ECA dataset and replace missing values with NaN

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

# Find columns with missing values and print their names

print("Columns with missing values:")

for col in eca\_df.columns:

if eca\_df[col].isna().sum() > 0:

print(col)

which resulted in following python output:-

Columns with missing values:

Claim\_ID

Actual

Terms

**Question 2**

Jupyter Python code:-

# Question 2

# Drop rows where Actual is missing since we cannot predict the payout amount

eca\_na = eca\_df.dropna(subset=["Actual"]).copy()

# Find rows where Claim\_ID is missing and replace with next sequential ID

missing\_claim\_ids = eca\_na["Claim\_ID"].isna()

last\_claim\_id = eca\_na.loc[~missing\_claim\_ids, "Claim\_ID"].max()

eca\_na.loc[missing\_claim\_ids, "Claim\_ID"] = range(

last\_claim\_id + 1, last\_claim\_id + 1 + missing\_claim\_ids.sum()

)

# Find the most probable Term value and replace missing values

most\_probable\_term = eca\_na["Terms"].value\_counts().idxmax()

eca\_na["Terms"] = eca\_na["Terms"].fillna(most\_probable\_term)

# Check the number of missing values

num\_missing = eca\_na.isna().sum().sum()

0

**Question 3**

Jupyter Python code:-

# Question 3

# Convert date columns to datetime

eca\_na["Planned"] = pd.to\_datetime(eca\_na["Planned"],

format="%d/%m/%Y")

eca\_na["Created"] = pd.to\_datetime(eca\_na["Created"], format="%Y%m%d")

eca\_na["Actual"] = pd.to\_datetime(eca\_na["Actual"], format="%d/%m/%Y %H:%M")

# Combine rare terms into a new category called "UC01"

term\_counts = eca\_na["Terms"].value\_counts(normalize=True)

rare\_terms = term\_counts[term\_counts < 0.01].index

eca\_na.loc[eca\_na["Terms"].isin(rare\_terms), "Terms"] = "UC01"

# Convert Amount column to float and replace 'O' with '0'

eca\_na["Amount"] = eca\_na["Amount"].str.replace("O",

"0").astype(float)

**Question 4**

Jupyter Python code for the first data insight:-

# Question 4.1

# Calculate the average claim amounts per policy term

avg\_claim\_amounts = eca\_na.groupby("Terms")["Amount"].mean()

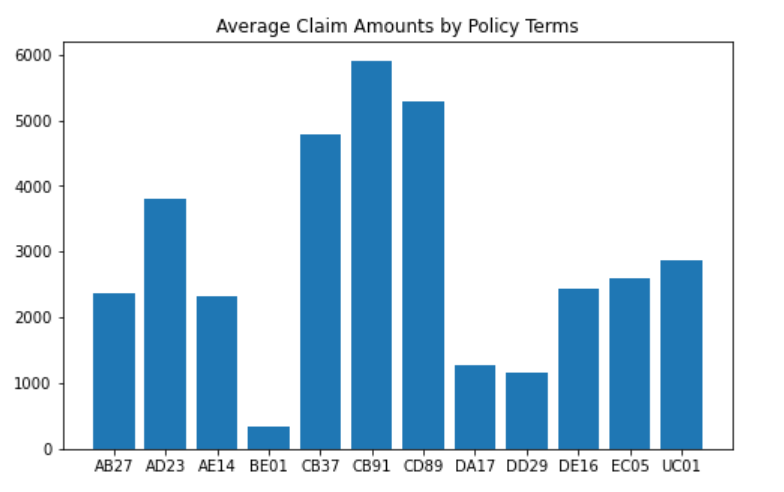
# Plot the results

plt.figure(figsize=(8, 5))

plt.title("Average Claim Amounts by Policy Terms")

plt.bar(x=avg\_claim\_amounts.index, height=avg\_claim\_amounts.values)

plt.show()



The average claim amounts for each term varied dramatically.

As evidenced by the above graph, BE01, DD29 and DA17 claims have a relatively low average claim amount.On the other hand, CB37, CB91 and CB89 claims have a high claim amount.

If the claim amount influences the time it takes the corporation to pay out the claim, we may have multicollinearity in the data, which we will further analyse below.

Jupyter Python code for the second data insight:-

# Question 4.2

# Calculate claim delay in days and plot it against claim amount

eca\_na["Claim\_Delay"] = (eca\_na["Actual"] - eca\_na["Planned"]).dt.days

plt.figure(figsize=(8, 5))

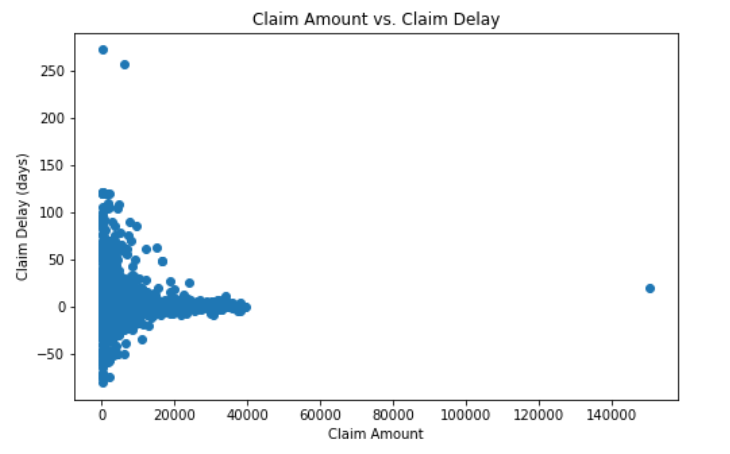
plt.title("Claim Amount vs. Claim Delay")

plt.scatter(x=eca\_na["Amount"], y=eca\_na["Claim\_Delay"])

plt.xlabel("Claim Amount")

plt.ylabel("Claim Delay (days)")

plt.show()



We can observe that claim amount does influences payout delay to a limited extent. The bigger the claim amount, the closer it is to zero. This implies that when the claim amount is significant, the insurance company is more concerned about timeliness.

We may also deduce that words with a high average claim amount, such as CB37 and CB91, are more likely to be on time, whereas terms with a low average claim amount, such as BE01, are more likely to be either early or late.

Jupyter Python code for the third data insight:-

# Question 4.3

# Check if the average claim amount differs significantly by other variables as well

for col in ["Type", "Region", "Category"]:

avg\_claim\_amt\_by\_col = eca\_na.groupby(col)["Amount"].mean()

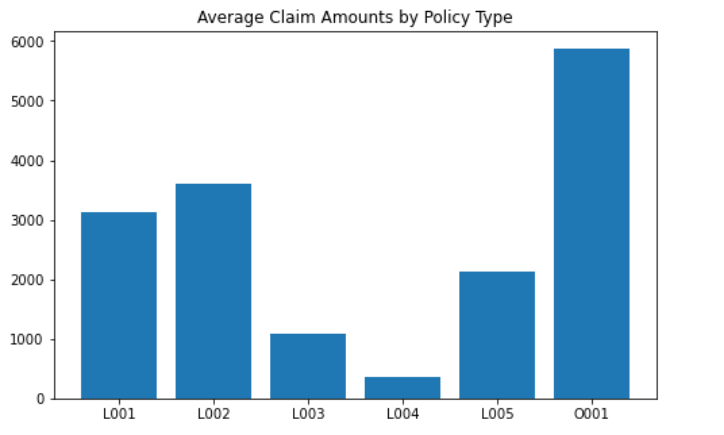
plt.figure(figsize=(8, 5))

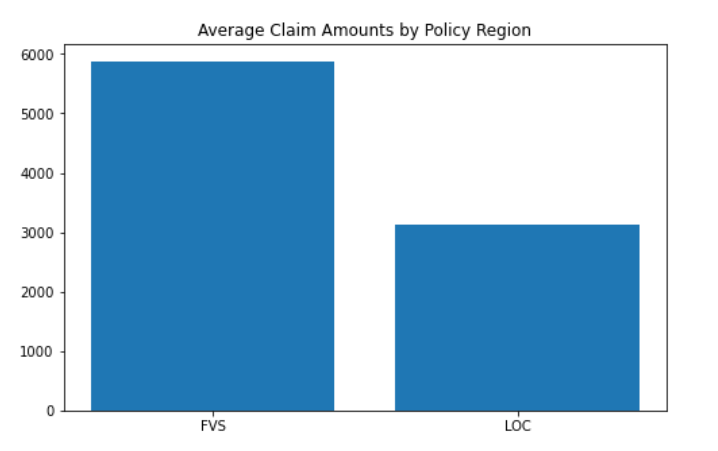
plt.title(f"Average Claim Amounts by Policy {col}")

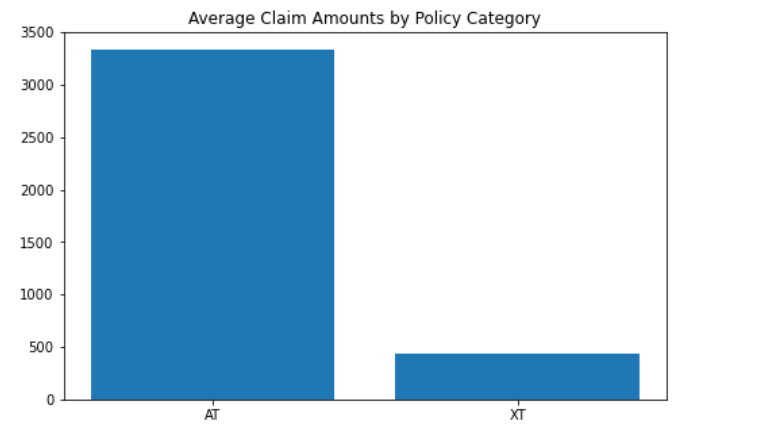
plt.bar(x=avg\_claim\_amt\_by\_col.index,

height=avg\_claim\_amt\_by\_col.values)

plt.show()







We can observe that there is a significant difference between average claim amounts in each class of variable for each category variable. This provides us confidence that the data has enough variation for the model to learn.

**Question 5**

Jupyter Python code:-

# Question 5

import pandas as pd

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

from sklearn.preprocessing import MinMaxScaler

# Prepare the data for regression

reg\_data = eca\_na.drop(columns=["Claim\_ID", "Policy\_No", "Name",

"Planned", "Actual", "Created"])

reg\_data = pd.get\_dummies(reg\_data)

reg\_variables = reg\_data.drop(columns="Delay")

reg\_target = reg\_data["Delay"]

# Split the data into training and testing sets

X\_train, X\_test, y\_train, y\_test = train\_test\_split(reg\_variables,

reg\_target, test\_size=0.4)

# Scale the data for use in linear regression

scaler = MinMaxScaler()

scaler.fit(X\_train)

X\_train\_scaled = scaler.transform(X\_train)

X\_test\_scaled = scaler.transform(X\_test)

# Linear Regression

from sklearn.linear\_model import LinearRegression

from sklearn.metrics import r2\_score

# Fit a linear regression model to the training data

lin\_reg = LinearRegression()

lin\_reg.fit(X\_train\_scaled, y\_train)

# Predict the target variable using the trained model

train\_pred = lin\_reg.predict(X\_train\_scaled)

# Calculate the r-squared score on the training data

r2\_train = r2\_score(y\_train, train\_pred)

print("Training r-squared score:", round(r2\_train, 4))

Model r2 0.139

The model performs badly on the data, which means that the coefficients of the model would not be reliable as well

for column, coef in zip(variables.columns, lr.coef\_):

print(f"{column}: {coef:.4f}")

Amount: -10.6717

Paid\_Yes: 0.0000

Category\_AT: -65.1169

Category\_XT: 65.1169

Terms\_AB27: -0.4704

Terms\_AD23: 2.7060

Terms\_AE14: 1.3298

Terms\_BE01: -6.1040

Terms\_CB37: 3.1676

Terms\_CB91: 7.4293

Terms\_CD89: 2.0466

Terms\_DA17: 0.6672

Terms\_DD29: -11.7280

Terms\_DE16: 2.1393

Terms\_EC05: -2.4136

Terms\_UC01: 1.2302

Region\_FVS: -1.3297

Region\_LOC: 1.3297

Type\_L001: -3.8960

Type\_L002: 2.4302

Type\_L003: 0.6682

Type\_L004: 5.1181

Type\_L005: -2.9909

Type\_O001: -1.3297

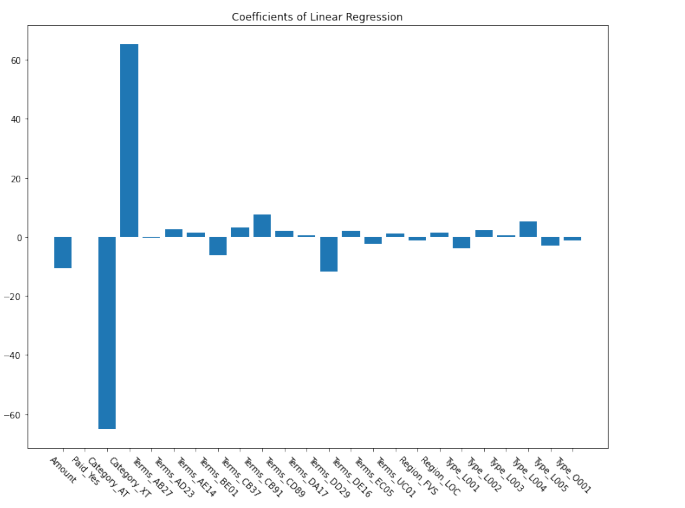
plt.figure(figsize = (12, 9))

plt.title("Coefficients of Linear Regression")

plt.bar(x = variables.columns, height = lr.coef\_)

plt.xticks(rotation = 315)

plt.show()



**Question 6**

We can observe that category has the greatest impact on payment delay for the trained model. We may deduce that category AT is likely to be early since it has a significant negative coefficient, indicating that policies in category AT would have a low delay value, thus being early.

Due to its high positive coefficient, category XT is typically late.