

**ANL252**

**Python for Data Analytics**

**ECA** **July 2022**

**Presentation**

K1711101

Lim Zheng Hui

04/09/22

Question 1

# importing required libraries

import pandas as pd

import numpy as np

from matplotlib import pyplot as plt

# read data from csv file into pandas dataframe

data\_f = pd.read\_csv( "ECA\_data.csv")

data\_f.head()

# check for nan values in data

data\_f.isna()

# categorical and numeric variables in this data

data\_f.columns

Categorical variables

* Rating
* Marital Status
* Gender
* Education

Numeric variables

* ID
* Limit
* Balance
* Income
* Age
* S1
* S2
* S3
* S4
* S5
* B1
* B2
* B3
* B4
* B5
* R1
* R2
* R3
* R4
* R5

Question 2

1. Removing nan values by removing any previously recorded nan values from the dataset to perform data cleaning.

# data pre-processing: removing nan values

data\_f = data\_f.dropna()

1. Removing special characters by eliminating the dollar sign from the data variables R3 and conversion of the column into integers.

# data pre-procssing: removing special characters

data\_f["R3"] = (data\_f["R3"].map(lambda x: x.strip("$").replace(",", ""))).astype('int')

1. Removing ID column that contains the name and ID as it just contains the serial number which is not informative.

# data pre-processing: removing column with name ID

data\_f = data\_f.drop(["ID"], axis=1)

1. Normalization so that the data will be normalized and standardized such that it will fall within a predetermined range.

# data pre-processing: normalization

data\_f = (data\_f - data\_f.min())/ (data\_f.max() - data\_f.min())

Question 3

# importing required library for graphs

import seaborn as sns

sns.countplot(x=data\_f['EDUCATION']).set(title="Education Counts for each level")

**Plot 1:**

Chart, bar chart

Description automatically generated

**Plot Insights:**

According to the graph above, there are generally four levels of education included in the data set. The percentage of individuals with education levels above Tertiary is significantly higher than that of people with education levels of high school and others.

sns.scatterplot(data=data\_f, x="BALANCE", y="B1").set(title = 'Distribution of Balance wrt B1')

**Plot 2**:

Chart, scatter chart

Description automatically generated

**Plot Insights:**

The relationship between Balance and B1 may be shown to be linear from the graph that was just presented.

sns.countplot(x=data\_f['MARITAL']).set(title="MARITAL Counts")

**Plot 3:**

**Chart, bar chart

Description automatically generated with medium confidence**

**Plot insights:**

The graph shows that the number of customers who are either single or married are much higher than others and majority of the customers are married.

sns.lineplot(x=data\_f['LIMIT'], y = data\_f['B1']).set(title="Limit vs B1" , xlabel = 'Limit' , ylabel = 'B1')

**Plot 4:**

**Chart

Description automatically generated**

**Plot insights**:

The graph shows that customers with higher limit value tends to exhibit higher B1 value.

**Plot 5:**

# showing approx normal distribution between Income and target variable B1

sns.lineplot(x=data\_f['INCOME'], y = data\_f['B1']).set(title="INCOME vs B1" , xlabel = 'INCOME' , ylabel = 'B1')

**Chart, histogram

Description automatically generated**

**Plot insights:**

The graph shows that majority of the customers have income that falls under 600000.

**Question 4**

The following data split ratio was utilized for regression modeling:

70% Training Set

30% Test Set

Additionally, provided data is fitted for regression using Sklearn's built-in regressor model, and then test data is utilized to make predictions.

# specifying target and features values

target = data\_f["B1"]

features = data\_f.drop(["B1"], axis=1)

# splitting data into Training and Test set

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

X\_train\_d, X\_test\_d, y\_train\_d, y\_test\_d = train\_test\_split( features, target, test\_size=0.30)

# importing libraries required for linear regression modeling

from sklearn.linear\_model import LinearRegression

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

# linear regression model instance

reg\_model = LinearRegression()

reg\_model.fit(X\_train\_d, y\_train\_d)

# predicting result

prediction\_t = reg\_model.predict(X\_test\_d)

# score of the model testing

print('R2\_score ', r2\_score(y\_test\_d, prediction\_t))

print('mean\_square\_error : ', mean\_squared\_error(y\_test\_d, prediction\_t) )

**Question 5**

# building equation of the model by getting coeficients value

equation\_val = []

for i in range(len(reg\_model.coef\_ )):

equation\_val.append(str( np.around(reg\_model.coef\_ [i], decimals = 4)) + " " +str(features.columns[i]))

print("equation = ", " + ".join(equation\_val))

Equation = 0.0076 LIMIT + 0.4332 BALANCE + -0.007 INCOME + -0.0002 RATING + -0.0003 GENDER + -0.0001 EDUCATION + 0.0006 MARITAL + -0.0007 AGE + -0.0095 S1 + 0.0171 S2 + -0.0032 S3 + -0.0019 S4 + -0.0013 S5 + 0.6693 B2 + -0.0098 B3 + -0.0664 B4 + 0.0602 B5 + -0.3917 R1 + 0.112 R2 + 0.0351 R3 + -0.0432 R4 + -0.0059 R5

**Insights:**

* Since the mean square error and R2 scores are quite close to the standard deviation of our data, the linear regression model has worked effectively for our data.
* The final equation displays the percentage that each column has contributed toward our objective, which is the balance due for the current month.
* The highest Coefficient of B2 illustrates how significantly previous outstanding balances affect an unpaid amount.