

**ANL252**

**Python for Data Analytics**

**July 2022 Presentation**

**End-of-Course Assessment**

**Submitted by:**

|  |  |
| --- | --- |
| **Name** | **PI No.** |
| **Tan Wee Min Andre** | **E2070932** |

**Tutorial Group: TG09**

**Instructor’s Name: Dr Munish Kumar**

**Submission Due Date: 5 September 2022**

“We declare that this assignment is our own work, unless otherwise acknowledge or credited by appropriate referencing. We have read and abide by the SUSS Honour Code and we are aware of the penalties associated with plagiarism and collusion listed in the SUSS Student Handbook.”

Question 1:

The categorical variables in the dataset are, namely, GENDER, EDUCATION, MARITAL, S1, S2, S3, S4, S5 and RATING.

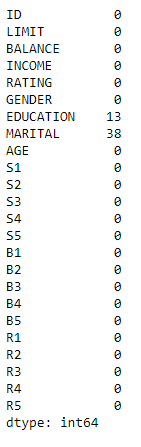
The numeric variables in the dataset are, namely, ID, LIMIT, BALANCE, INCOME, AGE, B1, B2, B3, B4, B5, R1, R2 R3, R4 and R5.

Question 2:

**Data Pre-processing stage**

There are 13 null values under EDUCATION column and 38 nulls values under MARITAL columns

df\_profile.isnull().sum()



missing\_pct = df\_profile.isnull().sum()/len(df\_profile)

df\_missing = pd.DataFrame({"% missing":missing\_pct})

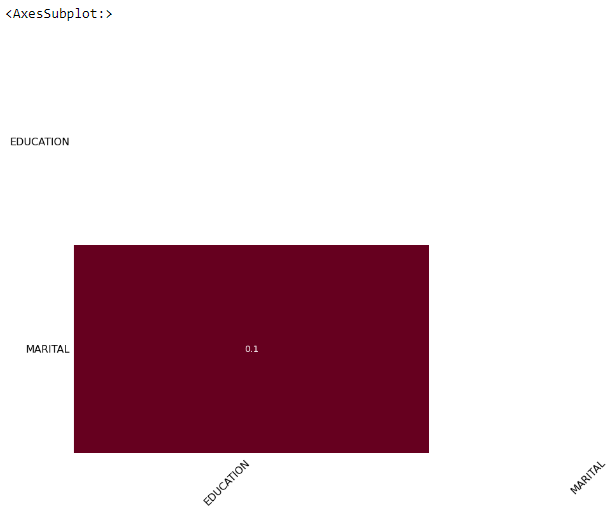
df\_missing



Heatmap shows the correlation of missingness between every 2 columns

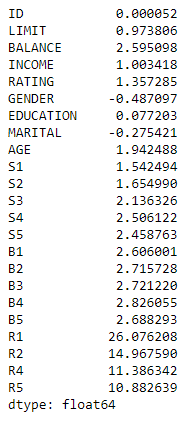
A value near 0 means there is no dependence between the occurrence of missing values of two variables. Therefore, there're no correlations.

msno.heatmap(df\_profile)



Checking the skewness of the data in each column, helps to identify whether each column is skewed or otherwise. For columns with skewed data, mean calculation is not advisable as mean values computed will tend to be distorted as it is either higher or lower than the median or mode. Hence, the preferred central tendency calculations are median and mode. For columns with fairly unskewed data, mean, median and mode values are fairly aligned and all are acceptable.

df\_profile.skew()



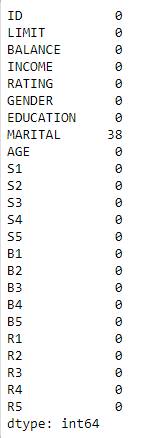
Data Pre-processing stage - Task 1:

Since proportion of missing values for education is miniscule and the distribution is highly symmetrical, as evident from the skewness value very close to 0, the missing education values will be replaced with the most commonly occuring education level.

df\_profile["EDUCATION"].fillna(df\_profile["EDUCATION"].mode()[0], inplace=True)

To check that nulls within EDUCATION have been resolved.

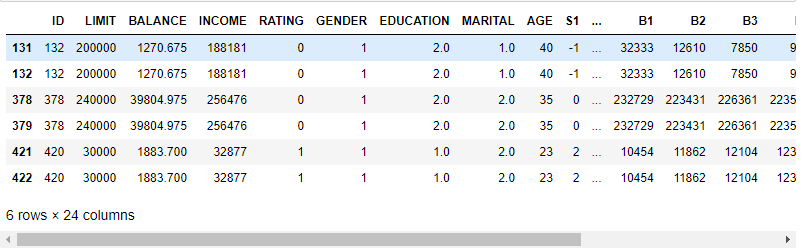
df\_profile.isnull().sum()



Data Pre-processing stage - Task 2:

Removing THREE duplicate IDs, namely, 132, 378 and 420

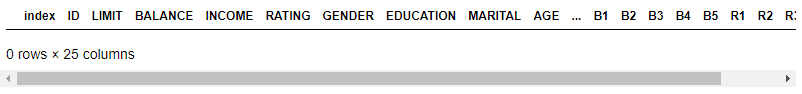
df\_profile[df\_profile.duplicated(subset="ID", keep=False)==True]



df\_profile.drop\_duplicates(subset="ID", inplace=True)

df\_profile.reset\_index(inplace=True)

df\_profile[df\_profile.duplicated(subset="ID", keep=False)==True]



Interim Transformation task:

This is to ascertain the spread and range of INCOME values



pd.unique(df\_profile["INCOME"].sort\_values())

print(pd.unique(df\_profile["INCOME"].sort\_values()))

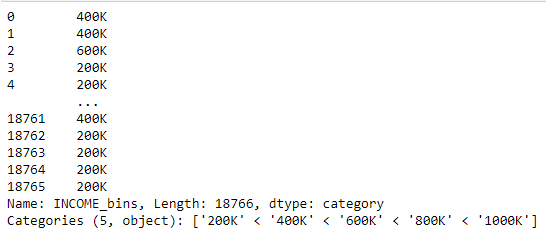
Discretise the INCOME values into different income range to facilitate data-preoprocessing and subsequent analysis. Since the minimum value is 10,000 and maximum value is 908,846, the most appropriate bin size is 200,000, resulting in 5 income-range categories.

bin\_range = [1, 200000, 400000, 600000, 800000, 1000000]

bin\_label = ["200K", "400K", "600K", "800K", "1000K"]

df\_profile["INCOME\_bins"] = pd.cut(x=df\_profile["INCOME"], bins=bin\_range, labels=bin\_label, right=False)

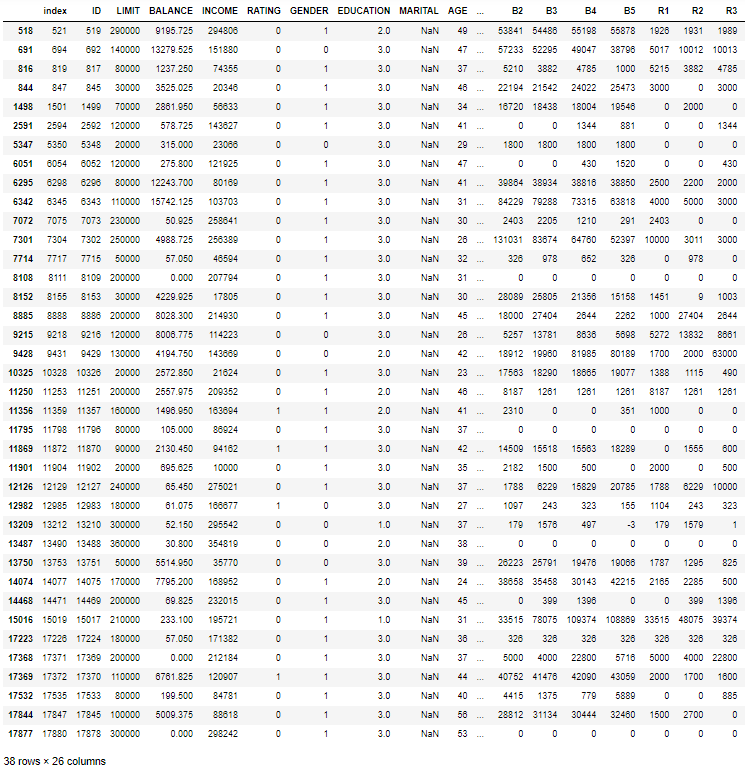
df\_profile["INCOME\_bins"]



Data Pre-processing stage - Task 3:

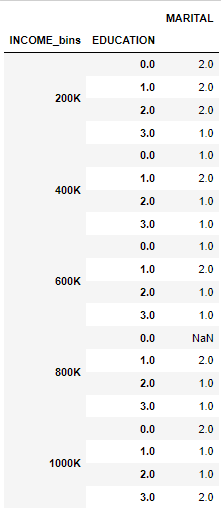
Records with missing marital status largely fall under Income group of 200K and 400K.

df\_profile[df\_profile["MARITAL"].isnull()]



Common profile of customer according to Income range and Education level. Assume both Male and Female are just as likely to get married or stay single. As MARITAL values are fairly symmetrically distributed, use of mode is reasonable.

df\_profile.groupby(by=["INCOME\_bins","EDUCATION"]).agg({"MARITAL":pd.Series.mode})



df\_profile["MARITAL"] = np.where((df\_profile["MARITAL"].isnull()) & (df\_profile["INCOME\_bins"]=="400K") & (df\_profile.EDUCATION==1), 2, df\_profile["MARITAL"])

df\_profile["MARITAL"] = np.where((df\_profile["MARITAL"].isnull()) & (df\_profile["INCOME\_bins"]=="400K") & (df\_profile.EDUCATION==2), 1, df\_profile["MARITAL"])

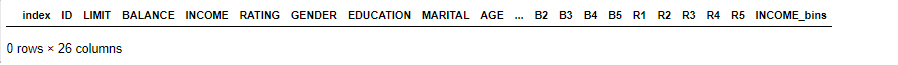
df\_profile["MARITAL"] = np.where((df\_profile["MARITAL"].isnull()) & (df\_profile["INCOME\_bins"]=="400K") & (df\_profile.EDUCATION==3), 1, df\_profile["MARITAL"])

df\_profile["MARITAL"] = np.where((df\_profile["MARITAL"].isnull()) & (df\_profile["INCOME\_bins"]=="200K") & (df\_profile.EDUCATION==1), 2, df\_profile["MARITAL"])

df\_profile["MARITAL"] = np.where((df\_profile["MARITAL"].isnull()) & (df\_profile["INCOME\_bins"]=="200K") & (df\_profile.EDUCATION==2), 2, df\_profile["MARITAL"])

df\_profile["MARITAL"] = np.where((df\_profile["MARITAL"].isnull()) & (df\_profile["INCOME\_bins"]=="200K") & (df\_profile.EDUCATION==3), 1, df\_profile["MARITAL"])

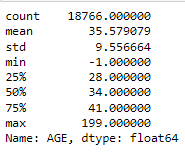
df\_profile[df\_profile["MARITAL"].isnull()]



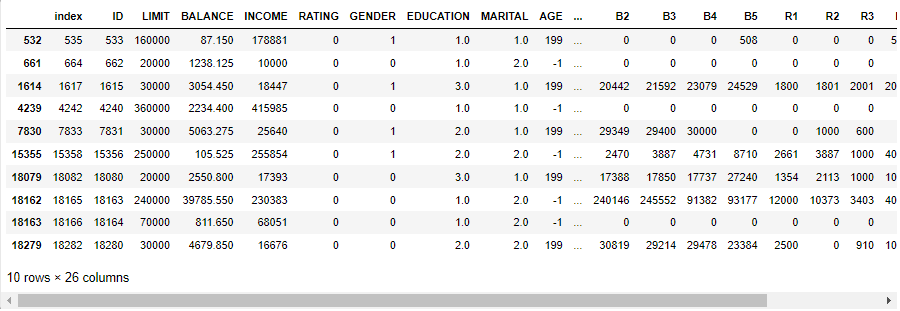
Data Pre-processing stage - Task 4:

AGE values cannot be -1 and 199. As AGE values are positively skewed and there are only FIVE error values, median values will be used.

df\_profile["AGE"].describe()



df\_profile[(df\_profile["AGE"] == -1) | (df\_profile["AGE"] == 199)]



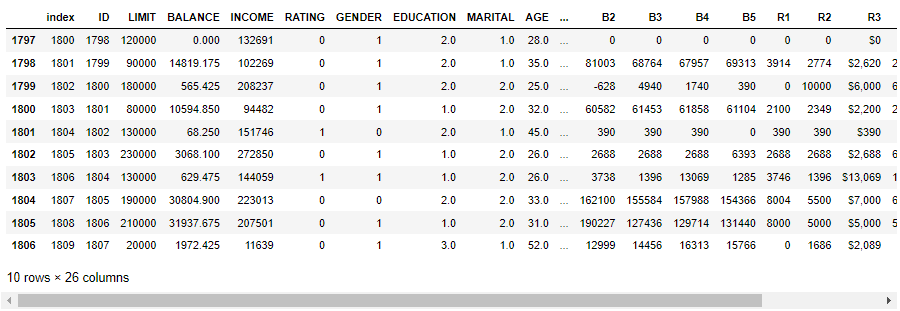
df\_profile["AGE"].replace(to\_replace={-1: df\_profile["AGE"].median(), 199: df\_profile["AGE"].median()}, value=None, inplace=True)

df\_profile[(df\_profile["AGE"] == -1) | (df\_profile["AGE"] == 199)]



Find values under column R3 with "$" string value:

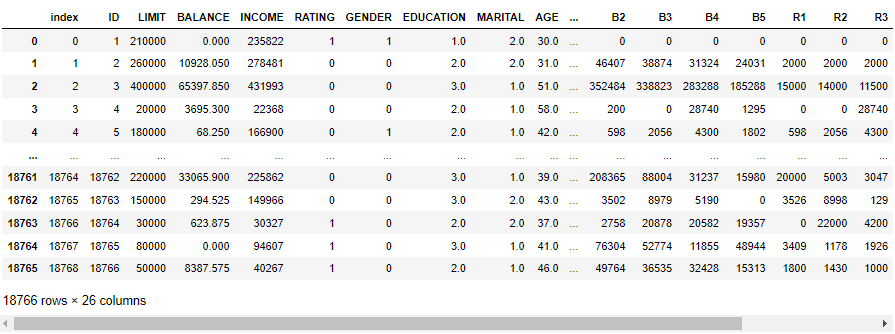
df\_profile[df\_profile["R3"].str.find("$")!=-1]



error\_dict = {"$0": 0, "$2,620": 2620, "$6,000": 6000, "$2,200": 2200, "$390": 390, "$2,688": 2688, "$13,069": 13069, "$7,000": 7000, "$5,000": 5000, "$2,089": 2089}

df\_profile["R3"].replace(to\_replace=error\_dict, value=None, inplace=True)

df\_profile.astype(dtype={"R3": "int"}, copy=False)



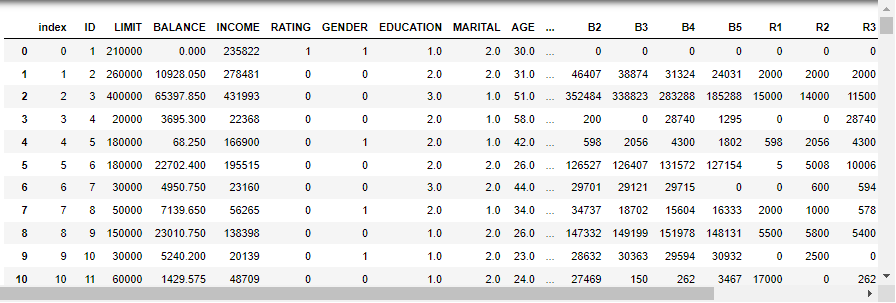
Question 3:

Insight 1:

Generally, people do not stretch their credit limit too far. Majority of the customers use only 25% of their credit limit, with only a minority spending 50% or more.

In addition, there are more female credit card users than males.

df\_profile.head(300)



temp = []

util\_grp=[]

temp = (df\_profile["BALANCE"]/df\_profile["LIMIT"])\*100

for i in range(len(temp)):

if(temp[i] <= 25):

util\_grp.append("25%")

elif(temp[i] <= 50):

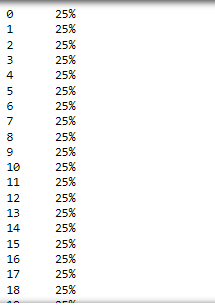
util\_grp.append("50%")

else:

util\_grp.append("75%")

df\_profile["Credit\_Util"]=util\_grp

df\_profile["Credit\_Util"].head(140)

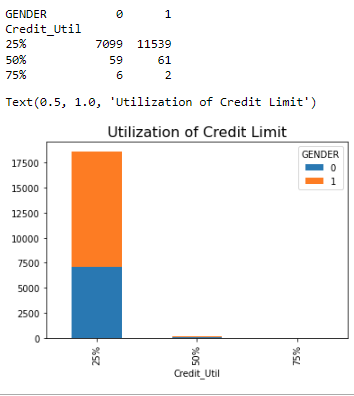


df\_result = pd.pivot\_table(df\_profile, values="ID", index="Credit\_Util", columns="GENDER", aggfunc="count")

print(df\_result)

df\_result.plot.bar(stacked=True)

plt.title("Utilization of Credit Limit", fontsize=16)



Insight 2:

Majority of customers has only credit limit that is within 1 times their annual income. However, a significant proportion of customers still holds credit limit that is twice their annual income.

temp = []

temp = (df\_profile["LIMIT"]/df\_profile["INCOME"])

cmultiple\_grp = []

for i in range(len(temp)):

if(temp[i] <= 1):

cmultiple\_grp.append("1x")

elif(temp[i] <= 2):

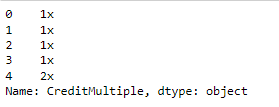
cmultiple\_grp.append("2x")

else:

cmultiple\_grp.append(">3x")

df\_profile["CreditMultiple"]=cmultiple\_grp

df\_profile["CreditMultiple"].head()

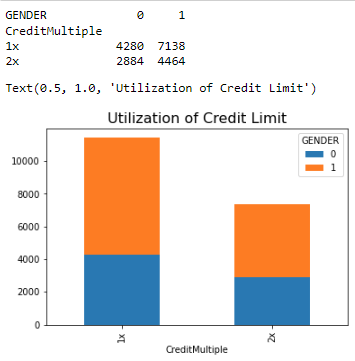


df\_result = pd.pivot\_table(df\_profile, values="ID", index="CreditMultiple", columns="GENDER", aggfunc="count")

print(df\_result)

df\_result.plot.bar(stacked=True)

plt.title("Utilization of Credit Limit", fontsize=16)



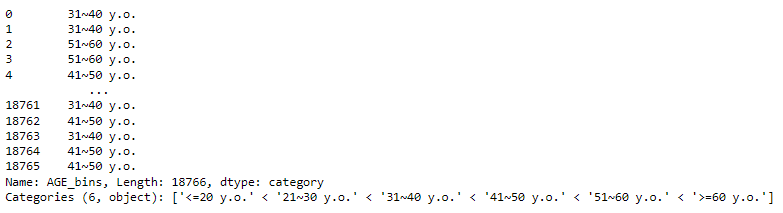
Interim Transformation task

bin\_range = [0, 20, 30, 40, 50, 60, np.inf]

bin\_label = ["<=20 y.o.", "21~30 y.o.", "31~40 y.o.", "41~50 y.o.", "51~60 y.o.", ">=60 y.o."]

df\_profile["AGE\_bins"] = pd.cut(x=df\_profile["AGE"], bins=bin\_range, labels=bin\_label, right=False)

df\_profile["AGE\_bins"]



Insight 3:

Profile of a person with prompt payment behaviour, i.e. paying on time for 5 consecutive months. Knowing this profile, allows credit facility to market the right credit product to customers. Based on the charts below, it would seem that a prompt paying customer exhibit the following characteristics:

-have good credit rating

-around the age of 31 to 40 years old

-between 31 to 40 years of age

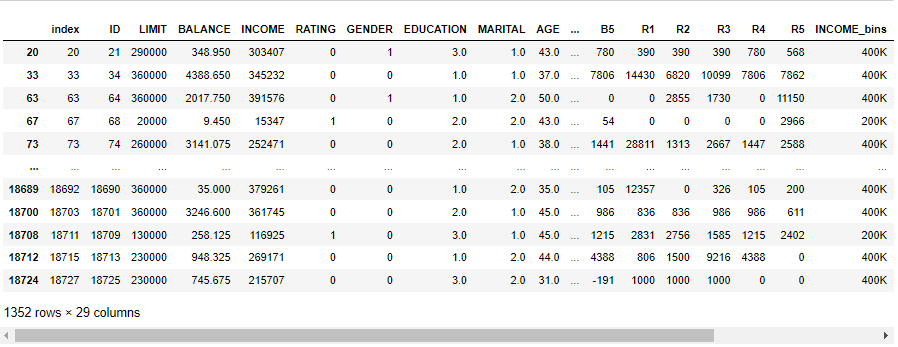
-received up till tertiary or post-graduate education

-either single or married

-normally earning annual salary range below 400K

df\_result = df\_profile[(df\_profile["S1"]==-1) & (df\_profile["S2"]==-1) & (df\_profile["S3"]==-1) & (df\_profile["S4"]==-1) & (df\_profile["S5"]==-1)]

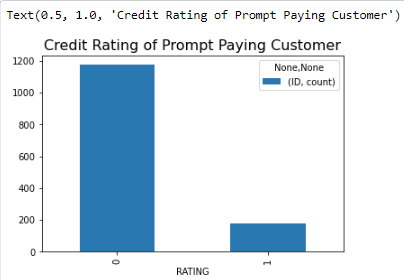
df\_result



result\_rating = df\_result.groupby(by="RATING").agg({"ID":["count"]})

result\_rating.plot(kind="bar")

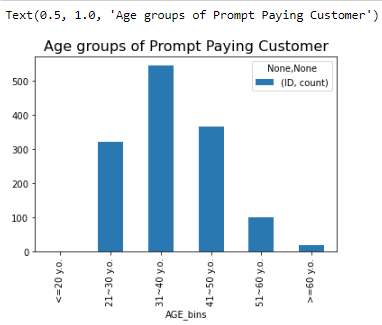
plt.title("Credit Rating of Prompt Paying Customer", fontsize=16)



result\_age = df\_result.groupby(by="AGE\_bins").agg({"ID":["count"]})

result\_age.plot(kind="bar")

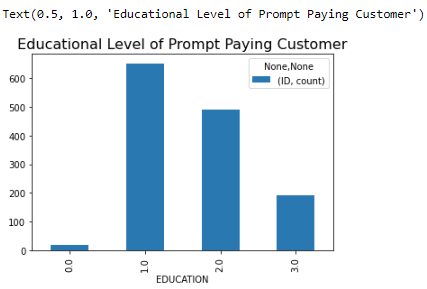
plt.title("Age groups of Prompt Paying Customer", fontsize=16)



result\_education = df\_result.groupby(by="EDUCATION").agg({"ID":["count"]})

result\_education.plot(kind="bar")

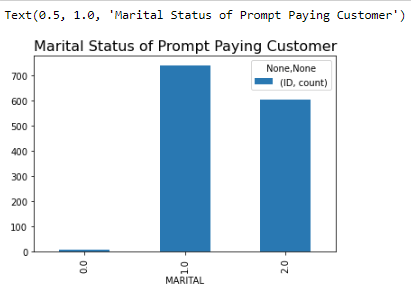
plt.title("Educational Level of Prompt Paying Customer", fontsize=16)



result\_marital = df\_result.groupby(by="MARITAL").agg({"ID":["count"]})

result\_marital.plot(kind="bar")

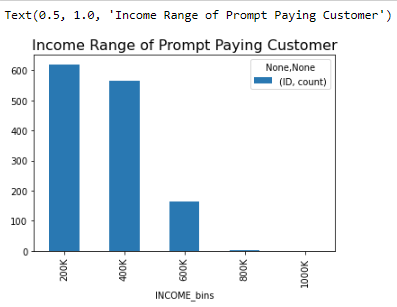
plt.title("Marital Status of Prompt Paying Customer", fontsize=16)



result\_income = df\_result.groupby(by="INCOME\_bins").agg({"ID":["count"]})

result\_income.plot(kind="bar")

plt.title("Income Range of Prompt Paying Customer", fontsize=16)



Insight 4:

Net owing calculates how much outstanding amount is still owing to the credit facility by deducting total repayments(R1 to R5) from total billings(B1 to B5). The dataset is then filtered to present only risk groups of concern where the amount owed exceeds credit limit and their corresponding RATING category.

If further examination is performed, would reveal that many such customer are only able to make minimum repayment or worse, i.e. Sn=0 and above, for many consecutive periods.

From the chart, it seems that many of the "good(0)" rating customers actually fall under this risk group and their credit RATING would reasonably need a review.

df\_profile["R3\_cast"] = df\_profile["R3"].astype("int64", copy=False)

df\_profile.drop("R3", axis=1, inplace=True)

df\_profile.rename(columns={"R3\_cast": "R3"}, inplace=True)

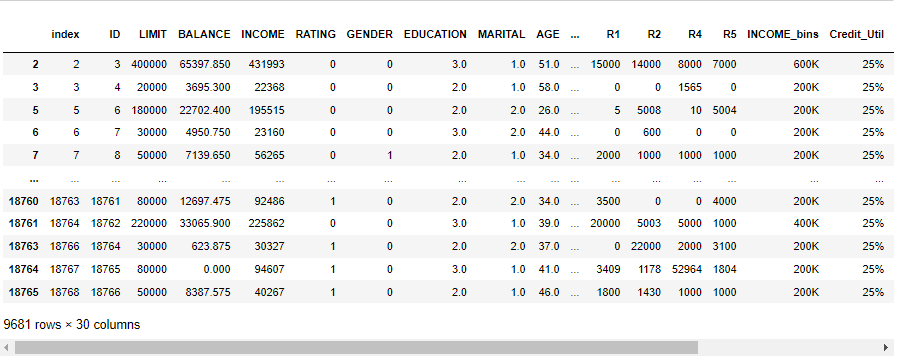
Bcol\_lst = ["B1", "B2", "B3", "B4", "B5"]

Rcol\_lst = ["R1", "R2", "R3", "R4", "R5"]

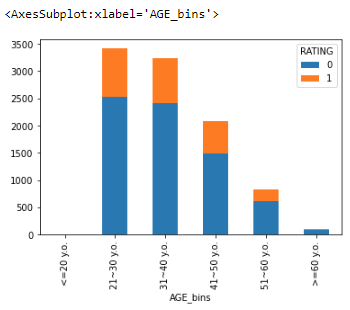
df\_profile["Net Owing"] = df\_profile[Bcol\_lst].sum(axis=1) - df\_profile[Rcol\_lst].sum(axis=1)

df\_riskgrp = df\_profile[df\_profile["Net Owing"]>df\_profile["LIMIT"]]

df\_riskgrp



df\_riskgrp.pivot\_table(values="ID", index="AGE\_bins", columns="RATING", aggfunc="count").plot.bar(stacked=True)



Insight 5:

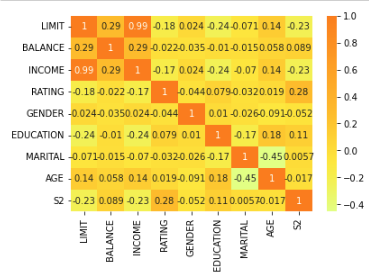
To determine if there are any existing relationship between the variables to allow credit facility to carry out better customer segmentation. From the outcome of the heatmap, it can be seen there're very low relationship amongst them. The strongest relationship is between INCOME and LIMIT which is rightful because the amount of credit limit given to customers is based on the amount income they can earn.

small\_data = ['LIMIT', 'BALANCE', 'INCOME', 'RATING', 'GENDER', 'EDUCATION', 'MARITAL', 'AGE', 'S2']

small\_data = df\_profile[small\_data].copy()

sns.heatmap(small\_data.corr(), cmap = 'Wistia', annot= True)

plt.show(sns)

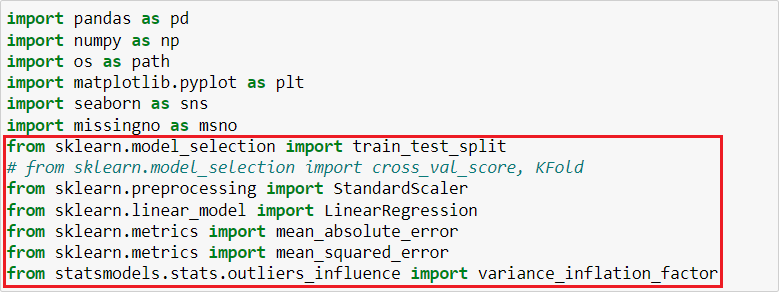


Question 4:

Building Multiple Linear Regression(MLR) model to predict dependent variable, B1

There are two types of supervised machine learning algorithms, namely regression and classification. Regression predicts continuous value outputs while classification predicts discrete outputs. The goal here is to ascertain what variables can help predict “B1”.

Firstly, required libraries need to be imported as shown in following diagram:

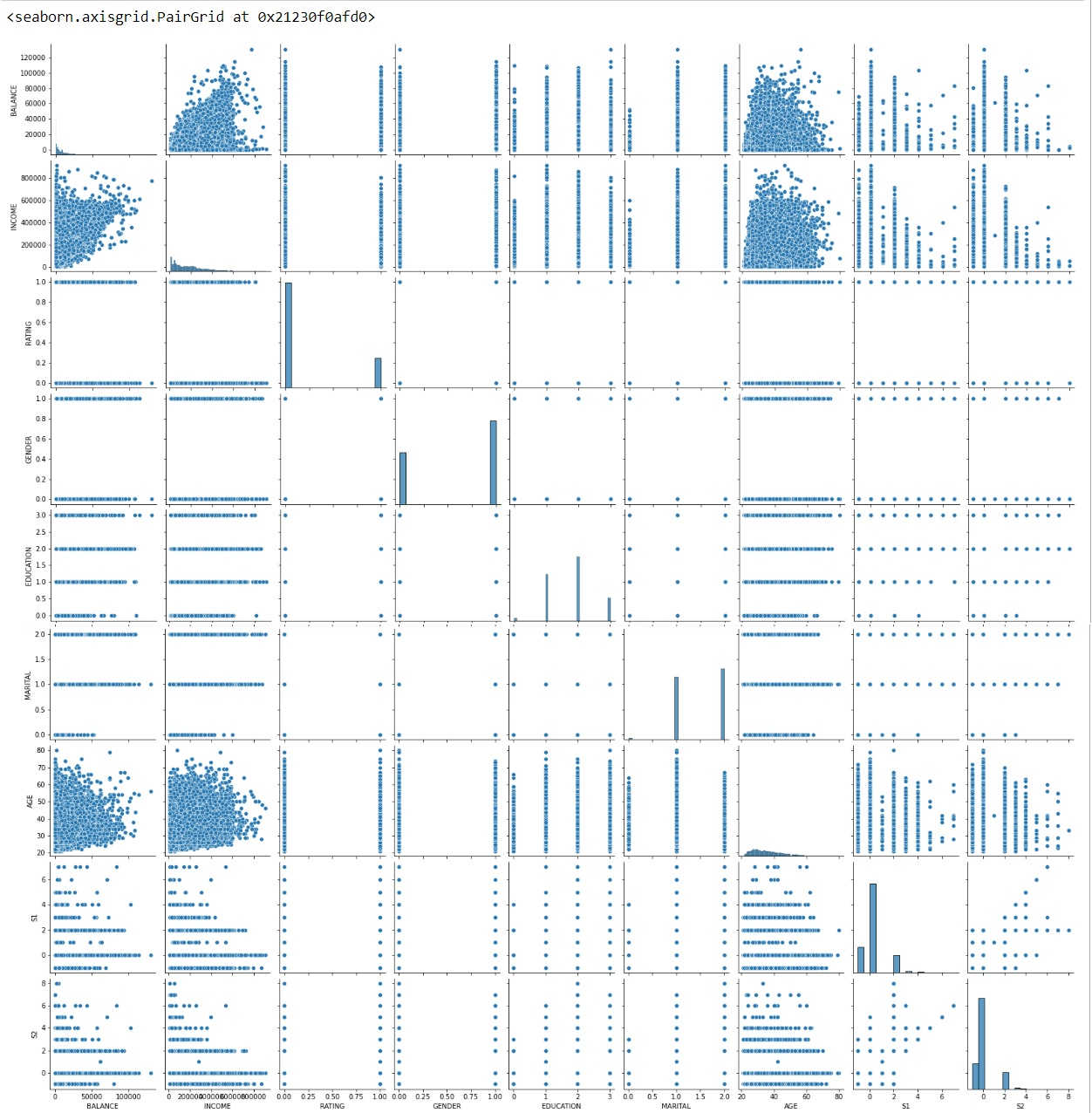


Then a pair plot is generated to try and visualize any relationship between dependent and independent variables. As shown in following diagram, only “S1” and “S2” relationship pair seem to exhibit some form of relationship. The code as follows:

pairplot\_vars = ["BALANCE", "INCOME", "RATING", "GENDER", "EDUCATION", "MARITAL", "AGE", "S1", "S2"]

df\_pairplot = df\_profile[pairplot\_vars].copy()

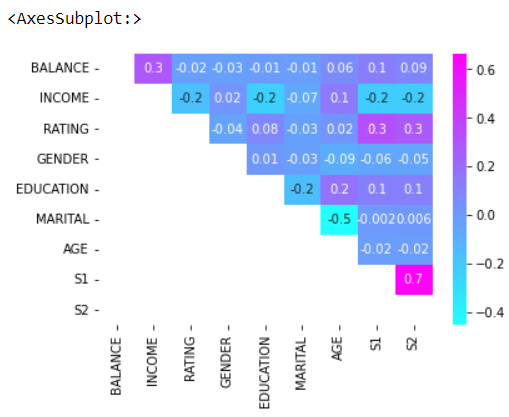
sns.pairplot(df\_pairplot)



This is further verified by the correlation pair plots as shown in following diagram where S1 and S2 are is showing positive correlation of 0.7. Hence, S2 will be selected as one of the independent variables in the MLR model. The code as follows:

mask = np.tril(df\_pairplot.corr())

sns.heatmap(df\_pairplot.corr(), fmt=".1g", annot=True, cmap= "cool", mask=mask)



The independent variables selected for the MLR model are, namely, BALANCE, INCOME, RATING, GENDER, EDUCATION, MARITAL, AGE, S2. The code as follows:

# select which numerical data as predictor/explanatory variables

# TODO exclude/include to arrive at the lowest RMSE and R²

predictor\_vars = ["BALANCE", "INCOME", "RATING", "GENDER", "EDUCATION", "MARITAL", "AGE", "S2"]

# predictor\_vars = ["LIMIT", "BALANCE", "INCOME", "AGE"]

x\_final = df\_profile[predictor\_vars].copy()

# define B1 as dependent variable

y\_final = df\_profile[["B1"]].copy()

Prior to executing the MLR model, an issue of concern is the dependency of independent variables with each other. This is one of the data pre-processing steps to be performed. Multicollinearity occurs when independent variables in the regression model becomes highly correlated to each other, making the model outcome difficult to interpret. It also causes an overfitting problem. This is a common condition that people test before selecting the variables for the regression model. Variance Inflation Factor (VIF) test is one way to determine such interdependency. Although correlation matrix and scatterplots may be used to find multicollinearity, their findings only show bivariate relationship between the independent variables. VIF is preferred as it can show the correlation of a variable with a group of other variables. The code to generate the VIF results are as follows:

# create VIF dataframe

vif\_data = pd.DataFrame()

vif\_data["Predictor"] = x\_final.columns

# calculating VIF for each feature

vif\_data["VIF"] = [variance\_inflation\_factor(x\_final.values, i)

for i in range(len(x\_final.columns))]

print(vif\_data)

The test was ran twice once before removing “AGE” independent variable and another time after removing it. Test outcome will be explained within Question 5.

Another data pre-processing step is the splitting of dataset. The dataset was divided into a training set and a test set in the proportion of 70% and 30% respectively. The random\_state parameter is used for initializing the internal random number generator, which will decide the splitting of data into train and test indices in your case. The random state = 0 was set so that output can be compared over multiple runs of the code using the same parameter. The code as follows:

# PRE\_PROCESSING

# split sample data into 30/70;

# 30% for testing the model (making prediction), and 70% for training the model

x\_train, x\_test, y\_train, y\_test = train\_test\_split(x\_final, y\_final, test\_size=0.30, random\_state=0)

Another step in pre-processing is called feature scaling. It is the process of changing the scale of certain features, independent variables, to a common one through normalization and standardization. Normalization is the process of converting data into a range between 0 and 1 and is more common for regression analysis. Standardization is the process of converting data so they have a mean of 0 and standard deviation of 1. This is more common for classification analysis. Feature scaling is used because there is a mixture of independent variables where the values are wide ranging like INCOME, BALANCE, etc and other narrow range ones like GENDER, MARITAL, etc. This helps in interpreting the outcome of the MLR model. In the following code, Standard scaling was used.

# PRE\_PROCESSING

# convert both train and test data to standard normal distribution

scaler = StandardScaler()

x\_train = scaler.fit\_transform(x\_train.astype(np.float64))

x\_test = scaler.transform(x\_test.astype(np.float64))

The training data is then fitted to the model. This is the training part of the modelling process. The α which is the intercept and βn coefficients can then be shown. The code and output of its execution as follows:

# RUN LINEAR REGRESSION TO TRAIN ALGORITHM

lr = LinearRegression().fit(x\_train, y\_train)

# INTERPRET RESULTS

# display intercept / constant term

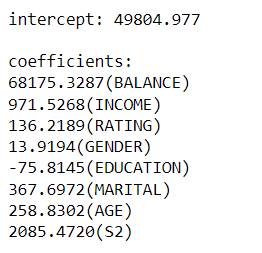
print('intercept: %.3f' % (lr.intercept\_))

# display coefficients of predictors

print('\ncoefficients:')

for i,j in enumerate(predictor\_vars, 0):

print('{0:.4f}({1})'.format(lr.coef\_[0][i], j))



A test dataset is independent of the training dataset and helps to show a better view of MLR ability to model the outcome. The code to show test data’s prediction result as follows:

#MAKING PREDICTIONS TO SEE HOW ACCURATE ALGORITHM IS IN PREDICTING "B1" VARIABLE

# make prediction using test data

# y\_train\_pred = lr.predict(x\_train)

y\_test\_pred = lr.predict(x\_test)

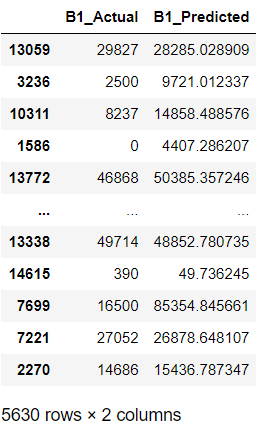
The predicted results can then be compared with the actual test data. The closer the B1\_Predicted values are to B1\_Actual, the more accurate the MLR model. Based on the comparison, the MLR model is moderately accurate in its prediction. The code and results as follows:

df\_compare = y\_test.copy()

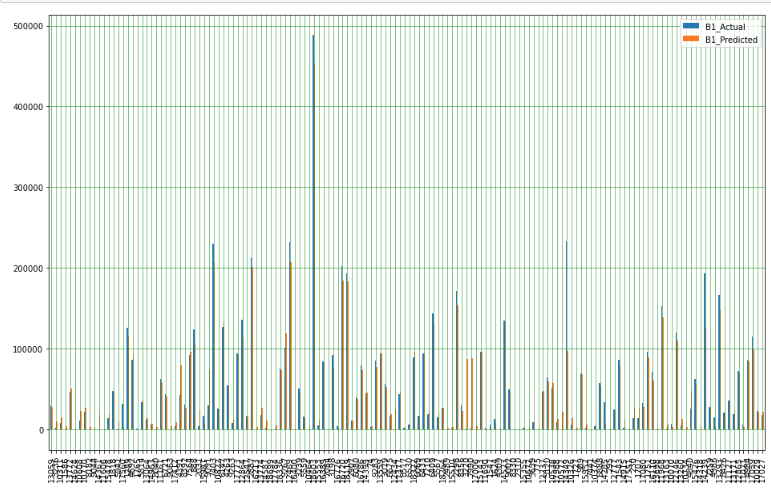
df\_compare.rename(columns={"B1":"B1\_Actual"}, inplace=True)

df\_compare["B1\_Predicted"] = y\_test\_pred

df\_compare



The comparison can also be visualized within a bar chart. However, the number of record are huge and not possible to show the entire record. Hence, only a small subset of the chart output and is shown as follows:



The results from the chart comparison yields similar results as the value comparison between B1\_Actual and B1\_Predicted table above.

Lastly, is evaluating MLR model’s performance. This is achieved by generating several statistical measurements from the code and output as follows:

# R-Squared is how well regression model explains the observed data;

# 0.90 reveals that 90% of variability observed in target variable is explained by the model

print('r-squared: %.3f' % (lr.score(x\_test, y\_test)))

print('Mean Absolute Error: {:.3f}'.format(mean\_absolute\_error(y\_test, y\_test\_pred)))

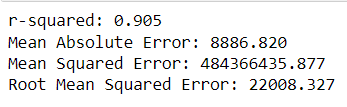
print('Mean Squared Error: {:.3f}'.format(mean\_squared\_error(y\_test, y\_test\_pred)))

# RMSE refers to root mean squared error;

# This tells us that the average deviation between the predicted and actual B1 value is 22008.218

# larger values mean larger error

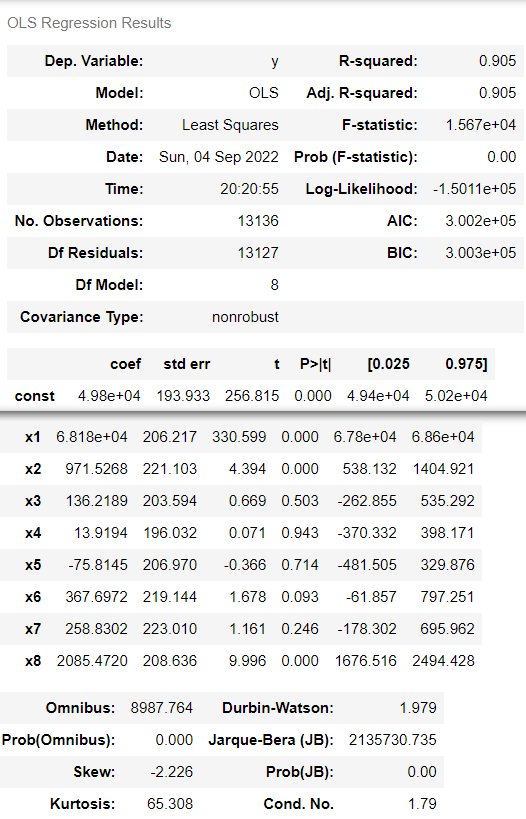
print('Root Mean Squared Error: {:.3f}'.format(np.sqrt(mean\_squared\_error(y\_test, y\_test\_pred))))



X2 = sm.add\_constant(x\_train)

model\_stats = sm.OLS(y\_train.values.reshape(-1,1), X2).fit()

model\_stats.summary()



Question 5:

This is a multiple linear regression model with more than one predictor variable and is modelled by equation: Yₑ = α + β₁X₁ + β₂X₂ + … + βₚXₚ, where:

* p is the number of independent variables
* Yₑ is the dependent variable
* Xₚ is the independent variable

βₚ is the average effect on Yₑ of one unit increase in Xₚ, assuming all other independent variables remain unchanged

Based on the dataset scenario, the model will be written as: B1 = α + (β1 x BALANCE) + (β2 x INCOME) + (β3 x RATING) + (β4 x GENDER) + (β5 x EDUCATION) + (β6 x MARITAL) + (β7 x AGE) + (β8 x S2)

After fitting in the results from earlier on, the MLR equation is as follows:

B1 = 49804.977 + (68175.3287 x BALANCE) + (971.5268 x INCOME) + (136.2189 x RATING) + (13.9194 x GENDER) + (-75.8145 x EDUCATION) + (367.6972 x MARITAL) + (258.8302 x AGE) + (2085.4720 x S2)

The interpretation is as follows:

For every dollar of BALANCE increase/decrease, the billing amount for B1 will increase/decrease by 68175.3287.

For every dollar of INCOME increase/decrease, the billing amount for B1 will increase/decrease by 971.5268.

For RATING increase/decrease, the billing amount for B1 will increase/decrease by 136.2189.

For GENDER increase/decrease, the billing amount for B1 will increase/decrease by 13.9194.

For EDUCATION increase/decrease, the billing amount for B1 will decrease/increase by 136.2189.

For MARITAL increase/decrease, the billing amount for B1 will increase/decrease by 367.6972.

For every year of AGE increase/decrease, the billing amount for B1 will increase/decrease by 258.8302.

For every dollar of S2 increase/decrease, the billing amount for B1 will increase/decrease by 2085.4720.

The intercept of value 49804.977 means the average billing amount for B1 is that value when all other independent factors are equal to zero. However, it would not be applicable to interpret as such for AGE since it could equal zero in reality.

Multicollinearity of the dataset was tested using Variance Inflation Factor(VIF) and the results are shown in the following diagram:

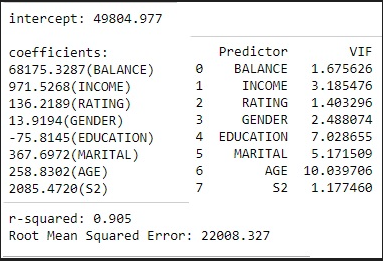


Figure. Before removal of AGE independent factor

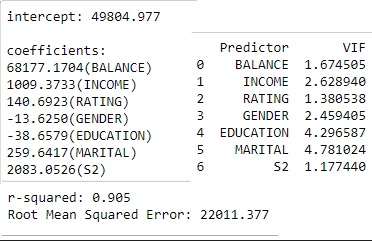


Figure. After removal of AGE independent factor

The “AGE” variable has a reasonably high VIF of 10.039706 and indicates that this variable has high likelihood of interdependency against the rest of the variables. After removing “AGE” variable, the VIF figures for the remaining variables improved. However, no significant impact can be seen from R2 and Root Mean Squared Error results. Thus, removal of “AGE” does not materially alter the accuracy and reliability of the model.

The MLR model has obtained an R2 or the coefficient of determination of 0.905. This figure is a measure that determines the proportion of variance in the dependent variable that can be explained by the independent variables. Simply put, R2 is the goodness of fit to the MLR. It can take a value between 0 to 1. The figure however, does not disclose information about the causation relationship between the independent and dependent variables. In addition, adding more independent variables always increase and never decrease the figure. The result of 0.905 means the model explains for 90.5% of the variability in B1 through the independent variables. This is a good accurate result.

Another related measure is the adjusted R2, Radj2, which also measures the same thing, but it also adjusts for the number of independent variables within the model. If more and more useless independent variable is added to the model, Radj2 will decrease. If more useful independent variable is added, it will increase. In addition, Radj2 is always lower than R2. The result shows 0.905, very similar but marginally lower value than R2. It also implies that most of the independent variables used in the model are useful.

In summary, the result is showing that the model’s prediction for B1 is fairly accurate and reliable based on the chosen independent variables.

Question 6:

The code from Jupyter notebook are as follows:

import pandas as pd

import numpy as np

import os as path

import matplotlib.pyplot as plt

import seaborn as sns

import missingno as msno

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

from sklearn.preprocessing import StandardScaler

from sklearn.linear\_model import LinearRegression

from sklearn.metrics import mean\_absolute\_error

from sklearn.metrics import mean\_squared\_error

from statsmodels.stats.outliers\_influence import variance\_inflation\_factor

import statsmodels.api as sm

####Defining/Importing data source:

path.chdir("D:\My Docs\Studies\Singapore Uni of Social Science (SUSS)\Jul'22 registration\Study Materials\ANL252\ECA\Original")

path.getcwd()

df\_profile = pd.read\_csv("ECA\_data.csv")

pd.set\_option("display.max\_rows",300)

df\_profile.info()

df\_profile.nunique()

#Data Pre-processing stage

#There are 13 null values under EDUCATION columns and 38 null values under MARITAL columns. The code and results as follows:

df\_profile.isnull().sum()

#The following code shows the null values above as a percentage of the number of records within the dataset.

missing\_pct = df\_profile.isnull().sum()/len(df\_profile)

df\_missing = pd.DataFrame({"% missing":missing\_pct})

df\_missing

#Heatmap shows the correlation of missingness between every 2 columns

#A value near 0 means there is no dependence between the occurrence of missing values of two variables. Therefore, there're no correlations.

msno.heatmap(df\_profile)

df\_profile.skew()

#### Data Pre-processing stage - Task 1:

#Since proportion of missing values for education is miniscule and the distribution is highly symmetrical, as evident from the skewness value very close to 0(between , the missing education values will be replaced with the most commonly occuring education level.

df\_profile["EDUCATION"].fillna(df\_profile["EDUCATION"].mode()[0], inplace=True)

df\_profile.isnull().sum()

#### Data Pre-processing stage - Task 2:

#Removing THREE duplicate IDs, namely, 132, 378 and 420

df\_profile[df\_profile.duplicated(subset="ID", keep=False)==True]

df\_profile.drop\_duplicates(subset="ID", inplace=True)

df\_profile.reset\_index(inplace=True)

df\_profile[df\_profile.duplicated(subset="ID", keep=False)==True]

#### Interim Transformation task:

This is to ascertain the spread and range of INCOME values.

pd.unique(df\_profile["INCOME"].sort\_values())

print(pd.unique(df\_profile["INCOME"].sort\_values()))

#Discretise the INCOME values into different income range to facilitate data-preoprocessing and subsequent analysis. Since the minimum value is 10,000 and maximum value is 908,846, the most appropriate bin size is 200,000, resulting in 5 income-range categories.

#bin\_range = [1,100000,200000,300000,400000,500000,600000,700000,800000,900000,1000000]

#bin\_label = ["100K","200K","300K","400K","500K","600K","700K","800K","900K","1000K"]

bin\_range = [1, 200000, 400000, 600000, 800000, 1000000]

bin\_label = ["200K", "400K", "600K", "800K", "1000K"]

df\_profile["INCOME\_bins"] = pd.cut(x=df\_profile["INCOME"], bins=bin\_range, labels=bin\_label, right=False)

df\_profile["INCOME\_bins"]

#### Data Pre-processing stage - Task 3:

Records with missing marital status largely fall under Income group of 200K and 400K.

df\_profile[df\_profile["MARITAL"].isnull()]

df\_profile.groupby(by=["INCOME\_bins","EDUCATION"]).agg({"MARITAL":pd.Series.mode})

df\_profile["MARITAL"] = np.where((df\_profile["MARITAL"].isnull()) & (df\_profile["INCOME\_bins"]=="400K") & (df\_profile.EDUCATION==1), 2, df\_profile["MARITAL"])

df\_profile["MARITAL"] = np.where((df\_profile["MARITAL"].isnull()) & (df\_profile["INCOME\_bins"]=="400K") & (df\_profile.EDUCATION==2), 1, df\_profile["MARITAL"])

df\_profile["MARITAL"] = np.where((df\_profile["MARITAL"].isnull()) & (df\_profile["INCOME\_bins"]=="400K") & (df\_profile.EDUCATION==3), 1, df\_profile["MARITAL"])

df\_profile["MARITAL"] = np.where((df\_profile["MARITAL"].isnull()) & (df\_profile["INCOME\_bins"]=="200K") & (df\_profile.EDUCATION==1), 2, df\_profile["MARITAL"])

df\_profile["MARITAL"] = np.where((df\_profile["MARITAL"].isnull()) & (df\_profile["INCOME\_bins"]=="200K") & (df\_profile.EDUCATION==2), 2, df\_profile["MARITAL"])

df\_profile["MARITAL"] = np.where((df\_profile["MARITAL"].isnull()) & (df\_profile["INCOME\_bins"]=="200K") & (df\_profile.EDUCATION==3), 1, df\_profile["MARITAL"])

df\_profile[df\_profile["MARITAL"].isnull()]

#### Data Pre-processing stage - Task 4:

AGE values cannot be -1 and 199. As AGE values are positively skewed and there are only FIVE error values, median values will be used.

df\_profile["AGE"].describe()

df\_profile[(df\_profile["AGE"] == -1) | (df\_profile["AGE"] == 199)]

df\_profile["AGE"].replace(to\_replace={-1: df\_profile["AGE"].median(), 199: df\_profile["AGE"].median()}, value=None, inplace=True)

df\_profile[(df\_profile["AGE"] == -1) | (df\_profile["AGE"] == 199)]

#### Find values under column R3 with "$" string value:

df\_profile[df\_profile["R3"].str.find("$")!=-1]

error\_dict = {"$0": 0, "$2,620": 2620, "$6,000": 6000, "$2,200": 2200, "$390": 390, "$2,688": 2688, "$13,069": 13069, "$7,000": 7000, "$5,000": 5000, "$2,089": 2089}

df\_profile["R3"].replace(to\_replace=error\_dict, value=None, inplace=True)

df\_profile.astype(dtype={"R3": "int"}, copy=False)

## Generate Insights

#### Insight 1:

Generally, people do not stretch their credit limit too far. Majority of the customers use only 25% of their credit limit, with only a minority spending 50% or more.

In addition, there are more female credit card users than males.

df\_profile.head(300)

temp = []

util\_grp=[]

temp = (df\_profile["BALANCE"]/df\_profile["LIMIT"])\*100

for i in range(len(temp)):

if(temp[i] <= 25):

util\_grp.append("25%")

elif(temp[i] <= 50):

util\_grp.append("50%")

else:

util\_grp.append("75%")

df\_profile["Credit\_Util"]=util\_grp

df\_profile["Credit\_Util"].head(140)

df\_result = pd.pivot\_table(df\_profile, values="ID", index="Credit\_Util", columns="GENDER", aggfunc="count")

print(df\_result)

df\_result.plot.bar(stacked=True)

plt.title("Utilization of Credit Limit", fontsize=16)

#### Insight 2:

Majority of customers has only credit limit that is within 1 times their annual income. However, a significant proportion of customers still holds credit limit that is twice their annual income.

temp = []

temp = (df\_profile["LIMIT"]/df\_profile["INCOME"])

cmultiple\_grp = []

for i in range(len(temp)):

if(temp[i] <= 1):

cmultiple\_grp.append("1x")

elif(temp[i] <= 2):

cmultiple\_grp.append("2x")

else:

cmultiple\_grp.append(">3x")

df\_profile["CreditMultiple"]=cmultiple\_grp

df\_profile["CreditMultiple"].head()

df\_result = pd.pivot\_table(df\_profile, values="ID", index="CreditMultiple", columns="GENDER", aggfunc="count")

print(df\_result)

df\_result.plot.bar(stacked=True)

plt.title("Utilization of Credit Limit", fontsize=16)

#### Interim Transformation task

bin\_range = [0, 20, 30, 40, 50, 60, np.inf]

bin\_label = ["<=20 y.o.", "21~30 y.o.", "31~40 y.o.", "41~50 y.o.", "51~60 y.o.", ">=60 y.o."]

df\_profile["AGE\_bins"] = pd.cut(x=df\_profile["AGE"], bins=bin\_range, labels=bin\_label, right=False)

df\_profile["AGE\_bins"]

#### Insight 3:

Profile of a person with prompt payment behaviour, i.e. paying on time for 5 consecutive months. Knowing this profile, allows credit facility to market the right credit product to customers. Based on the charts below, it would seem that a prompt paying customer exhibit the following characteristics:

-have good credit rating

-around the age of 31 to 40 years old

-between 31 to 40 years of age

-received up till tertiary or post-graduate education

-either single or married

-normally earning annual salary range below 400K

df\_result = df\_profile[(df\_profile["S1"]==-1) & (df\_profile["S2"]==-1) & (df\_profile["S3"]==-1) & (df\_profile["S4"]==-1) & (df\_profile["S5"]==-1)]

df\_result

result\_rating = df\_result.groupby(by="RATING").agg({"ID":["count"]})

result\_rating.plot(kind="bar")

plt.title("Credit Rating of Prompt Paying Customer", fontsize=16)

result\_age = df\_result.groupby(by="AGE\_bins").agg({"ID":["count"]})

result\_age.plot(kind="bar")

plt.title("Age groups of Prompt Paying Customer", fontsize=16)

result\_education = df\_result.groupby(by="EDUCATION").agg({"ID":["count"]})

result\_education.plot(kind="bar")

plt.title("Educational Level of Prompt Paying Customer", fontsize=16)

result\_marital = df\_result.groupby(by="MARITAL").agg({"ID":["count"]})

result\_marital.plot(kind="bar")

plt.title("Marital Status of Prompt Paying Customer", fontsize=16)

result\_income = df\_result.groupby(by="INCOME\_bins").agg({"ID":["count"]})

result\_income.plot(kind="bar")

plt.title("Income Range of Prompt Paying Customer", fontsize=16)

#### Insight 4:

Net owing calculates how much outstanding amount is still owing to the credit facility by deducting total repayments(R1 to R5) from total billings(B1 to B5). The dataset is then filtered to present only risk groups of concern where the amount owed exceeds credit limit and their corresponding RATING category.

If further examination is performed, would reveal that many such customer are only able to make minimum repayment or worse, i.e. Sn=0 and above, for many consecutive periods.

From the chart, it seems that many of the "good(0)" rating customers actually fall under this risk group and their credit RATING would reasonably need a review.

df\_profile["R3\_cast"] = df\_profile["R3"].astype("int64", copy=False)

df\_profile.drop("R3", axis=1, inplace=True)

df\_profile.rename(columns={"R3\_cast": "R3"}, inplace=True)

Bcol\_lst = ["B1", "B2", "B3", "B4", "B5"]

Rcol\_lst = ["R1", "R2", "R3", "R4", "R5"]

df\_profile["Net Owing"] = df\_profile[Bcol\_lst].sum(axis=1) - df\_profile[Rcol\_lst].sum(axis=1)

df\_riskgrp = df\_profile[df\_profile["Net Owing"]>df\_profile["LIMIT"]]

df\_riskgrp

df\_riskgrp.pivot\_table(values="ID", index="AGE\_bins", columns="RATING", aggfunc="count").plot.bar(stacked=True)

#### Insight 5:

#To determine if there are any existing relationship between the variables to allow credit facility to carry out better customer segmentation. From the outcome of the heatmap, it can be seen there're very low relationship amongst them. The strongest relationship is between INCOME and LIMIT which is rightful because the amount of credit limit given to customers is based on the amount income they can earn.

small\_data = ['LIMIT', 'BALANCE', 'INCOME', 'RATING', 'GENDER', 'EDUCATION', 'MARITAL', 'AGE', 'S2']

small\_data = df\_profile[small\_data].copy()

sns.heatmap(small\_data.corr(), cmap = 'Wistia', annot= True)

plt.show(sns)

## Building Multiple Linear Regression model to predict dependent variable, B1

This is a simple linear regression model with more than one predictor variable and is modelled by equation: Yₑ = α + β₁X₁ + β₂X₂ + … + βₚXₚ, where

-p is the number of predictors

-Yₑ is the dependent variable

-Xₚ is the predictor variable

-βₚ is the average effect on Yₑ of one unit increase in Xₚ, assuming all other predictors remains unchanged

Based on the dataset scenario, the model will be written as: B1 = α + (β1 x BALANCE) + (β2 x INCOME) + (β3 x RATING) + (β4 x GENDER) + (β5 x EDUCATION) + (β6 x MARITAL) + (β7 x AGE) + (β8 x S2)

# select which numerical data as predictor/explanatory variables

# TODO exclude/include to arrive at the lowest RMSE and R²

predictor\_vars = ["BALANCE", "INCOME", "RATING", "GENDER", "EDUCATION", "MARITAL", "AGE", "S2"]

# predictor\_vars = ["LIMIT", "BALANCE", "INCOME", "AGE"]

x\_final = df\_profile[predictor\_vars].copy()

# define B1 as dependent variable

y\_final = df\_profile[["B1"]].copy()

#### Multicollinearity and Calculating the Variance Inflation Factor(VIF):

Multicollinearity occurs when independent variables in the regression model becomes highly correlated to each other, making the model result difficult to interpret. It also cause an overfitting problem. This is a common condition that people test before selecting the variables for the regression model.

from statsmodels.stats.outliers\_influence import variance\_inflation\_factor

# create VIF dataframe

vif\_data = pd.DataFrame()

vif\_data["Predictor"] = x\_final.columns

# calculating VIF for each feature

vif\_data["VIF"] = [variance\_inflation\_factor(x\_final.values, i)

for i in range(len(x\_final.columns))]

print(vif\_data)

# PRE\_PROCESSING

# split sample data into 30/70;

# 30% for testing the model (making prediction), and 70% for training the model

x\_train, x\_test, y\_train, y\_test = train\_test\_split(x\_final, y\_final, test\_size=0.30, random\_state=0)

# PRE\_PROCESSING

# convert both train and test data to standard normal distribution

scaler = StandardScaler()

x\_train = scaler.fit\_transform(x\_train.astype(np.float64))

x\_test = scaler.transform(x\_test.astype(np.float64))

# RUN LINEAR REGRESSION TO TRAIN ALGORITHM

lr = LinearRegression().fit(x\_train, y\_train)

# INTERPRET RESULTS

# display intercept / constant term

print('intercept: %.3f' % (lr.intercept\_))

# display coefficients of predictors

print('\ncoefficients:')

for i,j in enumerate(predictor\_vars, 0):

print('{0:.4f}({1})'.format(lr.coef\_[0][i], j))

#MAKING PREDICTIONS TO SEE HOW ACCURATE ALGORITHM IS IN PREDICTING "B1" VARIABLE

# make prediction using test data

# y\_train\_pred = lr.predict(x\_train)

y\_test\_pred = lr.predict(x\_test)

df\_compare = y\_test.copy()

df\_compare.rename(columns={"B1":"B1\_Actual"}, inplace=True)

df\_compare["B1\_Predicted"] = y\_test\_pred

df\_compare

#### Visualize the comparison records:

Because the number of records is huge, it is not possible to visualize the entire record.

df\_select = df\_compare.head(150)

df\_select.plot(kind='bar',figsize=(16,10))

plt.grid(which='major', linestyle='-', linewidth='0.5', color='green')

plt.grid(which='minor', linestyle=':', linewidth='0.5', color='black')

plt.show()

#### Evaluate performance of algorithm:

Both RMSE and R² tells us how well a regression model fits the dataset.

The RMSE tells us the average deviation between the model's predicted B1 and actual B1 value. The lower the RMSE, the better the model fits the dataset.

On the other hand, R² tells us how well a model can predict the value of the dependent variable in percentage terms.

Based on the result, R² of 0.905 means the model explains 90.5% of the variability in B1 can be explained by .....

# R-Squared is how well regression model explains the observed data;

# 0.90 reveals that 90% of variability observed in target variable is explained by the model

print('r-squared: %.3f' % (lr.score(x\_test, y\_test)))

# RMSE refers to root mean squared error;

# This tells us that the average deviation between the predicted and actual B1 value is 22008.218

# larger values mean larger error

print('Root Mean Squared Error: {:.3f}'.format(np.sqrt(mean\_squared\_error(y\_test, y\_test\_pred))))

X2 = sm.add\_constant(x\_train)

model\_stats = sm.OLS(y\_train.values.reshape(-1,1), X2).fit()

model\_stats.summary()

# References

Wu, K. Y. (2022). *ANL252 Python for data analytics (study guide).* Singapore University of Social Sciences.

Zheng, F., Soh, I., & Tan, C. (2021). *BUS105 Statistics (study guide).* Singapore: Singapore University of Social Sciences.