Logo

Description automatically generated

**ANL252**

**Python for Data Analytics**

**End-of-Course Assessment - July Semester 2022**

**July 2022 Presentation**

**Submitted by:**

|  |  |
| --- | --- |
| **Name** | **PI No.** |
| **Timothy Yap** | **K2111660** |

**Tutorial Group: T09**

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

**Submission Date: 4 September 2022**

Contents

[Q1 3](#_Toc113219487)

[Q2 4](#_Toc113219488)

[Q3 6](#_Toc113219489)

[Q4 13](#_Toc113219490)

[Q5 17](#_Toc113219491)

[References 19](#_Toc113219492)

## Q1

The variable types (i.e. categorical or numeric) in this dataset are appended in the table below:

|  |  |
| --- | --- |
| **Variable** | **Type** |
| ID | Numeric |
| LIMIT | Numeric |
| BALANCE | Numeric |
| INCOME | Numeric |
| GENDER | Categorical |
| EDUCATION | Categorical |
| MARTIAL | Categorical |
| AGE | Numeric |
| S(n) | Numeric |
| B(n) | Numeric |
| R(n) | Numeric |
| RATING | Categorical |

## Q2

The first data pre-processing task is the handling of missing values. In this dataset, two of the columns have missing values, namely “EDUCATION” and “MARITAL”. Specifically, there are 13 missing values in the “EDUCATION” field and 38 missing values in the “MARITAL” field. Given that both these columns deal with either ordinal (i.e. “EDUCATION”) or nominal data (i.e. ‘’MARTIAL”), we can replace the missing values with the respective mode values of each field using Pandas’s fillna() function.

The second data pre-processing task is the handling of invalid values in the ‘AGE’ column. There are two values, namely -1 and 199, that are invalid. Specifically, there are five observations each that are -1 and 199. Given that both these fields deal with ratio data, these invalid values can be replaced with the median age (i.e. 34).

The third data pre-processing task is handling of invalid values in the ‘B1’, ‘B2’, ‘B3’, ‘B4’ and ‘B5’ columns. In these five columns, there are negative values. This is not possible, given that this pertains to billable amount and hence should only contain positive values. Given that these values deal with ratio data, these invalid values can be replaced with the respective median values from each of the 5 columns.

The fourth data pre-processing task is handling of invalid values in the ‘R3’ column. There are values that have a “$” prefix and others that have commas in them. This will pose an issue when generating a regression model later on as these data points will be treated as strings. Thus, these characters (i.e “$” and “,”) are removed and the ‘R3’ column thereafter casted as an integer.

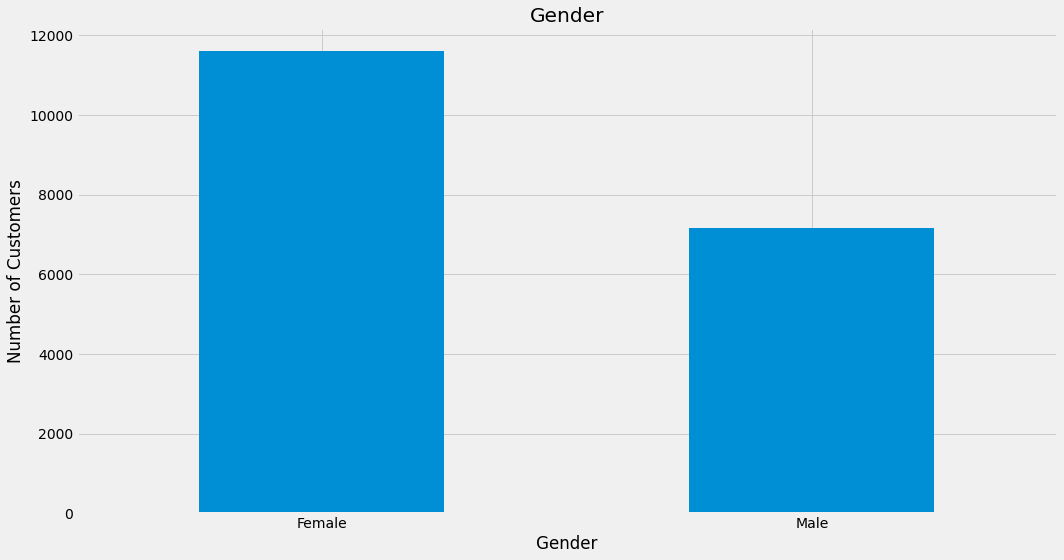
The code to perform these data cleaning procedures is appended below:

*#------Q2-----  
  
#To import the necessary library that is required for the code to run  
import* pandas *as* pd  
  
*#to read the CSV as a Pandas DataFrame*raw\_df = pd.read\_csv(r"C:\Users\timot\OneDrive\Documents\SUSS\ANL252 - Python\ECA\Raw data\ECA\_data.csv", index\_col='ID')  
  
*#to confirm the data types in columns read in the DataFrame*print(raw\_df.dtypes)  
  
*#to confirm if there are any missing values present, and if yes how many and in which column*print(raw\_df.isnull().sum())  
  
*#a copy of raw\_df is created for the purposes of cleaning the data*clean\_df = raw\_df.copy()  
  
  
*#1ST DATA PRE-PROCESSING TASK  
#as both Education and Marital are nominal data points, the missing values are replaced with the mode from each category*clean\_df['EDUCATION'].fillna(clean\_df['EDUCATION'].mode()[0], inplace=*True*)  
clean\_df['MARITAL'].fillna(clean\_df['MARITAL'].mode()[0], inplace=*True*)  
*#this is to confirm that the above procedure have correctly been effected and there are no longer any missing values*print(clean\_df.isnull().sum())  
  
  
*#2ND DATA PRE-PROCESSING TASK  
#as both Age is a ratio data point, the invalid values are replaced with the median value*clean\_df["AGE"].replace(to\_replace=[-1,199], value=clean\_df["AGE"].mode()[0], inplace=*True*)  
  
  
*#3RD DATA PRE-PROCESSING TASK  
#this is to identify the invalid billing values in columns B1 to B5 (i.e. the values that are negative) via boolean clean\_masking*B1\_filt = (clean\_df['B1'] < 0)  
B2\_filt = (clean\_df['B2'] < 0)  
B3\_filt = (clean\_df['B3'] < 0)  
B4\_filt = (clean\_df['B4'] < 0)  
B5\_filt = (clean\_df['B5'] < 0)  
  
  
clean\_df['B1'].loc[B1\_filt] = clean\_df['B1'].loc[~B1\_filt].median()  
clean\_df['B2'].loc[B2\_filt] = clean\_df['B2'].loc[~B2\_filt].median()  
clean\_df['B3'].loc[B3\_filt] = clean\_df['B3'].loc[~B3\_filt].median()  
clean\_df['B4'].loc[B4\_filt] = clean\_df['B4'].loc[~B4\_filt].median()  
clean\_df['B5'].loc[B5\_filt] = clean\_df['B5'].loc[~B5\_filt].median()  
  
  
*#4TH DATA PRE-PROCESSING TASK  
#this is to replace the invalid value in the 'R3' columns, i.e. $0, with 0 instead*clean\_df["R3"] = clean\_df["R3"].str.replace('$', '', regex=*True*)  
clean\_df["R3"] = clean\_df["R3"].str.replace(',', '', regex=*True*)  
*#this is go recast the 'R3' column as an integer*clean\_df["R3"].astype(int)

## Q3

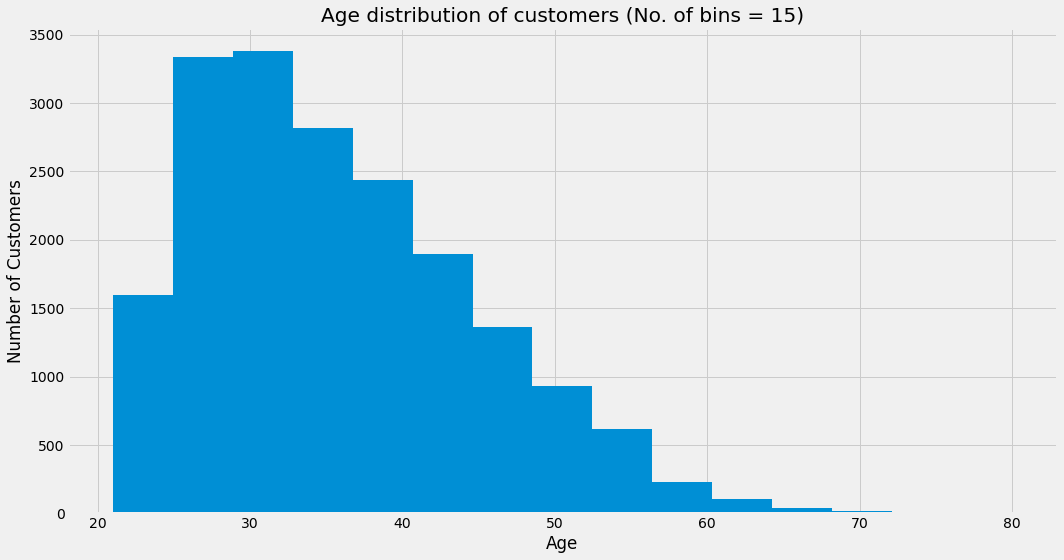
The first relevant insight is that the credit facility has significantly more female than male customers. Specifically, it has 11,605 female customers and 7,164 male customers.

We can graphically represent this via the following vertical bar chart:



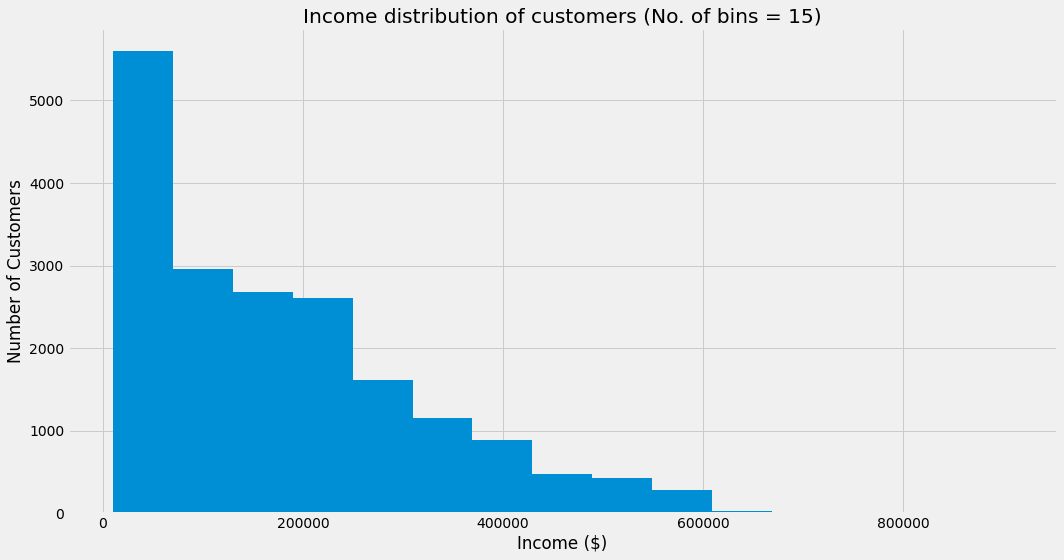
The second relevant insight is that of the distribution of the age groups of the credit facility’s customers. Specifically, the distribution of the ages of the credit facility’s customers has a long-tailed distribution and is positively skewed. This observation is confirmed as the mode age (29 years old) is lesser than the median age (34 years old) which is lesser than the mean age (35.5 years old [to 1 d.p.]).

We can graphically represent this via the histogram below, where there are 15 bins in total:



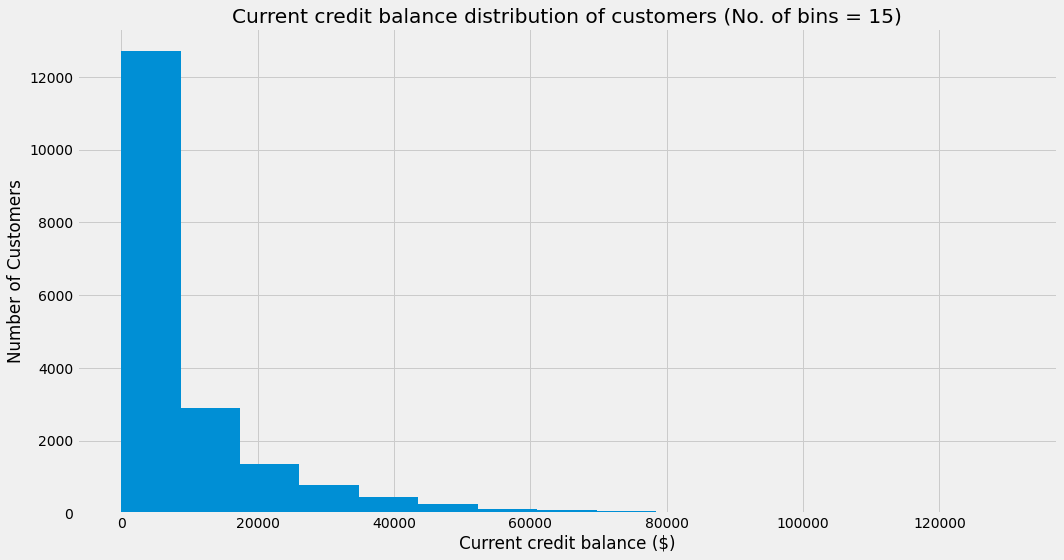
The third relevant insight is that of the distribution of income of the credit facility’s customers. Specifically, the distribution of the income of the credit facility’s customers has a long-tailed distribution and is positively skewed. It follows an exponential decay. This observation is confirmed as the mode income ($10,000) is lesser than the median income ($148,225) which is lesser than the mean income ($177,765.81 [to 2 d.p.]).

We can graphically represent this via the histogram below, where there are 15 bins in total:



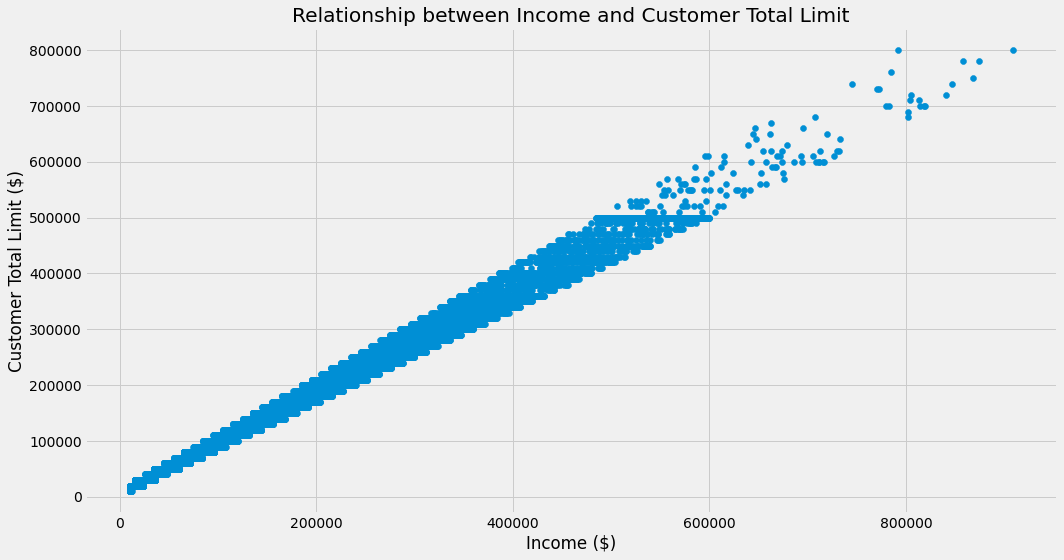
The fourth relevant insight is that of the distribution of the current credit balance of the credit facility’s customers. Specifically, the distribution of the current credit balance of the credit facility’s customers has a long-tailed distribution and is very positively skewed. It follows an exponential decay. This observation is confirmed as the mode current credit balance ($0) is lesser than the median current credit balance ($3,959.02 [to 2.dp.]) which is lesser than the mean current credit balance ($9,119.89 [to 2 d.p.]).

We can graphically represent this via the histogram below, where there are 15 bins in total:



The fifth relevant insight is that of the relationship between customer’s current income and customer total limit. There is a strong positive correlation between customer’s current income and customer total limit. This is to be expected as the credit facility will assess customers with higher incomes as those with having a lower risk of defaulting, and thus willing to extend a more liberal line of credit. However, an interesting observation is the higher the customer’s income, the more variation there is in their total limit. This is especially prevalent for total limits above $500,000

These observations above can be represented via the scatter plot below:



The code to produce the above charts is appended below:

*#------Q3-----  
  
#To import the necessary library that is required for the code to run  
import* matplotlib.pyplot *as* plt

*#this is to display the histograms of all variables in the dataset for high-level data exploration*clean\_df.hist(bins=10, figsize=(15, 10))

*#a copy of clean\_df is created (i.e. df\_Q3) for the purposes of further data manipulation for Q3  
#to plot the distribution of gender of customers, we can map the 0 and 1 values to 'Male' and 'Female' respectively  
#this also allows for a more intuitive interpretation of the dataframe*df\_Q3 = clean\_df.copy()  
df\_Q3['GENDER\_STRING'] = df\_Q3['GENDER']  
df\_Q3['GENDER\_STRING'].replace(to\_replace=[0,1], value=['Male','Female'], inplace=*True*)  
  
*#a function is defined to determine the skewness of distribution is defined  
#this function takes 3 arguments - mode, mean and mean  
 #if there is no mode value, the user will have to input 'no mode' as the argument for mode  
def* skewness(mode,median,mean):  
 *if not* mode == 'no mode':  
 *if* mode < median *and* median < mean:  
 print("The data is positively skewed")  
 *elif* mean < median *and* median < mode:  
 print("The data is negatively skewed")  
 *elif* mean == median *and* median == mode:  
 print("The data is normally distributed")  
 *elif* mode == 'no mode':  
 *if* median < mean:  
 print("The data is positively skewed")  
 *elif* median < mode:  
 print("The data is negatively skewed")  
 *elif* median == mode:  
 print("The data is normally distributed")  
  
*#FIRST RELEVANT INSIGHT  
#matplotlib is used to create a vertical bar chart with regards to Gender (Female vs Male). The parameters (e.g. title, style, etc.) are specified accordingly.*plt.figure(figsize=(15,8))  
plt.style.use('fivethirtyeight')  
  
gender = [0,1]  
title = 'Gender'  
  
*# plt.hist(df\_1b['GENDER\_STRING'],bins=['Male','Female'], color='darkorange', width=0.1)*df\_Q3['GENDER\_STRING'].value\_counts().plot(kind='bar')  
  
plt.xticks(rotation=0)  
plt.xlabel('Gender')  
plt.ylabel("Number of Customers")  
plt.title(title)  
  
plt.tight\_layout()  
  
plt.show()  
  
  
*#SECOND RELEVANT INSIGHT  
#matplotlib is used to create a histogram of the credit facility customers' age groups. The parameters (e.g. title, style, etc.) are specified accordingly.*plt.figure(figsize=(15,8))  
plt.style.use('fivethirtyeight')  
  
bin\_size = 15  
title = f'Age distribution of customers (No. of bins = {bin\_size})'  
  
plt.hist(df\_Q3['AGE'], bins=bin\_size)  
  
plt.xticks(rotation=0)  
plt.xlabel('Age')  
plt.ylabel("Number of Customers")  
plt.title(title)  
  
plt.tight\_layout()  
  
plt.show()  
  
*#to confirm that the distribution of age groups is indeed positively skewed*mode\_age = df\_Q3['AGE'].mode()  
median\_age = df\_Q3['AGE'].median()  
mean\_age = df\_Q3['AGE'].mean()  
  
print(f'The mode age is {mode\_age[0]} years old')  
print(f'The median age is {median\_age} years old')  
print(f'The mean age is {mean\_age} years old')  
  
skewness(mode\_age[0], median\_age, mean\_age)  
  
  
*#THIRD RELEVANT INSIGHT  
#matplotlib is used to create a histogram of the credit facility customers' income levels. The parameters (e.g. title, style, etc.) are specified accordingly.*plt.figure(figsize=(15,8))  
plt.style.use('fivethirtyeight')  
  
bin\_size = 15  
title = f'Income distribution of customers (No. of bins = {bin\_size})'  
  
plt.hist(df\_Q3['INCOME'],bins=bin\_size)  
  
plt.xticks(rotation=0)  
plt.xlabel('Income ($)')  
plt.ylabel("Number of Customers")  
plt.title(title)  
  
plt.tight\_layout()  
  
plt.show()  
  
*#to confirm that the distribution of income is indeed positively skewed*mode\_income = df\_Q3['INCOME'].mode()  
median\_income = df\_Q3['INCOME'].median()  
mean\_income = df\_Q3['INCOME'].mean()  
  
print(f'The mode income is ${round(mode\_income[0],2)}')  
print(f'The median income is ${round(median\_income,2)}')  
print(f'The mean income is ${round(mean\_income,2)}')  
  
skewness(mode\_income[0], median\_income, mean\_income)  
  
  
*#FOURTH RELEVANT INSIGHT  
#matplotlib is used to create a histogram of the credit facility customers' current credit balance. The parameters (e.g. title, style, etc.) are specified accordingly.*plt.figure(figsize=(15,8))  
plt.style.use('fivethirtyeight')  
  
bin\_size = 15  
title = f'Current credit balance distribution of customers (No. of bins = {bin\_size})'  
  
plt.hist(df\_Q3['BALANCE'],bins=bin\_size)  
  
plt.xticks(rotation=0)  
plt.xlabel('Current credit balance ($)')  
plt.ylabel("Number of Customers")  
plt.title(title)  
  
plt.tight\_layout()  
  
plt.show()  
  
*#to confirm that the distribution of current credit balance is indeed positively skewed*mode\_balance = df\_Q3['BALANCE'].mode()  
median\_balance = df\_Q3['BALANCE'].median()  
mean\_balance = df\_Q3['BALANCE'].mean()  
  
print(f'The mode balance is ${round(mode\_balance[0],2)}')  
print(f'The median balance is ${round(median\_balance,2)}')  
print(f'The mean balance is ${round(mean\_balance,2)}')  
  
skewness(mode\_balance[0], median\_balance, mean\_balance)  
  
  
*#FIFTH RELEVANT INSIGHT  
#matplotlib is used to create a scatter plot of the credit facility customers' Income and Total Limit. The parameters (e.g. title, style, etc.) are specified accordingly.*plt.figure(figsize=(15,8))  
plt.style.use('fivethirtyeight')  
  
title = 'Relationship between Income and Customer Total Limit'  
  
plt.scatter(df\_Q3['INCOME'],df\_Q3['LIMIT'])  
  
plt.xticks(rotation=0)  
plt.xlabel('Income ($)')  
plt.ylabel("Customer Total Limit ($)")  
plt.title(title)  
  
plt.tight\_layout()  
  
plt.show()

## Q4

The first step is to examine the correlations between the various variables in the dataset. This can be done using Pandas’ corr() method.

To produce an accurate regression model, it is imperative there are no significant correlation amongst the predictor variables in the regression model so as to prevent multicollinearity issues.

Firstly, it can be observed the S(n), B(n) and R(n) variables each have strong autocorrelations between a particular value of itself and its lagged values. To circumvent this issue, only the latest predictor variables (i.e. S(1), R(1)) are included in the regression model.

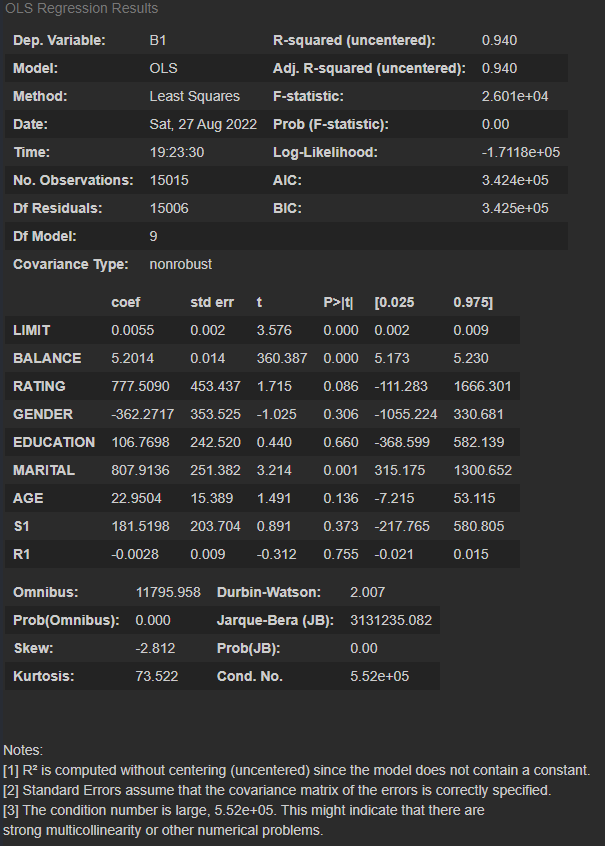
Thereafter, we can examine the remaining variables to see if there are any potential multicollinearity issues. As ‘LIMIT’ and ‘INCOME’ are strongly correlated (i.e. as discussed in Q3), the ‘INCOME’ variable will not be included in the regression model.

With the remaining variables, we can then split the dataset into training and testing partitions. In this case, 80% of the dataset is allocated for training and 20% for testing.

The OLS (ordinary least squares) module from the statsmodels library is imported to determine which of the remaining variables should be dropped from the regression model. Specifically, this module will allow us to compute the p-values of each variable and determine for which one(s) we are unable to reject the null hypothesis that there exists no correlation between the predictor variable and target variable. In this case, a threshold of 0.05 is used (i.e. any variable with a p-value of above 0.05 will be dropped from the regression model)

This extra library (i.e. statsmodels) has to be installed and imported as there is no function/method in scikit-learn that allows for calculation of these statistics

After running the code, the following summary table is given:



As ‘RATING’, ‘GENDER’, ‘EDUCATION’, ’AGE’, ‘S1’ and ‘R1’ have p-values greater than 0.05, these variables are dropped from the regression model. Following this, the training and testing data have to be re-partitioned again using the same parameters.

Thereafter, the OLS function is run again for the remainder variables to confirm that their p-values are all below 0.05 and that they are to be included in the regression model:



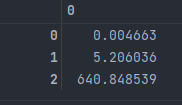
It is noted that normalisation/standardisation is not needed when constructing a linear regression model. This is because the calculation of the coefficients in the model will compensate for any potential scaling issues. In fact, in this case, normalisation/standardisation will compromise the interpretability of the linear regression equation (i.e. since ‘LIMIT’ and ‘BALANCE’ already use the same unit of measurement to begin with).

The code to construct this regression model is appended below:

*#------Q4-----  
  
#To import the necessary libraries that are required for the code to run  
from* sklearn.model\_selection *import* train\_test\_split  
*from* sklearn.linear\_model *import* LinearRegression  
*from* statsmodels.api *import* OLS  
  
*#the relationship between the various variables in the dataset is first explored to determine which should be included in the regression model*corr\_df = clean\_df.corr()  
print(corr\_df)  
  
*#as the S(n), B(n) and R(n) variables each have strong auto-correlations between the current value and its lagged values, the lagged values are dropped from the dataframe and excluded from the regression model*corr\_df.drop(['B2','B3','B4','B5','S2','S3','S4','S5','R2','R4','R5'],axis='columns', inplace=*True*)  
corr\_df.drop(['B2','B3','B4','B5','S2','S3','S4','S5','R2','R4','R5'],axis='rows', inplace=*True*)  
print(corr\_df[corr\_df>=0.75])  
  
*#the dataset is first partitioned into both training and testing datasets.  
#80% of the dataset will be used for training the model, and 20% for testing the model accuracy  
#the variables that are auto-correlated or have multicollinearity (i.e. 'LIMIT' and 'INCOME')are not included in the regression model*x = clean\_df[['LIMIT','BALANCE','RATING','GENDER','EDUCATION','MARITAL','AGE','S1','R1']]  
y = clean\_df['B1']  
  
x\_train, x\_test, y\_train, y\_test = train\_test\_split(x,y,test\_size=0.2,random\_state=0)  
  
*#this is to compute the p-values of all of the predictor variables*print(OLS(y\_train,x\_train).fit().summary())  
  
*#the variables that have a p-values > 0.05 are dropped from the regression model*x.drop(['RATING','GENDER','EDUCATION','AGE','S1','R1'], axis='columns', inplace=*True*)  
  
*#as the x dataframe has changed, the training and testing data will have to be re-partioned*x\_train, x\_test, y\_train, y\_test = train\_test\_split(x,y,test\_size=0.2,random\_state=0)  
  
*#this is to compute the p-values of all of the remaining predictor variables*print(OLS(y\_train,x\_train).fit().summary())  
  
*#this is to create an instance of the LinearRegression object as LM*lm =LinearRegression()  
  
*#this is to initialise the linear regression model using the training data*lm.fit(x\_train,y\_train)

## Q5

The LinearRegression coef\_ attribute can be used to derive the coefficients of the regression equation:



In this case, 0 is ‘LIMIT’, 1 is ‘BALANCE’ and 2 is ‘MARITAL’.

The LinearRegression intercept\_ attribute can be used to derive the y-intercept of the regression equation, that is 1306.308750.

Thus, the regression equation is as follows:

A direct interpretation of the y-intercept reveals for a customer with a total limit of $0, a current credit balance of $0 and who is single, his/her expected billable amount in the most recent month is $1306.31 (to 2 d.p.). However, this does not exactly make sense as it is impossible for there to be a customer who has a total limit of $0. Hence in this regression model, it may not be meaningful to attempt to interpret the y-intercept value.

The regression model reveals that a higher customer total limit and a higher customer current credit balance will result in a higher billable amount in the most recent month (i.e. B1).

Further to the abovementioned observation, the customer current credit balance has a significantly larger influence on the customer billable amount in the most recent month as compared to the customer limit. This is evident from the higher coefficient of ‘BALANCE’ as compared to ‘LIMIT’ (5.206036 vs 0.004663)

Specifically, assuming all the other predictor variables stay constant, for every $1 increase in customer limit, the billable amount in the most recent month increases by $0.004663. Secondly, assuming all the other predictor variables stay constant, for every $1 increase in customer current credit balance, the billable amount in the most recent month increases by $5.21 (to 2 d.p.).

The regression model also reveals that of the three marital status, ‘Others’ has no impact on B1 [i.e., ]; ‘Single’ has a positive impact on B1 and ‘Married’ has the highest positive impact on B1. Specifically, assuming all the other predictor variables stay constant, customers that are single will have a billable amount in the most recent month that is $640.85 (to 2 d.p.) higher [i.e., ]; and customers that are married will have a billable amount in the most recent month that is $1,281.70 (to 2 d.p.) higher [i.e., ]

The code to derive the regression equation is as follows:

*#------Q5-----  
  
#this is to obtain the coeffecients in the regression model*regression\_coeff = lm.coef\_

*#this is to obtain the y-intercept in the regression model*regression\_intercept = lm.intercept\_  
*#the regression output is displayed in a formatted string for the sake of readability*print(f'B1 = {regression\_coeff[0]}(LIMIT) + {regression\_coeff[1]}(BALANCE) + {regression\_coeff[2]}(MARITAL) + {regression\_intercept}')

## References

Downey, A. (2016). Think Python: How to Think Like a Computer Scientist (2nd ed.). O’Reilly Media.

Frost, J. (n.d.-a). *Choosing the Correct Type of Regression Analysis*. Statistics By Jim. Retrieved August 28, 2022, from https://statisticsbyjim.com/regression/choosing-regression-analysis/

Frost, J. (n.d.-b). *How to Interpret P-values and Coefficients in Regression Analysis*. Statistics By Jim. Retrieved August 28, 2022, from https://statisticsbyjim.com/regression/interpret-coefficients-p-values-regression/

Frost, J. (n.d.-c). *Regression Tutorial with Analysis Examples*. Statistics By Jim. Retrieved August 28, 2022, from https://statisticsbyjim.com/regression/regression-tutorial-analysis-examples/

Matthes, E. (2015). Python Crash Course: A Hands-On, Project-Based Introduction to Programming (1st ed.). No Starch Press.

McKinney, W. (2017). Python for Data Analysis. Van Duuren Media.

R. Severance, C., PhD. (2016). Python for Everybody. https://do1.dr-chuck.com/pythonlearn/EN\_us/pythonlearn.pdf

scikit-learn. (n.d.-a). *1.1. Linear Models*. Retrieved August 28, 2022, from https://scikit-learn.org/stable/modules/linear\_model.html

scikit-learn. (n.d.-b). *3.1. Cross-validation: evaluating estimator performance*. Retrieved August 28, 2022, from https://scikit-learn.org/stable/modules/cross\_validation.html#cross-validation

scikit-learn. (n.d.-c). *Linear Regression Example*. Retrieved August 28, 2022, from https://scikit-learn.org/stable/auto\_examples/linear\_model/plot\_ols.html#sphx-glr-auto-examples-linear-model-plot-ols-py

scikit-learn. (n.d.-d). *sklearn.linear\_model.LinearRegression*. Retrieved August 28, 2022, from https://scikit-learn.org/stable/modules/generated/sklearn.linear\_model.LinearRegression.html

scikit-learn. (n.d.-e). *sklearn.model\_selection.train\_test\_split*. Retrieved August 28, 2022, from https://scikit-learn.org/stable/modules/generated/sklearn.model\_selection.train\_test\_split.html

statsmodels. (n.d.). *statsmodels.regression.linear\_model.OLS — statsmodels*. Retrieved August 28, 2022, from https://www.statsmodels.org/dev/generated/statsmodels.regression.linear\_model.OLS.html

van Dorpe, S. (2018, December 14). *Preprocessing with sklearn: a complete and comprehensive guide*. Towards Data Science. Retrieved August 28, 2022, from https://towardsdatascience.com/preprocessing-with-sklearn-a-complete-and-comprehensive-guide-670cb98fcfb9

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

Wu, S. (2020, May 23). *3 Best metrics to evaluate Regression Model?* Towards Data Science. Retrieved August 28, 2022, from https://towardsdatascience.com/what-are-the-best-metrics-to-evaluate-your-regression-model-418ca481755b

Z. (2022, May 20). *What is a Long Tail Distribution? (Definition & Example)*. Statology. Retrieved August 28, 2022, from https://www.statology.org/long-tail-distribution/