

**ANL252**

**Python for Data Analytics**

**End-Of-Course Assessment**

**July 2022 Presentation**

|  |  |
| --- | --- |
| **Name** | **PI Number** |
| Chloe Tan Ying Yin | W2082681 |

**Tutorial Group: TG09**

**Instructor’s Name: Professor KUMAR MUNISH**

**Submission Date: 5 SEP 2022**

“I declare that this assignment is my own work, unless otherwise acknowledged or credited by appropriate referencing. I have read and abided by the SUSS Honor Code and I am aware of the penalties associated with plagiarism and collusion listed in the SUSS Student Handbook.”

# Table of contents

# 

[**Table of contents**](#_tmjlk3b2e5hb) **1**

[**Question 1**](#_ntyi5vy60wz0) **2**

[**Question 2**](#_ae43zhqozp0h) **3**

[Overview of data that requires cleaning](#_z05e89g7491c) 3

[Duplicated rows](#_2ilp5kgr0uqe) 3

[Object type](#_mehvogzeq4iw) 3

[Illogical values](#_vorejjwgvnah) 3

[Overview of NaN affected columns](#_x2o8quq0s1pl) 4

[EDUCATION Column : 13 Affected rows](#_l5blfo6d2uxq) 5

[MARITAL Column : 38 Affected rows](#_u9hq1prziczp) 6

[Cleaning dataset](#_lduqaxkx2l60) 6

[**Pre-Processing Task 1a: Dropping of duplicated ID rows**](#_29aw1ueg17yr) **6**

[**Pre-Processing Task 1b: Dropping of rows**](#_yhke82t5y3ne) **6**

[**Pre-Processing Task 2a: Replacing AGE=199 with randomly generated Values (with normal distribution)**](#_z1vsvj6sttcy) **6**

[**Pre-Processing Task 2b: Replacing NaN with modal/random values**](#_hgy23is43f3) **7**

[Pre-Requisite: Combine Cleaned dataframes together](#_o4ws978wnoit) 8

[**Pre-Processing Task 3: Stripping special characters from R3 Column**](#_mob6euilvwly) **8**

[**Pre-Processing Task 4: Discretisation**](#_87m2ueneq0sw) **8**

[Verification of cleaned dataset](#_xrk41os34w22) 9

[**Question 3**](#_5345njrx2sjv) **10**

[Chart 1](#_iniyys1puoss) 10

[Chart 2](#_1t4cs4okmuuj) 13

[Chart 3](#_2ct7re6wrt0p) 16

[Chart 4](#_dbgpwl2kp341) 19

[Chart 5](#_w2ecdeg573gb) 21

[**Question 4**](#_lpgz21eg12bg) **23**

[Conclusion of correlation matrix](#_ql1339t0n1bk) 24

[**Question 5**](#_f6dwuknny51l) **28**

[Regression Equation:](#_v1xtg8esnlmu) 28

[Key Insights:](#_7duqxj16dp1s) 29

[**References**](#_4yr2o8q1opqx) **30**

# Question 1

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

# Question 2

**\*text in blue are comments**

**\*\* text without formatting in black are code**

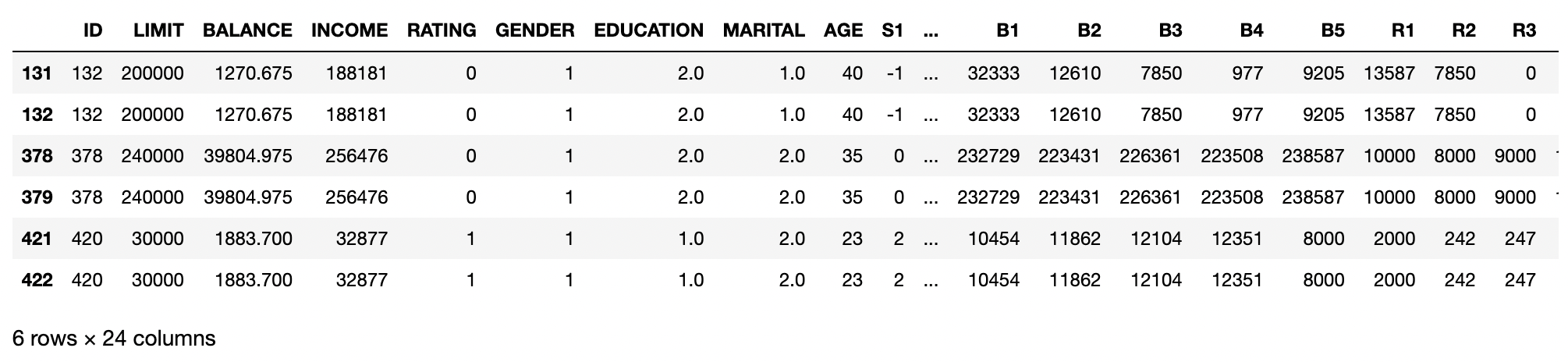
## 

## Overview of data that requires cleaning

### Duplicated rows

Checking for Duplicated Rows based on 'ID' that needs to be removed

DF[DF.duplicated(subset=['ID'],keep=False)]



### Object type

R3 Column is in object type and has characters like '$' and ',' that needs to be removed to convert column into numerical data type

#Column "R3" has additional '$' signs in value that needs to be stripped away

np.sort(DF["R3"].unique())

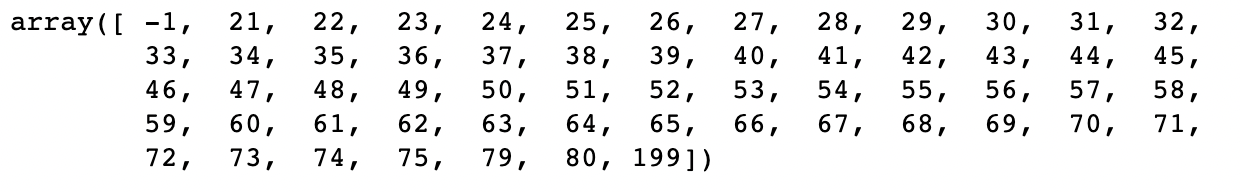


### Illogical values

Overview of AGE Column has illogical values like '-1' and '199' that needs to be either replaced/removed

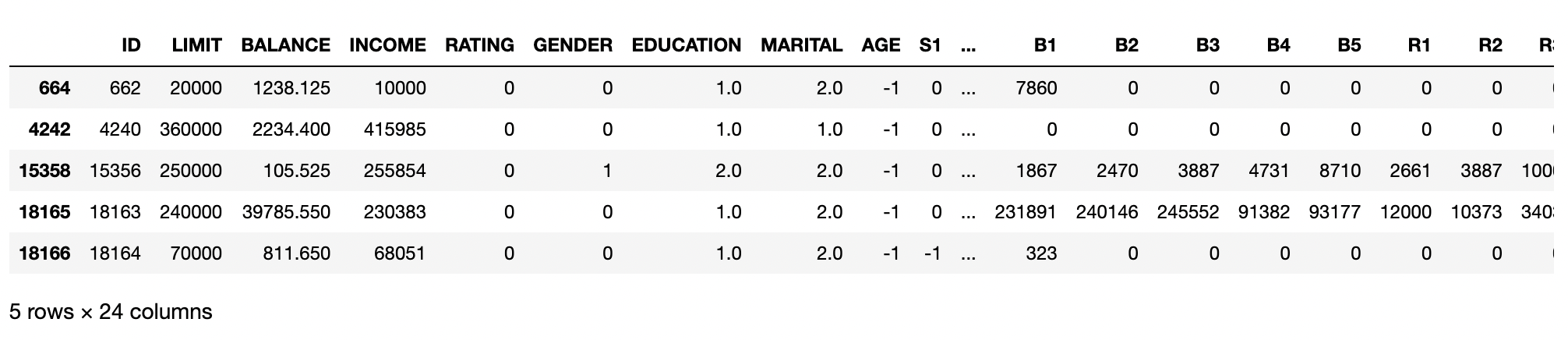
# Get an overview of unique values in column (Notice: -1 & 199)

np.sort(DF["AGE"].unique())



AGE=-1 : Mostly customers that has cleared their payment within the first bill

DF[DF["AGE"].isin([-1])]



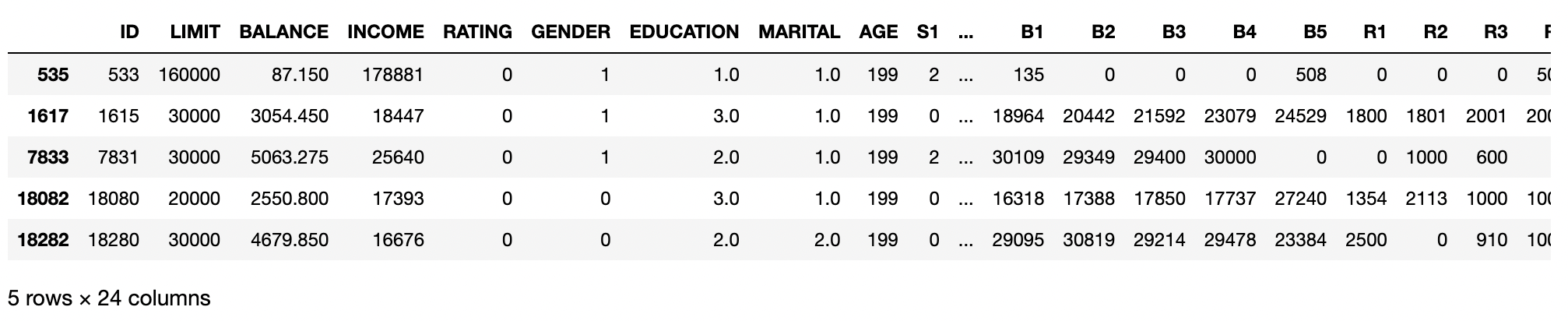
AGE=199 : Has more clients that has more than 1 payment

# AGE = 199

# Will use Random Normal distribution of age based on "RATING=0" and "MARITAL=1" criteria

# Require: mean & Std dev.

DF[DF["AGE"].isin([199])]

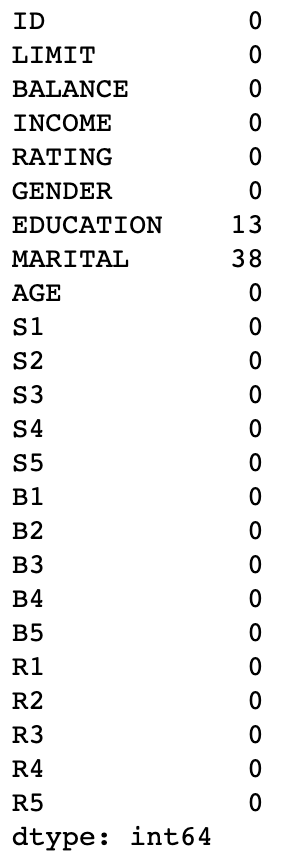


### Overview of NaN affected columns

Columns “Education” and “Marital” have 12 and 38 rows respectively containing NaNs

#NaN Overview

DF.isna().sum()



#### EDUCATION Column : 13 Affected rows

# 13 Affected rows

# Will replace with Modal value

DF[DF["EDUCATION"].isna()]



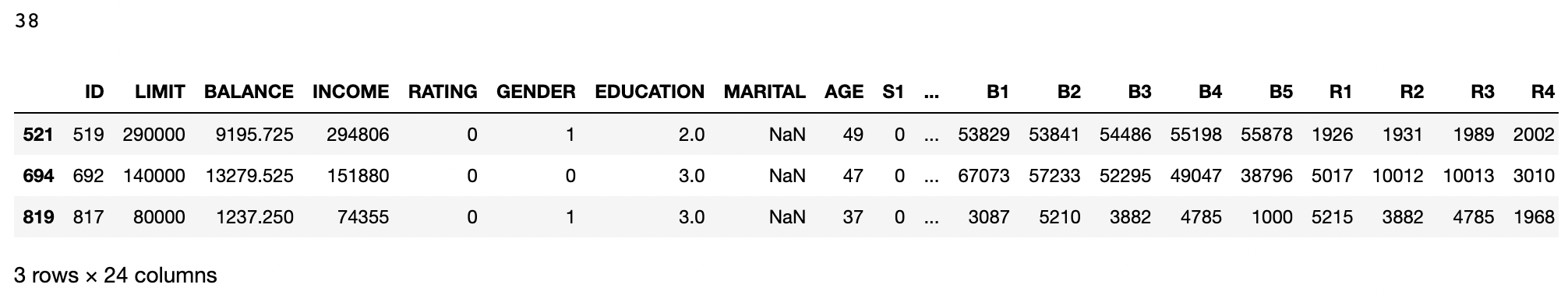
#### MARITAL Column : 38 Affected rows

# 38 Affected rows

# Generate Random Normally distributed values

print(len(DF[DF["MARITAL"].isna()]))

DF[DF["MARITAL"].isna()].iloc[:3,:]



## Cleaning dataset

### Pre-Processing Task 1a: Dropping of duplicated ID rows

# Drops duplicated row and keep first occurence of the duplicate only

DF = DF.drop\_duplicates(subset=["ID"],keep='first')

### Pre-Processing Task 1b: Dropping of rows

Removes rows with AGE = -1

# drop these rows completely

DF = DF[~DF["AGE"].isin([-1])]

### Pre-Processing Task 2a: Replacing AGE=199 with randomly generated Values (with normal distribution)

Create a reference data set to get mean and standard deviations as required

Split data affected by AGE=199 for easier value replacements later

# create clean data to generate required random data

# Dataframe without NaN

clean\_DF = DF.copy()

clean\_DF.dropna(axis=0, how="any", inplace=True)

# Drops rows with 2 both EDUCATION and MARITAL having NaN

# As replacing both values might introduce more variability in results

# Total of 4 rows dropped here

DF = DF.dropna(subset=["EDUCATION", "MARITAL"], how='all')

# Dataframe without illogical AGE values

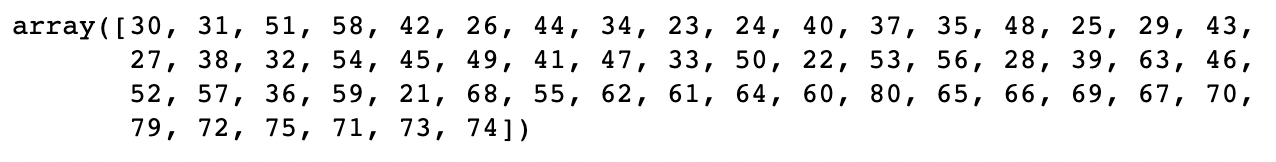
clean\_DF = clean\_DF[~clean\_DF["AGE"].isin([199])]

# Dataframe with illogical age values to be replaced

DF\_AGE\_199 = DF[DF["AGE"]==199]

# Displays ages in clean\_DF (for reference)

clean\_DF["AGE"].unique()



Generate Random Integers with normal distribution based on criteria of RATING=0 and MARITAL=1 from reference data

# Narrow down DF to get variables needed to generate random age with normal distribution

temp\_df = clean\_DF[(clean\_DF["RATING"]==0) & (clean\_DF["MARITAL"]==1)]

# Generate Normally distributed age and replace it column values

age\_list = np.random.normal( loc = int(temp\_df["AGE"].median()) , scale = round(temp\_df["AGE"].std(),2) , size= len(DF\_AGE\_199))

age\_list = list(map(int, age\_list))

# Replace values affected with randomly generated Integers with normal distributions

DF\_AGE\_199["AGE"] = age\_list # This will throw a warning, can ignore it.

### Pre-Processing Task 2b: Replacing NaN with modal/random values

# Handle EDUCATION NaN with Modal values

NAN\_EDU\_DF = DF[DF["EDUCATION"].isna()] # Create a dataframe only for EDUCATION with NaN

EDU\_LIST = [DF["EDUCATION"].mode().item()] \* len(NAN\_EDU\_DF) # Create a list of values based on education Modal value

# Handle MARITAL NaN with randomly generated Integers

NAN\_MAR\_DF = DF[DF["MARITAL"].isna()] # Create a dataframe only for MARITAL with NaN

marital\_list = np.random.randint(3, size=len(NAN\_MAR\_DF)) # Generates list of random integers from 0 - 2

# These will throw a warning, can ignore it.

NAN\_MAR\_DF["MARITAL"] = marital\_list # Replace NaN with generated values

NAN\_EDU\_DF["EDUCATION"] = EDU\_LIST # Replace NaN with modal values

#### Pre-Requisite: Combine Cleaned dataframes together

# ensure all column types are the same before combination

NAN\_MAR\_DF["MARITAL"] = NAN\_MAR\_DF["MARITAL"].astype('float64')

# combine data segments

frames = [ clean\_DF, DF\_AGE\_199, NAN\_MAR\_DF, NAN\_EDU\_DF]

DF\_CLEAN = pd.concat(frames)

### 

### Pre-Processing Task 3: Stripping special characters from R3 Column

# Removes '$' and ',' and change column type into integer 64

DF\_CLEAN[ "R3" ] = DF\_CLEAN[ "R3" ].apply( lambda x: x.strip('$,') )

DF\_CLEAN[ "R3" ] = DF\_CLEAN[ "R3" ].apply( lambda x: x.replace(",","") )

### Pre-Processing Task 4: Discretisation

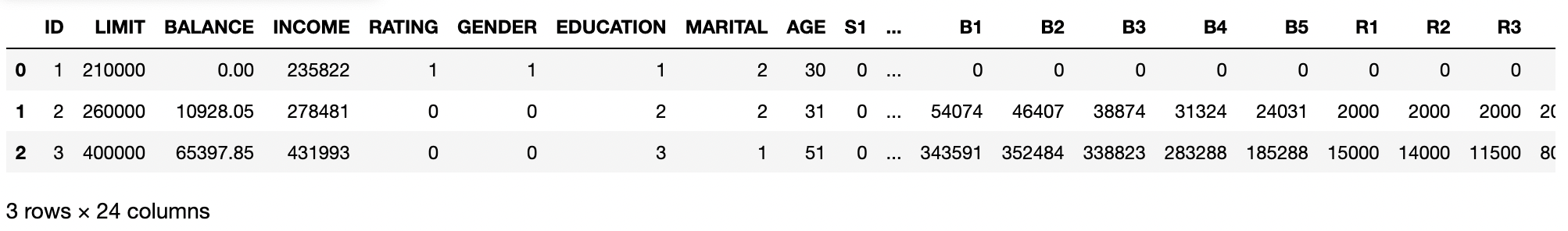
Converts EDUCATION, MARITAL & R3 into discrete column types

# DISCRETISATION of column EDUCATION, MARITAL & R3.

discretisation\_list = ["EDUCATION","MARITAL","R3"]

DF\_CLEAN[ discretisation\_list ] = DF\_CLEAN[ discretisation\_list ].astype('int64')

DF\_CLEAN.head(3)

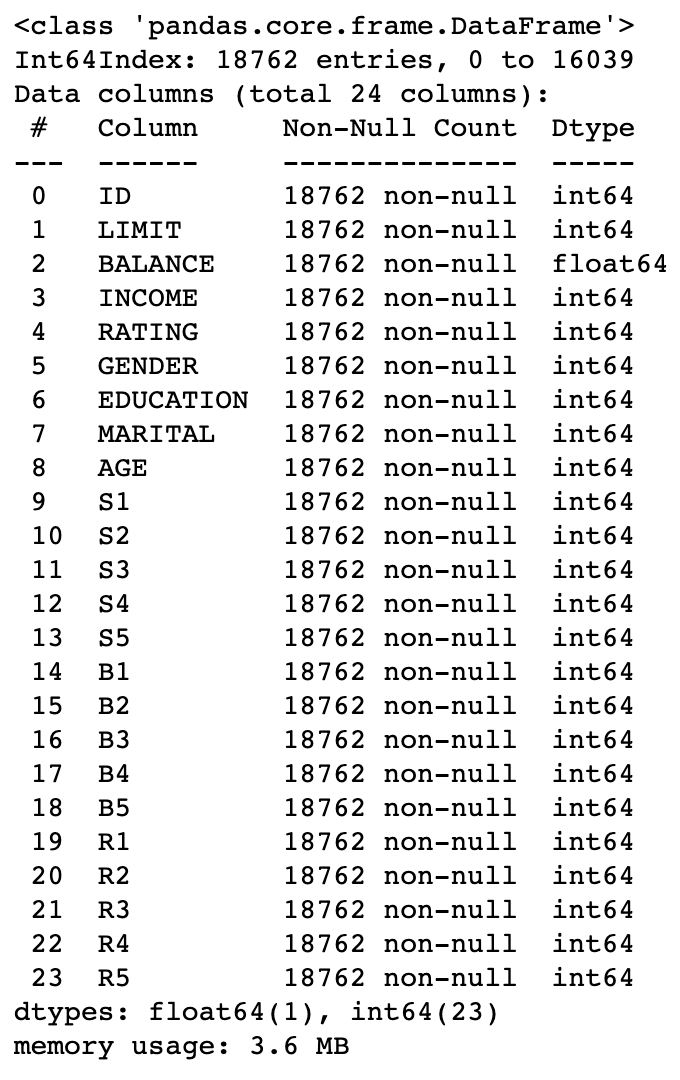
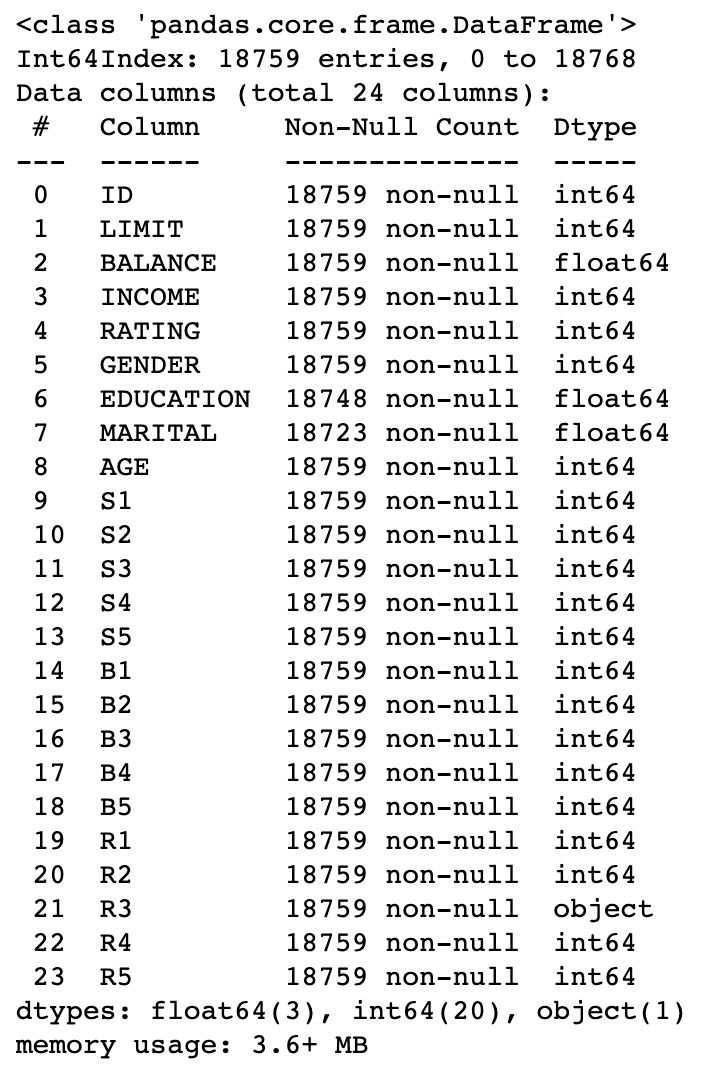


### Verification of cleaned dataset

Education and martial columns are converted from float to integer. R3 is converted from object to integer as well.

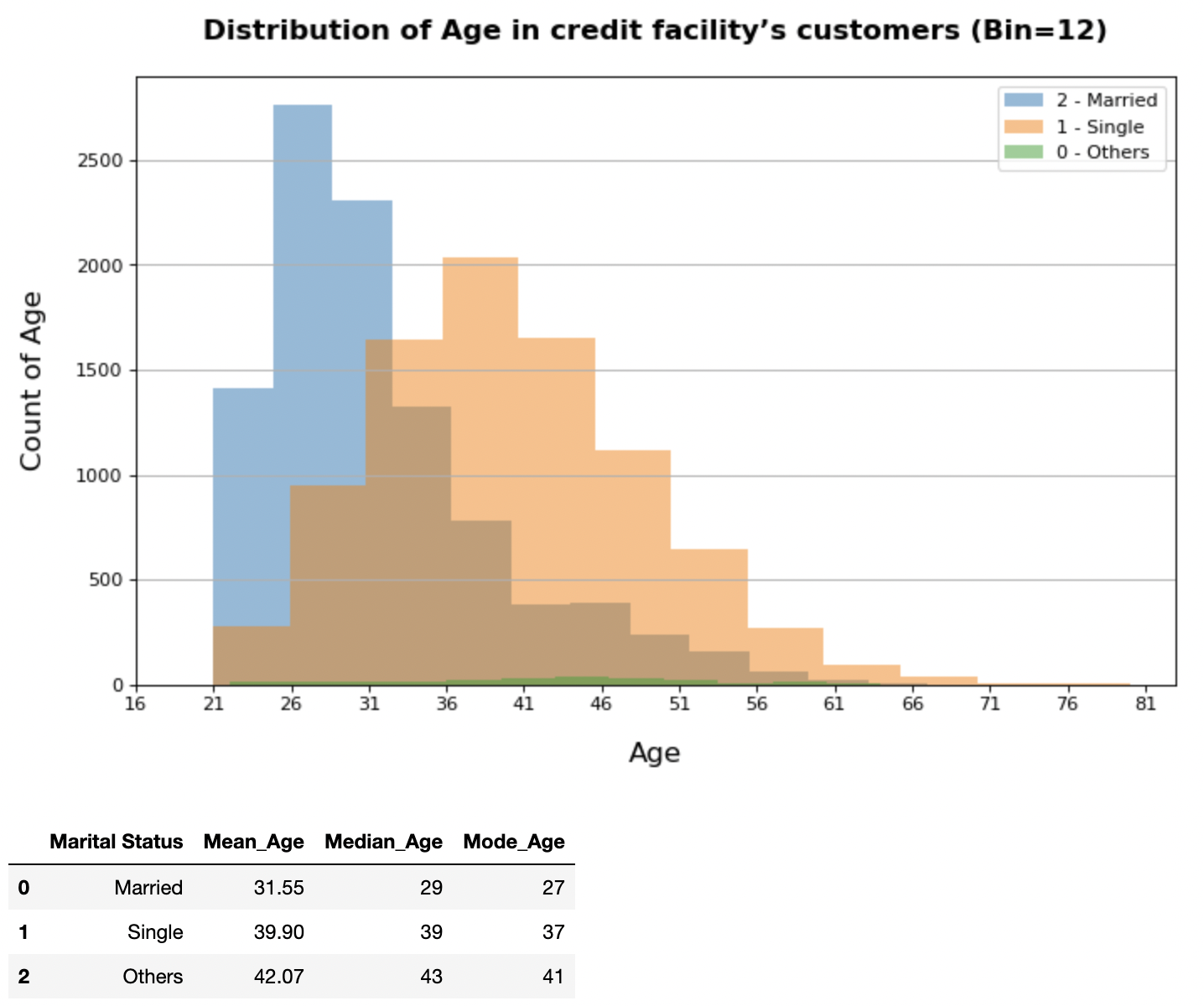
DF\_CLEAN.info()

OLD DF: CLEAN DF:



# Question 3

## Chart 1



*Figure 9: [Chart 1] Distribution of Age in credit facility’s customers*

**Insight 1 : There is a similar number of individuals that are married and single, however the age range of single individuals is higher than married individuals, at 21 - 36 and 41 - 51 respectively.**

Firstly, we will analyze the age range of customers in the credit facility. The dual axis chart showcases the age distribution of customers in the credit facility by marital status. The age range of married individuals normally falls around 21 - 36 while the age range of single individuals is higher, at 31 - 46. The number of individuals with the marital status one none is very low , with the age range being between 41 - 51. The average ages of others are the highest, followed by single and lastly marital ( 32, 40, 42 ).

Code:

# define plot settings

BINS = 12

COL = "AGE"

TITLE = "Distribution of Age in credit facility’s customers (Bin=12)"

Y\_LABEL = "Count of Age"

X\_LABEL = "Age"

legend\_dict = {

'0':"0 - Others",

'1':"1 - Single",

'2':"2 - Married"

}

data\_dict = {

"Marital Status": ["Married","Single","Others"],

"Mean\_Age":[],

"Median\_Age":[],

"Mode\_Age":[]

}

# Data splits

df\_interested = DF\_CLEAN.copy()

# create figure and plot histograms

plt.figure(figsize=(10, 6), dpi=80)

for marital\_stat in [2,1,0]:

df\_tmp = DF\_CLEAN[DF\_CLEAN["MARITAL"]==marital\_stat]

DAT = df\_tmp[COL]

plt.hist(DAT,alpha=0.5,bins = BINS,label=legend\_dict[str(marital\_stat)])

data\_dict["Mean\_Age"] = data\_dict["Mean\_Age"] + [round(df\_tmp[COL].mean(),2)]

data\_dict["Median\_Age"] = data\_dict["Median\_Age"] + [int(df\_tmp[COL].median())]

data\_dict["Mode\_Age"] = data\_dict["Mode\_Age"] + [df\_tmp[COL].mode().item()]

# Plots settings

plt.title(label=TITLE,pad=20,fontsize=15,weight='bold')

plt.xlabel(X\_LABEL,labelpad=15,fontsize=15)

plt.ylabel(Y\_LABEL,labelpad=15,fontsize=15)

plt.xticks([i for i in range(int(DF\_CLEAN[COL].min())-5,int(DF\_CLEAN[COL].max()+5),5)])

plt.legend()

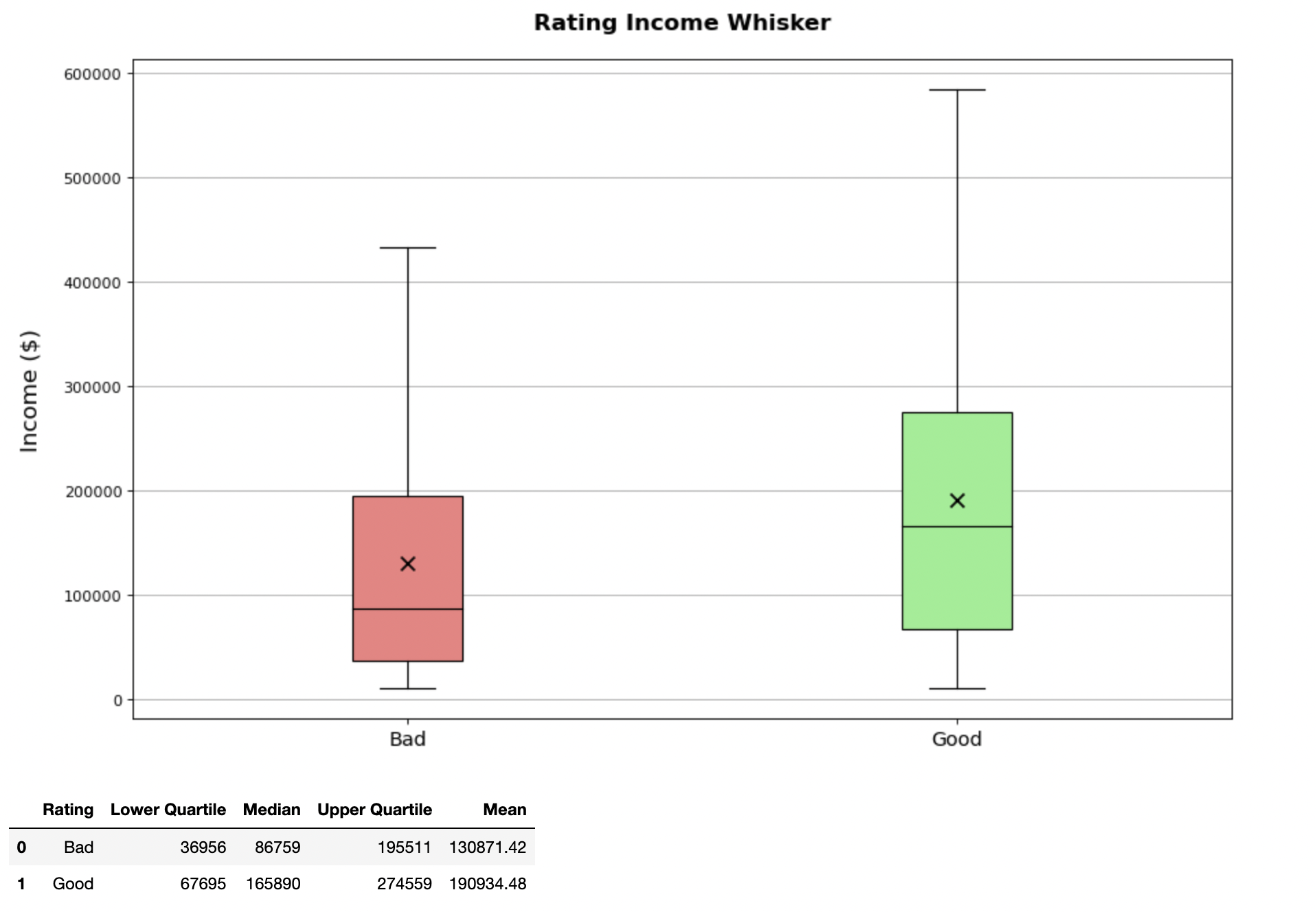
plt.grid(True,which="Major",axis='y')

plt.show()

# Display Data Table

pd.DataFrame(data\_dict)

## Chart 2

****

**Insight 2 : Customers with a good rating have higher incomes**

The box plot depicts the customer’s income ranges based on their ratings, with outliers removed. Customers with a good rating have a larger dispersion in income range, with an interquartile range from $67,685 - $165,890. Customers with a good rating ($$190,934) also have a higher average income as compared to customers with a bad rating ($130,871).

Customers with a bad rating have a positive skew, which means the customers with higher incomes are more varied. Overall, customers with a good rating have a higher income as compared to customers with a bad rating.

This could mean that having a higher income and financial stability gives the customer the ability to maintain a good rating (prompt payments).

Code:

from numpy import percentile

# define data

COL = "INCOME"

TITLE = "Rating Income Whisker"

Y\_LABEL = "Income ($)"

# Data splits

df\_interested = DF\_CLEAN.copy()

# initialise vairables

data = []

mean\_dat = []

legend\_dict = {

"0":"Good",

"1":"Bad"

}

df\_dict = {

"Rating":[],

"Lower Quartile":[],

"Median":[],

"Upper Quartile":[],

"Mean":[]

}

for rate in DF\_CLEAN["RATING"].unique():

df\_tmp = DF\_CLEAN[DF\_CLEAN["RATING"]==rate]

data.append(df\_tmp[COL])

mean\_dat.append(round(df\_tmp[COL].mean(),2))

df\_dict["Rating"] = df\_dict["Rating"] + [legend\_dict[str(rate)]]

# Calculation of data tables (mean,upper/lower quartile, median)

quartiles = percentile(df\_tmp[COL], [25, 50, 75])

df\_dict["Upper Quartile"] = df\_dict["Upper Quartile"] + [int(quartiles[2])]

df\_dict["Median"] = df\_dict["Median"] + [int(df\_tmp[COL].median())]

df\_dict["Lower Quartile"] = df\_dict["Lower Quartile"] + [int(quartiles[0])]

df\_dict["Mean"] = df\_dict["Mean"] + [round(df\_tmp[COL].mean(),2)]

# data plottings

fig\_sz = (10,6)

ax = plt.figure(figsize=fig\_sz, dpi=80).add\_axes([0, 0, 1, 1])

bp = ax.boxplot(data,patch\_artist=True,whis=1.5,showfliers=False,widths=0.2)

ax.scatter( x = [1,2], y = mean\_dat,

color = 'black',zorder=3,marker='x',s=80)

# Boxplot settings

for i,T in enumerate(DF\_CLEAN["RATING"].unique()):

if i == 0: bp["boxes"][i].set\_facecolor("lightcoral")

else: bp["boxes"][i].set\_facecolor("lightgreen")

bp["medians"][i].set\_color('black')

# Chart settings

plt.title(label=TITLE,pad=20,fontsize=15,weight='bold')

plt.ylabel(Y\_LABEL,labelpad=15,fontsize=15)

plt.xticks([1,2],labels=df\_dict["Rating"],fontsize=13)

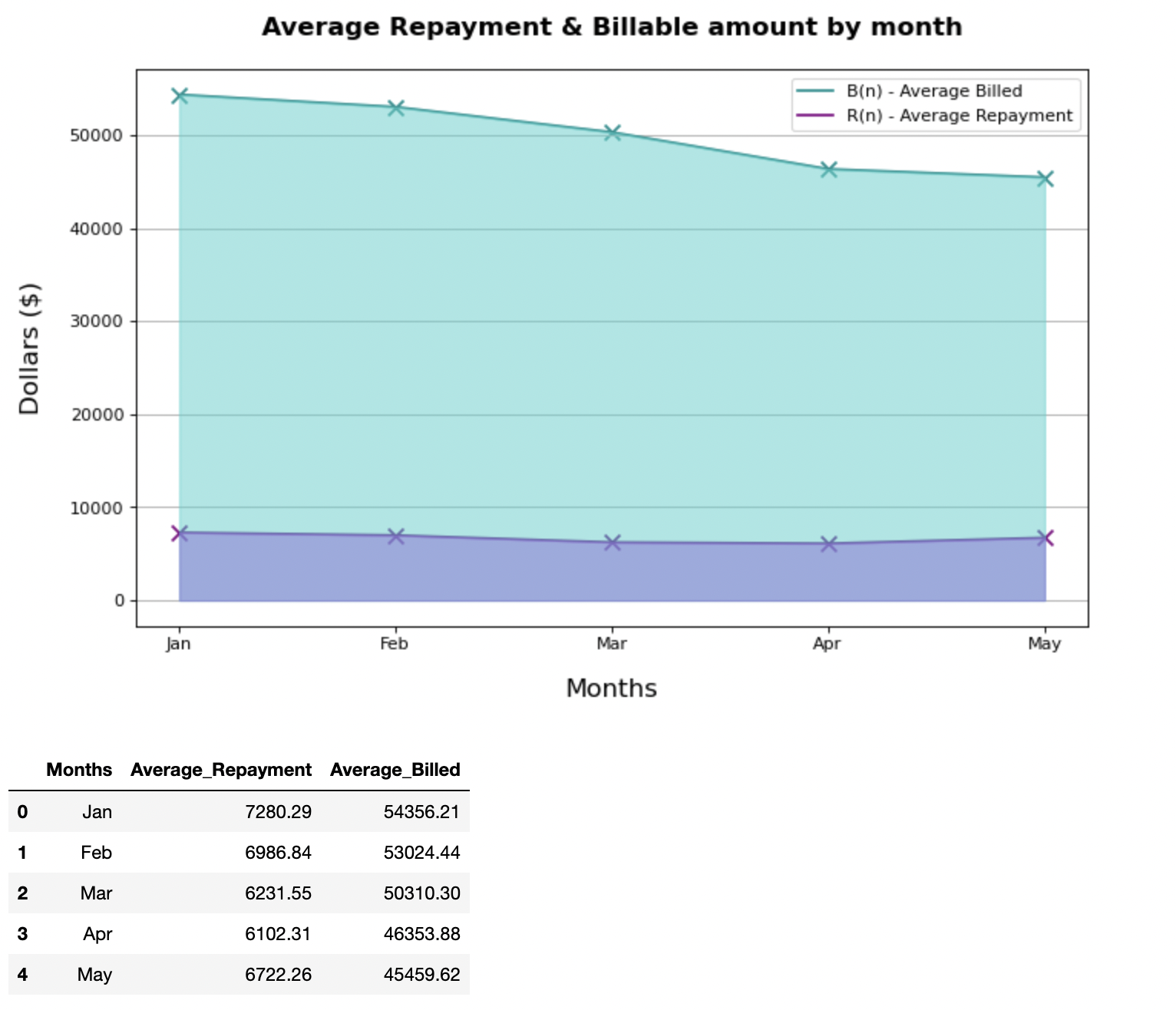
plt.grid(True,which="Major",axis='y')

plt.show()

# # make into display data table

pd.DataFrame(df\_dict)

## Chart 3

****

**Insight 3 : The average billed amount tends to be higher towards the start of the year.**

From January to May, the average billed amount decreases, ($54,356 - $45,459). This could be due to the festive seasons at the start of the year, such as Chinese New year in February as well as spring clearing and buying new clothes / furniture in January. As the year progresses, individuals tend to be more busy with work and school, thus spending less. As for repayment amounts, it is fairly consistent from January to May, but a small decrease can be observed ($7280 - $6722).

Code:

TITLE = "Average Repayment & Billable amount by month"

Y\_LABEL = "Dollars ($)"

X\_LABEL = "Months"

DF\_CLEAN

r\_list = []

b\_list = []

mth\_list = ['Jan',"Feb","Mar","Apr","May"]

for i in range(1,6):

b\_list.append(round(DF\_CLEAN[~DF\_CLEAN['B'+str(i)].isin([-1,0])]['B'+str(i)].mean(),2))

r\_list.append(round(DF\_CLEAN[~DF\_CLEAN['R'+str(i)].isin([-1,0])]['R'+str(i)].mean(),2))

df\_dict = {

"Months":mth\_list,

"Average\_Repayment":r\_list,

"Average\_Billed":b\_list

}

# create figure

ax = plt.figure(figsize=(10,6), dpi=80)

# plot lines

plt1 = plt.plot(df\_dict["Months"], df\_dict["Average\_Billed"],zorder=3,color='darkcyan',label="B(n) - Average Billed")

plt.scatter(df\_dict["Months"],df\_dict["Average\_Billed"],marker='x',s=80,c="darkcyan")

plt2 = plt.plot(df\_dict["Months"], df\_dict["Average\_Repayment"],zorder=3,color='purple',label="R(n) - Average Repayment")

plt.scatter(df\_dict["Months"],df\_dict["Average\_Repayment"],marker='x',s=80,c="purple")

# shade area below lines

plt.fill\_between(df\_dict["Months"],df\_dict["Average\_Billed"],alpha=0.5,color='mediumturquoise',zorder=3)

plt.fill\_between(df\_dict["Months"],df\_dict["Average\_Repayment"],alpha=0.5,color='mediumpurple',zorder=3)

# plot display settings for bar plots

plt.ylabel(Y\_LABEL,labelpad=15,fontsize=15)

plt.xlabel(X\_LABEL,labelpad=15,fontsize=15)

plt.grid(True,which="Major",axis='y')

# Plot settings

plt.title(label=TITLE,pad=20,fontsize=15,weight='bold')

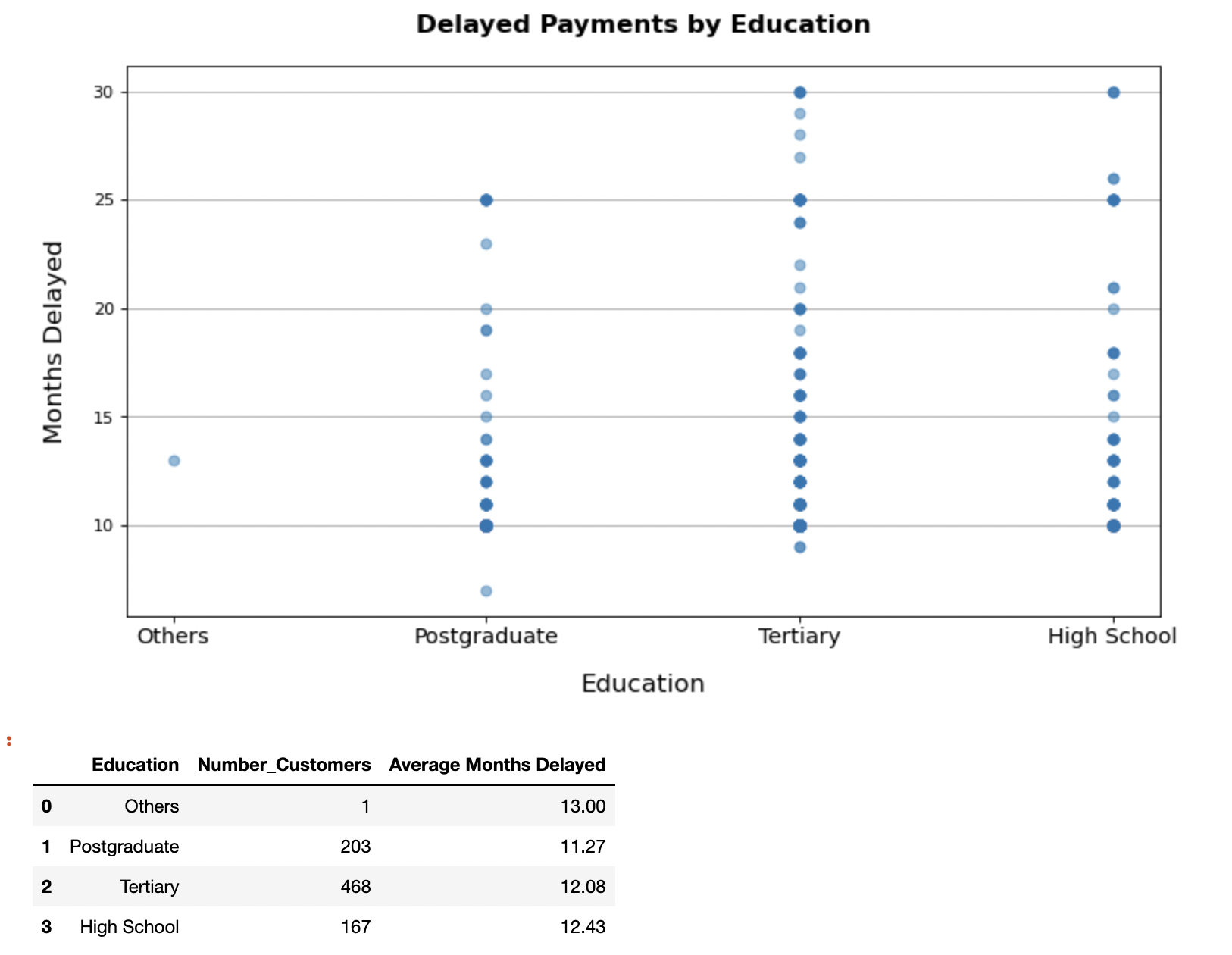
plt.legend()

plt.show()

pd.DataFrame(df\_dict)

## 

## Chart 4



**Insight 4: More customers with a tertiary education background tend to delay their payments for a long period of time.**

468 Customers with a tertiary education background had delayed payments. This is the largest number of customers as compared to customers with other education backgrounds. However, those that opted not to state their education background (others) had the highest average number of months of delayed payments. This could signal to the credit firm that individuals that do not wish to indicate their education level are not optimal customers.

Code:

# Define plot settings

TITLE = "Delayed Payments by Education"

Y\_LABEL = "Months Delayed"

X\_LABEL = "Education"

# prepare data for plotting

df\_interested = DF\_CLEAN[(DF\_CLEAN["S1"]>0)&(DF\_CLEAN["S2"]>0)&(DF\_CLEAN["S3"]>0)&(DF\_CLEAN["S4"]>0)&(DF\_CLEAN["S5"]>0)]

df\_interested["S\_Total"] = df\_interested["S1"] + df\_interested["S2"] + df\_interested["S3"] + df\_interested["S4"] + df\_interested["S5"] # This will throw a warning

# create and plot figure

plt.figure(figsize=(11, 6), dpi=80)

X = df\_interested["EDUCATION"]

Y = df\_interested["S\_Total"]

plt.scatter(X,Y,zorder=3,alpha=0.5)

# Plot settings

X\_TICKS = ["Others", "Postgraduate", "Tertiary", "High School"]

plt.xticks([0,1,2,3],labels=X\_TICKS,fontsize=13)

plt.title(label=TITLE,pad=20,fontsize=15,weight='bold')

plt.xlabel(X\_LABEL,labelpad=15,fontsize=15)

plt.ylabel(Y\_LABEL,labelpad=15,fontsize=15)

plt.grid(True,which="Major",axis='y')

plt.show()

tmp = []

tmp\_num = []

for edu in [0,1,2,3]:

tmp.append(round(df\_interested[df\_interested["EDUCATION"]==edu]["S\_Total"].mean(),2))

tmp\_num.append(len(df\_interested[df\_interested["EDUCATION"]==edu]))

df\_dict={

"Education": X\_TICKS,

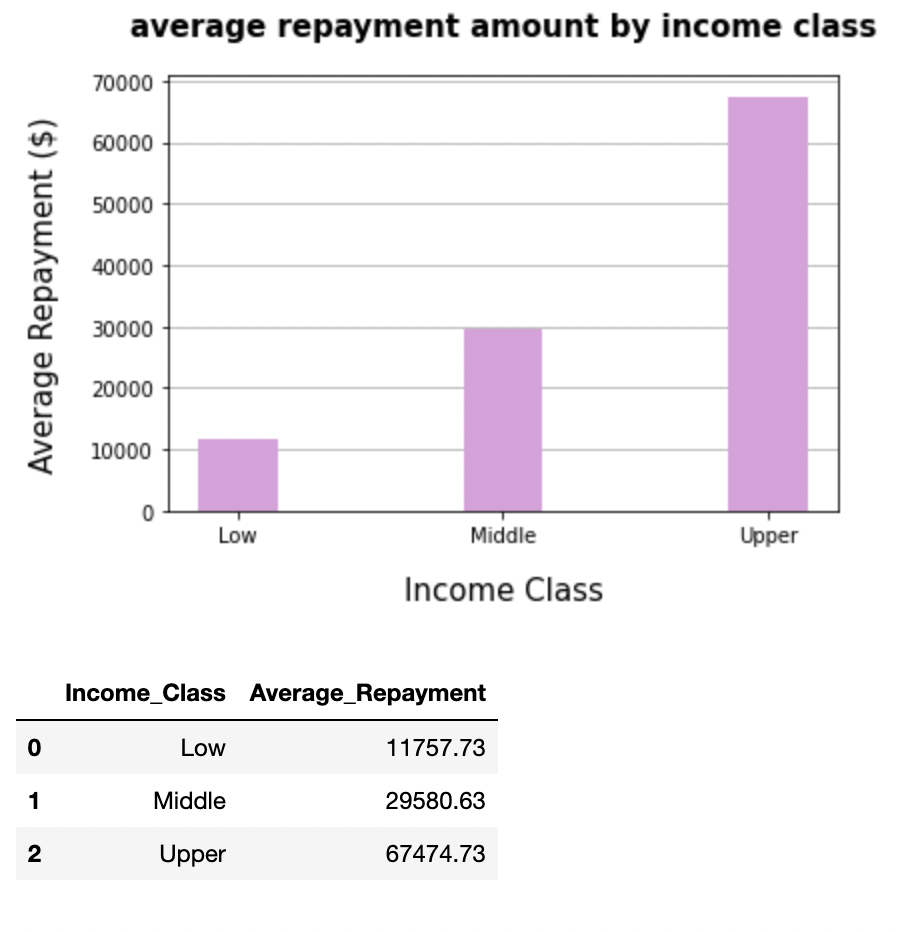
"Number\_Customers":tmp\_num,

"Average Months Delayed":tmp

}

pd.DataFrame(df\_dict)

## Chart 5

****

**Insight: customers within a higher income class have a higher dollar values of average repayments**

The higher the income class, the higher the sum of average repayments. The customers in the upper class have an average repayment of $67,474, while middle class has an average of $29,580 while lower class has $11,757.

Being of a higher income class allows the customer to accrue a higher sum of expenses, as well as being able to repay a higher sum as well. This means that upper income customers are more valuable to the credit facility.

Code:

# Define plot settings

TITLE = "average repayment amount by income class"

Y\_LABEL = "Average Repayment ($)"

X\_LABEL = "Income Class"

df\_dict = {

"Income\_Class": ["Low","Middle","Upper"],

"Average\_Repayment":[]

}

df\_low = DF\_CLEAN[DF\_CLEAN["INCOME"]<=50000]

df\_mid = DF\_CLEAN[(DF\_CLEAN["INCOME"]>50000)&(DF\_CLEAN["INCOME"]<=300000)]

df\_upp = DF\_CLEAN[DF\_CLEAN["INCOME"]>300000]

R\_M\_list = []

# Sums up all the mean of R(n)

for tmp\_df in [df\_low,df\_mid,df\_upp]:

r\_list = []

for i in range(1,6): r\_list.append(round(tmp\_df[~tmp\_df['R'+str(i)].isin([-1,0])]['R'+str(i)].mean(),2))

R\_M\_list.append(np.array(r\_list).sum())

df\_dict["Average\_Repayment"] = R\_M\_list

# plot bars

plt1 = plt.bar(df\_dict["Income\_Class"], R\_M\_list,width = 0.3,zorder=3,color='plum')

# plot display settings for bar plots

plt.ylabel(Y\_LABEL,labelpad=15,fontsize=15)

plt.xlabel(X\_LABEL,labelpad=15,fontsize=15)

plt.grid(True,which="Major",axis='y')

# Plot settings

plt.title(label=TITLE,pad=20,fontsize=15,weight='bold')

plt.show()

pd.DataFrame(df\_dict)

# Question 4

**Steps :**

1. **Use Correlation Matrix to narrow down useful columns (High correlation)**
2. **Perform linear regression for that column against B1**

As there are multiple possible columns to do linear regression with B1. The correlation matrix in the pandas class was used to get a quick overview of all the correlation coefficients calculated against each and every column available in the dataframe.

**Creating the Correlation Matrix**

*# Creates correlation matrix for between all column*

*cor\_mat = DF\_CLEAN.corr()*

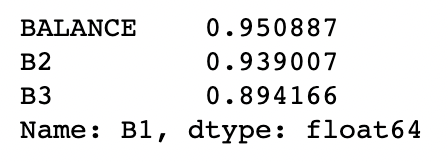
**Top 3 correlation variables (with B1)**

Next, the following was done to narrow down to the top 3 variables that have a high correlation with B1.

*# View Top 3 correlations for "B1" only and sort it.*

*# Exclude 1st row as B1 will always have 100% correlation with itself*

*cor\_mat.sort\_values(by=["B1"],ascending=False)["B1"].iloc[1:4,]*

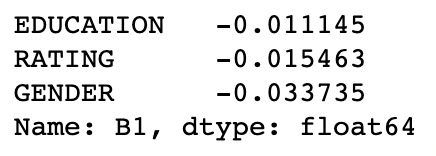


**Bottom 3 correlation variables (with B1)**

And for completeness, the lowest 3 correlation with B1 was viewed. These negative correlations are likely due to the fact that these variables are categorical.

*# least correlation*

*cor\_mat.sort\_values(by=["B1"],ascending=False)["B1"].iloc[-3:,]*

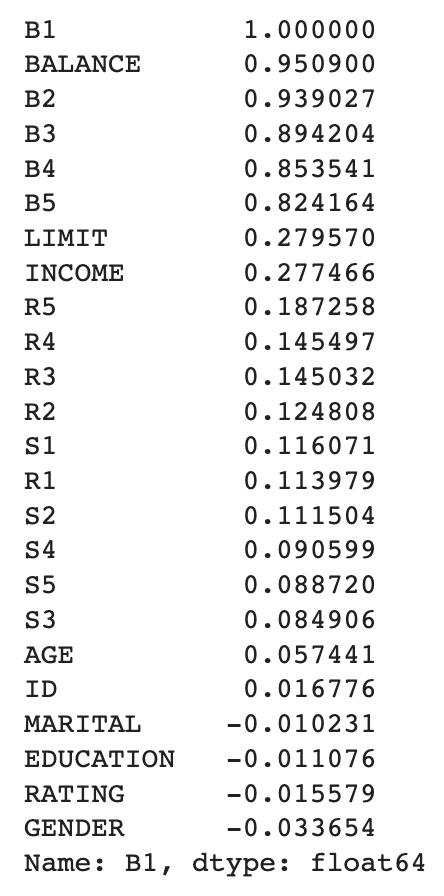


**Overall correlation ranking**

For reference this was the full ranking of correlation variables against B1.

*# Full Correlation rankings*

*cor\_mat.sort\_values(by=["B1"],ascending=False)["B1"].iloc[1:,]*

****

### Conclusion of correlation matrix

As B(n),S(n) and R(n) are variables based on B1 across a time period, those variables will not be considered. Hence, the ‘BALANCE’ variable will be used to perform linear regression against B1 as it has the highest correlation with B1.

Code:

from sklearn.linear\_model import LinearRegression

# Prepare data to be fed to the model

# X: BALANCE / LIMIT / INCOME - Recommended: Either of the Top 3 high correlation variables are good, Not logical to use Billable variables to predict 1st payment

# Y: B1

# Gets User Input to find with variable to regress B1 with

while True:

print("Available columns are:",list(DF\_CLEAN.columns))

x\_col = input("Please Enter Column to perform linear regression with B1: ") # Change desired column to do linear regression with B1 here

if x\_col not in DF\_CLEAN.columns: print("Please Enter value column")

else: break

# x\_col = "BALANCE" # Change desired column to do linear regression with B1 here

# Reads data into X and Y for model feeding

X = DF\_CLEAN[[x\_col]]

Y = DF\_CLEAN['B1']

# perform linear regression

linear\_regressor = LinearRegression() # Create linear regression model base

linear\_regressor.fit(X, Y) # Feed data into model

Y\_pred = linear\_regressor.predict(X) # predictions

# plot the model and its predictions against itself

plt.figure(figsize=[15, 7]) # Create blank canvas

plt.scatter(X, Y,zorder=3) # Plot raw data of desired column against B1

plt.plot(X, Y\_pred, color='black',zorder=3) # Plots predicted values from regression model (Black line)

# Chart Aesthetics (X/Y axis labels, grid lines settings, title)

plt.title('\n'+x\_col+" Vs B1",fontsize=15,weight='bold',pad=20)

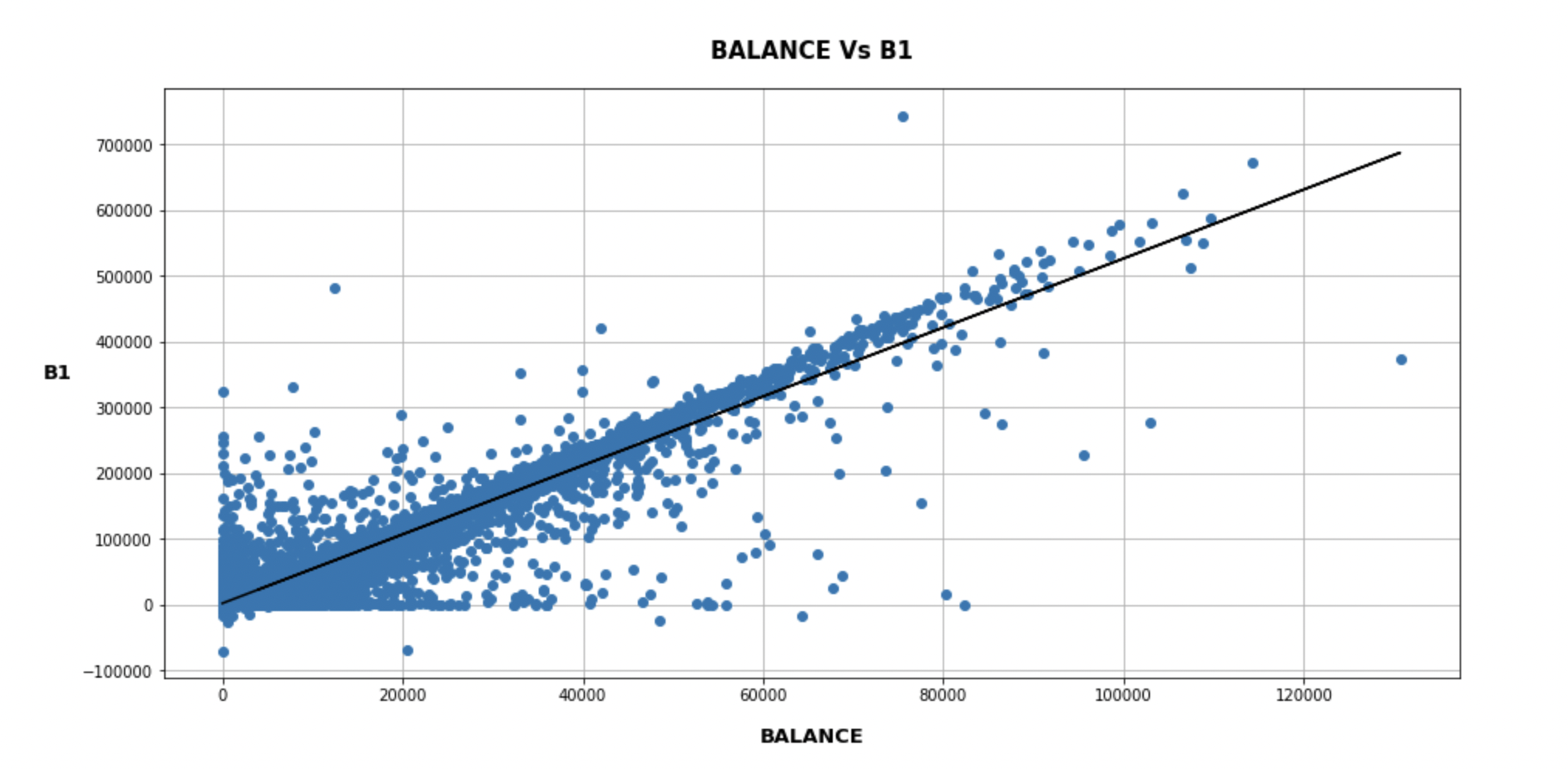
plt.xlabel(x\_col,fontsize=13,weight='bold',labelpad=15)

plt.ylabel("B1",fontsize=13,weight='bold',rotation=0,labelpad=15)

plt.grid(True)

plt.show()

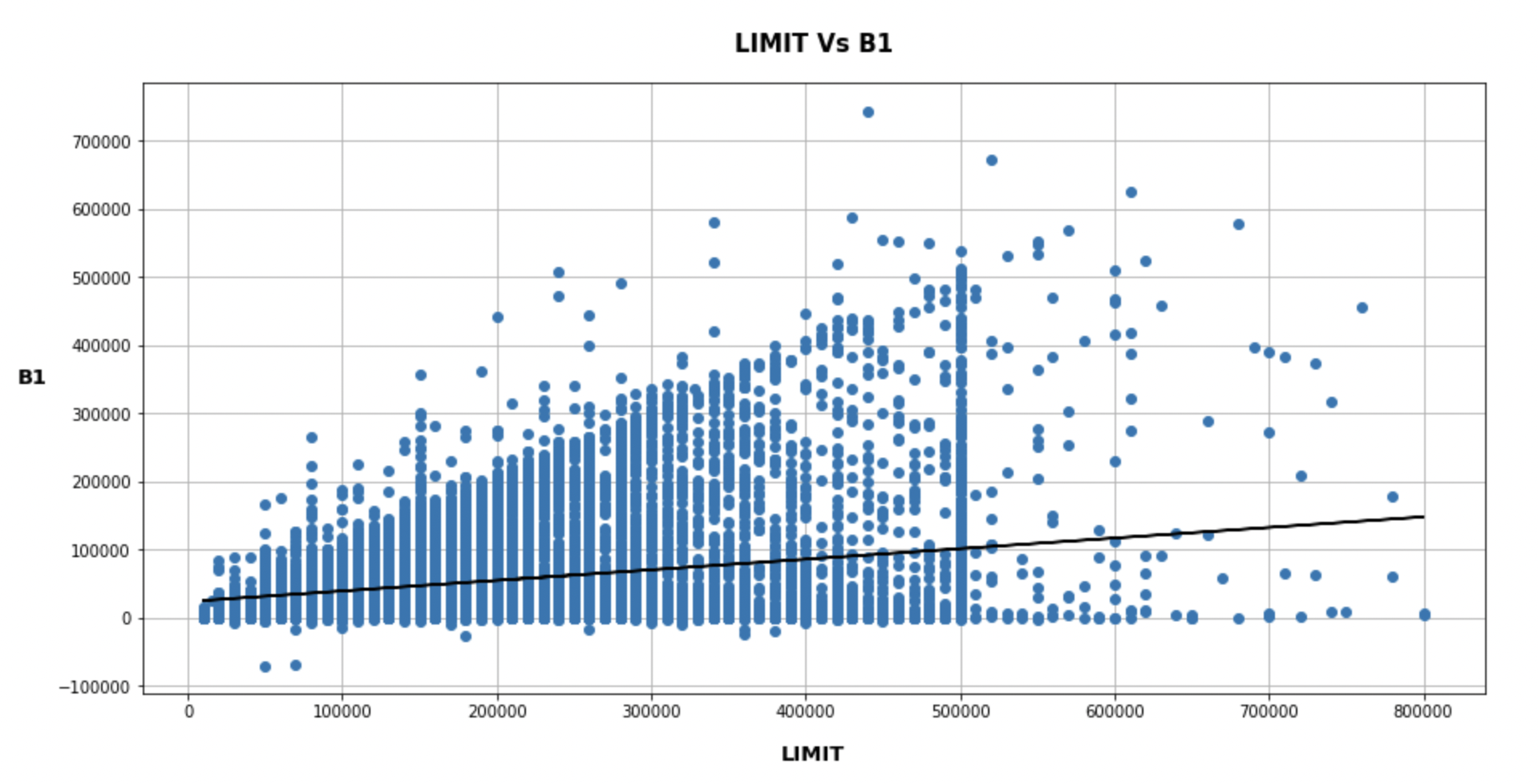
**Regression model for Balance (highest correlation, 0.950887)**

****

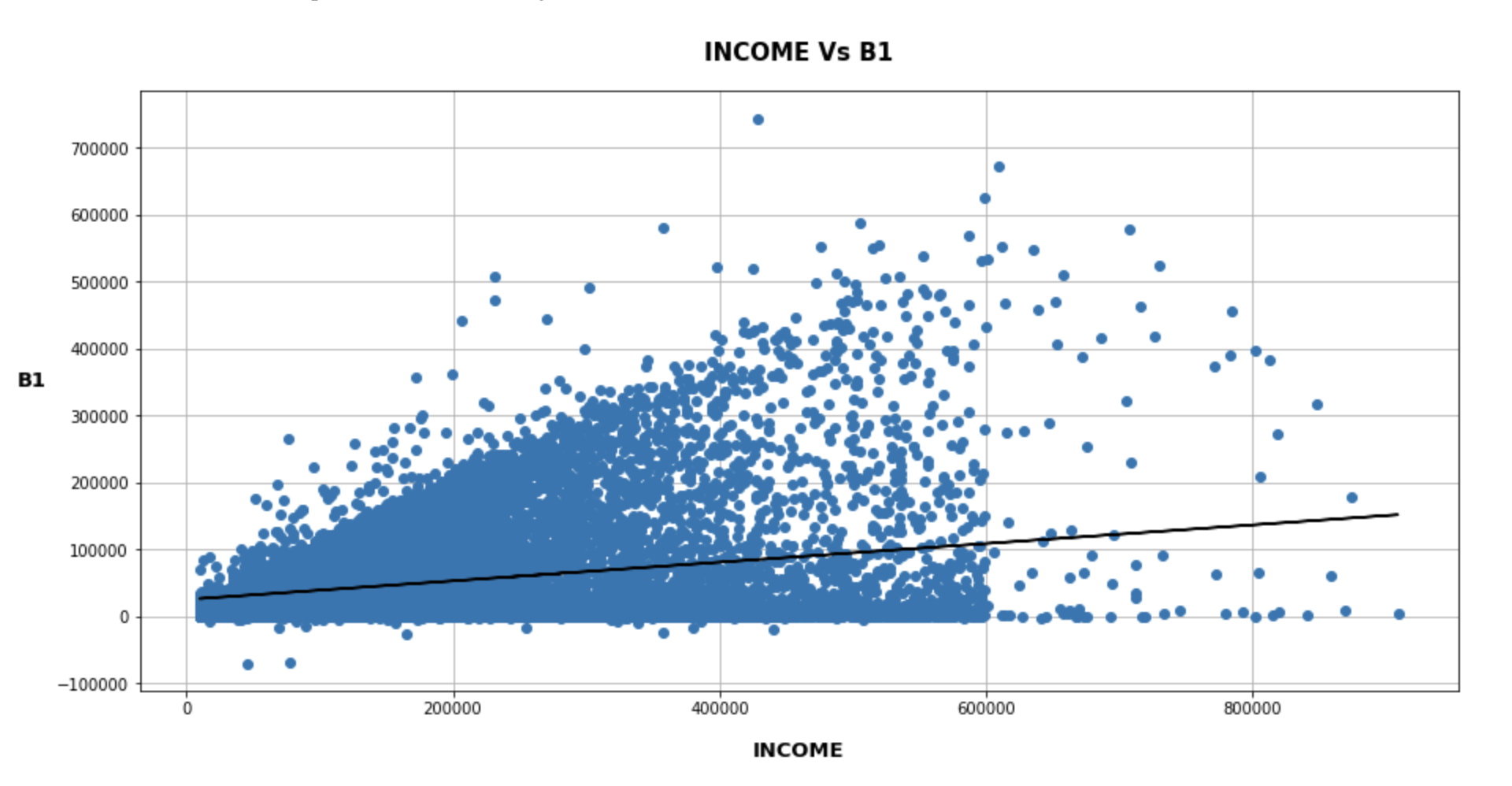
**Linear Regression Insights**

As most data points are close to the regression line and the Balance variable correlation coefficient is considered high, as it is greater than 0.75 (Mindrila & Balentyne, 2017). This shows that the dependent variable, B1 is strongly correlated to the independent variable, balance. Hence, allowing a relatively accurate prediction (~90%) based on this linear regression model.

**Regression model for Limit ( second highest correlation , 0.279529 )**

****

**Regression model for Income ( third highest correlation, 0.277423 )**



**Alternative linear regression insights**

Although Limit and income are the second and third most correlated independent variables to the dependent variable (B1) , their values are 0.279529 and 0.277423 respectively. A correlation coefficient between 0.25 and 0.5 is generally considered weak, and thus it cannot be said that limit and income are good determinants for B1. Hence, we will not be analysing them (Zach, 2021).

# Question 5

## Regression Equation:

The following are the (Accuracy) value and the linear regression equation itself.

With the strong correlation between the billable amount in the first month (Y) and the customer’s credit balance ( ) a high accuracy is achieved. The indicates that with the above stated linear regression amount a billable amount in the first month can be predicted with 90.4% accuracy.

Code:

# Gets Parameters from linear regression model and display it

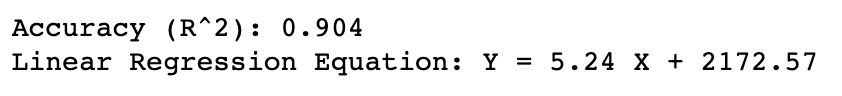
r\_sq = round(linear\_regressor.score(X, Y),3)

M = round(linear\_regressor.coef\_[0],2)

C = round(linear\_regressor.intercept\_,2)

print("Accuracy (R^2):", r\_sq)

print("Linear Regression Equation: Y =",M,"X +",C)



## 

## 

## Key Insights:

1. There are 3 variables that correlate with well 1st billable payment (**Balance, Limit & Income)**
   1. Balance

Customers with a higher bank balance tend to pay a higher amount in their first bill. Having more money in their bank account does indicate that they have a higher amount of money at their disposal. Hence with a greater spending ability they are able to incur a larger bill.

* 1. Limit

Customers with a higher total credit limit also tend to pay more in their first bill. Customers with a higher credit limit tend to have higher spending power, thus allowing them to accrue a larger bill.

* 1. Income

Customers that have a higher income pay more in their first bill. A higher income indicates that they are able to spend more per month, thus they tend to have higher expenditure which results in incurring a larger bill.

As balance has the strongest correlation, as calculated in the correlation matrix, it indicates that having a higher bank balance will be the strongest determining factor, as compared to limit and income, in predicting how high the customer’s first bill will be.

1. Conversely, gender, credit ratings and education have a weak correlation with the first billable payment due to the negative correlation value. This indicates that the first billable amount can't be determined well based on the customer's gender, ratings nor education.

# References

Mindrila, D., & Balentyne, P. (2017). *Scatterplots and Correlation*. <https://www.westga.edu/academics/research/vrc/assets/docs/scatterplots_and_correlation_notes.pdf>

Zach. (2021, April 27). *What is Considered to Be a “Weak” Correlation?* Statology. <https://www.statology.org/what-is-a-weak-correlation/#:~:text=As%20a%20rule%20of%20thumb,weak%E2%80%9D%20correlation%20between%20two%20variables>

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

‌