

**ANL252**

**Python for Data Analytics**

**End-of-Course Assessment**

**July 2022 Presentation**

**Submitted by:**

|  |  |
| --- | --- |
| **Name** | **PI No.** |
| **Muhammad Azfar Bin Razak** | **H2170546** |

**Tutorial Group: ­­­­­­­­­­­­ T 09**

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

**Submission Date: /2022**

Table of Contents

[Question 1 1](#_Toc113124947)

[Code 1 1](#_Toc113124948)

[Output 1: Dataframe output of ECA\_Data.csv 1](#_Toc113124949)

[Table 1: List of categorical and numeric variables 1](#_Toc113124950)

[Question 2 2](#_Toc113124951)

[1st Data Pre-Processing 2](#_Toc113124952)

[Code 2.1 2](#_Toc113124953)

[2nd Data Pre-Processing 3](#_Toc113124954)

[Code 2.2 3](#_Toc113124955)

[3rd Data Pre-Processing 4](#_Toc113124956)

[Code 2.3 4](#_Toc113124957)

[4th Data Pre-Processing 5](#_Toc113124958)

[Code 2.4 5](#_Toc113124959)

[Question 3 6](#_Toc113124960)

[Visualisation 1 – Count of Gender 6](#_Toc113124961)

[Figure 1: Percentage of Customers by Gender 6](#_Toc113124962)

[Table 2: Count of Customers by Gender 6](#_Toc113124963)

[Visualisation 2 – Average Income by Gender 8](#_Toc113124964)

[Figure 2: Average Income by Gender 8](#_Toc113124965)

[Table 3: Table of Average Income by Gender 8](#_Toc113124966)

[Code 3.2 9](#_Toc113124967)

[Visualisation 3 – Age Distribution 10](#_Toc113124968)

[Output 6: Cell 3.3.1 10](#_Toc113124969)

[Figure 3: Age Distribution By Age Group 10](#_Toc113124970)

[Table 4: Count of Age Group 10](#_Toc113124971)

[Code 3.3 11](#_Toc113124972)

[Visualisation 4 – Education Level by Gender 12](#_Toc113124973)

[Figure 4: Education of Customer by Gender 12](#_Toc113124974)

[Table 5: Count of Education Type by Gender 13](#_Toc113124975)

[Code 3.4 13](#_Toc113124976)

[Visualisation 5 – Rating by Gender 14](#_Toc113124977)

[Figure 5: Rating of Gender 14](#_Toc113124978)

[Table 6: Rating of Gender by Count 14](#_Toc113124979)

[Code 3.5 14](#_Toc113124980)

[Question 4 16](#_Toc113124981)

[Further Data Pre-Processing 16](#_Toc113124982)

[Code 4.1 16](#_Toc113124983)

[Output 7: Result of Dropped Categorical Columns 17](#_Toc113124984)

[Linear Regression Model Training 17](#_Toc113124985)

[Output 8: Result of X and Y shape 17](#_Toc113124986)

[Code 4.2 18](#_Toc113124987)

[Predicting B1 Variable of Index 5 18](#_Toc113124988)

[Output 9: DataFrame of Index 5 18](#_Toc113124989)

[Code 4.3 19](#_Toc113124990)

[Output 10: Predicted Value of Index 5 19](#_Toc113124991)

[Code 4.4 19](#_Toc113124992)

[Question 5 20](#_Toc113124993)

[Linear Regression Equation 20](#_Toc113124994)

[Code 5.1 20](#_Toc113124995)

[Mean Squared Error 21](#_Toc113124996)

[Code 5.2 21](#_Toc113124997)

[Visual Aid Actual vs Predicted Value 21](#_Toc113124998)

[Code 5.3 23](#_Toc113124999)

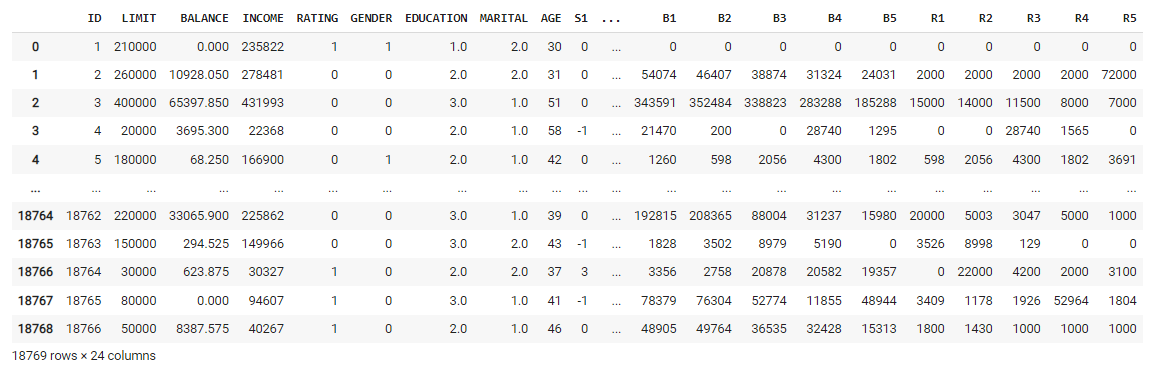
[References 24](#_Toc113125000)

# Question 1

Before presenting the DataFrame, I will first load the required libraries in order to locate the category and numerical variables in the dataset given by code 1.

I will present the categories and numerical variables from Output 1 in table style which is presented in table 1.

|  |
| --- |
| Code 1 |
| ## Importing the necessary packages  import numpy as np  import pandas as pd  import matplotlib.pyplot as plt  from datetime import date  # Reading csv file into df(Data Frame) and displaying the dataframe  df = pd.read\_csv("ECA\_Data.csv")  df |



Output 1: Dataframe output of ECA\_Data.csv

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

## Table 1: List of categorical and numeric variables

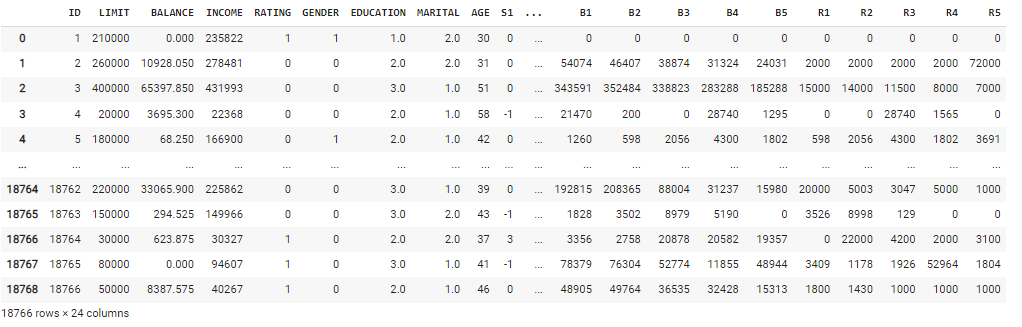
# Question 2

## 1st Data Pre-Processing

It is mentioned in Appendix 1 that ID is a unique customer number, and so there should be no duplicate ID. When I checked the csv file, I discovered three IDs that were duplicated and had the identical values for the rest of the columns in the same row.

The IDs that have been identified are 132, 378, and 420. Executing code 2.1 will remove the duplicate row while retaining the first row.

|  |
| --- |
| Code 2.1 |
| # First data pre-processing - dropping duplicate, Duplicate values - 132, 378, 420  df = df.drop\_duplicates(keep='first')  # Displaying and checking if the duplicated values are dropped  df |



Output 2: Duplicate IDs dropped

Output 1 shows that there are 18,769 rows, and Output 2 shows that there are 18,766 rows after eliminating the duplicate IDs but maintaining the first row, indicating that 3 rows were eliminated.

Alternatively, executing the following code to export the data after removing the columns to see whether there are any repeated IDs 132,378,420.

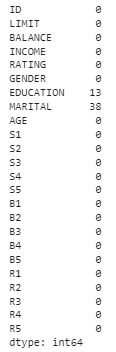
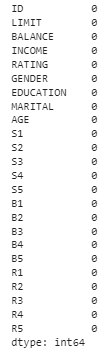
df.to\_excel ('test\_1.xlsx', index = False, header=True)

## 2nd Data Pre-Processing

The strategy is to execute cell 2.2.1 to check for any null or missing values in the DataFrame as part of the data cleansing. Output 3 shows that EDUCATION and MARITAL have null values. As a result, the following step in cell 2.2.2 is to discard the rows that contain null values since the data for that row is incomplete. If kept, we are unable to interpret any future results by visual or insight to represent.

Cell 2.2.3 checked to see if cell 2.2.2 was performed successfully, therefore there are no further null values in the DataFrame, as seen in Output 4.

|  |  |
| --- | --- |
| Cell | Code 2.2 |
| 2.2.1 | # Second data pre-processing - Dropping null values  # count the number of null values in each column of DataFrame  df.isnull().sum(axis = 0) |
| 2.2.2 | # Dealing with missing values which the rows contain NAN values  df.dropna(inplace= True) |
| 2.2.3 | # Checking dataframe after removing the NAN Values  df.isnull().sum(axis = 0) |

|  |  |
| --- | --- |
| Output 3: Cell 2.2.1 | Output 4: Cell 2.2.3 |

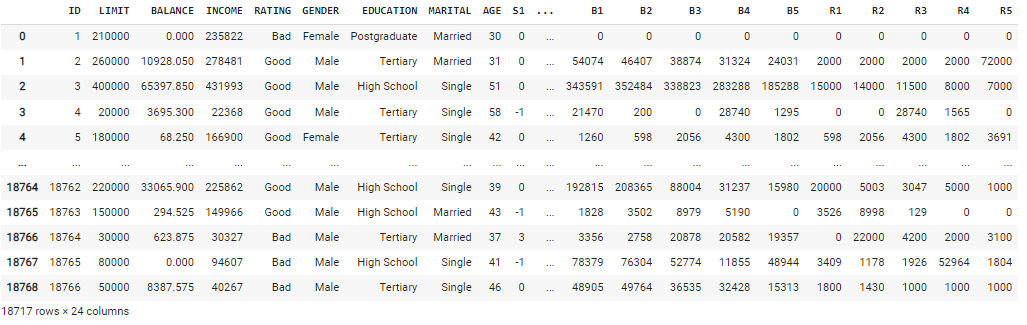
## 3rd Data Pre-Processing

First, we identified and corrected certain quality issues. In the category columns of GENDER, EDUCATION, MARITAL, and RATING, numerical values are utilised to indicate what the columns show. Too much referring may cause confusion in the latter portion.

With reference to Appendix 1, the strategy for cell 2.3.1 is to convert the indicated category column that includes 0, 1, 2, or 3 to be substituted with what they represent in the following:

* GENDER: 0 - Male, 1 – Female
* EDUCATION: 0 - Others, 1 - Postgraduate, 2 - Tertiary, 3 – Highschool
* MARITAL: 0 - Others, 1 - Single, 2 - Married
* RATING: 0 - Good, 1 - Bad

|  |  |
| --- | --- |
| Cell | Code 2.3 |
| 2.3.1 | # Third Data Preprocessing - Replacing specific categorical columns with specific attributes  # Replace Gender types with 0 - Male, 1 - Female  df['GENDER'].replace(to\_replace=[0,1], value=["Male","Female"], inplace=True)  # Replace the education with the following:  # 0 - Others, 1 - Postgraduate, 2 - Tertiary, 3 - Highschool  df['EDUCATION'].replace(to\_replace=[0, 1, 2, 3],                          value=['Others', 'Postgraduate', 'Tertiary', 'High School'],                          inplace = True)  # Replace Marital Status with 0 - Others, 1 - Single, 2 - Married  df['MARITAL'].replace(to\_replace=[0,1,2],                       value=['Others', 'Single', 'Married'],                       inplace = True)  # Replace Customer rating with 0 - Good, 1 - Bad  df['RATING'].replace(to\_replace=[0,1],                       value=['Good', 'Bad'],                       inplace = True) |
| 2.3.2 | # By displaying the dataframe to check if the replacement has been done correctly  df |



Output 5: Cell 2.3.2

## 4th Data Pre-Processing

There are outliers for the AGE column in the early stages of verifying the dataset of the csv file. -1 and 199 are the values. According to Wikipedia, the oldest person who has lived was 122 years old, hence age 199 is not applicable in this dataset. Most credit card companies only allow anyone aged 21 and over to apply for credit cards, therefore -1 years old will be dropped.

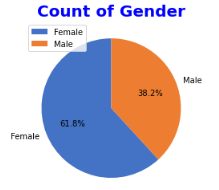
The technique, as shown in code 2.4 below, is to remove the outlier values in the age column that contain -1 and 199.

|  |
| --- |
| Code 2.4 |
| # Dropping age values that contain -1,199  # During the data preparation, it is observed that there are age of -1 and 199  # As according to wikipedia, the only person that has ever lived was 122 years old, hence age 199 years old is not applicable  # Most of the credit card company only allows for 21 years old and above to apply for credit card, hence -1 years old will be drop  # Hence we are dropping the outliers values in age that contains -1 and 199 in the age column  age\_outliers1 = df[ (df['AGE'] == -1)].index  df.drop(age\_outliers1, inplace=True)    age\_outliers199 = df[ (df['AGE'] == 199)].index  df.drop(age\_outliers199, inplace=True) |

# Question 3

## Visualisation 1 – Count of Gender

During the initial step of cleaning the data, it is discovered that there is a substantial disparity in customer gender. According to an early check of the csv file, there are more than 50% females. To corroborate the conclusions, we will first look at the value count shown in table 2 that was executed by code 3.1. Following that, display the appropriate pie chart to visualise the number of customers by gender type. Because there are only two types of datasets for gender to present, a pie chart is used.

|  |  |
| --- | --- |
| Figure 1: Percentage of Customers by Gender | Table 2: Count of Customers by Gender |

Figure 1 is a pie chart that shows the number of customers by gender proportion. Running the code 3.1 reveals in figure 1 that more than half of the customers are female (61.8%), while 38.2% are male, indicating that there are more female customers than men.

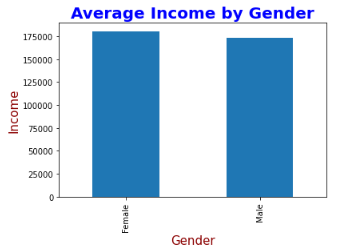
Table 2 displays the client count, which includes 11,563 females and 7,144 males. It is also proved by figures in table form that females outnumber males as credit facility customers.

|  |
| --- |
| Code 3.1 |
| # To display percentage and count of the gender that uses credit card  # Storing the gender type in x variable  pie\_chart\_x =list(df['GENDER'].value\_counts().keys())  # Storing the count of gender in y variable  pie\_chart\_y= list(df['GENDER'].value\_counts())  mycolours\_table1 = ['#4472C4', '#ED7D31']  font1 = {'color':'blue', 'size':20}  # Plotting pie chart with respect to number of gender  plt.pie(pie\_chart\_y, labels=pie\_chart\_x, startangle = 90, colors = mycolours\_table1, autopct='%2.1f%%')  # Sets the pie chart's title and legend  plt.title('Count of Gender', fontdict = font, fontweight='bold')  plt.legend()  # Count of gender and displaying its value  values = df['GENDER'].value\_counts()  print(values) |

## Visualisation 2 – Average Income by Gender

Because there are more female customers than male customers, I'd like to see if there is a significant variation in average income by gender. Because there are just two categories of gender in this information, a bar chart would be an excellent visual choice.

First, in cell 3.2.1, we run the code to display the average income of female and male customers. Second, we executed the code in cell 3.2.2 to display the bar chart by gender average income.

|  |  |
| --- | --- |
| Figure 2: Average Income by Gender | Table 3: Table of Average Income by Gender |

According to Table 3, the average income of female customers is $180,513.28 and that of male customers is $173,531.21. Despite the fact that there are more female consumers than male customers, an interesting fact is that female customers have a higher income than male customers.

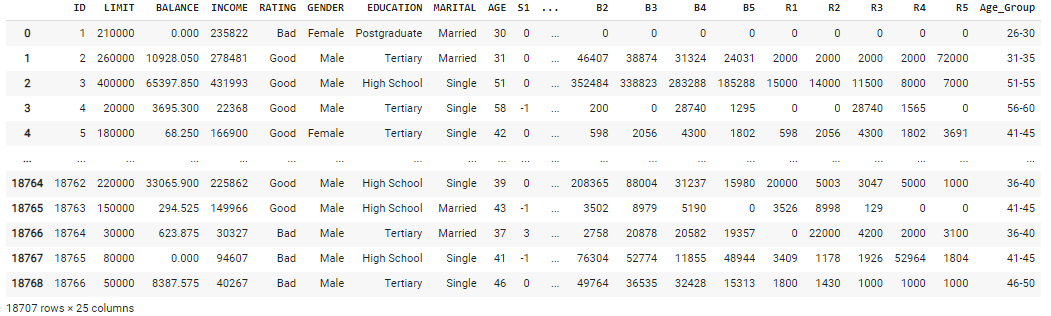
Figure 2 shows that the relative higher income of female customers is not far behind that of male customers.

|  |  |
| --- | --- |
| Cell | Code 3.2 |
| 3.2.1 | # Displaying the average of income by gender  income\_mean = df.groupby('GENDER').INCOME.mean()  income\_mean |
| 3.2.2 | # From Figure 1 we would like to see if there is any large difference in income since female count is higher than male  # From table 3, we will display the plot bar  font1 = {'color':'blue', 'size':20}  font2 = {'color':'darkred', 'size':15}  income\_mean.plot.bar()  plt.title('Average Income by Gender', fontdict = font1, fontweight='bold')  plt.xlabel('Gender',fontdict = font2)  plt.ylabel('Income',fontdict = font2)  plt.show() |

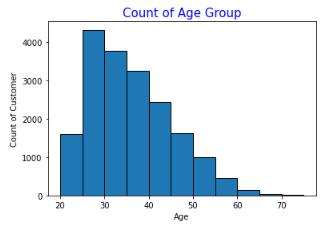
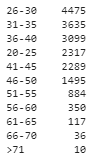
## Visualisation 3 – Age Distribution

Figure 1 shows that there are more female customers, whereas Figure 2 shows that the average income is very close. Now, I'd want to determine which age groups have the most customers, since the dataset shows that the age varies from 21 to 80 years old after eliminating outliers in question 2 of the data pre-processing.

We executed the code in cell 3.3.1 to categorise the ages. As seen in Output 6, first allocating ages to age groups using conditions and a list of conditions, then filling in the values and appending it to the end of the DataFrame.



### Output 6: Cell 3.3.1

|  |  |
| --- | --- |
| Figure 3: Age Distribution By Age Group | Table 4: Count of Age Group |

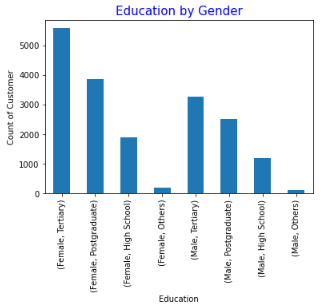
Figure 3 depicts the customer age distribution by age group. The distribution is skewed to the right, indicating that it is positively skewed. The distribution has a high count in the younger age groups and a low count in the older age groups. Outliers have been removed, therefore the data given is entirely older than 21 years. The majority of customers are between the ages of 26 and 30.

According to table 4 and executing code in 3.3.2, the largest age group of credit facility clients is 4,475 persons, with the majority of them being between the ages of 26 and 30. It is consistent with the facts provided visually in figure 3. Customers tend to be younger, since they are at the age where they have just begun working and receiving credit cards.

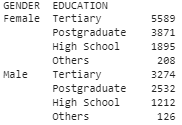
|  |  |
| --- | --- |
| Cell | Code 3.3 |
| 3.3.1 | # Assigning ages to age group using conditions  # Using Numpy Select to Set Values using Multiple Conditions  # Creating a list of conditions  conditions = [      (df['AGE'] >= 20) & (df['AGE'] <26), (df['AGE'] >= 26) & (df['AGE'] <31),      (df['AGE'] >= 31) & (df['AGE'] <36), (df['AGE'] >= 36) & (df['AGE'] <41),      (df['AGE'] >= 41) & (df['AGE'] <46), (df['AGE'] >= 46) & (df['AGE'] <51),      (df['AGE'] >= 51) & (df['AGE'] <56), (df['AGE'] >= 56) & (df['AGE'] <61),      (df['AGE'] >= 61) & (df['AGE'] <66), (df['AGE'] >= 66) & (df['AGE'] <71),      (df['AGE'] >= 71),  ]  # Creating corresponding values to fill  age\_values = ['20-25', '26-30', '31-35', '36-40',                '41-45', '46-50', '51-55', '56-60',                '61-65', '66-70',                '>71']  df['Age\_Group'] = np.select(conditions, age\_values)  df |
| 3.3.2 | # Table Count of Age Group  age\_group\_count = df['Age\_Group'].value\_counts()  print(age\_group\_count) |
| 3.3.3 | %matplotlib inline  # Plotting and displaying histogram  plt.hist(df['AGE'], bins=range(20,80,5), edgecolor ='black')  # Sets font1 as blue colour with size 15  font1 = {'color':'blue', 'size':15}  # Sets font2 as black colour with size 10  font2 = {'color':'black', 'size':10}  # Sets chart title  plt.title('Count of Age Group', fontdict = font1)  # Sets x-axis label  plt.xlabel("Age", fontdict = font2)  # Sets y-axis label  plt.ylabel("Count of Customer", fontdict = font2)  plt.show() |

## Visualisation 4 – Education Level by Gender

We are curious to see if education has any effect on income because females earn significantly more, as shown in Figure 2. To begin, execute cell 3.4.1 to show the educational values by gender and place them in the table. Then, using a bar chart, we would execute cell 3.4.2 to graphically portray the data of education level by gender.



### Figure 4: Education of Customer by Gender



### Table 5: Count of Education Type by Gender

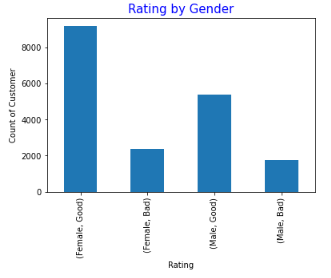
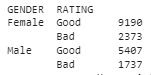
Table 5 reveals that female customers are more educated than male customers, with 5,589 and 3,871 female customers having tertiary and postgraduate education, respectively, and 3,274 and 2,532 male customers having tertiary and postgraduate education. It can be observed that this might be one of the reasons for the higher income shown in figure 2.

Figure 4 shows that the number of female customers has a much higher level of education than male customers.

|  |  |
| --- | --- |
| Cell | Code 3.4 |
| 3.4.1 | # Displaying the values of education by gender  education\_group\_gender = df.groupby('GENDER').EDUCATION.value\_counts()  education\_group\_gender |
| 3.4.2 | # Plotting the bar chart  education\_group\_gender.plot.bar()  # Sets font1 as blue colour with size 15  font1 = {'color':'blue', 'size':15}  # Sets font2 as black colour with size 10  font2 = {'color':'black', 'size':10}  # Sets chart title  plt.title('Education by Gender', fontdict = font1)  # Sets x-axis label  plt.xlabel("Education", fontdict = font2)  # Sets y-axis label  plt.ylabel("Count of Customer", fontdict = font2)  plt.show() |

## Visualisation 5 – Rating by Gender

Another finding from the data is the overall ratio of gender ratings. As a result, we are curious to examine if, based on the numbers shown thus far, females are on the dominating side and if females have a higher customer rating than males. First, we execute cell 3.5.1 to display the ratings values by gender. Second, we execute cell 3.5.2 to see the previously run code by utilising a bar chart to chart comparison.

|  |  |
| --- | --- |
| Figure 5: Rating of Gender | Table 6: Rating of Gender by Count |

According to table 6, the overall ratio is considerably better in 'customer,' as there are more Good ratings in females (9,190 vs. 5,407 males).

Figure 5 demonstrates that the majority of customers have been rated 'good,' and this includes both males and females.

|  |  |
| --- | --- |
| Cell | Code 3.5 |
| 3.5.1 | # Displaying the values of ratings by gender  rating\_group\_gender = df.groupby('GENDER').RATING.value\_counts()  rating\_group\_gender |
| 3.5.2 | # Plotting the bar chart  rating\_group\_gender.plot.bar()  # Sets chart title  plt.title('Rating by Gender', fontdict = font1)  # Sets x-axis label  plt.xlabel("Rating", fontdict = font2)  # Sets y-axis label  plt.ylabel("Count of Customer", fontdict = font2)  plt.show() |

# Question 4

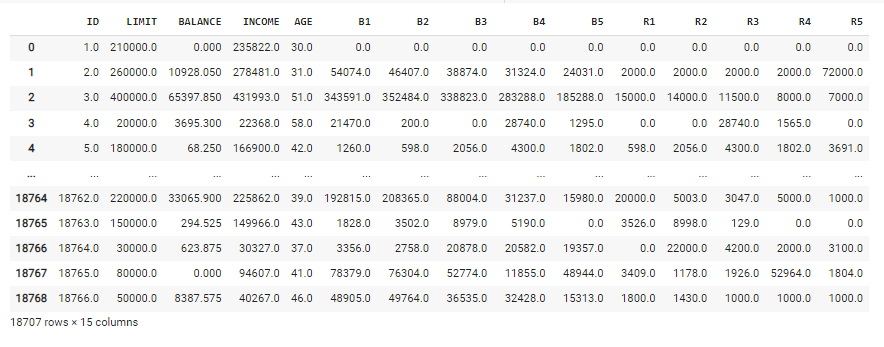
Before performing the linear regression modelling to predict the variable ,B1, we shall conduct further data pre-processing, followed by Linear Regression Model training, lastly, predicting the variable B1

## Further Data Pre-Processing

In code 4.1, we first import the libraries needed for the complete question 4. We remove the following categorical variables from the table in the second code, including the Age Group that was included in visualisation. The categorical columns are GENDER, EDUCATION, RATING, MARITAL, Age Group, and S1 through S5.

It should also be noticed that certain columns include the characters ',' and '$'. As a result, we delete the aforementioned strings from all of those columns. We run the following code to test if the additional data pre-processing was completed appropriately, as shown in Output 7. Alternatively, we may validate the DataFrame by exporting it to an Excel file.

|  |
| --- |
| Code 4.1 |
| # First we begin with the data pre-processing  # Starts, with importing the relevant libraries required  from sklearn.linear\_model import LinearRegression  from sklearn.decomposition import PCA  from sklearn.metrics import mean\_squared\_error  from sklearn.preprocessing import StandardScaler |
| # We shall remove the categorical attributes which includes the Age group that was created in visualisation 3  df = df.drop(columns=['GENDER', 'EDUCATION','RATING','MARITAL',                        'Age\_Group', 'S1', 'S2', 'S3', 'S4', 'S5'])  # We also replace the following: ',' and '$' signs  for col in df:        df[col] = df[col].astype(str).str.replace('$','')      df[col] = df[col].str.replace(',','').astype(float) |
| # We shall check the if the data pre-processing was executed correctly  df  # Alternatively, we can execute an command to export the dataframe to excel file to check  # df.to\_excel ('test\_2.xlsx', index = False, header=True) |



### Output 7: Result of Dropped Categorical Columns

## Linear Regression Model Training

To train the linear regression model, we split the data into training and test sets in code 4.2, and then we start training the model. We split the input and target attributes from the data set, then I transformed them to NumPy arrays before storing them in the variables 'X train' and 'y train'.

Because the data ranges for all of the features differ, we create variables 'X transformers' and 'y transformers' to apply data standardization on all of the input and target attributes.

Next, to improve the model's efficiency, we employ the fit transform approach, which is a hybrid of the fit and transform methods. It applies the combination procedure to the input data at once, transforming the data points. Following that, display the 'X train' and 'y train' shapes found in Output 8.

I developed a linear regression model and trained it after doing additional data pre-processing steps and ready to train it.



### Output 8: Result of X and Y shape

|  |
| --- |
| Code 4.2 |
| # Next we shall split the dataset into X\_train & y\_train to prepare for data standardization  X\_train = df[['LIMIT','BALANCE','INCOME','AGE',                'B2','B3','B4','B5',                'R1','R2','R3','R4','R5']].to\_numpy()  y\_train = df['B1'].to\_numpy()  # Introducing new transforms for the data standardization  X\_transformers = StandardScaler()  y\_transformers = StandardScaler()  # We shall perform the data standardization on all the features using fit transform method  X\_train = X\_transformers.fit\_transform(X\_train)  y\_train = y\_transformers.fit\_transform(y\_train.reshape(-1, 1))  # We shall then print the respective shape for both X\_train and y\_train  print("The X shape is ", X\_train.shape)  print("The Y shape is ", y\_train.shape)  # Next is to create a new linear regression model to prepare for fitting  model = LinearRegression()  # Lastly is to fit the model into our dataset  model.fit(X\_train, y\_train) |

## Predicting B1 Variable of Index 5

After training the linear regression model, I choose index 5 in code 4.3 to predict the variable B1 on a fresh dataset using our test data and evaluate how well it predicts. Following that, when the DataFrame of index 5 is displayed in Output 9, it is seen that the B1 variable of index 5 is 132,185.0.



### Output 9: DataFrame of Index 5

|  |
| --- |
| Code 4.3 |
| # Next we shall predict the variable B1 on a new dataset  # We shall pick an index 5 from the data to test  new\_dataset = X\_train[5]  # We shall print the selected sample that was picked  print("The dataframe of the chosen sample are:")  df.iloc[[5]] |

In order to incorporate the selected sample into the model, code 4.4 requires it to be reshaped. Following that, the aforementioned input is input into the model of prediction B1 value and the outcome is stored in variable predicted b1 value.

The model's output is then inverse transformed to display the predicted value of index 5, which is 130,953.12, as shown in Output 10. It has a 0.9% inaccuracy when the difference between the actual and predicted values of index 5 is divided by the actual value.



### Output 10: Predicted Value of Index 5

|  |
| --- |
| Code 4.4 |
| # We shall reshape the selected sample so we can input it to the model  new\_dataset = new\_dataset.reshape([1,X\_train.shape[1]])  # Next is to feed the input to the model of predict b1 and storing the result in predicted\_b1\_value  predicted\_b1\_value = model.predict(new\_dataset)  # Performing an inverse transform on the output of model  predicted\_b1\_value = y\_transformers.inverse\_transform(predicted\_b1\_value)  # We shall print the predicted value  print("Predicted value: ", predicted\_b1\_value) |

# Question 5

## Linear Regression Equation

Following the training of the linear regression model, we execute code 5.1 to acquire the following equation, output 11 to obtain the outcome of the learnt parameters, and output 12 to obtain the intercept.

B1 = (-8.79581622 × 10-18 ) + (0.013\*LIMIT) + (0.47\*BALANCE) – (0.011\*INCOME) + (0.0000011\* AGE) + (0.55\*B2) – (0.017\*B3) – (0.034\*B4) + (0.041\*B5) - (0.10\*R1) + (0.036\*R2) + (0.0099\*R3) - (0.012\*R4) - (0.0015\*R5)

From output 11 and with referencing to the above equation, the linear regression model gives a more importance to BALANCE and B2. It also contains negative weights for INCOME, B3, B4, R1, R4, and R5.



Output 11: Learned Parameters of Linear Regression Model



Output 12: Intercept value

|  |
| --- |
| Code 5.1 |
| # Printing the learned parameters of the linear regression model  print("Learned Parameters of our linear regression model: ",model.coef\_)  # Intercept value  print("Intercept Value: ", model.intercept\_) |

## Mean Squared Error

The MSE is defined as the average squared difference between observed and predicted values. We run code 5.2 to obtain output 13 which is the value of mean squared error of 0.05.

Considering that the output values are standardized, the error is excessive. This suggests that linear regression is ineffective for predicting B1 and that a more complicated model is required.



Output 13: Mean Squared Error Value

|  |
| --- |
| Code 5.2 |
| # Calculation of the Mean Squared Error (MSE)  y\_pred = model.predict(X\_train)  print('Mean Squared Error (MSE): ',mean\_squared\_error(y\_train,y\_pred)) |

## Visual Aid Actual vs Predicted Value

In table form, we may compare the actual values in column B1 to the predicted values. Due to the large number of records, I'm only showing the top 25 rows for clarity. To obtain Table 7, we ran code 5.3 to display the top 25 rows of a DataFrame.

We can also see the comparison result of Table 7 by running the following line of code in code 5.3, which displays Figure 6. Although the model is not accurate, the predicted percentages are close to the actual ones.



Table 7: Actual & Predicted values of B1

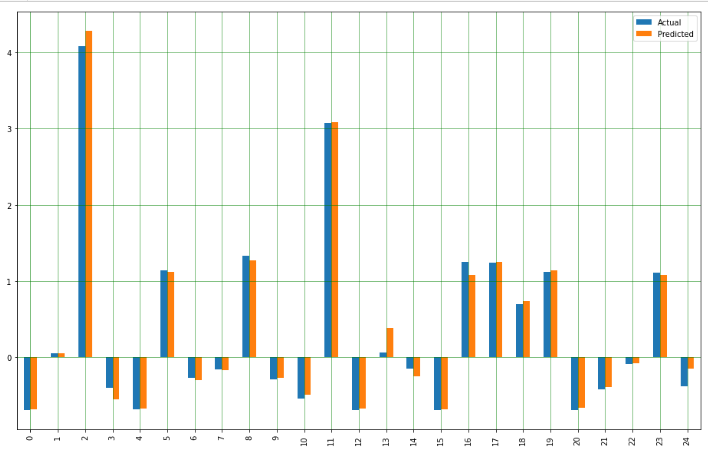


Figure 6: B1 Bar Graph Actual vs Predicted

|  |
| --- |
| Code 5.3 |
| # Creation of Actual and Predicted DataFrame  df1 = pd.DataFrame({'Actual': y\_train.flatten(), 'Predicted': y\_pred.flatten()})  df1.head(25) |
| # Plotting bar chart of Actual vs Predicted  df2 = df1.head(25)  df2.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() |

# References

6.3. Preprocessing data. (n.d.). Scikit-Learn. https://scikit-learn.org/stable/modules/preprocessing.html

Badole, M. (2021, April 30). Difference Between fit(), transform(), fit\_transform() methods in Scikit-Learn (with Python Code). Analytics Vidhya. https://www.analyticsvidhya.com/blog/2021/04/difference-between-fit-transform-fit\_transform-methods-in-scikit-learn-with-python-code/

A Beginner’s Guide to Linear Regression in Python with Scikit-Learn. (n.d.). KDnuggets. https://www.kdnuggets.com/2019/03/beginners-guide-linear-regression-python-scikit-learn.html

Brownlee, J. (2020a, June 10). How to Use StandardScaler and MinMaxScaler Transforms in Python. Machine Learning Mastery. https://machinelearningmastery.com/standardscaler-and-minmaxscaler-transforms-in-python/

Brownlee, J. (2020b, July 24). Train-Test Split for Evaluating Machine Learning Algorithms. Machine Learning Mastery. https://machinelearningmastery.com/train-test-split-for-evaluating-machine-learning-algorithms/

Change column type in pandas. (n.d.). Stack Overflow. https://stackoverflow.com/questions/15891038/change-column-type-in-pandas

Crail, C. (2022, March 19). Negative Balance On A Credit Card: What To Do? Forbes Advisor. https://www.forbes.com/advisor/credit-cards/what-is-a-negative-balance-on-a-credit-card/

GeeksforGeeks. (2019, June 30). Python | Mean Squared Error. https://www.geeksforgeeks.org/python-mean-squared-error/

Lawton, G. (2022, January 31). data preprocessing. SearchDataManagement. https://www.techtarget.com/searchdatamanagement/definition/data-preprocessing#:%7E:text=Data%20preprocessing%2C%20a%20component%20of,for%20the%20data%20mining%20process.

Real Python. (2022, May 24). Linear Regression in Python. https://realpython.com/linear-regression-in-python/

sklearn.linear\_model.LinearRegression. (n.d.). Scikit-Learn. https://scikit-learn.org/stable/modules/generated/sklearn.linear\_model.LinearRegression.html

Team, D. (2022, March 6). Drop Rows By Multiple Conditions In Pandas Dataframe. DevEnum.Com. https://devenum.com/drop-rows-by-multiple-conditions-in-pandas-dataframe/

Wikipedia contributors. (2022, September 3). List of the verified oldest people. Wikipedia. https://en.wikipedia.org/wiki/List\_of\_the\_verified\_oldest\_people

Z. (2020, April 24). Understanding the Standard Error of the Regression. Statology. https://www.statology.org/standard-error-regression/