****

**ANL252**

**Python for Data Analysis**

**End-of-Course Assessment**

**July 2022 Presentation**

**Submitted by:**

|  |  |
| --- | --- |
| **Name** | **PI No.** |
| **Kris Lennings** | **J1910196** |

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

**Submission Date: 04/09/2022**

Table of Contents

**Qn11**

**Qn2** **1**

1st stage- Import Data for Quality Assessment 1

2nd stage- Data Cleaning9

3rd stage- Data Transformation11

4th stage- Data Reduction12

**Qn3** **16**

**Qn4** **23**

**Qn5** **26**

**Qn6** **27**

**References29**

**Qn1**

|  |  |
| --- | --- |
| **Variable** | **Data Type** |
| ID | Nominal |
| 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 |
| Rating | Categorical |

**Qn2**

**1st stage- Import Data for Quality Assessment**

In this stage of data pre-processing, we need to perform a data quality assessment of the ECA-data. Here, we need to explore if there are any missing variables or outliers contained in the data. Such data could hinder the performance of our machine learning technique that result in a biased statistical estimation (Wu, 2022).

**i.**

First, we need to import the following packages and modules into the Jupyter environment.

*import pandas as pd*

*import numpy as np*

*from matplotlib import pyplot as plt*

*import seaborn as sns*

*import warnings*

*warnings.filterwarnings('ignore')*

**ii.**

Next, we'll load the ECA\_data and examine its structure as a data frame (df).

*missing\_values = ["$0"]*

*df\_raw = pd.read\_csv('ECA\_data.csv', na\_values = missing\_values)*

*df\_raw.drop(df\_raw.columns[0], axis=1, inplace=True)*

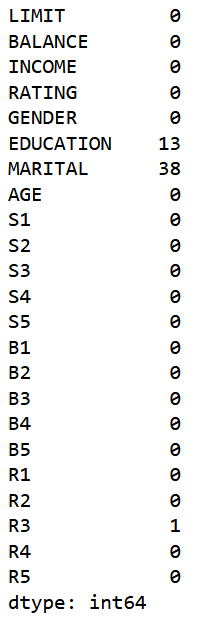
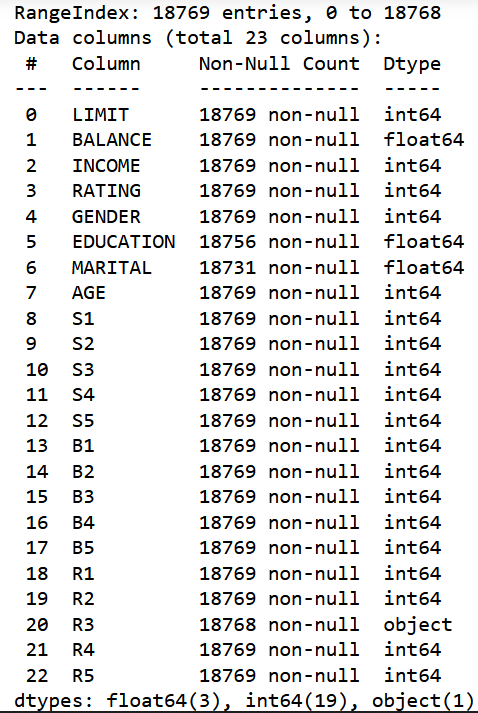
*df\_raw*

**iii.**

To obtain a quick overview of the loaded dataset, we will use following codes for python to summarise the missing values contained in the dataset. We can observe that there are 13, 38 and 1 missing values contained in the 'EDUCATION', 'MARITAL' and ‘R3’ columns respectively.

*df.info()*

*df.isna().sum()*

*Figure 1. Missing values contained in Figure 2. R3 is captured as an object*

*the EDUCATION, MARITAL and R3 columns*

We noticed that the 'R3' variable is an object rather than an integer or a float, and that it contains an empty value. As a result, we must address this issue immediately because it may have an impact on our mathematical and statistical analyses during our data pre-processing stage.

First, we must replace the missing values in the 'R3' column with the 'Most Frequent' variable count.

*df\_raw['R3'].value\_counts()*

*df\_raw['R3'].fillna(df\_raw['R3'].value\_counts().index[0]*

*,inplace = True)*

Next, the 'R3' column must then be converted from an object to an integer.

*df\_raw['R3'] = df\_raw['R3'].str.replace('$','')*

*df\_raw['R3'] = df\_raw['R3'].str.replace(',','')*

*df\_raw['R3'] = df\_raw['R3'].astype(int)*

*df\_raw.info()*

**iv.**

Before we can detect outliers, we must first understand the characteristics of the numerical values in the dataset.

We should expect some outliers in the 'LIMIT,' 'BALANCE,' and 'INCOME' columns. This is because income varies from person to person, and a few individuals may have earned more than the rest of the population. As a result, the 'LIMIT' and 'BALANCE' amounts for these individuals may differ, and this is likely to be reflected as an outlier datapoint.

*LIMIT = (df\_raw['LIMIT'])*

*BALANCE = (df\_raw['BALANCE'])*

*INCOME = (df\_raw['INCOME'])*

*data = [LIMIT, BALANCE, INCOME]*

*fig = plt.figure(figsize =(10, 7))*

*ax = fig.add\_subplot(111)*

*# Creating axes instance*

*bp = ax.boxplot(data, patch\_artist = True,*

*notch ='True', vert = 0)*

*# x-axis labels*

*ax.set\_yticklabels(['LIMIT', 'BALANCE', 'INCOME'])*

*# Adding title*

*plt.title("Box plot for LIMIT, BALANCE & INCOME ")*

*# Removing top axes and right axes*

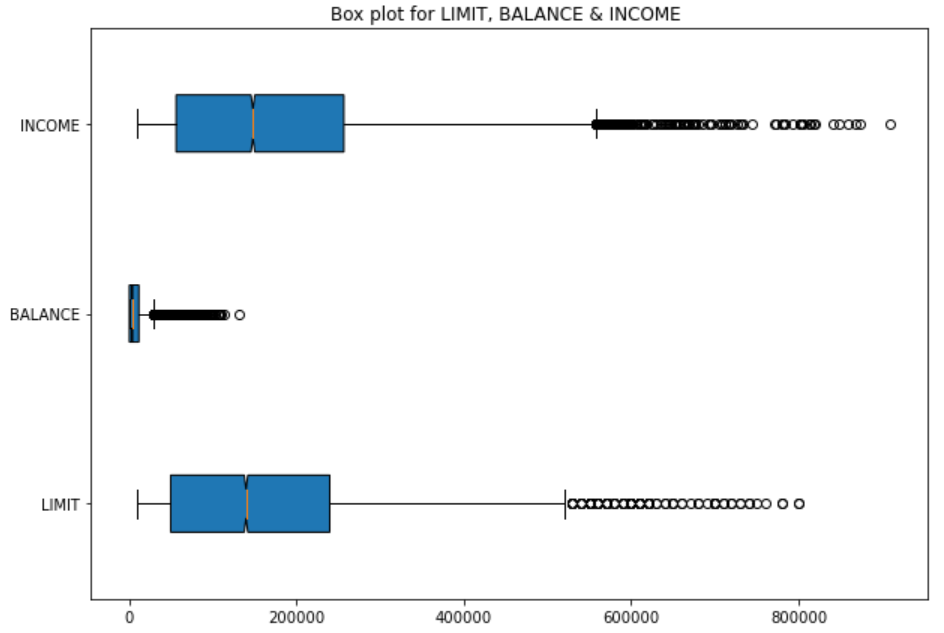
*# ticks*

*ax.get\_xaxis().tick\_bottom()*

*ax.get\_yaxis().tick\_left()*

*# show plot*

*plt.show()*



*Figure 3. Box plot for INCOME, BALANCE & LIMIT columns with outliers*

In terms of the 'B(n)' and 'R(n)' columns, the amount and ability to repay the bill may differ from one individual to the next. This is because their ability to repay their bills may be linked to their 'INCOME' earned. As a result, the bill repayment amount contained in the 'B(n)' and 'R(n)' columns may differ and be reflected as an outlier datapoint.

*B1 = (df\_raw['B1'])*

*B2 = (df\_raw['B2'])*

*B3 = (df\_raw['B3'])*

*B4 = (df\_raw['B4'])*

*B5 = (df\_raw['B5'])*

*data = [B1, B2, B3, B4, B5]*

*fig = plt.figure(figsize =(10, 7))*

*ax = fig.add\_subplot(111)*

*# Creating axes instance*

*bp = ax.boxplot(data, patch\_artist = True,*

*notch ='True', vert = 0)*

*# x-axis labels*

*ax.set\_yticklabels(['B1', 'B2', 'B3', 'B4', 'B5'])*

*# Adding title*

*plt.title("Box plot for B1 to B5 ")*

*# Removing top axes and right axes*

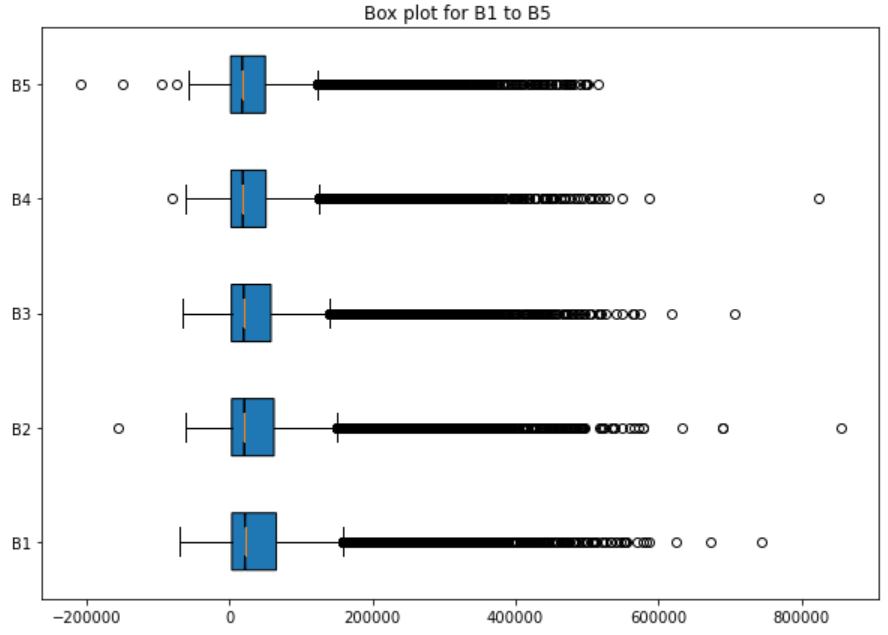
*# ticks*

*ax.get\_xaxis().tick\_bottom()*

*ax.get\_yaxis().tick\_left()*

*# show plot*

*plt.show()*



*Figure 4. Box plot for B1 to B5 columns with outliers*

*R1 = (df\_raw['R1'])*

*R2 = (df\_raw['R2'])*

*R3 = (df\_raw['R3'])*

*R4 = (df\_raw['R4'])*

*R5 = (df\_raw['R5'])*

*data = [R1, R2, R3, R4, R5]*

*fig = plt.figure(figsize =(10, 7))*

*ax = fig.add\_subplot(111)*

*# Creating axes instance*

*bp = ax.boxplot(data, patch\_artist = True,*

*notch ='True', vert = 0)*

*# x-axis labels*

*ax.set\_yticklabels(['R1', 'R2', 'R3', 'R4', 'R5'])*

*# Adding title*

*plt.title("Box plot for R1 to R5 ")*

*# Removing top axes and right axes*

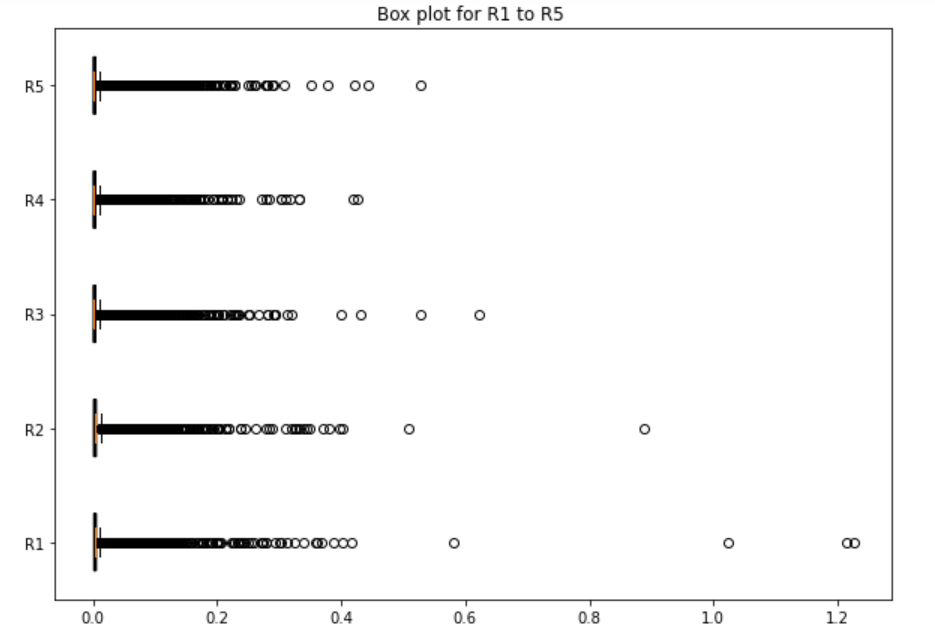
*# ticks*

*ax.get\_xaxis().tick\_bottom()*

*ax.get\_yaxis().tick\_left()*

*# show plot*

*plt.show()*



*Figure 5. Box plot for R1 to R5 columns with outliers*

However, the same cannot be said for the ‘AGE” column. Since age is an important factor in determining a customer's creditworthiness and loan eligibility, it is unlikely that the customer is under 0 or over 100 years old. We can see from the box plot that the 'AGE' column contains 10 extreme outlier data points.

*AGE = (df\_raw['AGE'])*

*data = [AGE]*

*fig = plt.figure(figsize =(10, 7))*

*ax = fig.add\_subplot(111)*

*# Creating axes instance*

*bp = ax.boxplot(data, patch\_artist = True,*

*notch ='True', vert = 0)*

*# x-axis labels*

*ax.set\_yticklabels(['AGE'])*

*# Adding title*

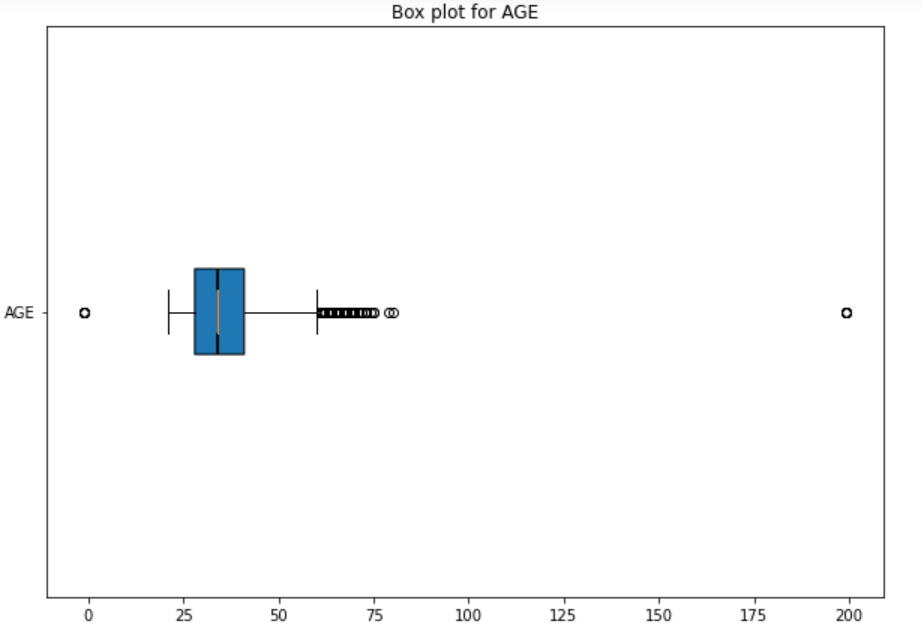
*plt.title("Box plot for AGE")*

*# Removing top axes and right axes*

*# ticks*

*ax.get\_xaxis().tick\_bottom()*

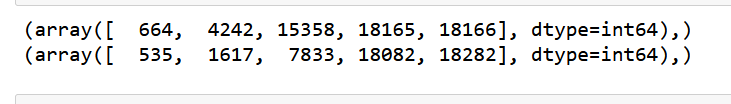
*ax.get\_yaxis().tick\_left()*



*Figure 6. Box plot for AGE columns with extreme outliers*

*print(np.where(df\_raw['AGE']<0))*

*print(np.where(df\_raw['AGE']>100))*



*Figure 7. Location of extreme outliers in the ‘AGE’ column*

**2nd Stage - Data Cleaning**

In this stage of data pre-processing, we will use the 'Most Frequent' variable count to replace the 'Blanks' in the 'EDUCATION' and 'MARITAL' columns respectively, as well as treat the outliers contained in the dataset.

**i.**

We will use the following codes to replace the 'Blanks' in the 'EDUCATION' and 'MARITAL' columns.

*df\_raw['EDUCATION'].value\_counts()*

*df\_raw['EDUCATION'].fillna(df\_raw['EDUCATION'].value\_counts().index[0]*

*,inplace = True)*

*df\_raw['MARITAL'].value\_counts()*

*df\_raw['MARITAL'].fillna(df\_raw['MARITAL'].value\_counts().index[0]*

*,inplace = True)*

**ii.**

Although it is preferable to remove all outliers from the dataset, we can help to minimise their presence by using the interquartile range method. This is because some of the outliers in the dataset may represent legitimate observations for the analysis. From the results below, we can observe that the shape of our data frame has significantly been reduced after the outliers have been removed.

*def outliers(df, ft):*

*Q1 = df[ft].quantile(0.25)*

*Q3 = df[ft].quantile(0.75)*

*IQR = Q3 - Q1*

*lower\_bound = Q1 - 1.5 \* IQR*

*upper\_bound = Q3 + 1.5 \* IQR*

*ls = df.index[(df[ft] < lower\_bound) | (df[ft] > upper\_bound)]*

*return ls*

*index\_list = []*

*for feature in ['LIMIT','BALANCE','INCOME','AGE',*

*'B1','B2','B3','B4','B5',*

*'R1','R2','R3','R4','R5']:*

*index\_list.extend(outliers(df\_raw, feature))*

*index\_list*

*def remove(df, ls):*

*ls = sorted(set(ls))*

*df = df.drop(ls)*

*return df*

*df\_cleaned = remove(df\_raw, index\_list)*

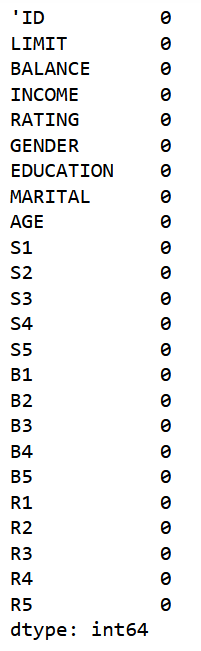
*df\_cleaned.shape*



*Figure 8. Shape of new data frame with outliers removed*

For this process, we have replaced the 'Blank' values as well as treated the outliers in the dataset.

*df.isna().sum()*



*Figure 9. Output obtained for df.isna().sum() code*

**3rd stage - Data Transformation**

In this stage of data preparation, we will normalise the numeric values in all the columns. This is because measuring the data on the same scale can contribute to a more accurate analysis for our study (Wu, 2022).

Before normalising the variables for the current dataset, we need to make a backup copy of the cleaned dataset for data exploration.

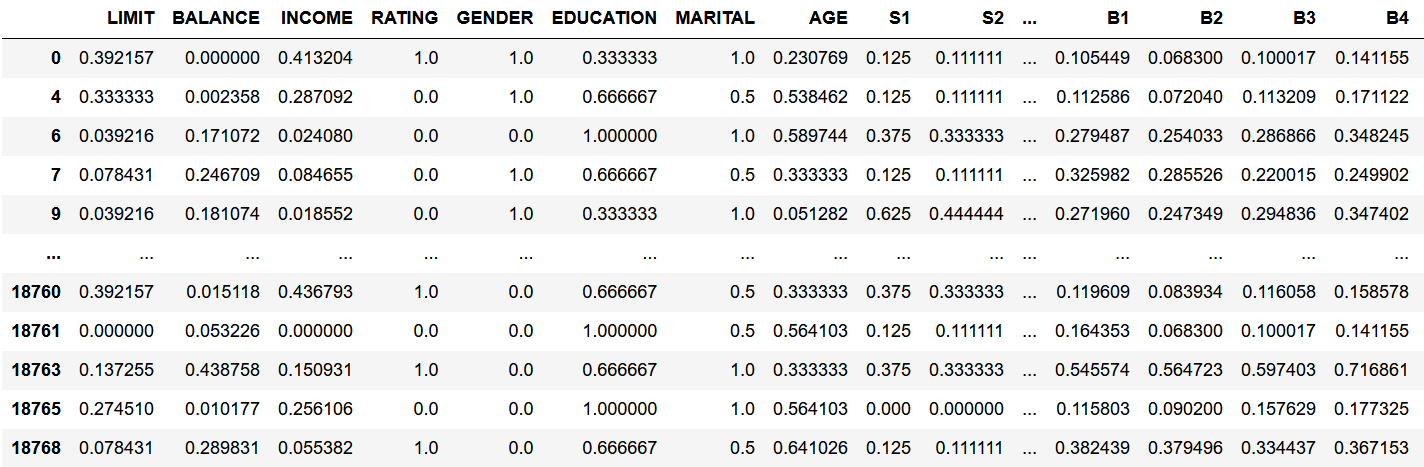
*df\_cleaned2 = df\_cleaned.copy()*

The following codes will be used to normalise the dataset columns.

*x = df\_cleaned.iloc[:,:]*

*df\_cleaned.iloc[:,:] = (x-x.min())/ (x.max() - x.min())*

*df\_cleaned*



*Table 1. Normalised data for all columns*

For this process, we have normalised all the column values contained in the dataset.

**4th stage - Data reduction**

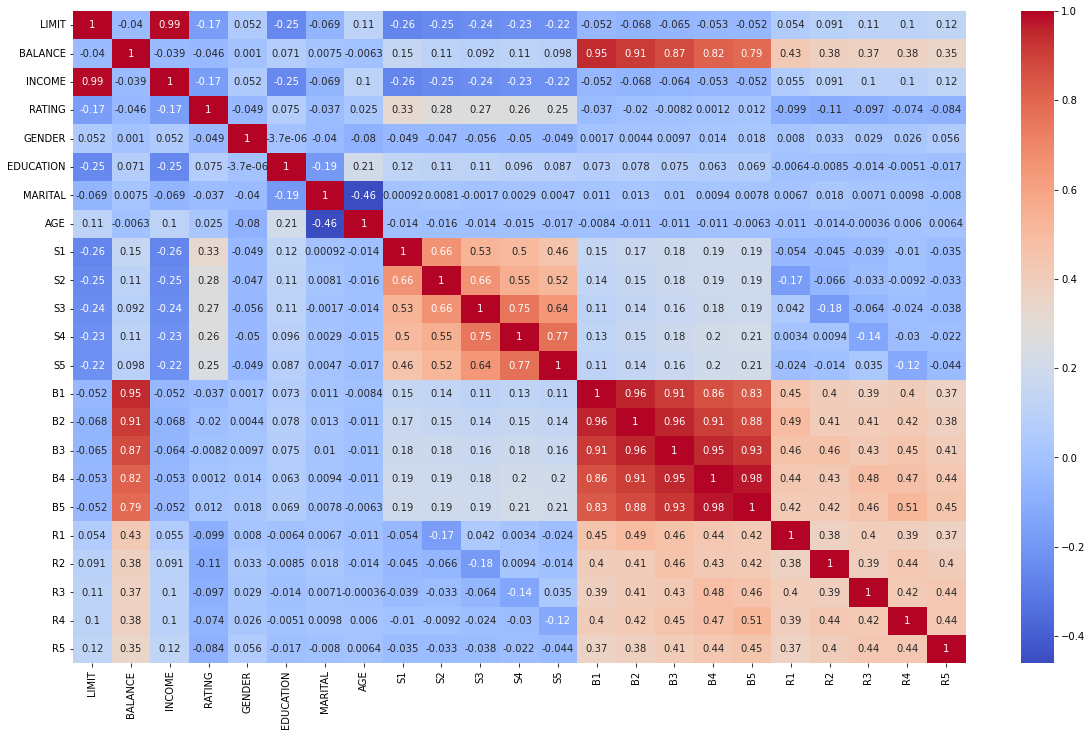
In this stage of data pre-processing, we will reduce the dimension of the dataset in order to obtain the optimal number of variables to feed into the machine learning algorithm. According to Sidall (2002), we should use the fewest number of entities to explain the occurrence of an event, i.e. the Principle of Parsimony. To accomplish this, we must examine the pair correlation values for each variable in the dataset.

**i.**

Using 'B1' as our dependent variable, we can create a correlation heatmap to identify the variables that are highly correlated. Here, we can list the top 15 independent variables with the highest correlation with 'B1.' They are: 'LIMIT', 'INCOME', 'RATING', 'AGE', 'GENDER', 'MARITAL', 'EDUCATION', 'S3', 'S5' & 'S4'.

*plt.figure(figsize = (20,12))*

*sns.heatmap(df\_cleaned.corr(), annot = True, cmap="coolwarm")*

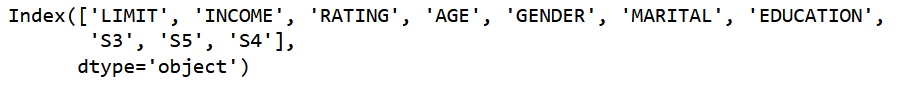
**

*Figure 10. Correlation heatmap*

*#sort the high correlation variables associated with 'INCOME'*

*cls = df\_cleaned.corr()['B1'].sort\_values().head(10).index*

*cls*



*Figure 11. Top 10 variables correlated with B1*

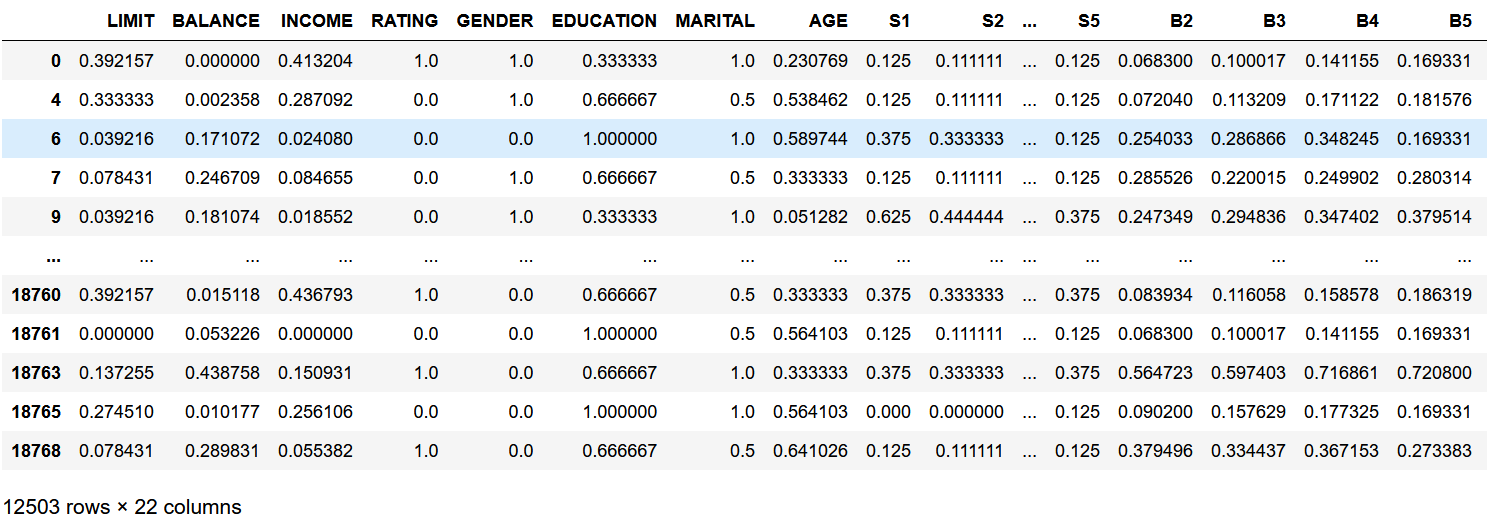
**ii.**

Alternatively, we can use PCA to break down all of the variables in the data frame into smaller components. First, we must remove the 'B1' column (target variable) from the dataset. Since we have normalised our dataset, we do not need to rescale the dataset again at this stage.

*X = df\_cleaned.drop(['B1’'],axis=1)*

*y = df\_cleaned['B1']*

*X*



*Figure 12. X data frame with B1 column removed*

Then, we need to create a covariance matrix.

*print('NumPy covariance matrix: \n%s' %np.cov(X.T))*

Following that, we compute the Eigen Values and Eigen Vectors to understand the multidimensional vector space of the *X* dataset.

*cov\_mat = np.cov(X.T)*

*eig\_vals, eig\_vecs = np.linalg.eig(cov\_mat)*

*print('Eigenvectors \n%s' %eig\_vecs)*

*print('\nEigenvalues \n%s' %eig\_vals)*

*eig\_pairs = [(np.abs(eig\_vals[i]), eig\_vecs[:,i]) for i in range(len(eig\_vals))]*

*print('Eigenvalues in descending order:')*

*for i in eig\_pairs:*

*print(i[0])*

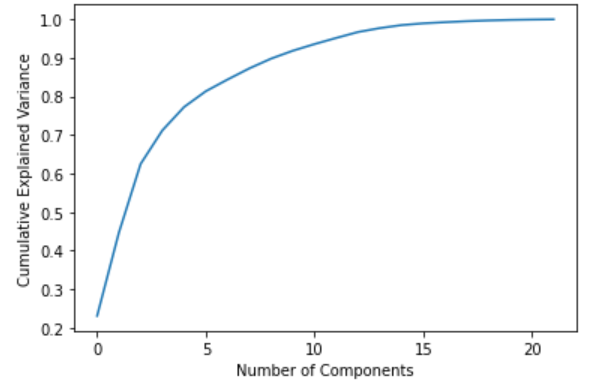
*pca = PCA().fit(X)*

*plt.plot(np.cumsum(pca.explained\_variance\_ratio\_))*

*plt.xlabel('Number of Components')*

*plt.ylabel('Cumulative Explained Variance')*

*plt.show()*



*Figure 13. Scree plot displaying cumulative variance of PCA components*

Using the following codes, we can determine that the optimal number of PCA components is 6. This is because the sum of the variance ratios for these components can explain approximately 80.8% of the variance in the *X* dataset.

*pca =PCA(n\_components=6)*

*X\_pca =pca.fit\_transform(X)*

*print(pca.explained\_variance\_ratio\_)*



*Figure 14. Variation of each PCA component*

*pca.explained\_variance\_ratio\_.sum()*



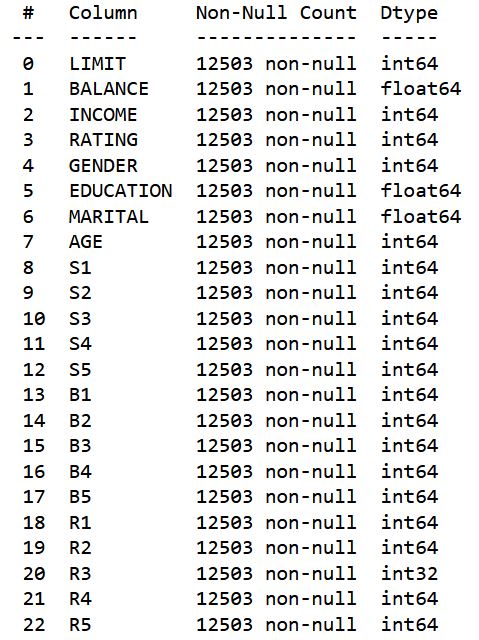
*Figure 15. Sum of variance of six PCA components*

For this process, we have reduced the dimension of the data frame that can be used for the next predictive modelling phase.

**Qn3**

In order to perform the data visualisation process, we need to convert the categorical variables such as 'RATING', 'GENDER', 'EDUCATION', 'MARITAL', 'AGE', 'S1', 'S2', 'S3', 'S4', & 'S5' as strings.

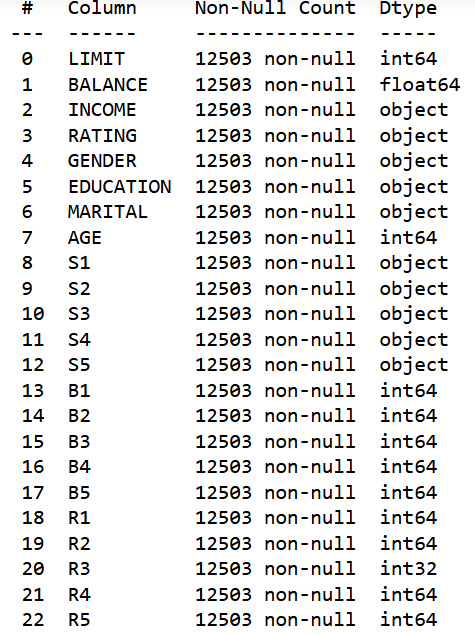
*df\_cleaned2.info()*



*Figure 16. Figure 1. Output obtained for df.info() code*

*df\_cleaned2[['RATING', 'GENDER', 'EDUCATION', 'MARITAL', 'S1', 'S2', 'S3', 'S4', 'S5']] = df\_cleaned2[['RATING', 'GENDER', 'EDUCATION', 'MARITAL', 'S1', 'S2', 'S3', 'S4', 'S5']].astype(str)*

*df\_cleaned2.info()*



*Figure 17. Updated output obtained for df.info() code*

**i.**

Generate a column chart displaying the GENDER distribution by EDUCATION level.

*Gender\_order = ['0','1']*

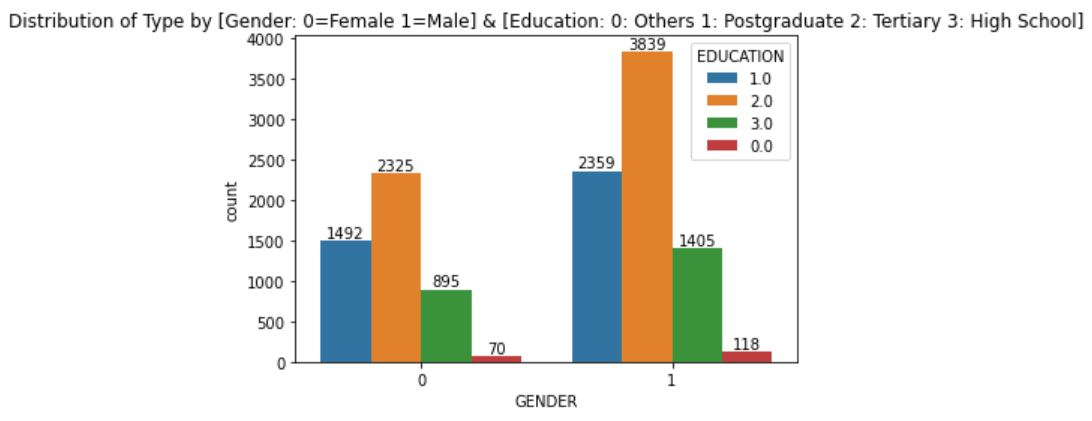
*ax = sns.countplot(x='GENDER',data=df\_cleaned2, hue='EDUCATION', order=Gender\_order)*

*for container in ax.containers:*

*ax.bar\_label(container)*

*p = ax*

*p.set\_title("Distribution of Type by [Gender: 0=Female 1=Male] & [Education: 0: Others 1: Postgraduate 2: Tertiary 3: High School]")*

*Figure 18. A column chart displaying the gender distribution by education level*

**ii.**

Generate a column chart displaying the GENDER distribution by RATING level.

*Gender\_order = ['0','1']*

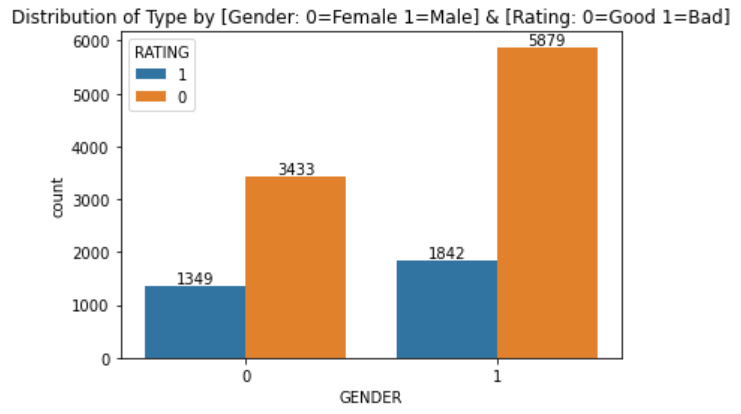
*ax = sns.countplot(x='GENDER',data=df\_cleaned2, hue='RATING', order=Gender\_order)*

*for container in ax.containers:*

*ax.bar\_label(container)*

*p = ax*

*p.set\_title("Distribution of Type by [Gender: 0=Female 1=Male] & [Rating: 0=Good 1=Bad]")*



*Figure 19. A column chart displaying the gender distribution by ratings*

**iii.**

Generate a column chart displaying the GENDER distribution by MARITAL status.

*Gender\_order = ['0','1']*

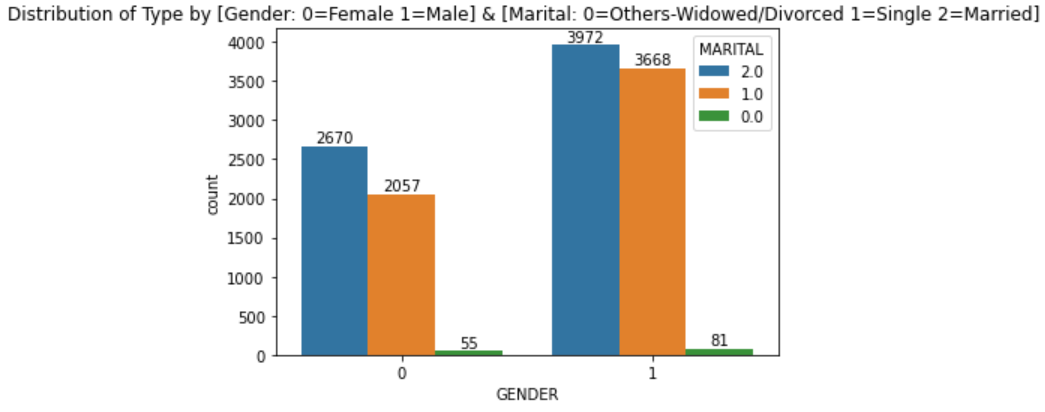
*ax = sns.countplot(x='GENDER',data=df\_cleaned2, hue='MARITAL', order=Gender\_order)*

*for container in ax.containers:*

*ax.bar\_label(container)*

*p = ax*

*p.set\_title("Distribution of Type by [Gender: 0=Female 1=Male] & [Marital: 0=Others-Widowed/Divorced 1=Single 2=Married] ")*



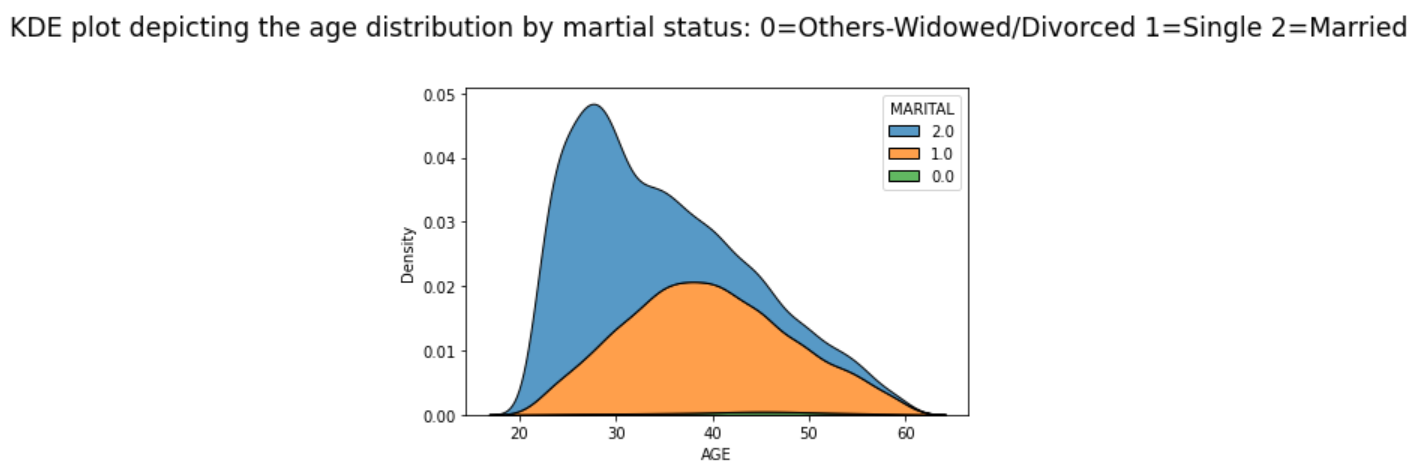
*Figure 20. A column chart displaying the gender distribution by marital status*

**iv.**

Generate a Kernel Density Estimation (KDE) plot displaying the AGE distribution by MARITAL status.

*sns.kdeplot(data=df\_cleaned2, x='AGE', hue = 'MARITAL', multiple='stack')*

*plt.suptitle('KDE plot depicting the age distribution by martial status: 0=Others-Widowed/Divorced 1=Single 2=Married', x=0.5, y=1.05, ha='center', fontsize='xx-large')*



*Figure 21. A KDE chart displaying the age distribution by marital status*

**v.**

Generate a histogram chart displaying the INCOME distribution by GENDER.

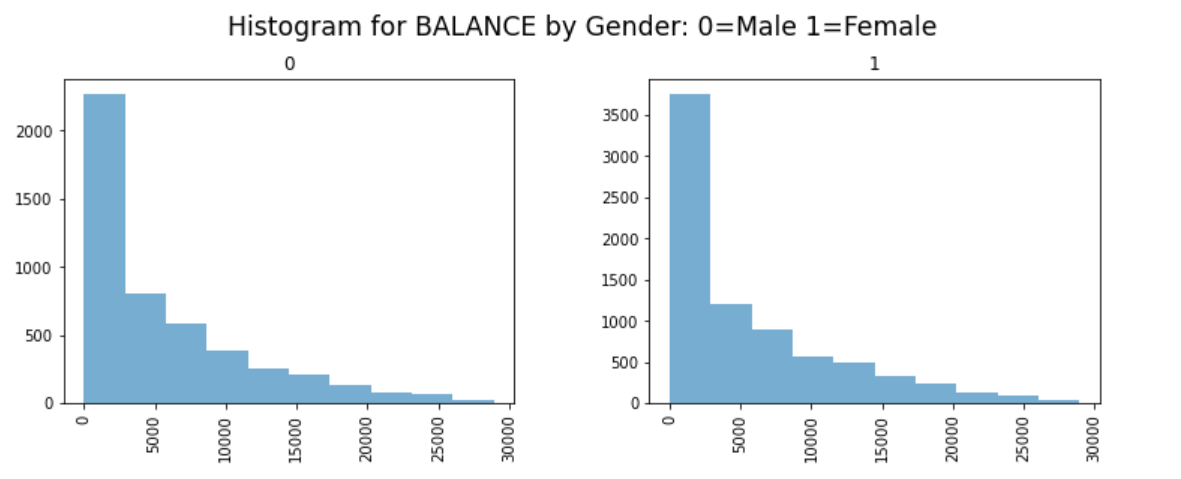
*df\_cleaned2.hist(column = 'INCOME', by = 'GENDER', bins=10,*

*figsize=(12,8),*

*alpha=0.6,*

*grid=False)*

*plt.suptitle('Histogram for INCOME by Gender: 0=Male 1=Female', x=0.5, y=1.05, ha='center', fontsize='xx-large')*



*Figure 22. A histogram chart displaying the income distribution by gender*

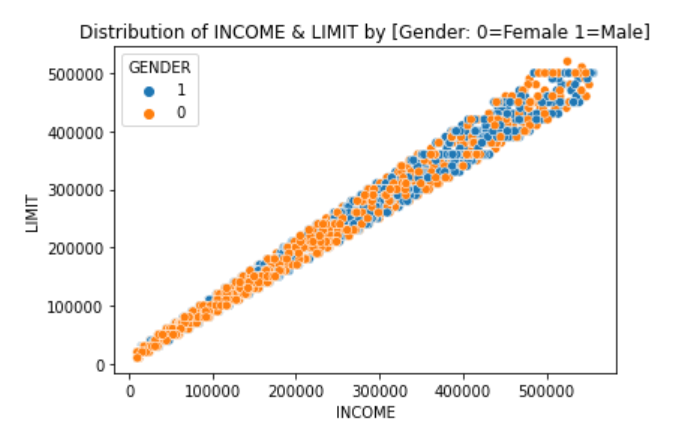
**vi.**

Generate a scatter plot displaying the INCOME and LIMIT distribution by GENDER

*sns.scatterplot( x='INCOME', y='LIMIT', hue = 'GENDER',*

*data = df\_cleaned2);*

*plt.title('Distribution of INCOME & LIMIT by [Gender: 0=Female 1=Male]')*



*Figure 23. A scatter plot chart displaying the income & limit distribution by gender*

**vii.**

Generate a hexbin plot displaying the AGE and INCOME distribution.

*plt.hexbin(df\_cleaned2['AGE'], df\_cleaned2['INCOME'], gridsize = (10,5), cmap = plt.cm.Blues)*

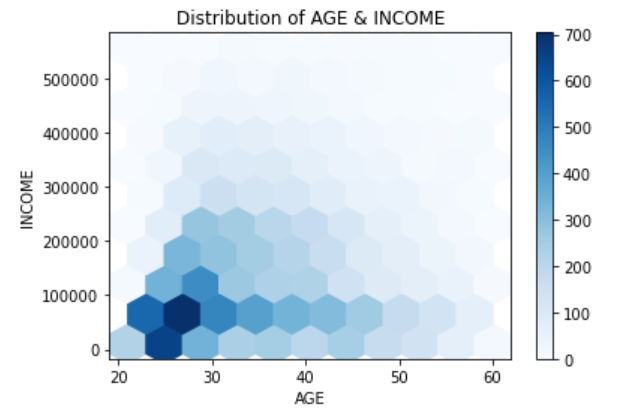
*plt.colorbar()*

*plt.xlabel ('AGE')*

*plt.ylabel ('INCOME')*

*plt.title('Distribution of AGE & INCOME')*

*plt.show()*



*Figure 24. A hexbin chart displaying the age & income distribution*

**Insights and Findings**

We can see from the above charts that the male population has received more postgraduate education than the female population (refer to Figure 17). It is also worth noting that the male population frequently provides favourable ratings for the services provided by financial institutions/banks (refer Figure 18). This male population is also said to be married (refer Figure 19), and their age group belongs to the mid to late 20s (refer Figure 20), as well as having a higher savings balance in their bank account than the female population (refer to Figure 21).

On the contrary, we can see that the total LIMIT for the female population is higher than that of the male population, which earns $300,000 per year (refer to Figure 22). And those males who earn less than $100,000 per year are observed to be in their mid-twenties (refer to Figure 23).

**Qn4**

**i.**

To accomplish this, we must first import the following modules into Jupyter notebook.

*from sklearn import datasets, linear\_model*

*from sklearn.linear\_model import LinearRegression*

*import statsmodels.api as sm*

*from scipy import stats*

**ii.**

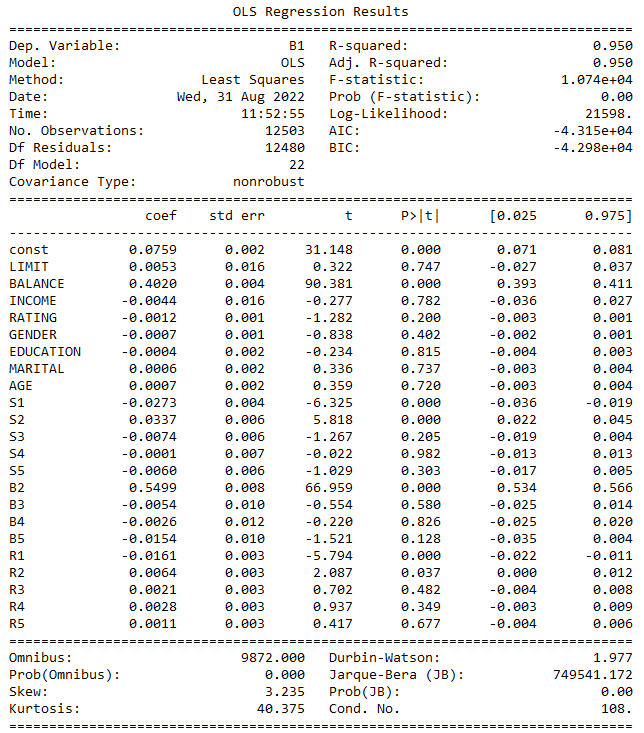
To begin, we must generate an Ordinary Least Squares (OLS) Regression Results table. Check the R-squared value of the results in the table and use the p-value (0.05) to select the significant variable to be used in the Linear Regression model. In this case, we got a good R-squared value of 95% (which means that 95% of the variations of the variables in the model can be used to explain the occurrence of the target variable B1). Therefore, the variables 'BALANCE,' 'S1,' 'S2,' 'B2,' and 'R1' will be chosen.

*X2 = sm.add\_constant(X)*

*est = sm.OLS(y, X2)*

*est2 = est.fit()*

*print(est2.summary())*



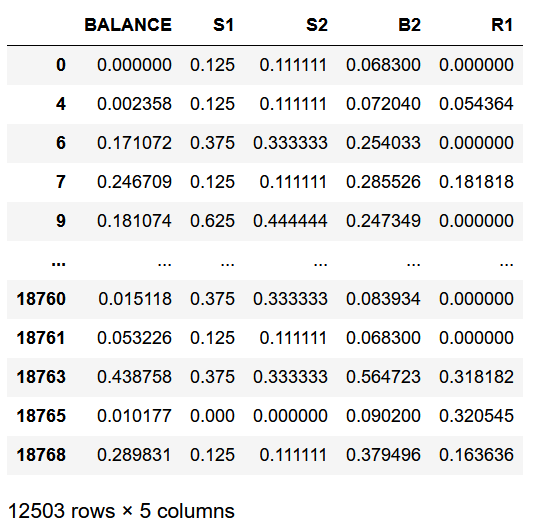
*Table 2. OLS Regression Results table for all variables*

**iii.**

Using the information from the OLS regression table, we can proceed to remove the insignificant variables and create a new data frame for the independent variables (x).

*X\_new= X.drop(['LIMIT','INCOME', 'RATING', 'GENDER', 'EDUCATION','MARITAL','AGE', 'S3', 'S4', 'S5', 'B3', 'B4', 'B5', 'R2', 'R3', 'R4', 'R5'],axis=1)*

*X\_new*



*Table 3. New data frame table for selected x variables (p-value <0.05)*

**iv.**

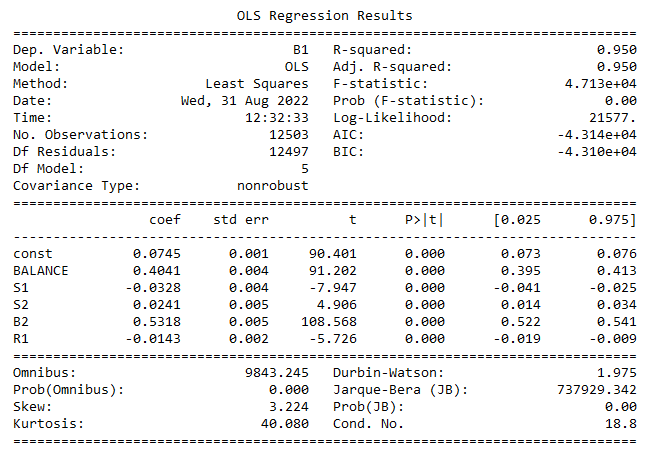
We can observe that the R-squared value remains constant at 95% when the significant variables are used in the model. It can be stated that for the occurrence of 'B1', we only need to use the independent variables 'BALANCE', 'S1', 'S2', 'B2', and 'R1'.

*X3 = sm.add\_constant(X\_new)*

*est = sm.OLS(y, X3)*

*est2 = est.fit()*

*print(est2.summary())*



*Table 4. OLS Regression Results table using selected x variables (p-value <0.05)*

**Qn5**

Assume a customer from the month of May exhibits the following characteristics.

* Balance = $1000
* S1 = -1 (performing a prompt payment in the month of May)
* S2 = -1 (performing a prompt payment in the month of Apr)
* B2 = $800 (amount billed in the month of Apr)
* R2 = $500 (repayment paid in the month of Apr)

Therefore, by using the linear regression model

*= 0 + 1 BALANCE + 2 S1+ 3 S2 + 4 B2 + 5 R2 +i*

with zero conditional mean assumption

*𝐸(i│BALANCE) = 0, 𝐸(i│S1) = 0, 𝐸(i│ S2) = 0, 𝐸(i│ B2) = 0, 𝐸(i│ R2) = 0*

We can estimate that the average billable amount for this customer in the month of May (B1) is around

= 0.0745 + (1000\*0.4041) + (-1\*-0.0328) + (-1\*0.0241) + (800\*0.5318) + (500\*-0.0143)

= $822.47 ≈ $822

**Qn6**

**i.**

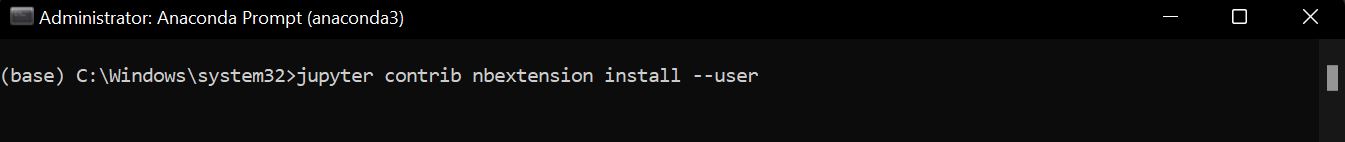
In order to accomplish this, we must first install the jupyter nbextensions configurator package in the note book.

*pip install jupyter\_contrib\_nbextensions*

**ii.**

Next, we need to type the following command into Anaconda Prompt environment.

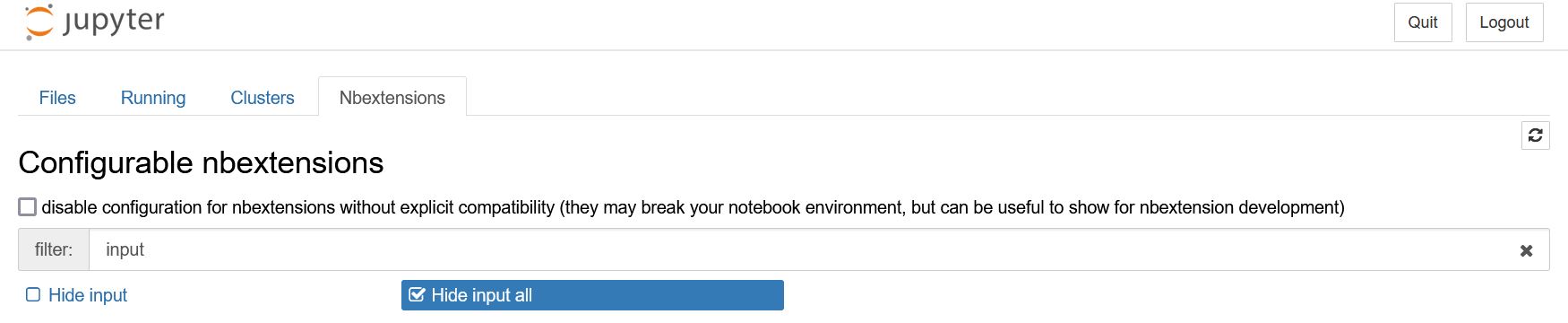
*jupyter contrib nbextension install –user*



*Figure 25. Anaconda prompt environment*

**iii.**

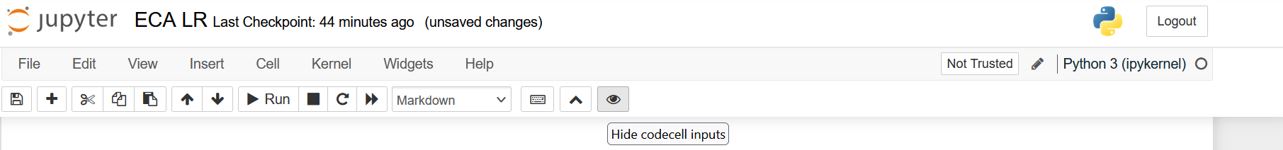
Once installed, we can search for and select the Hide input all option under the Nbextensions tab.



*Figure 26. Hide input all option under the Nbextensions tab*

**iv.**

Then we click the Hide codecell inputs button to hide all the codes in order to have a more well-documented and presented notebook.



*Figure 27. Location of the* *Hide codecell inputs button in notebook*

**References**

Siddall, M.E. (2002). *Parsimony Analysis. In: DeSalle, R., Giribet, G., Wheeler, W. (eds) Techniques in Molecular Systematics and Evolution*. Methods and Tools in Biosciences and Medicine. Birkhäuser, Basel, 31-54.

https://doi.org/10.1007/978-3-0348-8125-8\_2

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