

**ANL252**

**Python for Data Analytics**

# **End-of-Course Assessment**

# **July 2022 Presentation**

Dr Munish Kumar

|  |  |
| --- | --- |
| **Name:** | **Tng Ting Xu** |
| **PI Number:** | **K2181584** |
| **Date of Submission:** | **5 September 2022** |

**Question 1**

**Categorical Variables:**

1. Gender
2. Education
3. Marital
4. S(n)
5. Rating
6. ID

**Numeric Variables:**

1. Limit
2. Balance
3. Income
4. Age
5. B(n)
6. R(n)

**Question 2**

*# Import Libraries*

import pandas as pd

import numpy as np

from matplotlib import pyplot as plt

*# Reads data from file*

PATH = r"C:\Users\Tng Ting Xu\Desktop\SUSS\Y2S1\ANL252\ECA\ECA\_data.csv"

DF = pd.read\_csv(PATH)

*# To check the dataset*

display(DF)

**Data Pre-Processing Task 1**: **Removal of Null Values**

The first data pre-processing task involves checking for null values in the dataset. A total of 13 rows and 38 rows containing null values were identified under ‘EDUCATION’ and ‘MARITAL’ columns respectively. A check was done to find the percentage of null values under each respective column. This helps to ascertain if changing the null values would have a significant impact on the accuracy of the dataset. The percentage of null values were found to be relatively low – 6% for ‘EDUCATION’ and 20% for ‘MARITAL’. Since ‘EDUCATION’ and ‘MARITAL’ columns are considered categorical variables and given that the set of permitted values for categorical variables are usually fixed, we will be replacing the null values with the modal value.

*# Get the sum of null for columns with at least one null value*

DF[DF.columns[DF.isnull().any()]].isnull().sum()

*# Find out the percentage of null in each column*

percent\_missing = DF.isnull().sum() \* 100 / len(DF)

DF1 = pd.DataFrame({'percent\_missing': percent\_missing})

display(DF1)

*# Replace null values in the "EDUCATION" and "MARTIAL" columns with modal value since it’s a categorical numerical*

for column in ['EDUCATION', 'MARITAL']:

DF[column].fillna(DF[column].mode()[0], inplace=True)

*# Check if DF is free of null values*

DF[DF.columns[DF.isnull().any()]].isnull().sum()

**Data Pre-Processing Task 2**: **Removal of Abnormal Values**

The second pre-processing task that was done was to remove abnormal values found under the “AGE” column. A total of 10 abnormal values containing “-1” and “199” were identified. Since age cannot be negative and 199 is abnormal, the rows were dropped.

*# First, check for the number of unique values for each column*

DF.nunique()

*# Shows an overview of unique values in the "AGE" column*

np.unique(DF["AGE"])

*# Since age cannot be -1 & 199, drop these rows*

DF = DF.drop(DF.index[DF["AGE"].isin([-1, 199])])

*# Check if rows have been dropped*

np.unique(DF["AGE"])

**Data Pre-Processing Task 3**: **Converting Columns from Object to Int64**

The third pre-processing task involved converting columns "EDUCATION", "MARITAL" and "R3" from Object to Int64. Data under column “R3” was found to contain special characters such as “$” and “,” which prevented the conversion from Object to Int64. To rectify this, the special characters had to be dropped before conversion.

*# Check the data types*

DF.dtypes

*# Check for special characters*

np.unique(DF["R3"])

*# Pass them to df.replace() to remove “$” and “,” sign*

DF['R3'] = DF['R3'].replace({'\$': '', ',': ''}, regex=True)

*# Run a check again*

np.unique(DF["R3"])

*# Change columns "EDUCATION", "MARITAL" and "R3" from object to int64*

DF['EDUCATION'] = DF['EDUCATION'].astype('Int64')

DF['MARITAL'] = DF['MARITAL'].astype('Int64')

DF['R3'] = DF['R3'].astype('float').astype('Int64')

*# Run a check*

DF.dtypes

**Data Pre-Processing Task 4: Removing Duplicated Rows**

The fourth pre-process task involved the removal of duplicated rows that were detected in the dataset. By using the ID as a unique identifier, 3 sets of data were identified to be duplicated. The duplicated rows were dropped, leaving only the first row. This helps to maintain data accuracy as duplication of data can affect overall results.

*# To check for duplicates*

DF\_duplicates = DF.duplicated().sum()

print('Number of Duplicates')

print(DF\_duplicates)

*# Use the DataFrame.duplicated() method to check for duplicate values using 'ID' column as unique identifier*

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

*# Drop duplicate rows*

DF.drop\_duplicates(inplace = True)

*# To check if duplicates have been removed*

DF\_duplicates = DF.duplicated().sum()

print('Number of Duplicates')

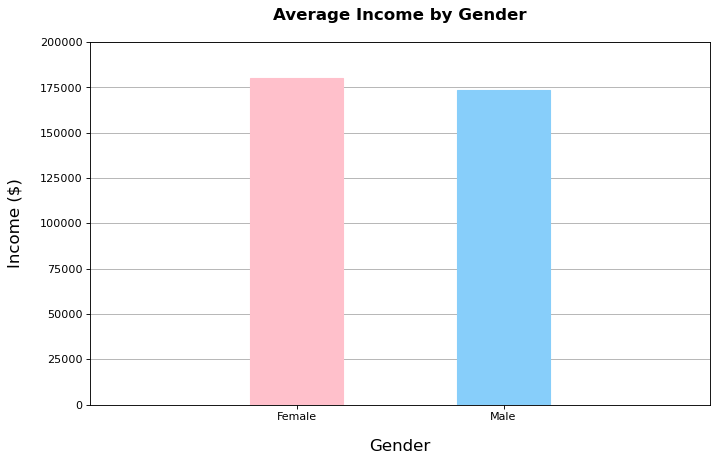
print(DF\_duplicates)

**Question 3**

*# Copy over and rename dataframe*

DF\_CLEAN = DF.copy()

**Relevant Insight 1: Income by Gender**

****

Through categorisation of income by gender, we are able to see that Females have a slightly higher average income as compared to Males.

**Codes in Text for Insight 1:**

*# Define the plot settings*

COL = "INCOME"

TITLE = "Income by Gender"

Y\_LABEL = "Income"

X\_LABEL = "Gender"

gen\_dict = { "0":"Male", "1":"Female"}

df\_dict = {

"Gender":DF\_CLEAN["GENDER"].unique(),

"Income":[]

}

tmp = []

*# Prepare data for plotting*

for gen in df\_dict["Gender"]:

df\_gen = DF\_CLEAN[DF\_CLEAN["GENDER"]==gen]

df\_dict["Income"] = df\_dict["Income"] + [round(df\_gen["INCOME"].mean(), 2)]

tmp.append(gen\_dict[str(gen)])

df\_dict["Gender"] = tmp

*# Plot & create figure*

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

*# Plot the bars*

plt1 = plt.bar(df\_dict["Gender"], df\_dict["Income"], width = 0.45, zorder=3)

*# Change colours*

plt1[0].set\_color('pink')

plt1[1].set\_color('lightskyblue')

*# Change the current limits of both x and y axis*

plt.ylim(0, 200000)

plt.xlim(-1, 2)

*# Plot the display settings for bar plots*

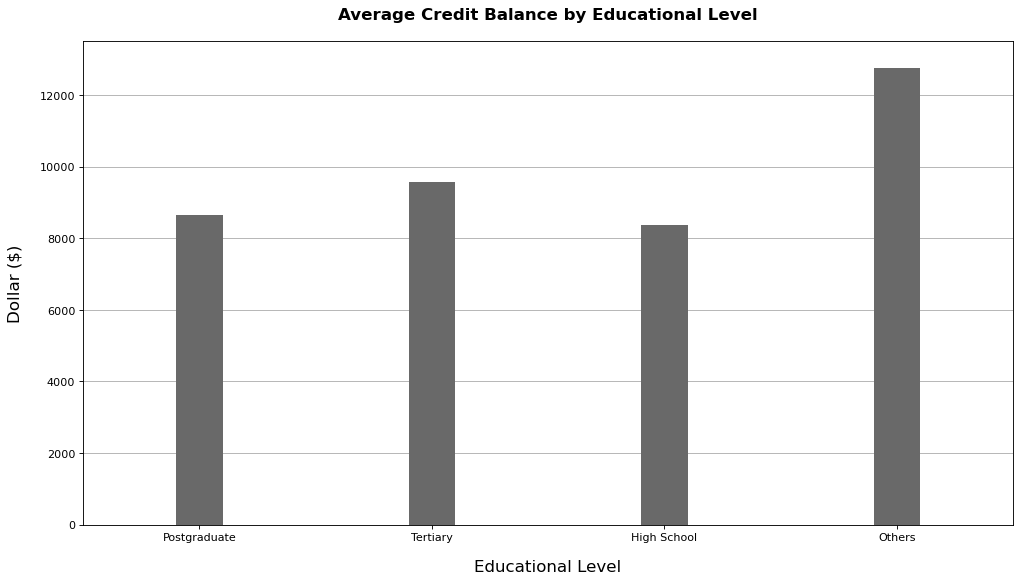
plt.ylabel(Y\_LABEL, labelpad=15, fontsize=15)

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

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

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

plt.show()

**Relevant Insight 2: Average Credit Balance by Educational Level**

|  |  |  |
| --- | --- | --- |
|  | **Education** | **Average Balance** |
| 0 | Postgraduate | 8662.20 |
| 1 | Tertiary | 9581.33 |
| 2 | High School | 8366.04 |
| 3 | Others | 12756.65 |

Through categorisation of current credit balance by level of education, we are able to see that customers that are still in “High School” have the lowest average credit balance while customers that are categorised as “Others” have the highest credit balance. The low credit balance of “High School” customers can be attributed to their age and lack of income, Conversely, the high credit balance of customers categorised as “Others” can be attributed to a longer time spent working instead of pursing an education which results in a longer accumulation of wealth.

**Codes in Text for Insight 2:**

*# Create title and labels*

TITLE = "Average Balance by Education"

Y\_LABEL = "Dollar ($)"

X\_LABEL = "Educational Level"

*# Create dict*

df\_dict = {

"Education": DF\_CLEAN["EDUCATION"].unique(),

"Average\_Balance":[]

}

edu\_dict = {

"0":"Others",

"1":"Postgraduate",

"2":"Tertiary",

"3":"High School"

}

tmp = []

for edu in df\_dict["Education"]:

df\_dict["Average\_Balance"] = df\_dict["Average\_Balance"] + [round(DF\_CLEAN[DF\_CLEAN["EDUCATION"]==edu]["BALANCE"].mean(),2)]

tmp.append(edu\_dict[str(edu)])

df\_dict["Education"] = tmp

*# Create figure*

ax = plt.figure(figsize=(15, 8), dpi=80)

*# Plot bars*

plt1 = plt.bar(df\_dict["Education"], df\_dict["Average\_Balance"], width = 0.2, zorder=3, color='dimgrey')

*# Change the current limits of both x and y axis*

plt.ylim(0, 13500)

plt.xlim(-0.5, 3.5)

*# Plot the 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*

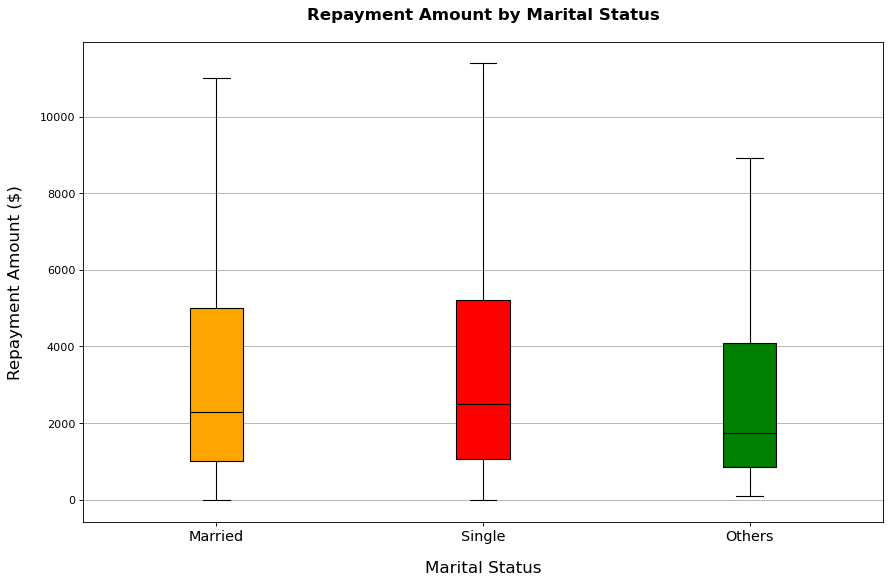
plt.title(label=TITLE, pad=20, fontsize=15, weight='bold')

plt.show()

*# Display Data*

pd.DataFrame(df\_dict)

**Relevant Insight 3: Repayment Amount by Marital Status**



Through categorisation of marital status, we are able to see that customers that are categorised as “Single” have a higher median expenditure amount. Conversely, customers that are categorised as “Others” have the lowest median expenditure amount.

**Codes in Text for Insight 3:**

*# Define plot settings*

TITLE = "Repayment Amount by Marital Status"

Y\_LABEL = "Repayment Amount ($)"

X\_LABEL = "Marital Status"

legend\_dict = {

'0':"Others",

'1':"Single",

'2':"Married"

}

df\_dict = {

"Marital\_status":DF\_CLEAN["MARITAL"].unique(),

"Repayment":[],

"Mean":[]

}

legend = []

for idx,mar in enumerate(df\_dict["Marital\_status"]):

tmp\_df = DF\_CLEAN[DF\_CLEAN["MARITAL"]==mar]

for i in range(1,6): tmp\_list = list(tmp\_df[~tmp\_df['R'+str(i)].isin([-1,0])]['R'+str(i)])

df\_dict["Mean"] = df\_dict["Mean"] + [np.array(tmp\_list).mean()]

legend.append(legend\_dict[str(mar)])

if idx == 0: tmp = [tmp\_list]

else: tmp.append(tmp\_list)

df\_dict["repayment"] = tmp

data = df\_dict["repayment"]

df\_dict["Marital\_status"] = legend

*# 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 = [1,2,3], y = df\_dict["Mean"],

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

*# Boxplot settings*

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

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

elif i == 1: bp["boxes"][i].set\_facecolor("red")

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

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

*# Chart settings*

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

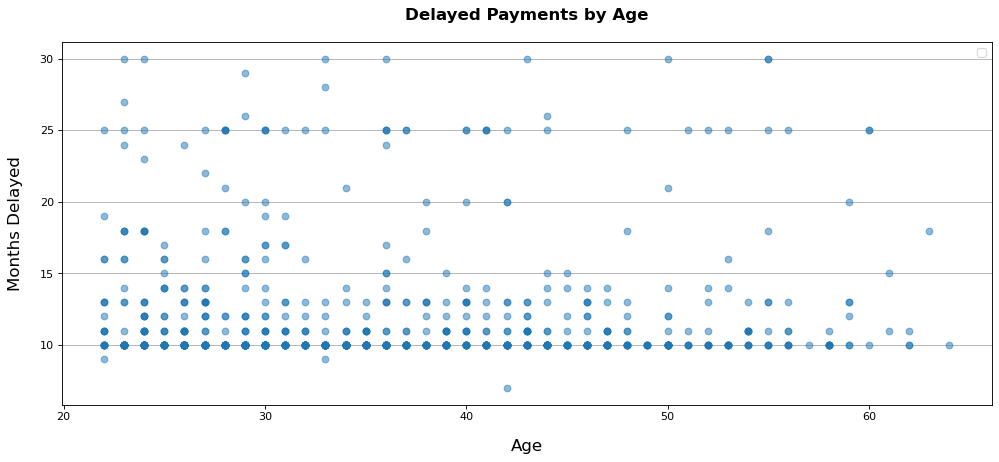
plt.ylabel(Y\_LABEL,labelpad=15,fontsize=15)

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

plt.xticks([1,2,3],labels=df\_dict["Marital\_status"],fontsize=13)

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

plt.show()

**Relevant Insight 4: Delayed Payments by Age**

Through categorisation of age, we are able to compare delayed repayments across the different age groups. We can see that younger customers have a higher tendency to delay payments as compared to older customers. Additionally, the majority of customers also delay payments by a minimum of 10 months.

**Codes in Text for Insight 4:**

*# Define plot settings*

TITLE = "Delayed Payments by Age"

Y\_LABEL = "Months Delayed"

X\_LABEL = "Age"

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"]

plt.figure(figsize=(15, 6), dpi=80)

X = df\_interested["AGE"]

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

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

*# Plots settings*

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

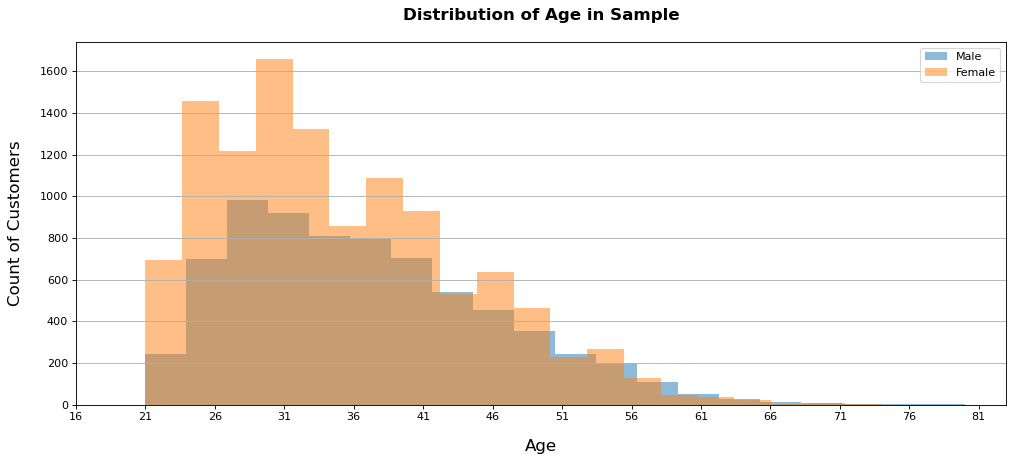
plt.xlabel(X\_LABEL,labelpad=15,fontsize=15)

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

plt.legend()

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

plt.show()

**Relevant Insight 5:**

Through the use of a histogram, we are to observe the age spread of the credit facility’s customers grouped by gender. Firstly, younger customers make up a larger proportion of the total customers for both genders. Secondly, the number of female customers outnumber the number of male customers. This is especially evident with its younger customers.

**Codes in Text for Insight 5:**

*# Define plot settings*

BINS = 20

COL = "AGE"

TITLE = "Distribution of Age in Sample (Bin=20)"

Y\_LABEL = "Count of Customers"

X\_LABEL = "Age"

gen\_dict = { "0":"Male", "1":"Female"}

*# Create figures and plot histograms*

plt.figure(figsize=(15, 6), dpi=80)

for gen in [0,1]:

df\_tmp = DF\_CLEAN[DF\_CLEAN["GENDER"]==gen]

DAT = df\_tmp[COL]

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

*# Plot 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)

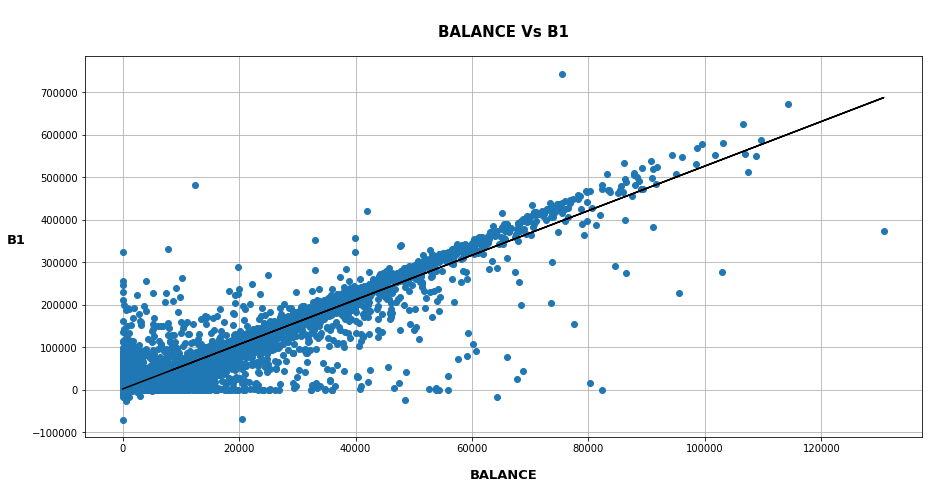
plt.xticks([i for i in range(int(DF\_CLEAN[COL].min())-5,int(DF\_CLEAN[COL].max()+5),5)])

plt.legend()

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

plt.show()

**Question 4**

****



Firstly, we will be creating a correlation matrix to determine which variables have high correlation and will be useful in the regression analysis. The top 3 variables observed to have the highest correlation are – “BALANCE”, “INCOME” and “LIMIT”.

The next step would be to plot the linear regression for any of the high correlation columns and B1.

**Codes for Linear Regression Modelling (Balance vs B1 / Gender vs B1):**

*# Creation of Correlation Matrix*

cor\_mat = DF\_CLEAN.corr()

*# Correlation rankings*

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

*# Full Correlation rankings*

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

*# import LinearRegression library*

from sklearn.linear\_model import LinearRegression

*# Preparation of data to be fed to the linear regression model*

# Y values: B1

# X values: BALANCE

while True:

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

x\_col = input("Please Enter Column to perform linear regression with B1: ") # Allows user input to 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 = DF\_CLEAN[[x\_col]]

Y = DF\_CLEAN['B1']

*# Perform linear regression*

# Creates the base of the linear regression model

linear\_regressor = LinearRegression()

*# Feeds data into model & prediction*

linear\_regressor.fit(X, Y)

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

*# Plot the regression model and its variables*

plt.figure(figsize=[15, 7])

*# Plot data from desired column against B1*

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

*# Plot the predicted values from regression model*

plt.plot(X, Y\_pred, color='black',zorder=3)

*# Plot settings*

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)

*# Plot the graph*

plt.show()

**Question 5**

Accuracy (R2): 0.904

Linear Regression Equation: Y = 5.24 X + 2171.31

An R2 of 0.904 indicates that 90.4% of the variance in Y (B1) is explained by X (Balance). This shows that there is a strong correlation between customers’ billable amount in nth month and the customer’s current credit balance.

**Codes to Linear Regression Equation:**

*# Gets parameters from linear regression model and displays 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 Observed:**

1. Customers with a higher bank balance have the tendency to pay off more of what they owe within the first bill. This can be seen by the number of data points recorded above the best fit line as the balance amount increases.
2. Females are observed to pay off more of what they owe within the first bill as compared to Males. This can be seen from the “Gender vs B1” regression model where there are a higher number of data points recorded under (1) which represents the Female customers.

1. Based on the correlation rankings, “Gender”, “Credit Rating”, “Marital” and “Education” were observed to have negative correlation with B1. This indicates that the first bill amount (B1) is unlikely to be determined by these variables and should not be used in the regression model.