

**ANL252**

**Python for Data Analytics**

**End Course Assignment (ECA)**

**July 2022 Presentation**

**Name:**

Pay Jiarre Lyn

**PI Number:**

Y1981896

**Submission Date: 4 Sep 2022**

**Question 1**

|  |  |
| --- | --- |
| **ID** | Typeless |
| **Limit** | Continuous |
| **Balance** | Continuous |
| **Income** | Continuous |
| **Gender** | Categorical - Flag |
| **Education** | Categorical - Nominal |
| **Marital** | Categorical - Nominal |
| **Age** | Continuous |
| **S(n)** | Categorical - Nominal |
| **B(n)** | Continuous |
| **R(n)** | Continuous |
| **Rating** | Categorical - Flag |

To begin, we will load libraries and modules to utilize functions that are not inbuilt in Python. Using modules to use existing code allows us to make our programmes more stable and effective.

**# Importing Libraries**

import pandas as pd

import numpy as np

from datetime import datetime

import datetime

import math

**# Plotting modules**

import plotly.express as px

from plotly.subplots import make\_subplots

import plotly.graph\_objects as go

**# Importing module**

import statsmodels.api as sm

from sklearn.linear\_model import LinearRegression

from sklearn.model\_selection import train\_test\_split

from sklearn.metrics import r2\_score

from sklearn.metrics import mean\_squared\_error

from sklearn.metrics import mean\_absolute\_error

pd.set\_option('display.max\_columns', 500)

Secondly, we will load our dataset into the data frame and understand more about the current data through summary statistics.

**# Reading File to Pandas DataFrame**

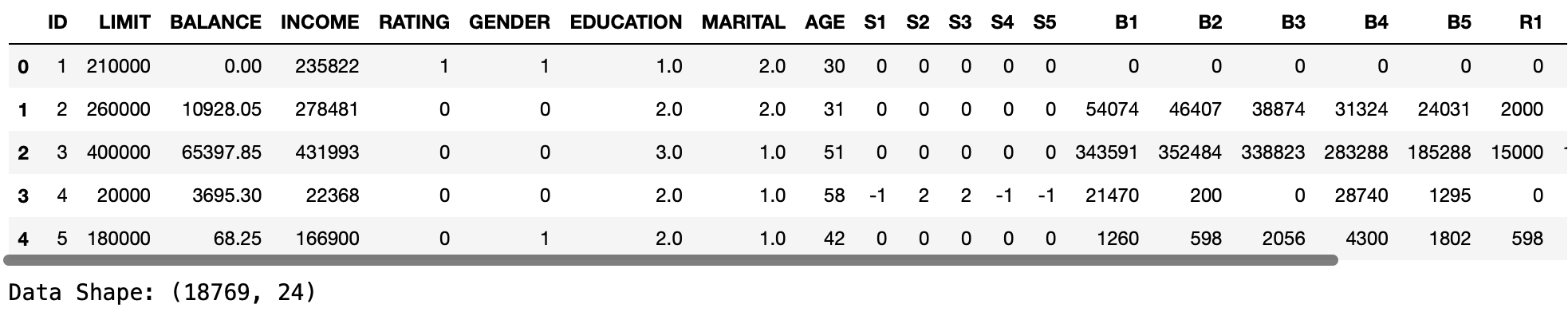
credit\_df = pd.read\_csv(r'ECA\_data.csv')

**# Show the first 5 rows of the dataframe**

display(credit\_df.head())

**# Print the shape of the data**

print(f'Data Shape: {credit\_df.shape}')



**# Description of data in the dataframe**

display(credit\_df.describe().round(2))

Table

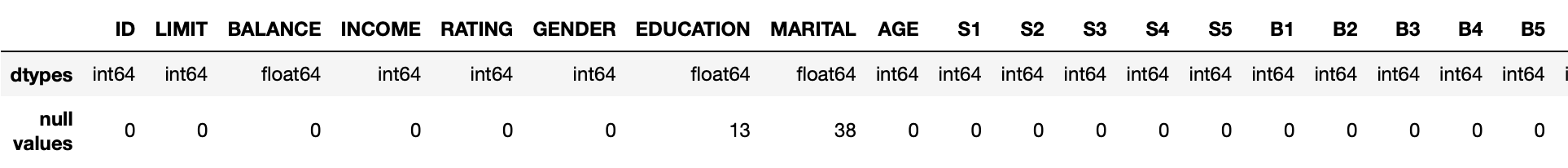
Description automatically generated

From the above, one significant error that we can spot is that age shows erroneous data with a minimum of -1 and a maximum of 199, which should be excluded during the preprocessing task.

**# Description of the data types and the number of null values**

display(pd.concat([credit\_df.dtypes.to\_frame(name = 'dtypes').T,

credit\_df.isna().sum().to\_frame(name = 'null values').T]))



**Question 2**

A crucial part of data preparation is the transformation of data into a manner that can be treated more effectively and efficiently in data mining, deep learning, and other machine learning tasks. The fundamental purpose of preprocessing is to validate the data's quality.

1. Data Inconsistency: Error in Data Format

From the figure below, we can see that the data format for R(n) repayment amount is inconsistent, with comparable data throughout "R1, R2, R4, and R5", but distinct inputs for "R3". There is no stated monetary unit or comma delimiter for the numerical inputs. However, "R3" data include both the monetary unit and a comma to denote the separation of dollars. This can lead to erroneous and incomprehensible data, thus we will correct the wrong data type.

**# R3 data type shows an object, hence, we can filter out to see the records with the wrong data type**

credit\_df[pd.isna(pd.to\_numeric(credit\_df['R3'], errors='coerce'))]

Table

Description automatically generated

**# Replace '$' and ',' in R3 column**

credit\_df['R3'] = credit\_df['R3'].replace(['\$', ','], '', regex=True)

**# Convert R3 datatype to an integer**

credit\_df['R3'] = credit\_df['R3'].astype(int)

**# Print results to see the change**

credit\_df.iloc[1800:1810]

Table

Description automatically generated

1. Replacing Missing Values

The practice of removing incorrect, incomplete, or inaccurate data from datasets and restoring any missing information is known as data cleaning. In this instance, missing values are represented as null values 'NaN' Here, we substitute missing values with mode, the most common value in the entire feature column. It is recommended to consider replacement missing values with mode values when the data is skewed.

**# Display all missing values**

credit\_df[credit\_df.isna().any(axis = 1)]

Table

Description automatically generated

**# Replacing null values**

display(credit\_df['EDUCATION'].value\_counts())

display(credit\_df['GENDER'].value\_counts())

Text

Description automatically generated

**# Fillna with mode**

credit\_df['EDUCATION']

=. credit\_df['EDUCATION'].fillna(credit\_df['EDUCATION'].mode()[0])

credit\_df['MARITAL'] = credit\_df['MARITAL'].fillna(credit\_df['MARITAL'].mode()[0])

1. Binning of Age Variables

Binning of variables is used to treat or smooth down noisy data. It is an example of data discretisation to transform numeric age attributes into categorical attributes characterized by discontinuous ranges called bins (Lee, 2021).

Age will be classified based on the following codes: "Age 30," "Age 30-59," and "Age 60+." The resulting array will be allocated to a new census variable called "AGEGROUP". First, we will take the age variable from the census DataFrame and transform it into a NumPy array before assigning it to the parameter x. Then, we exclude the right edges by setting right = False and specify them in bins for each of the three bins in the tuple (0, 30, 60, 100). This configuration allows us to employ bin edges of 30 instead of 29 and 60 instead of 59. Since the cut() procedure does not include the highest option, we have to manually make the rightmost edge of the last group greater than the DataFrame's upper age limit. The final step is to specify the same labels for our bins as a list for the parameter labels.

**# Cut to segment age into bins to transform continuous variable to categorical variable**

credit\_df['AGEGROUP'] = pd.cut(x = np.array(credit\_df['AGE']), bins = (0, 30, 60, 100), right = False, labels = ['Age < 30', 'Age 30 - 59', 'Age 60+'])

display(credit\_df)

Table

Description automatically generated

After binning the variables, we noticed that there are some null values under "AGEGROUP", where we noticed that there are anomalies in the data where the minimum age is "-1" and the maximum age is "199". These variables are considered outliers as they are out of the age range. Thus, such variables should be excluded.

**# Summary Statistics of Age**

credit\_df['AGE'].describe()

Text

Description automatically generated

**# Display Age below 1 and above 100**

display(credit\_df[(credit\_df['AGE'] < 0) | (credit\_df['AGE'] > 100)])

A picture containing table

Description automatically generated

**# Filtering out Age below 1 and above 100**

credit\_df = credit\_df[(credit\_df['AGE'] > 0) | (credit\_df['AGE'] < 100)]

1. Creating Dummy Variables for Regression Model

Following this, we will transform the categorical variables in the DataFrame into dummy variables during pre-processing because they require special handling compared to continuous or ordinal variables. (Wu, 2022). Dummy variables are purely numerical variables that can only take on the values 0 or 1. A dummy variable can have the value of 1 if the observation falls into a specific group, or 0 if it does not.

When training linear regression, the predictor variables should be quantitative or numeric. However, if one or more predictor variables are categorical, we need a technique to incorporate this categorical variable into the linear regression model (Kumar, 2022). In this case, dummy variables come into play.

**# Categorising Education Variables**

def education\_categorisation(x):

if x == 0:

return 'OTHERS'

elif x == 1:

return 'POSTGRADUATE'

elif x == 2:

return 'TERTIARY'

else:

return 'HIGH\_SCHOOL'

**# Categorisation of 'MARITAL','GENDER','EDUCATION' & 'RATING' Variables**

credit\_df['MARITAL'] = credit\_df['MARITAL'].apply(lambda x: 'Others' if x == 0 else('SINGLE' if x == 1 else 'MARRIED'))

credit\_df['GENDER'] = credit\_df['GENDER'].apply(lambda x: 'MALE' if x == 0 else 'FEMALE')

credit\_df['EDUCATION'] = credit\_df['EDUCATION'].apply(education\_categorisation)

credit\_df['RATING'] = credit\_df['RATING'].apply(lambda x: 'GOOD' if x == 0 else 'BAD')

credit\_df.head()Table

Description automatically generated with medium confidence

**# Preparation for Regression Model**

model\_df = credit\_df.copy()

model\_df = pd.get\_dummies(model\_df)

model\_df.head()

A picture containing graphical user interface

Description automatically generated

Table

Description automatically generated

Table

Description automatically generated

From the above, we can see that the categorical variables have been replaced by the dummy variables and can be employed in the linear regression models.

**Question 3**

Chart 1: Comparison of different age groups Credit Limit, Balance and Income Levels

**# Create dataframe for plotting**

description\_df = credit\_df[['AGE', 'LIMIT', 'BALANCE', 'INCOME', 'AGEGROUP']].groupby('AGEGROUP').mean()

description\_df[['LIMIT', 'BALANCE', 'INCOME']] = description\_df[['LIMIT', 'BALANCE', 'INCOME']].round(decimals = 2)

**# Plotting bargraph with plotly, setting x-axis as age groups, y-axis as limit, balance and income values**

fig = make\_subplots(rows=1, cols=3,

subplot\_titles=('LIMIT', 'BALANCE', 'INCOME'))

fig.add\_trace(go.Bar(x = description\_df.BALANCE.keys(), y = description\_df.BALANCE.values,

text = description\_df.BALANCE.values,

textposition = 'outside'),

row = 1, col = 1)

fig.add\_trace(go.Bar(x = description\_df.LIMIT.keys(), y = description\_df.LIMIT.values,

text = df\_description.LIMIT.values,

textposition = 'outside'),

row = 1, col = 2)

fig.add\_trace(go.Bar(x = description\_df.INCOME.keys(), y = description\_df.INCOME.values,

text = description\_df.INCOME.values,

textposition = 'outside'),

row = 1, col = 3)

**# Updating chart labels**

fig.update\_layout(barmode = "relative", title = "Comparison of different age groups Credit Limit, Balance and Income Levels")

**# Update the x-axis, y-axis and title label**

fig.update\_layout(height=400, width=800,

xaxis={'showticklabels': True}, showlegend = False,

barmode='group')

# **Show graph**

fig.show()

Chart, bar chart

Description automatically generated

From the graph above, strong positive correlations are anticipated between AGEGROUP, Limit, Balance, and Income Levels as age increases. By this age, customers will leave their working careers and enter retirement, thus their account limits, balances, and income would have reached their maximum amounts.

For example, the overall limit will increase with age as a result of longer account history. As customers age, they often make more money as they gather more job experiences. Besides, side investments will also increase their income levels. In addition, the credit balance will increase as a result of the accumulation of repayments on a particular credit card account or loan over a predetermined period. Nonetheless, this is a pessimistic prognosis because the larger the credit card balances, the greater the interest rates they have to pay.

Chart 2: Percentage count of Rating by Education Levels

**# To show Per Percentage Count**

description\_df = credit\_df.groupby(['EDUCATION','RATING']).size().reset\_index()

description\_df['PERCENTAGE'] = credit\_df.groupby(['EDUCATION','RATING']).size().groupby(level=0).apply(lambda x:100 \* x/float(x.sum())).values

**# Rename columns**

description\_df.columns= ['EDUCATION','RATING', 'COUNTS', 'PERCENTAGE']

description\_df['PERCENTAGE'] = description\_df['PERCENTAGE'].astype(float, errors = 'raise')

description\_df.round(2)

**Table

Description automatically generated**

**# Plotting bargraph with plotly, setting x-axis as Education, y-axis as Percentage, categories by Rating**

fig = px.bar(description\_df, x = "EDUCATION", y = "PERCENTAGE",color = 'RATING',

title = "Bar Plot",

template = "plotly\_white",

height = 600,

text = description\_df['PERCENTAGE'].round(2))

**# Updating chart labels**

fig.update\_layout(barmode = "relative", title = "Percentage count of Rating by Education Levels")

**# Formatting layout**

fig.update\_traces(textposition = 'inside')

fig.update\_layout(uniformtext\_minsize = 8, uniformtext\_mode = 'hide')

fig.update\_layout(plot\_bgcolor = 'white')

fig.update\_yaxes(showline = False, showgrid = False)

fig.update\_xaxes(showline = False, showgrid = False)

**#Show graph**

fig.show()

**Chart, bar chart

Description automatically generated**

From the graph above, we noticed that bad credit ratings for customers in High\_School are the highest, followed by Tertiary and Postgraduates. A poor credit rating implies that a borrower may have difficulty making payments, whereas a good credit rating shows that the borrower will likely return the loan in full without any problems (Kagan, 2022).

In this case, bad credit ratings from customers in High\_Schools may be due to their young age and may not have enough prior experiences when dealing with loans.

Besides, the assumption was that 'Others' here refers to Doctorate (Ph.D.) and Master’s Degree. This group of customers, they have the least bad ratings and most good ratings. Those with a greater level of education earn more money, have more savings, and are generally happier. Thus, having higher rates of school attainment can have better loan repayment and better credit ratings.

Chart 3: Relationship between Customers' Income Level, Limit and Ratings

**# Plotting scatterplot with plotly, setting x-axis as Limit, y-axis as Income, categories by Rating**

fig = px.scatter(credit\_df, x = "LIMIT", y = "INCOME", color = "RATING",

size = 'INCOME', hover\_data = ['RATING'], trendline = 'ols')

**# Updating chart labels**

fig.update\_layout(barmode = "relative", title = "Relationship between Customers' Income Level, Limit and Ratings")

**#Show graph**

fig.show()

**Chart

Description automatically generated**

The scatterplot above shows a linear relationship between Customers’ income level and total limit. As one income rises, its total limit will also increase. High-risk customers typically receive lower credit limits due to their lack of money and capability to repay the loan. Low-risk customers with a higher income often receive higher credit limits, allowing them greater spending flexibility.

In addition, we classified the Customers' credit in the analysis, with a blue input indicating a poor rating and a red input indicating a good rating. As income increases, we can also assume that the vast majority of clients have excellent credit ratings. However, there is a limited group of buyers with poor credit scores in the $550K to $750K limit bracket. This may be due to unforeseen monetary struggles, retrenchment, or too much debt.

Chart 4: Percentage count of the mode of repayment status by age category

**# To show the mode of the repayment status of the various age groups**

description\_df = credit\_df.loc[:, ['S1', 'S2', 'S3', 'S4', 'S5']]

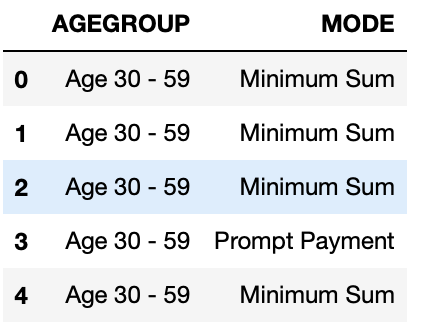
description\_df['MODE'] = description\_df.apply(lambda row: row.mode(), axis=1)[0]

description\_df['MODE'] = description\_df['MODE'].apply(lambda x: 'Prompt Payment' if x < 0 else ('Minimum Sum' if x == 0 else 'Delayed Payment'))

description\_df['AGEGROUP'] = credit\_df.loc[:, 'AGEGROUP']

description\_df = description\_df[['AGEGROUP', 'MODE']]

description\_df.head()

****

**# To show Per Percentage Count**

df\_stack = description\_df.groupby(['AGEGROUP','MODE']).size().reset\_index()

df\_stack['Percentage'] = description\_df.groupby(['AGEGROUP','MODE']).size().groupby(level=0).apply(lambda x:100 \* x/float(x.sum())).values

df\_stack.columns = ['AGEGROUP','MODE', 'COUNTS', 'PERCENTAGE']

df\_stack['PERCENTAGE'] = df\_stack['PERCENTAGE'].astype(float, errors = 'raise')

df\_stack.round(2)

fig = px.bar(df\_stack, x = "AGEGROUP", y = "PERCENTAGE",color = 'MODE',

title = "Bar Plot",

template = "plotly\_white",

height = 600,

text = df\_stack['PERCENTAGE'].round(2))

**# Updating chart labels**

fig.update\_layout(barmode = "relative", title = "Percentage count of the mode of repayment status by age category")

**# Formatting layout**

fig.update\_traces(textposition = 'inside')

fig.update\_layout(uniformtext\_minsize = 8, uniformtext\_mode = 'hide')

fig.update\_layout(plot\_bgcolor = 'white')

fig.update\_yaxes(showline = False, showgrid = False)

fig.update\_xaxes(showline = False, showgrid = False)

**#Show graph**

fig.show()

**Chart, bar chart

Description automatically generated**

The stacked bar chart above suggests that the majority of the customers from the three age groups usually pay the minimum sum of their repayment amounts.

AGEGROUP below 30 years old has the least prompt payment (15.68%) and the most delayed payment (10.02%). AGEGROUP between 30 and 59 years old has the highest prompt payment (20.48%) and age above 60 years old have the next highest rating (18.59%). However, AGEGROUP between 30 and 59 years old have higher delayed payments (8.61%) than those aged above 60 (8.04%).

Although the credit card minimum payment requirement is meant to protect customers, making just the minimum payments every month is a dangerous practice. For one, it can cause them to carry a balance for an extremely long time as well as harm their credit ratings. Only making the minimum payments can result in them having to pay their balance for several years or more.

Chart 5: Customer Repayment Status S(1) VS Billable Amount B(1) in most recent month

**# Constructing Boxplot graph and updating format**

boxplot\_df = credit\_df.loc[:, ['S1', 'B1']]

boxplot\_df['S1'] = boxplot\_df['S1'].apply(lambda x: 'Prompt Payment' if x < 0 else ('Minimum Sum' if x == 0 else 'Delayed Payment'))

**# Updating chart labels**

fig = px.box(boxplot\_df, x="B1", y="S1", orientation='h')

fig.update\_traces(boxpoints=False)

fig.update\_layout(barmode = "relative", title = "Customer Repayment Status VS Billable Amount in most recent month¶")

**# Display Graph**

fig.show()

**Chart, box and whisker chart

Description automatically generated**

**# Summary Statistics**

boxplot\_df.groupby('S1')['B1'].describe()

**Table

Description automatically generated**

The boxplot above compares the Customer Repayment Status and their Billable amount in the most recent month. From the graph above, we can deduce that in 'Delayed Payment', 'Minimum Sum', and 'Prompt Payment', about 75% billable amount is lesser than $62995, $83687.25, and $7141.50 respectively.

The least billable amount falls in 'Prompt Payment' under the third percentile as customers can pay off their credit payments easier when the amount is lower. The highest amount was seen in 'Minimum Sum' due to the accumulated repayments and interests across the recent months as compared to 'Delayed Payment'. From this, we can infer that as the billable amount is higher, customers may delay their repayment or even just pay the minimum sum first for the payment to be deemed "on time" to avoid late fees and other penalties.

In conclusion, from the various graphs above, the credit facility can draw significant insights that may not be visible previously. The use of data visualisation, provides users with a clear understanding of the information's significance, making it easier to discover trends, patterns, and outliers in such vast amounts of data as a whole.

**Question 4**

To execute linear regression, we will first determine the connection between the independent and dependent variables using correlation analysis. A correlation coefficient is a statistical measure of the extent to which changes in the value of the independent variables predict changes in the values of the dependent variable 'B1'. A positive correlation suggests that both variables are moving in the same direction, whereas a negative correlation shows that they are moving in opposite directions. A correlation close to 1 suggests a stronger, nearly perfect positive relationship (Cherry, 2022).

**# Perform correlation**

correlation\_df = credit\_df.loc[:, ['S1', 'S2', 'S3', 'S4', 'S5', 'B1', 'B2', 'B3', 'B4', 'B5', 'R1', 'R2', 'R3', 'R4','R5']]

correlation\_df.corr(method = 'pearson')['B1'].reset\_index(name = 'Correlation against B1')

**Table

Description automatically generated**

All of the independent variables exhibit a positive association, as seen above. However, we also observed multicollinearity when independent variables in the regression model have a strong correlation with the dependent variable. Variables 'B2', 'B3', 'B4', and 'B5' should be eliminated because a change in one variable would trigger a change in another variable, causing the model results to fluctuate dramatically. This makes the model difficult to comprehend and generates an overfitting issue. Besides, we will also drop 'AGE' as the transformed variable 'AGEGROUP' will be used in the regression model. 'ID' will be removed as it yields no significant results to the analysis.

**# Drop those that have very high correlation**

model\_df.drop(columns = ['B2', 'B3', 'B4', 'B5', 'AGE', 'ID'], inplace = True)

model\_df.head()

**A picture containing application

Description automatically generated**

After which, we will transform the numerical value of 'S(n)' to categorical variables and apply the function the respective columns respectively. We will then convert them to dummy variables for regression analysis.

**# Transforming Numerical variables of Repayment Status to Categorical variables**

**def repayment\_categorisation(x):**

if x == -1:

return 'Prompt Payment'

elif x == 0:

return 'Minimum Sum'

else:

return 'Delayed Payment'

**# Applying Functions to S(n) Columns**

column\_categorisation = ['S1', 'S2', 'S3', 'S4', 'S5']

for columns in column\_categorisation:

model\_df[columns] = model\_df[columns].apply(repayment\_categorisation)

model\_df

**Table

Description automatically generated with medium confidence**

**# Converting S(n) Columns to Dummy Variables**

model\_df = pd.get\_dummies(model\_df)

model\_df

**Table

Description automatically generated**

Lastly, as the continuous data have infinite values and do not adhere to the bell curve, we will log modify this data to make it "normal" so that the statistical analysis findings derived from this data are more accurate. This log modification can decrease the deviation of our initial data.

**# Logging of Variables**

log\_columns = ['LIMIT', 'BALANCE', 'INCOME', 'B1', 'R1', 'R2', 'R3', 'R4', 'R5']

model\_df[log\_columns] = np.log(model\_df[log\_columns])

**# Dropping infinite values from data frame**

model\_df = model\_df[~model\_df.isin([-np.inf, np.inf]).any(axis = 1)]

model\_df = model\_df.dropna()

**Question 5**

Multiple Linear Regression Analysis

We will perform multiple linear regression for the dataset due the multiple independent variables being used to predict the dependent variable. Both Scikit Learn and Statsmodels are Python libraries that permits users to investigate data, estimate statistical models, and conduct statistical tests.

1. Statsmodel

**# Dependent Variable = 'B1', Independent Variables = All other variables except'B1'**

x = model\_df.drop(columns = ['B1'])

y = model\_df['B1']

x = sm.add\_constant(x)

model = sm.OLS(y, x).fit()

predictions = model.predict(x)

print\_model = model.summary()

print(print\_model)

Table

Description automatically generated with medium confidence

The Dep variable here is 'B1' which is the only dependent variable in the data. Ordinary Least Square is the Model and Method utilized here, where the model attempts to create a linear expression for the dataset that minimizes the sum of residual squares. We have total 10136 observation and 28 model features.

R-Squared measures the strength of the model's correlation with our dependent variable. Greater R-Squared values suggest less deviations between observed data and actual values. A high R-square value of 0.831 indicates that the bulk of variable values lie along the regression line. After changing sample size and number of components, the modified r-square measures the variation. The high value of 0.83 for the modified r-square indicates that the variables fit the model well.

The log-likelihood number quantifies the model's fit with the given data. The greater the log-likelihood number, the better the model matches the data. In this instance, the log-likelihood value is -10405, suggesting that the model fits the data very poorly.

Table

Description automatically generated

Table

Description automatically generated

From the model, we can infer the regression equation as:

B1 (Y)  = 0.6159 – 0.0129 (LIMIT) + 0.7690 (BALANCE) + 0.081(INCOME) + 0.091(R1) … + 0.2207 (S5\_Minimum Sum)

The coef column contains the coefficients and intercept values for each independent variable. For example, for each additional unit of 'LIMIT,' 'B1' will decrease by 1.29%. and each additional unit of 'BALANCE,' 'B1' will increase by 76.9%. We can determine the chance that a linear relationship exists between the dependent and independent variables based on the p-value.

If the p-value is close to 0, a linear relationship between the independent and dependent variables exists and is statistically significant. If the p-value is less than 0.05, there is significant evidence against the null hypothesis, which asserts that the independent variable has no effect on the dependent variable. Specifically, the p-value of 0.291 for 'AGEGROUP age 30' suggests that there is a possibility of 29.1% that this group has no effect on 'B1'.

In addition, we observed that variables such as 'RATING GOOD' and 'RATING BAD' had a p-value of 0, signifying that the data is statically important because it is less than the critical value (0.05). In this case, we can disregard the null hypothesis and infer that 'B1' is significantly affected by these data.

1. Scikit Learn

**# Dependent Variable = 'B1', Independent Variables = All other variables except 'B1'**

X = model\_df.drop(columns = ['B1'])

y = model\_df['B1']

x\_train, x\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size = 0.2, random\_state = 42)

**# creating an object of LinearRegression class**

LR = LinearRegression()

**# fitting the training data**

LR.fit(x\_train,y\_train)

y\_prediction = LR.predict(x\_test)

y\_prediction

**# Predicting the accuracy score**

score = r2\_score(y\_test, y\_prediction)

print('R2 score is ',score)

print('Mean Sqrd Error is ',mean\_squared\_error(y\_test,y\_prediction))

print('Root Mean Sqrd Error of is ',np.sqrt(mean\_squared\_error(y\_test,y\_prediction)))

print('Mean Absolute Error is ', mean\_absolute\_error(y\_test, y\_prediction))

Text

Description automatically generated

**# Displaying Coefficients**

coefficient = pd.DataFrame(list(zip(X.columns.tolist(), LR.coef\_)), columns = ['Columns', 'Coefficient']).set\_index('Columns')

coefficient

Table

Description automatically generatedTable

Description automatically generated

We can observe from the preceding that Scikit Learn provides different results than Statsmodel. This is because scikit-learn by default regularizes logistic regressions. This is beneficial when attempting to predict variables, since it prevents the model from becoming overfit to its training data. Therefore, it is a suitable default when applying scikit-learn.

In addition, Statsmodels provides modeling from a statistical standpoint, while Scikit-learn provides similar models from a machine learning perspective (Srinivasan, 2020). Using the p-values, Statsmodels identifies which of our variables are statistically significant, which is useful for tuning the model.

Since both models have their own useful capabilities, both shall be employed when applying statistical modelling. Scikit Learn would be recommend to create your final model, but statsmodels is an excellent tool for analyzing the data prior to incorporating it into the model.

In conclusion, there are substantial correlations between all independent variables and 'B1'. Note, however, that the selected characteristics may not necessarily apply to the demographics of all clients for all credit facilities. We utilized a macro viewpoint and picked representative variables for the investigation, drawing conclusions and reviewing the data pertaining to the impact of each variable on the most current Billable Amount.

**References**

Cherry, K. (2022, April 15). *The role of correlations in psychology research*. Verywell Mind. Retrieved September 4, 2022, from https://www.verywellmind.com/what-is-correlation-2794986

Kagan, J. (2022, August 29). *How credit rating works*. Investopedia. Retrieved September 4, 2022, from https://www.investopedia.com/terms/c/creditrating.asp

Kumar, A. (2022, April 20). *Dummy variables in regression models: Python, R*. Data Analytics. Retrieved September 4, 2022, from https://vitalflux.com/dummy-variables-in-regression-models-python-r/#What\_are\_dummy\_variables\_in\_linear\_regression\_models

Lee, K. H. C. (2021). ANL303 Fundamentals of data mining (study guide). Singapore: Singapore University of Social Sciences.

Srinivasan, H. (2020, May 19). *Linear regression in Scikit-learn vs Statsmodels*. Medium. Retrieved September 4, 2022, from https://medium.com/@hsrinivasan2/linear-regression-in-scikit-learn-vs-statsmodels-568b60792991

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