****

**ANL252**

**PYTHON FOR DATA ANALYTICS**

# **End-of-Course Assessment**

# **July 2022 Presentation**

**Submitted by:**

|  |  |
| --- | --- |
| **Name** | **PI No.** |
| **Toh Li Ting** | **M2172239** |

**Submission Date: 05/09/2022**

**Qn1**

The categorial variables are ID, gender, education, martial, age and rating.

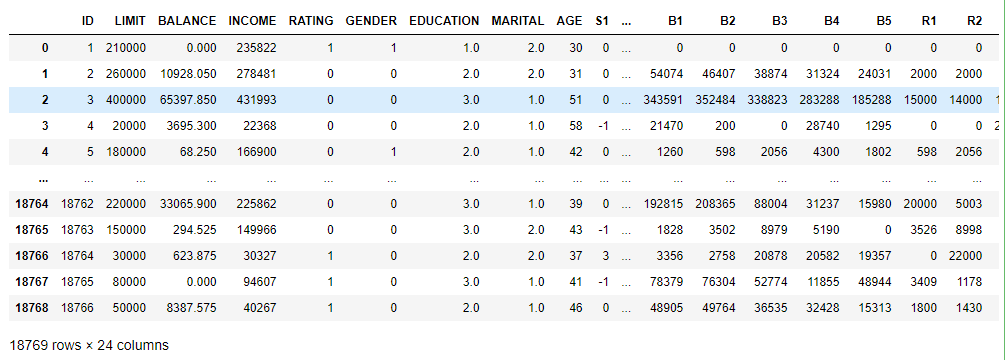
The numeric variables are limit, balance, income, S(n), B(n) and R(n).

**Qn2**

df\_credit = pd.read\_csv('ECA\_data.csv')

print (df\_credit)

df\_credit.head(18769)



Identify data information with the following code:

df\_credit.info()

<class 'pandas.core.frame.DataFrame'>

RangeIndex: 18769 entries, 0 to 18768

Data columns (total 24 columns):

# Column Non-Null Count Dtype

--- ------ -------------- -----

0 ID 18769 non-null int64

1 LIMIT 18769 non-null int64

2 BALANCE 18769 non-null float64

3 INCOME 18769 non-null int64

4 RATING 18769 non-null int64

5 GENDER 18769 non-null int64

6 EDUCATION 18756 non-null float64

7 MARITAL 18731 non-null float64

8 AGE 18769 non-null int64

9 S1 18769 non-null int64

10 S2 18769 non-null int64

11 S3 18769 non-null int64

12 S4 18769 non-null int64

13 S5 18769 non-null int64

14 B1 18769 non-null int64

15 B2 18769 non-null int64

16 B3 18769 non-null int64

17 B4 18769 non-null int64

18 B5 18769 non-null int64

19 R1 18769 non-null int64

20 R2 18769 non-null int64

21 R3 18769 non-null object

22 R4 18769 non-null int64

23 R5 18769 non-null int64

dtypes: float64(3), int64(20), object(1)

In Data cleaning, replace missing values. Identify missing values using the following code:

df\_credit.isnull().sum()

LIMIT 0

BALANCE 0

INCOME 0

RATING 0

MARITAL 38

AGE 0

S1 0

S2 0

S3 0

S4 0

S5 0

B1 0

B2 0

B3 0

B4 0

B5 0

R1 0

R2 0

R3 0

R4 0

R5 0

dtype: int64

Using isnull(), 38 missing values are found under the category “Marital”.

To replace the missing value, use the following code to identify the mode of “Marital” and smoothen out missing values by replacing them:

df\_credit['MARITAL'] = df\_credit['MARITAL'].replace(['2.0'],'Married')

df\_credit['MARITAL'] = df\_credit['MARITAL'].replace(['1.0'],'Single')

df\_credit['MARITAL'].value\_counts()

Married 9834

Single 8708

0.0 189

nan 38

Name: MARITAL, dtype: int64

df\_credit['MARITAL'] = df\_credit['MARITAL'].replace(['0.0'],'Married')

df\_credit['MARITAL'] = df\_credit['MARITAL'].replace(['nan'],'Married')

df\_credit['MARITAL'].value\_counts()

Married 10061

Single 8708

Name: MARITAL, dtype: int64

In this step, also remove outliers for “Age”:

df\_credit.drop([531,660,1613,4238,7829,15354,1807,18161,18162,18278])

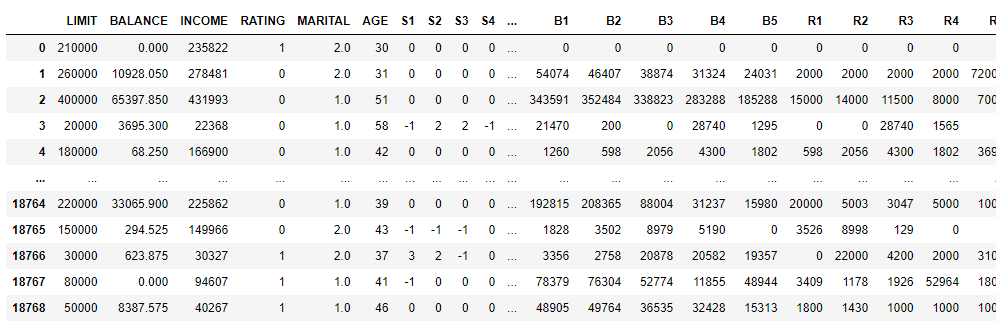
to remove age -1 & 199

In Data integration, as a data preparation step to combine multiple datasets into one.

In Data reduction, reduce unnecessary data. In this case, such as ID & Education.

df\_credit.drop(['ID','EDUCATION'], axis=1, inplace=True)

print (df\_credit)



In Data transformation, ensure that data are in the correct format to work on.

print dtype: objectt (df\_credit.dtypes)

From initial:

LIMIT int64

BALANCE float64

INCOME int64

RATING int64

MARITAL float64

AGE int64

S1 int64

S2 int64

S3 int64

S4 int64

S5 int64

B1 int64

B2 int64

B3 int64

B4 int64

B5 int64

R1 int64

R2 int64

R3 object

R4 int64

R5 int64

dtype: object

df\_credit['MARITAL'] = df\_credit['MARITAL'].astype(str)

df\_credit['BALANCE'] = df\_credit['BALANCE'].astype(int)

print (df\_credit.dtypes)

LIMIT int64

BALANCE int32

INCOME int64

RATING int64

MARITAL object

AGE int64

S1 int64

S2 int64

S3 int64

S4 int64

S5 int64

B1 int64

B2 int64

B3 int64

B4 int64

B5 int64

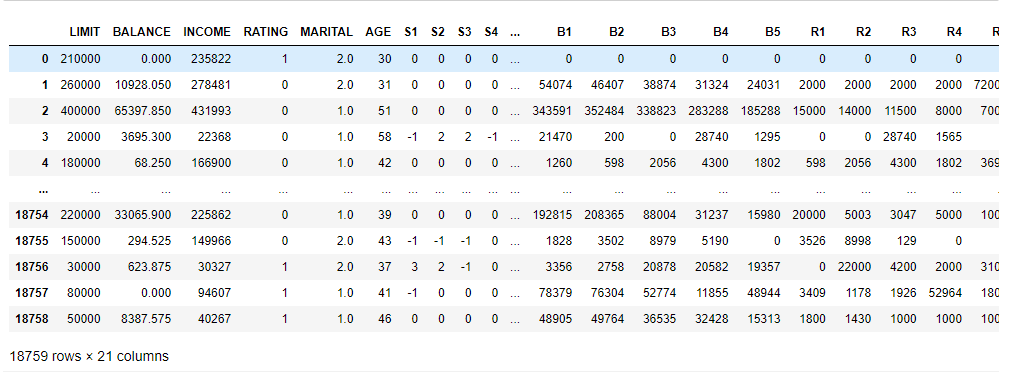
R1 int64

R2 int64

R3 int32

R4 int64

R5 int64



**Qn3**

1. # generate histogram

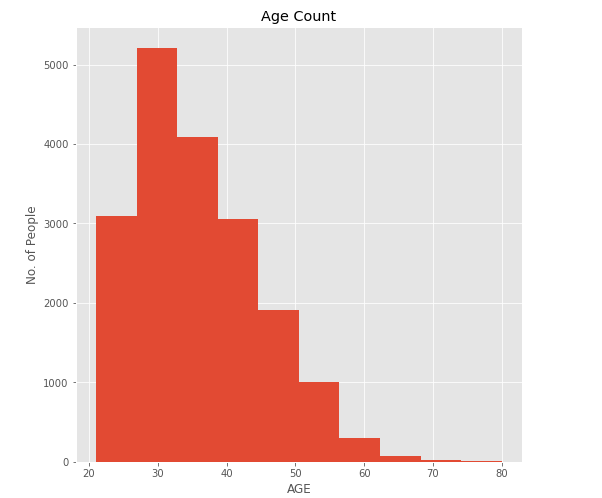
df\_credit.hist(column=['AGE'], figsize=(8, 8))

plt.title('Age Count')

plt.ylabel('No. of People')

plt.xlabel('AGE')

plt.show()



From the chart above, we are able to see the age group of the customers.

Having majority of them in their 30s-40s.

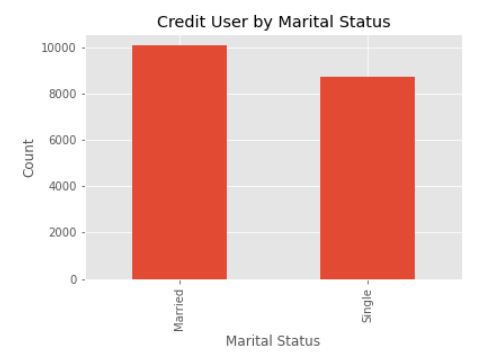
1. Below chart shows the distribution of the credit user by Marital Status.

df\_credit["MARITAL"].value\_counts().plot.bar()

plt.title('Credit User by Marital Status')

plt.ylabel('Count')

plt.xlabel('Marital Status')



From the chart above, we are able to see that there are more married credit card users compared to people who are single.

1. Customer Rating

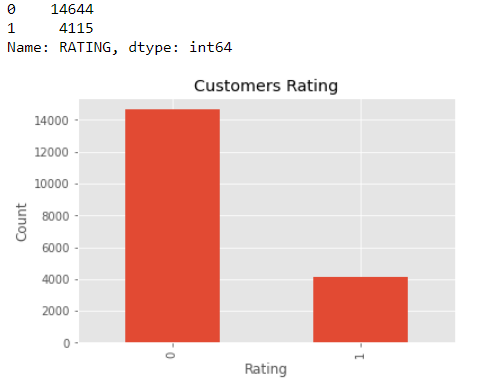
df\_credit["RATING"].value\_counts().plot.bar()

plt.title('Customers Rating')

plt.ylabel('Count')

plt.xlabel('Rating')

df\_credit['RATING'].value\_counts()



From the chart, we are able to see that the customer base has a rating of 14,664 customers with good rating at a distribution of 78% and 4,115 customers with bad rating at a distribution of 22%.

1. Customers repayment status in nth month.

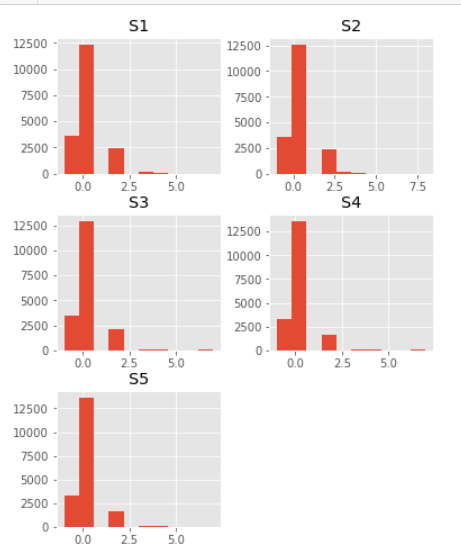
df\_credit.hist(column=['S1', 'S2', 'S3','S4','S5'], figsize=(6, 8))

plt.title('Customer repayment status')

plt.ylabel('Count')

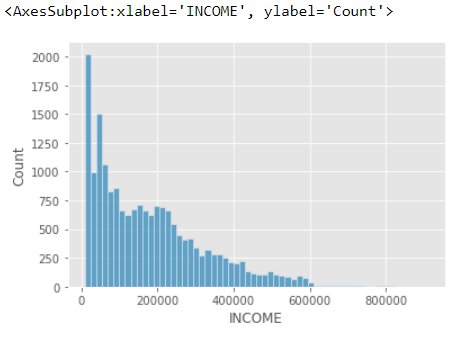
plt.xlabel('Month')

plt.show()



From the chart above, we are able to see that most customers cleared their bills in the payment period or at the next month and some cleared in the 2nd month and there are some delay payments.

1. Distribution of Income.



sns.histplot(data=df\_credit,x="INCOME")

From the chart we are able see that most of the customers have an income below $200,000.

**Qn4**

Removing of object column & y-asix column

x=df\_credit.drop(["B1",'MARITAL'],axis=1).values

y=df\_credit["B1"].values

print(x)

print(y)

[[ 0 210000 0 ... 0 0 0]

[ 1 260000 10928 ... 2000 2000 72000]

[ 2 400000 65397 ... 11500 8000 7000]

...

[ 18756 30000 623 ... 4200 2000 3100]

[ 18757 80000 0 ... 1926 52964 1804]

[ 18758 50000 8387 ... 1000 1000 1000]]

[ 0 54074 343591 ... 3356 78379 48905]

Import sklearn model and testing of test size:

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

x\_train,x\_test,y\_train,y\_test=train\_test\_split(x,y,test\_size=0.3,random\_state=0)

from sklearn.linear\_model import LinearRegression

ml=LinearRegression()

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

y\_pred=ml.predict(x\_test)

print(y\_pred)

To check on the prediction confident level:

from sklearn.metrics import r2\_score

r2\_score(y\_test,y\_pred)

0.9485239044148921

Scatter Plot

import matplotlib.pyplot as plt

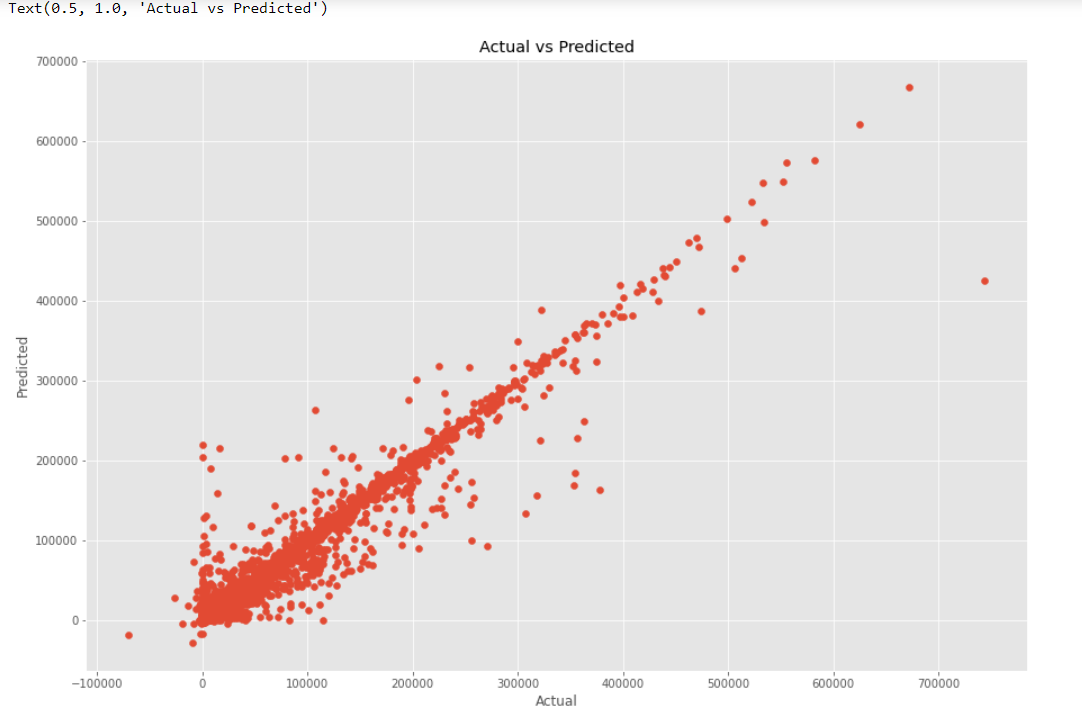
plt.figure(figsize=(15,10))

plt.scatter(y\_test,y\_pred)

plt.xlabel('Actual')

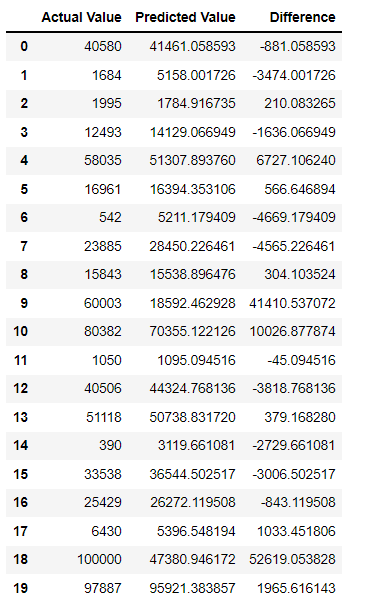
plt.ylabel('Predicted')

plt.title('Actual vs Predicted')



pred\_y\_df\_credit=pd.DataFrame({'Actual Value':y\_test,'Predicted Value':y\_pred,'Difference':y\_test-y\_pred})

pred\_y\_df\_credit[0:20]



**Qn5**

From Question 4, we are able to find a prediction for B1 with a confidence level of 94%.

y = mx + b