DATA MINING ASSIGNMENT-1

CSE-5334 : DATA MINING

EXPLORATORY ANALYSIS OVER DATASET_R

1.INTRODUCTION:

R: R is a programming language commonly used for the purpose of data visualization and analyzing statistics. The language R is purpose-specific which is handling, visualising and manipulating data. In other words , it is a statistics analysis tool. It is widely used in a load of usecases including but not limiting to business analytics, data mining , data science and data analysis etc.

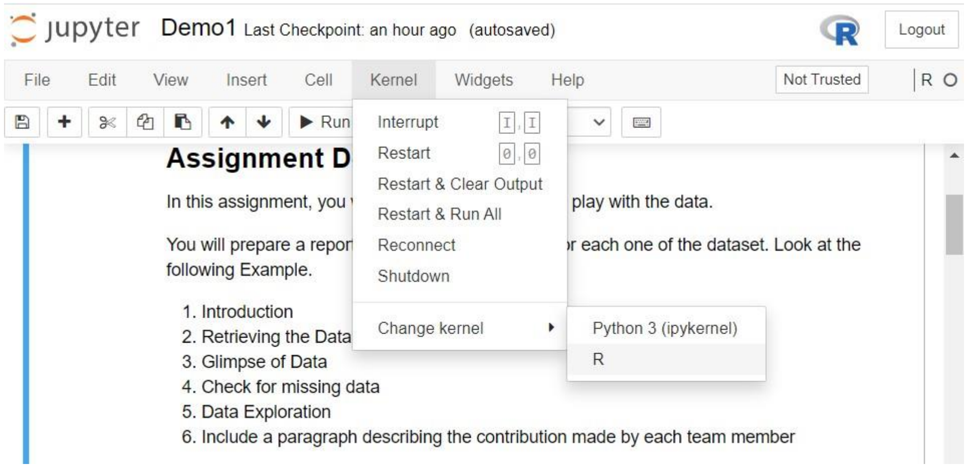
The objective of the given assignment is to explore, manipulate and play the data provided to us through an .csv file "dataset_R". As per requirement, the project is to be performed in Jupyter_notebook .

The first and foremost task to be performed inorder to be able to perform the requirements is installing R kernel into existing Jupyter_notebook.

In order to do so,

We have utilised the below command in Anaconda prompt.

After finshing up the installation of r kernel to jupyter, we open the given assignment file by the name "Assignment 1 R.pynyb" . By default the kernel and programming language in jupyter_notebook is Python3 . We have to change the kernel to R which also automatically changes the programming language to R in the IDE. This is shown below.



With this step , we are ready to jump into the assignment.

Before we proceed to step two of the assignment , we are in need of importing the required libraries inorder to perform the desired tasks . In our assignment the required libraries are : dplyr and ggplot2.

DPLYR : The library dplyr is called inorder to provide us with a grammar for the desired code. It is a major step towards manipulating data as the contents carried with the library provide a backbone for R . One such example is the use of "%>%" which is called a forward pipe operator. It is used for chaining commands.

GGPLOT2: The tasks 3 a, 3 b & 4 require us to visualise data from the dataframe. We utilised various data representations like bar graphs , density graphs and pie charts which are only possible due to the import of ggplot2 library. The functions utilised in these tasks are only valid as long as we have imported this library.

We have achieved this with the following lines of code :

```
# Import R packages
library('dplyr')
library('ggplot2')
```

library('dplyr') library('ggplot2')

2.RETRIEVING DATA:

After importing our libraries , we get to perform the tasks. But we still don't have the data we are to perform the tasks on, accessible to the IDE. We need to bring the data from the csv file to our file. This part is called retrieving data. We achieve this by using the following command to read the file. Reading can in a way be seen as loading the data into the IDE.

The command is as follows :

load_df<- read.csv('C:\\Users\\Administrator\\Desktop\\Assignment1UPDATED\\dataset_R.csv') *load_df is the object we have chosen to utilise throughtout the course of the project . Read.csv is the function to link the dataframe to the IDE and We provided the path to the required dataframe on my device.*

```
In [3]:  # Read the file
         loan_df <- read.csv('C:\\Users\\Administrator\\Desktop\\Assignment1UPDATED\\dataset_R.csv')
```

3.GLIMPSE OF THE DATA:

The follow up task after importing the data from dataframe is to check if the import is fully functional. To do so, we check through making a glimpse of the data by using the following command .

head(loan_df,5)

Head prints the values from the top of the table . 5 results in printing the first 5 values from the top.

```
# return the first 5 rows of the dataset
head(loan_df,5)
```

Loan_ID	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	CoapplicantIncome	LoanAmount	Loan_Amount_Term	Credit_History
LP001002	Male	No	0	Graduate	No	5849	0	NA	360	1
LP001003	Male	Yes	1	Graduate	No	4583	1508	128	360	1
LP001005	Male	Yes	0	Graduate	Yes	3000	0	66	360	1
LP001006	Male	Yes	0	Not Graduate	No	2583	2358	120	360	1
LP001008	Male	No	0	Graduate	No	6000	0	141	360	1

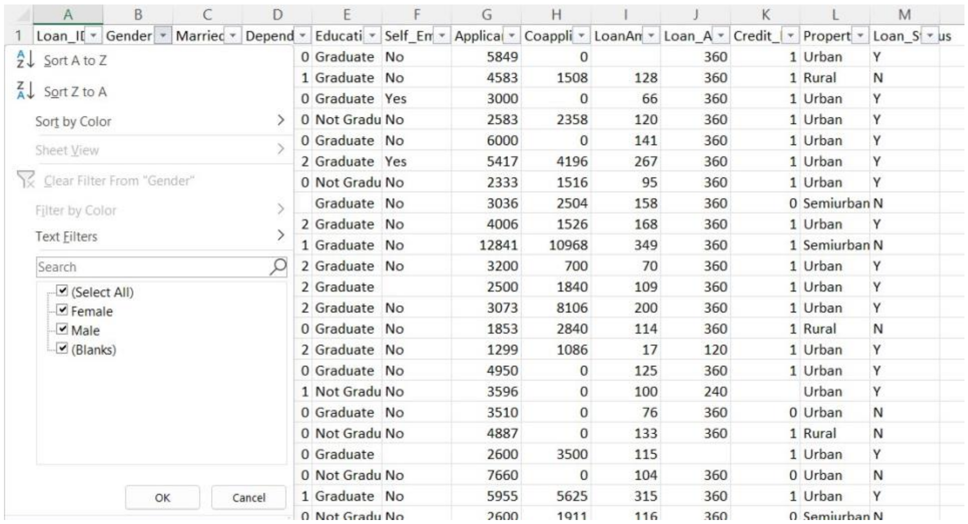
J1 fx Loan_Amount_Term

	A	B	C	D	E	F	G	H	I	J	K	L	M
1	Loan_ID	Gender	Married	Dependen	Education	Self_Empl	ApplicantI	Coapplicar	LoanAmou	Loan_Amc	Credit_His	Property_/	Loan_Status
2	LP001002	Male	No	0	Graduate	No	5849	0		360	1	Urban	Y
3	LP001003	Male	Yes	1	Graduate	No	4583	1508	128	360	1	Rural	N
4	LP001005	Male	Yes	0	Graduate	Yes	3000	0	66	360	1	Urban	Y
5	LP001006	Male	Yes	0	Not Gradu	No	2583	2358	120	360	1	Urban	Y
6	LP001008	Male	No	0	Graduate	No	6000	0	141	360	1	Urban	Y
7	LP001011	Male	Yes	2	Graduate	Yes	5417	4196	267	360	1	Urban	Y
8	LP001013	Male	Yes	0	Not Gradu	No	2333	1516	95	360	1	Urban	Y
9	LP001014	Male	Yes	3+	Graduate	No	3036	2504	158	360	0	Semiurban	N
10	LP001018	Male	Yes	2	Graduate	No	4006	1526	168	360	1	Urban	Y

We can see that the data had been consistently read when comparing the table from our task and the csv table.

4.CHECK FOR MISSING DATA (Task-1)

As part of our Task-1a , we had to deal with missing data. From our understanding of the csv file, we can use the operation ctrl+shift+l to browse through the blank entries in our columns.



Loan_ID	Gender	Married	Depend	Educati	Self_Em	Applica	Coappli	LoanAm	Loan_A	Credit_	Propert	Loan_Status
			0	Graduate	No	5849	0		360	1	Urban	Y
			1	Graduate	No	4583	1508	128	360	1	Rural	N
			0	Graduate	Yes	3000	0	66	360	1	Urban	Y
			0	Not Gradu	No	2583	2358	120	360	1	Urban	Y
			0	Graduate	No	6000	0	141	360	1	Urban	Y
			2	Graduate	Yes	5417	4196	267	360	1	Urban	Y
			0	Not Gradu	No	2333	1516	95	360	1	Urban	Y
				Graduate	No	3036	2504	158	360	0	Semiurban	N
			2	Graduate	No	4006	1526	168	360	1	Urban	Y
			1	Graduate	No	12841	10968	349	360	1	Semiurban	N
			2	Graduate	No	3200	700	70	360	1	Urban	Y
			2	Graduate		2500	1840	109	360	1	Urban	Y
			2	Graduate	No	3073	8106	200	360	1	Urban	Y
			0	Graduate	No	1853	2840	114	360	1	Rural	N
			2	Graduate	No	1299	1086	17	120	1	Urban	Y
			0	Graduate	No	4950	0	125	360	1	Urban	Y
			1	Not Gradu	No	3596	0	100	240		Urban	Y
			0	Graduate	No	3510	0	76	360	0	Urban	N
			0	Not Gradu	No	4887	0	133	360	1	Rural	N
			0	Graduate		2600	3500	115		1	Urban	Y
			0	Not Gradu	No	7660	0	104	360	0	Urban	N
			1	Graduate	No	5955	5625	315	360	1	Urban	Y
			0	Not Gradu	No	2600	1911	116	360	0	Semiurban	N

From checking all the columns, it has been noticed that a total of 6 columns contain null or missing values. From my learning of R, we can utilise the function is.na() to find out the null values . This however does not cover the missing values. I have performed the is.na() in task 1 a. The following has been observed.

```r
# 1-a If any, print the total number of null values for each column in the dataset. Explain how you handle the null values (H
sapply(loan_df,function(loan_df) sum(is.na(loan_df)))
#summary(loan_df)
```

Loan_ID	0
Gender	0
Married	0
Dependents	0
Education	0
Self_Employed	0
ApplicantIncome	0
CoapplicantIncome	0
LoanAmount	22
Loan_Amount_Term	14
Credit_History	50
Property_Area	0
Loan_Status	0

A total of 86 nulls has been reported and the represenatation is done column. The missing values in the columns Gender, Married and Self Employed can also viewed when we use the summary() function.

```
# 1-a If any, print the total number of null values for each column in the dataset. Explain how you handle the null values (H
#sapply(loan_df,function(loan_df) sum(is.na(loan_df)))
summary(loan_df)
```

```
     Loan_ID       Gender     Married   Dependents        Education
 LP001002:  1         : 13       :  3     : 15     Graduate    :480
 LP001003:  1   Female:112   No :213   0 :345     Not Graduate:134
 LP001005:  1   Male  :489   Yes:398   1 :102
 LP001006:  1                          2 :101
 LP001008:  1                          3+: 51
 LP001011:  1
 (Other) :608
 Self_Employed ApplicantIncome CoapplicantIncome   LoanAmount
    : 32       Min.   :  150   Min.   :    0     Min.   :  9.0
 No :500       1st Qu.: 2878   1st Qu.:    0     1st Qu.:100.0
 Yes: 82       Median : 3812   Median : 1188     Median :128.0
               Mean   : 5403   Mean   : 1621     Mean   :146.4
               3rd Qu.: 5795   3rd Qu.: 2297     3rd Qu.:168.0
               Max.   :81000   Max.   :41667     Max.   :700.0
                                                 NA's   :22
 Loan_Amount_Term Credit_History     Property_Area Loan_Status
 Min.   : 12      Min.   :0.0000   Rural    :179   N:192
 1st Qu.:360      1st Qu.:1.0000   Semiurban:233   Y:422
 Median :360      Median :1.0000   Urban    :202
 Mean   :342      Mean   :0.8422
 3rd Qu.:360      3rd Qu.:1.0000
 Max.   :480      Max.   :1.0000
 NA's   :14       NA's   :50
```

From the image, in the columns a "   : integer" represents the missing cells along with the number of missing cells. For null values in integer form , we can check the count via NA : integer representation in the summarized column details. Even though not being the most efficient method , this is the most effective method we were able to put forward.

Handling the missing values :

1. The first three columns with missing values, we can proceed by replacing the blanks with the string "NA".
2. For the next three columns with missing values , we can proceed by replacing the blanks with integer 0.

By using these two steps we will be able to bring in a more consistent dataframe with more understandable data.

5. DATA EXPLORATION

Data exploration is the process of surfing through our data in a manner that'll enable us to manipulate data while discovering useful patterns. This is what we do through our tasks that follow 1-a.

TASK-1B : # 1-b Print the details of dataframe

Here we need to print the details of our given dataframe , and we achieved this with the following line of code :

str(loan_df)

```
# 1-b Print the details of dataframe
str(loan_df)
```

```
'data.frame':	614 obs. of  13 variables:
 $ Loan_ID          : Factor w/ 614 levels "LP001002","LP001003",..: 1 2 3 4 5 6 7 8 9 10 ...
 $ Gender           : Factor w/ 3 levels "","Female","Male": 3 3 3 3 3 3 3 3 3 3 ...
 $ Married          : Factor w/ 3 levels "","No","Yes": 2 3 3 3 2 3 3 3 3 3 ...
 $ Dependents       : Factor w/ 5 levels "","0","1","2",..: 2 3 2 2 2 4 2 5 4 3 ...
 $ Education        : Factor w/ 2 levels "Graduate","Not Graduate": 1 1 1 2 1 1 2 1 1 1 ...
 $ Self_Employed    : Factor w/ 3 levels "","No","Yes": 2 2 3 2 2 3 2 2 2 2 ...
 $ ApplicantIncome  : int  5849 4583 3000 2583 6000 5417 2333 3036 4006 12841 ...
 $ CoapplicantIncome: num  0 1508 0 2358 0 ...
 $ LoanAmount       : int  NA 128 66 120 141 267 95 158 168 349 ...
 $ Loan_Amount_Term : int  360 360 360 360 360 360 360 360 360 360 ...
 $ Credit_History   : int  1 1 1 1 1 1 1 0 1 1 ...
 $ Property_Area    : Factor w/ 3 levels "Rural","Semiurban",..: 3 1 3 3 3 3 3 2 3 2 ...
 $ Loan_Status      : Factor w/ 2 levels "N","Y": 2 1 2 2 2 2 2 1 2 1 ...
```

<u>Task-1C :</u> # 1-c Find the number of rows and columns in dataset

As part of task 1-C, we need to finf the number of rows and columns in our dataset. We achieve this with the following lines of code :

print(paste0('Total number of rows:', nrow(loan_df)))

print(paste0('Total number of columns:', ncol(loan_df)))

```
# 1-c Find the number of rows and columns in dataset
print(paste0('Total number of rows:', nrow(loan_df)))
print(paste0('Total number of columns:', ncol(loan_df)))
```

```
[1] "Total number of rows:614"
[1] "Total number of columns:13"
```

We can infer that number of rows and columns in our data set are :

Rows -> 614

Columns -> 13

<u>TASK 1-D :</u> # 1-d Print descriptive detail of a column in dataset

As part of task 1-d we are required to show the descriptive details of any singular column from our dataset. We were able to achieve this using the following line of code: *summary(loan_df$ApplicantIncome)*

```
# 1-d Print descriptive detail of a column in dataset
summary(loan_df$ApplicantIncome)
```

```
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
    150    2878    3812    5403    5795   81000 
```

Data Pre-Processing for R :

```
# Data Pre-processing for R
loan_df <- read.csv('C:\\Users\\Administrator\\Desktop\\Assignment1UPDATED\\dataset_R.csv',na.strings= c("","NA"))
```

In data pre-processing , we can phase out all the missing values while reading itself. To do so, the above code helps out. It is the same as the reading code provided above but adds ,na.strings=c(””,”NA”). This will remove all the rows containg null and missing values, giving a data set with more consistent entries.

<u>Task-2a :</u> # Task 2-a: Find out the number of graduates from rural area

Here we are to print values from two different columns based on a condition on each column respectively. We achieve this by using the following line of code : *loan_df %>%*

filter(Education == 'Graduate' & Property_Area == 'Rural') %>%

nrow()

```
# Task 2-a: Find out the number of graduates from rural area
loan_df %>%
filter( Education == 'Graduate' & Property_Area == 'Rural' ) %>%
nrow()
```

131

From the code , we utilize our object loan_df to access the functions and dataframe. %>% is utilised to maintain a chain of commands. Filter function is used to subset data from our dataframe.

We do this by passing education and property_area columns as parameters while selectively only sending where the values are = graduate and rural .

<u>Task 2-b :</u> # Task 2-b: determine the overall number of men who did not graduate

Similar to task 2-a , we are two print selective number of combinational results from the columns Education and Gender as we are required to print men who did not graduate.

We are able to achieve this with following line of code.

loan_df %>% filter(Education == 'Not Graduate' & Gender

== 'Male') %>% nrow()

```
# Task 2-b: determine the overall number of men who did not graduate
loan_df %>%
filter( Education == 'Not Graduate' & Gender == 'Male' ) %>%
nrow()
```

113

<u>Task 2-c</u>: # Task 2-c: Find the top 10 men who graduated and had the highest applicant income

We achieve this by the code: loan_df %>% arrange(-ApplicantIncome) %>% filter(Education ==

'Graduate' & Gender == 'Male') %>%

#select(Loan_ID) %>%

#loan_df(loan_df$Education == 'Graduate',) %>%

head(10)

```
# Task 2-c: Find the top 10 men who graduated and had the highest applicant income
loan_df %>%
 arrange(-ApplicantIncome) %>%
filter( Education == 'Graduate' & Gender == 'Male' ) %>%
#select(Loan_ID) %>%
 #loan_df(loan_df$Education == 'Graduate',) %>%
 head(10)
```

Loan_ID	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	CoapplicantIncome	LoanAmount	Loan_Amount_Term	Credit_History
LP002317	Male	Yes	3+	Graduate	No	81000	0	360	360	0
LP002101	Male	Yes	0	Graduate		63337	0	490	180	1
LP001536	Male	Yes	3+	Graduate	No	39999	0	600	180	0
LP001640	Male	Yes	0	Graduate	Yes	39147	4750	120	360	1
LP002422	Male	No	1	Graduate	No	37719	0	152	360	1
LP001637	Male	Yes	1	Graduate	No	33846	0	260	360	1
LP002624	Male	Yes	0	Graduate	No	20833	6667	480	360	NA
LP001922	Male	Yes	0	Graduate	No	20667	0	NA	360	1
LP001996	Male	No	0	Graduate	No	20233	0	480	360	1
LP001469	Male	No	0	Graduate	Yes	20166	0	650	480	NA

The rows of the top 10 highest earning males are printed through this code including the required loan_id attribute. Using filter function In a required manner.

<u>Task 2-d :</u> # Task 2-d: Find the number of self-employed male applicants from urban area

Here, we use the function filter again to selectively print the number of the self-employed males with property area as rural. We check the attributes of rows to match with set of attributes (Yes , Male, Urban) for columns (Self_Employed, Gender, Property_Area) respectively.
This gives us a count of 19.

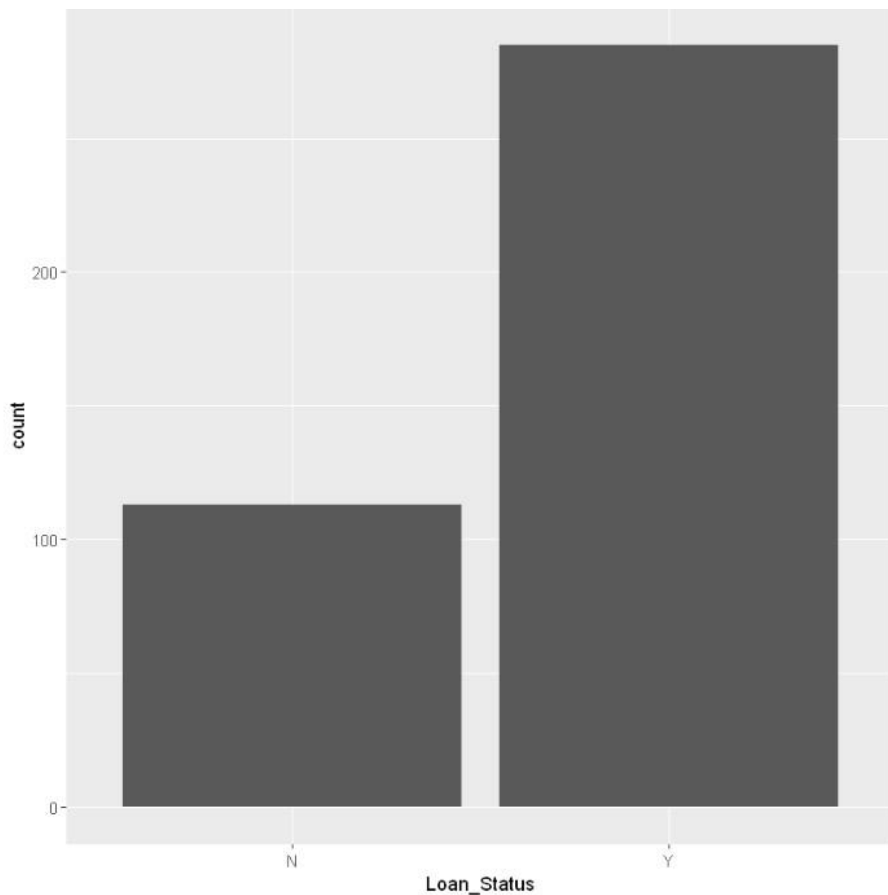
```
# Task 2-d: Find the number of self-employed male applicants from urban area
loan_df %>%
filter( Self_Employed == 'Yes' & Gender == 'Male' & Property_Area == 'Urban' ) %>%
nrow()
```

19

Task 3-a: # Task 3-a Make a plot where married applicants are granted loans

For visualising the requirements, we have chosen bar graph. For X-axis ,we have chose married attribute with value 'Yes' and for yaxis we have chosen loan status with both values of 'Y' and 'N'. We utilised the function ggplolt to construct the bar graph.

```
# Task 3-a Make a plot where married applicants are granted loans
dump <- subset(loan_df, Married == 'Yes')
ggplot(dump,aes(x=Loan_Status)) + geom_bar()
```



Task 3-b: # Task 3-b Create a pie chart for Property_area and display percentages in legend respectively

We have created a piechart as required by using ggplot function.

Since it is a pie chart , we let the value of x to be null , y for count as n and filled the chart with property area as required . For printing it percentage wise, we utilise :

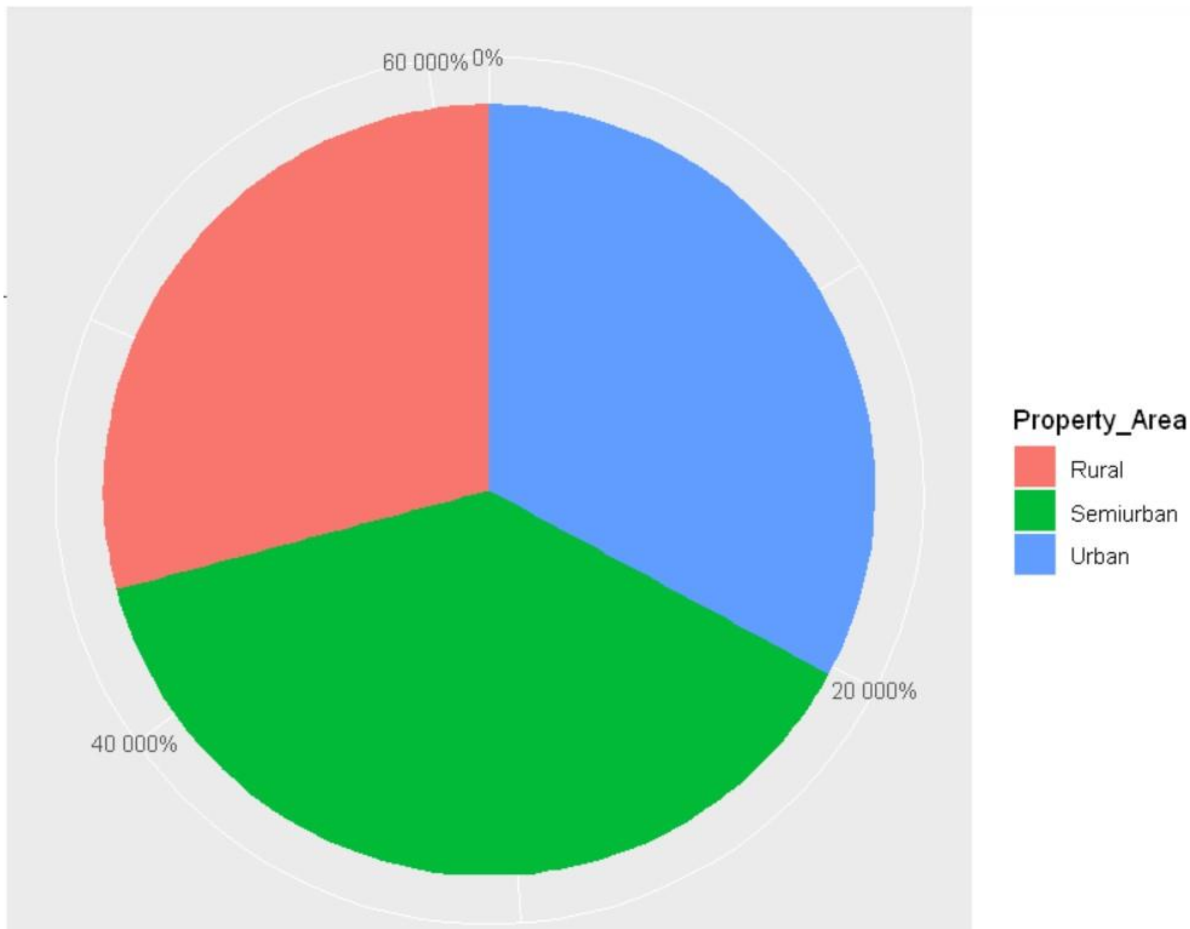
+ scale_y_continuous(labels = scales::percent)

loan_df %>% group_by(Property_Area) %>%

tally() %>%

ggplot(aes(x="", y=n, fill=Property_Area)) + geom_bar(stat='identity',width=1)+coord_polar("y", start=0) + scale_y_continuous(labels = scales::percent)

```
# Task 3-b Create a pie chart for Property_area and display percentages in legend respectively
loan_df %>%
group_by(Property_Area) %>%
tally() %>%
ggplot(aes(x="", y=n, fill=Property_Area)) + geom_bar(stat='identity',width=1)+coord_polar("y", start=0 )  + scale_y_continuc
```

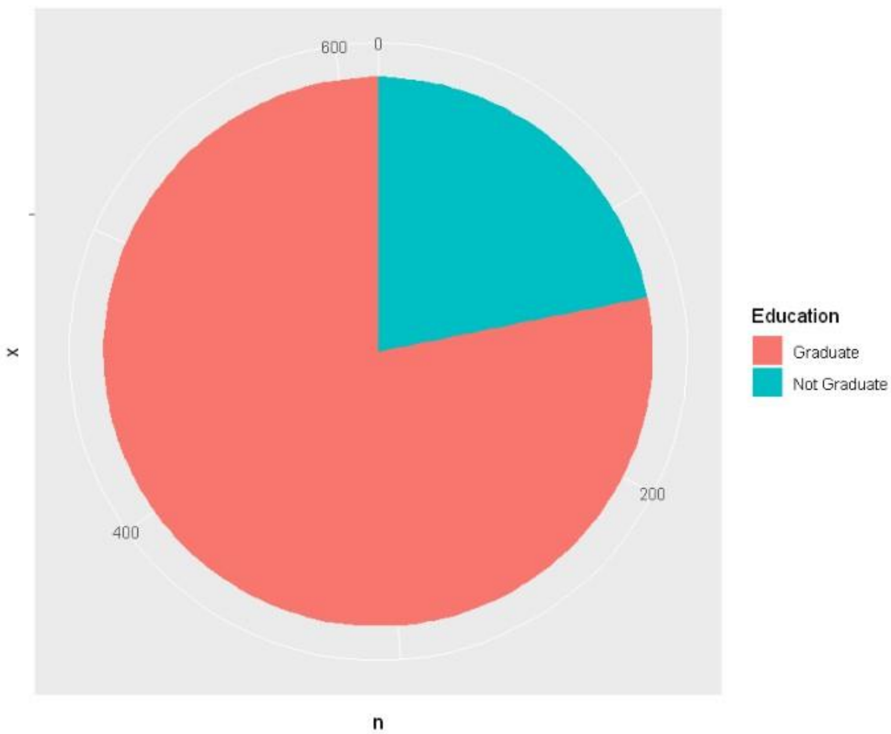


Task – 4 :

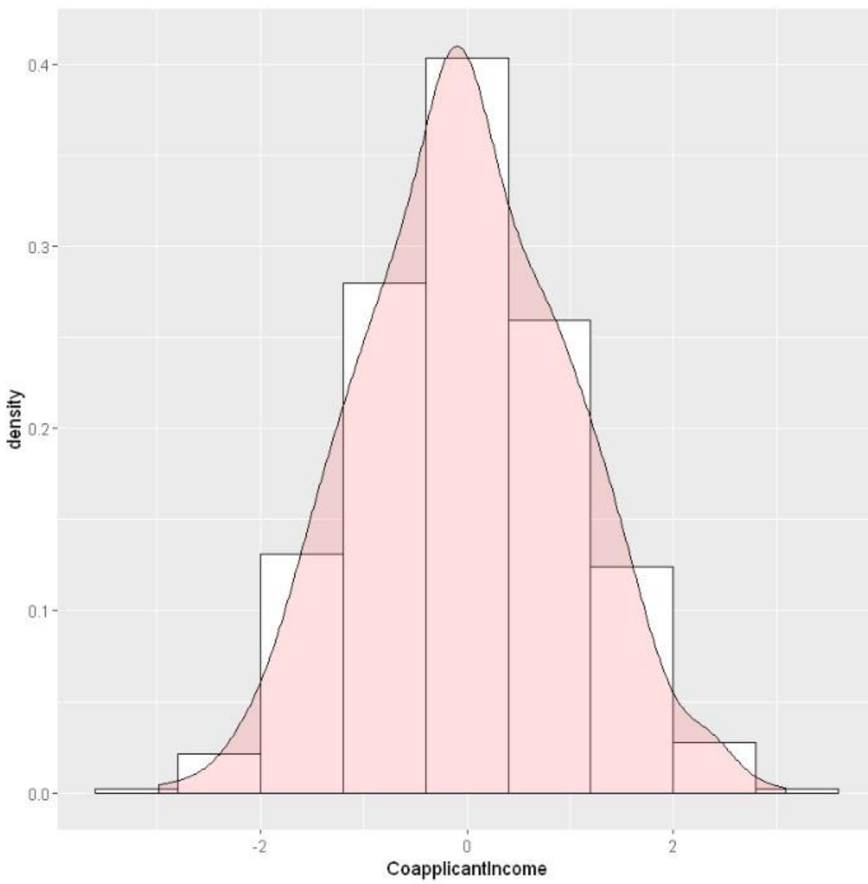
In Task 4, we are required to find out and add two visualizations that gives valuable insights into the dataset. For that I have selected two visualisation a pie chart and a density map involving education and coapplicant income . Since we are dealing with education, we have deicded to create a piechart with education , both graduate and nongraduate . In the denisty plot , coapplicant income is the x axis while drawing a reference to education.

The two visualisations are provided below :

```
# Code and explaination for loan_df %>%
loan_df %>%
group_by(Education) %>%
tally() %>%
ggplot(aes(x="", y=n, fill=Education)) + geom_bar(stat='identity',width=1)+coord_polar("y", start=0)
```




```
dat <- data.frame(Education = factor(rep(c("A","B"), each = 615)), CoapplicantIncome = c(rnorm(615),rnorm(615)))
ggplot(dat, aes(x = CoapplicantIncome)) + geom_histogram(aes(y =..density..),binwidth=.8, colour ="black",fill="white")+geom_
```



0.4
0.3
0.2
0.1
0.0
density
-2
0
2
CoapplicantIncome