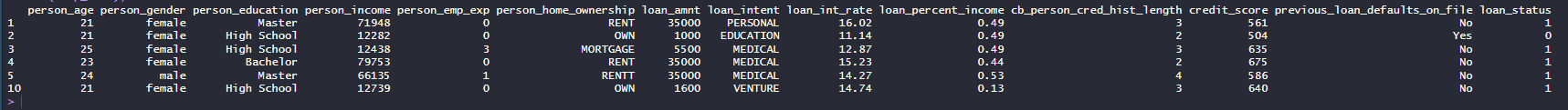
1.Handle missing value

* Drop missing value:

|  |
| --- |
| missing\_rows <- data[!complete.cases(data), ] head(missing\_rows) |

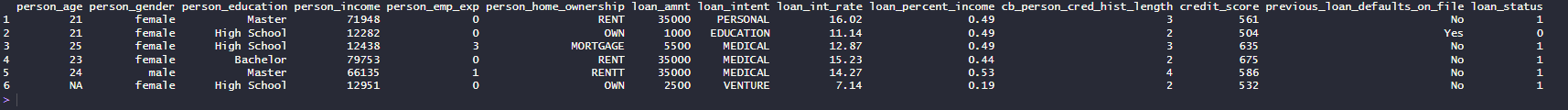


* Replace with mean

df\_mean <-data

df\_mean$person\_income[is.na(df\_mean$person\_income)] <- mean(df\_mean$person\_income,na.rm=TRUE)

head(df\_mean)

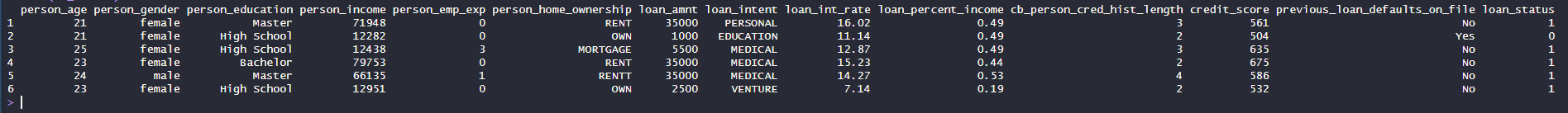


* Replace with median

df\_median <- df\_mean#replace by median

df\_median$person\_age[is.na(df\_median$person\_age)] <- median(df\_median$person\_age,na.rm = TRUE);

head(df\_median)

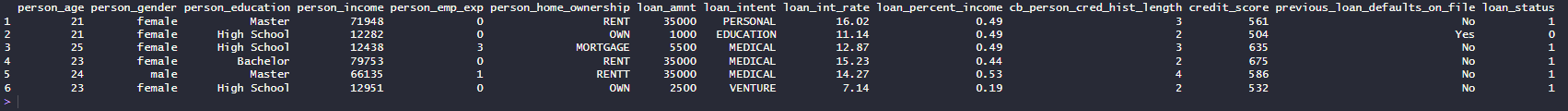


* Replace with mode

mode\_val <- mlv(df\_mode$loan\_status,method = "mfv",na.rm = TRUE);

df\_mode$loan\_status[is.na(df\_mode$loan\_status)] <- mode\_val

head(as.data.frame(df\_mode))



2. see missing values on a graph

vis\_miss(data)

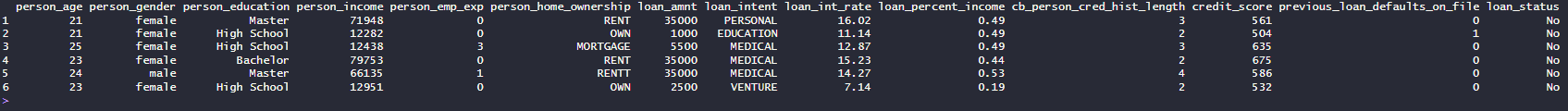
A graph with text on it

AI-generated content may be incorrect.

3. convert attributes from numeric to categorical or categorical to numeric

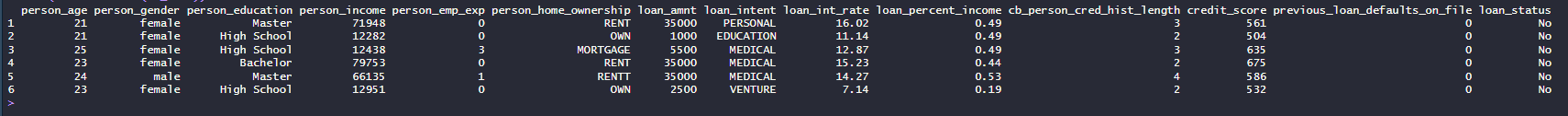
df\_mode$loan\_status <-ifelse(df\_mode$loan\_status == 1 ,"Yes","No");

head(df\_mode)



df\_mode$previous\_loan\_defaults\_on\_file <- ifelse(df\_mode$previous\_loan\_defaults\_on\_file == "Yes", 1,0);

head(df\_mode)

****

4.

get\_outliers <- function(data, colname) {

if (!colname %in% names(data)) stop("Column not found.")

if (!is.numeric(data[[colname]])) stop("Column must be numeric.")

Q1 <- quantile(data[[colname]], 0.25, na.rm = TRUE)

Q3 <- quantile(data[[colname]], 0.75, na.rm = TRUE)

IQR <- Q3 - Q1

lower <- Q1 - 1.5 \* IQR

upper <- Q3 + 1.5 \* IQR

# return only the outlier values from dataset

outliers <- data[[colname]][data[[colname]] < lower | data[[colname]] > upper]

return(outliers)

}

outlier\_detect<-get\_outliers(df, "person\_age")

head(outlier\_detect)



fix\_outliers <- function(data, colname) {

if (!colname %in% names(data)) stop("Column not found.")

if (!is.numeric(data[[colname]])) stop("Column must be numeric.")

Q1 <- quantile(data[[colname]], 0.25, na.rm = TRUE)

Q3 <- quantile(data[[colname]], 0.75, na.rm = TRUE)

IQR <- Q3 - Q1

lower <- Q1 - 1.5 \* IQR

upper <- Q3 + 1.5 \* IQR

is\_outlier <- data[[colname]] < lower | data[[colname]] > upper

n\_out <- sum(is\_outlier, na.rm = TRUE)

if (n\_out > 0) {

data[[colname]] <- pmin(pmax(data[[colname]], lower), upper)

message("Fixed ", n\_out, " outliers in ", colname,

" (capped to [", round(lower, 2), ", ", round(upper, 2), "])")

} else {

message("No outliers to fix in ", colname)

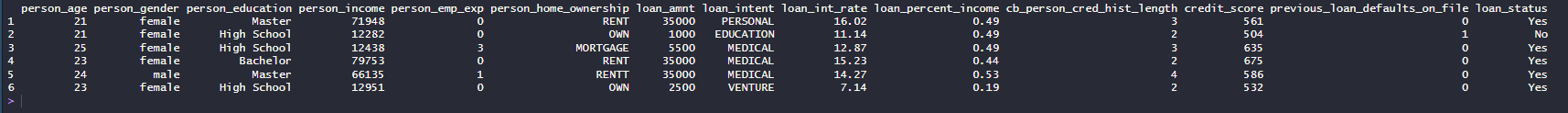
}

return(data)

}

df\_clean <- fix\_outliers(df, "person\_age")

head(as.data.frame(df\_clean))

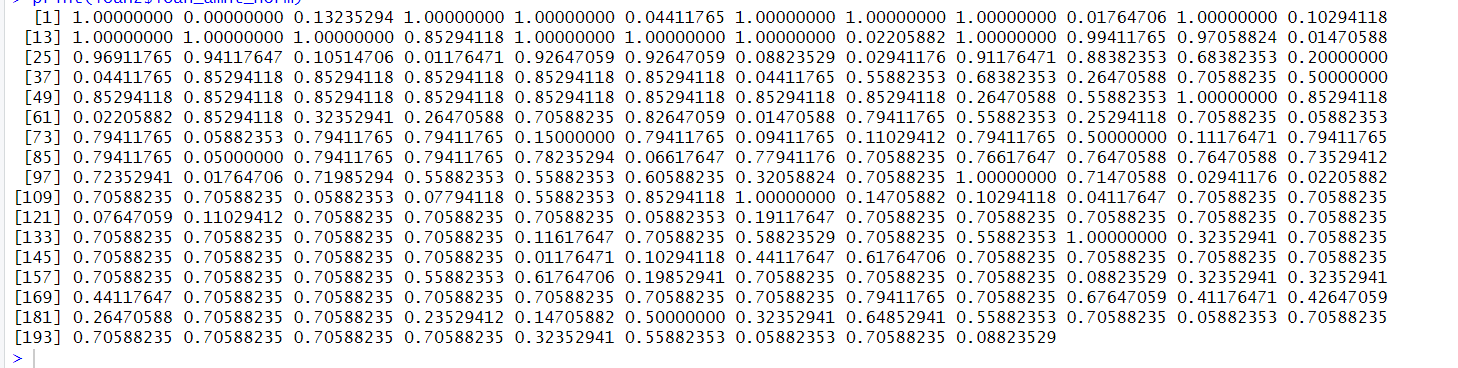
****

Task 5: Normalization method for any continuous attribute.

Code:

|  |
| --- |
| loan2 <- Loan\_Datas  loan2$loan\_amnt\_norm <- (loan2$loan\_amnt - min(loan2$loan\_amnt, na.rm = TRUE)) /  (max(loan2$loan\_amnt, na.rm = TRUE) - min(loan2$loan\_amnt, na.rm = TRUE))  print(loan2$loan\_amnt\_norm) |

**Output:**



**Description:**

Here used normalization using one equation that is current\_value - min\_value/ max\_value - min\_value

Task 6: Find and remove duplicate rows to reduce congestion.

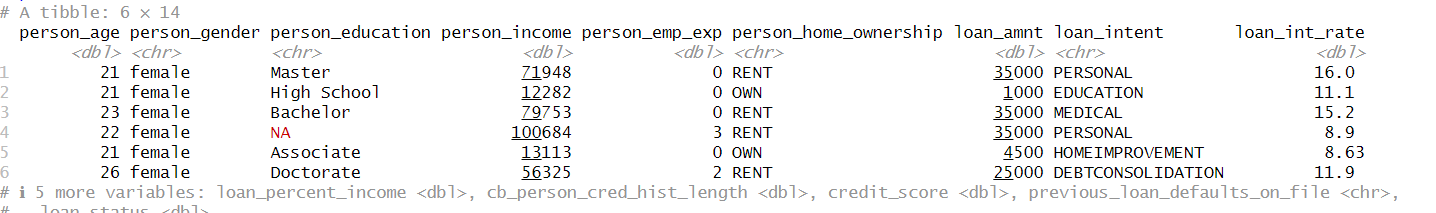
loan3 <- Loan\_Datas

library(dplyr)

distinct\_data <- distinct(loan3, person\_education, .keep\_all = TRUE)

print(distinct\_data)

**Output:**



**Description:**

**Task 7**: Filter Data to find specific numerical value , row, categorical value

loan4<-Loan\_Datas

filtered\_data <- filter(loan3, loan\_amnt > 10000)

print(filtered\_data)

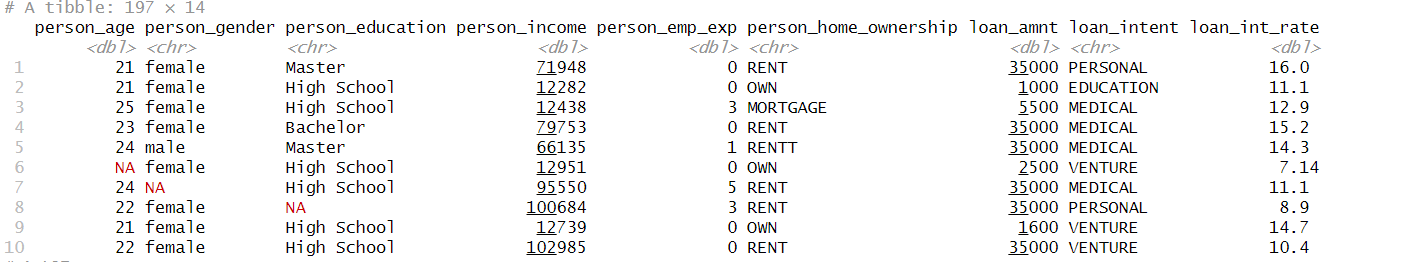
filtered\_data <- filter(loan3, !is.na(person\_income) & person\_income > 20000)

print(filtered\_data)

filtered\_data <- filter(loan3, person\_education %in% c("Bachelor", "Master"))

print(filtered\_data)

**Output:**



**Description:**

We used filter method we can use multiple operation for filter that is more than 10000 value or whether the value is null or not If is null then delete this row. Then filter only master, bachelor row of person\_income column. Also filter more than 20000 of person\_income and without null value.

**Task8:** Detect invalid value then remove this row or replace mean value

loans5<-Loan\_Datas

loans5 <- na.omit(loans5)

print(nrow(loans5))

print(loans5)

loan5 <- Loan\_Datas

loan3\_clean <- na.omit(loan5)

print(nrow(loan3\_clean))

loan3\_clean

num\_cols <- sapply(loan5, is.numeric)

cat\_cols <- sapply(loan5, is.character)

loan5[num\_cols] <- lapply(loan5[num\_cols], function(x) {

x[is.na(x)] <- mean(x, na.rm = TRUE)

return(x)

})

loan5[cat\_cols] <- lapply(loan5[cat\_cols], function(x) {

mode\_val <- names(sort(table(x), decreasing = TRUE))[1]

x[is.na(x)] <- mode\_val

return(x)

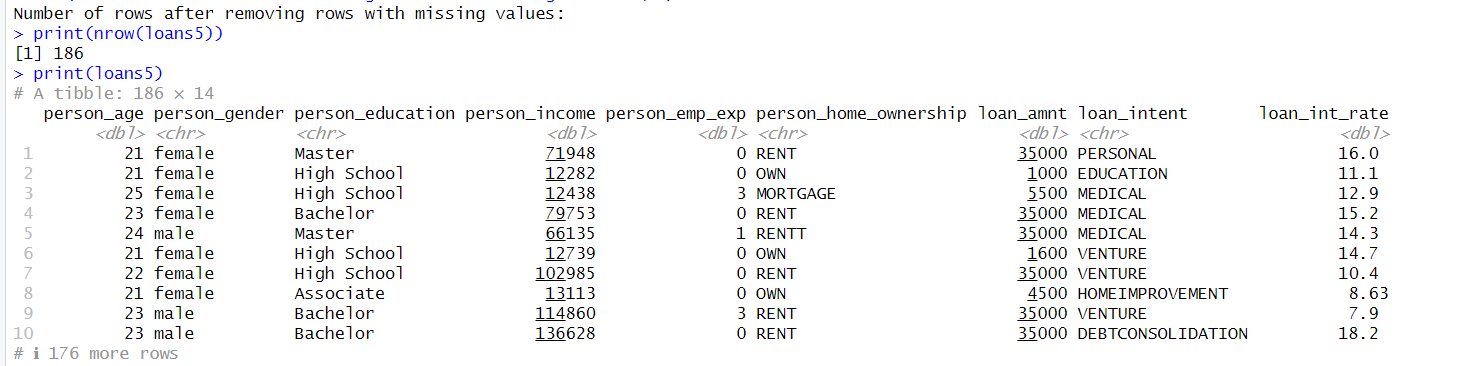
})

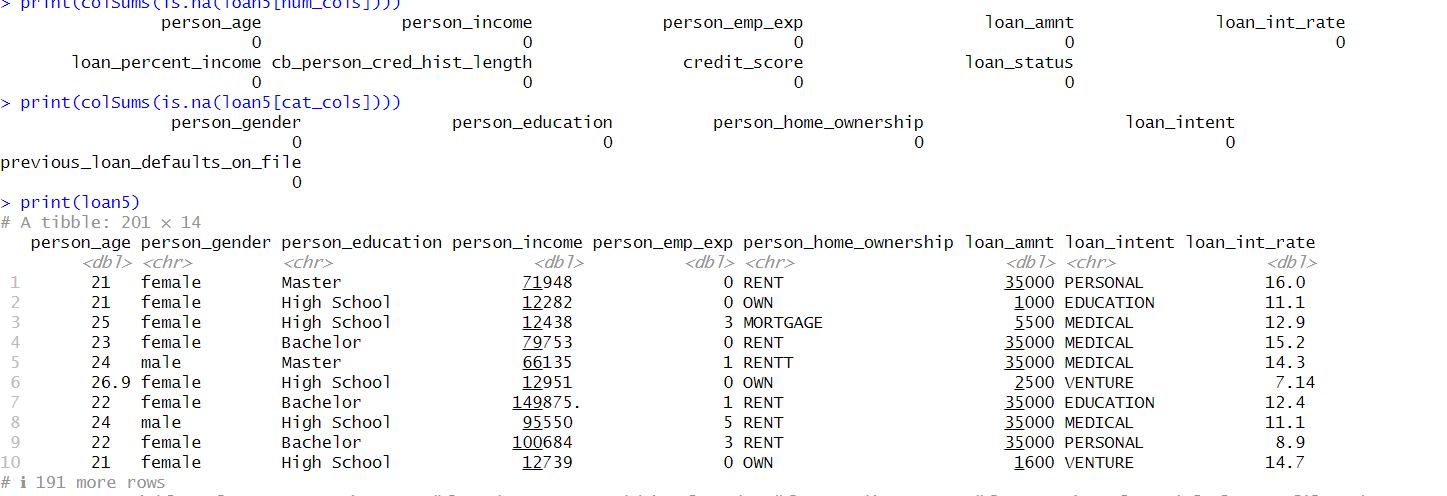
print(colSums(is.na(loan5[num\_cols])))

print(colSums(is.na(loan5[cat\_cols])))

print(loan5)

**Output:**





**Description:**

First delete null value . Then assign numeric value in num\_cols and categorical value in cat\_cols. lapply() applies a function to each element of a list. Then use mean function to find out mean value and replace this all by mean value. Table(x) counts the frequency of each unique value in the column x. Sort( , decreasing = TRUE) sorts the frequencies from highest to lowest. names()[1] extracts the name (value) of the most frequent category, the mode. So mode\_val is the most common category in column x.

**Task 10:** Split the dataset for Training and Testing , 70% row for Training data and 30% for Testing data

set.seed(123)

n <- nrow(laons5)

train\_index <- sample(1:n, size = 0.7 \* n)

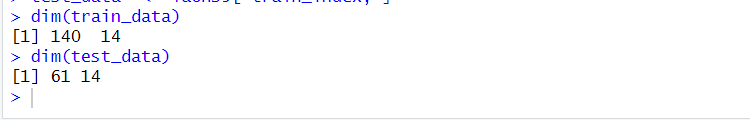
train\_data <- laons5[train\_index, ]

test\_data <- laons5[-train\_index, ]

dim(train\_data)

dim(test\_data)

Output:



**Description:**

**11.** statistics and interpret the results for the following numerical

variables between two target classes (loan\_status = 1 and loan\_status = 0)

➢ person\_age

➢ Person Income

age\_summary <- aggregate(person\_age ~ loan\_status, data = df,

FUN = function(x) c(mean = mean(x), median = median(x),

sd = sd(x), min = min(x), max = max(x)))

age\_summary <- do.call(data.frame, age\_summary)

# Create summary table for Income

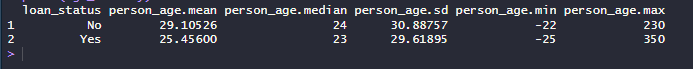
income\_summary <- aggregate(person\_income ~ loan\_status, data = df,

FUN = function(x) c(mean = mean(x), median = median(x),

sd = sd(x), min = min(x), max = max(x)))

income\_summary <- do.call(data.frame, income\_summary)

print(age\_summary)



print(income\_summary)

A screenshot of a computer

AI-generated content may be incorrect.

**Task 12:**

**Code:**

descriptive\_stats <- loan5\_balanced %>%

group\_by(loan\_status) %>%

summarise(

mean\_age = mean(person\_age, na.rm = TRUE),

median\_age = median(person\_age, na.rm = TRUE),

sd\_age = sd(person\_age, na.rm = TRUE),

min\_age = min(person\_age, na.rm = TRUE),

max\_age = max(person\_age, na.rm = TRUE),

mean\_income = mean(person\_income, na.rm = TRUE),

median\_income = median(person\_income, na.rm = TRUE),

sd\_income = sd(person\_income, na.rm = TRUE),

min\_income = min(person\_income, na.rm = TRUE),

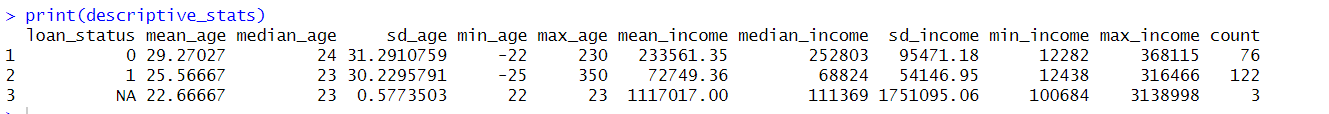
max\_income = max(person\_income, na.rm = TRUE),

count = n() # number of rows per class

)

print(descriptive\_stats)

Output:



**Description:**

Here group by function used for find out all value such as mean, median, mode, max, min for each group. Here has 0 and 1 group. Summarise function use computing statistics like mean, median, sum, count etc.

**13.** **Compare spread in person\_emp\_exp for customers with different levels of**

**person\_education**

compare\_spread <- function(data, group\_col, value\_col) {

library(dplyr)

if (!group\_col %in% names(data)) stop("Group column not found.")

if (!value\_col %in% names(data)) stop("Value column not found.")

data %>%

group\_by(.data[[group\_col]]) %>%

summarise(

count = n(),

mean = mean(.data[[value\_col]], na.rm = TRUE),

sd = sd(.data[[value\_col]], na.rm = TRUE),

min = min(.data[[value\_col]], na.rm = TRUE),

max = max(.data[[value\_col]], na.rm = TRUE),

IQR = IQR(.data[[value\_col]], na.rm = TRUE),

.groups = "drop"

)

}

aggregate(credit\_score ~ loan\_status, data = data, mean, na.rm = TRUE)

A screenshot of a computer code

AI-generated content may be incorrect.

compare <- compare\_spread(df\_clean, "person\_education", "person\_emp\_exp")

head(as.data.frame(compare))

A screenshot of a computer screen

AI-generated content may be incorrect.