Loan Approval Classification Dataset:

The Loan Approval Classification Dataset contains 45,000 records and 14 variables, designed to model and predict whether a loan application will be approved or rejected.The dataset includes both numerical variables (person's age, income, employment length, loan amount, interest rate, credit score) and categorical variables are home ownership status, loan purpose, loan grade, credit default history. The target variable is loan_status, a binary indicator where 1 = approved and 0 = rejected.

The sample dataset is derived from the Loan Approval Classification dataset and contains 201 records with a mix of numerical and categorical variables related to loan applications. Unlike the full dataset, this sample includes 17 missing values distributed across five variables: person_age (4), person_gender (4), person_income (4), person_loan_status (3), and person_education (2). Our goal is to apply data preprocessing techniques, handling missing values and exploratory data analysis.

Code Description:

install.packages("dplyr")

library(dplyr)

library(ggplot2)

Installed and loaded the "dplyr" and "ggplot2" packages for data manipulation and visualization.

Code:

mydata<-read.csv("C:/R/Midterm_Dataset_Section(A).csv",header=TRUE,sep=",")

print(mydata)

```
   person_age person_gender person_education person_income
1          21        female            Master         71948
2          21        female       High School         12282
3          25        female       High School         12438
4          23        female          Bachelor         79753
5          24          male            Master         66135
6          NA        female       High School         12951
7          22        female          Bachelor            NA
8          24                     High School         95550
9          22        female                            100684
10         21        female       High School         12739
```

This code reads mydata.csv dataset into the R studio as mydata and prints all rows and columns.

Code:

colSums(is.na(mydata) | mydata == "")

```
> colSums(is.na(mydata) | mydata == "" )
                    person_age                  person_gender               person_education                  person_income 
                             4                              4                              2                              4 
                person_emp_exp             person_home_ownership                     loan_amnt                    loan_intent 
                             0                              0                              0                              0 
                 loan_int_rate           loan_percent_income cb_person_cred_hist_length                   credit_score 
                             0                              0                              0                              0 
previous_loan_defaults_on_file                    loan_status 
                             0                              3 
```

This code counts the number of missing values in each column for both NA and empty strings ("") as missing as missing value.

Code:

colSums(is.na(mydata) | mydata == "")

summary(mydata)

missing_data <- data.frame(

variable = names(mydata),

missing_count = sapply(mydata, function(x) sum(is.na(x) | x == ""))

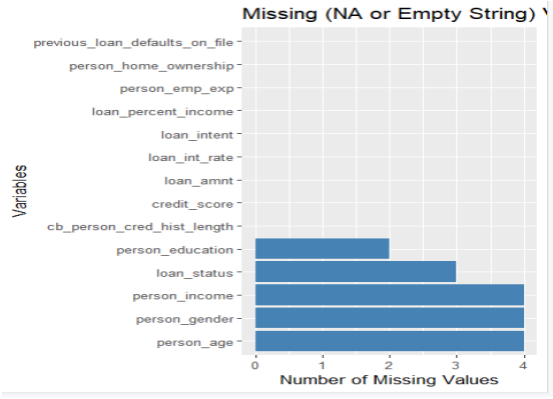
)

Counts missing values in each column, considering both NA and empty strings ("")

ggplot(missing_data, aes(x = reorder(variable, -missing_count), y = missing_count)) +

geom_bar(stat = "identity", fill = "steelblue") +

coord_flip() +labs(title = "Missing (NA or Empty String) Values per Variable", x = "Variables", y = "Number of Missing Values")



Visualize the number of missing values in a bar plot.

Code:

duplicated(mydata)

```
> duplicated(mydata)
  [1] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
 [21] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE  TRUE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
 [41] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
 [61] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
 [81] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
[101] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
[121] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
[141] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
[161] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
[181] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
[201] FALSE
>
```

If there are any duplicate row it will show TRUE, else it will show FALSE

Code:

mean_age <- mean(mydata$person_age, na.rm = TRUE)

mean_age <-as.integer(round(mean_age))

mydata$person_age[is.na(mydata$person_age)] <- mean_age

➔ This code calculates mean value with mean() function. Then fills missing person_age values with the rounded mean age.

mode_gender <- names(sort(table(mydata$person_gender),decreasing = TRUE))[1]

mydata$person_gender[mydata$person_gender == ""] <- mode_gender

➔ This code calculates the most frequent gender (mode) and replaces all empty strings in the person_gender column with this mode.

mode_education <- names(sort(table(mydata$person_education),decreasing = TRUE))[1]

mydata$person_education[mydata$person_education == ""] <- mode_education

➔ This code finds the most frequent value (mode) in person_education and replaces all empty strings in that column with the mode.

mean_income <- mean(mydata$person_income, na.rm = TRUE)

mean_income <-as.integer(round(mean_income))

mydata$person_income[is.na(mydata$person_income)] <- mean_income

➔ This code computes the mean of person_income rounds it to the nearest integer, and replaces all missing person_income with this mean.

unique(mydata$loan_status)

mode_loan_status <- names(sort(table(mydata$loan_status),decreasing = TRUE))[1]

mydata$loan_status[is.na(mydata$loan_status)] <-mode_loan_status

➔ This code finds the mode in loan_status and replaces with the mode.

After handling missing value run,

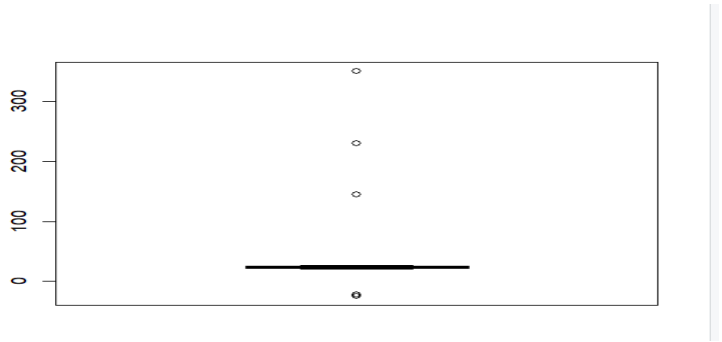
colSums(is.na(mydata) | mydata == "")

```
> colSums(is.na(mydata) | mydata == "" )
                    person_age                 person_gender              person_education                 person_income
                             0                             4                             2                             4
                person_emp_exp          person_home_ownership                     loan_amnt                    loan_intent
                             0                             0                             0                             0
                 loan_int_rate            loan_percent_income cb_person_cred_hist_length                  credit_score
                             0                             0                             0                             0
previous_loan_defaults_on_file                   loan_status
                             0                             3
>
```

No missing value in our dataset.

Code: Outliers

boxplot(mydata$person_age)



➔ 4 outleir in person_age

quantile(mydata$person_age)

```
> quantile(mydata$person_age)
  0%  25%  50%  75% 100%
 -25   22   23   25  350
```

➔ Q1=22, Q3=25

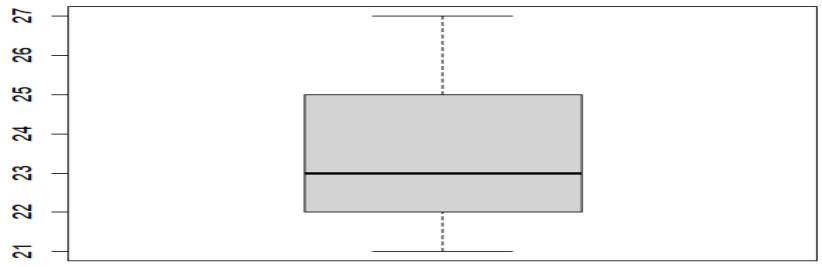
x <- mydata$person_age

x

```
iqr <- IQR(x)

iqr

lower_bound_x <-  22 -  1.5 * iqr

upper_bound_x <-  25 + 1.5 * iqr

x[x < lower_bound_x | x > upper_bound_x] <- mean(x)

x <- as.integer(round(x))

mydata$person_age<- x
```

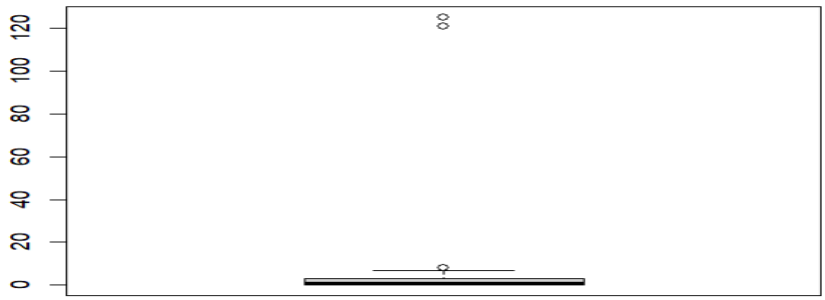
➔ This code calculates the IQR of person_age, determines outlier bounds, replaces values outside these bounds with the mean

```
boxplot(mydata$person_age)
```



➔ Outliers removed

```
boxplot(mydata$person_emp_exp)
```



➔ person_emp_exp has three outlier

```
boxplot(mydata$person_emp_exp)

quantile(mydata$person_emp_exp)

exp <- mydata$person_emp_exp
```

exp

iqr_exp <- IQR(exp)

iqr_exp

lower_bound_exp <- 0 - 1.5 * iqr_exp

upper_bound_exp <- 3 + 1.5 * iqr_exp

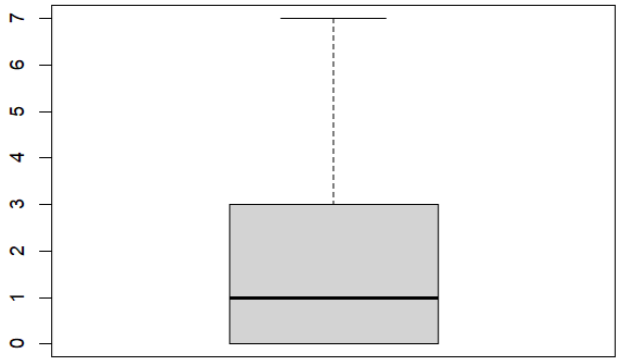
exp[exp < lower_bound_exp | exp > upper_bound_exp] <- mean(exp)

exp <- as.integer(round(exp)

mydata$person_emp_exp<- exp

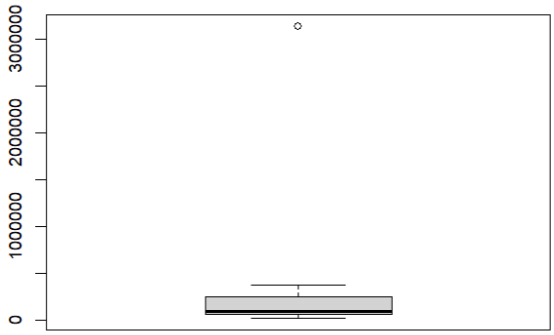
➔ This code calculates the IQR of person_emp_exp, determines outlier bounds, replaces values outside these bounds with the mean

boxplot(mydata$person_emp_exp)



➔ Outliers removed

boxplot(mydata$person_income)

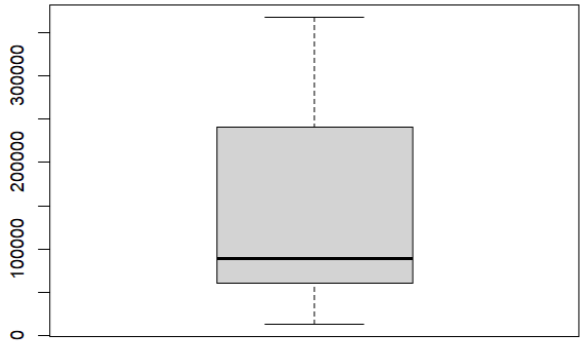


-> person_income has outlier

```
quantile(mydata$person_income)

i <- mydata$person_income

iqr_i <- IQR(i)

iqr_i

lower_bound_i <-  60747 -  1.5 * iqr_i

upper_bound_i <-  149875 + 1.5 * iqr_i

mean(i)

i[i < lower_bound_i | i > upper_bound_i] <- mean(i)

i <- as.integer(round(i))

mydata$person_income<-i
```

➔ This code calculates the IQR of person_income, determines outlier bounds, replaces values outside these bounds with the mean

```
boxplot($person_income)
```



➔ Outlier removed

Code: Noisy Value

```
unique(mydata$person_gender)
```

```
> unique(mydata$person_gender)
[1] "female"  "male"    ""        "malee"   "feemale"
```

➔ Here “malee” and “feemale” are noisy value

```
mydata <- mydata %>%
 mutate(gender = case_when(
  tolower(person_gender) %in% c("male", "malee") ~ "male",
  tolower(person_gender) %in% c("female", "feemale") ~ "female"
 ))
```

```
mydata$person_gender<-mydata$gender
mydata$gender <- NULL
```

➔ This code creates a new Gender collum, convert and store misspelled entries into valid format. After that it replaced the values of person_gende with Gender collum. Then replaced the value of gender collum with Null and Gender collum will be removed.

```
unique(mydata$person_gender)
```

```
> unique(mydata$person_gender)
[1] "female" "male"
```

-> Noisy value replaced

```
unique(mydata$person_home_ownership)
```

```
> unique(mydata$person_home_ownership)
[1] "RENT"     "OWN"      "MORTGAGE" "RENTT"    "OOWN"     "OTHER"
>
```

➔ RENTT , OOWN are noisy value

```
unique(mydata$person_home_ownership)
mydata <- mydata %>%
 mutate(home = case_when(
  toupper(person_home_ownership) %in% c("RENT", "RENTT") ~ "RENT",
```

```
  toupper(person_home_ownership) %in% c("OWN", "OOWN") ~ "OWN",
  toupper(person_home_ownership) %in% c("MORTGAGE") ~ "MORTGAGE",
  toupper(person_home_ownership) %in% c("OTHER") ~ "OTHER" ))
mydata$person_home_ownership<-mydata$home
mydata$home <- NULL
```

➔ This code creates a new Homecollum, convert and store misspelled entries into valid format. After that it replaced the values ofperson_home_ownership with Home collum. Then replaced the value of Home collum with Null and Gender collum will be removed.

➔

unique(mydata$person_home_ownership)

```
> unique(mydata$person_home_ownership)
[1] "RENT"     "OWN"      "MORTGAGE" "OTHER"
```

-> Noisy value removed

Categorical to Numeric:

```
mydata$person_gender <- factor(mydata$person_gender,
                               levels = c("female", "male"),labels = c(1,2))
```

```
> unique(mydata$person_gender)
[1] "female" "male"
```

```
> unique(mydata$person_gender)
[1] 1 2
Levels: 1 2
```

➔ Converted “female” , “male” to 0, 1.

unique(mydata$previous_loan_defaults_on_file)

```
mydata$previous_loan_defaults_on_file <- factor(mydata$previous_loan_defaults_on_file,
                                                levels = c("No", "Yes"),labels = c(0,1))
```

```
> unique(mydata$previous_loan_defaults_on_file)
[1] "No"  "Yes"
```

```
> unique(mydata$previous_loan_defaults_on_file)
[1] 0 1
Levels: 0 1
```

➔ Converted “No” , “Yes” to 0, 1.

Numeric to Categorical:

unique(mydata$loan_status)

mydata$loan_status <- factor(mydata$loan_status,

levels = c(0,1),labels = c("No","Yes"))

```
> unique(mydata$loan_status)
[1] "1" "0"
```

```
> unique(mydata$loan_status)
[1] Yes No
Levels: No Yes
```

-> Converted 1,0 to “Yes”, “No”

Code: Min-Max Normalization

normalization <- function(n) {

(n - min(n, na.rm = TRUE)) / (max(n, na.rm = TRUE) - min(n, na.rm = TRUE))

}

mydata$person_income <- normalization(mydata$person_income)

➔ This code defines a min-max normalization function to scale numeric values between 0 and 1 and applies it to the person_income column.

Code: Filter data

filtered_data1 <- mydata %>%

filter(loan_amnt > 20000 | person_age < 20)

filtered_data1

str(filtered_data1)

```
> str(filtered_data1)
'data.frame':   126 obs. of  14 variables:
 $ person_age                    : int  21 23 24 22 24 22 22 23 27 23 ...
 $ person_gender                 : Factor w/ 2 levels "1","2": 1 1 2 1 2 1 1 2 2 1 ...
 $ person_education              : chr  "Master" "Bachelor" "Master" "Bachelor" ...
 $ person_income                 : int  71948 79753 66135 149875 95550 100684 102985 114860 130713 149875 ...
 $ person_emp_exp                : int  0 0 1 1 5 3 0 3 0 0 ...
 $ person_home_ownership         : chr  "RENT" "RENT" "RENT" "RENT" ...
 $ loan_amnt                     : int  35000 35000 35000 35000 35000 35000 35000 35000 35000 35000 ...
 $ loan_intent                   : chr  "PERSONAL" "MEDICAL" "MEDICAL" "EDUCATION" ...
 $ loan_int_rate                 : num  16 15.2 14.3 12.4 11.1 ...
 $ loan_percent_income           : num  0.49 0.44 0.53 0.37 0.37 0.35 0.34 0.3 0.27 0.25 ...
 $ cb_person_cred_hist_length    : int  3 2 4 3 4 2 4 2 4 4 ...
 $ credit_score                  : int  561 675 586 701 585 544 621 573 708 583 ...
 $ previous_loan_defaults_on_file: Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 1 1 1 ...
 $ loan_status                   : Factor w/ 2 levels "No","Yes": 2 2 2 2 2 2 2 2 2 2 ...
>
```

➔ This code filters the dataset to include rows where loan_amnt is greater than 20,000 or person_age is less than 20, then displays the filtered data and its structure.

```
filtered_data2 <- mydata %>%
 filter(loan_status == "Yes" & person_age > 20)
filtered_data2
str(filtered_data2)
```

```
> str(filtered_data2)
'data.frame':   125 obs. of  14 variables:
 $ person_age                    : int  21 25 23 24 27 22 24 22 21 22 ...
 $ person_gender                 : Factor w/ 2 levels "1","2": 1 1 1 2 1 1 2 1 1 1 ...
 $ person_education              : chr  "Master" "High School" "Bachelor" "Master" ...
 $ person_income                 : int  71948 12438 79753 66135 12951 149875 95550 100684 12739 102985 ...
 $ person_emp_exp                : int  0 3 0 1 0 1 5 3 0 0 ...
 $ person_home_ownership         : chr  "RENT" "MORTGAGE" "RENT" "RENT" ...
 $ loan_amnt                     : int  35000 5500 35000 35000 2500 35000 35000 35000 1600 35000 ...
 $ loan_intent                   : chr  "PERSONAL" "MEDICAL" "MEDICAL" "MEDICAL" ...
 $ loan_int_rate                 : num  16.02 12.87 15.23 14.27 7.14 ...
 $ loan_percent_income           : num  0.49 0.49 0.44 0.53 0.19 0.37 0.37 0.35 0.13 0.34 ...
 $ cb_person_cred_hist_length    : int  3 3 2 4 2 3 4 2 3 4 ...
 $ credit_score                  : int  561 635 675 586 532 701 585 544 640 621 ...
 $ previous_loan_defaults_on_file: Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 1 1 1 ...
 $ loan_status                   : Factor w/ 2 levels "No","Yes": 2 2 2 2 2 2 2 2 2 2 ...
>
```

➔ This code selects rows where loan_status is "Yes" and person_age is greater than 20, then shows the filtered dataset and its structure.

Sampling: Oversampling method

```
table(mydata$person_gender)
 balanced_data_over <- mydata %>%
  group_by(person_gender) %>%
```

```
  slice_sample(n = max(table(mydata$person_gender)), replace = TRUE) %>%
  ungroup()
```

mydata | clean_data | Untitled1* | iqr_removed | DataScienceProject.R* | balanced_data_over | class_distribution_balanced | filtered_data2 | filt

Filter

	person_age	person_gender	person_education	person_income	person_emp_exp	person_home_ownership	loan_amnt	loan_intent	loan_int_rate	loan_percen
1	22	1	Bachelor	79054	1	RENT	32500	HOMEIMPROVEMENT	9.99	
2	22	1	Master	79255	0	RENT	34000	EDUCATION	17.58	
3	24	1	Bachelor	83853	3	RENT	27600	HOMEIMPROVEMENT	11.01	
4	23	1	Associate	79183	0	RENT	28000	EDUCATION	7.90	
5	22	1	Associate	72608	3	RENT	25000	PERSONAL	10.99	
6	23	1	High School	86811	0	RENT	30000	DEBTCONSOLIDATION	11.01	
7	23	1	High School	111369	0	RENT	35000	MEDICAL	20.00	
8	23	1	Associate	117250	0	RENT	30000	VENTURE	10.65	
9	23	1	Bachelor	17149	0	RENT	4950	DEBTCONSOLIDATION	7.90	
10	22	1	High School	68824	0	RENT	25000	HOMEIMPROVEMENT	14.96	

-> This code counts the number of records for each gender and then creates a balanced dataset by oversampling the minority gender to match the count of the majority gender.

```
 table(balanced_data_over$person_gender)
 class_distribution_balanced <- balanced_data_over %>%
  count(person_gender) %>%
        mutate(percentage = n / sum(n) * 100)
```

Filter

	person_gender	n	percentage
1	1	123	50
2	2	123	50

➔ This code shows the number of records for each gender in the balanced dataset and creates a table with both counts and their percentage of the total.

Test and Training dataset

```
set.seed(123)
train_data <- mydata %>%
  sample_frac(0.8)
test_data <- mydata %>%
  anti_join(train_data)
 str(train_data)
```

str(test_data)

-> This code splits the dataset into a training set (80% of the data) and a test set (remaining 20%), then displays the structure of the training set.

```
>   str(train_data)
'data.frame':   161 obs. of  14 variables:
 $ person_age                    : int  26 22 27 24 25 22 23 24 26 26 ...
 $ person_gender                 : Factor w/ 2 levels "1","2": 2 2 2 1 2 1 2 1 1 2 ...
 $ person_education              : chr  "Associate" "High School" "Master" "Associate" ...
 $ person_income                 : int  68066 241030 130713 82741 65742 80838 16599 15082 241114 22674
 $ person_emp_exp                : int  2 3 0 0 3 0 0 0 5 0 ...
 $ person_home_ownership         : chr  "RENT" "RENT" "RENT" "RENT" ...
 $ loan_amnt                     : int  25000 15000 35000 25000 25000 30000 2400 2500 3000 20000 ...
 $ loan_intent                   : chr  "DEBTCONSOLIDATION" "EDUCATION" "EDUCATION" "PERSONAL" ...
 $ loan_int_rate                 : num  12.9 14.8 18.4 14.1 16.8 ...
 $ loan_percent_income           : num  0.37 0.06 0.27 0.3 0.38 0.37 0.14 0.17 0.01 0.09 ...
 $ cb_person_cred_hist_length    : int  2 3 4 3 4 3 2 3 3 2 ...
 $ credit_score                  : int  637 626 708 688 672 645 505 631 608 621 ...
 $ previous_loan_defaults_on_file: Factor w/ 2 levels "0","1": 1 2 1 1 1 1 1 1 2 1 ...
 $ loan_status                   : Factor w/ 2 levels "No","Yes": 2 1 2 2 2 2 2 2 1 2 ...
```

➔ Tain data total 161 obs

```
>   str(test_data)
'data.frame':   40 obs. of  14 variables:
 $ person_age                    : int  21 23 23 23 27 27 22 26 26 23 ...
 $ person_gender                 : Factor w/ 2 levels "1","2": 1 1 1 2 2 2 2 2 2 2 ...
 $ person_education              : chr  "High School" "Associate" "High School" "Bachelor" ...
 $ person_income                 : int  12739 149875 111369 136628 14293 144855 361547 360977 337133 333399 ...
 $ person_emp_exp                : int  0 0 0 0 0 1 0 5 7 1 ...
 $ person_home_ownership         : chr  "OWN" "RENT" "RENT" "RENT" ...
 $ loan_amnt                     : int  1600 35000 35000 35000 1400 32000 24250 25000 10000 35000 ...
 $ loan_intent                   : chr  "VENTURE" "EDUCATION" "MEDICAL" "DEBTCONSOLIDATION" ...
 $ loan_int_rate                 : num  14.74 7.9 20 18.25 9.32 ...
 $ loan_percent_income           : num  0.13 0.25 0.31 0.26 0.1 0.22 0.07 0.07 0.03 0.1 ...
 $ cb_person_cred_hist_length    : int  3 4 4 4 3 2 2 3 3 4 ...
 $ credit_score                  : int  640 583 694 709 607 586 637 695 623 609 ...
 $ previous_loan_defaults_on_file: Factor w/ 2 levels "0","1": 1 1 1 1 1 2 2 2 2 1 ...
 $ loan_status                   : Factor w/ 2 levels "No","Yes": 2 2 2 2 1 1 1 1 1 1 ...
```

➔ Test data total 40 obs

Descriptive Statistics

```
summarize_age_income<- mydata %>%
  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)
  )
summarize_age_income
```

```
summarize_age_income
A tibble: 2 × 11
loan_status mean_age median_age sd_age min_age max_age mean_income median_income sd_income min_income max_income
<fct>          <dbl>      <dbl>  <dbl>   <int>   <int>       <dbl>         <dbl>     <dbl>      <int>      <int>
No              24.1         24   1.76      21      27     232460.        249174    95317.      12282     368115
Yes             23.4         23   1.69      21      27      75750.         72608    54691.      12438     316466
```

Average credit_score between customers with loan_status 1 and with loan_status 0

```
avg_credit <- mydata %>%
 group_by(loan_status) %>%
 summarise(
   mean_credit  = mean(credit_score, na.rm = TRUE),
   median_credit = median(credit_score, na.rm = TRUE),
   sd_credit    = sd(credit_score, na.rm = TRUE),
   count       = n()
 )
avg_credit
```

```
> avg_credit
# A tibble: 2 × 5
  loan_status mean_credit median_credit sd_credit count
  <fct>             <dbl>         <dbl>     <dbl> <int>
1 No                 630.          624.      52.2    76
2 Yes                628.          634       50.0   125
>
```

Spread in person_emp_exp for customers with different levels of person_education

```
emp_exp_stats <- mydata %>%
 group_by(person_education) %>%
 summarise(
  mean_exp   = mean(person_emp_exp, na.rm = TRUE),
  median_exp = median(person_emp_exp, na.rm = TRUE),
  sd_exp     = sd(person_emp_exp, na.rm = TRUE),
  min_exp    = min(person_emp_exp, na.rm = TRUE),
  max_exp    = max(person_emp_exp, na.rm = TRUE),
  IQR_exp    = IQR(person_emp_exp, na.rm = TRUE),
  n          = n()
 )
emp_exp_stats
unique(mydata$person_education)
```

```
  person_education mean_exp median_exp sd_exp min_exp max_exp IQR_exp     n
  <chr>               <dbl>      <dbl>  <dbl>   <int>   <int>   <dbl> <int>
1 Associate            1.20          1   1.41       0       5     2      46
2 Bachelor             1.79          1   1.72       0       5     3      73
3 Doctorate            2             2  NA          2       2     0       1
4 High School          1.41          1   1.80       0       7     3      58
5 Master               1.74          1   1.89       0       6     2.5    23
> unique(mydata$person_education)
[1] "Master"      "High School" "Bachelor"    "Associate"   "Doctorate"
```