# Description of the dataset

The Loan Approval Classification dataset is a synthetic dataset created for binary classification tasks focused on loan approval decisions. It contains information about individuals applying for loans, described by 14 attributes. These attributes provide a comprehensive profile of each applicant and their loan application details. The dataset includes the applicant's age (person_age), gender (person_gender), highest education level (person_education), annual income (person_income), and years of employment experience (person_emp_exp). Additionally, it captures the applicant's home ownership status (person_home_ownership) and their creditworthiness through the credit history length (cb_person_cred_hist_length) and credit score (credit_score).The dataset also includes information specific to the loan application, such as the loan amount requested (loan_amnt), the purpose of the loan (loan_intent), the loan interest rate (loan_int_rate), and the loan amount as a percentage of the applicant's annual income (loan_percent_income). An indicator of previous loan defaults (previous_loan_defaults_on_file) is also provided. The target variable, loan_status, indicates the final approval status of the loan, where 1 represents approval and 0 represents rejection.

This dataset contains both numerical and categorical data, offering a rich set of features for exploring and modeling factors that influence loan approval decisions.

# Loading the dataset

**Code**

```
install.packages("dplyr")

library(dplyr)

install.packages("readxl");

library(readxl);

dataSet_1 <- read_excel("E:/FALL 24-25/INTRODUCTION TO DATA SCIENCE/MID/Mid Term
Project/Metarials/Midterm_Dataset_Section(C).xlsx");

print(dataSet_1, n = nrow(dataSet_1));
```

**Output**

```
Console   Terminal ×   Background Jobs ×
R 4.2.1 · E:/FALL 24-25/INTRODUCTION TO DATA SCIENCE/MID/Mid Term Project/
> library(dplyr)
> library(readxl);
> dataSet_1 <- read_excel("E:/FALL 24-25/INTRODUCTION TO DATA SCIENCE/MID/Mid Term Project/Metarials/Midterm_Dataset_Section(C).xlsx");
> print(dataSet_1, n = nrow(dataSet_1));
# A tibble: 201 × 14
   person_age person_gender person_education person_income person_emp_exp person_home_ownership loan_amnt loan_intent       loan_int_rate
        <dbl> <chr>         <chr>                    <dbl>          <dbl> <chr>                     <dbl> <chr>                     <dbl>
 1         21 female        Master                   71948              0 RENT                      35000 PERSONAL                  16.0
 2         21 female        High School              12282              0 OWN                        1000 EDUCATION                 11.1
 3         25 female        High School              12438              3 MORTGAGE                   5500 MEDICAL                   12.9
 4         23 female        Bachelor                 79753              0 RENT                      35000 MEDICAL                   15.2
 5         24 male          Master                   66135              1 RENTT                     35000 MEDICAL                   14.3
 6         NA female        High School              12951              0 OWN                        2500 VENTURE                    7.14
 7         22 female        Bachelor                    NA              1 RENT                      35000 EDUCATION                 12.4
 8         24 NA            High School              95550              5 RENT                      35000 MEDICAL                   11.1
 9         22 female        NA                      100684              3 RENT                      35000 PERSONAL                   8.9
10         21 female        High School              12739              0 OWN                        1600 VENTURE                   14.7
11         22 female        High School             102985              0 RENT                      35000 VENTURE                   10.4
12         21 female        Associate                13113              0 OWN                        4500 HOMEIMPROVEMENT            8.63
13         23 male          Bachelor                114860              3 RENT                      35000 VENTURE                    7.9
14         NA male          Master                  130713              0 RENT                      35000 EDUCATION                 18.4
15         23 female        Associate              3138998              0 RENT                      35000 EDUCATION                  7.9
16         23 female        NA                          NA              5 MORTGAGE                  30000 DEBTCONSOLIDATION         10.6
17         23 NA            Bachelor                144943              0 RENT                      35000 EDUCATION                  7.9
18         23 female        High School             111369              0 RENT                      35000 MEDICAL                   20
19         23 male          Bachelor                136628              0 RENT                      35000 DEBTCONSOLIDATION         18.2
20         24 female        Master                   14283              1 MORTGAGE                   1750 EDUCATION                 11.0
21         25 male          Bachelor                195718              0 RENT                      35000 VENTURE                    7.49
22         25 male          High School             165792              4 RENT                      34800 PERSONAL                  16.8
23         22 female        Master                   79255              0 RENT                      34000 EDUCATION                 17.6
24         24 female        Bachelor                 13866              0 OWN                        1500 PERSONAL                   7.29
```

**Description**

The dplyr package is installed and loaded for data manipulation, while the readxl package is installed and loaded for reading Excel files. The dataset Midterm_Dataset_Section(C).xlsx is read from the specified file path into dataSet_1. Finally, the entire dataset is printed using print with all rows displayed.In the screenshot above, we can see the first 24 instances of the dataset. Though the output displayed all the instances of the dataset.

# Dataset Dimensions and Structure

**code**

```
no_of_col <- ncol(dataSet_1)

no_of_row <- nrow(dataSet_1)

cat("No of row in the dataset: ", no_of_row)

cat("No of column in the d Description of_col)

str(dataSet_1)
```

### Output

```
> no_of_col <- ncol(dataSet_1)
> no_of_row <- nrow(dataSet_1)
> cat("No of row in the dataset: ", no_of_row)
No of row in the dataset:  201
> cat("No of column in the dataset: ", no_of_col)
No of column in the dataset:  14
> str(dataSet_1)
tibble [201 × 14] (S3: tbl_df/tbl/data.frame)
 $ person_age                    : num [1:201] 21 21 25 23 24 NA 22 24 22 21 ...
 $ person_gender                 : chr [1:201] "female" "female" "female" "female" ...
 $ person_education              : chr [1:201] "Master" "High School" "High School" "Bachelor" ...
 $ person_income                 : num [1:201] 71948 12282 12438 79753 66135 ...
 $ person_emp_exp                : num [1:201] 0 0 3 0 1 0 1 5 3 0 ...
 $ person_home_ownership         : chr [1:201] "RENT" "OWN" "MORTGAGE" "RENT" ...
 $ loan_amnt                     : num [1:201] 35000 1000 5500 35000 35000 2500 35000 35000 35000 1600 ...
 $ loan_intent                   : chr [1:201] "PERSONAL" "EDUCATION" "MEDICAL" "MEDICAL" ...
 $ loan_int_rate                 : num [1:201] 16 11.1 12.9 15.2 14.3 ...
 $ loan_percent_income           : num [1:201] 0.49 NA 0 0.44 0.53 0.19 0.37 0.37 0.35 0.13 ...
 $ cb_person_cred_hist_length    : num [1:201] 3 2 3 2 4 2 3 4 2 3 ...
 $ credit_score                  : num [1:201] 561 504 635 675 586 532 701 585 544 640 ...
 $ previous_loan_defaults_on_file: chr [1:201] "No" "Yes" "No" "No" ...
 $ loan_status                   : num [1:201] 1 0 1 1 1 1 1 1 NA 1 ...
> 
```

### Description

This code calculates and prints the number of rows and columns in the dataset dataSet_1 using the nrow and ncol functions. Additionally, the str function provides a detailed overview of the dataset's structure, including the column names, data types, and sample values for each column. The output includes both the dataset's dimensions and a concise summary of its attributes.

## Extracting Unique Values from Dataset Columns

### Code

```
unique(dataSet_1$person_age)

unique(dataSet_1$person_gender)
```

### Output

```
> unique(dataSet_1$person_age)
 [1]  21  25  23  24  NA  22 230  26 350 144
> unique(dataSet_1$person_gender)
[1] "female" "male"   NA
> 
```

### Description

We are extracting unique values from specific columns in the dataset dataSet_1. By applying the unique function to the person_age column, we retrieve all distinct age values, while applying it to the person_gender column provides a list of unique gender categories. We can see NULL values in person_gender columns , we will deal with them later.

## Removing Duplicate Rows from the Dataset

### Code

```
remo_dupli_dataset <- distinct(dataSet_1);
```

```
remo_dupli_dataset

cat ("No of row and column after removing duplicate instances: ",
nrow(remo_dupli_dataset), ncol(remo_dupli_dataset))
```

**Output**

```
> remo_dupli_dataset <- distinct(dataSet_1);
> remo_dupli_dataset
# A tibble: 200 × 14
   person_age person_gender person_education person_income person_emp_exp person_home_ownership loan_amnt loan_intent loan_int_rate
        <dbl> <chr>         <chr>                    <dbl>          <dbl> <chr>                     <dbl> <chr>               <dbl>
 1         21 female        Master                   71948              0 RENT                      35000 PERSONAL             16.0
 2         21 female        High School              12282              0 OWN                        1000 EDUCATION            11.1
 3         25 female        High School              12438              3 MORTGAGE                   5500 MEDICAL              12.9
 4         23 female        Bachelor                 79753              0 RENT                      35000 MEDICAL              15.2
 5         24 male          Master                   66135              1 RENTT                     35000 MEDICAL              14.3
 6         NA female        High School              12951              0 OWN                        2500 VENTURE               7.14
 7         22 female        Bachelor                    NA              1 RENT                      35000 EDUCATION            12.4
 8         24 NA            High School              95550              5 RENT                      35000 MEDICAL              11.1
 9         22 female        NA                      100684              3 RENT                      35000 PERSONAL              8.9
10         21 female        High School              12739              0 OWN                        1600 VENTURE              14.7
# ℹ 190 more rows
# ℹ 5 more variables: loan_percent_income <dbl>, cb_person_cred_hist_length <dbl>, credit_score <dbl>, previous_loan_defaults_on_file <chr>,
#   loan_status <dbl>
# ℹ Use `print(n = ...)` to see more rows
> cat ("No of row and column after removing duplicate instances: ", nrow(remo_dupli_dataset), ncol(remo_dupli_dataset))
No of row and column after removing duplicate instances:  200 14
> |
```

**Description**

We are removing duplicate rows from the dataset dataSet_1 to ensure each entry is unique. The resulting dataset, stored in remo_dupli_dataset, is displayed, followed by the total number of rows and columns remaining after duplicate removal.One instance has been removed, the the dimension of the new dataset id 200 rows and 14 columns

# Handling Invalid Values

**Code**

```
fresh_dataset <- remo_dupli_dataset;

unique(fresh_dataset$person_gender)

unique(fresh_dataset$person_education)

unique(fresh_dataset$person_home_ownership)

unique(fresh_dataset$loan_intent)

unique(fresh_dataset$previous_loan_defaults_on_file);


fresh_dataset$person_age[is.na(as.numeric(as.character(fresh_dataset$person_age)))]

fresh_dataset$person_income[is.na(as.numeric(as.character(fresh_dataset$person_income)))]

fresh_dataset$person_emp_exp[is.na(as.numeric(as.character(fresh_dataset$person_emp_exp)))]

fresh_dataset$loan_amnt[is.na(as.numeric(as.character(fresh_dataset$loan_amnt)))]

fresh_dataset$loan_int_rate[is.na(as.numeric(as.character(fresh_dataset$loan_int_rate)))]

fresh_dataset$loan_percent_income[is.na(as.numeric(as.character(fresh_dataset$loan_percent_inc
ome)))]
```

```
fresh_dataset$cb_person_cred_hist_length[is.na(as.numeric(as.character(fresh_dataset$cb_person
_cred_hist_length)))]

fresh_dataset$credit_score[is.na(as.numeric(as.character(fresh_dataset$credit_score)))]

fresh_dataset$loan_status[is.na(as.numeric(as.character(fresh_dataset$loan_status)))]


deal_invalid_dataset <- fresh_dataset;

deal_invalid_dataset$person_home_ownership <- ifelse(

  substr(toupper(deal_invalid_dataset$person_home_ownership), 1, 2) == "OT", "OTHER",

  ifelse(

    substr(toupper(deal_invalid_dataset$person_home_ownership), 1, 1) == "O", "OWN",

    ifelse(

      substr(toupper(deal_invalid_dataset$person_home_ownership), 1, 1) == "R", "RENT",

      ifelse(

        substr(toupper(deal_invalid_dataset$person_home_ownership), 1, 1) == "M", "MORTGAGE",

        "NA"

      )

    )

  )

)
```

**Output**

```
> fresh_dataset <- remo_dupli_dataset;
> unique(fresh_dataset$person_gender)
[1] "female" "male"   NA
> unique(fresh_dataset$person_education)
[1] "Master"      "High School" "Bachelor"    NA           "Associate"   "Doctorate"
> unique(fresh_dataset$person_home_ownership)
[1] "RENT"     "OWN"      "MORTGAGE" "RENTT"    "OOWN"     "OTHER"
> unique(fresh_dataset$loan_intent)
[1] "PERSONAL"          "EDUCATION"         "MEDICAL"           "VENTURE"           "HOMEIMPROVEMENT"   "DEBTCONSOLIDATION"
> unique(fresh_dataset$previous_loan_defaults_on_file);
[1] "No"  "Yes"
> fresh_dataset$person_age[is.na(as.numeric(as.character(fresh_dataset$person_age)))]
[1] NA NA NA NA
> fresh_dataset$person_income[is.na(as.numeric(as.character(fresh_dataset$person_income)))]
[1] NA NA NA NA
> fresh_dataset$person_emp_exp[is.na(as.numeric(as.character(fresh_dataset$person_emp_exp)))]
numeric(0)
> fresh_dataset$loan_amnt[is.na(as.numeric(as.character(fresh_dataset$loan_amnt)))]
numeric(0)
> fresh_dataset$loan_int_rate[is.na(as.numeric(as.character(fresh_dataset$loan_int_rate)))]
numeric(0)
> fresh_dataset$loan_percent_income[is.na(as.numeric(as.character(fresh_dataset$loan_percent_income)))]
[1] NA
> fresh_dataset$cb_person_cred_hist_length[is.na(as.numeric(as.character(fresh_dataset$cb_person_cred_hist_length)))]
numeric(0)
> fresh_dataset$credit_score[is.na(as.numeric(as.character(fresh_dataset$credit_score)))]
numeric(0)
> fresh_dataset$loan_status[is.na(as.numeric(as.character(fresh_dataset$loan_status)))]
[1] NA NA NA
> deal_invalid_dataset <- fresh_dataset;
> deal_invalid_dataset$person_home_ownership <- ifelse(
+   substr(toupper(deal_invalid_dataset$person_home_ownership), 1, 2) == "OT", "OTHER",
+   ifelse(
+     substr(toupper(deal_invalid_dataset$person_home_ownership), 1, 1) == "O", "OWN",
+     ifelse(
+       substr(toupper(deal_invalid_dataset$person_home_ownership), 1, 1) == "R", "RENT",
+       ifelse(
+         substr(toupper(deal_invalid_dataset$person_home_ownership), 1, 1) == "M", "MORTGAGE",
+         "NA"
+       )
+     )
+   )
+ )
> unique(deal_invalid_dataset $person_home_ownership)
[1] "RENT"     "OWN"      "MORTGAGE" "OTHER"
> 
```

**Description**

We first checked the unique values of categorical columns in the dataset, such as person_gender, person_education, person_home_ownership, loan_intent, and previous_loan_defaults_on_file. Next, we verified whether numerical columns contained only valid numeric values, and identified any missing values (NAs) which would be addressed later. We then focused on cleaning the person_home_ownership column, where invalid values were present. We assumed that if the value starts with "OT," it should be categorized as "OTHER." Similarly, values starting with "O" were classified as "OWN," those starting with "R" as "RENT," those starting with "M" as "MORTGAGE," and any other value was assigned "NA." Finally, we confirmed the changes by examining the unique values in each column using the unique() function.

# Dealing with Missing Values

## Discard instances

**Code**

```
fresh_dataset <- deal_invalid_dataset;

deal_miss_value_dataset <- fresh_dataset;
```

```
colSums(is.na(deal_miss_value_dataset));

which(is.na(deal_miss_value_dataset$ person_age))

deal_miss_value_dataset <- na.omit(deal_miss_value_dataset);

colSums(is.na(deal_miss_value_dataset));
```

**Output**

```
> fresh_dataset <- deal_invalid_dataset;
> deal_miss_value_dataset <- fresh_dataset;
> colSums(is.na(deal_miss_value_dataset));
                    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                             1                             0                             0
previous_loan_defaults_on_file                   loan_status
                             0                             3
> which(is.na(deal_miss_value_dataset$ person_age))
[1]  6 14 28 34
> deal_miss_value_dataset <- na.omit(deal_miss_value_dataset);
> colSums(is.na(deal_miss_value_dataset));
                    person_age                 person_gender              person_education                 person_income
                             0                             0                             0                             0
                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                             0
> |
```

**Description**

We began by examining the missing values in the dataset by using colSums(is.na()) to get a summary of missing values across all columns. The number of missing values in each column is displayed. Then, we identified specific rows where the person_age column contained missing values with the which(is.na()) function. To address these missing values, we removed any rows containing NA values from the dataset using the na.omit() function. Finally, we reassessed the dataset to confirm that all missing values were successfully removed by applying colSums(is.na()) once more.We can see no column contains any missing values

## Top-Down and Bottom-Up Approach

```
top_down_dataset <- fresh_dataset %>% fill(person_age,person_gender, person_education, person_income,loan_percent_income, loan_status, .direction = 'down')

colSums(is.na(top_down_dataset));

bottom_up_dataset <- fresh_dataset %>% fill(person_age,person_gender, person_education, person_income,loan_percent_income, loan_status, .direction = 'up')

colSums(is.na(bottom_up_dataset));
```

**Output**

```
> top_down_dataset <- fresh_dataset %>% fill(person_age,person_gender, person_education, person_income,loan_percent_income, loan_status, .direction =
'down')
> colSums(is.na(top_down_dataset));
                person_age              person_gender           person_education              person_income
                         0                          0                          0                          0
            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                          0
> bottom_up_dataset <- fresh_dataset %>% fill(person_age,person_gender, person_education, person_income,loan_percent_income, loan_status, .direction =
'up')
> colSums(is.na(bottom_up_dataset));
                person_age              person_gender           person_education              person_income
                         0                          0                          0                          0
            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                          0
>
> |
```

**Description**

We applied two approaches to fill missing values in the dataset. In the Top-Down approach, we used the fill() function with the .direction = 'down' parameter to fill missing values by propagating the previous value downward across selected columns. We then checked for any remaining missing values using colSums(is.na()).

In the Bottom-Up approach, we again used the fill() function but with the .direction = 'up' parameter, which propagates missing values upward. We confirmed the absence of any remaining missing values by examining the result with colSums(is.na()).

## Replace by Most Frequent/Average Value

## For categorical columns (Mode)

**Code**

```
deal_miss_value_mode <- fresh_dataset;

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

deal_miss_value_mode$person_gender[is.na(deal_miss_value_mode$person_gender)] <-
mode_person_gender

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

deal_miss_value_mode$person_education[is.na(deal_miss_value_mode$person_education)] <-
mode_person_education

mode_person_home_ownership <- names(sort(table(deal_miss_value_mode$person_home_ownership),
decreasing = TRUE))[1]

deal_miss_value_mode$person_home_ownership[is.na(deal_miss_value_mode$person_home_ownership)]
<- mode_person_home_ownership
```

```
mode_loan_intent <- names(sort(table(deal_miss_value_mode$loan_intent), decreasing = TRUE))[1]

deal_miss_value_mode$loan_intent[is.na(deal_miss_value_mode$loan_intent)] <- mode_loan_intent

mode_previous_loan_defaults_on_file <-
names(sort(table(deal_miss_value_mode$previous_loan_defaults_on_file), decreasing = TRUE))[1]

deal_miss_value_mode$previous_loan_defaults_on_file[is.na(deal_miss_value_mode$previous_loan_d
efaults_on_file)] <- mode_previous_loan_defaults_on_file

colSums(is.na(deal_miss_value_mode))
```

**Output**

```
> deal_miss_value_mode <- fresh_dataset;
> mode_person_gender <- names(sort(table(deal_miss_value_mode$person_gender), decreasing = TRUE))[1]
> deal_miss_value_mode$person_gender[is.na(deal_miss_value_mode$person_gender)] <- mode_person_gender
> mode_person_education <- names(sort(table(deal_miss_value_mode$person_education), decreasing = TRUE))[1]
> deal_miss_value_mode$person_education[is.na(deal_miss_value_mode$person_education)] <- mode_person_education
> mode_person_home_ownership <- names(sort(table(deal_miss_value_mode$person_home_ownership), decreasing = TRUE))[1]
> deal_miss_value_mode$person_home_ownership[is.na(deal_miss_value_mode$person_home_ownership)] <- mode_person_home_ownership
> mode_loan_intent <- names(sort(table(deal_miss_value_mode$loan_intent), decreasing = TRUE))[1]
> deal_miss_value_mode$loan_intent[is.na(deal_miss_value_mode$loan_intent)] <- mode_loan_intent
> mode_previous_loan_defaults_on_file <- names(sort(table(deal_miss_value_mode$previous_loan_defaults_on_file), decreasing = TRUE))[1]
> deal_miss_value_mode$previous_loan_defaults_on_file[is.na(deal_miss_value_mode$previous_loan_defaults_on_file)] <- mode_previous_loan_defaults_on_fi
e
> colSums(is.na(deal_miss_value_mode))
                    person_age                  person_gender               person_education                  person_income
                             4                              0                              0                              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                              1                              0                              0
previous_loan_defaults_on_file                    loan_status
                             0                              3
> |
```

**Description**

We handled missing values in categorical columns by replacing them with the most frequent value (mode). For each categorical column—person_gender, person_education, person_home_ownership, loan_intent, and previous_loan_defaults_on_file—we first identified the mode using the sort() and table() functions. Then, we replaced any missing values with these most common values. Finally, we checked if any missing values remained in the dataset by summarizing with colSums(is.na()).

## For numerical columns (mean)

**Code**

```
deal_miss_value_mean <- deal_miss_value_mode;

for(col_name in c("person_age", "person_income", "loan_percent_income", "loan_status")) {

  if(is.numeric(deal_miss_value_mean[[col_name]])) {


    column_mean <- mean(deal_miss_value_mean[[col_name]], na.rm = TRUE)

    deal_miss_value_mean[[col_name]][is.na(deal_miss_value_mean[[col_name]])] <- column_mean

    deal_miss_value_mean[[col_name]] <- round(deal_miss_value_mean[[col_name]], digits = 0)
```

```
  }
}
colSums(is.na(deal_miss_value_mean))
```

**Output**

```
> deal_miss_value_mean <- deal_miss_value_mode;
> for(col_name in c("person_age", "person_income", "loan_percent_income", "loan_status")) {
+   if(is.numeric(deal_miss_value_mean[[col_name]])) {
+
+     column_mean <- mean(deal_miss_value_mean[[col_name]], na.rm = TRUE)
+     deal_miss_value_mean[[col_name]][is.na(deal_miss_value_mean[[col_name]])] <- column_mean
+     deal_miss_value_mean[[col_name]] <- round(deal_miss_value_mean[[col_name]], digits = 0)
+   }
+ }
> colSums(is.na(deal_miss_value_mean))
                    person_age                  person_gender               person_education                  person_income
                             0                              0                              0                              0
                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                              0
> |
```

**Description**

We replaced missing values in numerical columns—person_age, person_income, loan_percent_income, and loan_status—by using the mean value of each respective column. For each column, we calculated the mean while excluding missing values, rounded the result to the nearest integer, and substituted any missing entries with this mean. Finally, we checked if any missing values remained in the dataset using colSums(is.na()).

# Converting Categorical Columns to Numeric Factors

**Code**

```
fresh_dataset <- deal_miss_value_dataset;

dataSet_num <- fresh_dataset;

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

dataSet_num$person_education <-  factor(dataSet_num$person_education, levels = c("High School", "Bachelor", "Master", "Associate", "Doctorate"), labels = c(1,2,3,4,5));

dataSet_num$loan_intent <-  factor(dataSet_num$loan_intent, levels = c("PERSONAL","EDUCATION","MEDICAL","VENTURE","HOMEIMPROVEMENT", "DEBTCONSOLIDATION"), labels = c(1,2,3,4,5, 6));

dataSet_num$person_home_ownership <-  factor(dataSet_num$person_home_ownership, levels = c("RENT","OWN","MORTGAGE","OTHER"), labels = c(1,2,3,4));

dataSet_num$previous_loan_defaults_on_file <- factor(dataSet_num$previous_loan_defaults_on_file, levels = c("Yes", "No"), labels = c(1,2));

dataSet_num
```

### Ouput

```
> fresh_dataset <- deal_miss_value_dataset;
> dataSet_num <- fresh_dataset;
> dataSet_num$person_gender <- factor(dataSet_num$person_gender, levels = c("male", "female"), labels = c(1,2));
> dataSet_num$person_education <-  factor(dataSet_num$person_education, levels = c("High School", "Bachelor", "Master", "Associate", "Doctorate"), labe
ls = c(1,2,3,4,5));
> dataSet_num$loan_intent <-  factor(dataSet_num$loan_intent, levels = c("PERSONAL","EDUCATION","MEDICAL","VENTURE","HOMEIMPROVEMENT", "DEBTCONSOLIDATI
ON"), labels = c(1,2,3,4,5, 6));
> dataSet_num$person_home_ownership <-  factor(dataSet_num$person_home_ownership, levels = c("RENT","OWN","MORTGAGE","OTHER"), labels = c(1,2,3,4));
> dataSet_num$previous_loan_defaults_on_file <-  factor(dataSet_num$previous_loan_defaults_on_file, levels = c("Yes", "No"), labels = c(1,2));
> dataSet_num
# A tibble: 184 × 14
   person_age person_gender person_education person_income person_emp_exp person_home_ownership loan_amnt loan_intent loan_int_rate
        <dbl> <fct>         <fct>                    <dbl>          <dbl> <fct>                     <dbl> <fct>               <dbl>
 1         21 2             3                        71948              0 1                         35000 1                   16.0 
 2         25 2             1                        12438              3 3                          5500 3                   12.9 
 3         23 2             2                        79753              0 1                         35000 3                   15.2 
 4         24 1             3                        66135              1 1                         35000 3                   14.3 
 5         21 2             1                        12739              0 2                          1600 4                   14.7 
 6         22 2             1                       102985              0 1                         35000 4                   10.4 
 7         21 2             4                        13113              0 2                          4500 5                    8.63
 8         23 1             2                       114860              3 1                         35000 4                    7.9 
 9         23 1             2                       136628              0 1                         35000 6                   18.2 
10         24 2             3                        14283              1 3                          1750 2                   11.0 
# ℹ 174 more rows
# ℹ 5 more variables: loan_percent_income <dbl>, cb_person_cred_hist_length <dbl>, credit_score <dbl>, previous_loan_defaults_on_file <fct>,
#   loan_status <dbl>
# ℹ Use `print(n = ...)` to see more rows
> |
```

### Description

We deal the missing values with 3 methods.But we will use the dataset we got after discarding missing values for further analysis. We converted all categorical columns into numeric factors to prepare the dataset for analysis. The conversion included the following mappings: person_gender was encoded as 1 (male) and 2 (female), person_education was mapped across five levels (High School, Associate, Bachelor, Master, Doctorate), loan_intent was mapped to six loan purposes (Personal, Education, Medical, Venture, Home Improvement, Debt Consolidation), person_home_ownership was categorized into four types (Rent, Own, Mortgage, Other), and previous_loan_defaults_on_file was encoded as 1 (Yes) and 2 (No).

## Identifying Outliers

### Code

```
detect_outlier <- function(dataframe, columns) {

  for (col in columns) {

    if (is.numeric(dataframe[[col]])) {

      Quantile1 <- quantile(dataframe[[col]], probs = 0.25)

      Quantile3 <- quantile(dataframe[[col]], probs = 0.75)

      IQR <- Quantile3 - Quantile1

      outlier_flags <- dataframe[[col]] > Quantile3 + (IQR * 1.5) | dataframe[[col]] <
Quantile1 - (IQR * 1.5)

      outliers <- dataframe[[col]][outlier_flags]

      if (length(outliers) > 0) {

        cat("Outliers detected in column", col, ":\n")
```

```
        print(outliers)
      } else {
        cat("No outliers detected in column", col, "\n")
      }
    } else {
      cat("Column", col, "is not numeric, skipped\n")
    }
  }
}
detect_outlier(fresh_dataset, names(fresh_dataset))
```

**Output**

```
> detect_outlier(fresh_dataset, names(fresh_dataset))
Outliers detected in column person_age :
[1] 230 350 144 144
Column person_gender is not numeric, skipped
Column person_education is not numeric, skipped
No outliers detected in column person_income
Outliers detected in column person_emp_exp :
[1] 125   8 121
Column person_home_ownership is not numeric, skipped
No outliers detected in column loan_amnt
Column loan_intent is not numeric, skipped
Outliers detected in column loan_int_rate :
[1]  5.42 19.91
No outliers detected in column loan_percent_income
No outliers detected in column cb_person_cred_hist_length
Outliers detected in column credit_score :
[1] 789 484 807
Column previous_loan_defaults_on_file is not numeric, skipped
No outliers detected in column loan_status
>
```

**Description**

We applied a user defined detect_outlier function to identify outlier values in each numeric column of the dataset. This function uses the Interquartile Range (IQR) approach to detect extreme values, ensuring that any anomalies that could affect data analysis are identified. The method outputs details about outliers for each column, helping to pinpoint potential issues that may need to be addressed separately. If outliers present in a column, then it displayes, if not then it displays no ouliers, if the column is categorical(not numeric) the, it skips the columns

# Removing Outliers

**Code**

```
remove_outlier <- function(dataframe, columns) {
  for (col in columns) {
    if (is.numeric(dataframe[[col]])) {
      Quantile1 <- quantile(dataframe[[col]], probs = 0.25)
```

```
Quantile3 <- quantile(dataframe[[col]], probs = 0.75)
      IQR <- Quantile3 - Quantile1
      dataframe <- dataframe[!(
        dataframe[[col]] > Quantile3 + (IQR * 1.5) |
          dataframe[[col]] < Quantile1 - (IQR * 1.5)
      ), ]
    }
  }
  return(dataframe)
}
without_outlier_data <- remove_outlier(fresh_dataset, names(fresh_dataset))
without_outlier_data
detect_outlier(without_outlier_data, names(without_outlier_data))
fresh_dataset <- without_outlier_data;
```

# Output

```
> remove_outlier <- function(dataframe, columns) {
+   for (col in columns) {
+     if (is.numeric(dataframe[[col]])) {
+       Quantile1 <- quantile(dataframe[[col]], probs = 0.25)
+       Quantile3 <- quantile(dataframe[[col]], probs = 0.75)
+       IQR <- Quantile3 - Quantile1
+
+       dataframe <- dataframe[!(
+         dataframe[[col]] > Quantile3 + (IQR * 1.5) |
+           dataframe[[col]] < Quantile1 - (IQR * 1.5)
+       ), ]
+     }
+   }
+   return(dataframe)
+ }
> without_outlier_data <- remove_outlier(fresh_dataset, names(fresh_dataset))
> without_outlier_data
# A tibble: 176 x 14
   person_age person_gender person_education person_income person_emp_exp person_home_ownership loan_amnt loan_intent loan_int_rate
        <dbl> <fct>         <fct>                    <dbl>          <dbl> <fct>                     <dbl> <fct>               <dbl>
 1         21 2             3                        71948              0 1                         35000 1                   16.0
 2         25 2             1                        12438              3 3                          5500 3                   12.9
 3         23 2             2                        79753              0 1                         35000 3                   15.2
 4         24 1             3                        66135              1 1                         35000 3                   14.3
 5         21 2             1                        12739              0 2                          1600 4                   14.7
 6         22 2             1                       102985              0 1                         35000 4                   10.4
 7         21 2             4                        13113              0 2                          4500 5                    8.63
 8         23 1             2                       114860              3 1                         35000 4                    7.9
 9         23 1             2                       136628              0 1                         35000 6                   18.2
10         24 2             3                        14283              1 3                          1750 2                   11.0
# i 166 more rows
# i 5 more variables: loan_percent_income <dbl>, cb_person_cred_hist_length <dbl>, credit_score <dbl>, previous_loan_defaults_on_file <fct>,
#   loan_status <dbl>
# i Use `print(n = ...)` to see more rows
> detect_outlier(without_outlier_data, names(without_outlier_data))
No outliers detected in column person_age
Column person_gender is not numeric, skipped
Column person_education is not numeric, skipped
No outliers detected in column person_income
No outliers detected in column person_emp_exp
Column person_home_ownership is not numeric, skipped
No outliers detected in column loan_amnt
Column loan_intent is not numeric, skipped
No outliers detected in column loan_int_rate
No outliers detected in column loan_percent_income
No outliers detected in column cb_person_cred_hist_length
No outliers detected in column credit_score
Column previous_loan_defaults_on_file is not numeric, skipped
No outliers detected in column loan_status
> 
```

### Description

We used the **remove_outlier** function to eliminate outlier values from all numeric columns in the dataset. This method applies the Interquartile Range (IQR) approach to filter out extreme values, ensuring that our dataset is free from anomalies that could skew analysis results. After removing the outliers, we re-applied the **detect_outlier** function to confirm that the dataset no longer contains any extreme values.We can see from the output that, after removing outliers, that new dataset contains 176 rows and 14 columns

## Normalizing the Dataset

### Code

```
normalize_dataset <- fresh_dataset;

min_age <- min(normalize_dataset$person_age, na.rm = TRUE)

max_age <- max(normalize_dataset$person_age, na.rm = TRUE)

normalize_dataset$person_age <- (normalize_dataset$person_age - min_age) / (max_age - min_age)


min_income <- min(normalize_dataset$person_income, na.rm = TRUE)

max_income <- max(normalize_dataset$person_income, na.rm = TRUE)

normalize_dataset$person_income <- (normalize_dataset$person_income - min_income) /
(max_income - min_income)


min_loan_amnt <- min(normalize_dataset$loan_amnt, na.rm = TRUE)

max_loan_amnt <- max(normalize_dataset$loan_amnt, na.rm = TRUE)

normalize_dataset$loan_amnt <- (normalize_dataset$loan_amnt - min_loan_amnt) / (max_loan_amnt
- min_loan_amnt);


min_loan_int_rate <- min(normalize_dataset$loan_int_rate, na.rm = TRUE)

max_loan_int_rate <- max(normalize_dataset$loan_int_rate, na.rm = TRUE)

normalize_dataset$loan_int_rate <- (normalize_dataset$loan_int_rate - min_loan_int_rate) /
(max_loan_int_rate - min_loan_int_rate);


min_credit_score <- min(normalize_dataset$credit_score, na.rm = TRUE)

max_credit_score <- max(normalize_dataset$credit_score, na.rm = TRUE)

normalize_dataset$credit_score <- (normalize_dataset$credit_score - min_credit_score) /
(max_credit_score - min_credit_score );

normalize_dataset

fresh_dataset <- normalize_dataset;
```

**Output**

```
> fresh_dataset <- without_outlier_data;
> normalize_dataset <- fresh_dataset;
> min_age <- min(normalize_dataset$person_age, na.rm = TRUE)
> max_age <- max(normalize_dataset$person_age, na.rm = TRUE)
> normalize_dataset$person_age <- (normalize_dataset$person_age - min_age) / (max_age - min_age)
> min_income <- min(normalize_dataset$person_income, na.rm = TRUE)
> max_income <- max(normalize_dataset$person_income, na.rm = TRUE)
> normalize_dataset$person_income <- (normalize_dataset$person_income - min_income) / (max_income - min_income)
> min_loan_amnt <- min(normalize_dataset$loan_amnt, na.rm = TRUE)
> max_loan_amnt <- max(normalize_dataset$loan_amnt, na.rm = TRUE)
> normalize_dataset$loan_amnt <- (normalize_dataset$loan_amnt - min_loan_amnt) / (max_loan_amnt - min_loan_amnt);
> min_loan_int_rate <- min(normalize_dataset$loan_int_rate, na.rm = TRUE)
> max_loan_int_rate <- max(normalize_dataset$loan_int_rate, na.rm = TRUE)
> normalize_dataset$loan_int_rate <- (normalize_dataset$loan_int_rate - min_loan_int_rate) / (max_loan_int_rate - min_loan_int_rate);
> min_credit_score <- min(normalize_dataset$credit_score, na.rm = TRUE)
> max_credit_score <- max(normalize_dataset$credit_score, na.rm = TRUE)
> normalize_dataset$credit_score <- (normalize_dataset$credit_score - min_credit_score) / (max_credit_score - min_credit_score );
> normalize_dataset
# A tibble: 176 × 14
   person_age person_gender person_education person_income person_emp_exp person_home_ownership loan_amnt loan_intent loan_int_rate
        <dbl> <fct>         <fct>                    <dbl>          <dbl> <fct>                     <dbl> <fct>               <dbl>
 1        0   2             3                     0.170               0 1                       1       1                   0.747
 2        0.8 2             1                     0                   3 3                       0.122   3                   0.513
 3        0.4 2             2                     0.193               0 1                       1       3                   0.689
 4        0.6 1             3                     0.154               1 1                       1       3                   0.617
 5        0   2             1                     0.000862            0 2                       0.00595 4                   0.652
 6        0.2 2             1                     0.259               0 1                       1       4                   0.326
 7        0   2             4                     0.00193             0 2                       0.0923  5                   0.197
 8        0.4 1             2                     0.293               3 1                       1       4                   0.142
 9        0.4 1             2                     0.356               0 1                       1       6                   0.914
10        0.6 2             3                     0.00528             1 3                       0.0104  2                   0.373
# ℹ 166 more rows
# ℹ 5 more variables: loan_percent_income <dbl>, cb_person_cred_hist_length <dbl>, credit_score <dbl>, previous_loan_defaults_on_file <fct>,
#   loan_status <dbl>
# ℹ Use `print(n = ...)` to see more rows
> |
```

# Description

We applied **min-max normalization** to scale selected numeric columns—person_age, person_income, loan_amnt, and loan_int_rate—to a range between 0 and 1. These specific columns were chosen because they had significant variations in their data points. For instance, person_income had extremely high values, while loan_amnt and loan_int_rate also showed large ranges. Such disparities could skew analytical processes. Normalizing these columns helps in reducing this imbalance, ensuring all features contribute equally during model training and analysis. Other numerical columns were not prioritized as their value ranges were relatively consistent and did not pose the same scaling issues.

# Descriptive Statistics

## Displaying summary of the dataset

**Code**

```
summary(fresh_dataset);
```

**Output**

```
> fresh_dataset <- normalize_dataset;
> summary(fresh_dataset);
   person_age     person_gender person_education person_income    person_emp_exp  person_home_ownership   loan_amnt       loan_intent
 Min.   :0.0000   1:108         1:49             Min.   :0.0000   Min.   :0.000   1:169                 Min.   :0.0000   1:27
 1st Qu.:0.2000   2: 68         2:63             1st Qu.:0.1382   1st Qu.:0.000   2:  3                 1st Qu.:0.3147   2:49
 Median :0.4000                 3:21             Median :0.2085   Median :1.000   3:  3                 Median :0.7024   3:22
 Mean   :0.4966                 4:42             Mean   :0.3532   Mean   :1.523   4:  1                 Mean   :0.5648   4:29
 3rd Qu.:0.8000                 5: 1             3rd Qu.:0.6549   3rd Qu.:3.000                         3rd Qu.:0.7775   5:21
 Max.   :1.0000                                  Max.   :1.0000   Max.   :7.000                         Max.   :1.0000   6:28
 loan_int_rate    loan_percent_income cb_person_cred_hist_length  credit_score    previous_loan_defaults_on_file  loan_status
 Min.   :0.0000   Min.   :0.0000      Min.   :2.000              Min.   :0.0000   1: 46                          Min.   :0.0000
 1st Qu.:0.3726   1st Qu.:0.0900      1st Qu.:2.000              1st Qu.:0.4366   2:130                          1st Qu.:0.0000
 Median :0.4423   Median :0.2300      Median :3.000              Median :0.5986                                  Median :1.0000
 Mean   :0.4718   Mean   :0.2293      Mean   :3.006              Mean   :0.5834                                  Mean   :0.6193
 3rd Qu.:0.6304   3rd Qu.:0.3425      3rd Qu.:4.000              3rd Qu.:0.7477                                  3rd Qu.:1.0000
 Max.   :1.0000   Max.   :0.5300      Max.   :4.000              Max.   :1.0000                                  Max.   :1.0000
> |
```

**Description**

By using the summary() function, we obtained a statistical overview of each column in the dataset, including measures like minimum, 1st quartile, median, mean, 3rd quartile, and maximum values. This helps in understanding the distribution, spread, and potential outliers across all features. It provides insights into each attribute's central tendency and variability, ensuring data integrity and highlighting areas that may need further cleaning, transformation, or normalization.

# Measure of Central Tendancy

**Code**

```
calculate_stats <- function(dataset, columns) {
  for (column_name in columns) {
    column_data <- dataset[[column_name]]

    # If column is numeric, calculate mean and median
    if (is.numeric(column_data)) {
      column_mean <- mean(column_data, na.rm = TRUE)
      column_median <- median(column_data, na.rm = TRUE)

      cat("Mean of column", column_name, "is", column_mean, "\n")
      cat("Median of column", column_name, "is", column_median, "\n")
      cat("\n")
    } else {
      # If column is categorical, calculate the mode
      column_mode <- names(sort(table(column_data), decreasing = TRUE))[1]
      cat("Mode of column", column_name, "is", column_mode, "\n")
      cat("\n")
    }
  }
}
calculate_stats(fresh_dataset,names(fresh_dataset))
```

### Output

```
> calculate_stats(fresh_dataset,names(fresh_dataset))
Mean of column person_age is 0.4965909
Median of column person_age is 0.4

Mode of column person_gender is 1

Mode of column person_education is 2

Mean of column person_income is 0.3532345
Median of column person_income is 0.2084707

Mean of column person_emp_exp is 1.522727
Median of column person_emp_exp is 1

Mode of column person_home_ownership is 1

Mean of column loan_amnt is 0.5648251
Median of column loan_amnt is 0.702381

Mode of column loan_intent is 2

Mean of column loan_int_rate is 0.4718492
Median of column loan_int_rate is 0.4422504

Mean of column loan_percent_income is 0.2292614
Median of column loan_percent_income is 0.23

Mean of column cb_person_cred_hist_length is 3.005682
Median of column cb_person_cred_hist_length is 3

Mean of column credit_score is 0.5833867
Median of column credit_score is 0.5985915

Mode of column previous_loan_defaults_on_file is 2

Mean of column loan_status is 0.6193182
Median of column loan_status is 1
```

### Description

We calculated mean and median for numeric columns (e.g., person_age, loan_amnt) to understand their central tendencies, while mode was calculated for categorical columns (e.g., loan_intent, person_home_ownership) to identify the most common categories. These statistics help in better understanding data distributions and ensuring that features contribute meaningfully to analysis and modeling.

## Measure of spread

### Code

```
columns_to_analyze <- c(

  "person_age",

  "person_income", "person_emp_exp",

  "loan_amnt", "loan_int_rate",

  "loan_percent_income","cb_person_cred_hist_length",

  "credit_score"

)

calculate_spread <- function(dataset, columns) {

  for (col_name in columns) {

    if (is.numeric(dataset[[col_name]])) {

      column_data <- dataset[[col_name]]
```

```
      column_range <- range(column_data, na.rm = TRUE)

      column_iqr <- IQR(column_data, na.rm = TRUE)

      column_sd <- sd(column_data, na.rm = TRUE)

      column_variance <- var(column_data, na.rm = TRUE)

      cat("For column", col_name, ":\n")

      cat("  Range:", column_range[2]- column_range[1], "\n")

      cat("  IQR:", column_iqr, "\n")

      cat("  Standard Deviation:", column_sd, "\n")

      cat("  Variance:", column_variance, "\n")

      cat("\n")


    }

  }

}
```

```
calculate_spread(fresh_dataset, columns_to_analyze)
```

**Output**

```
> calculate_spread(fresh_dataset, columns_to_analyze)
For column person_age :
  Range: 1
  IQR: 0.6
  Standard Deviation: 0.3221176
  Variance: 0.1037597

For column person_income :
  Range: 1
  IQR: 0.5167097
  Standard Deviation: 0.3061338
  Variance: 0.09371791

For column person_emp_exp :
  Range: 7
  IQR: 3
  Standard Deviation: 1.700267
  Variance: 2.890909

For column loan_amnt :
  Range: 1
  IQR: 0.4627976
  Standard Deviation: 0.306725
  Variance: 0.0940802

For column loan_int_rate :
  Range: 1
  IQR: 0.2578241
  Standard Deviation: 0.2209408
  Variance: 0.04881482

For column loan_percent_income :
  Range: 0.53
  IQR: 0.2525
  Standard Deviation: 0.1427646
  Variance: 0.02038174

For column cb_person_cred_hist_length :
  Range: 2
  IQR: 2
  Standard Deviation: 0.7745757
  Variance: 0.5999675

For column credit_score :
  Range: 1
  IQR: 0.3110329
  Standard Deviation: 0.2130489
  Variance: 0.04538983

>
```

**Description**

We analyzed the spread for selected numeric columns (e.g., person_age, loan_amnt) by calculating key metrics like range, interquartile range (IQR), standard deviation, and variance. These metrics help in understanding data dispersion, identifying potential outliers, and ensuring better model performance. Categorical columns were not analyzed in this process, as spread metrics like range or standard deviation are more meaningful for numeric data and do not apply to categorical variables.

# Handling imbalance dataset

## Oversampling

### Code

```
class_distribution <- table(fresh_dataset$loan_status)

print(class_distribution)

if (class_distribution[1] > class_distribution[2]) {

  majority <- filter(fresh_dataset, loan_status == 0)

  minority <- filter(fresh_dataset, loan_status == 1)

} else {

  majority <- filter(fresh_dataset, loan_status == 1)

  minority <- filter(fresh_dataset, loan_status == 0)

}

set.seed(123)

oversampled_minority <- minority %>% sample_n(nrow(majority), replace = TRUE)

oversampled_data <- bind_rows(majority, oversampled_minority)

table(oversampled_data$loan_status)

oversampled_data
```

### Output

```
> class_distribution <- table(fresh_dataset$loan_status)
> print(class_distribution)

 0   1 
67 109 
> if (class_distribution[1] > class_distribution[2]) {
+   majority <- filter(fresh_dataset, loan_status == 0)
+   minority <- filter(fresh_dataset, loan_status == 1)
+ } else {
+   majority <- filter(fresh_dataset, loan_status == 1)
+   minority <- filter(fresh_dataset, loan_status == 0)
+ }
> set.seed(123)
> oversampled_minority <- minority %>% sample_n(nrow(majority), replace = TRUE)
> oversampled_data <- bind_rows(majority, oversampled_minority)
> table(oversampled_data$loan_status)

  0   1 
109 109 
> oversampled_data
# A tibble: 218 × 14
   person_age person_gender person_education person_income person_emp_exp person_home_ownership loan_amnt loan_intent loan_int_rate
        <dbl> <fct>         <fct>                    <dbl>          <dbl> <fct>                     <dbl> <fct>               <dbl>
 1        0   2             3                     0.170                 0 1                       1       1                   0.747
 2        0.8 2             1                     0                     3 3                       0.122   3                   0.513
 3        0.4 2             2                     0.193                 0 1                       1       3                   0.689
 4        0.6 1             3                     0.154                 1 1                       1       3                   0.617
 5        0   2             1                     0.000862              0 2                       0.00595 4                   0.652
 6        0.2 2             1                     0.259                 0 1                       1       4                   0.326
 7        0   2             4                     0.00193               0 2                       0.0923  5                   0.197
 8        0.4 1             2                     0.293                 3 1                       1       4                   0.142
 9        0.4 1             2                     0.356                 0 1                       1       6                   0.914
10        0.6 2             3                     0.00528               1 3                       0.0104  2                   0.373
# ℹ 208 more rows
# ℹ 5 more variables: loan_percent_income <dbl>, cb_person_cred_hist_length <dbl>, credit_score <dbl>, previous_loan_defaults_on_file <fct>,
#   loan_status <dbl>
# ℹ Use `print(n = ...)` to see more rows
> |
```

### Description

To address the class imbalance in the loan_status column, the majority and minority classes were first identified based on their counts. Depending on which class had more samples, it was assigned as the majority, while the other was designated as the minority. The **oversampling** technique was then applied to balance these classes by duplicating the minority class data to match the size of the majority class using the sample_n() function with replacement. The bind_rows() function was subsequently used to merge the datasets back into a single, balanced dataset.

## Undersampling

### Code

```
undersampled_majority <- majority %>% sample_n(nrow(minority), replace = FALSE)

undersampled_data <- bind_rows(undersampled_majority, minority)

table(undersampled_data$loan_status)

undersampled_data

fresh_dataset <- oversampled_data
```

### Output

```
> undersampled_majority <- majority %>% sample_n(nrow(minority), replace = FALSE)
> undersampled_data <- bind_rows(undersampled_majority, minority)
> table(undersampled_data$loan_status)

 0  1 
67 67 
> undersampled_data
# A tibble: 134 × 14
   person_age person_gender person_education person_income person_emp_exp person_home_ownership loan_amnt loan_intent loan_int_rate
        <dbl> <fct>         <fct>                    <dbl>          <dbl> <fct>                     <dbl> <fct>               <dbl>
 1        0.4 1             2                      0.293                3 1                        1      4                   0.142
 2        0   2             2                      0.00847              0 1                        0.0551 6                   0.722
 3        0.4 1             2                      0.155                0 1                        0.702  2                   0.400
 4        0.8 2             4                      0.113                3 1                        0.702  2                   0.772
 5        0.4 1             1                      0.146                0 1                        0.702  1                   0.539
 6        0.8 1             2                      0.209                3 1                        0.702  2                   0.687
 7        0.8 1             2                      0.139                0 1                        0.702  5                   0    
 8        0.4 1             1                      0.139                0 1                        0.702  4                   0.435
 9        0.2 1             3                      0.339                0 1                        0.792  2                   0.374
10        0.8 2             4                      0.731                0 1                        0.702  3                   0.724
# ℹ 124 more rows
# ℹ 5 more variables: loan_percent_income <dbl>, cb_person_cred_hist_length <dbl>, credit_score <dbl>, previous_loan_defaults_on_file <fct>,
#   loan_status <dbl>
# ℹ Use `print(n = ...)` to see more rows
> 
```

### Description

To further address class imbalance in the loan_status column, **undersampling** was applied. In this approach, the majority class was reduced by randomly selecting a sample equal to the size of the minority class, using the sample_n() function without replacement. The minority and the newly undersampled majority datasets were then combined using bind_rows(). This resulted in a balanced dataset where both classes were equally represented, ensuring a more stable foundation for training and analysis. The balanced dataset was stored back into fresh_dataset.