

**AMERICAN INTERNATIONAL UNIVERSITY – BANGLADESH**

**INTRODUCTION TO DATA SCIENCE**

**FALL- 2024-2025**

Supervised By

**Tohedul Islam**

**Assignment: Mid Term Project Report**

**Group No: 02**

**Section: C**

Submitted By:

|  |  |
| --- | --- |
| **NAME** | **ID** |
| FARJANA YESMIN OPI | 22-47018-1 |
| MD. ABU TOWSIF | 22-47019-1 |
| MOST. SAYMA KHATUN | 22-47035-1 |

**TABLE OF CONTENTS**

**Topics Page no.**

**1. Description of the dataset 03**

**2. Loading the dataset 03-04**

**3. Dataset dimensions and structures 04-05**

**5. Extracting unique values from dataset columns 05**

**6. Removing duplicate rows from the dataset 06**

**7. Handling invalid values 06-08**

**8. Dealing with Missing Values 09-13**

**9. Converting Categorical Columns to Numeric Factors 13-14**

**10. Identifying Outliers 14-15**

**11. Removing Outliers 15-16**

**12. Normalizing the Dataset 17-18**

**13.Displaying Summary of the 18-19**

**14. Measure of Central Tendency 19-20**

**15. Measure of Spread 20-22**

**16. Handling Imbalance dataset 23-24**

## 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);

endenc\_1 <- read\_excel(“E:/FALL 24-25/INTRODUCTION TO DATA SCIENCE/MID/Mid Term Project/Metarials/Midterm\_Dataset\_Section©.xlsx”);

print(endenc\_1, n = nrow(endenc\_1));

#### **A screenshot of a computer Description automatically generated Output**

#### **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©.xlsx is read from the specified file path into endenc\_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(endenc\_1)

no\_of\_row <- nrow(endenc\_1)

cat(“No of row in the dataset: “, no\_of\_row)

cat(“No of column in the dataset: “, no\_of\_col)

str(endenc\_1)

**A close up of a computer screen

Description automatically generatedOutput**

#### **Description**

This code calculates and prints the number of rows and columns in the dataset endenc\_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(endenc\_1$person\_age)

unique(endenc\_1$person\_gender)

#### **Output**

#### **Description**

We are extracting unique values from specific columns in the dataset endenc\_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(endenc\_1);

remo\_dupli\_dataset

cat (“No of row and column after removing duplicate instances: “, nrow(remo\_dupli\_dataset), ncol(remo\_dupli\_dataset))

#### 

#### **Output**A close-up of a computer code Description automatically generated

#### **Description**

We are removing duplicate rows from the dataset endenc\_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 is 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\_income)))]

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”

)

)

)

)

A screenshot of a computer

Description automatically generated**Output**

#### **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));

**A screenshot of a computer code

Description automatically generatedOutput**

#### **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**

#### **Code**

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));

A white background with black text

Description automatically generated**Output**

#### **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\_defaults\_on\_file)] <- mode\_previous\_loan\_defaults\_on\_file

colSums(is.na(deal\_miss\_value\_mode))

A close-up of a computer screen

Description automatically generated**Output**

**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))

#### **A white screen with text Description automatically generated with medium confidenceOutput**

#### **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;

endenc\_num <- fresh\_dataset;

endenc\_num$person\_gender <- factor(endenc\_num$person\_gender, levels = c(“male”, “female”), labels = c(1,2));

endenc\_num$person\_education <- factor(endenc\_num$person\_education, levels = c(“High School”, “Bachelor”, “Master”, “Associate”, “Doctorate”), labels = c(1,2,3,4,5));

endenc\_num$loan\_intent <- factor(endenc\_num$loan\_intent, levels = c(“PERSONAL”,”EDUCATION”,”MEDICAL”,”VENTURE”,”HOMEIMPROVEMENT”, “DEBTCONSOLIDATION”), labels = c(1,2,3,4,5, 6));

endenc\_num$person\_home\_ownership <- factor(endenc\_num$person\_home\_ownership, levels = c(“RENT”,”OWN”,”MORTGAGE”,”OTHER”), labels = c(1,2,3,4));

endenc\_num$previous\_loan\_defaults\_on\_file <- factor(endenc\_num$previous\_loan\_defaults\_on\_file, levels = c(“Yes”, “No”), labels = c(1,2));

endenc\_num

A screenshot of a computer

Description automatically generated

**O*ut*put**

#### **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))

A computer screen shot of a computer code

Description automatically generated**Output**

#### **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**

A screenshot of a computer code

Description automatically generated

#### **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, the 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;

#### A screenshot of a computer Description automatically generated **Output**

### **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);

**A close-up of a computer screen

Description automatically generatedOutput**

#### **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 endency

#### **Code**

calculate\_stats <- function(dataset, columns) {

for (column\_name in columns) {

column\_data <- dataset[[column\_name]]

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 {

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))

#### A screen shot of a computer Description automatically generated**Output**

#### **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)

A screenshot of a computer

Description automatically generated

#### **Output**

#### **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)

A screenshot of a computer program

Description automatically generatedoversampled\_data

#### **Output**

#### **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**

A screenshot of a computer

Description automatically generated

#### **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.