INTRODUCTION TO DATA SCIENCE
MID PROJECT SEC - D GROUP - 4

NAME	ID	CONTRIBUTION
SADMAN SAMIR RAFITH	22-46018-1	25%
LABONI SOMODDAR	22-47301-1	25%
ABID HASSAN BHUIYAN	22-46571-1	25%
MOHAMMAD ARIYAN PATHAN	22-46011-1	25%

Dataset Description:

The given dataset is a modified version of the "Loan Approval Classification Dataset" available in Kaggle. The dataset contains information about individuals seeking loan approval. This is a supervised dataset with the target variable being "loan_status" containing 2 categorical values: 0 (Rejected) and 1 (Accepted).

Apart from the target variable, this dataset contains 13 attributes (excluding the target) with a mixture of numeric and categorical types. The total dataset contains 201 instances. A brief overview of the dataset attribute types (with the total class values for categorical) is given below:

Column Name	Description	Type
person_age	Age of the person	Numeric (integer)
person_gender	Gender of the person	Categorical (male, female)
person_eduation	Highest education level of the person	Categorical (Associate, Bachelor, Doctorate, High School, Master)
person_income	Annual income	Numeric (Continuous)
person_emp_exp	Years of employment experience	Numeric (integer)
person_home_ownership	Home ownership status	Categorical (MORTGAGE, OWN, OTHER, RENT)
loan_amnt	Loan amount request	Numeric (Continuous)
loan_intent	Purpose of the loan	Categorical (PERSONAL, EDUCATION, MEDICAL, VENTURE, DEBTCONSOLIDATION)
loan_int_rate	Loan interest rate	Numeric (Continuous)
loan_percent_income	Loan amount as a percentage of annual income	Numeric (Continuous)
cb_person_cred_hist_length	Length of credit history in years	Numeric (integer)
credit_score	Credit score of the person	Numeric (Continuous)
previous_loan_defaults_on_file	Indicator of previous loan defaults	Categorical (YES, NO)
loan_status	Loan approval status	Categorical (accepted, rejected)

The primary questions explored in this project were:

- What are the key demographic and financial factors that influence whether a loan gets approved or rejected?
- How do categorical attributes (such as education level, gender, and home ownership) impact loan approval?
- What relationships or correlations exist among the numeric variables (income, loan amount, credit score, etc.)?
- How can missing values, outliers, and class imbalance be effectively handled to ensure the dataset is suitable for modeling?
- After preprocessing, does the dataset show a more balanced, reliable structure for future predictive analysis?

1. Load all the libraries needed:
Code: install.packages("dplyr") install.packages('openxlsx') install.packages("stringdist") install.packages("corrplot") library(dplyr) library(openxlsx) library(stringdist) library(corrplot)
Description: **dplyr**: To manipulate the column & row contents of dataframes. **openxlsx**: Open, Read & Write to an Excel file. **stringdist**: Matching strings with predefined valid values. **Corrplot**: used to visualize correlation matrices in a graphical and easy-to-interpret way**.**

2. Load the data:
Code: data <- read.xlsx("Midterm_Dataset_Section(C).xlsx")
Description: By using the ‘openxlsx’ library, the Excel file contents were converted to an R data frame.

3. Check the data summary:

Code:

str(data)
summary(data)

Description:
str(data) shows a small overview of the columns in 'data'. And **summary(data)** shows a short summary of each column (minimum and maximum values, mean, median, 1st quartile, and 3rd quartile values for numeric attributes, and the instance count for categorical attributes and the number of missing values for all attributes).

Screenshot:

```
> str(data)
'data.frame':	201 obs. of  14 variables:
 $ person_age                    : num  21 21 25 23 24 NA 22 24 22 21 ...
 $ person_gender                 : chr  "female" "female" "female" "female" ...
 $ person_education              : chr  "Master" "High School" "High School" "Bachelor" ...
 $ person_income                 : num  71948 12282 12438 79753 66135 ...
 $ person_emp_exp                : num  0 0 3 0 1 0 1 5 3 0 ...
 $ person_home_ownership         : chr  "RENT" "OWN" "MORTGAGE" "RENT" ...
 $ loan_amnt                     : num  35000 1000 5500 35000 35000 2500 35000 35000 35000 1600 ...
 $ loan_intent                   : chr  "PERSONAL" "EDUCATION" "MEDICAL" "MEDICAL" ...
 $ loan_int_rate                 : num  16 11.1 12.9 15.2 14.3 ...
 $ loan_percent_income           : num  0.49 NA 0 0.44 0.53 0.19 0.37 0.37 0.35 0.13 ...
 $ cb_person_cred_hist_length    : num  3 2 3 2 4 2 3 4 2 3 ...
 $ credit_score                  : num  561 504 635 675 586 532 701 585 544 640 ...
 $ previous_loan_defaults_on_file: chr  "No" "Yes" "No" "No" ...
 $ loan_status                   : num  1 0 1 1 1 1 1 1 NA 1 ...
> summary(data)
   person_age     person_gender      person_education   person_income     person_emp_exp    person_home_ownership   loan_amnt    
 Min.   : 21.00   Length:201         Length:201         Min.   :  12282   Min.   :  0.000   Length:201            Min.   : 1000  
 1st Qu.: 22.00   Class :character   Class :character   1st Qu.:  60501   1st Qu.:  0.000   Class :character      1st Qu.:10000  
 Median : 23.00   Mode  :character   Mode  :character   Median :  85284   Median :  1.000   Mode  :character      Median :25000  
 Mean   : 27.39                                         Mean   : 149875   Mean   :  2.761                         Mean   :20553  
 3rd Qu.: 25.00                                         3rd Qu.: 241060   3rd Qu.:  3.000                         3rd Qu.:28000  
 Max.   :350.00                                         Max.   :3138998   Max.   :125.000                         Max.   :35000  
 NA's   :4                                              NA's   :4                                                                 
 loan_intent        loan_int_rate   loan_percent_income cb_person_cred_hist_length  credit_score   previous_loan_defaults_on_file
 Length:201         Min.   : 5.42   Min.   :0.0000      Min.   :2.00                Min.   :484.0   Length:201                    
 Class :character   1st Qu.:10.65   1st Qu.:0.0900      1st Qu.:2.00                1st Qu.:595.0   Class :character              
 Mode  :character   Median :11.83   Median :0.2350      Median :3.00                Median :630.0   Mode  :character              
                    Mean   :12.29   Mean   :0.2293      Mean   :2.99                Mean   :628.5                                 
                    3rd Qu.:14.42   3rd Qu.:0.3425      3rd Qu.:4.00                3rd Qu.:665.0                                 
                    Max.   :20.00   Max.   :0.5300      Max.   :4.00                Max.   :807.0                                 
                                    NA's   :1                                                                                     
  loan_status    
 Min.   :0.0000  
 1st Qu.:0.0000  
 Median :1.0000  
 Mean   :0.6162  
 3rd Qu.:1.0000  
 Max.   :1.0000  
 NA's   :3       
> 
```

4. Check and Remove Duplicate Rows:

Code:

```
nrow(data)
nrow(distinct(data))

distinct_data <- distinct(data)
```

Description:

nrow() returns the number of instances of the dataframe.
distinct() returns another dataset with only the unique instances.
Finally, the dataset returned using **distinct()** has been saved to a new dataframe named ‘distinct_data’.

Screenshot:

```
> nrow(data)
[1] 201
> nrow(distinct(data))
[1] 200
> 
```

5. Annotating Target Attribute:

Code:

```
annotated <- distinct_data
annotated$loan_status <- factor(annotated$loan_status,
                        levels = c(0, 1),
                        labels = c("rejected", "accepted"))
```

Description:

factor() is used to rename/annotate the current attribute values to a new one. ‘levels’ parameter contains the current attribute values in ‘loan_status’ column and it maps them in the following way:
0 -> “rejected” & 1: “accepted”

Screenshot:

Before:	**After:**

<table><tr><th>loan_status</th></tr><tr><td>1</td></tr><tr><td>0</td></tr><tr><td>1</td></tr><tr><td>1</td></tr><tr><td>1</td></tr><tr><td>1</td></tr><tr><td>1</td></tr></table>	<table><tr><th>loan_status</th></tr><tr><td>rejected</td></tr><tr><td>rejected</td></tr><tr><td>rejected</td></tr><tr><td>rejected</td></tr><tr><td>rejected</td></tr><tr><td>rejected</td></tr><tr><td>rejected</td></tr></table>

6. Visualizing the class distribution for categorical columns:

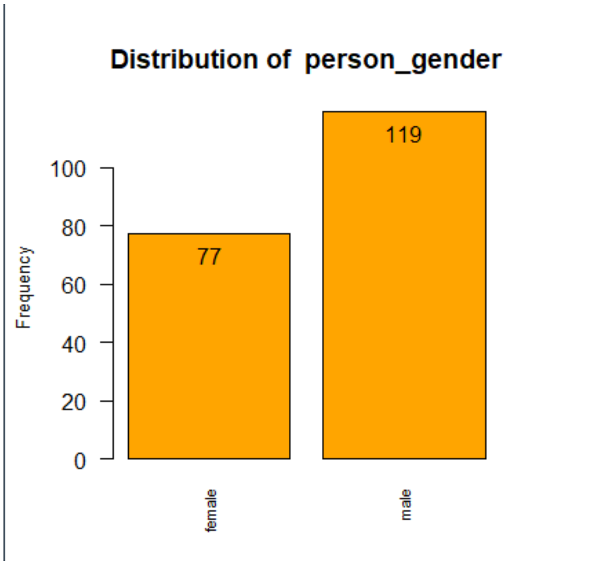
Code:

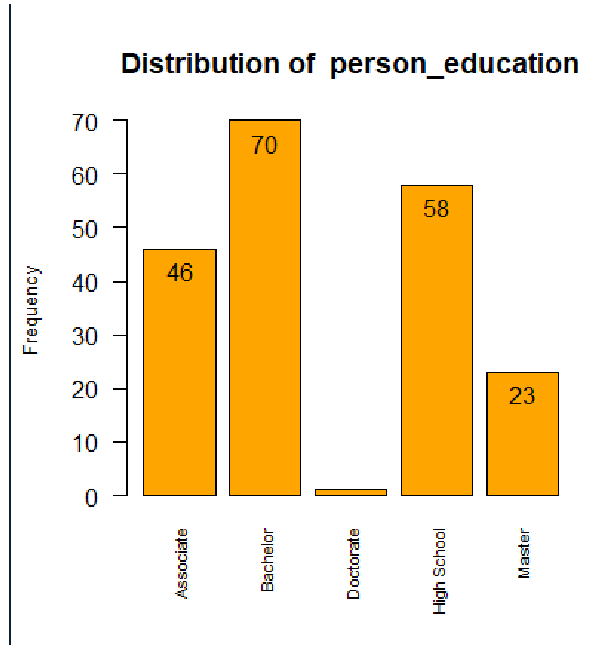
```
categorical_cols <- names(annotated)[sapply(annotated, function(x) is.factor(x) |
is.character(x))]
categorical_cols

plotCategoricalCols <- function(data = annotated, col)
{
  counts <- table(data[[col]])
  bar_positions <- barplot(counts,
                           main = paste("Distribution of ", col),
                           col = "orange",
                           xlab = "",
                           ylab = "Frequency",
                           cex.lab = 0.8,
                           cex.names = 0.9,
                           las = 2)

  text(bar_positions, counts, labels = counts, pos = 1, cex = 1)
}

plotCategoricalCols(annotated, "person_gender")
plotCategoricalCols(annotated, "person_education")
plotCategoricalCols(annotated, "person_home_ownership")
plotCategoricalCols(annotated, "loan_intent")
plotCategoricalCols(annotated, "previous_loan_defaults_on_file")
plotCategoricalCols(annotated, "loan_status")
```

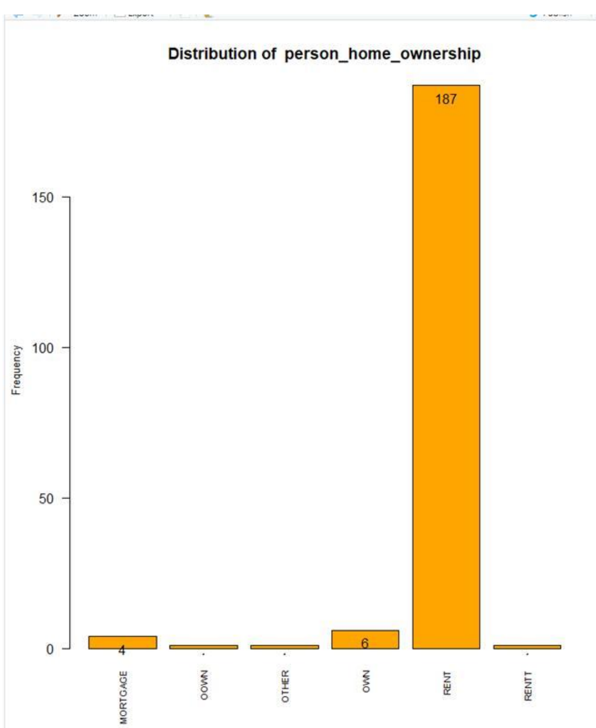
Description:

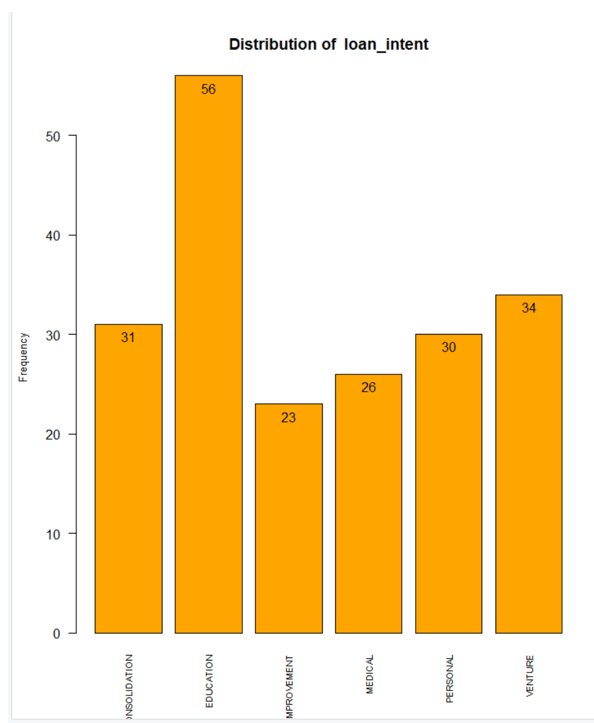
Returns the frequency of unique values of a specific column and visualizes them into a bar plot using the **barplot()** function. The ‘categorical_cols’ array holds the values of the names of all the columns that are holding character or factor type data.

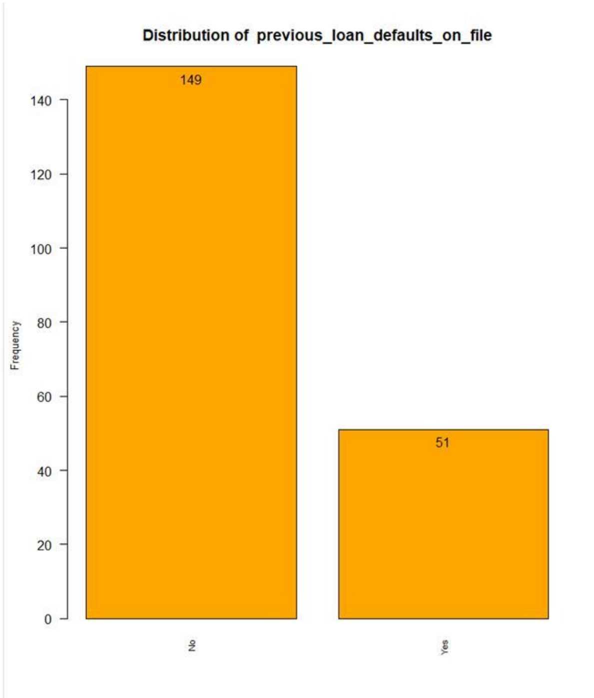
Screenshots:

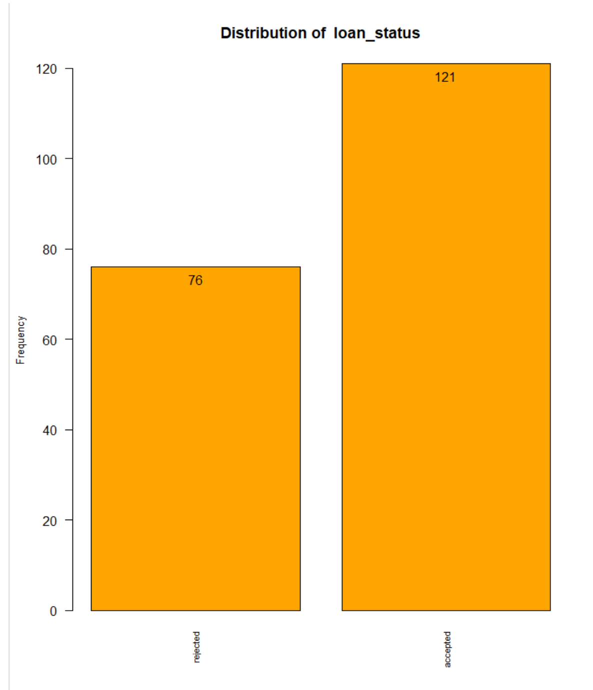












7. Fixing Invalid Values in the Categorical Columns:

Code:

```
fixed_invalid_categorical <- annotated

valid_values <- c("MORTGAGE", "OWN", "OTHER", "RENT")

fix_values <- function(column, valid_values) {
  sapply(column, function(value) {
    closest <- valid_values[which.min(stringdist::stringdist(value, valid_values))]
    return(closest)
  })
}

fixed_invalid_categorical <- fixed_invalid_categorical %>%
  mutate(person_home_ownership = fix_values(person_home_ownership, valid_values))

plotCategoricalCols(fixed_invalid_categorical, "person_home_ownership")
```

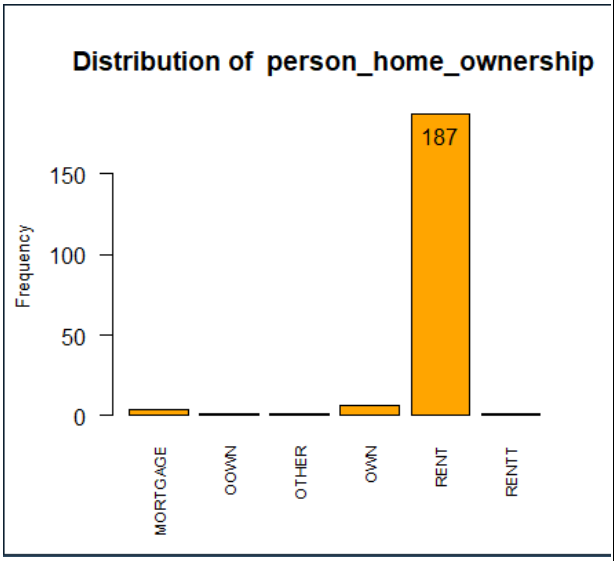
Description:

Fixing the invalid attribute values of a column by matching them with the valid values. If the values are valid, then keep them as they are, but if the values are invalid, replace them with the closest matching valid value.

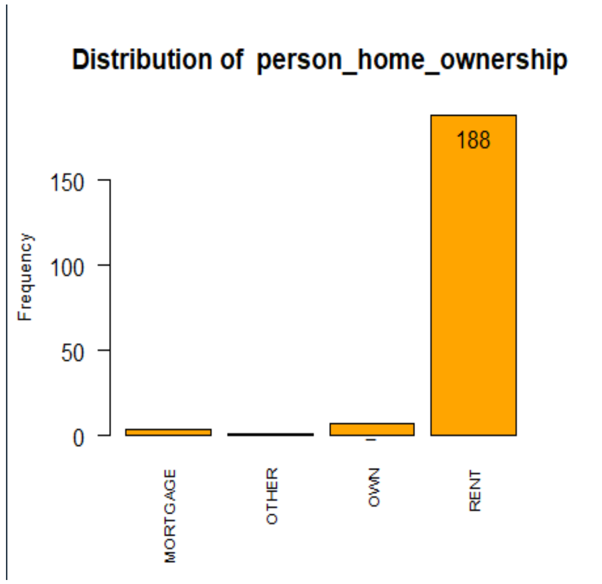
This line: **closest <- valid_values[which.min(stringdist::stringdist(value, valid_values))]**

Matches a 'value' with the list of values in 'valid_values' using the function from 'stringdist' library. If the 'value' is already valid, it will match with the 'valid_values' and if it's an invalid value then it will match with the list of 'valid values' and the closest matching 'valid_value' is returned and replaced from the invalid.

Screenshot (Before correction):



Screenshot (After correction):



8. Convert the values of categorical columns into lowercase letters.

Code:

```
categorical_cols

lowered <- fixed_invalid_categorical
for (col in categorical_cols) {
  lowered[[col]] <- tolower(lowered[[col]])
}
```

Description:

The attribute values of categorical columns were in both capital letters and small letters, which is a critical aspect while mapping them to numeric values. To overcome this issue, all the attributes of all the categorical columns have been converted to lowercase letters using the **tolower()** method.

Screenshots:

Before:

person_home_ownership	loan_amnt	loan_intent
RENT	35000	PERSONAL
OWN	1000	EDUCATION
MORTGAGE	5500	MEDICAL
RENT	35000	MEDICAL
RENT	35000	MEDICAL
OWN	2500	VENTURE
RENT	35000	EDUCATION

After:

person_home_ownership	loan_amnt	loan_intent
rent	35000	personal
own	1000	education
mortgage	5500	medical
rent	35000	medical
rent	35000	medical
own	2500	venture
rent	35000	education

9. Visualizing the missing values

Code:

1.

```
colSums(is.na(lowered[categorical_cols]))
for (col_name in categorical_cols)
{
  cat(col_name, " -> ", which(is.na(lowered[col_name])), "\n")
}
```

2.

```
barplot(colSums(is.na(lowered[categorical_cols])), las = 2, col = "blue",
        main = "Missing Values per Categorical Column",
        xlab = "", ylab = "Count of missing Values",
        cex.lab = 0.9,
        cex.names = 0.9)
```

Description:

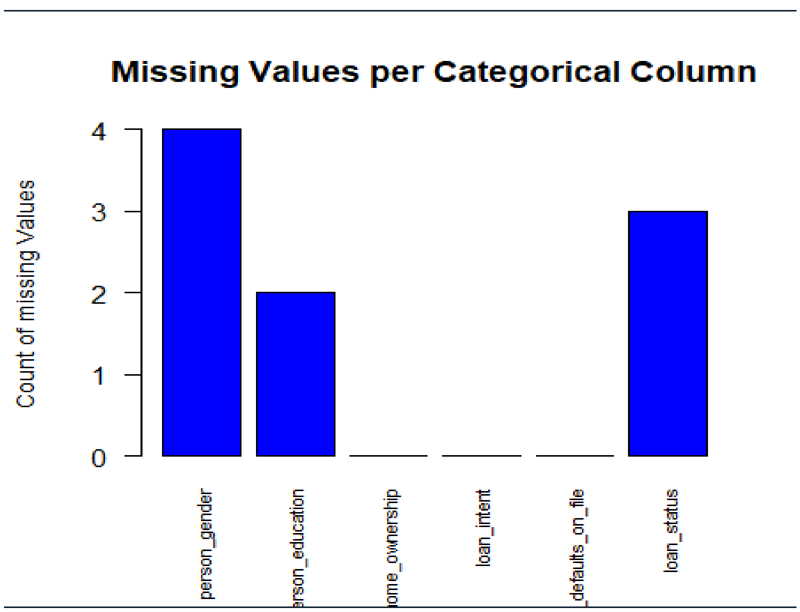
The 1st code snippet returns the number of missing values for every categorical column using the **colSums()** function that sums all the occurrences of ‘TRUE’ values returned by **is.na()** function.

The 2nd snippet shows the instance index where the missing values are present for each of the categorical columns. For this **which()** function is used which returns the indexes of instances where at least one attribute value is ‘NA’.

The 3rd code snippet returns a bar plot showing the missing values in all the categorical columns using the **barplot()** function.

Screenshot:
1st & 2nd code output:

```
> colSums(is.na(lowered[categorical_cols]))
               person_gender             person_education       person_home_ownership                 loan_intent
                           4                            2                           0                           0
previous_loan_defaults_on_file                  loan_status
                           0                            3
> for (col_name in categorical_cols)
+ {
+   cat(col_name, " -> ", which(is.na(lowered[col_name])), "\n")
+ }
person_gender  ->  8 17 189 197
person_education  ->  9 16
person_home_ownership  ->
loan_intent  ->
previous_loan_defaults_on_file  ->
loan_status  ->  9 15 18
>
```



10. Discard rows with NULL values for Categorical Columns

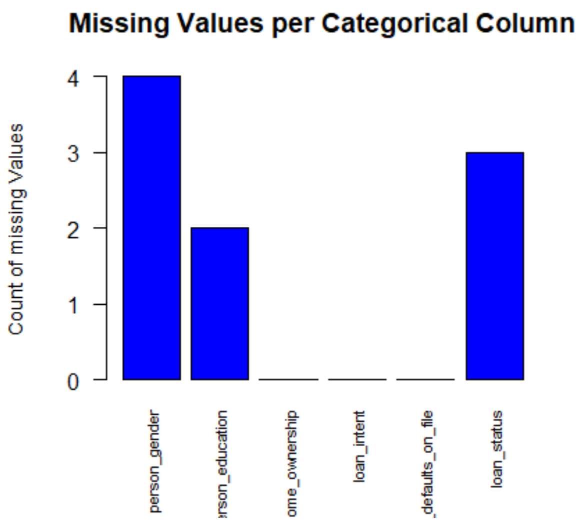
Code:

```
discraded_null <- lowered
discraded_null <- na.omit(discraded_null)
```

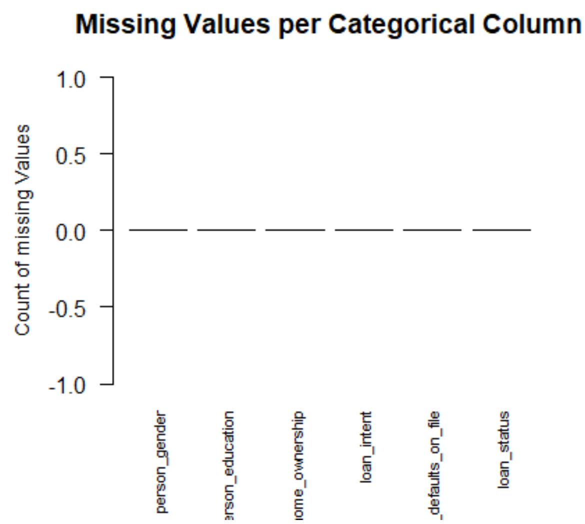
Description:
This is one of the techniques to handle null values. This technique removes all the instances containing NULL or missing values. The **na.omit(discraded_null)** function returns the dataset with all of its null values removed.

Screenshot:

Before:



After:



11. Handle NULL values with Top down Approach for Categorical Columns.

Code:

```
top_down <- lowered
categorical_cols <- names(top_down)[sapply(top_down, function(x) is.factor(x) |
is.character(x))]
for (col in categorical_cols) {
  for (i in seq_len(nrow(top_down))[-1]) {
```

```
    if (is.na(top_down[[col]][i])) {
      top_down[[col]][i] <- top_down[[col]][i - 1]
    }
  }
}
```

Description:
This technique replaces the NULL values with the previous instance value of the same column. A loop is running till the end of the column and checking if any instance value is null or not; if the condition finds any null values then replace them with the previous value. Here is the condition: **if (is.na(bottom_up[[col]][i])).**
top_down[[col]][i] <- top_down[[col]][i - 1] this line replacing the previous instance value with the NULL value.

Screenshot:

Before:

person_age	person_gender	person_education
21	female	high school
22	female	high school
21	female	associate
23	male	bachelor
NA	male	master
23	female	associate
23	female	NA
23	NA	bachelor
23	female	high school
23	male	bachelor
24	female	master

After:

person_age	person_gender	person_education
21	female	high school
22	female	high school
21	female	associate
23	male	bachelor
NA	male	master
23	female	associate
23	female	associate
23	female	bachelor
23	female	high school
23	male	bachelor
24	female	master

12. Handle NULL values with Bottom Up Approach for Categorical Columns.

Code:

```
bottom_up <- lowered
categorical_cols <- names(bottom_up)[sapply(bottom_up, function(x) is.factor(x) |
is.character(x))]

for (col in categorical_cols) {
  for (i in seq_len(nrow(bottom_up) - 1)) {
    if (is.na(bottom_up[[col]][i])) {
      bottom_up[[col]][i] <- bottom_up[[col]][i + 1]
    }
  }
```

```
}
```

Description:
This technique handles the NULL values by replacing them with the value of the next instance of the same column.
A loop runs till the end of a column and checks if any instance is NULL or not. If the condition finds any NULL value then replace the value with the next instance value .Here is the condition: **if (is.na(bottom_up[[col]][i]))**
bottom_up[[col]][i] <- bottom_up[[col]][i + 1] this line replacing the NULL value with the next instance value.

Screenshot:

Before:

person_age	person_gender	person_education
21	female	high school
22	female	high school
21	female	associate
23	male	bachelor
NA	male	master
23	female	associate
23	female	NA
23	NA	bachelor
23	female	high school
23	male	bachelor
24	female	master

After:

person_age	person_gender	person_education
21	female	high school
22	female	high school
21	female	associate
23	male	bachelor
NA	male	master
23	female	associate
23	female	bachelor
23	female	bachelor
23	female	high school
23	male	bachelor
24	female	master

13. Replace NULL values with MODE for categorical columns

Code:

```
most_frequent_data <- lowered
categorical_cols <- names(most_frequent_data)[sapply(most_frequent_data, function(x)
is.factor(x) | is.character(x))]

for (col in categorical_cols) {
  most_frequent <- names(sort(table(most_frequent_data[[col]]), decreasing = TRUE))[1]

  most_frequent_data[[col]][which(is.na(most_frequent_data[[col]]))] <- most_frequent
}
```

Description:
Mode is the most frequent value of the whole column, and for handling null values for categorical columns, the mode value (the attribute value with more instances) was used.
A loop runs till the end of the column for finding Null values. If the condition finds a null value then replace them with mode value.
names(sort(table(most_frequent_data[[col]]), decreasing = TRUE))[1], this line of code returns the mode value of a column.
most_frequent_data[[col]][which(is.na(most_frequent_data[[col]]))] <- most_frequent, this line of code replacing the null values with the mode value.

Screenshot:

Before:

person_age	person_gender	person_education
21	female	master
21	female	high school
25	female	high school
23	female	bachelor
24	male	master
NA	female	high school
22	female	bachelor
24	NA	high school
22	female	NA
21	female	high school
22	female	high school

After:

person_age	person_gender	person_education
21	female	master
21	female	high school
25	female	high school
23	female	bachelor
24	male	master
NA	female	high school
22	female	bachelor
24	male	high school
22	female	bachelor
21	female	high school
22	female	high school

For handling the missing values of categorical columns, Mean and Median cannot be used. Generally, the most frequent value is used to handle the missing values. So, Mode has been used to replace all the “NA” values.

14. Label Encoding for the Categorical Columns

Code:

```
label_encoded <- most_frequent_data

gender <- unique(label_encoded$person_gender)
label_encoded$person_gender <- factor(label_encoded$person_gender,
                              levels = gender,
                              labels = 0:(length(gender) - 1))

education <- unique(label_encoded$person_education)
label_encoded$person_education <- factor(label_encoded$person_education,
```

```
                                    levels = education,
                                    labels = 0:(length(education) - 1))

home_ownership <- unique(label_encoded$person_home_ownership)
label_encoded$person_home_ownership <- factor(label_encoded$person_home_ownership,
                                        levels = home_ownership,
                                        labels = 0:(length(home_ownership) - 1))

intent <- unique(label_encoded$loan_intent)
label_encoded$loan_intent <- factor(label_encoded$loan_intent,
                                levels = intent,
                                labels = 0:(length(intent) - 1))

loan_defaults_on_file <- unique(label_encoded$previous_loan_defaults_on_file)
label_encoded$previous_loan_defaults_on_file <-
factor(label_encoded$previous_loan_defaults_on_file,
                                                levels = loan_defaults_on_file,
                                                labels = 0:(length(loan_defaults_on_file) - 1))
```

Description:

This technique maps all the attributes of a categorical column to a numeric value.
gender <- unique(label_encoded$person_gender) , this line of code takes all the unique values of a categorical column.
label_encoded$person_gender <- factor(label_encoded$person_gender, this line of code maps all the unique values to a numeric value.

Screenshot:

Before:

person_age	person_gender	person_education	person_income	person_emp_exp	person_home_ownership
21	female	master	71948	0	rent
21	female	high school	12282	0	own
25	female	high school	12438	3	mortgage
23	female	bachelor	79753	0	rent
24	male	master	66135	1	rent
NA	female	high school	12951	0	own
22	female	bachelor	NA	1	rent
24	male	high school	95550	5	rent
22	female	bachelor	100684	3	rent
21	female	high school	12739	0	own
22	female	high school	102985	0	rent
21	female	associate	13113	0	own
23	male	bachelor	114860	3	rent
NA	male	master	130713	0	rent

After:

person_age	person_gender	person_education	person_income	person_emp_exp	person_home_ownership
21	0	0	71948	0	0
21	0	1	12282	0	1
25	0	1	12438	3	2
23	0	2	79753	0	0
24	1	0	66135	1	0
NA	0	1	12951	0	1
22	0	2	NA	1	0
24	1	1	95550	5	0
22	0	2	100684	3	0
21	0	1	12739	0	1
22	0	1	102985	0	0
21	0	3	13113	0	1
23	1	2	114860	3	0
NA	1	0	130713	0	0

15. Summary of the Numeric Columns

Code:

```
str(label_encoded)
numeric_cols <- names(label_encoded)[sapply(label_encoded, is.numeric)]
summary(label_encoded[numeric_cols])
```

Description:

str(data) shows a small overview of the numeric columns. And **summary()** shows a short summary of each numeric column (minimum and maximum values, mean, median, 1st quartile, and 3rd quartile values for numeric attributes and the number of missing values for all attributes).

Screenshot:

```
+                                                    labels = 0:(length(loan_defaults_on_file) - 1))
> str(label_encoded)
'data.frame':	200 obs. of  14 variables:
 $ person_age                    : num  21 21 25 23 24 NA 22 24 22 21 ...
 $ person_gender                 : Factor w/ 2 levels "0","1": 1 1 1 1 2 1 1 2 1 1 ...
 $ person_education              : Factor w/ 5 levels "0","1","2","3",..: 1 2 2 3 1 2 3 2 3 2 ...
 $ person_income                 : num  71948 12282 12438 79753 66135 ...
 $ person_emp_exp                : num  0 0 3 0 1 0 1 5 3 0 ...
 $ person_home_ownership         : Factor w/ 4 levels "0","1","2","3": 1 2 3 1 1 2 1 1 1 2 ...
 $ loan_amnt                     : num  35000 1000 5500 35000 35000 2500 35000 35000 35000 1600 ...
 $ loan_intent                   : Factor w/ 6 levels "0","1","2","3",..: 1 2 3 3 3 4 2 3 1 4 ...
 $ loan_int_rate                 : num  16 11.1 12.9 15.2 14.3 ...
 $ loan_percent_income           : num  0.49 NA 0 0.44 0.53 0.19 0.37 0.37 0.35 0.13 ...
 $ cb_person_cred_hist_length    : num  3 2 3 2 4 2 3 4 2 3 ...
 $ credit_score                  : num  561 504 635 675 586 532 701 585 544 640 ...
 $ previous_loan_defaults_on_file: Factor w/ 2 levels "0","1": 1 2 1 1 1 1 1 1 1 1 ...
 $ loan_status                   : chr  "accepted" "rejected" "accepted" "accepted" ...
> numeric_cols <- names(label_encoded)[sapply(label_encoded, is.numeric)]
> summary(label_encoded[numeric_cols])
   person_age     person_income     person_emp_exp     loan_amnt      loan_int_rate   loan_percent_income cb_person_cred_hist_length
 Min.   : 21.00   Min.   :  12282   Min.   :  0.00   Min.   : 1000   Min.   : 5.42   Min.   :0.0000      Min.   :2.00
 1st Qu.: 22.00   1st Qu.:  60342   1st Qu.:  0.00   1st Qu.:10000   1st Qu.:10.65   1st Qu.:0.0900      1st Qu.:2.00
 Median : 23.00   Median :  86048   Median :  1.00   Median :25000   Median :11.85   Median :0.2300      Median :3.00
 Mean   : 27.42   Mean   : 150236   Mean   :  2.77   Mean   :20493   Mean   :12.30   Mean   :0.2284      Mean   :2.99
 3rd Qu.: 25.00   3rd Qu.: 241074   3rd Qu.:  3.00   3rd Qu.:28000   3rd Qu.:14.45   3rd Qu.:0.3400      3rd Qu.:4.00
 Max.   :350.00   Max.   :3138998   Max.   :125.00   Max.   :35000   Max.   :20.00   Max.   :0.5300      Max.   :4.00
 NA's   :4        NA's   :4                                                          NA's   :1
  credit_score
 Min.   :484.0
 1st Qu.:594.8
 Median :629.0
 Mean   :628.2
 3rd Qu.:664.2
 Max.   :807.0

> 
```

16. Plotting the Numeric Columns

Code:

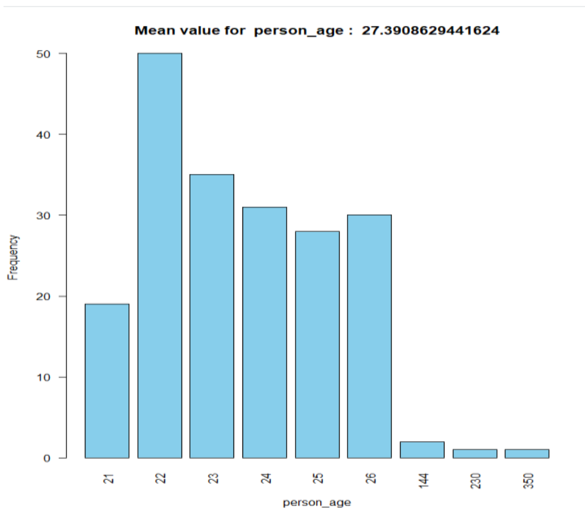
```
plotFreq <- function(col_name)
{
  # Create bar plot
  barplot(table(data[[col_name]]),
          main = paste("Mean value for ", col_name, ": ", mean(data[[col_name]], na.rm =
TRUE)),
          col = "skyblue",
          xlab = col_name,
          ylab = "Frequency",
          las = 2)
}

plotFreq("person_age")
plotFreq("person_income")
plotFreq("person_emp_exp")
plotFreq("loan_amnt")
```

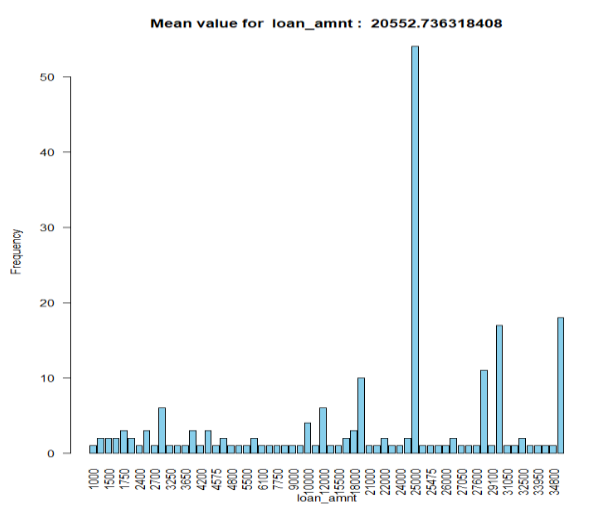
```
plotFreq("loan_int_rate")
plotFreq("loan_percent_income")
plotFreq("cb_person_cred_hist_length")
plotFreq("credit_score")
```

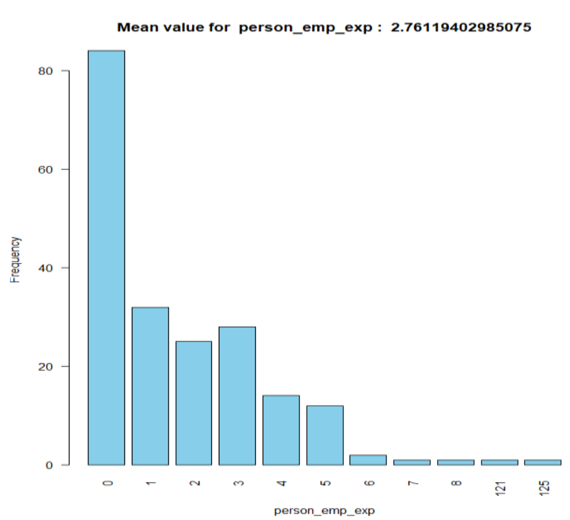
Description:

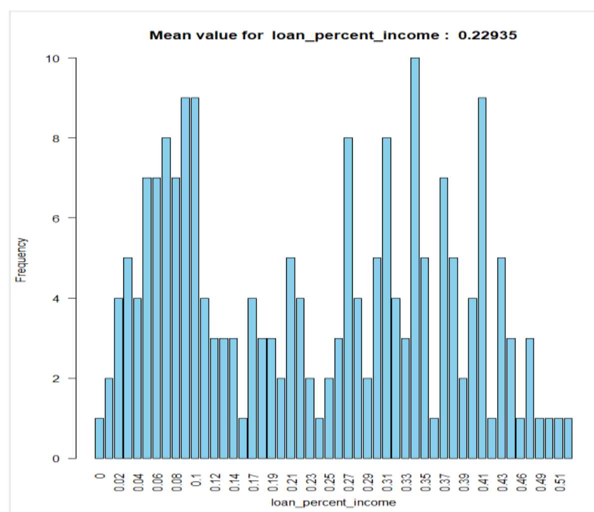
This code snippet returns the frequency of the values in all the numeric columns. Then the frequency is shown using a **barplot().**

Screenshot:









17. Plotting the NULL values of Numeric Columns

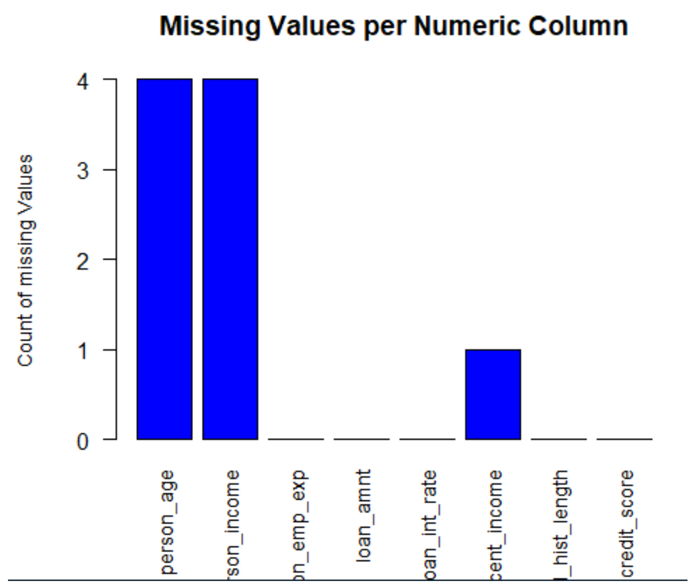
Code:

```
colSums(is.na(label_encoded))
for (col_name in numeric_cols)
{
  cat(col_name, " -> ", which(is.na(label_encoded[col_name])), "\n")
}

barplot(colSums(is.na(label_encoded[numeric_cols])), las = 2, col = "blue",
        main = "Missing Values per Numeric Column",
        xlab = "", ylab = "Count of missing Values",
        cex.lab = 0.9,
        cex.names = 0.9)
```

Description:

This code snippet returns a plot with all the numeric columns containing missing or NULL values.

Screenshot:



18. Discard numeric entries with missing or NULL values for Numerical Columns.

Code:

```
discraded_null_numeric <- label_encoded
discraded_null_numeric <- na.omit(discraded_null_numeric)
```

Description:
This is one of the techniques to handle null values. This technique removes all the instances containing NULL or missing values. The **na.omit(discraded_null_numeric)** function returns the dataset with all of its null values removed from the numeric columns.

Screenshot:

Before:

person_age	person_gender	person_education	person_income
21	0	0	71948
21	0	1	12282
25	0	1	12438
23	0	2	79753
24	1	0	66135
NA	0	1	12951
22	0	2	NA
24	1	1	95550
22	0	2	100684
21	0	1	12739
22	0	1	102985
21	0	3	13113
23	1	2	114860
NA	1	0	130713
23	0	3	3138998

ing 1 to 15 of 200 entries, 14 total columns

After:

person_age	person_gender	person_education	person_income
21	0	0	71948
25	0	1	12438
23	0	2	79753
24	1	0	66135
24	1	1	95550
22	0	2	100684
21	0	1	12739
22	0	1	102985
21	0	3	13113
23	1	2	114860
23	0	3	3138998
23	1	2	144943
23	0	1	111369
23	1	2	136628
24	0	0	14283

ing 1 to 15 of 191 entries, 14 total columns

19. Handling NULL values with Top Down Approach for Numerical Columns.

Code:

```
top_down_numeric_null <- label_encoded

for (col in numeric_cols) {
  for (i in seq_len(nrow(top_down_numeric_null))[-1]) {
    if (is.na(top_down_numeric_null[[col]][i])) {
      top_down_numeric_null[[col]][i] <- top_down_numeric_null[[col]][i - 1]
    }
  }
}
```

Description:

This approach is for replacing NULL values using the previous value of the column. This is a similar technique with 11 no technique.

Screenshot:

Before:

person_age	person_gender	person_education	person_income
21	0	0	71948
21	0	1	12282
25	0	1	12438
23	0	2	79753
24	1	0	66135
NA	0	1	12951
22	0	2	NA
24	1	1	95550
22	0	2	100684
21	0	1	12739
22	0	1	102985
21	0	3	13113
23	1	2	114860
NA	1	0	130713
23	0	3	3138998

ing 1 to 15 of 200 entries, 14 total columns

After:

person_age	person_gender	person_education	person_income
21	0	0	71948
21	0	1	12282
25	0	1	12438
23	0	2	79753
24	1	0	66135
24	0	1	12951
22	0	2	12951
24	1	1	95550
22	0	2	100684
21	0	1	12739
22	0	1	102985
21	0	3	13113
23	1	2	114860
23	1	0	130713
23	0	3	3138998

ing 1 to 15 of 200 entries, 14 total columns

20. Handling NULL values with Bottom Up Approach for Numerical Columns.

Code:

```
bottom_up_numeric_null <- label_encoded

for (col in numeric_cols) {
  for (i in seq_len(nrow(bottom_up_numeric_null) - 1)) {
    if (is.na(bottom_up_numeric_null[[col]][i])) {
      bottom_up_numeric_null[[col]][i] <- bottom_up_numeric_null[[col]][i + 1]
    }
  }
}
```

Description:

This approach is for replacing NULL values using the next value of the column. This is a similar technique with 12 no technique.

Screenshot:

Before:

person_age	person_gender	person_education	person_income
21	0	0	71948
21	0	1	12282
25	0	1	12438
23	0	2	79753
24	1	0	66135
NA	0	1	12951
22	0	2	NA
24	1	1	95550
22	0	2	100684
21	0	1	12739
22	0	1	102985
21	0	3	13113
23	1	2	114860
NA	1	0	130713
23	0	3	3138998

ving 1 to 15 of 200 entries, 14 total columns

After:

person_age	person_gender	person_education	person_income
21	0	0	71948
21	0	1	12282
25	0	1	12438
23	0	2	79753
24	1	0	66135
22	0	1	12951
22	0	2	95550
24	1	1	95550
22	0	2	100684
21	0	1	12739
22	0	1	102985
21	0	3	13113
23	1	2	114860
23	1	0	130713
23	0	3	3138998

ving 1 to 15 of 200 entries, 14 total columns

21. Handling NULL values with MODE Approach for Numerical Columns.

Code:

```
mode_replaced_numeric_null <- label_encoded

for (col in numeric_cols) {
  most_frequent <- names(sort(table(mode_replaced_numeric_null[[col]]), decreasing =
TRUE))[1]
  mode_replaced_numeric_null[[col]][which(is.na(mode_replaced_numeric_null[[col]]))] <-
most_frequent
}
```

Description:

This approach is for replacing NULL values using the MODE value of the column. This is a similar technique with 13 no technique.

Screenshot:

Before:

After:

person_age	person_gender	person_education	person_income
21	0	0	71948
21	0	1	12282
25	0	1	12438
23	0	2	79753
24	1	0	66135
NA	0	1	12951
22	0	2	NA
24	1	1	95550
22	0	2	100684
21	0	1	12739
22	0	1	102985
21	0	3	13113
23	1	2	114860
NA	1	0	130713
23	0	3	3138998

ving 1 to 15 of 200 entries, 14 total columns

person_age	person_gender	person_education	person_income
21	0	0	71948
21	0	1	12282
25	0	1	12438
23	0	2	79753
24	1	0	66135
22	0	1	12951
22	0	2	15229
24	1	1	95550
22	0	2	100684
21	0	1	12739
22	0	1	102985
21	0	3	13113
23	1	2	114860
22	1	0	130713
23	0	3	3138998

ving 1 to 15 of 200 entries, 14 total columns

22. Handling NULL values with MEAN Approach for Numerical Columns.

Code:

```
mean_replaced_numeric_null <- label_encoded

for (col in numeric_cols) {
  if (any(is.na(mean_replaced_numeric_null[[col]]))) {
    mean_value <- round(mean(mean_replaced_numeric_null[[col]], na.rm = TRUE))
    mean_replaced_numeric_null[[col]][which(is.na(mean_replaced_numeric_null[[col]]))] <-
mean_value
  }
}
```

Description:

This approach is for replacing NULL values with the MEAN value of the column.
A loop is running until the last attribute of a column to check for NULL values.
if (any(is.na(mean_replaced_numeric_null[[col]]))), this is the condition to check for NULL values and if the condition finds a NULL value it replace it with the MEAN value.
mean_replaced_numeric_null[[col]][which(is.na(mean_replaced_numeric_null[[col]]))] <- mean_value, this is the code for replacing the NULL value with MEAN value.

Screenshot:

Before:	After:

person_age	person_gender	person_education	person_income
21	0	0	71948
21	0	1	12282
25	0	1	12438
23	0	2	79753
24	1	0	66135
NA	0	1	12951
22	0	2	NA
24	1	1	95550
22	0	2	100684
21	0	1	12739
22	0	1	102985
21	0	3	13113
23	1	2	114860
NA	1	0	130713
23	0	3	3138998

ving 1 to 15 of 200 entries, 14 total columns

person_age	person_gender	person_education	person_income
21	0	0	71948
21	0	1	12282
25	0	1	12438
23	0	2	79753
24	1	0	66135
27	0	1	12951
22	0	2	150236
24	1	1	95550
22	0	2	100684
21	0	1	12739
22	0	1	102985
21	0	3	13113
23	1	2	114860
27	1	0	130713
23	0	3	3138998

ving 1 to 15 of 200 entries, 14 total columns

23. Handling NULL values with MEDIAN Approach for Numerical Columns.

Code:

```
median_replaced_numeric_null <- label_encoded

for (col in numeric_cols) {
  if (any(is.na(median_replaced_numeric_null[[col]]))) {
    median_value <- median(median_replaced_numeric_null[[col]], na.rm = TRUE)
    median_replaced_numeric_null[[col]][which(is.na(median_replaced_numeric_null[[col]]))] <-
median_value
  }
}
```

Description:

This technique replaces the NULL values with the median value of a column.
A loop is running in a column until the column's last attribute to find the NULL value from the column.
if (any(is.na(median_replaced_numeric_null[[col]]))), this is the line of the condition of checking NULL values.
median_replaced_numeric_null[[col]][which(is.na(median_replaced_numeric_null[[col]]))] <- median_value, this replace the median value with the NULL value

Screenshot:

Before:	After:

person_age	person_gender	person_education	person_income
21	0	0	71948
21	0	1	12282
25	0	1	12438
23	0	2	79753
24	1	0	66135
NA	0	1	12951
22	0	2	NA
24	1	1	95550
22	0	2	100684
21	0	1	12739
22	0	1	102985
21	0	3	13113
23	1	2	114860
NA	1	0	130713
23	0	3	3138998

ving 1 to 15 of 200 entries, 14 total columns

person_age	person_gender	person_education	person_income
21	0	0	71948.0
21	0	1	12282.0
25	0	1	12438.0
23	0	2	79753.0
24	1	0	66135.0
23	0	1	12951.0
22	0	2	86047.5
24	1	1	95550.0
22	0	2	100684.0
21	0	1	12739.0
22	0	1	102985.0
21	0	3	13113.0
23	1	2	114860.0
23	1	0	130713.0
23	0	3	3138998.0

ving 1 to 15 of 200 entries, 14 total columns

For handling 'NA' values there are several techniques, but Median has been used because the dataset contains outlier values and mean does not work well when the dataset contains outliers. Aside from the missing values, the distribution of the dataset is skewed. So, the mode value is not suitable for numeric columns. That's why the median has been used.

24. Finding the Standard Deviation for all the Numeric Columns

Code:

median_replaced_numeric_null %>% summarise_if(is.numeric, sd)

Description:

This returns the standard Deviation of all the numeric columns.

Screenshot:

```
> median_replaced_numeric_null %>% summarise_if(is.numeric, sd)
  person_age person_income person_emp_exp loan_amnt loan_int_rate loan_percent_income cb_person_cred_hist_length credit_score
1   29.72641      237273.4       12.24199  10740.73      3.156219           0.1408948                  0.7829207     50.61006
> 
```

25. Applying Z score to check and handle all the Outlier in the Numerical Columns

Code:

```
z_score_outlier_handeled <- median_replaced_numeric_null
for (col in numeric_cols) {
  z_scores <- scale(z_score_outlier_handeled[[col]])

  z_score_outlier_handeled <- z_score_outlier_handeled[abs(z_scores) <= 3, ]
}
```

Description:

This technique helps to check all the outliers in the numeric columns. This identifies and handles all the outliers from the numeric values.
z_scores <- scale(z_score_outlier_handeled[[col]]), this line returns all the outliers.
z_score_outlier_handeled <- z_score_outlier_handeled[abs(z_scores) <= 3, by this line of code all the outliers have been handled in between the z-score 3. If the value of z is greater than 3, it marks those values as outliers and discard them.

Screenshot:

Before:

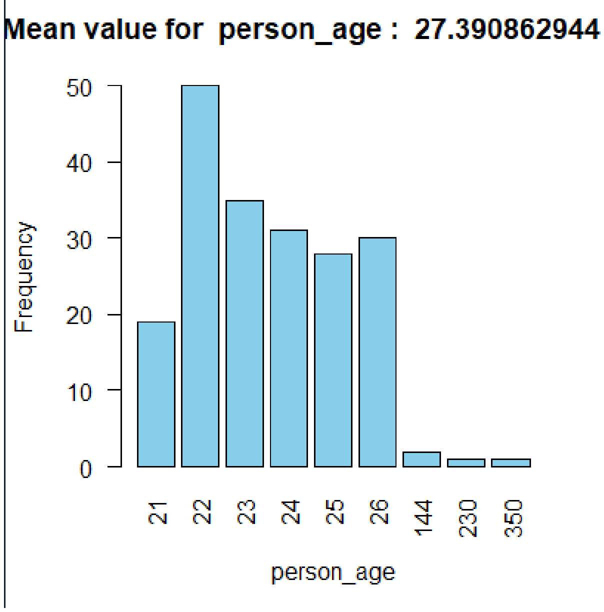
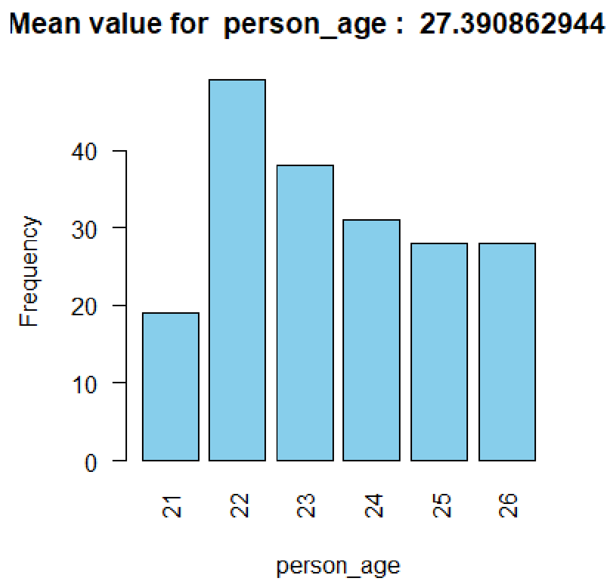
person_age	person_gender	person_education	person_income
23	1	0	130713.0
23	0	3	3138998.0
23	0	2	86047.5
23	1	2	144943.0
23	0	1	111369.0
23	1	2	136628.0
24	0	0	14283.0
25	1	2	195718.0
25	1	1	165792.0
22	0	0	79255.0
24	0	2	13866.0
22	1	2	97420.0
24	0	1	82443.0
21	0	3	14288.0
23	1	1	14293.0
22	0	2	79054.0
21	0	2	14988.0
21	1	1	86047.5
230	1	2	144855.0

wing 14 to 32 of 200 entries, 14 total columns

After:

person_age	person_gender	person_education	person_income
23	1	2	114860.0
23	1	0	130713.0
23	0	2	86047.5
23	1	2	144943.0
23	0	1	111369.0
23	1	2	136628.0
24	0	0	14283.0
25	1	2	195718.0
25	1	1	165792.0
22	0	0	79255.0
24	0	2	13866.0
22	1	2	97420.0
24	0	1	82443.0
21	0	3	14288.0
23	1	1	14293.0
22	0	2	79054.0
21	0	2	14988.0
21	1	1	86047.5
26	1	2	114645.0

ing 12 to 31 of 193 entries, 14 total columns

26. Using IQR to check and handle the outliers

Code:

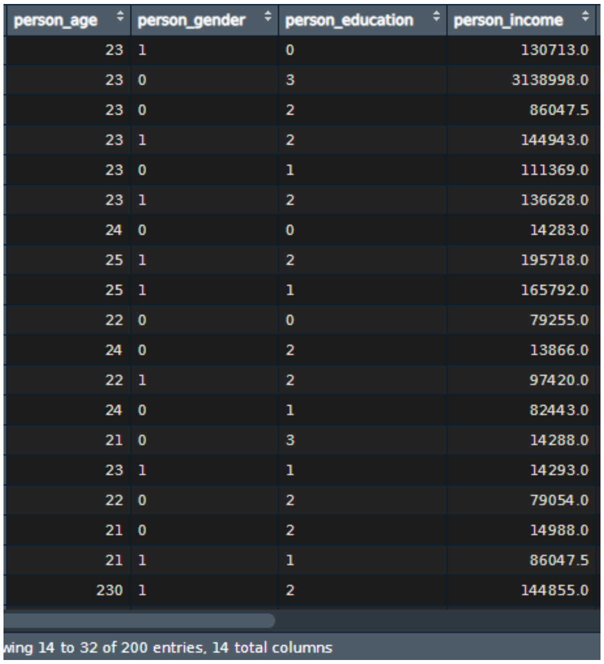
```
iqr_outlier_handled <- median_replaced_numeric_null

for (col in numeric_cols) {
  Q1 <- quantile(iqr_outlier_handled[[col]], 0.25, na.rm = TRUE)
  Q3 <- quantile(iqr_outlier_handled[[col]], 0.75, na.rm = TRUE)
  IQR <- Q3 - Q1

  lower_bound <- Q1 - 1.5 * IQR
  upper_bound <- Q3 + 1.5 * IQR

  iqr_outlier_handled <- iqr_outlier_handled[iqr_outlier_handled[[col]] >= lower_bound &
iqr_outlier_handled[[col]] <= upper_bound, ]
}
```

Description:

This technique finds and handles outliers. **IQR <- Q3 - Q1,** by this line of code, it detects the outliers. If the value is greater than Q3 and less than Q1, then the value is marked as an outlier.
iqr_outlier_handled <- iqr_outlier_handled[iqr_outlier_handled[[col]] >= lower_bound & iqr_outlier_handled[[col]] <= upper_bound, this code is removing all the values that are higher than **upper_bound** and lower than **lower_bound.**

Screenshot:

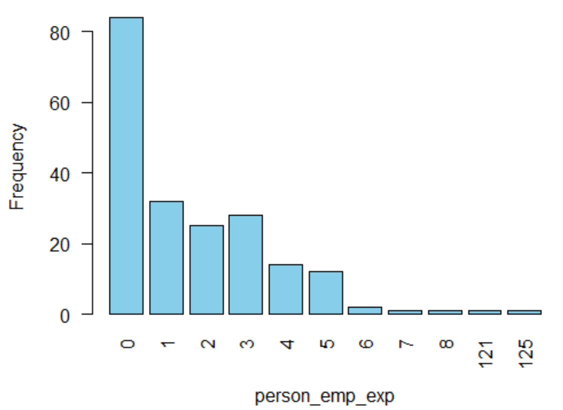
Before:



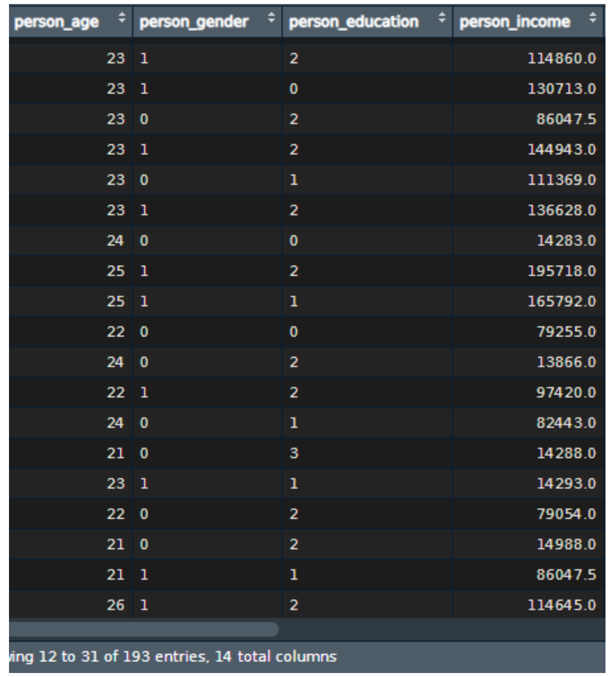
person_age	person_gender	person_education	person_income
23	1	0	130713.0
23	0	3	3138998.0
23	0	2	86047.5
23	1	2	144943.0
23	0	1	111369.0
23	1	2	136628.0
24	0	0	14283.0
25	1	2	195718.0
25	1	1	165792.0
22	0	0	79255.0
24	0	2	13866.0
22	1	2	97420.0
24	0	1	82443.0
21	0	3	14288.0
23	1	1	14293.0
22	0	2	79054.0
21	0	2	14988.0
21	1	1	86047.5
230	1	2	144855.0

wing 14 to 32 of 200 entries, 14 total columns

Mean value for person_emp_exp : 2.7611940298507



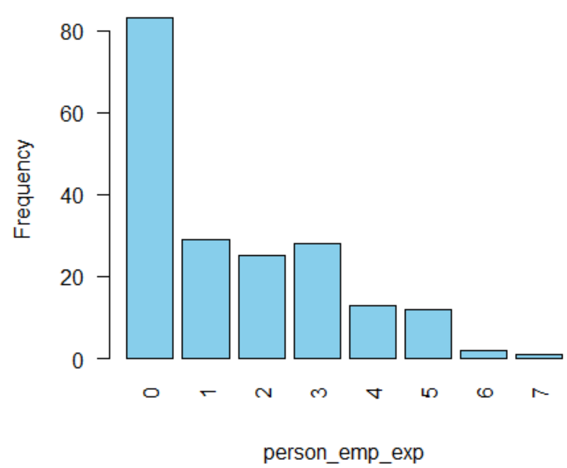
After:



person_age	person_gender	person_education	person_income
23	1	2	114860.0
23	1	0	130713.0
23	0	2	86047.5
23	1	2	144943.0
23	0	1	111369.0
23	1	2	136628.0
24	0	0	14283.0
25	1	2	195718.0
25	1	1	165792.0
22	0	0	79255.0
24	0	2	13866.0
22	1	2	97420.0
24	0	1	82443.0
21	0	3	14288.0
23	1	1	14293.0
22	0	2	79054.0
21	0	2	14988.0
21	1	1	86047.5
26	1	2	114645.0

ing 12 to 31 of 193 entries, 14 total columns

Mean value for person_emp_exp : 2.761194029



27. Applying CHI square to get the range of Numeric columns

Code:

```
sapply(z_score_outlier_handeled[numeric_cols], function(x) if(is.numeric(x)) sd(x, na.rm =
TRUE))
```

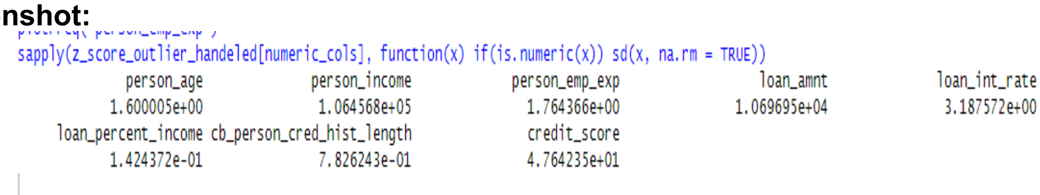
```
chi_squared <- z_score_outlier_handeled

person_income_bins <- cut(chi_squared$person_income, breaks = 4)
levels(person_income_bins)
levels(person_income_bins) <- c("Low", "Lower Middle", "Upper Middle", "High")
chi_squared$person_income <- person_income_bins

amount <- cut(chi_squared$loan_amnt, breaks = 3)
levels(amount)
levels(amount) <- c("Small", "Medium", "Large")
chi_squared$loan_amnt <- amount

str(chi_squared)
```

Description:

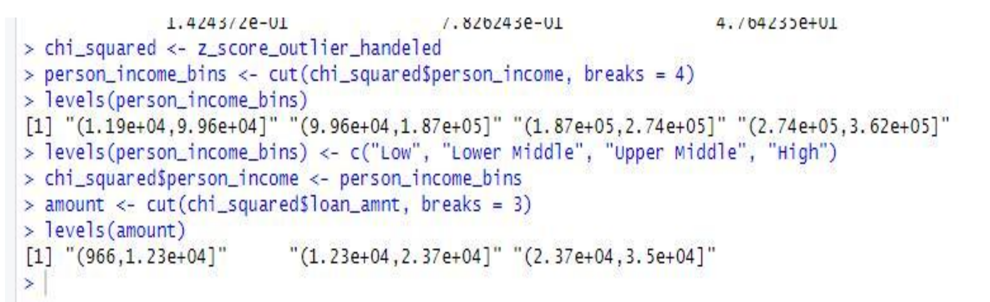
This code helps to convert numeric column to categorical column and helps to find the perfect range for doing so.

cut(chi_squared$person_income, breaks = 4), By this code, the person_income_bins column has been partitioned into 4 categories.
chi_squared$person_income <- person_income_bins, this line of code replacing the numeric values to the categorical values.

Screenshot:



```
sapply(z_score_outlier_handeled[numeric_cols], function(x) if(is.numeric(x)) sd(x, na.rm = TRUE))
          person_age        person_income       person_emp_exp        loan_amnt        loan_int_rate
        1.600005e+00         1.064568e+05         1.764366e+00     1.069695e+04         3.187572e+00
 loan_percent_income cb_person_cred_hist_length         credit_score
        1.424372e-01         7.826243e-01         4.764235e+01
```



```
> chi_squared <- z_score_outlier_handeled
> person_income_bins <- cut(chi_squared$person_income, breaks = 4)
> levels(person_income_bins)
[1] "(1.19e+04,9.96e+04]" "(9.96e+04,1.87e+05]" "(1.87e+05,2.74e+05]" "(2.74e+05,3.62e+05]"
> levels(person_income_bins) <- c("Low", "Lower Middle", "Upper Middle", "High")
> chi_squared$person_income <- person_income_bins
> amount <- cut(chi_squared$loan_amnt, breaks = 3)
> levels(amount)
[1] "(966,1.23e+04]"      "(1.23e+04,2.37e+04]" "(2.37e+04,3.5e+04]"
>
```

Before:	**After:**

person_income	person_emp_exp	person_home_ownership	loan_amnt
71948.0	0	0	35000
12282.0	0	1	1000
12438.0	3	2	5500
79753.0	0	0	35000
66135.0	1	0	35000
12951.0	0	1	2500
86047.5	1	0	35000
95550.0	5	0	35000
100684.0	3	0	35000
12739.0	0	1	1600
102985.0	0	0	35000
13113.0	0	1	4500
114860.0	3	0	35000
130713.0	0	0	35000
86047.5	5	2	30000
144943.0	0	0	35000
111369.0	0	0	35000
136628.0	0	0	35000
14283.0	1	2	1750

person_income	person_emp_exp	person_home_ownership	loan_amnt
Low	0	0	Large
Low	0	1	Small
Low	3	2	Small
Low	0	0	Large
Low	1	0	Large
Low	0	1	Small
Low	1	0	Large
Low	5	0	Large
Low	3	0	Large
Low	0	1	Small
Lower Middle	0	0	Large
Low	0	1	Small
Lower Middle	3	0	Large
Lower Middle	0	0	Large
Low	5	2	Large
Lower Middle	0	0	Large
Lower Middle	0	0	Large
Lower Middle	0	0	Large
Low	1	2	Small

28. Normalizing the numeric values

Code:

```
normalized_numeric  <- chi_squared

col_min <- min(normalized_numeric[["credit_score"]])
col_max <- max(normalized_numeric[["credit_score"]])
normalized_numeric[["credit_score"]] <- (normalized_numeric[["credit_score"]] - col_min) /
(col_max - col_min)

normalized_numeric[["loan_int_rate"]] <- (normalized_numeric[["loan_int_rate"]] / 100)
```

Description:

This technique helps to normalize the numeric value and convert every value in a range on 0-1. The large numbers are squeezed between 0-1 for easy representation.
normalized_numeric[["credit_score"]] <- (normalized_numeric[["credit_score"]] - col_min) / (col_max - col_min), Min-max algorithm has been used to normalize the numeric data.

normalized_numeric[["loan_int_rate"]] <- (normalized_numeric[["loan_int_rate"]] / 100, this line of code convert all the values to 0 - 1.

Screenshot:

Before:

loan_int_rate	loan_percent_income	cb_person_cred_hist_length	credit_score
16.02	0.49	3	561
11.14	0.23	2	504
12.87	0.00	3	635
15.23	0.44	2	675
14.27	0.53	4	586
7.14	0.19	2	532
12.42	0.37	3	701
11.11	0.37	4	585
8.90	0.35	2	544
14.74	0.13	3	640
10.37	0.34	4	621
8.63	0.34	2	651
7.90	0.30	2	573
18.39	0.27	4	708
10.65	0.05	3	670
7.90	0.24	4	663
20.00	0.31	4	694
18.25	0.26	4	709
10.99	0.12	2	679

After:

loan_int_rate	loan_percent_income	cb_person_cred_hist_length	credit_score
0.1602	0.49	3	0.32905983
0.1114	0.23	2	0.08547009
0.1287	0.00	3	0.64529915
0.1523	0.44	2	0.81623932
0.1427	0.53	4	0.43589744
0.0714	0.19	2	0.20512821
0.1242	0.37	3	0.92735043
0.1111	0.37	4	0.43162393
0.0890	0.35	2	0.25641026
0.1474	0.13	3	0.66666667
0.1037	0.34	4	0.58547009
0.0863	0.34	2	0.71367521
0.0790	0.30	2	0.38034188
0.1839	0.27	4	0.95726496
0.1065	0.05	3	0.79487179
0.0790	0.24	4	0.76495726
0.2000	0.31	4	0.89743590
0.1825	0.26	4	0.96153846
0.1099	0.12	2	0.83333333

29. Filtering the numeric values

Code:

```
normalized_numeric_filtered <- median_replaced_numeric_null %>% filter(person_age < 80)
```

Description:
This code is filtering the outliers and replacing those with median values.

Screenshot:

Before:

After:

person_age
22
350
22
21
22
24
24
144
25
26
26
22
23
24
21
22
22
26
22

ving 74 to 93 of 2

person_age
22
22
21
22
24
24
25
26
26
22
23
24
21
22
22
26
22
24
23

ving 74 to 93 of 1

30. Using upsampling in the numeric columns to balance the dataset

Code:

```
balanced_data <- normalized_numeric_filtered
table(balanced_data$loan_status)
plotCategoricalCols(balanced_data, "loan_status")

minority_class <- filter(balanced_data, loan_status == "rejected")
majority_class <- filter(balanced_data, loan_status == "accepted")

num_to_add <- nrow(majority_class) - nrow(minority_class)
num_to_add <- num_to_add + 20

upsampled_minority <- slice_sample(minority_class, n = num_to_add, replace = FALSE)

balanced_data <- bind_rows(majority_class, minority_class, upsampled_minority)

table(balanced_data$loan_status)

plotCategoricalCols(balanced_data, "loan_status")
```

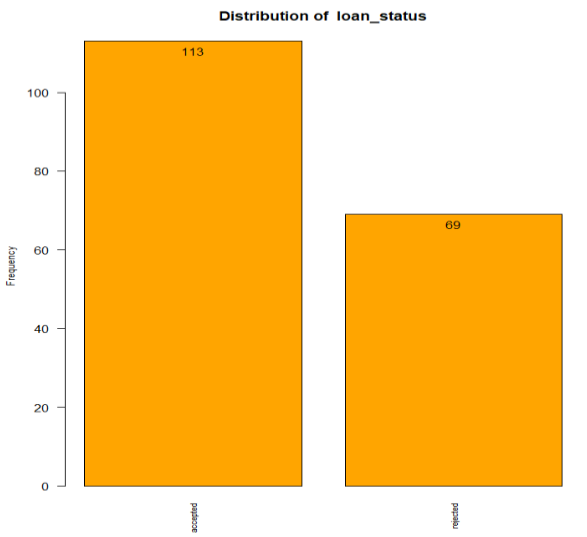
Description:

This technique has been used to make the dataset balanced. By doing upsampling, the minor

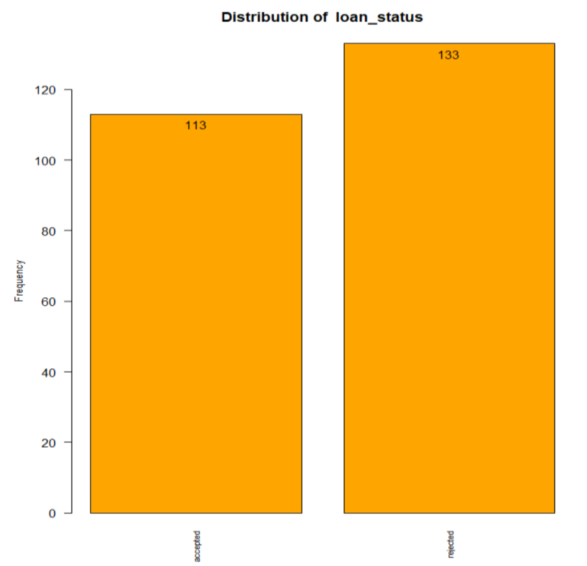
category increases its instance numbers.
upsampled_minority <- slice_sample(minority_class, n = num_to_add, replace = FALSE), by this line of code, the minor category has been increased.

Screenshot:

Before:



After:



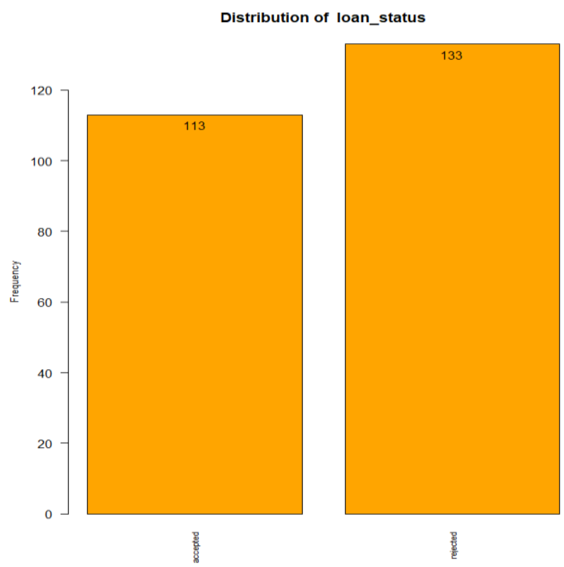
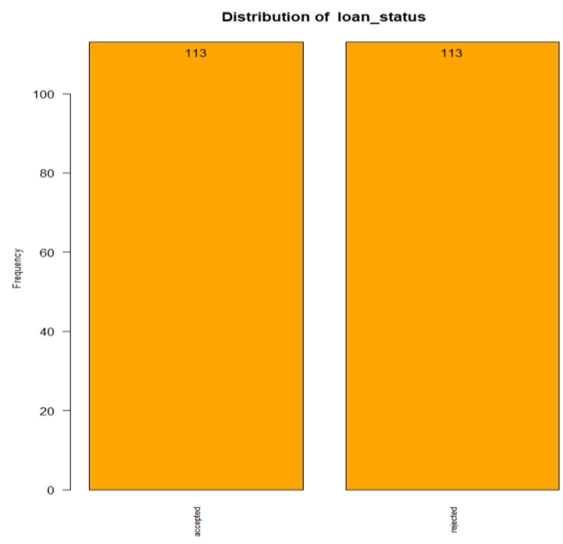
31. Applying Downsampling to balance the dataset

Code:

```
minority_class <- filter(balanced_data, loan_status == "accepted")
majority_class <- filter(balanced_data, loan_status == "rejected")

downsampled_majority_class <- majority_class %>% sample_n(nrow(minority_class))

balanced_data <- bind_rows(downsampled_majority_class, minority_class)

table(balanced_data$loan_status)
plotCategoricalCols(balanced_data, "loan_status")
```

Description:
This technique helps to reduce the size of instances of the major class of a column.
downsampled_majority_class <- majority_class %>% sample_n(nrow(minority_class)), this code helps to reduce the number of instances of the majority class in a column.

Screenshot:

Before:	After:
	

32. Summary after the preprocessed dataset

Code:
```
str(balanced_data)
summary(balanced_data)
```

Description:
This shows the summary of the dataset after preprocessing

Screenshot:

Before Preprocessing:

```
> str(data)
'data.frame':	201 obs. of  14 variables:
 $ person_age                    : num  21 21 25 23 24 NA 22 24 22 21 ...
 $ person_gender                 : chr  "female" "female" "female" "female" ...
 $ person_education              : chr  "Master" "High School" "High School" "Bachelor" ...
 $ person_income                 : num  71948 12282 12438 79753 66135 ...
 $ person_emp_exp                : num  0 0 3 0 1 0 1 5 3 0 ...
 $ person_home_ownership         : chr  "RENT" "OWN" "MORTGAGE" "RENT" ...
 $ loan_amnt                     : num  35000 1000 5500 35000 35000 2500 35000 35000 35000 1600 ...
 $ loan_intent                   : chr  "PERSONAL" "EDUCATION" "MEDICAL" "MEDICAL" ...
 $ loan_int_rate                 : num  16 11.1 12.9 15.2 14.3 ...
 $ loan_percent_income           : num  0.49 NA 0 0.44 0.53 0.19 0.37 0.37 0.35 0.13 ...
 $ cb_person_cred_hist_length    : num  3 2 3 2 4 2 3 4 2 3 ...
 $ credit_score                  : num  561 504 635 675 586 532 701 585 544 640 ...
 $ previous_loan_defaults_on_file: chr  "No" "Yes" "No" "No" ...
 $ loan_status                   : num  1 0 1 1 1 1 1 1 NA 1 ...
> summary(data)
   person_age     person_gender      person_education   person_income     person_emp_exp    person_home_ownership   loan_amnt
 Min.   : 21.00   Length:201         Length:201         Min.   :  12282   Min.   :  0.000   Length:201            Min.   : 1000
 1st Qu.: 22.00   Class :character   Class :character   1st Qu.:  60501   1st Qu.:  0.000   Class :character      1st Qu.:10000
 Median : 23.00   Mode  :character   Mode  :character   Median :  85284   Median :  1.000   Mode  :character      Median :25000
 Mean   : 27.39                                         Mean   : 149875   Mean   :  2.761                         Mean   :20553
 3rd Qu.: 25.00                                         3rd Qu.: 241060   3rd Qu.:  3.000                         3rd Qu.:28000
 Max.   :350.00                                         Max.   :3138998   Max.   :125.000                         Max.   :35000
 NA's   :4                                              NA's   :4
 loan_intent         loan_int_rate    loan_percent_income cb_person_cred_hist_length  credit_score    previous_loan_defaults_on_file
 Length:201         Min.   : 5.42   Min.   :0.0000      Min.   :2.00               Min.   :484.0   Length:201
 Class :character   1st Qu.:10.65   1st Qu.:0.0900      1st Qu.:2.00               1st Qu.:595.0   Class :character
 Mode  :character   Median :11.83   Median :0.2350      Median :3.00               Median :630.0   Mode  :character
                    Mean   :12.29   Mean   :0.2293      Mean   :2.99               Mean   :628.5
                    3rd Qu.:14.42   3rd Qu.:0.3425      3rd Qu.:4.00               3rd Qu.:665.0
                    Max.   :20.00   Max.   :0.5300      Max.   :4.00               Max.   :807.0
                                    NA's   :1
  loan_status
 Min.   :0.0000
 1st Qu.:0.0000
 Median :1.0000
 Mean   :0.6162
 3rd Qu.:1.0000
 Max.   :1.0000
 NA's   :3
>
```

After Preprocessing:

```
> str(balanced_data)
'data.frame':	238 obs. of  14 variables:
 $ person_age                    : num  22 24 26 24 21 22 25 22 25 21 ...
 $ person_gender                 : Factor w/ 2 levels "0","1": 2 1 2 2 1 2 1 2 1 2 ...
 $ person_education              : Factor w/ 5 levels "0","1","2","3",..: 4 3 3 3 3 1 3 4 3 2 ...
 $ person_income                 : Factor w/ 4 levels "Low","Lower Middle",..: 3 1 4 3 2 3 4 4 4 3 ...
 $ person_emp_exp                : num  1 0 5 3 0 1 0 2 0 0 ...
 $ person_home_ownership         : Factor w/ 4 levels "0","1","2","3": 1 2 1 1 1 1 1 1 1 1 ...
 $ loan_amnt                     : Factor w/ 3 levels "Small","Medium",..: 1 1 2 1 3 3 2 3 3 2 ...
 $ loan_intent                   : Factor w/ 6 levels "0","1","2","3",..: 2 1 4 3 4 4 5 1 3 2 ...
 $ loan_int_rate                 : num  0.1148 0.0729 0.0788 0.1269 0.0599 ...
 $ loan_percent_income           : num  0.05 0.11 0.06 0.05 0.19 0.09 0.06 0.09 0.08 0.07 ...
 $ cb_person_cred_hist_length    : num  3 3 4 2 4 2 2 2 4 4 ...
 $ credit_score                  : num  NA NA NA NA NA NA NA NA NA NA ...
 $ previous_loan_defaults_on_file: Factor w/ 2 levels "0","1": 1 2 2 1 2 2 2 2 2 2 ...
 $ loan_status                   : chr  "rejected" "rejected" "rejected" "rejected" ...
>
```

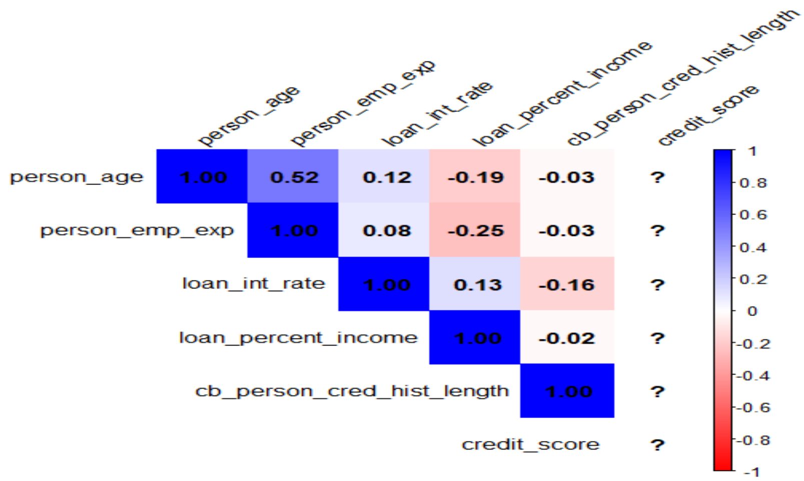
```
> summary(balanced_data)
   person_age    person_gender person_education      person_income person_emp_exp  person_home_ownership  loan_amnt    loan_intent
 Min.   :21.00   0: 92         0:25              Low         :117   Min.   :0.000   0:222                 Small : 68   0:36
 1st Qu.:22.00   1:146         1:66              Lower Middle: 24   1st Qu.:0.000   1: 10                 Medium: 35   1:67
 Median :23.00                 2:92              Upper Middle: 63   Median :1.000   2:  5                 Large :135   2:33
 Mean   :23.55                 3:54              High        : 34   Mean   :1.735   3:  1                              3:37
 3rd Qu.:25.00                 4: 1                                 3rd Qu.:3.000                                      4:29
 Max.   :26.00                                                      Max.   :8.000                                      5:36

 loan_int_rate    loan_percent_income cb_person_cred_hist_length  credit_score previous_loan_defaults_on_file loan_status
 Min.   :0.0542   Min.   :0.0000      Min.   :2.000              Min.   : NA   0:158                          Length:238
 1st Qu.:0.1063   1st Qu.:0.0800      1st Qu.:2.000              1st Qu.: NA   1: 80                          Class :character
 Median :0.1184   Median :0.1750      Median :3.000              Median : NA                                  Mode  :character
 Mean   :0.1230   Mean   :0.2026      Mean   :2.996              Mean   :NaN
 3rd Qu.:0.1433   3rd Qu.:0.3275      3rd Qu.:4.000              3rd Qu.: NA
 Max.   :0.2000   Max.   :0.5300      Max.   :4.000              Max.   : NA
                                                                 NA's   :238
> 
```

From the before and after summaries of the dataset, it can be seen that handling the missing values & outliers, the overall measures of central tendencies as well as the spread have decreased, which was the initial goal of data preprocessing. The target attribute was encoded, as well as some of the numeric columns with high standard deviation (person_income, loan_amnt) which were also encoded.

33. Correlation Matrix

numeric_data <- dplyr::select_if(balanced_data, is.numeric)
 	
cor_matrix <- cor(numeric_data, use = "pairwise.complete.obs")
 	
library(corrplot)
corrplot(cor_matrix,
        method = "color",
        type = "upper",
        addCoef.col = "black",
        tl.col = "black",
        tl.srt = 45,
        col = colorRampPalette(c("red", "white", "blue"))(200))

Description: This technique demonstrates **downsampling**, which is used to handle class imbalance in a dataset. When one class (the *majority class*) has far more instances than another (the *minority class*), models may become biased toward the majority.
Screenshot:



34. Export preprocessed dataset

Code:

```
write.xlsx(balanced_data, "data_preprocessed.xlsx")
```

Description:

This code exported the preprocessed dataset named “data_preprocessed” in .xlsx format.

Screenshot:

Before preprocess

person_age	person_gender	person_education	person_income	person_emp_exp	person_home_ownership	loan_amnt	loan_intent	loan_int_rate	loan_percent_	cb_person_cred_hist_length	credit_score	previous_l	loan_status
21	female	Master	71948	0	RENT	35000	PERSONAL	16.02	0.49	3	561	No	1
21	female	High School	12282	0	OWN	1000	EDUCATION	11.14	#VALUE!	2	504	Yes	0
25	female	High School	12438	3	MORTGAGE	5500	MEDICAL	12.87	0	3	635	No	1
23	female	Bachelor	79753	0	RENT	35000	MEDICAL	15.23	0.44	2	675	No	1
24	male	Master	66135	1	RENTT	35000	MEDICAL	14.27	0.53	4	586	No	1
	female	High School	12951	0	OWN	2500	VENTURE	7.14	0.19	2	532	No	1
22	female	Bachelor		1	RENT	35000	EDUCATION	12.42	0.37	3	701	No	1
24		High School	95550	5	RENT	35000	MEDICAL	11.11	0.37	4	585	No	1
22	female		100684	3	RENT	35000	PERSONAL	8.9	0.35	2	544	No	
21	female	High School	12739	0	OWN	1600	VENTURE	14.74	0.13	3	640	No	1
22	female	High School	102985	0	RENT	35000	VENTURE	10.37	0.34	4	621	No	1
21	female	Associate	13113	0	OWN	4500	HOMEIMPROVEMI	8.63	0.34	2	651	No	1
23	male	Bachelor	114860	3	RENT	35000	VENTURE	7.9	0.3	2	573	No	1
	male	Master	130713	0	RENT	35000	EDUCATION	18.39	0.27	4	708	No	1
23	female	Associate	3138998	0	RENT	35000	EDUCATION	7.9	0.25	4	583	No	
23	female			5	MORTGAGE	30000	DEBTCONSOLIDA	10.65	0.05	3	670	Yes	0
23		Bachelor	144943	0	RENT	35000	EDUCATION	7.9	0.24	4	663	No	0
23	female	High School	111369	0	RENT	35000	MEDICAL	20	0.31	4	694	No	
23	male	Bachelor	136628	0	RENT	35000	DEBTCONSOLIDA	18.25	0.26	4	709	No	1
24	female	Master	14283	1	MORTGAGE	1750	EDUCATION	10.99	0.12	2	679	No	1
25	male	Bachelor	195718	0	RENT	35000	VENTURE	7.49	0.18	4	684	Yes	0
25	male	High School	165792	4	RENT	34800	PERSONAL	16.77	0.21	2	662	No	0
22	female	Master	79255	0	RENT	34000	EDUCATION	17.58	0.43	4	691	No	1
24	female	Bachelor	13866	0	OWN	1500	PERSONAL	7.29	0.11	3	600	Yes	0

After preprocess:

person_age	person_genc	person_educ	person_inco	person_emp	person_hom	loan_amnt	loan_intent	loan_int_rat	loan_percen	cb_person_c	credit_score	previous_loa	loan_status
22	1	1	Upper Middl	2	0	Small	1	0.0788	0.05	2	0.60683761	1	rejected
23	0	1	High	3	0	Large	5	0.1385	0.07	3	0.35470085	1	rejected
26	1	0	Upper Middl	5	0	Medium	4	0.1149	0.07	4	0.79487179	1	rejected
23	1	2	Upper Middl	0	0	Large	2	0.1479	0.14	3	0.57264957	1	rejected
22	0	0	Upper Middl	2	0	Small	3	0.1038	0.05	2	0.79059829	1	rejected
26	1	2	High	5	0	Medium	3	0.0788	0.06	4	0.85470085	1	rejected
22	0	2	High	0	0	Small	3	0.1158	0.04	2	0.5042735	1	rejected
24	1	2	Upper Middl	3	0	Small	2	0.1269	0.05	2	0.64102564	0	rejected
22	1	1	High	0	0	Large	1	0.1941	0.07	2	0.65384615	1	rejected
25	1	1	Lower Middl	4	0	Large	0	0.1677	0.21	2	0.76068376	0	rejected
26	1	1	High	3	0	Large	2	0.1417	0.09	2	0.67521368	1	rejected
26	1	2	High	5	0	Large	1	0.1533	0.07	3	0.9017094	1	rejected
21	1	2	Upper Middl	0	0	Large	5	0.1399	0.1	2	0.91880342	0	rejected
26	1	2	High	5	0	Large	1	0.1533	0.07	3	0.9017094	1	rejected
24	1	1	Upper Middl	0	0	Small	2	0.1101	0.02	4	0.1025641	1	rejected
25	0	2	Upper Middl	3	0	Large	4	0.1991	0.11	2	0.65384615	1	rejected
22	1	1	Upper Middl	3	0	Medium	1	0.1479	0.06	3	0.60683761	1	rejected
23	1	2	High	1	0	Large	0	0.1101	0.1	4	0.53418803	0	rejected
25	0	2	Lower Middl	1	0	Large	5	0.1269	0.23	3	0.58974359	0	rejected
25	1	3	Upper Middl	2	0	Medium	0	0.1435	0.1	3	0.4017094	0	rejected
24	0	1	Upper Middl	4	0	Small	1	0.0849	0.04	4	0.51282051	1	rejected
22	0	3	Upper Middl	3	0	Medium	0	0.1183	0.09	4	0.44444444	1	rejected
23	1	3	Upper Middl	2	0	Medium	5	0.089	0.08	3	0.93589744	0	rejected
25	0	2	High	0	0	Medium	4	0.1442	0.06	2	0.75641026	1	rejected

Conclusion:

Through this project, the dataset was carefully examined and preprocessed to handle issues such as missing values, invalid categorical entries, skewed distributions, and outliers. Multiple strategies (including top-down, bottom-up, mean, median, and mode imputation) were tested for missing data, while outlier detection techniques such as Z-score and IQR were applied. Class imbalance was addressed using both upsampling and downsampling to create a more balanced dataset.

The main findings indicate that variables such as credit score, income, loan amount, and loan percent income play a significant role in influencing loan approval decisions. Correlation analysis further revealed important relationships among numeric features that could guide predictive modeling.

However, some limitations were encountered:

- The dataset size (201 instances) is relatively small, which may restrict generalizability.
- Imputation methods for missing values may introduce bias, especially when replacing with median or mode.
- Even after balancing, some degree of information loss occurs due to downsampling the majority class.

Overall, the preprocessing steps improved the dataset's quality and readiness for further machine learning applications, ensuring more reliable insights in loan approval prediction tasks.