American International University-Bangladesh (AIUB)

Faculty of Science & Technology (FST)

Department of Computer Science

Introduction to Data Science

Mid-Term Project Report

Summer 2024-2025

Section:A

Group:4

SL #	Student Name	Student ID	Contribution (%)
1.	Md Saniul Islam Sony	22-46400-1	100%
2.	Tanvir Ahmed Tuhin	22-46475-1	100%
3.	MD.Ashraful Islam	22-46479-1	100%

Dataset Description

A credit-risk dataset where each row is one borrower and one loan application. It combines basic demographics, finances, and loan details, plus two binary outcome fields.

All columns:

person_age — borrower's age (years)

person_gender — gender

person_education — highest education level

person_income — annual income

person_emp_exp — years of employment experience

person_home_ownership — housing status (e.g., RENT, OWN, MORTGAGE)

loan_amnt — requested loan amount

loan_intent — purpose of the loan (e.g., PERSONAL, MEDICAL)

loan_int_rate — interest rate (%)

loan_percent_income — loan payment as a share of income (0–1)

cb_person_cred_hist_length — credit history length (years)

credit_score — credit score (approx. 300–850)

previous_loan_defaults_on_file — whether the borrower ever defaulted before (0/1)

loan_status — outcome of this loan application or performance (0/1)

1.Handle missing value:

Description:

- **Drop missing rows (complete cases).**

We first list rows that contain any NA to inspect what's missing, then create a complete-case view by removing those rows.

- **Replace with mean (numeric).**

For a numeric column that is roughly symmetric (here, person_income), we fill NA values with the column mean calculated from the available (non-missing) values.

- **Replace with median (numeric).**

For a numeric column that may be skewed or affected by outliers (here, person_age), we fill NA values with the median, which is more robust than the mean.

- **Replace with mode (categorical).**

For a categorical/label column (here, loan_status), we fill NA values with the mode (the most frequent category). The mlv(..., method="mfv") function returns that most frequent value, which we use to impute the missing entries.

<u>Code:</u>

- Drop missing value:

```
missing_rows <- data[!complete.cases(data), ]
head(missing_rows)
```

Output:

	person_age	person_gender	person_education	person_income	person_emp_exp	person_home_ownership	loan_amnt	loan_intent	loan_int_rate	loan_percent_income	cb_person_cred_hist_length	credit_score	previous_loan_defaults_on_file	loan_status
1	21	female	Master	71948	0	RENT	35000	PERSONAL	16.02	0.49	3	561	No	1
2	21	female	High School	12282	0	OWN	1000	EDUCATION	11.14	0.49	2	504	Yes	0
3	25	female	High School	12438	3	MORTGAGE	5500	MEDICAL	12.87	0.49	3	635	No	1
4	23	female	Bachelor	79753	0	RENT	35000	MEDICAL	15.23	0.44	2	675	No	1
5	24	male	Master	66135	1	RENTT	35000	MEDICAL	14.27	0.53	4	586	No	1
10	21	female	High School	12739	0	OWN	1600	VENTURE	14.74	0.13	3	640	No	1

- Replace with mean

```
df_mean <-data
df_mean$person_income[is.na(df_mean$person_income)] <-
mean(df_mean$person_income,na.rm=TRUE)
head(df_mean)
```

Output:

	person_age	person_gender	person_education	person_income	person_emp_exp	person_home_ownership	loan_amnt	loan_intent	loan_int_rate	loan_percent_income	cb_person_cred_hist_length	credit_score	previous_loan_defaults_on_file	loan_status
1	21	female	Master	71948	0	RENT	35000	PERSONAL	16.02	0.49	3	561	No	1
2	21	female	High School	12282	0	OWN	1000	EDUCATION	11.14	0.49	2	504	Yes	0
3	25	female	High School	12438	3	MORTGAGE	5500	MEDICAL	12.87	0.49	3	635	No	1
4	23	female	Bachelor	79753	0	RENT	35000	MEDICAL	15.23	0.44	2	675	No	1
5	24	male	Master	66135	1	RENTT	35000	MEDICAL	14.27	0.53	4	586	No	1
6	NA	female	High School	12951	0	OWN	2500	VENTURE	7.14	0.19	2	532	No	1

- Replace with median

```
df_median <- df_mean#replace by median
df_median$person_age[is.na(df_median$person_age)] <-
median(df_median$person_age,na.rm = TRUE);
head(df_median)
```

Output:

	person_age	person_gender	person_education	person_income	person_emp_exp	person_home_ownership	loan_amnt	loan_intent	loan_int_rate	loan_percent_income	cb_person_cred_hist_length	credit_score	previous_loan_defaults_on_file	loan_status
1	21	female	Master	71948	0	RENT	35000	PERSONAL	16.02	0.49	3	561	No	1
2	21	female	High School	12282	0	OWN	1000	EDUCATION	11.14	0.49	2	504	Yes	0
3	25	female	High School	12438	3	MORTGAGE	5500	MEDICAL	12.87	0.49	3	635	No	1
4	23	female	Bachelor	79753	0	RENT	35000	MEDICAL	15.23	0.44	2	675	No	1
5	24	male	Master	66135	1	RENTT	35000	MEDICAL	14.27	0.53	4	586	No	1
6	23	female	High School	12951	0	OWN	2500	VENTURE	7.14	0.19	2	532	No	1

- Replace with mode

```
mode_val <- mlv(df_mode$loan_status,method = "mfv",na.rm = TRUE);
df_mode$loan_status[is.na(df_mode$loan_status)] <- mode_val
head(as.data.frame(df_mode))
```

Output:

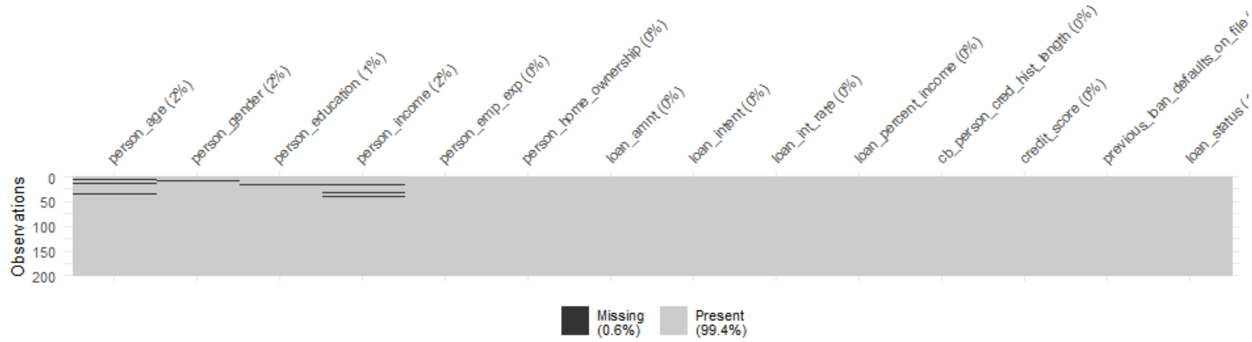
	person_age	person_gender	person_education	person_income	person_emp_exp	person_home_ownership	loan_amnt	loan_intent	loan_int_rate	loan_percent_income	cb_person_cred_hist_length	credit_score	previous_loan_defaults_on_file	loan_status
1	21	female	Master	71948	0	RENT	35000	PERSONAL	16.02	0.49	3	561	No	1
2	21	female	High School	12282	0	OWN	1000	EDUCATION	11.14	0.49	2	504	Yes	0
3	25	female	High School	12438	3	MORTGAGE	5500	MEDICAL	12.87	0.49	3	635	No	1
4	23	female	Bachelor	79753	0	RENT	35000	MEDICAL	15.23	0.44	2	675	No	1
5	24	male	Master	66135	1	RENTT	35000	MEDICAL	14.27	0.53	4	586	No	1
6	23	female	High School	12951	0	OWN	2500	VENTURE	7.14	0.19	2	532	No	1

2. see missing values on a graph:

Description:

In this task we will use the vis_miss() function from the *naniar* package to visualize missing values in the dataset with a graph. The graph will display variables as columns and records as rows, where shaded blocks indicate missing entries. This makes it easy to see which columns contain missing data and how much is missing, giving us a clear picture of the dataset's quality.

```
vis_miss(data)
```



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

Description:

In this task we will convert attributes from numeric to categorical or from categorical to numeric depending on how we need them for analysis. For example, the column loan_status is stored as numbers (0/1), but for easier interpretation we convert it into labels "Yes" and "No". On the other hand, the column previous_loan_defaults_on_file may be stored as text ("Yes"/"No"), so we convert it back into numeric values (1/0) for modeling.

We solve this by using simple ifelse() statements:

```
df_mode$loan_status <-ifelse(df_mode$loan_status == 1
,"Yes","No");
head(df_mode)
```

Output:

```
  person_age person_gender person_education person_income person_emp_exp person_home_ownership loan_amnt loan_intent loan_int_rate loan_percent_income cb_person_cred_hist_length credit_score previous_loan_defaults_on_file loan_status
1         21        female           Master         71948              0                  RENT     35000    PERSONAL         16.02                0.49                          3          561                              0          No
2         21        female      High School         12282              0                   OWN      1000   EDUCATION         11.14                0.49                          2          504                              1          No
3         25        female      High School         12438              3              MORTGAGE      5500     MEDICAL         12.87                0.49                          3          635                              0          No
4         23        female         Bachelor         79753              0                  RENT     35000     MEDICAL         15.23                0.44                          2          675                              0          No
5         24          male           Master         66135              1                 RENTT     35000     MEDICAL         14.27                0.53                          4          586                              0          No
6         23        female      High School         12951              0                   OWN      2500     VENTURE          7.14                0.19                          2          532                              0          No
>
```

```
df_mode$previous_loan_defaults_on_file <-
ifelse(df_mode$previous_loan_defaults_on_file == "Yes",
1,0);
head(df_mode)
```

Output:

```
  person_age person_gender person_education person_income person_emp_exp person_home_ownership loan_amnt loan_intent loan_int_rate loan_percent_income cb_person_cred_hist_length credit_score previous_loan_defaults_on_file loan_status
1         21        female           Master         71948              0                  RENT     35000    PERSONAL         16.02                0.49                          3          561                              0          No
2         21        female      High School         12282              0                   OWN      1000   EDUCATION         11.14                0.49                          2          504                              0          No
3         25        female      High School         12438              3              MORTGAGE      5500     MEDICAL         12.87                0.49                          3          635                              0          No
4         23        female         Bachelor         79753              0                  RENT     35000     MEDICAL         15.23                0.44                          2          675                              0          No
5         24          male           Master         66135              1                 RENTT     35000     MEDICAL         14.27                0.53                          4          586                              0          No
6         23        female      High School         12951              0                   OWN      2500     VENTURE          7.14                0.19                          2          532                              0          No
>
```

4. Detect outliers in the data set and use the appropriate approach to handle those values:

Description:

In this task we detect and handle extreme values in numeric columns using the Interquartile Range (IQR) method. First, we calculate Q1 (25th percentile), Q3 (75th percentile), and the IQR (Q3 − Q1). Any value smaller than Q1 − 1.5 × IQR is treated as a lower outlier, and any value greater than Q3 + 1.5 × IQR is treated as an upper outlier.

The get_outliers() function lists these values so we can see which records fall outside the normal range. Then the fix_outliers() function corrects them:

- If a value is less than the lower bound, it is replaced by the lower bound.
- If a value is greater than the upper bound, it is replaced by the upper bound.

This way, the dataset does not lose any rows, but all values are adjusted to stay within reasonable limits, reducing the influence of extreme numbers on further analysis.

```
get_outliers <- function(data, colname) {
  if (!colname %in% names(data)) stop("Column not found.")
  if (!is.numeric(data[[colname]])) stop("Column must be
numeric.")

  Q1 <- quantile(data[[colname]], 0.25, na.rm = TRUE)
  Q3 <- quantile(data[[colname]], 0.75, na.rm = TRUE)
  IQR <- Q3 - Q1

  lower <- Q1 - 1.5 * IQR
  upper <- Q3 + 1.5 * IQR

  # return only the outlier values from dataset
  outliers <- data[[colname]][data[[colname]] < lower |
data[[colname]] > upper]

  return(outliers)
}

outlier_detect<-get_outliers(df, "person_age")
head(outlier_detect)
```

Output:

```
> outlier_detect<-get_outliers(df, "person_age")
> head(outlier_detect)
[1] 230 -22 -25 350 144 144
```

```
fix_outliers <- function(data, colname) {
  if (!colname %in% names(data)) stop("Column not found.")
  if (!is.numeric(data[[colname]])) stop("Column must be
numeric.")

  Q1 <- quantile(data[[colname]], 0.25, na.rm = TRUE)
  Q3 <- quantile(data[[colname]], 0.75, na.rm = TRUE)
  IQR <- Q3 - Q1

  lower <- Q1 - 1.5 * IQR
  upper <- Q3 + 1.5 * IQR
```

```
  is_outlier <- data[[colname]] < lower | data[[colname]] >
upper
  n_out <- sum(is_outlier, na.rm = TRUE)

  if (n_out > 0) {
    data[[colname]] <- pmin(pmax(data[[colname]], lower),
upper)
    message("Fixed ", n_out, " outliers in ", colname,
            " (capped to [", round(lower, 2), ", ",
round(upper, 2), "])")
  } else {
    message("No outliers to fix in ", colname)
  }

  return(data)
}
df_clean <- fix_outliers(df, "person_age")
head(as.data.frame(df_clean))
```

Output:

	person_age	person_gender	person_education	person_income	person_emp_exp	person_home_ownership	loan_amnt	loan_intent	loan_int_rate	loan_percent_income	cb_person_cred_hist_length	credit_score	previous_loan_defaults_on_file	loan_status
1	21	female	Master	71948	0	RENT	35000	PERSONAL	16.02	0.49	3	561	0	Yes
2	21	female	High School	12282	0	OWN	1000	EDUCATION	11.14	0.49	2	504	1	No
3	25	female	High School	12438	3	MORTGAGE	5500	MEDICAL	12.87	0.49	3	635	0	Yes
4	23	female	Bachelor	79753	0	RENT	35000	MEDICAL	15.23	0.44	2	675	0	Yes
5	24	male	Master	66135	1	RENTT	35000	MEDICAL	14.27	0.53	4	586	0	Yes
6	23	female	High School	12951	0	OWN	2500	VENTURE	7.14	0.19	2	532	0	Yes

Task 5: Normalization method for any continuous attribute:

Description: In this task we apply normalization to a continuous column so that all its values are scaled into the range 0 to 1. We use the min–max normalization formula:

$$X_{norm} = \frac{x - \min(x)}{\max(x) - \min(x)}$$

Here we normalize the column loan_amnt. The smallest loan amount becomes 0, the largest loan amount becomes 1, and all other values are scaled proportionally between 0 and 1.

This step ensures that attributes measured on different scales can be compared fairly

```
df_nor <- df
normalize <- function(x) {
  return( (x - min(x)) / (max(x) - min(x)) )
}
df_nor$loan_amnt <- normalize(df_nor$loan_amnt)
head(df_nor)
```

Output:

```
> head(df_nor)
  person_age person_gender person_education person_income person_emp_exp person_home_ownership  loan_amnt loan_intent loan_int_rate loan_percent_income cb_person_cred_hist_length credit_score previous_loan_defaults_on_file loan_status
1         21        female           Master         71948              0                  RENT 1.00000000    PERSONAL         16.02                0.49                          3          561                              0           1
2         21        female      High School         12282              0                   OWN 0.00000000   EDUCATION         11.14                0.49                          2          504                              1           0
3         25        female      High School         12438              3              MORTGAGE 0.13235294     MEDICAL         12.87                0.49                          3          635                              0           1
4         23        female         Bachelor         79753              0                  RENT 1.00000000     MEDICAL         15.23                0.44                          2          675                              0           1
5         24          male           Master         66135              1                 RENTT 1.00000000     MEDICAL         14.27                0.53                          4          586                              0           1
6         23        female      High School         12951              0                   OWN 0.04411765     VENTURE          7.14                0.19                          2          532                              0           1
> |
```

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

Description:

In this task we look for duplicate rows in the dataset and remove them to make the data cleaner and less congested. We use the distinct() function from the dplyr package. This function checks for duplicates based on a chosen column (here person_education) and keeps only the first occurrence, while the option .keep_all = TRUE ensures that all other columns in that row are also preserved.

```
df_unique <- df
df_unique <- distinct(df, person_education  ,.keep_all = TRUE)
head(as.data.frame(df_unique))
```

Output:

```
> head(as.data.frame(df_unique))
  person_age person_gender person_education person_income person_emp_exp person_home_ownership loan_amnt       loan_intent loan_int_rate loan_percent_income cb_person_cred_hist_length credit_score previous_loan_defaults_on_file loan_status
1         21        female           Master         71948              0                  RENT     35000          PERSONAL         16.02                0.49                          3          561                              0           1
2         21        female      High School         12282              0                   OWN      1000         EDUCATION         11.14                0.49                          2          504                              1           0
3         23        female         Bachelor         79753              0                  RENT     35000           MEDICAL         15.23                0.44                          2          675                              0           1
4         21        female        Associate         13113              0                   OWN      4500   HOMEIMPROVEMENT          8.63                0.34                          2          651                              0           1
5         26        female        Doctorate         56325              2                  RENT     25000 DEBTCONSOLIDATION         11.86                0.44                          4          690                              0           1
> 
```

Description:

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

Description:

There are many ways to filter data in R, but here we are using the filter() function from the dplyr package. This allows us to select rows that match certain conditions. For example:

- Filtering numerical values such as loan_amnt > 10000 to get only loans greater than 10,000.
- Filtering based on both missingness and value, such as keeping rows where person_income is not missing and greater than 20,000.
- Filtering categorical values, such as selecting rows where person_education is "Bachelor" or "Master".

```
filtered_data1 <- filter(df_clean, loan_amnt > 10000)
head(as.data.frame(filtered_data1))
filtered_data2 <- filter(df_clean, !is.na(person_income) &
person_income > 20000)
head(as.data.frame(filtered_data2))
filtered_data3 <- filter(df_clean, person_education %in%
c("Bachelor", "Master"))
head(as.data.frame(filtered_data3))
```

Output:

```
> head(as.data.frame(filtered_data1))
  person_age person_gender person_education person_income person_emp_exp person_home_ownership loan_amnt loan_intent loan_int_rate loan_percent_income cb_person_cred_hist_length credit_score previous_loan_defaults_on_file loan_status
1         21        female           Master       71948.0              0                  RENT     35000    PERSONAL         16.02                0.49                          3          561                              0         Yes
2         23        female         Bachelor       79753.0              0                  RENT     35000     MEDICAL         15.23                0.44                          2          675                              0         Yes
3         24          male           Master       66135.0              1                 RENTT     35000     MEDICAL         14.27                0.53                          4          586                              0         Yes
4         22        female         Bachelor      149874.8              1                  RENT     35000   EDUCATION         12.42                0.37                          3          701                              0         Yes
5         24          male      High School       95550.0              5                  RENT     35000     MEDICAL         11.11                0.37                          4          585                              0         Yes
6         22        female         Bachelor      100684.0              3                  RENT     35000    PERSONAL          8.90                0.35                          2          544                              0         Yes
> head(as.data.frame(filtered_data2))
  person_age person_gender person_education person_income person_emp_exp person_home_ownership loan_amnt loan_intent loan_int_rate loan_percent_income cb_person_cred_hist_length credit_score previous_loan_defaults_on_file loan_status
1         21        female           Master       71948.0              0                  RENT     35000    PERSONAL         16.02                0.49                          3          561                              0         Yes
2         23        female         Bachelor       79753.0              0                  RENT     35000     MEDICAL         15.23                0.44                          2          675                              0         Yes
3         24          male           Master       66135.0              1                 RENTT     35000     MEDICAL         14.27                0.53                          4          586                              0         Yes
4         22        female         Bachelor      149874.8              1                  RENT     35000   EDUCATION         12.42                0.37                          3          701                              0         Yes
5         24          male      High School       95550.0              5                  RENT     35000     MEDICAL         11.11                0.37                          4          585                              0         Yes
6         22        female         Bachelor      100684.0              3                  RENT     35000    PERSONAL          8.90                0.35                          2          544                              0         Yes
> head(as.data.frame(filtered_data3))
  person_age person_gender person_education person_income person_emp_exp person_home_ownership loan_amnt loan_intent loan_int_rate loan_percent_income cb_person_cred_hist_length credit_score previous_loan_defaults_on_file loan_status
1         21        female           Master       71948.0              0                  RENT     35000    PERSONAL         16.02                0.49                          3          561                              0         Yes
2         23        female         Bachelor       79753.0              0                  RENT     35000     MEDICAL         15.23                0.44                          2          675                              0         Yes
3         24          male           Master       66135.0              1                 RENTT     35000     MEDICAL         14.27                0.53                          4          586                              0         Yes
4         22        female         Bachelor      149874.8              1                  RENT     35000   EDUCATION         12.42                0.37                          3          701                              0         Yes
5         22        female         Bachelor      100684.0              3                  RENT     35000    PERSONAL          8.90                0.35                          2          544                              0         Yes
6         23          male         Bachelor      114860.0              3                  RENT     35000     VENTURE          7.90                0.30                          2          573                              0         Yes
>
```

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

Description:

There are many ways to handle invalid/missing values; here we use two simple approaches:

- ❖ Remove rows with any missing values using na.omit() to get a clean subset and check how many rows remain.
- ❖ Impute (fill) missing values in the full dataset:
    - For numeric columns, replace NA with the mean of that column.
    - For categorical columns, replace NA with the mode (most frequent category).

After imputation, we verify the result by printing the count of remaining NAs per column group and previewing the cleaned data.

```
loans5<-Loan_Datas
loans5 <- na.omit(loans5)
print(nrow(loans5))
print(loans5)

loan5 <- Loan_Datas
loan3_clean <- na.omit(loan5)
print(nrow(loan3_clean))
loan3_clean
num_cols <- sapply(loan5, is.numeric)
```

```
cat_cols <- sapply(loan5, is.character)
loan5[num_cols] <- lapply(loan5[num_cols], function(x) {
  x[is.na(x)] <- mean(x, na.rm = TRUE)
  return(x)
})
loan5[cat_cols] <- lapply(loan5[cat_cols], function(x) {
  mode_val <- names(sort(table(x), decreasing = TRUE))[1]
  x[is.na(x)] <- mode_val
  return(x)
})
print(colSums(is.na(loan5[num_cols])))
print(colSums(is.na(loan5[cat_cols])))
print(loan5)
```

Output:

```
> head(as.data.frame(loans5))
  person_age person_gender person_education person_income person_emp_exp person_home_ownership loan_amnt loan_intent loan_int_rate loan_percent_income cb_person_cred_hist_length credit_score previous_loan_defaults_on_file loan_status
1         21        female           Master         71948              0                  RENT     35000    PERSONAL         16.02                0.49                          3          561                             No           1
2         21        female      High School         12282              0                   OWN      1000   EDUCATION         11.14                0.49                          2          504                            Yes           0
3         25        female      High School         12438              3              MORTGAGE      5500     MEDICAL         12.87                0.49                          3          635                             No           1
4         23        female         Bachelor         79753              0                  RENT     35000     MEDICAL         15.23                0.44                          2          675                             No           1
5         24          male           Master         66135              1                 RENTT     35000     MEDICAL         14.27                0.53                          4          586                             No           1
6         21        female      High School         12739              0                   OWN      1600     VENTURE         14.74                0.13                          3          640                             No           1
> head(as.data.frame(loan3_clean))
  person_age person_gender person_education person_income person_emp_exp person_home_ownership loan_amnt loan_intent loan_int_rate loan_percent_income cb_person_cred_hist_length credit_score previous_loan_defaults_on_file loan_status
1         21        female           Master         71948              0                  RENT     35000    PERSONAL         16.02                0.49                          3          561                             No           1
2         21        female      High School         12282              0                   OWN      1000   EDUCATION         11.14                0.49                          2          504                            Yes           0
3         25        female      High School         12438              3              MORTGAGE      5500     MEDICAL         12.87                0.49                          3          635                             No           1
4         23        female         Bachelor         79753              0                  RENT     35000     MEDICAL         15.23                0.44                          2          675                             No           1
5         24          male           Master         66135              1                 RENTT     35000     MEDICAL         14.27                0.53                          4          586                             No           1
6         21        female      High School         12739              0                   OWN      1600     VENTURE         14.74                0.13                          3          640                             No           1
> head(as.data.frame(loan5))
  person_age person_gender person_education person_income person_emp_exp person_home_ownership loan_amnt loan_intent loan_int_rate loan_percent_income cb_person_cred_hist_length credit_score previous_loan_defaults_on_file loan_status
1   21.00000        female           Master         71948              0                  RENT     35000    PERSONAL         16.02                0.49                          3          561                             No           1
2   21.00000        female      High School         12282              0                   OWN      1000   EDUCATION         11.14                0.49                          2          504                            Yes           0
3   25.00000        female      High School         12438              3              MORTGAGE      5500     MEDICAL         12.87                0.49                          3          635                             No           1
4   23.00000        female         Bachelor         79753              0                  RENT     35000     MEDICAL         15.23                0.44                          2          675                             No           1
5   24.00000          male           Master         66135              1                 RENTT     35000     MEDICAL         14.27                0.53                          4          586                             No           1
6   26.91371        female      High School         12951              0                   OWN      2500     VENTURE          7.14                0.19                          2          532                             No           1
> 
```

Task 9. We can convert the imbalanced data set into the balanced data set

Description:

In our dataset, the target variable was highly imbalanced, so we applied three resampling techniques to balance it: undersampling, oversampling, and SMOTE.

- **Undersampling**: Here, we reduced the majority class to match the minority class. In the code, N_under was calculated as twice the size of the minority class, and the ovun.sample() function with method = "under" was used. This produced a balanced dataset but at the cost of discarding many majority samples, which reduced the overall dataset size.
- **Oversampling**: In this method, we increased the minority class to match the majority class. We set N_over as twice the size of the majority class and used ovun.sample() with

method = "over". This balanced the dataset by duplicating minority class samples, keeping all majority data but introducing the risk of overfitting due to repeated rows.

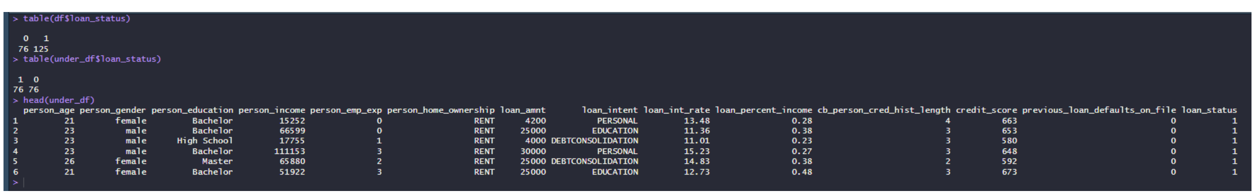
- **SMOTE (Synthetic Minority Oversampling Technique)**: Unlike oversampling, SMOTE generates synthetic minority samples instead of duplicating them. In the code, categorical variables were converted into factors, incomplete rows were removed, and the ROSE() function was applied. This enriched the dataset with synthetic examples, helping the model generalize better while maintaining class balance.

Overall, undersampling removes data, oversampling duplicates data, and SMOTE creates synthetic data. These methods make the dataset more balanced, ensuring fairer and more accurate model training.

- **Undersampling:**

```
library(ROSE)
df$loan_status <- factor(df$loan_status, levels = c(0,1))
table(df$loan_status)
N_under <- 2 * min(table(df$loan_status))
set.seed(199)
under_df <- ovun.sample(loan_status ~ ., data = df,method = "under",
N = N_under, seed = 199)$data
table(under_df$loan_status)
head(under_df)
```

Output:



```
> table(df$loan_status)

  0   1
 76 125
> table(under_df$loan_status)

 1  0
76 76
> head(under_df)
  person_age person_gender person_education person_income person_emp_exp person_home_ownership loan_amnt       loan_intent loan_int_rate loan_percent_income cb_person_cred_hist_length credit_score previous_loan_defaults_on_file loan_status
1         21        female         Bachelor         15252              0                  RENT      4200          PERSONAL         13.48                0.28                          4          663                              0           1
2         23          male         Bachelor         66599              0                  RENT     25000         EDUCATION         11.36                0.38                          3          653                              0           1
3         23          male      High School         17755              1                  RENT      4000 DEBTCONSOLIDATION         11.01                0.23                          3          580                              0           1
4         23          male         Bachelor        111153              3                  RENT     30000          PERSONAL         15.23                0.27                          3          648                              0           1
5         26        female           Master         65880              2                  RENT     25000 DEBTCONSOLIDATION         14.83                0.38                          2          592                              0           1
6         21        female         Bachelor         51922              3                  RENT     25000         EDUCATION         12.73                0.48                          3          673                              0           1
> 
```

- **Oversampling:**

```
library(ROSE)

df$loan_status <- factor(df$loan_status, levels = c(0,1))
table(df$loan_status)
N_over <- 2 * max(table(df$loan_status))
set.seed(199)
```

```
over_df <- ovun.sample(loan_status ~ ., data = df,
                              method = "over", N = N_over, seed = 199)$data
table(over_df$loan_status)
head(over_df)
```

Output:

```
> table(df$loan_status)

  0   1
 76 125
> table(over_df$loan_status)

  1   0
125 125
> head(over_df)
  person_age person_gender person_education person_income person_emp_exp person_home_ownership loan_amnt loan_intent loan_int_rate loan_percent_income cb_person_cred_hist_length credit_score previous_loan_defaults_on_file loan_status
1         21        female           Master       71948.0              0                  RENT     35000    PERSONAL         16.02                0.49                          3          561                              0           1
2         25        female      High School       12438.0              3              MORTGAGE      5500     MEDICAL         12.87                0.49                          3          635                              0           1
3         23        female         Bachelor       79753.0              0                  RENT     35000     MEDICAL         15.23                0.44                          2          675                              0           1
4         24          male           Master       66135.0              1                 RENTT     35000     MEDICAL         14.27                0.53                          4          586                              0           1
5         23        female      High School       12951.0              0                   OWN      2500     VENTURE          7.14                0.19                          2          532                              0           1
6         22        female         Bachelor      149874.8              1                  RENT     35000   EDUCATION         12.42                0.37                          3          701                              0           1
>
```

- **Smote**

```
library(ROSE)
set.seed(199)
df_smote$previous_loan_defaults_on_file <-
factor(df_smote$previous_loan_defaults_on_file, levels = c(0,1))
df_smote[sapply(df_smote, is.character)] <-
lapply(df_smote[sapply(df_smote, is.character)], factor)
df_smote <- df_smote[complete.cases(df_smote), ]
table(df_smote$previous_loan_defaults_on_file)
rose_df <- ROSE(previous_loan_defaults_on_file ~ ., data = df_smote,
N = 2000, p = 0.5)$data
table(rose_df$previous_loan_defaults_on_file)
head(rose_df)
```

Output:

```
> table(df_smote$previous_loan_defaults_on_file)

  0   1
150  51
> table(rose_df$previous_loan_defaults_on_file)

   0    1
1004  996
> head(rose_df)
  person_age person_gender person_education person_income person_emp_exp person_home_ownership loan_amnt       loan_intent loan_int_rate loan_percent_income cb_person_cred_hist_length credit_score previous_loan_defaults_on_file loan_status
1   21.22442        female         Bachelor     26171.050      4.1156402                  RENT  32192.34         EDUCATION      12.75038           0.3615402                   3.438469     700.3967                              0         Yes
2   29.10654        female      High School    297281.332     10.1892669                  RENT  24690.35   HOMEIMPROVEMENT      13.98243           0.4329616                   1.981410     598.3117                              0         Yes
3   21.66972        female           Master    334125.360      7.1406131                  RENT  29846.81          PERSONAL      16.06708           0.4950736                   3.925310     567.3151                              0         Yes
4   25.49819        female         Bachelor    -70203.018    -13.4600624                  RENT  20956.51 DEBTCONSOLIDATION      12.79445           0.1623564                   3.718361     610.1541                              0          No
5   25.10655          male         Bachelor     92247.763      2.4591532                  RENT  15756.08         EDUCATION      13.88743           0.2648101                   2.709815     573.3217                              0         Yes
6   26.28129          male         Bachelor     -4538.721     -0.8790874                  RENT  22708.11         EDUCATION      16.05170           0.2265051                   2.175995     625.9735                              0         Yes
>
```

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

Description:

In this task we divide the dataset into two parts: 70% for training and 30% for testing. The function initial_split() from the rsample package is used to create the split. From this split object, the training() function extracts the training set, and the testing() function extracts the testing set.

The training set is used to fit and build the model, while the testing set is kept aside to check how well the model performs on new, unseen data. Finally, the dim() function shows the number of rows and columns in each set to confirm that the split was done correctly (about 70% of rows in training and 30% in testing).

```
set.seed(123)
split <- initial_split(df, prop = 0.7)
train_data <- training(split)
test_data  <- testing(split)
dim(train_data)
dim(test_data)
```

Output:

```
> dim(train_data)
[1] 140  14
> dim(test_data)
[1] 61 14
> |
```

Description:

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

variables between two target classes (loan_status = 1 and loan_status = 0)

➢ person_age

➢ Person Income:

Description:

In this task we calculate summary statistics of two numerical variables — person_age and person_income — separately for the two target classes of loan_status (0 and 1).

We use the aggregate() function with a custom summary that computes the mean, median, standard deviation, minimum, and maximum for each variable within each class. This produces grouped summaries:

- For loan_status = 0 (non-default customers)
- For loan_status = 1 (default customers)

The results help us interpret how age and income differ between the two classes. For example, we can see whether defaulters tend to be younger or older, and whether their average income is higher or lower compared to non-defaulters. This comparison gives useful insights into which demographic or financial factors may be linked to loan outcomes.

```
age_summary <- aggregate(person_age ~ loan_status, data = df,
                         FUN = function(x) c(mean = mean(x), median =
median(x),
                                              sd = sd(x), min =
min(x), max = max(x)))
age_summary <- do.call(data.frame, age_summary)

# Create summary table for Income
income_summary <- aggregate(person_income ~ loan_status, data = df,
                            FUN = function(x) c(mean = mean(x),
median = median(x),
                                                 sd = sd(x), min =
min(x), max = max(x)))
income_summary <- do.call(data.frame, income_summary)

print(age_summary)
```

Output:

```
  loan_status person_age.mean person_age.median person_age.sd person_age.min person_age.max
1          No        29.10526                24      30.88757            -22            230
2         Yes        25.45600                23      29.61895            -25            350
>
```

```
print(income_summary)
```

Output:

```
  loan_status person_income.mean person_income.median person_income.sd person_income.min person_income.max
1          No          232460.21               249174         95317.19             12282            368115
2         Yes           99662.79                72608        279363.03             12438           3138998
>
```

Task 12 .Compare average credit_score between customers with loan_status 1 and with loan_status 0:

Description:

To compare the average credit score between customers with different loan statuses, we applied the aggregate() function in R. The dataset was grouped by loan_status, where 0 represents customers without default (good status) and 1 represents customers with default (bad status). The mean credit score was then calculated for each group, rounded to two decimal places, and missing values were ignored (na.rm=TRUE).

The results show that customers with loan_status = 0 have an average credit score of 630.09, while customers with loan_status = 1 have an average credit score of 627.55. This indicates that, on average, customers without default (0) have slightly higher credit scores than those with default (1).

```
aggregate(credit_score ~ loan_status, data = df, FUN = function(x)
round(mean(x, na.rm=TRUE), 2))
```

Output:

```
> aggregate(credit_score ~ loan_status, data = df, FUN = function(x) round(mean(x, na.rm=TRUE), 2))
  loan_status credit_score
1           0       630.09
2           1       627.55
>
```

Description:

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

13. Compare spread in person_emp_exp for customers with different levels of person_education:

Description:

In this task we are analyzing how the variable person_emp_exp (employment experience in years) is distributed across different categories of person_education (such as High School, Bachelor, Master, etc.).

We created a custom function compare_spread(), which groups the dataset by the specified categorical variable (person_education) and then calculates descriptive spread measures for the numerical variable (person_emp_exp). Specifically, it computes:

- Count: Number of records in each education level.
- Mean: The average employment experience.
- Standard Deviation (SD): How much the experience varies within that group.
- Minimum and Maximum: The range of values observed.
- Interquartile Range (IQR): The spread of the middle 50% of the data.

By comparing these statistics, we can see whether people with higher education levels tend to have more or less work experience, and whether the variation (spread) in experience differs between groups. For example, graduates may have higher average experience but also larger variation compared to high school educated individuals.

CODE:

```
compare_spread <- function(data, group_col, value_col) {
  library(dplyr)

  if (!group_col %in% names(data)) stop("Group column not found.")
  if (!value_col %in% names(data)) stop("Value column not found.")

  data %>%
    group_by(.data[[group_col]]) %>%
    summarise(
      count = n(),
      mean = mean(.data[[value_col]], na.rm = TRUE),
      sd = sd(.data[[value_col]], na.rm = TRUE),
      min = min(.data[[value_col]], na.rm = TRUE),
      max = max(.data[[value_col]], na.rm = TRUE),
      IQR = IQR(.data[[value_col]], na.rm = TRUE),
      .groups = "drop"
    )
}
aggregate(credit_score ~ loan_status, data = data, mean, na.rm =
TRUE)
```

Output:

```
  loan_status credit_score
1           0     630.0921
2           1     628.0574
>
```

```
compare <- compare_spread(df_clean, "person_education",
"person_emp_exp")
head(as.data.frame(compare))
```

Output:

```
> head(as.data.frame(compare))
  person_education count     mean        sd min max IQR
1        Associate    46 3.760870 17.724039   0 121 2.0
2         Bachelor    73 3.534247 14.533565   0 125 3.0
3        Doctorate     1 2.000000        NA   2   2 0.0
4      High School    58 1.413793  1.797020   0   7 3.0
5           Master    23 1.739130  1.888178   0   6 2.5
>
```

Project Code

```
library(readxl)
library(modeest)
library(naniar)
library(dplyr)
library(rsample)

data <- read_excel("D:/data science project-
mid/data/Midterm_Dataset_Section(A).xlsx")



missing_rows <- data[!complete.cases(data), ]
head(missing_rows)

drop_data <- na.omit(data);
head(drop_data);

df_mean <- data
df_mean$person_income[is.na(df_mean$person_income)] <-
mean(df_mean$person_income, na.rm = TRUE);
head(df_mean)

df_median <- df_mean
df_median$person_age[is.na(df_median$person_age)] <-
median(df_median$person_age, na.rm = TRUE);
head(df_median)

df_mode <- df_median
mode_val <- mlv(df_mode$loan_status, method = "mfv", na.rm
= TRUE);
df_mode$loan_status[is.na(df_mode$loan_status)] <- mode_val
head(as.data.frame(df_mode))

mode_val <- mlv(df_mode$person_gender, method = "mfv",
na.rm = TRUE);
df_mode$person_gender[is.na(df_mode$person_gender)] <-
mode_val
```

```
mode_val <- mlv(df_mode$person_education, method = "mfv",
na.rm = TRUE);
df_mode$person_education[is.na(df_mode$person_education)]
<- mode_val

print(sum(is.na(df_mode)))



vis_miss(data)



df_mode$loan_status <- ifelse(df_mode$loan_status == 1,
"Yes", "No");
head(as.data.frame(df_mode))

df_mode$previous_loan_defaults_on_file <-
ifelse(df_mode$previous_loan_defaults_on_file == "Yes", 1,
0);
head(as.data.frame(df_mode))
df <- df_mode



get_outliers <- function(data, colname) {
  if (!colname %in% names(data)) stop("Column not found.")
  if (!is.numeric(data[[colname]])) stop("Column must be
numeric.")

  Q1 <- quantile(data[[colname]], 0.25, na.rm = TRUE)
  Q3 <- quantile(data[[colname]], 0.75, na.rm = TRUE)
  IQR <- Q3 - Q1

  lower <- Q1 - 1.5 * IQR
  upper <- Q3 + 1.5 * IQR
```

```
  outliers <- data[[colname]][data[[colname]] < lower |
data[[colname]] > upper]
  return(outliers)
}

fix_outliers <- function(data, colname) {
  if (!colname %in% names(data)) stop("Column not found.")
  if (!is.numeric(data[[colname]])) stop("Column must be
numeric.")

  Q1 <- quantile(data[[colname]], 0.25, na.rm = TRUE)
  Q3 <- quantile(data[[colname]], 0.75, na.rm = TRUE)
  IQR <- Q3 - Q1

  lower <- Q1 - 1.5 * IQR
  upper <- Q3 + 1.5 * IQR

  is_outlier <- data[[colname]] < lower | data[[colname]] >
upper
  n_out <- sum(is_outlier, na.rm = TRUE)

  if (n_out > 0) {
    data[[colname]] <- pmin(pmax(data[[colname]], lower),
upper)
  }

  return(data)
}

df <- df_mode
outlier_detect <- get_outliers(df, "person_age")
head(outlier_detect)

df_clean <- fix_outliers(df, "person_age")
head(as.data.frame(df_clean))



df_nor <- df
```

```
normalize <- function(x) {
  return((x - min(x)) / (max(x) - min(x)))
}
df_nor$loan_amnt <- normalize(df_nor$loan_amnt)
head(df_nor)



df_unique <- df
df_unique <- distinct(df, person_education, .keep_all =
TRUE)
head(as.data.frame(df_unique))



filtered_data1 <- filter(df_clean, loan_amnt > 10000)
head(as.data.frame(filtered_data1))

filtered_data2 <- filter(df_clean, !is.na(person_income) &
person_income > 20000)
head(as.data.frame(filtered_data2))

filtered_data3 <- filter(df_clean, person_education %in%
c("Bachelor", "Master"))
head(as.data.frame(filtered_data3))



loans5 <- data
loans5 <- na.omit(loans5)
print(nrow(loans5))
head(as.data.frame(loans5))

loan5 <- data
loan3_clean <- na.omit(loan5)
print(nrow(loan3_clean))
head(as.data.frame(loan3_clean))

num_cols <- sapply(loan5, is.numeric)
```

```
cat_cols <- sapply(loan5, is.character)
loan5[num_cols] <- lapply(loan5[num_cols], function(x) {
  x[is.na(x)] <- mean(x, na.rm = TRUE)
  return(x)
})
loan5[cat_cols] <- lapply(loan5[cat_cols], function(x) {
  mode_val <- names(sort(table(x), decreasing = TRUE))[1]
  x[is.na(x)] <- mode_val
  return(x)
})
print(colSums(is.na(loan5[num_cols])))
print(colSums(is.na(loan5[cat_cols])))
head(as.data.frame(loan5))



df$loan_status <- ifelse(df$loan_status == "Yes", 1, 0);
under_df <- df

library(ROSE)
df$loan_status <- factor(df$loan_status, levels = c(0,1))
table(df$loan_status)
N_under <- 2 * min(table(df$loan_status))
set.seed(199)
under_df <- ovun.sample(loan_status ~ ., data = df, method
= "under", N = N_under, seed = 199)$data
table(under_df$loan_status)
head(under_df)

df$loan_status <- factor(df$loan_status, levels = c(0,1))
table(df$loan_status)
N_over <- 2 * max(table(df$loan_status))
set.seed(199)
over_df <- ovun.sample(loan_status ~ ., data = df, method =
"over", N = N_over, seed = 199)$data
table(over_df$loan_status)
head(over_df)

df_smote <- df_clean
```

```
set.seed(199)
df_smote$previous_loan_defaults_on_file <-
factor(df_smote$previous_loan_defaults_on_file, levels =
c(0,1))
df_smote[sapply(df_smote, is.character)] <-
lapply(df_smote[sapply(df_smote, is.character)], factor)
df_smote <- df_smote[complete.cases(df_smote), ]

table(df_smote$previous_loan_defaults_on_file)
rose_df <- ROSE(previous_loan_defaults_on_file ~ ., data =
df_smote, N = 2000, p = 0.5)$data
table(rose_df$previous_loan_defaults_on_file)
head(rose_df)

orig <- names(df_clean)
cat_vars <- setdiff(orig[sapply(df_smote, function(x)
is.factor(x) || is.character(x))],
                    "previous_loan_defaults_on_file")

pretty_df <- df_clean
for (v in cat_vars) {
  d <- grep(paste0("^", v, "_"), names(pretty_df), value =
TRUE)
  if (length(d)) {
    M  <- as.matrix(pretty_df[, d, drop = FALSE])
    lv <- sub(paste0("^", v, "_"), "", d)
    pretty_df[[v]] <- factor(lv[max.col(M, ties.method =
"first")],
                             levels =
levels(df_smote[[v]]))
    pretty_df[d] <- NULL
  }
}

pretty_df <- pretty_df[, orig, drop = FALSE]
head(as.data.frame(pretty_df))
```

```
set.seed(123)
split <- initial_split(df, prop = 0.7)
train_data <- training(split)
test_data  <- testing(split)
dim(train_data)
dim(test_data)



age_summary <- aggregate(person_age ~ loan_status, data =
df,
                         FUN = function(x) c(mean =
mean(x), median = median(x),
                                             sd = sd(x),
min = min(x), max = max(x)))
age_summary <- do.call(data.frame, age_summary)

income_summary <- aggregate(person_income ~ loan_status,
data = df,
                            FUN = function(x) c(mean =
mean(x), median = median(x),
                                                sd = sd(x),
min = min(x), max = max(x)))
income_summary <- do.call(data.frame, income_summary)

print(age_summary)
print(income_summary)



aggregate(credit_score ~ loan_status, data = df, FUN =
function(x) round(mean(x, na.rm=TRUE), 2))



compare_spread <- function(data, group_col, value_col) {
  library(dplyr)
```

```
  if (!group_col %in% names(data)) stop("Group column not
found.")
  if (!value_col %in% names(data)) stop("Value column not
found.")

  data %>%
    group_by(.data[[group_col]]) %>%
    summarise(
      count = n(),
      mean = mean(.data[[value_col]], na.rm = TRUE),
      sd = sd(.data[[value_col]], na.rm = TRUE),
      min = min(.data[[value_col]], na.rm = TRUE),
      max = max(.data[[value_col]], na.rm = TRUE),
      IQR = IQR(.data[[value_col]], na.rm = TRUE),
      .groups = "drop"
    )
}

aggregate(credit_score ~ loan_status, data = data , mean,
na.rm = TRUE)
compare <- compare_spread(df_clean, "person_education",
"person_emp_exp")
head(as.data.frame(compare))
```