

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: B

Group:05

|  |  |  |  |
| --- | --- | --- | --- |
| **SL**  **#** | **Student**  **Name** | **Student**  **ID** | **Contribution (%)** |
| 1. | Md.Hachibur Rahman | 22-46426-1 | 25% |
| 2. | Sakif Khan | 22-46425-1 | 25% |
| 3. | Mehedi Hasan | 22-46907-1 | 25% |
| 4. | Ushmita Mallick | 22-47703-2 | 25% |

**Dataset Description:**

This dataset contains information on 502 students with 11 attributes related to lifestyle, academic, and psychological factors. The variables include demographic data such as Gender and Age, as well as academic and health-related factors like Academic Pressure, StudySatisfaction, Sleep Duration, Dietary Habits, Study Hours, and Financial Stress. In addition, it also records sensitive mental health indicators such as whether the student has ever had suicidal thoughts, their family history of mental illness, and whether they are diagnosed with depression, which serves as the target variable.

The dataset provides a mix of categorical variables (e.g., Gender, Sleep Duration, Dietary Habits, Suicidal Thoughts, Family History, Depression) and numeric variables (e.g., Age, Academic Pressure, Study Satisfaction, Study Hours, Financial Stress). Ages range roughly from late teens to early thirties, while stress and satisfaction levels are captured on a scale from 1 to 5. Sleep patterns are grouped into categories such as less than 5 hours, 5–6 hours, 7–8 hours, and more than 8 hours. Dietary habits are classified as Healthy, Moderate, or Unhealthy.

This dataset is particularly valuable for studying the relationship between academic life, lifestyle habits, and mental health outcomes. It enables exploratory data analysis to identify trends, for example, how financial stress or sleep duration might influence depression levels among students. Furthermore, it can be used for building predictive models, such as classification algorithms, to determine the likelihood of depression based on the provided factors. Overall, the dataset offers a comprehensive view of student mental health and the variables that potentially affect it.

**Column Descriptions:**

**1.Gender:**Indicates the gender of the student (Male, Female). There are 54% of male & 46% of female students.

**2. Age:**The age of the student (numeric value).

**3.**  **Academic Pressure**:  
Level of academic stress experienced by the student (likely on a scale, 1 to 5, with higher values indicating more pressure).

**4.** **Study Satisfaction**:  
How satisfied the student is with their study habits or academic experience (again, likely on a scale, 1 to 5).

**5. Sleep Duration**:  
Reported sleep range per day (“5-6 hours”, “7-8 hours”, “More than 8 hours”).

**6.** **Dietary Habits**:  
Student's diet type (Healthy, Moderate, Unhealthy).

**7.** **Have you ever had suicidal thoughts?**   
Indicates if the student has ever experienced suicidal thoughts (Yes/No).

**8.** **Study Hours**:  
Average number of hours studied per day.

**9.** **Financial Stress**:  
Level of financial pressure experienced by the student (likely a scale from low to high, 1 to 5).

**10.** **Family History of Mental Illness**:  
Whether there is a known family history of mental health issues (Yes/No).

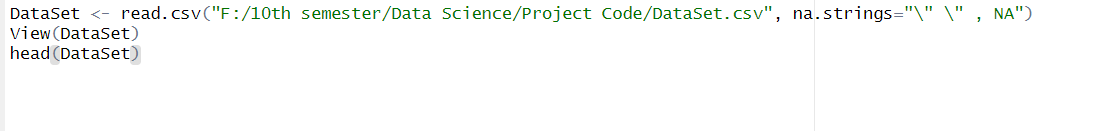
**11.** **Depression**:  
Indicates if the student is classified as experiencing depression (Yes/No), which may be the target variable.

**Data Loading and Initial Overview:**

**Description:**

Reads the CSV file into a dataframe, treating empty strings and "NA" as missing values.

**Code:**

****

**Output:**

**A computer screen shot of a computer code

AI-generated content may be incorrect.**

**Code Description:**

In the R script accesses the CSV file named DataSet.csv located in the directory F:/10th semester/Data Science/Project Code/. During the loading process, any blank cells (“ “) or the text “NA” is considered missing information (NA in R). The loaded information is kept in the variable DataSet. Using the command View(DataSet) allows the user to open the dataset to a spreadsheet-like viewer within RStudio displaying the entire dataset. To assess the dataset preview, head(DataSet) displays the first few rows of the dataset.

**Summarize Dataset:**

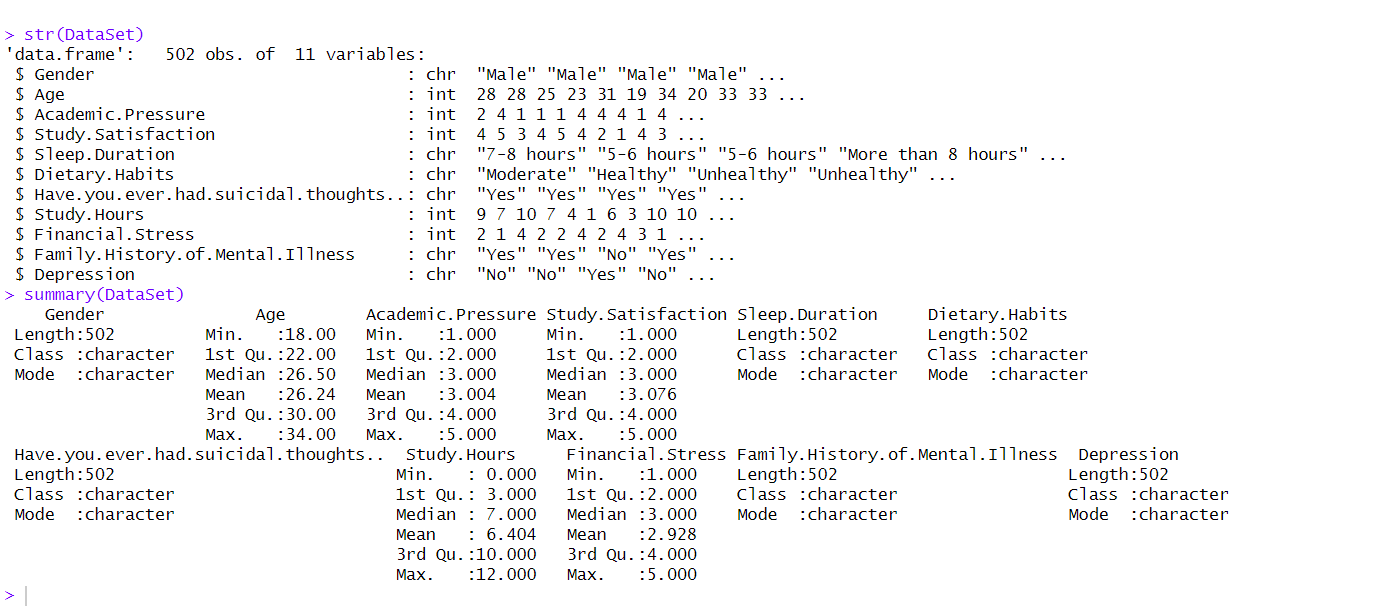
Provides a detailed summary of the dataset using skimr.

**Code:**

**A black text on a white background

AI-generated content may be incorrect.**

**Output:**

****

**Code Description:**

The str(DataSet) function displays the structure of the dataset, how many rows and columns it has the type of data in each column (like numbers, text, or dates), and a small sample of the values. The summary(DataSet) function provides a quick statistical overview for numeric columns, it shows details like the minimum, maximum, mean, and quartiles, while for text or categorical data, it shows how often each category appears. So, these commands give a fast and clear picture of dataset’s layout and key details.

**Define Column Types:**

**Description:**

Separates columns into numeric and categorical types and print them.

**Code:A screenshot of a computer code

AI-generated content may be incorrect.**

**Output:**

**A screenshot of a computer code

AI-generated content may be incorrect.**

**Code Description**

The CSV file "DataSet.csv" is read then loads it into a data frame named "df." After that, it divides its columns into categories and numeric types. Numerical columns are extracted using the sapply() method with is.numeric, and their names are stored in a character vector called numericcols. The logical negation (!) is used in sapply() to choose columns that are NOT numeric, much as the reasoning for numericcols. A character vector known as categoricalcols contains the names of categorical columns. The column names for categorical\_cols and numericcols, which differentiate the two data types, are then printed.

**Summary Statistics for Columns:**

**Description:**

Prints the summary of numeric columns and counts unique values in categorical columns.

**Code:**

**A screenshot of a computer program

AI-generated content may be incorrect.**

**Output:A screen shot of a computer code

AI-generated content may be incorrect.**

**Code Description:**

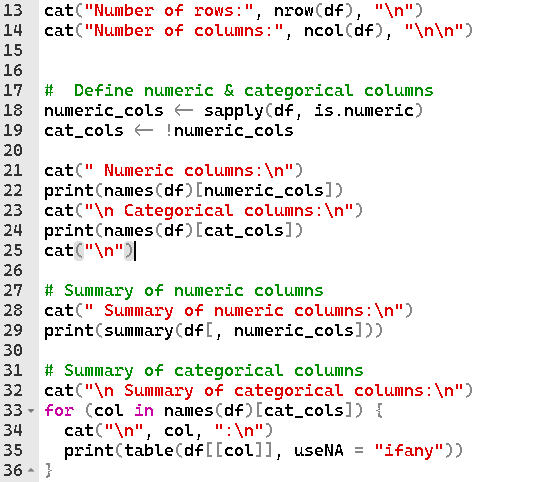
Use this R script to download a dataset from the file named DataSet.csv to get two types of information. The data for all the numeric columns and calculates the summary for each of the relevant numeric fields. It computes the minimum, maximum, mean, median, and quartiles. In the next summary step, all non-numeric fields are extracted and for each non-numeric fields, the count of different unique values is determined. In the end the script prints out all the numeric summary and the unique value counts together, in a clear labelled manner making all the results very easy to read and interpret.

**Numeric And Categorical Column Define:**

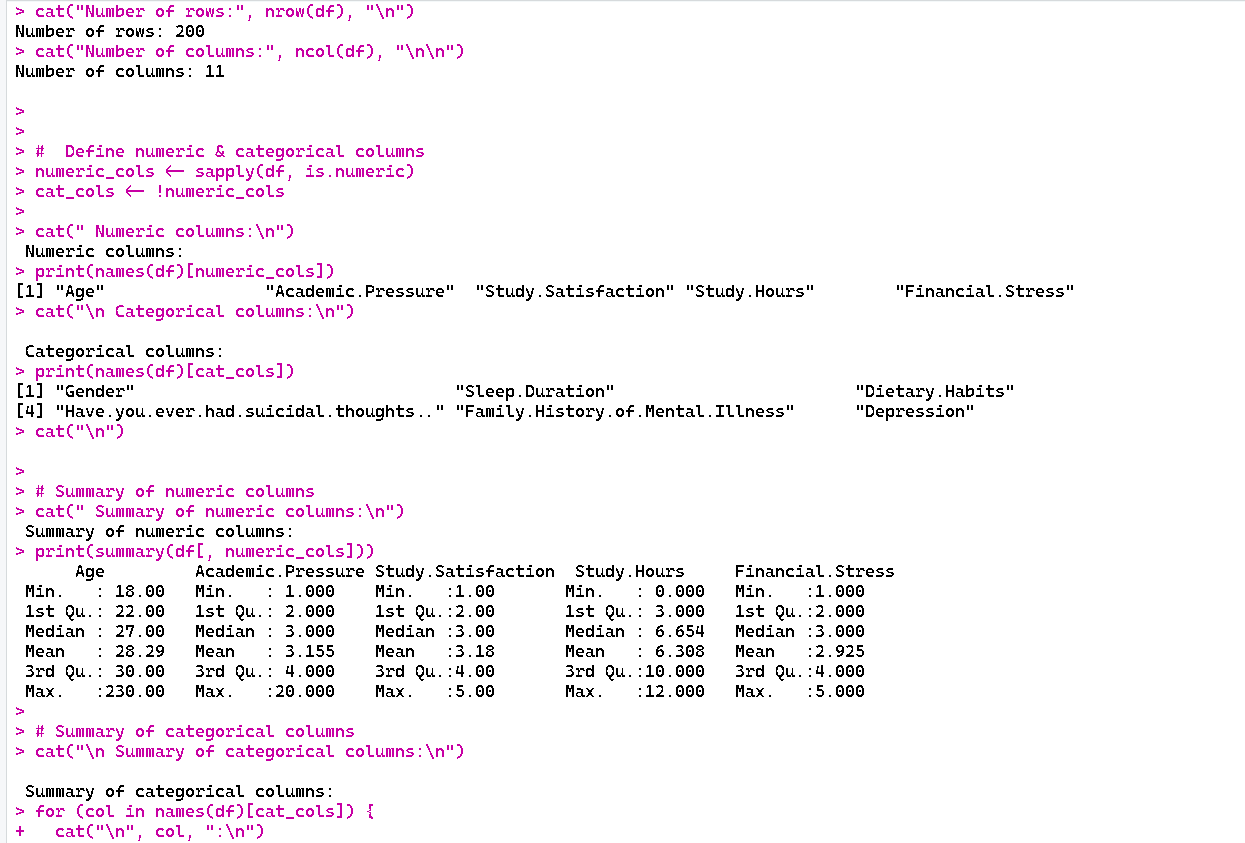
**Description:**

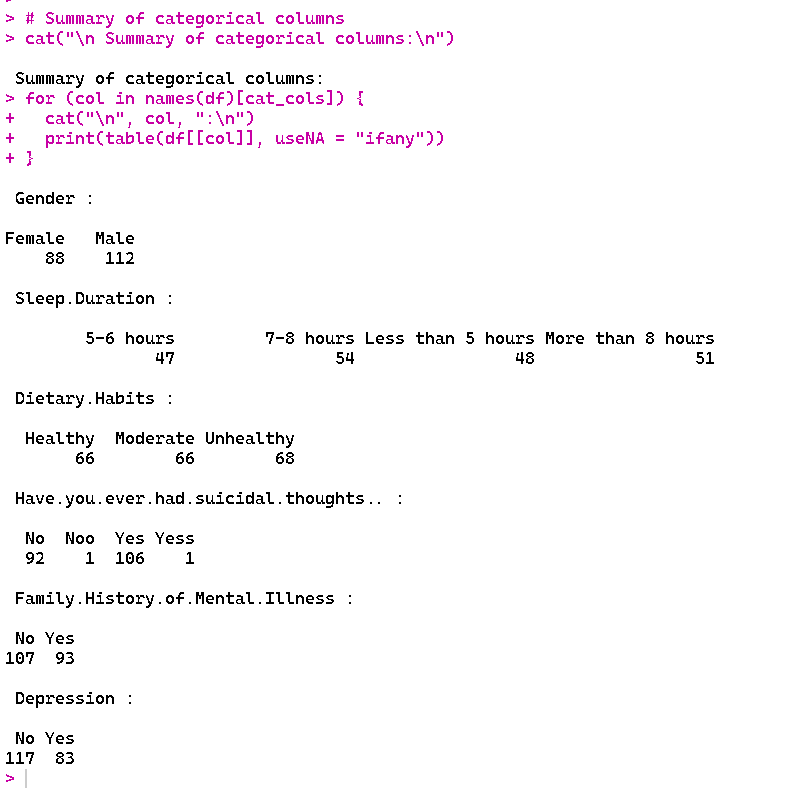
Print the Numeric And Categorical Column and also summary of Numeric And Categorical Column.

**Code :**

****

**Output:**

****

****

**Code Description:**

The given R code summarizes the structure and contents of the dataset df in order to do a first exploratory study. Using nrow(df) and ncol(df), it first outputs the total number of rows and columns. In this instance, it reveals that the dataset has 201 rows and 11 columns. The code then determines whether columns are category and numeric. It generates a logical vector numeric\_cols that indicates which columns are numeric using sapply(df, is.numeric). Categorical columns are identified by cat\_cols, the vector's complement. "Age", "Academic.Pressure", "Study.Satisfaction", "Study.Hours", and "Financial.Stress" are the numeric columns selected from the output. "Gender," "Sleep Duration," "Dietary Habits," "Have you ever had suicidal thoughts?" "Family History of Mental Illness," and "Depression" are the category columns.The code generates a summary for numeric columns using summary(df[, numeric\_cols]), which gives each column's lowest, first quartile, median, mean, third quartile, maximum, and NA counts. For instance, "Age" contains three missing values and ranges from a minimum of 18, a median of 26, a mean of 28.25, and a maximum of 230. In a similar vein, additional numerical columns like "Academic.Pressure" and "Study.Hours" display their distribution and aid in identifying anomalies or outliers (such as "Age" max = 230, which could be a mistake).When dealing with categorical columns, the code iterates over each one and uses table() to create a frequency table that includes the number of missing values (useNA = "ifany"). Distributions like these are displayed in the output:

"Gender": 110 females, 88 men, and 3 absent.

"Sleep.Duration": the majority of pupils sleep for seven to eight hours or longer.

"Dietary.Habits": a good mix of moderate, unhealthy, and healthy habits.

Some of the entries in "Have.you.ever.had.suicidal.thoughts.." are inconsistent, having "Noo" and "Yess" in between.

"Family.History.of.Mental.Illness" has 93 historical entries and 108 without.

115 "Yes" and 83 "No" responses for "Depression"; 3 are missing.

**Check for duplicate:**

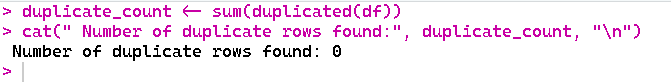
**Description:**

Counts and displays the number of duplicate rows.

**Code:**

****

**Output:**

****

**Code Description:**

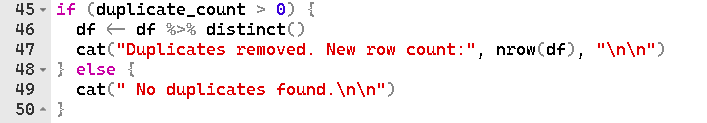
This R function is intended to identify and tally duplicate rows in the df data frame. It makes use of the duplicated(df) function, which looks at every row and gives back a logical vector that indicates if a row is a copy of one that came before it. Duplicate rows are indicated by TRUE, whereas unique rows are indicated by FALSE. Since TRUE is considered as 1 and FALSE as 0 in arithmetic operations, the code determines the total number of duplicate rows by applying sum(duplicated(df)). The cat() function is then used to report the outcome, printing the duplicate count and a descriptive message. There was just one duplicate row in the output displayed. Finding duplicates is an essential step in data cleaning that helps preserve dataset accuracy and prevent biased modeling or analysis errors.

**Handle duplicate:**

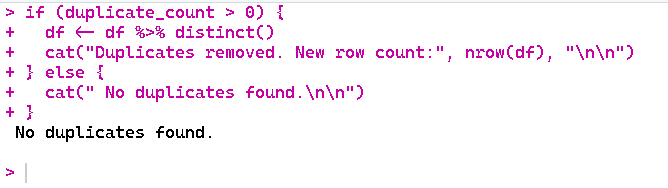
**Description:**

Remove duplicate row if there has any duplicate row.

**Code:**

****

**Output:**

****

**Code Description:**

The purpose of this R code snippet is to identify and eliminate duplicate rows from the data frame `df`. The variable `duplicate\_count`, which should hold the number of duplicate rows, is first checked to see if it is larger than zero. If duplicates are present, the code removes all duplicate rows using the `distinct()` function from the `dplyr` package, leaving only unique entries. Following the removal of duplicates, it uses `nrow(df)` to display the updated number of rows in the data frame and produces a message stating that duplicates have been eliminated. The `else` block runs and prints a message stating that no duplicates were identified if `duplicate\_count` is 0. Prior to additional analysis, this guarantees that the data is clear and devoid of duplicate entries. The result indicates that 200 rows are left.

**Missing values:**

**Description:**

Check the null value and there has any missing value then Convert missing values to NA.

**Code:**

****

**Output:**

****

**Code Description:**

We supplied In order to manage and detect blank cells in a dataset, R code transforms them into NA, which stands for missing or null values in R. df[df == "" | df == " "], the first line \- NA searches the full data frame df using logical indexing to find any cells that are either entirely empty ("") or have a single space (" "). Upon identifying such cells, they are substituted with NA, therefore designating them as missing values. This enables R functions and packages to appropriately identify and manage them throughout the analytic process. The second line, cat(" Blank cells converted to NA (null values in R)"), gives users unambiguous feedback on the data cleaning stage by indicating that the conversion has been completed. By doing this, mistakes brought on by inconsistent or blank entries are prevented and the dataset is guaranteed to be normalized for further analysis.

**Null Value:**

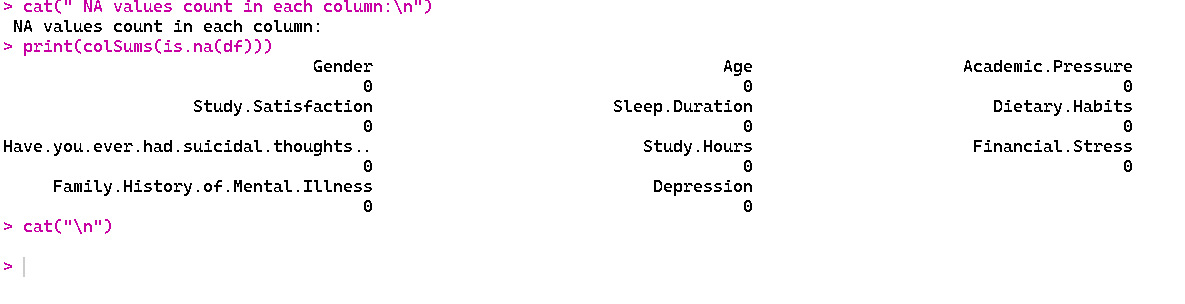
**Description:**

Count NA values and show the null value

**Code:**

****

**Output:**

****

**Code Description:**

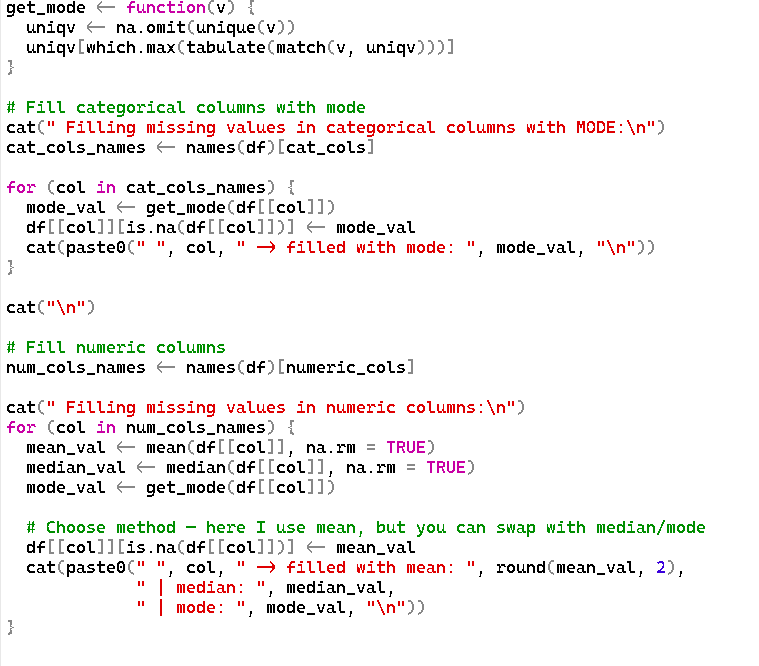
The purpose of the offered R code is to detect and manage blank or empty cells in order to clean a dataset. R's standard representation of missing or null values, `NA`, is used in the first line, `df[df == "" | df == " "] <- NA`, to search the full data frame `df` for cells that are either totally empty (`""`) or contain a single space (`" "`). This prevents blank strings from being mistakenly interpreted as genuine entries in later studies, ensuring that missing data is appropriately recognized. After that, a message stating that the blank cells have been changed to `NA` is printed using the `cat()` method. Using `colSums(is.na(df))`, the output further provides a summary of missing values in each column, displaying the precise number of `NA`s in each column. Preprocessing data and evaluating its quality depend on this stage.

**Handle missing values:**

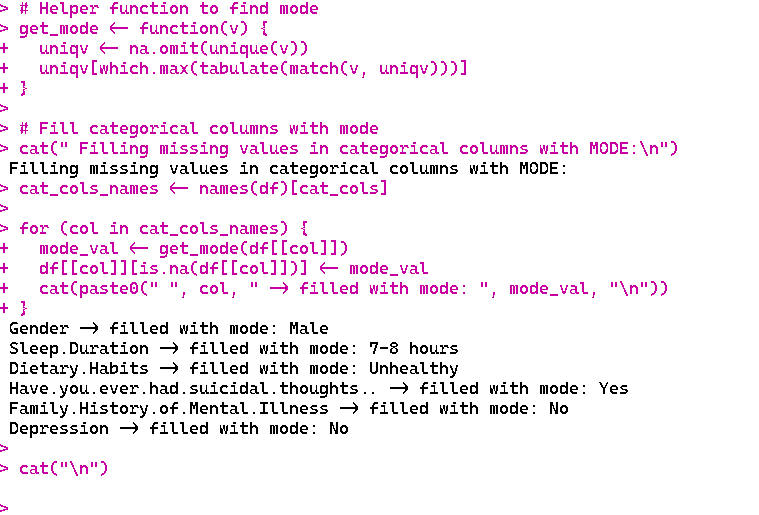
**Description:**

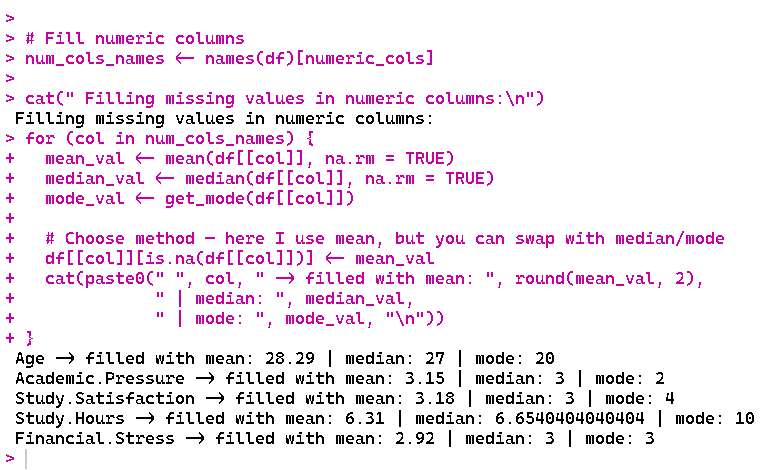
Iterating over all columns and reporting the changes, the code fills in missing values by substituting the mean (or optionally the median/mode) in numeric columns and the mode in categorical columns for `NA`s.

**Code:**

****

**Output:**

****

****

**Code Description:**

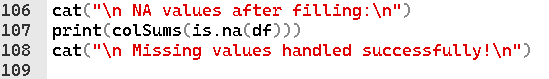
This R function fills up the numeric and category columns with the relevant summary statistics to deal with missing values in a dataset. To determine a vector's mode while disregarding `NA` values, a utility method called `get\_mode()` is developed. The most common category (mode) is used to replace missing values in categorical columns. This process iterates through all categorical column names and reports the replacements. The mean of the column is used to fill in missing values for numeric columns, and the median and mode are also computed and shown for reference. The code gives you the option of impute using the mean, median, or mode. To make sure there are no more `NA`s in the dataset, `cat()` is used throughout to produce explicit console output that indicates which columns were modified and the values utilized.

**Check Missing Value:**

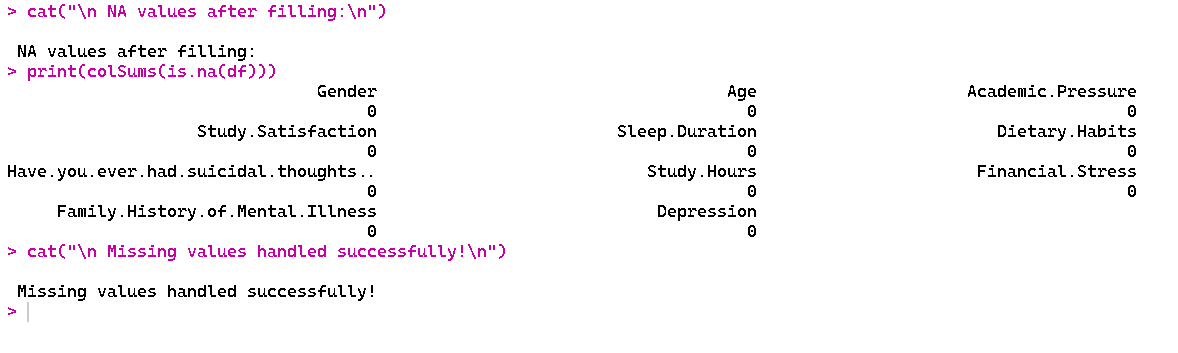
**Description:**

Final Null value check

**Code:**

****

**Output:**

****

**Code Description:**

The purpose of this R code snippet is to validate that handling missing values in a dataset was successful. The code initially prints the unambiguous message, "NA values after filling," to signal the beginning of the verification procedure following imputation or replacement of NA values in earlier phases. The total number of remaining NA values in each column of the data frame df is then determined using the colSums(is.na(df)) function, giving a column-by-column summary of any missing items.

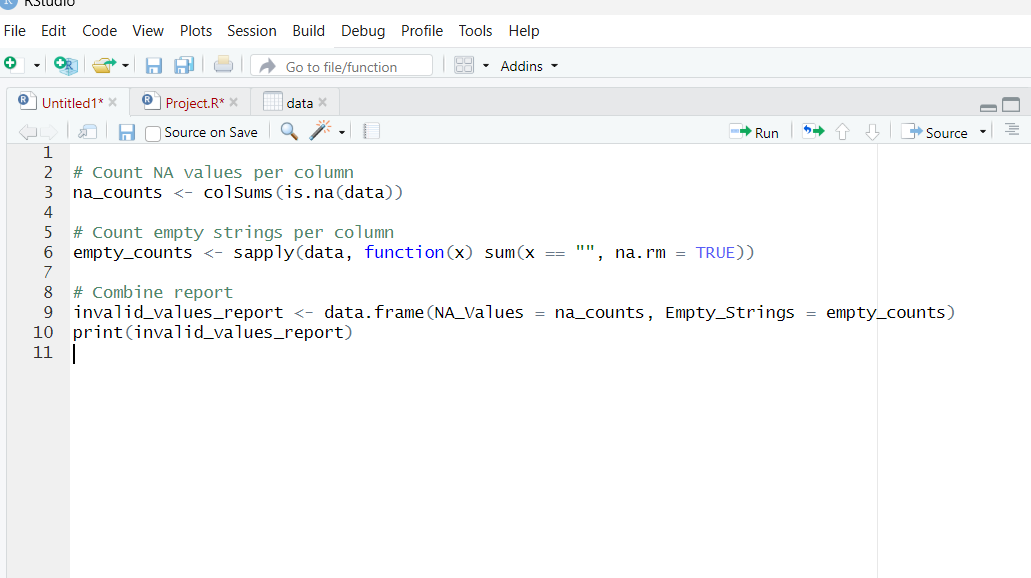
This data, which indicates that all previously missing values in columns like Gender, Age, Academic.Pressure, and others have been properly filled, is produced in a legible manner using the print() method. The user is reassured that the dataset is now complete and prepared for additional analysis when the confirmation message, "Missing values handled successfully," is finally shown.

**Invalid Value:**

**Description:**

Invalid values are data entries that do not conform to the expected format, range, or type for a given variable.

**Code:**

****

**Output:**

**A screenshot of a computer

AI-generated content may be incorrect.**

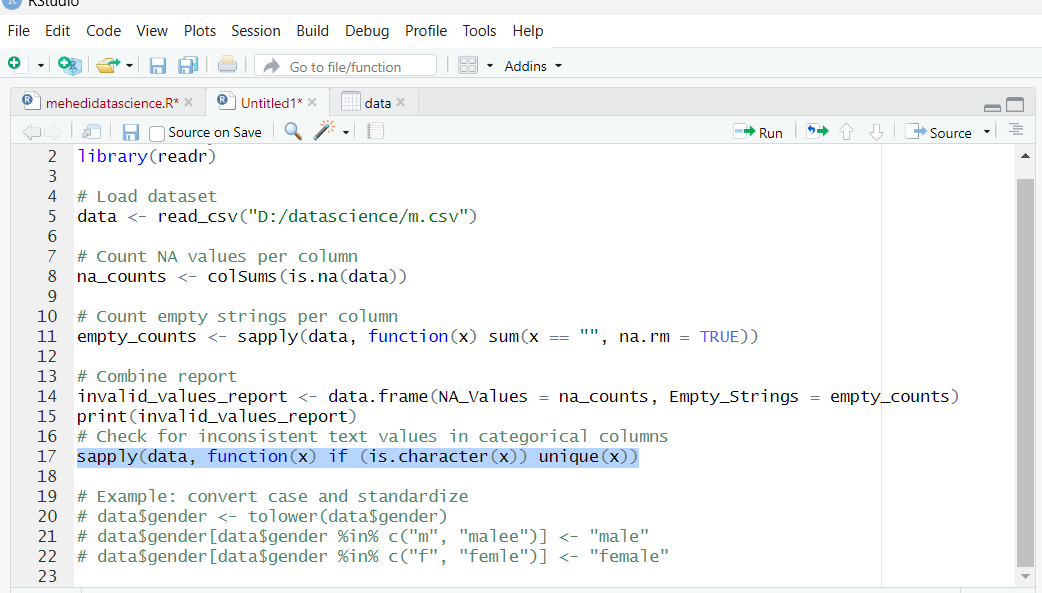
**Code Description:**

It checks each column of your dataset for missing values (NA) and emptystrings (""), counts them separately, and combines the results into a table (invalid\_values\_report) showing how many of each type appear in every column.

**Noisy Value:**

**Description:**

Noisy values are random, inconsistent, or meaningless variations in data that do not represent the true underlying pattern.

**Code:**

**Output:**

**A screenshot of a computer

AI-generated content may be incorrect.**

**Code Description:**

It loops through each column, and for text columns, lists all unique values so you can spot inconsistent or noisy entries.

sapply(data, ...) → Loops over each column in the data dataset.

if (is.character(x)) → Checks if the current column (x) is a text (character) column.

unique(x) → Returns all the distinct values in that column (removes duplicates). For each categorical (text) column, you get a list of all unique entries. This helps you spot inconsistent or noisy labels, such as "Male", "male", "M" in the same column.

**Measure of central tendency:**

**Description:**

Measures of central tendency are statistical values that describe the center or typical value of a dataset.  
They indicate where most data points are clustered and provide a single value that represents the entire dataset.

**Code:**

A close-up of a computer code

Description automatically generated

**Output:**

A white background with black and white clouds

Description automatically generated

**Code Description:**

This code defines a function to calculate the mean, median, and mode for numeric columns in a dataset, applies it to all numeric columns, and prints the results.

**Measure of spread:**

**Description:**

A **measure of spread** describes how much the values in a dataset vary or deviate from the central value, such as the mean or median. It provides insight into the variability and consistency of the data, indicating whether the values are closely clustered or widely dispersed. Common measures of spread include the **range**, which shows the difference between the maximum and minimum values; the **variance** and **standard deviation**, which quantify the average distance of values from the mean; and the **interquartile range (IQR),** which measures the spread of the middle 50% of the data. Understanding the spread is essential for interpreting the reliability, patterns, and distribution of the dataset

**Code:**

A screen shot of a computer code

Description automatically generated

**Output:**

A screenshot of a computer

Description automatically generated

**Code Description:**

The R code basically go-to for cranking out stats on every numeric column in this thing called DataSet. First off, it does a little detective work with sapply(DataSet, is.numeric), sniffing out which columns are actually filled with numbers—not, like, someone’s random text entries or whatever. That info lands in a thing called numeric\_cols. Now, buckle up for the classic for loop. It just barrels through each numeric column, one by one. For each one, it literally spells out the name of the column (in case you forgot what you were looking at), then it throws down four stats that every stats nerd loves: Range (that’s max minus min, nothing fancy), Interquartile Range (IQR for the cool kids), Variance, and Standard Deviation. Oh, and it’s not going to freak out if there are any NAs lurking—na.rm=TRUE makes sure those get quietly ignored.

**Outliers:**

**Description:**

Outliers are data points that are significantly different from most other observations in a dataset.

**Code:**A screenshot of a computer program

AI-generated content may be incorrect.

**Output:**

**A white rectangular object with a black border

AI-generated content may be incorrect.**

**Code Description:**

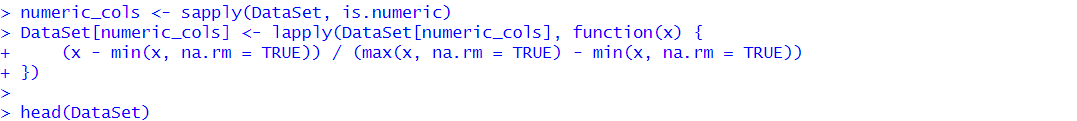
This code finds outliers in all numeric columns of data using the IQR(Interquartile Range) method and returns their row positions. This code loops through each column, checks if it’s numeric, calculates the IQR (Interquartile Range), and flags values that fall outside Q1 − 1.5×IQR or Q3 + 1.5×IQR as outliers, returning their row positions. Each element corresponds to a column.The values are row numbers of detected outliers in that column.

**Normalization:**

**Description:**

Data normalization is the process of transforming numeric data into a common scale—usually between 0 and 1 or −1 and 1—without changing the differences in the range of values.  
It ensures that all features contribute equally to analysis or machine learning models, preventing variables with large ranges from dominating results.

**Code:**



**Output:**

A white background with colorful text

Description automatically generated with medium confidence

**Code Description:**

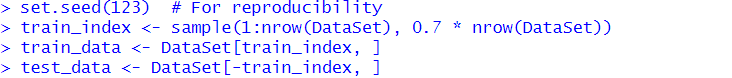
This code finds numeric columns, replaces their missing values with the column mean, scales them between 0 and 1 using Min-Max normalization, and shows their summary statistics.

**Train Test Split:**

**Description:**

Randomly split the dataset into 70% training and 30% testing data.

**Code:**



**Output:**

A close-up of a computer code

Description automatically generated

**Code Description:**

This code performs a **70-30 split** of the dataset into training and testing sets. First, set.seed(123) is used to ensure the random selection of rows can be reproduced in future runs. The sample() function then randomly selects 70% of the row indices from the dataset (1:nrow(DataSet)) and stores them in train\_index. Using these indices, train\_data is created by selecting the corresponding rows from DataSet, while test\_data contains the remaining rows (those not in train\_index). Finally, the cat() function prints the sizes of the resulting datasets, showing that the training set contains **351 rows** and the testing set contains **151 rows.**

**Sampling:**

**Description:**

**Sampling** refers to randomly selecting a portion of the dataset’s rows to create subsets for model training and testing. It ensures that each row has an equal chance of being chosen, helping maintain the dataset’s overall distribution while preventing bias.

**Code:**



**Output:**

A screenshot of a computer

Description automatically generated

**Code Description:**

This output shows a random sample of 10 rows taken from the normalized dataset. The displayed records include various features such as Gender, normalized Age, Academic Pressure, Study Satisfaction, Sleep Duration, Dietary Habits, history of suicidal thoughts, normalized Study Hours, Financial Stress, Family History of Mental Illness, and Depression status. The normalized numeric values (e.g., Age, Academic Pressure, Study Satisfaction) range between 0 and 1, while categorical variables (e.g., Gender, Sleep Duration, Dietary Habits) remain in their original form. This sample demonstrates a mix of male and female participants, varied academic pressures, different sleep patterns, and dietary habits. It also reflects diversity in mental health indicators, with some participants reporting suicidal thoughts, family history of mental illness, or depression, and others not showing such indicators. This small subset provides a quick overview of the dataset’s diversity while maintaining the characteristics of the normalized data.

**Project Code:**

> DataSet <- read.csv("E:\\Aiub\\11TH SEMESTER MID TERM\\INTRODUCTION TO DATASCIENCE\\Project\\Hachibur Rahman\\Depression Student Dataset.csv",

+ header = TRUE, na.strings = c("", "NA", "NaN", "NULL"))

>

> head(DataSet)

Gender Age Academic.Pressure Study.Satisfaction Sleep.Duration Dietary.Habits Have.you.ever.had.suicidal.thoughts.. Study.Hours Financial.Stress

1 Male 28 2 4 7-8 hours Moderate Yes 9 2

2 Male 28 4 5 5-6 hours Healthy Yes 7 1

3 Male 25 1 3 5-6 hours Unhealthy Yes 10 4

4 Male 23 1 4 More than 8 hours Unhealthy Yes 7 2

5 Female 31 1 5 More than 8 hours Healthy Yes 4 2

6 Male 19 4 4 5-6 hours Unhealthy Yes 1 4

Family.History.of.Mental.Illness Depression

1 Yes No

2 Yes No

3 No Yes

4 Yes No

5 Yes No

6 Yes Yes

> str(DataSet)

'data.frame': 502 obs. of 11 variables:

$ Gender : chr "Male" "Male" "Male" "Male" ...

$ Age : int 28 28 25 23 31 19 34 20 33 33 ...

$ Academic.Pressure : num 2 4 1 1 1 4 4 4 1 4 ...

$ Study.Satisfaction : num 4 5 3 4 5 4 2 1 4 3 ...

$ Sleep.Duration : chr "7-8 hours" "5-6 hours" "5-6 hours" "More than 8 hours" ...

$ Dietary.Habits : chr "Moderate" "Healthy" "Unhealthy" "Unhealthy" ...

$ Have.you.ever.had.suicidal.thoughts..: chr "Yes" "Yes" "Yes" "Yes" ...

$ Study.Hours : int 9 7 10 7 4 1 6 3 10 10 ...

$ Financial.Stress : int 2 1 4 2 2 4 2 4 3 1 ...

$ Family.History.of.Mental.Illness : chr "Yes" "Yes" "No" "Yes" ...

$ Depression : chr "No" "No" "Yes" "No" ...

> summary(DataSet)

Gender Age Academic.Pressure Study.Satisfaction Sleep.Duration

Length:502 Min. :18.00 Min. :1.000 Min. :1.000 Length:502

Class :character 1st Qu.:22.00 1st Qu.:2.000 1st Qu.:2.000 Class :character

Mode :character Median :26.50 Median :3.000 Median :3.000 Mode :character

Mean :26.24 Mean :3.004 Mean :3.076

3rd Qu.:30.00 3rd Qu.:4.000 3rd Qu.:4.000

Max. :34.00 Max. :5.000 Max. :5.000

Dietary.Habits Have.you.ever.had.suicidal.thoughts.. Study.Hours Financial.Stress

Length:502 Length:502 Min. : 0.000 Min. :1.000

Class :character Class :character 1st Qu.: 3.000 1st Qu.:2.000

Mode :character Mode :character Median : 7.000 Median :3.000

Mean : 6.404 Mean :2.928

3rd Qu.:10.000 3rd Qu.:4.000

Max. :12.000 Max. :5.000

Family.History.of.Mental.Illness Depression

Length:502 Length:502

Class :character Class :character

Mode :character Mode :character

> categorical\_cols <- c("Gender", "Sleep.Duration", "Dietary.Habits",

+ "Have.you.ever.had.suicidal.thoughts..",

+ "Family.History.of.Mental.Illness", "Depression")

>

> DataSet[categorical\_cols] <- lapply(DataSet[categorical\_cols], as.factor)

>

> # Confirm structure

> str(DataSet)

'data.frame': 502 obs. of 11 variables:

$ Gender : Factor w/ 2 levels "Female","Male": 2 2 2 2 1 2 1 1 1 2 ...

$ Age : int 28 28 25 23 31 19 34 20 33 33 ...

$ Academic.Pressure : num 2 4 1 1 1 4 4 4 1 4 ...

$ Study.Satisfaction : num 4 5 3 4 5 4 2 1 4 3 ...

$ Sleep.Duration : Factor w/ 4 levels "5-6 hours","7-8 hours",..: 2 1 1 4 4 1 4 4 4 3 ...

$ Dietary.Habits : Factor w/ 3 levels "Healthy","Moderate",..: 2 1 3 3 1 3 2 1 2 3 ...

$ Have.you.ever.had.suicidal.thoughts..: Factor w/ 2 levels "No","Yes": 2 2 2 2 2 2 2 2 1 2 ...

$ Study.Hours : int 9 7 10 7 4 1 6 3 10 10 ...

$ Financial.Stress : int 2 1 4 2 2 4 2 4 3 1 ...

$ Family.History.of.Mental.Illness : Factor w/ 2 levels "No","Yes": 2 2 1 2 2 2 1 2 1 1 ...

$ Depression : Factor w/ 2 levels "No","Yes": 1 1 2 1 1 2 2 2 1 2 ...

>

> cat\_cols <- names(DataSet)[sapply(DataSet, is.factor)]

> for (col in cat\_cols) {

+ cat("\n---", col, "---\n")

+ print(table(DataSet[[col]], useNA = "ifany"))

+ }

--- Gender ---

Female Male

235 267

--- Sleep.Duration ---

5-6 hours 7-8 hours Less than 5 hours More than 8 hours

123 128 123 128

--- Dietary.Habits ---

Healthy Moderate Unhealthy

161 172 169

--- Have.you.ever.had.suicidal.thoughts.. ---

No Yes

242 260

--- Family.History.of.Mental.Illness ---

No Yes

265 237

--- Depression ---

No Yes

250 252

|  |
| --- |
| > dup\_count <- sum(duplicated(DataSet))  > cat("Duplicate Rows:", dup\_count, "\n")  Duplicate Rows: 0  >  > if (dup\_count > 0) {  + DataSet <- DataSet[!duplicated(DataSet), ]  + }  >  > # 2. Convert all possible missing-like values to NA  > DataSet[DataSet == ""] <- NA  > DataSet[DataSet == "NA"] <- NA  > DataSet[DataSet == "NaN"] <- NA  > DataSet[DataSet == "NULL"] <- NA  >  > cat("\nMissing Values per Column:\n")  Missing Values per Column:  > print(colSums(is.na(DataSet)))  Gender Age  0 0  Academic.Pressure Study.Satisfaction  0 0  Sleep.Duration Dietary.Habits  0 0  Have.you.ever.had.suicidal.thoughts.. Study.Hours  0 0  Financial.Stress Family.History.of.Mental.Illness  0 0  Depression  0  >  > if ("Age" %in% colnames(DataSet)) {  + invalid\_age <- DataSet$Age < 18 | DataSet$Age > 120  + cat("\nInvalid Age Count:", sum(invalid\_age, na.rm = TRUE), "\n")  + DataSet$Age[invalid\_age] <- NA # convert invalid to NA  + }  Invalid Age Count: 0  >  > for (col in colnames(DataSet)) {  + if (is.numeric(DataSet[[col]])) {  + # Replace NA with mean (average)  + mean\_val <- mean(DataSet[[col]], na.rm = TRUE)  + DataSet[[col]][is.na(DataSet[[col]])] <- mean\_val  + } else {  + # Replace NA with most frequent value (mode)  + freq\_val <- names(sort(table(DataSet[[col]]), decreasing = TRUE))[1]  + DataSet[[col]][is.na(DataSet[[col]])] <- freq\_val  + }  + }  >  > cat("\nMissing Values after cleaning:\n")  Missing Values after cleaning:  > print(colSums(is.na(DataSet)))  Gender Age  0 0  Academic.Pressure Study.Satisfaction  0 0  Sleep.Duration Dietary.Habits  0 0  Have.you.ever.had.suicidal.thoughts.. Study.Hours  0 0  Financial.Stress Family.History.of.Mental.Illness  0 0  Depression  0  > |
|  |
| |  | | --- | | > | |

> numeric\_cols <- names(DataSet)[sapply(DataSet, is.numeric)]

>

> cat("\nOutlier Detection:\n")

Outlier Detection:

> for (col in numeric\_cols) {

+ Q1 <- quantile(DataSet[[col]], 0.25, na.rm = TRUE)

+ Q3 <- quantile(DataSet[[col]], 0.75, na.rm = TRUE)

+ IQR\_val <- Q3 - Q1

+ lower <- Q1 - 1.5 \* IQR\_val

+ upper <- Q3 + 1.5 \* IQR\_val

+ outlier\_count <- sum(DataSet[[col]] < lower | DataSet[[col]] > upper, na.rm = TRUE)

+ cat("Outliers in", col, ":", outlier\_count, "\n")

+ }

Outliers in Age : 0

Outliers in Academic.Pressure : 0

Outliers in Study.Satisfaction : 0

Outliers in Study.Hours : 0

Outliers in Financial.Stress : 0

>

> cat("\nMeasure of Central Tendency:\n")

Measure of Central Tendency:

> for (col in numeric\_cols) {

+ mode\_val <- as.numeric(names(sort(table(DataSet[[col]]), decreasing = TRUE))[1])

+ cat(col,

+ "\n Mean:", mean(DataSet[[col]], na.rm = TRUE),

+ "\n Median:", median(DataSet[[col]], na.rm = TRUE),

+ "\n Mode:", mode\_val, "\n\n")

+ }

Age

Mean: 26.24104

Median: 26.5

Mode: 28

Academic.Pressure

Mean: 3.003984

Median: 3

Mode: 3

Study.Satisfaction

Mean: 3.075697

Median: 3

Mode: 4

Study.Hours

Mean: 6.404382

Median: 7

Mode: 10

Financial.Stress

Mean: 2.928287

Median: 3

Mode: 1

> cat("\nMeasure of Spread:\n")

Measure of Spread:

> for (col in numeric\_cols) {

+ cat(col,

+ "\n SD:", sd(DataSet[[col]], na.rm = TRUE),

+ "\n Variance:", var(DataSet[[col]], na.rm = TRUE),

+ "\n IQR:", IQR(DataSet[[col]], na.rm = TRUE),

+ "\n Range:", diff(range(DataSet[[col]], na.rm = TRUE)), "\n\n")

+ }

Age

SD: 4.896501

Variance: 23.97572

IQR: 8

Range: 16

Academic.Pressure

SD: 1.390007

Variance: 1.93212

IQR: 2

Range: 4

Study.Satisfaction

SD: 1.37349

Variance: 1.886474

IQR: 2

Range: 4

Study.Hours

SD: 3.742434

Variance: 14.00581

IQR: 7

Range: 12

Financial.Stress

SD: 1.425053

Variance: 2.030775

IQR: 2

Range: 4

> normalize <- function(x) {

+ return((x - min(x, na.rm = TRUE)) / (max(x, na.rm = TRUE) - min(x, na.rm = TRUE)))

+ }

>

> DataSet[numeric\_cols] <- lapply(DataSet[numeric\_cols], normalize)

>

> cat("\nAfter Normalization (0–1 scale):\n")

After Normalization (0–1 scale):

> summary(DataSet[numeric\_cols])

Age Academic.Pressure Study.Satisfaction Study.Hours Financial.Stress

Min. :0.0000 Min. :0.000 Min. :0.0000 Min. :0.0000 Min. :0.0000

1st Qu.:0.2500 1st Qu.:0.250 1st Qu.:0.2500 1st Qu.:0.2500 1st Qu.:0.2500

Median :0.5312 Median :0.500 Median :0.5000 Median :0.5833 Median :0.5000

Mean :0.5151 Mean :0.501 Mean :0.5189 Mean :0.5337 Mean :0.4821

3rd Qu.:0.7500 3rd Qu.:0.750 3rd Qu.:0.7500 3rd Qu.:0.8333 3rd Qu.:0.7500

Max. :1.0000 Max. :1.000 Max. :1.0000 Max. :1.0000 Max. :1.0000

>

> set.seed(123)

>

> train\_index <- sample(1:nrow(DataSet), size = 0.7 \* nrow(DataSet))

>

> train\_data <- DataSet[train\_index, ]

> test\_data <- DataSet[-train\_index, ]

>

> cat("Training Data Rows:", nrow(train\_data), "\n")

Training Data Rows: 351

> cat("Testing Data Rows:", nrow(test\_data), "\n")

Testing Data Rows: 151

>

set.seed(123)

> sample\_data <- DataSet[sample(1:nrow(DataSet), 100), ] rows

>

> View(sample\_data)

> head(sample\_data, 10)

Gender Age Academic.Pressure Study.Satisfaction Sleep.Duration Dietary.Habits

415 Female 0.0625 0.00 1.00 More than 8 hours Moderate

463 Female 0.0625 0.00 0.25 More than 8 hours Healthy

179 Female 0.8125 0.00 0.25 7-8 hours Moderate

14 Male 0.4375 0.00 0.00 5-6 hours Moderate

195 Female 0.0000 0.75 0.00 More than 8 hours Healthy

426 Male 0.5000 1.00 0.75 Less than 5 hours Unhealthy

306 Female 0.3125 0.25 0.25 More than 8 hours Healthy

118 Male 0.4375 0.50 1.00 7-8 hours Unhealthy

299 Female 0.9375 0.75 1.00 7-8 hours Healthy

229 Male 0.4375 0.75 0.25 Less than 5 hours Moderate

Have.you.ever.had.suicidal.thoughts.. Study.Hours Financial.Stress

415 No 0.16666667 0.00

463 No 0.58333333 1.00

179 Yes 0.75000000 0.75

14 Yes 1.00000000 0.50

195 No 0.83333333 0.00

426 Yes 0.91666667 0.00

306 No 0.08333333 0.00

118 Yes 0.16666667 0.00

299 No 1.00000000 0.75

229 No 0.08333333 0.50

Family.History.of.Mental.Illness Depression

415 Yes No

463 No No

179 No No

14 Yes Yes

195 Yes Yes

426 Yes Yes

306 Yes No

118 No No

299 Yes No

229 Yes Yes