

American International University- Bangladesh

Data Science

Project Report

Summer 22-23

Name : Nayeem Abdul Qaiyum

ID : 20-43581-1

Section : C

Date Of Submit: 18.07.23

Submitted By

# Abdus Salam

Assistant Professor, CS

**Description:**

The Titanic dataset is a widely-known data science collection of information about passengers aboard the Titanic, including their age, gender, siblings, parents/children, passenger fare, Port of Embarkation, ticket class, categories, and survival status. The dataset contains numerous rows and 10 columns, with some data points missing. It includes integer, numeric, and character attributes, and the goal is to obtain a clean preprocessed dataset. The dataset includes various types of passengers, including man, women, children, and those who were alone or not.

Table of Contents

* Import data
* Find the shape of the dataset
* Show the attributes names of the dataset
* Find the structure of the dataset
* The first few rows of dataset
* Summary of the dataset
* Find the type of attribute
* Measure of spread range and standard deviation
* Find the missing value for all column
* Missing value of Gender & Age
* Most frequent value
* Replacing missing value by most frequent value for gender attributes
* Visualization
* Data Cleaning
* Annotate
* Outlier
* Data Transformation
* Invalid Value

**Project Solution**

**Import data:**

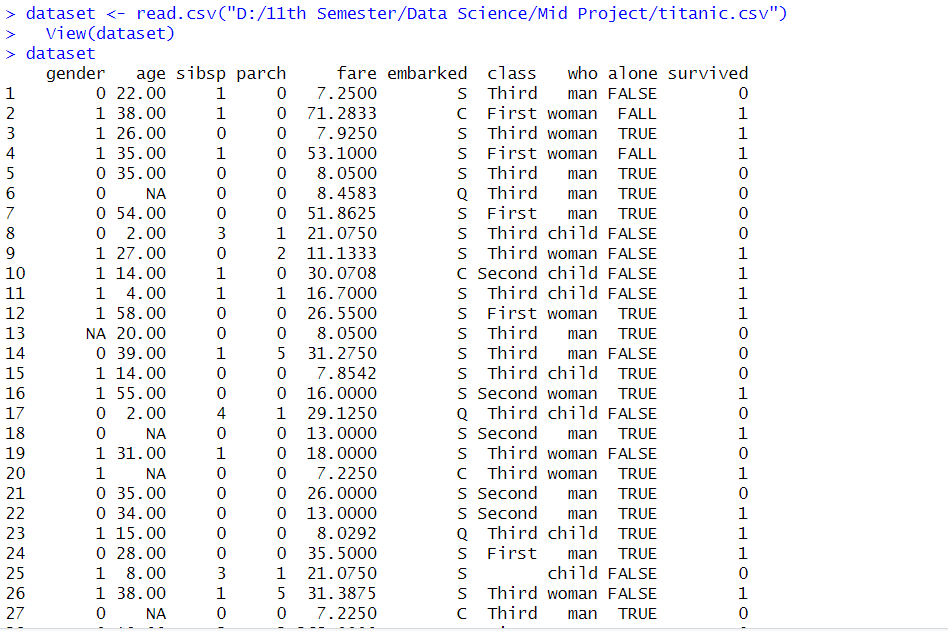
**Explanation:** Insert all of the data from the excel file first, and then save the document as a dataset file. then convert the dataset file's format to a CSV file. After importing my CSV file into RStudio, I add the following code.

**Code Segment:**

dataset <- read.csv("D:/11th Semester/Data Science/Mid Project/titanic.csv")

dataset

**Output:**

****

**Find the shape of the dataset:**

**Code Segment:**

nrow(dataset)

**Output:**



**Code Segment:**

ncol(dataset)

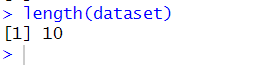
**Output:**



**Code Segment:**

length(dataset)

**Output:**

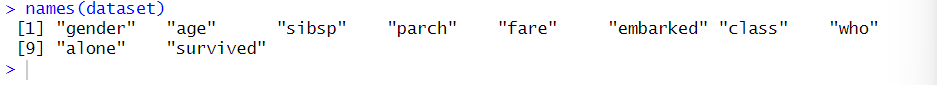


**Show the attributes names of the dataset:**

**Code Segment:**

names(dataset)

**Output:**

****

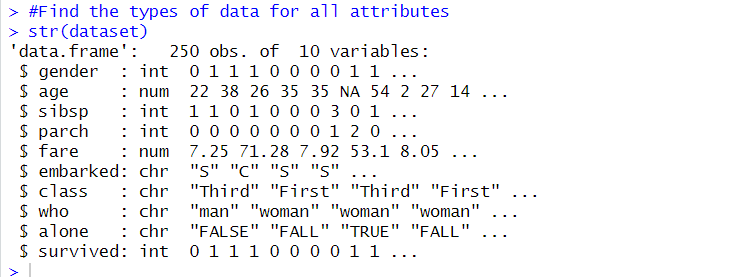
**Find the structure of the dataset:**

**Explanation:** The dataset's structure, including the variables, their data types, and the initial values, is shown using the str() function. This will provide us with a summary of the dataset.

**Code Segment:**

str(dataset)

**Output:**



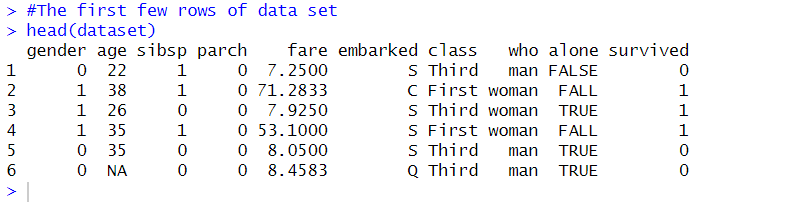
**The first few rows of dataset:**

**Explanation:** The first few rows of the dataset are shown by the head() function. This will allow us to understand the data and ensure that it was imported properly.

**Code Segment:**

head(dataset)

**Output:**

****

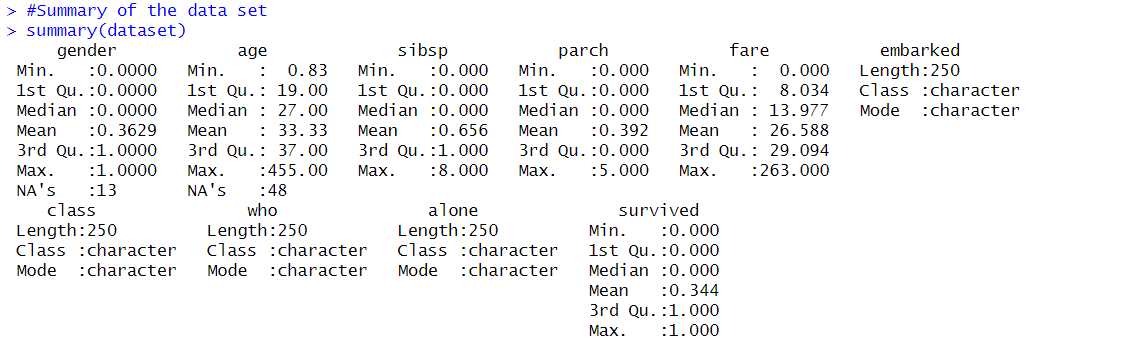
**Summary of the dataset:**

**Explanation:** For numerical variables in the dataset, summary statistics (count, mean, median, etc.) are provided using the summary() function. This will give us insights into the distribution and central tendencies of the variables.

**Code Segment:**

summary(dataset)

**Output:**

****

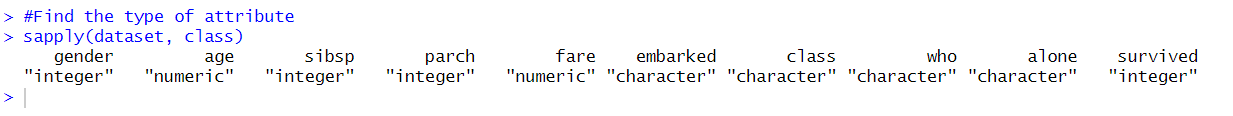
**Find the type of attribute:**

**Explanation:** We can identify the type of a column by using sapply.

**Code Segment:**

sapply(dataset, class)

**Output:**

****

**Measure of spread range and standard deviation:**

**Explanation**: The mean gives the average value, the median shows the middle value, and the mode shows the value which occurs most frequently for each feature. These measurements help me understand the dataset's distribution and usual values, which I can use to draw inferences and make comparisons as part of our research.

**Code Segment:**

**For Gender:**

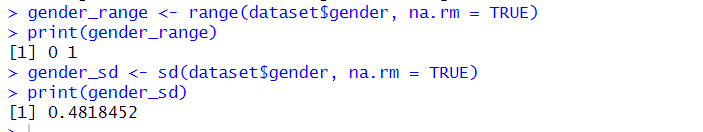
gender\_range <- range(dataset$gender, na.rm = TRUE)

print(gender\_range)

gender\_sd <- sd(dataset$gender, na.rm = TRUE)

print(gender\_sd)

**Output:**

****

**For Age:**

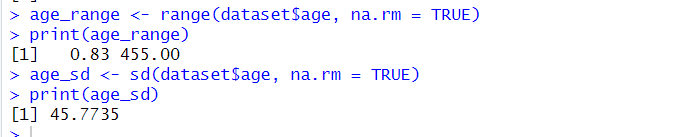
age\_range <- range(dataset$age, na.rm = TRUE)

print(age\_range)

age\_sd <- sd(dataset$age, na.rm = TRUE)

print(age\_sd)

**Output:**



**For sibsp:**

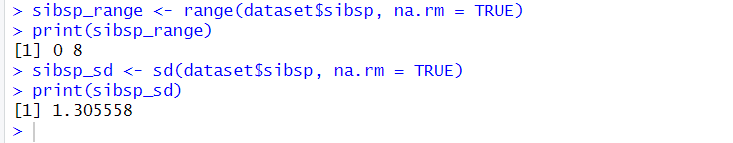
sibsp\_range <- range(dataset$sibsp, na.rm = TRUE)

print(sibsp\_range)

sibsp\_sd <- sd(dataset$sibsp, na.rm = TRUE)

print(sibsp\_sd)

**Output:**



**For parch:**

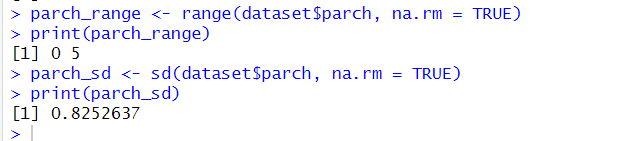
parch\_range <- range(dataset$parch, na.rm = TRUE)

print(parch\_range)

parch\_sd <- sd(dataset$parch, na.rm = TRUE)

print(parch\_sd)

**Output:**

****

**For fare:**

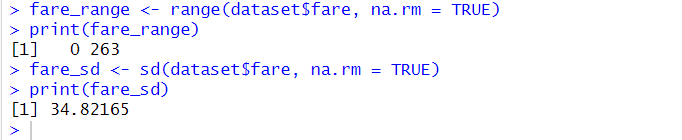
fare\_range <- range(dataset$fare, na.rm = TRUE)

print(fare\_range)

fare\_sd <- sd(dataset$fare, na.rm = TRUE)

print(fare\_sd)

**Output:**

****

**For Survived:**

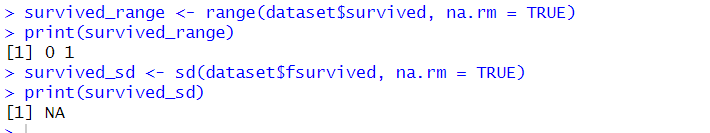
survived\_range <- range(dataset$survived, na.rm = TRUE)

print(survived\_range)

survived\_sd <- sd(dataset$fsurvived, na.rm = TRUE)

print(survived\_sd)

**Output:**



**Visualization:**

**Explanation:** Standard deviation measures the difference in a data set from the mean, with high deviation indicating wide data points and low deviation indicating closer points. Here I also create histogram for

**For age:**

**Code Segment:**

mean\_val <- 33.33

sd\_val <- 45.7735

age\_range <- c(0.83, 455)

age\_data <- runif(1000, min = age\_range[1], max = age\_range[2])

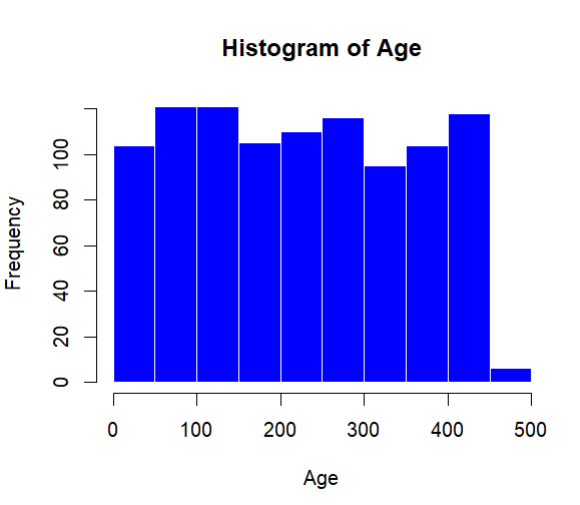
hist(age\_data,

main = "Histogram of Age",

xlab = "Age", ylab = "Frequency",

col = "blue", border = "white")

**Output:**



**For sibsp:**

mean\_val <- 0.656

sd\_val <- 1.305558

sibsp\_range <- c(0, 8)

sibsp\_data <- runif(1000, min = sibsp\_range[1], max = sibsp\_range[2])

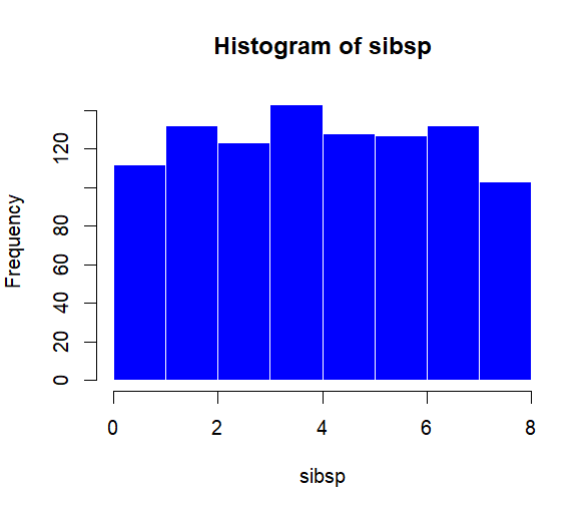
hist(sibsp\_data,

main = "Histogram of sibsp",

xlab = "sibsp", ylab = "Frequency",

col = "blue", border = "white")

**Output:**

****

**For parch:**

mean\_val <- 0.392

sd\_val <- 0.8252637

parch\_range <- c(0, 5)

parch\_data <- runif(1000, min = parch\_range[1], max = parch\_range[2])

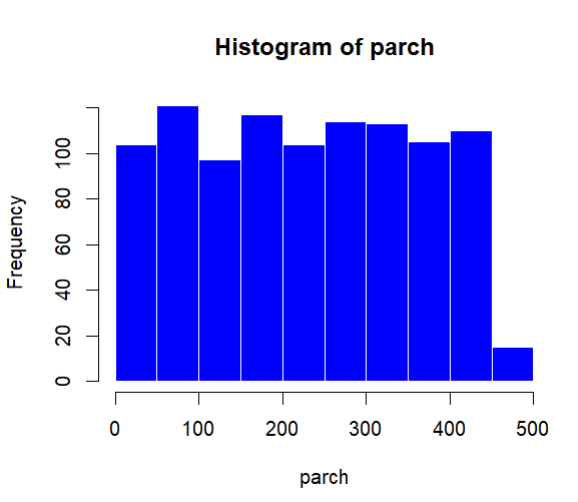
hist(age\_data,

main = "Histogram of parch",

xlab = "parch", ylab = "Frequency",

col = "blue", border = "white")

**Output:**

****

**For fare:**

mean\_val <- 26.588

sd\_val <- 34.82165

fare\_range <- c(0, 263)

fare\_data <- runif(1000, min =fare\_range[1], max = fare\_range[2])

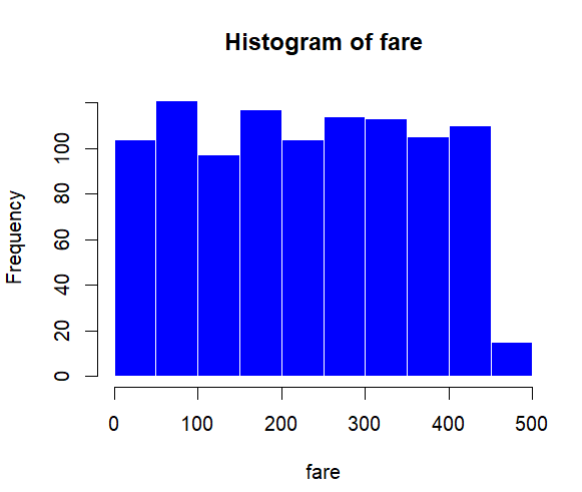
hist(age\_data,

main = "Histogram of fare",

xlab = "fare", ylab = "Frequency",

col = "blue", border = "white")

**Output:**

****

**For survived:**

mean\_val <- 0.344

sd\_val <- NA

survived\_range <- c(0, 1)

survived\_data <- runif(1000, min =survived\_range[1], max = survived\_range[2])

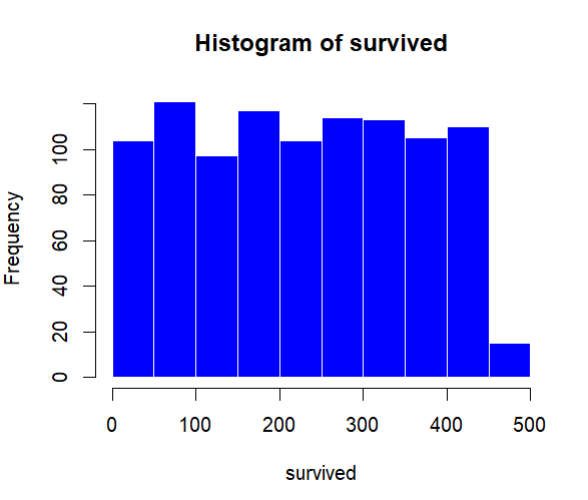
hist(age\_data,

main = "Histogram of survived",

xlab = "survived", ylab = "Frequency",

col = "blue", border = "white")

**Output:**

****

**Find the missing value for all column:**

**Explanation**: It's crucial to identify and handle missing values in a dataset since they can introduce biased and affect the accuracy of our research and findings. I can identify the missing values in every column using the following code.

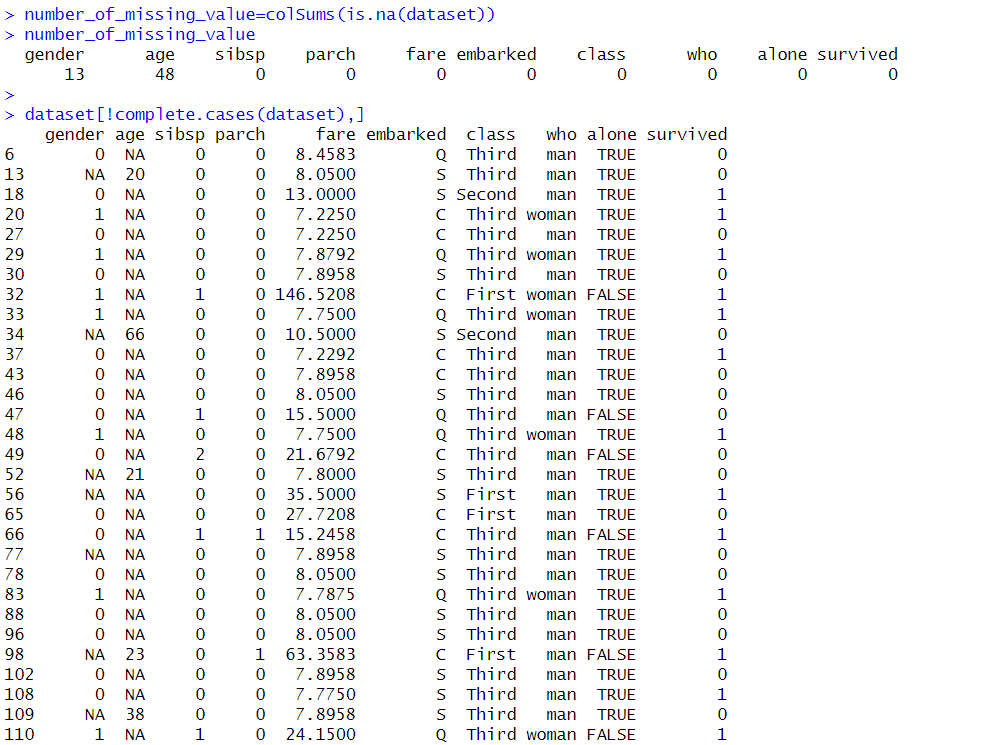
To extract the rows with missing values from the "dataset" dataset, use the code "dataset[!complete.cases(dataset),]".

**Code Segment:**

number\_of\_missing\_value=colSums(is.na(dataset))

number\_of\_missing\_value

**Output:**



**Missing value of Gender & Age:**

**Explanation:** I can identify which rows are missing values by using the code below.

**Code Segment:**

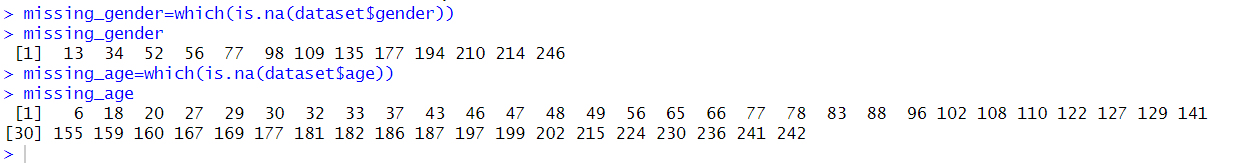
missing\_gender=which(is.na(dataset$gender))

missing\_gender

missing\_age=which(is.na(dataset$age))

missing\_age

**Output:**

****

**Most frequent value:**

**Explanation:** The gender column in our data set has an invalid value. We may extract the gender attribute's most frequent value using function and code.

find\_mode <- function(x) {

u <- unique(x)

tab <- tabulate(match(x, u))

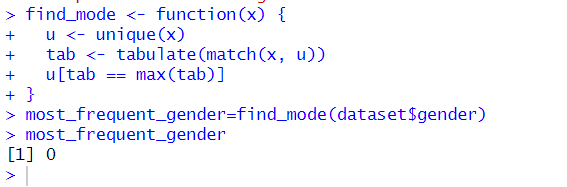
u[tab == max(tab)]

}

most\_frequent\_gender=find\_mode(dataset$gender)

most\_frequent\_gender

**Output:**

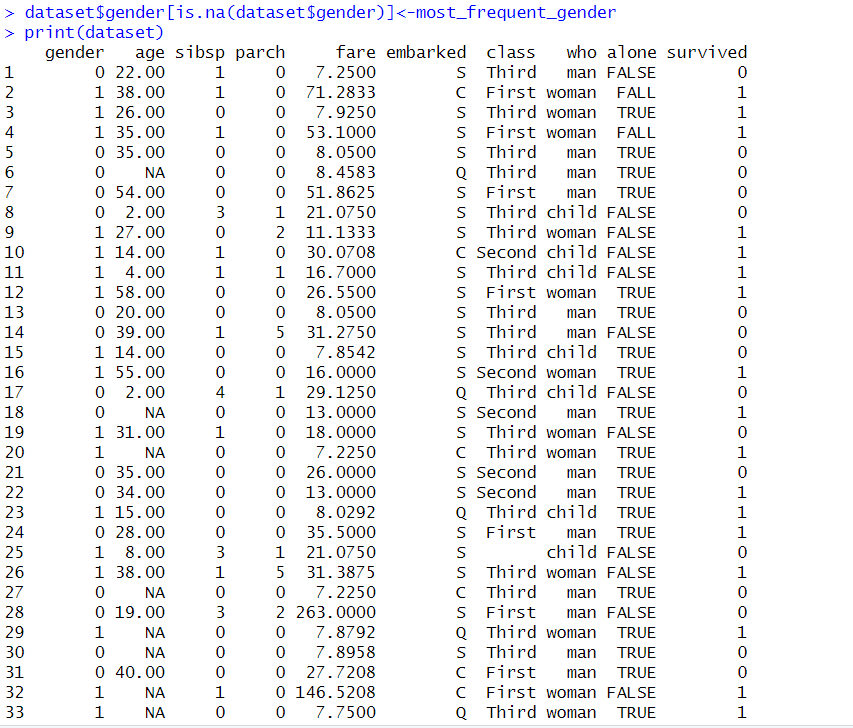
****

**Replacing missing value by most frequent value for gender attributes:**

dataset$gender[is.na(dataset$gender)]<-most\_frequent\_gender

print(dataset)

**Output:**



**Data Cleaning:**

**Explanation:** We may create a clean dataset by removing missing values, which will enable us to analyze our dataset more efficiently.

1. Rows with missing values should be deleted. We can remove the row of missing values by using the na.omit() function.It is a particular type of cleaning missing value.
2. Using the mean value, recover missing values.
3. Using the mode value, recover missing values.

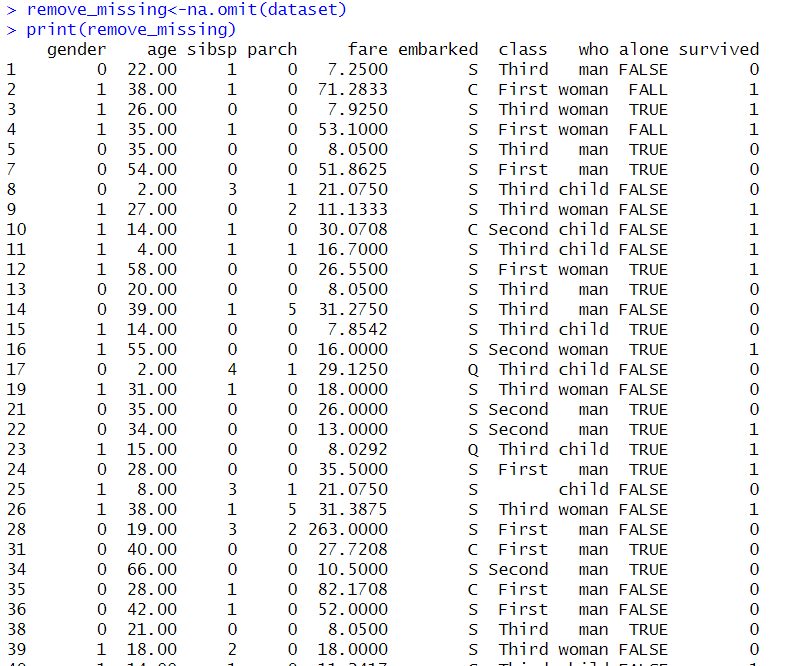
**Deleting row for clean data:**

**Code Segment:**

remove\_missing<-na.omit(dataset)

print(remove\_missing)

**Output:**

****

**Using Mean:**

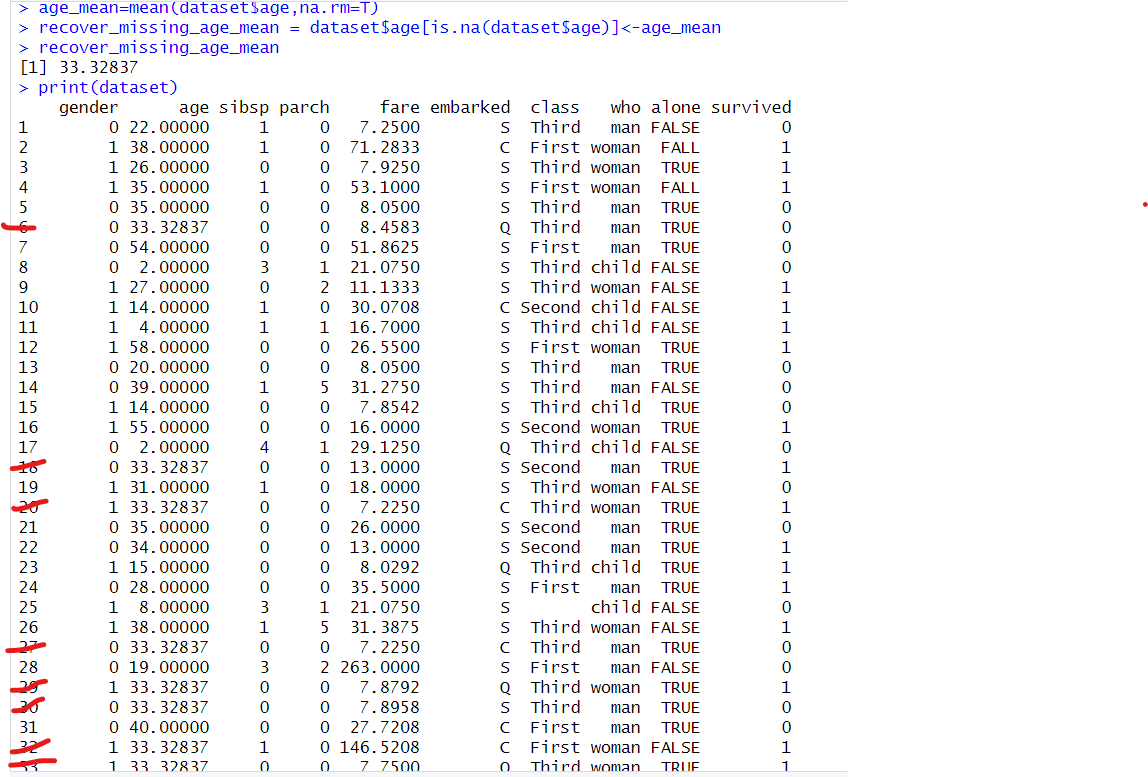
age\_mean=mean(dataset$age,na.rm=T)

recover\_missing\_age\_mean = dataset$age[is.na(dataset$age)]<-age\_mean

recover\_missing\_age\_mean

print(dataset)

**Output:**

****

**Using Mode:**

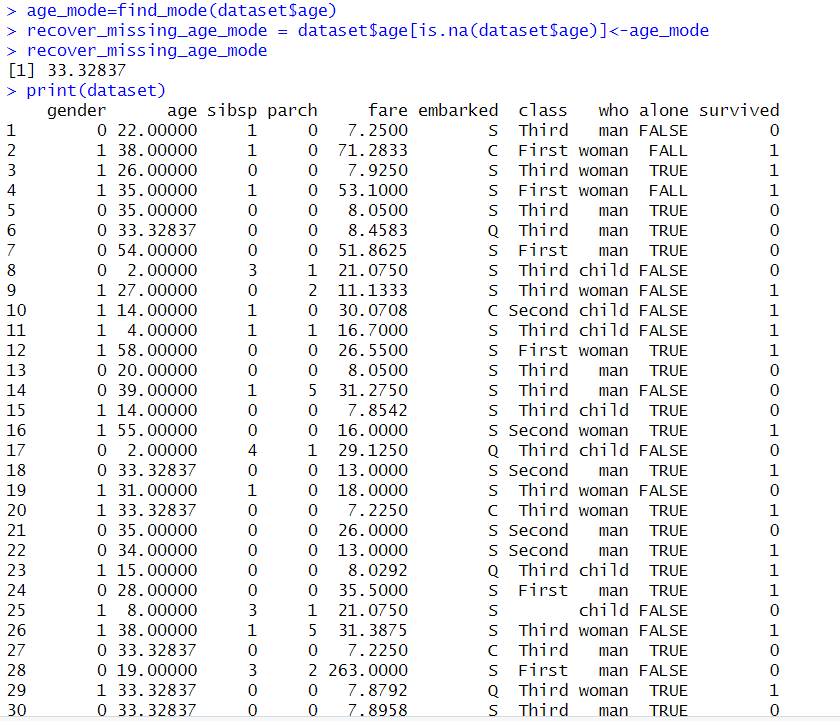
age\_mode=find\_mode(dataset$age)

recover\_missing\_age\_mode = dataset$age[is.na(dataset$age)]<-age\_mode

recover\_missing\_age\_mode

print(dataset)

**Output:**

****

**Annotate:**

**Explanation:** To improve data accuracy, interpretability, and analysis for better decision-making, I use annotations in this case. Here, I annotate embarked, class, who and alone .

**Code Segment:**

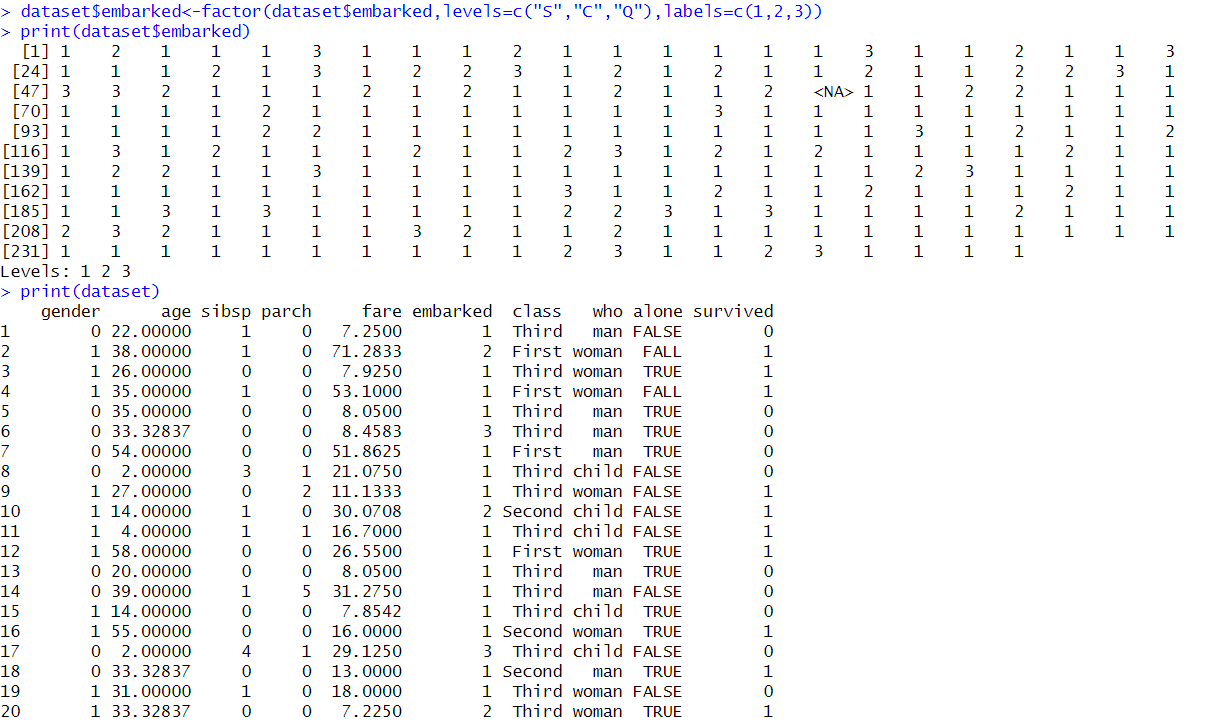
**For embarked:**

dataset$embarked<-factor(dataset$embarked,levels=c("S","C","Q"),labels=c(1,2,3))

print(dataset$embarked)

print(dataset)

**Output:**

****

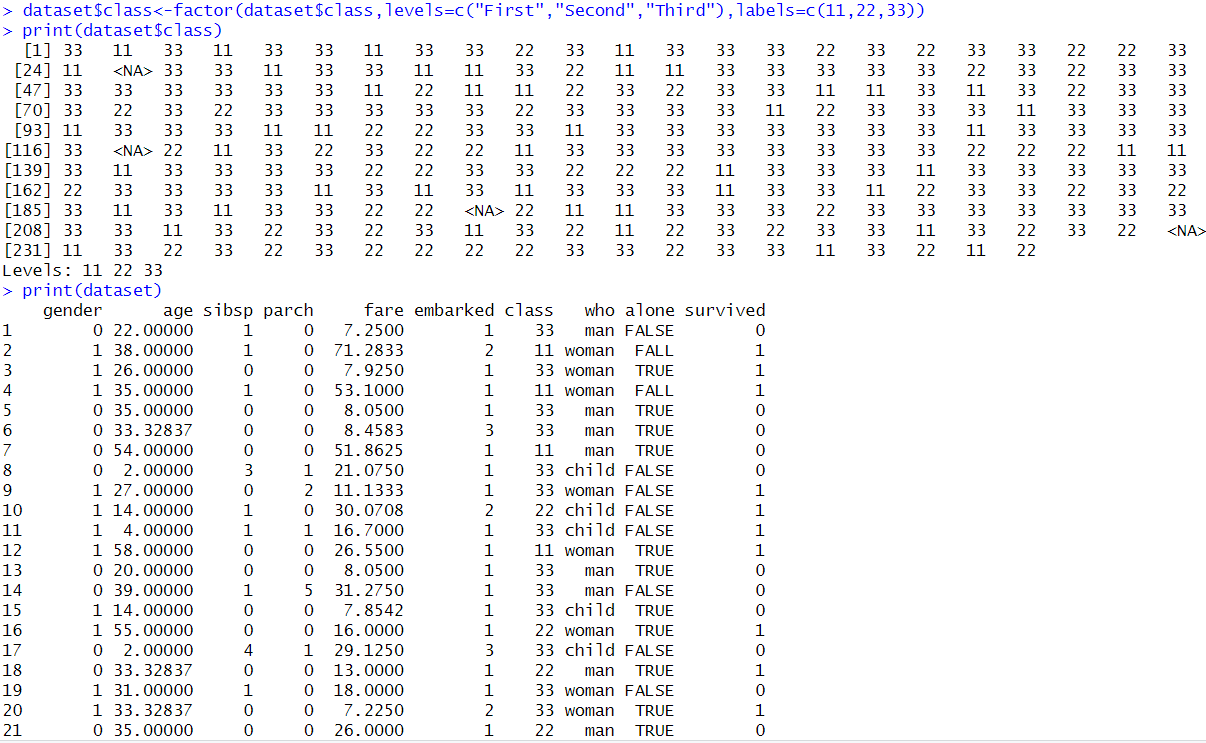
**For class:**

dataset$class<-factor(dataset$class,levels=c("First","Second","Third"),labels=c(11,22,33))

print(dataset$class)

print(dataset)

**Output:**

****

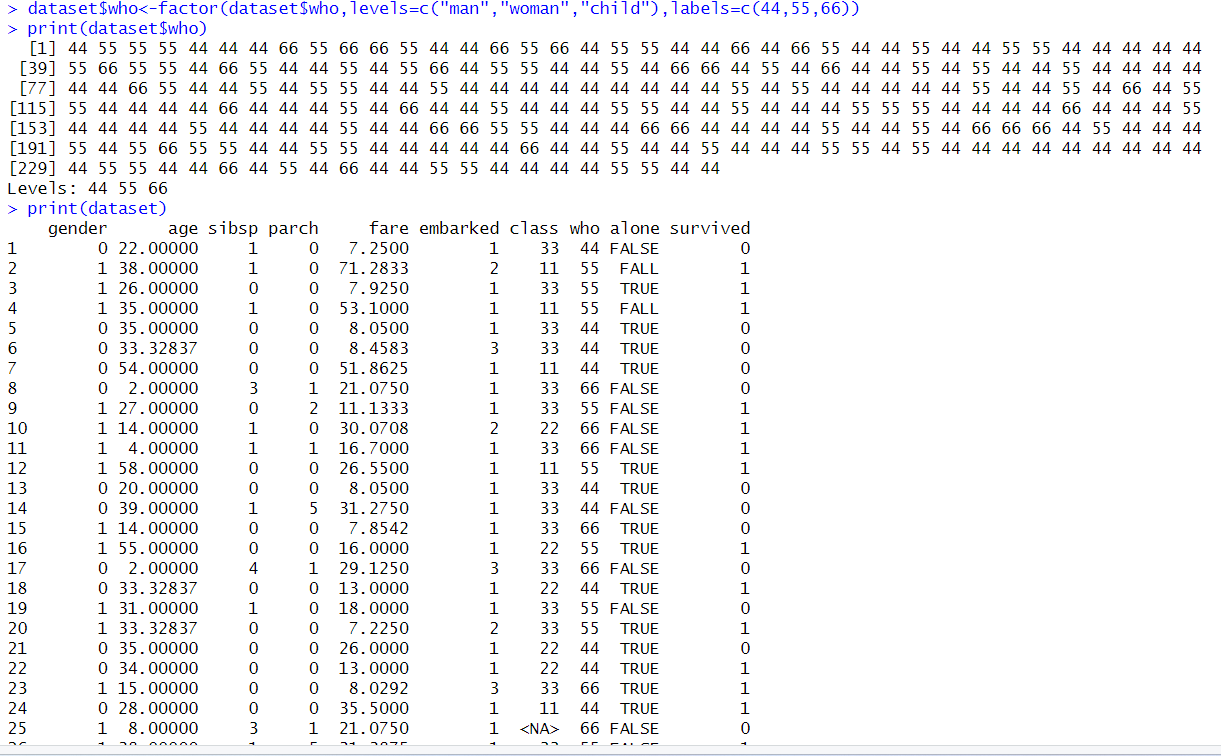
**For who:**

dataset$who<-factor(dataset$who,levels=c("man","woman","child"),labels=c(44,55,66))

print(dataset$who)

print(dataset)

**Output:**

****

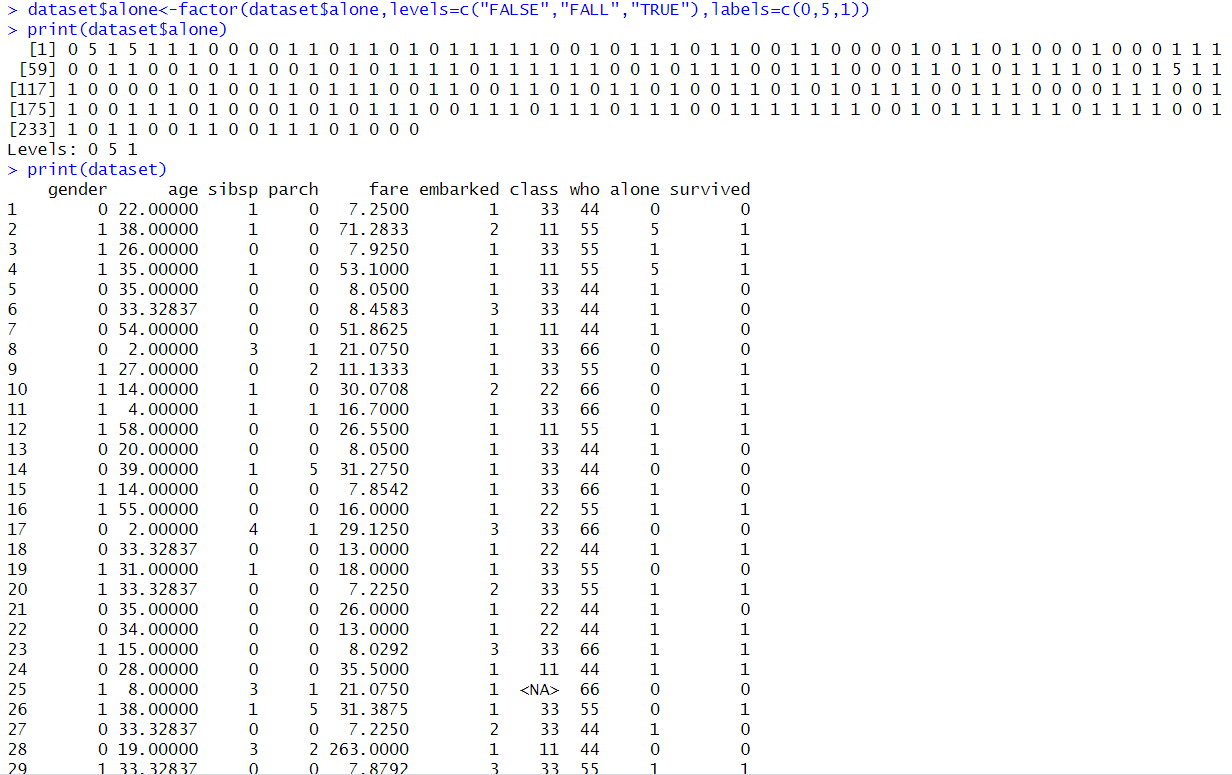
**For alone:**

dataset$alone<-factor(dataset$alone,levels=c("FALSE","FALL","TRUE"),labels=c(0,5,1))

print(dataset$alone)

print(dataset)

**Output:**

****

**Outlier:**

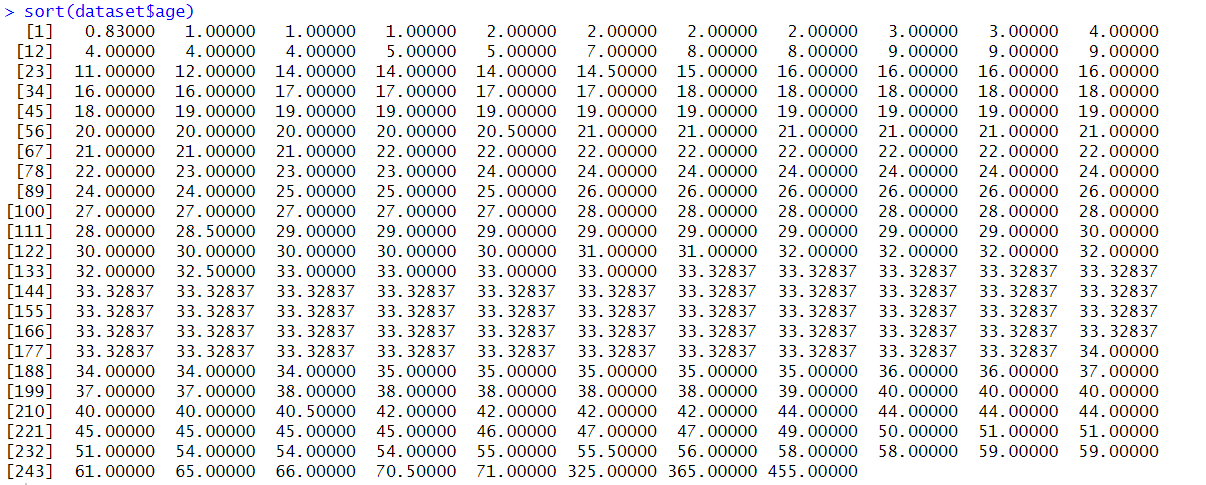
**Explanation**: A dataset's outliers can be utilized to spot problems with data quality, understand data distribution, identify deviations, and improve model performance.

**Code Segment:**

**For age:**

sort(dataset$age)

**Output:**

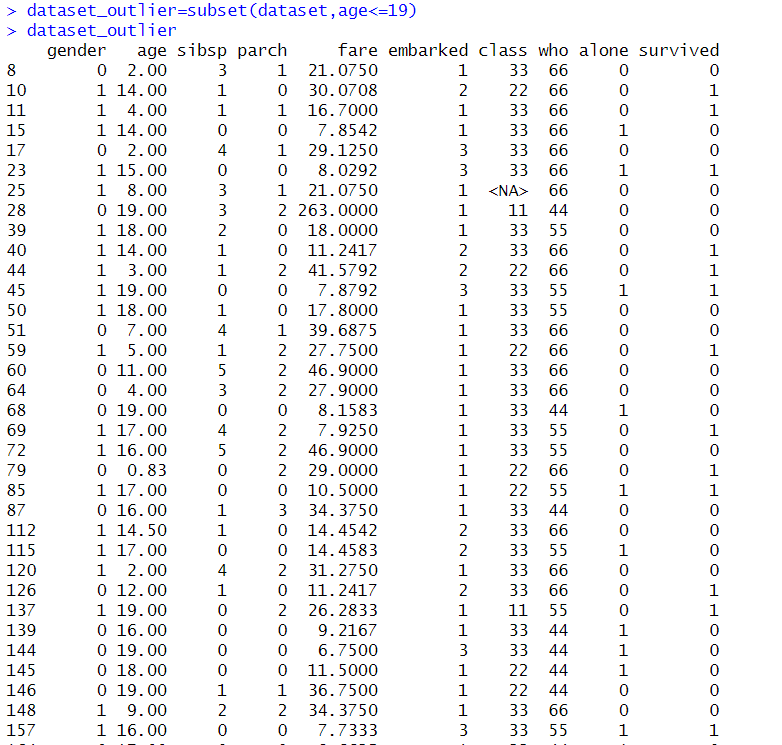
****

**Code Segment:**

dataset\_outlier=subset(dataset,age<=19)

dataset\_outlier

**Output:**

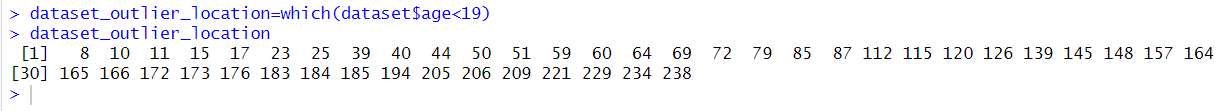
****

**Code Segment:**

dataset\_outlier\_location=which(dataset$age<19)

dataset\_outlier\_location

**Output:**

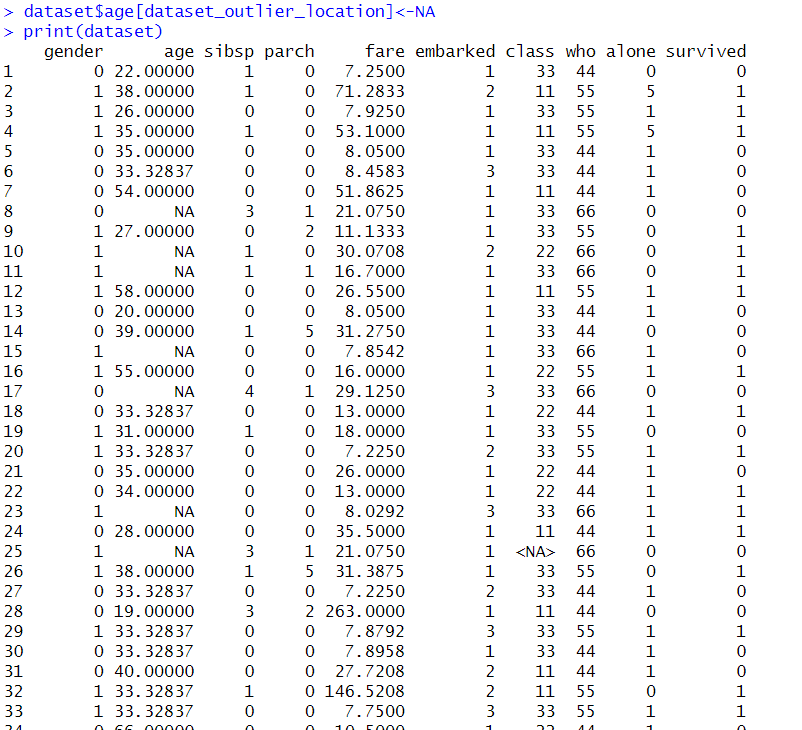


**Code Segment:**

dataset$age[dataset\_outlier\_location]<=NA

print(dataset)

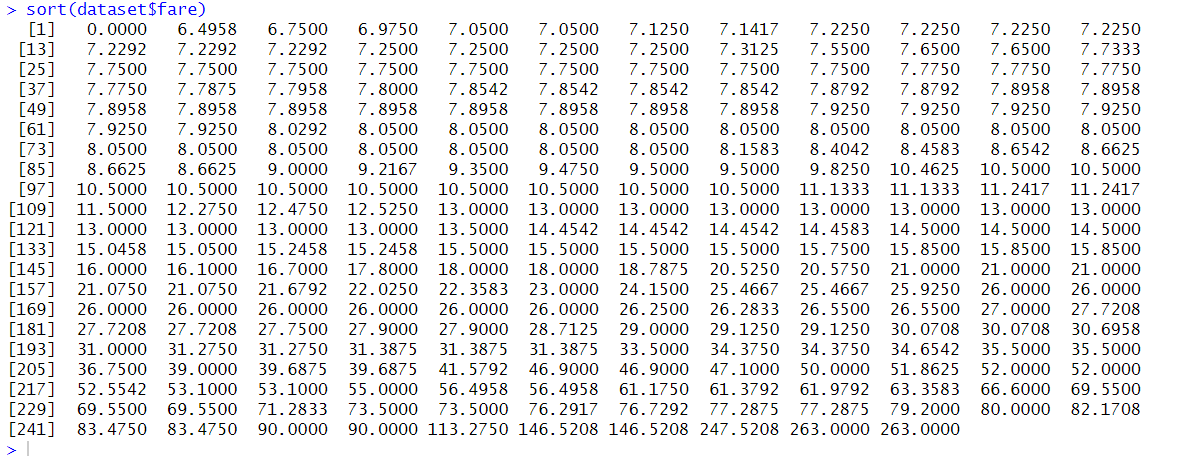
**Output:**

****

**For fare:**

sort(dataset$fare)

**Output:**

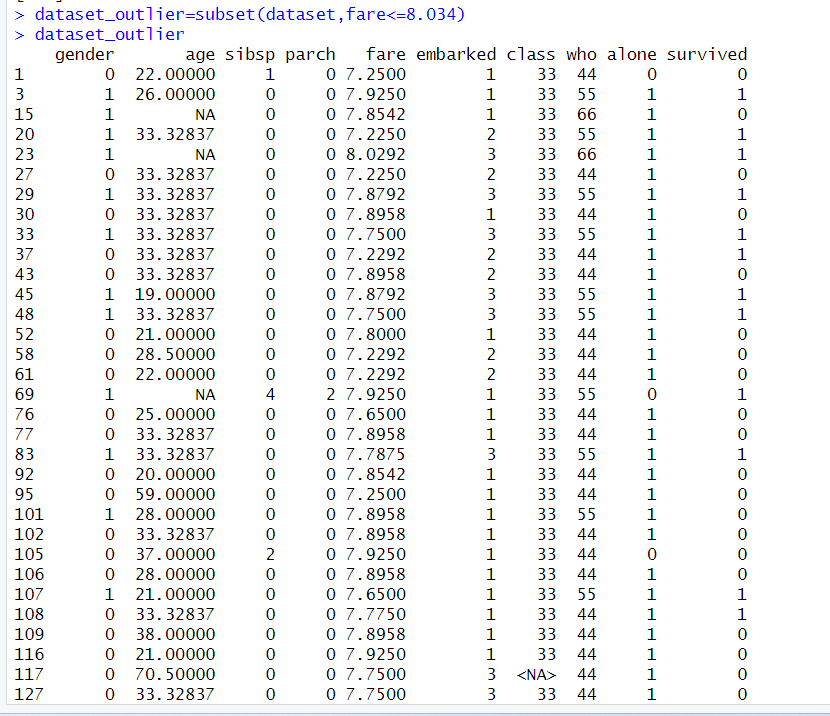


**Code Segment:**

dataset\_outlier=subset(dataset,fare<=8.034)

dataset\_outlier

**Output:**

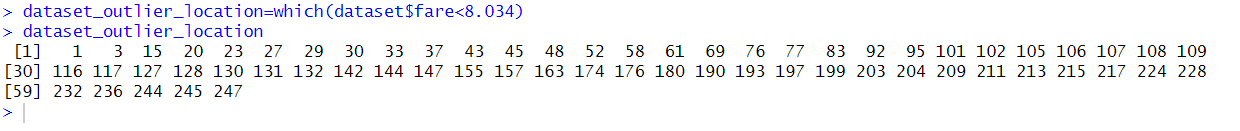
****

**Code Segment:**

dataset\_outlier\_location=which(dataset$fare<8.034)

dataset\_outlier\_location

**Output:**

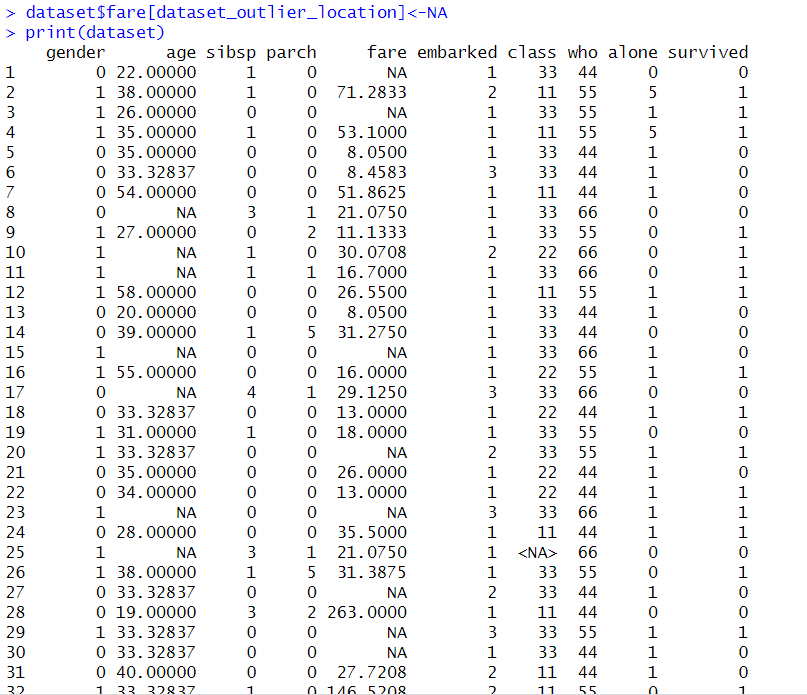
****

**Code Segment:**

dataset$fare[dataset\_outlier\_location]<-NA

print(dataset)

**Output:**

****

**Data Transformation:**

**Explanation**: As we already know, normalization, summarization, noise removal, smoothing, and data summarization are all processes in the data transformation process. I applied normalization to the data set we utilized.

**Normalization:**

**Explanation:** Normalization techniques have a favorable effect on the statistical distribution of the data since they enable us to reduce the size of the variables. I've standardized the columns in this data set to range from 1 to 5.

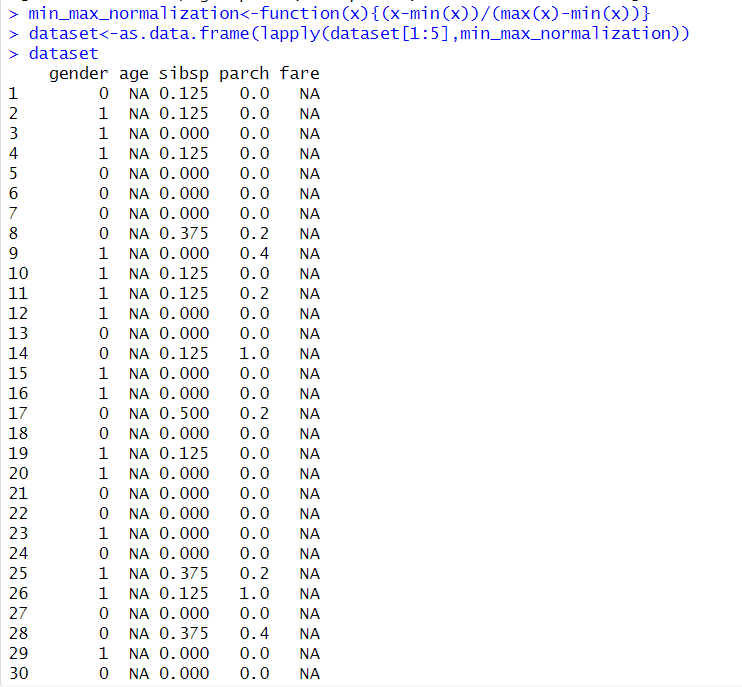
**Code Segment:**

min\_max\_normalization<-function(x){(x-min(x))/(max(x)-min(x))}

dataset<-as.data.frame(lapply(dataset[1:5],min\_max\_normalization))

dataset

**Output:**



**Invalid Value:**

**Explanation**: Invalid values in a dataset are used to represent missing or unknown data, ensuring data completeness and providing a standardized representation for missing information.

**Code Segment:**

**For who(Most Frequent Value):**

find\_mode <- function(x) {

u <- unique(x)

tab <- tabulate(match(x, u))

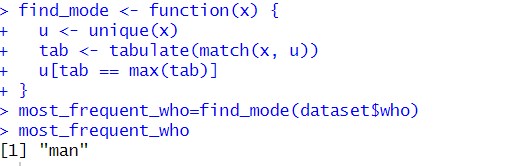
u[tab == max(tab)]

}

most\_frequent\_who=find\_mode(dataset$who)

most\_frequent\_who

**Output:**

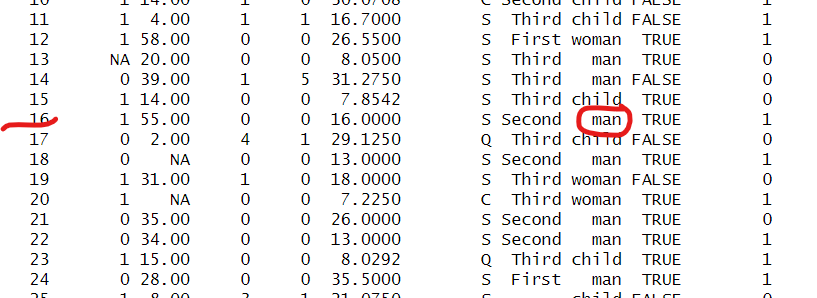
****

**Code Segment:**

dataset$who[16]<-most\_frequent\_who

print(dataset)

**Output:**

****