AMERICAN INTERNATIONAL UNIVERSITY-BANGLADESH

Faculty of Science and Technology

Assignment Cover Sheet

Assignment Title:	Midterm Project 1		
Assignment No:		Date of Submission:	26 April, 2025
Course Title:	Introduction to Data Science		
Course Code:	01812	Section:	A
Semester:	Spring 24-25	Course Teacher:	Abdus Salam

Declaration and Statement of Authorship:

1. I/we hold a copy of this Assignment/Case-Study, which can be produced if the original is lost/damaged.
2. This Assignment/Case-Study is my/our original work and no part of it has been copied from any other student's work or from any other source except where due acknowledgement is made.
3. No part of this Assignment/Case-Study has been written for me/us by any other person except where such collaborationhas been authorized by the concerned teacher and is clearly acknowledged in the assignment.
4. I/we have not previously submitted or currently submitting this work for any other course/unit.
5. This work may be reproduced, communicated, compared and archived for the purpose of detecting plagiarism.
6. I/we give permission for a copy of my/our marked work to be retained by the Faculty for review and comparison, including review by external examiners.
7. I/we understand thatPlagiarism is the presentation of the work, idea or creation of another person as though it is your own. It is a formofcheatingandisaveryseriousacademicoffencethatmayleadtoexpulsionfromtheUniversity. Plagiarized material can be drawn from, and presented in, written, graphic and visual form, including electronic data, and oral presentations. Plagiarism occurs when the origin of them arterial used is not appropriately cited.
8. I/we also understand that enabling plagiarism is the act of assisting or allowing another person to plagiarize or to copy my/our work.

* *Student(s) must complete all details except the faculty use part.*
** Please submit all assignments to your course teacher or the office of the concerned teacher.

Group Name/No. **05**

No	Name	ID	Program	Signature
1	Mithi Zaman	21-44753-1	BSc.CSE	
2	Zannatul Ferdows Wafi	21-45924-3	BSc.CSE	
3	Tasfi Islam	21-45504-3	BSc.CSE	
4	Nusrat Jannat	21-45795-3	BSc.CSE	

Faculty use only

FACULTYCOMMENTS	**Marks Obtained**	
	Total Marks	

Dataset Description:

This is a dataset named UCI Heart Disease Data which contains medical reports of patients with 14 key attributes such as age, sex, chest pain type, resting blood pressure, serum cholesterol, fasting blood sugar, resting electrocardiographic results, maximum heart rate achieved, exercise-induced angina, old peak — ST depression induced by exercise relative to rest, the slope of the peak exercise ST segment, number of major vessels and Thalassemia. One of the main tasks on this dataset is to predict whether a patient has heart disease based on their attributes. Another is the experimental task of diagnosing the patient and learning different insights from the dataset that could help better understand the issue.

The dataset containing following attributes:

- **id**: Patient ID — used to uniquely identify each record.
- **age:** Patient's age — helps find age-related heart risk.
- **sex:** Gender (Male/Female) — used to study heart disease patterns between genders.
- **dataset:** Source dataset name — useful for tracking where the data came from.
- **cp (Chest Pain Type):** Type of chest pain — helps detect types of heart issues.
- **trestbps:** Resting blood pressure — high pressure can signal heart problems
- **chol:** Cholesterol level — higher cholesterol increases heart disease risk
- **fbs (Fasting Blood Sugar):** Blood sugar >120 mg/dl (True/False) — diabetes is a heart risk factor.
- **restecg (Resting ECG Result):** Heart's electrical activity — detects heart abnormalities.
- **thalch:** Maximum heart rate during exercise — checks heart function under stress.
- **exang (Exercise Induced Angina):** Chest pain during exercise (True/False) — shows exercise-related heart problems.
- **oldpeak:** ST depression — measures heart stress from exercise.
- **slope:** Slope of ST segment — identifies heart function after exercise.
- **ca:** Number of blocked blood vessels — shows severity of heart disease
- **thal:** Blood disorder test result — related to heart health risks.
- **num:** Heart disease diagnosis (0 = No disease, 1 = Disease) — the main target for prediction.

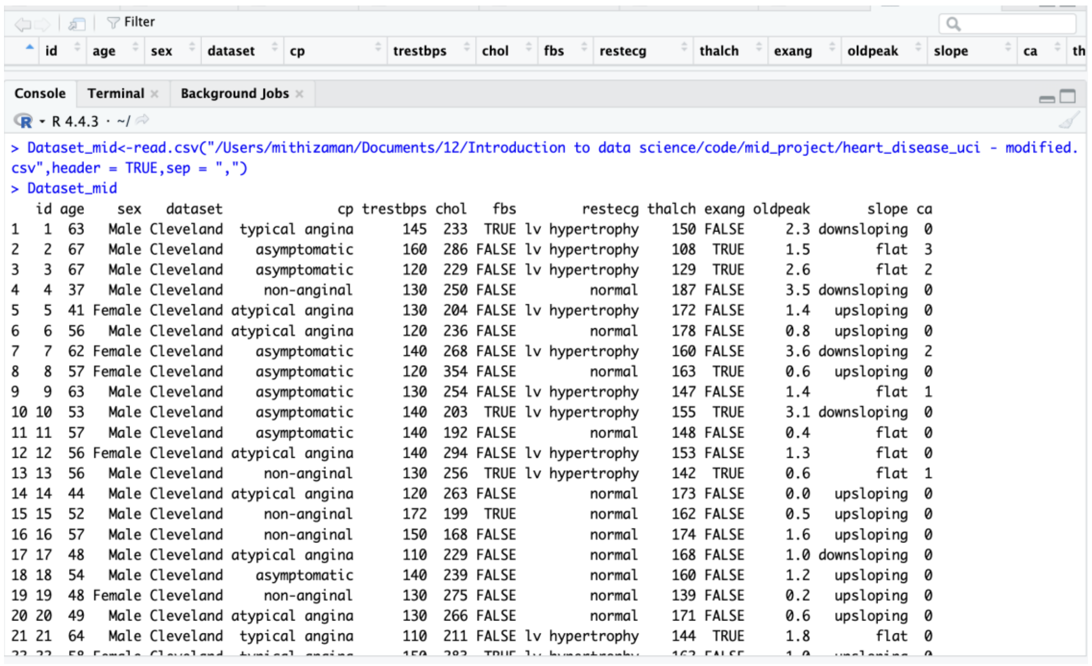
Data Pre-processing:

1. Importing the Dataset

To begin the data preprocessing process, we first need to import the dataset into R to begin the data preprocessing process. The dataset file is named heart_disease_uci-modified.csv and is in the working directory. We use the read.csv () function to read the file and store it as a data frame named Dataset_mid.

CODE:

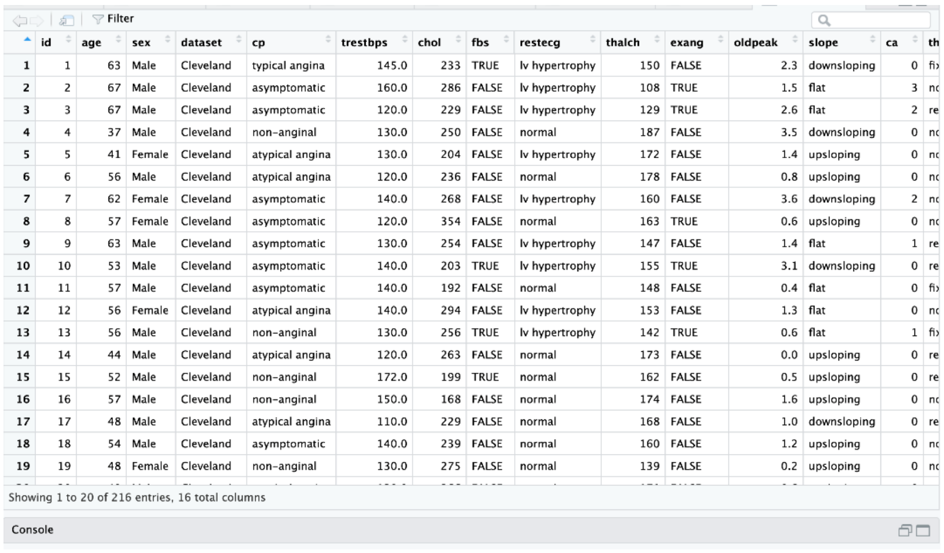
```
Dataset_mid<-read.csv("/Users/mithizaman/Documents/12/Introduction to data
science/code/mid_project/heart_disease_uci - modified.csv",header = TRUE,sep = ",")
Dataset_mid
```



```
> Dataset_mid<-read.csv("/Users/mithizaman/Documents/12/Introduction to data science/code/mid_project/heart_disease_uci - modified.
csv",header = TRUE,sep = ",")
> Dataset_mid
   id age    sex  dataset              cp trestbps chol   fbs        restecg thalch exang oldpeak       slope ca
1   1  63   Male Cleveland  typical angina      145  233  TRUE lv hypertrophy    150 FALSE     2.3 downsloping  0
2   2  67   Male Cleveland    asymptomatic      160  286 FALSE lv hypertrophy    108  TRUE     1.5        flat  3
3   3  67   Male Cleveland    asymptomatic      120  229 FALSE lv hypertrophy    129  TRUE     2.6        flat  2
4   4  37   Male Cleveland     non-anginal      130  250 FALSE         normal    187 FALSE     3.5 downsloping  0
5   5  41 Female Cleveland atypical angina      130  204 FALSE lv hypertrophy    172 FALSE     1.4   upsloping  0
6   6  56   Male Cleveland atypical angina      120  236 FALSE         normal    178 FALSE     0.8   upsloping  0
7   7  62 Female Cleveland    asymptomatic      140  268 FALSE lv hypertrophy    160 FALSE     3.6 downsloping  2
8   8  57 Female Cleveland    asymptomatic      120  354 FALSE         normal    163  TRUE     0.6   upsloping  0
9   9  63   Male Cleveland    asymptomatic      130  254 FALSE lv hypertrophy    147 FALSE     1.4        flat  1
10 10  53   Male Cleveland    asymptomatic      140  203  TRUE lv hypertrophy    155  TRUE     3.1 downsloping  0
11 11  57   Male Cleveland    asymptomatic      140  192 FALSE         normal    148 FALSE     0.4        flat  0
12 12  56 Female Cleveland atypical angina      140  294 FALSE lv hypertrophy    153 FALSE     1.3        flat  0
13 13  56   Male Cleveland     non-anginal      130  256  TRUE lv hypertrophy    142  TRUE     0.6        flat  1
14 14  44   Male Cleveland atypical angina      120  263 FALSE         normal    173 FALSE     0.0   upsloping  0
15 15  52   Male Cleveland     non-anginal      172  199  TRUE         normal    162 FALSE     0.5   upsloping  0
16 16  57   Male Cleveland     non-anginal      150  168 FALSE         normal    174 FALSE     1.6   upsloping  0
17 17  48   Male Cleveland atypical angina      110  229 FALSE         normal    168 FALSE     1.0 downsloping  0
18 18  54   Male Cleveland    asymptomatic      140  239 FALSE         normal    160 FALSE     1.2   upsloping  0
19 19  48 Female Cleveland     non-anginal      130  275 FALSE         normal    139 FALSE     0.2   upsloping  0
20 20  49   Male Cleveland atypical angina      130  266 FALSE         normal    171 FALSE     0.6   upsloping  0
21 21  64   Male Cleveland  typical angina      110  211 FALSE lv hypertrophy    144  TRUE     1.8        flat  0
```

OUTPUT:

This is the imported Dataset:



Filter

	id	age	sex	dataset	cp	trestbps	chol	fbs	restecg	thalch	exang	oldpeak	slope	ca	th
1	1	63	Male	Cleveland	typical angina	145.0	233	TRUE	lv hypertrophy	150	FALSE	2.3	downsloping	0	fi
2	2	67	Male	Cleveland	asymptomatic	160.0	286	FALSE	lv hypertrophy	108	TRUE	1.5	flat	3	nc
3	3	67	Male	Cleveland	asymptomatic	120.0	229	FALSE	lv hypertrophy	129	TRUE	2.6	flat	2	re
4	4	37	Male	Cleveland	non-anginal	130.0	250	FALSE	normal	187	FALSE	3.5	downsloping	0	nc
5	5	41	Female	Cleveland	atypical angina	130.0	204	FALSE	lv hypertrophy	172	FALSE	1.4	upsloping	0	nc
6	6	56	Male	Cleveland	atypical angina	120.0	236	FALSE	normal	178	FALSE	0.8	upsloping	0	nc
7	7	62	Female	Cleveland	asymptomatic	140.0	268	FALSE	lv hypertrophy	160	FALSE	3.6	downsloping	2	nc
8	8	57	Female	Cleveland	asymptomatic	120.0	354	FALSE	normal	163	TRUE	0.6	upsloping	0	nc
9	9	63	Male	Cleveland	asymptomatic	130.0	254	FALSE	lv hypertrophy	147	FALSE	1.4	flat	1	re
10	10	53	Male	Cleveland	asymptomatic	140.0	203	TRUE	lv hypertrophy	155	TRUE	3.1	downsloping	0	re
11	11	57	Male	Cleveland	asymptomatic	140.0	192	FALSE	normal	148	FALSE	0.4	flat	0	fi
12	12	56	Female	Cleveland	atypical angina	140.0	294	FALSE	lv hypertrophy	153	FALSE	1.3	flat	0	nc
13	13	56	Male	Cleveland	non-anginal	130.0	256	TRUE	lv hypertrophy	142	TRUE	0.6	flat	1	fi
14	14	44	Male	Cleveland	atypical angina	120.0	263	FALSE	normal	173	FALSE	0.0	upsloping	0	re
15	15	52	Male	Cleveland	non-anginal	172.0	199	TRUE	normal	162	FALSE	0.5	upsloping	0	re
16	16	57	Male	Cleveland	non-anginal	150.0	168	FALSE	normal	174	FALSE	1.6	upsloping	0	nc
17	17	48	Male	Cleveland	atypical angina	110.0	229	FALSE	normal	168	FALSE	1.0	downsloping	0	re
18	18	54	Male	Cleveland	asymptomatic	140.0	239	FALSE	normal	160	FALSE	1.2	upsloping	0	nc
19	19	48	Female	Cleveland	non-anginal	130.0	275	FALSE	normal	139	FALSE	0.2	upsloping	0	nc

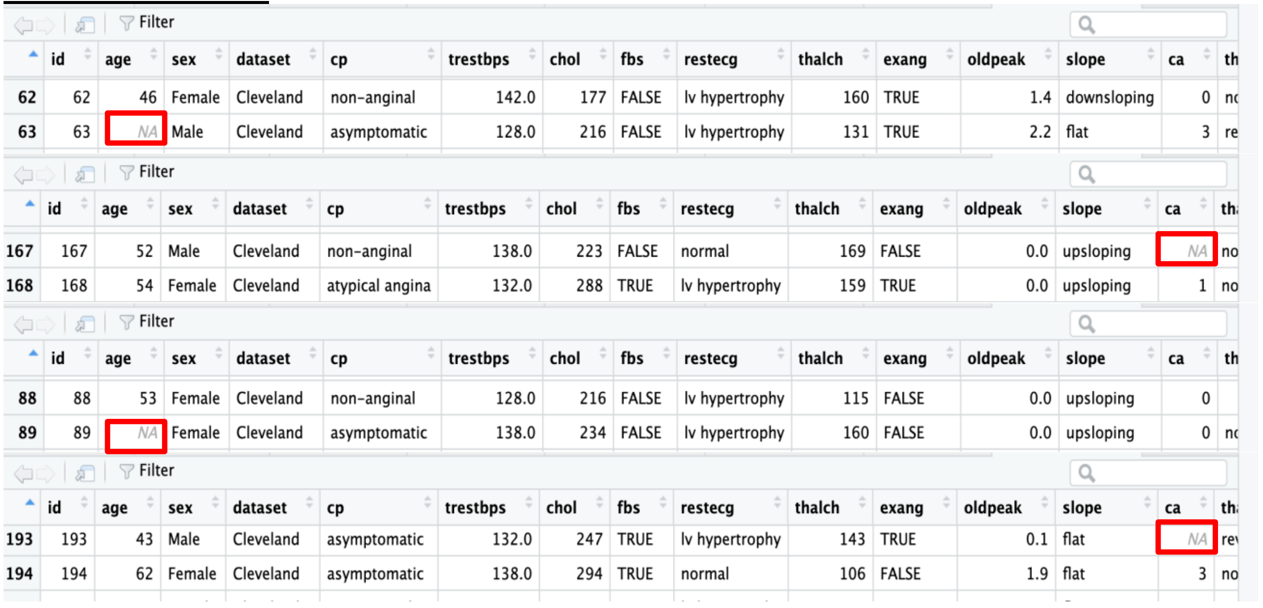
Showing 1 to 20 of 216 entries, 16 total columns

Console

2. **Identifying Missing Data:**

To check for missing values (NA) in specific columns such as Age[2], ca[2]. We need to use the given code to find the missing value.

This is the dataset.:



Filter

	id	age	sex	dataset	cp	trestbps	chol	fbs	restecg	thalch	exang	oldpeak	slope	ca	th
62	62	46	Female	Cleveland	non-anginal	142.0	177	FALSE	lv hypertrophy	160	TRUE	1.4	downsloping	0	nc
63	63	NA	Male	Cleveland	asymptomatic	128.0	216	FALSE	lv hypertrophy	131	TRUE	2.2	flat	3	re

Filter

	id	age	sex	dataset	cp	trestbps	chol	fbs	restecg	thalch	exang	oldpeak	slope	ca	th
167	167	52	Male	Cleveland	non-anginal	138.0	223	FALSE	normal	169	FALSE	0.0	upsloping	NA	no
168	168	54	Female	Cleveland	atypical angina	132.0	288	TRUE	lv hypertrophy	159	TRUE	0.0	upsloping	1	no

Filter

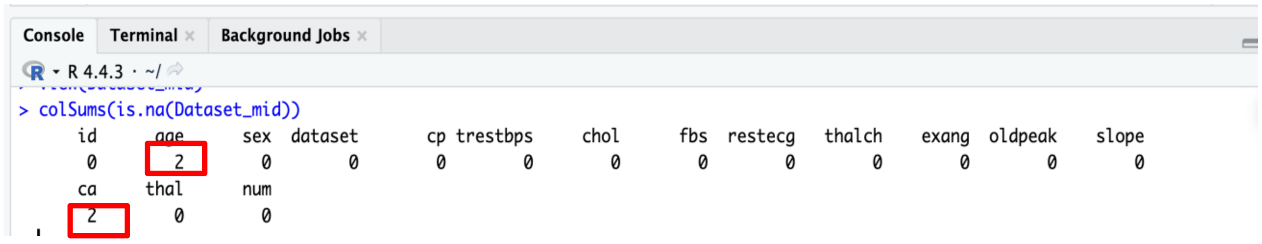
	id	age	sex	dataset	cp	trestbps	chol	fbs	restecg	thalch	exang	oldpeak	slope	ca	th
88	88	53	Female	Cleveland	non-anginal	128.0	216	FALSE	lv hypertrophy	115	FALSE	0.0	upsloping	0	
89	89	NA	Female	Cleveland	asymptomatic	138.0	234	FALSE	lv hypertrophy	160	FALSE	0.0	upsloping	0	nc

Filter

	id	age	sex	dataset	cp	trestbps	chol	fbs	restecg	thalch	exang	oldpeak	slope	ca	th
193	193	43	Male	Cleveland	asymptomatic	132.0	247	TRUE	lv hypertrophy	143	TRUE	0.1	flat	NA	re
194	194	62	Female	Cleveland	asymptomatic	138.0	294	TRUE	normal	106	FALSE	1.9	flat	3	no

CODE:

```
colSums(is.na(Dataset_mid))
```



```
> colSums(is.na(Dataset_mid))
      id      age      sex  dataset       cp trestbps     chol      fbs  restecg   thalch    exang  oldpeak    slope 
       0        2        0        0        0        0        0        0        0        0        0        0        0 
      ca     thal      num 
       2        0        0 
```

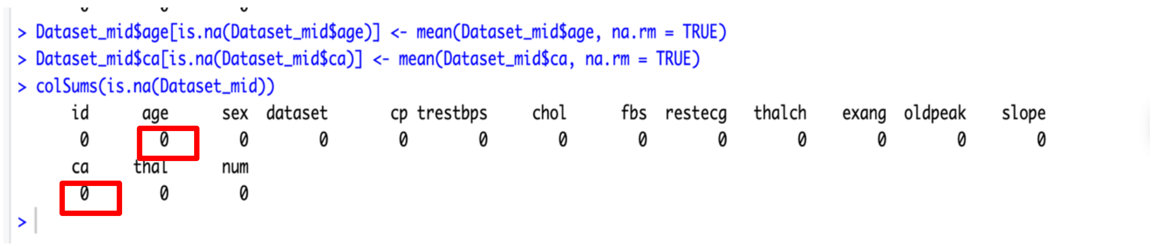
3. Handling missing value:

Missing values can be handled using two approaches.

- Replacing them with the most frequent value the average (mean) value.
  In the age and ca columns, we handle missing values by replacing them with the mean.

OUTPUT CODE:

```
Dataset_mid$age[is.na(Dataset_mid$age)] <- mean(Dataset_mid$age, na.rm = TRUE)
Dataset_mid$ca[is.na(Dataset_mid$ca)] <- mean(Dataset_mid$ca, na.rm = TRUE)
colSums(is.na(Dataset_mid))
```



```
> Dataset_mid$age[is.na(Dataset_mid$age)] <- mean(Dataset_mid$age, na.rm = TRUE)
> Dataset_mid$ca[is.na(Dataset_mid$ca)] <- mean(Dataset_mid$ca, na.rm = TRUE)
> colSums(is.na(Dataset_mid))
      id      age      sex  dataset       cp trestbps     chol      fbs  restecg   thalch    exang  oldpeak    slope 
       0        0        0        0        0        0        0        0        0        0        0        0        0 
      ca     thal      num 
       0        0        0 
> 
```

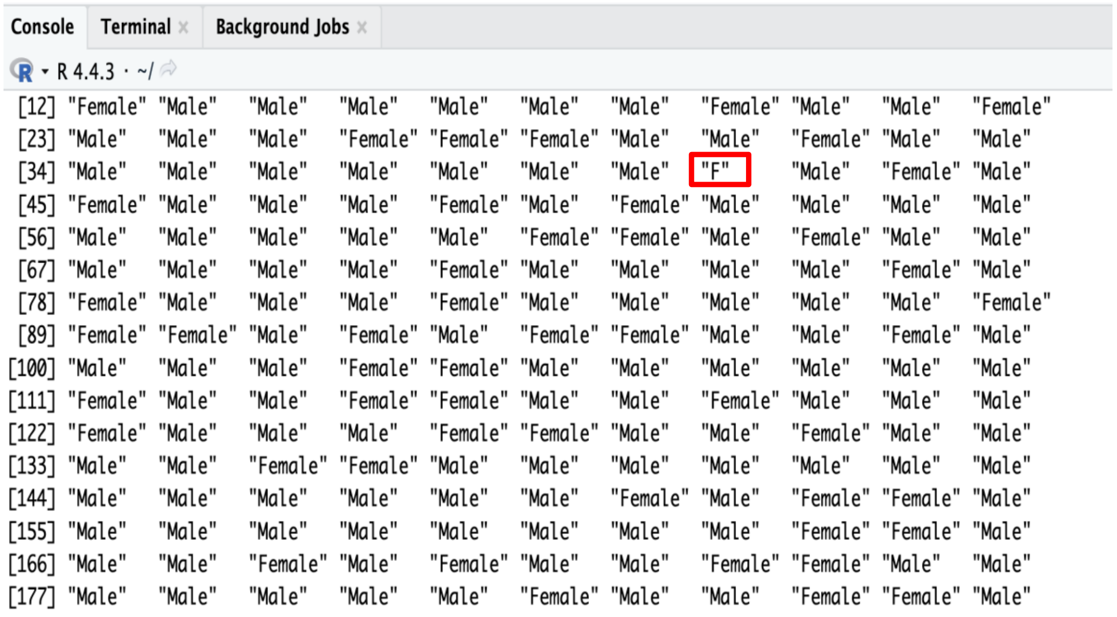
In the age and ca columns, we handle missing values by replacing them with the mean.

4. Handling Invalid Value :

The dataset contains an invalid value in the sex column. We need to fix or remove the invalid value. The following is the incorrect value we found:

CODE:

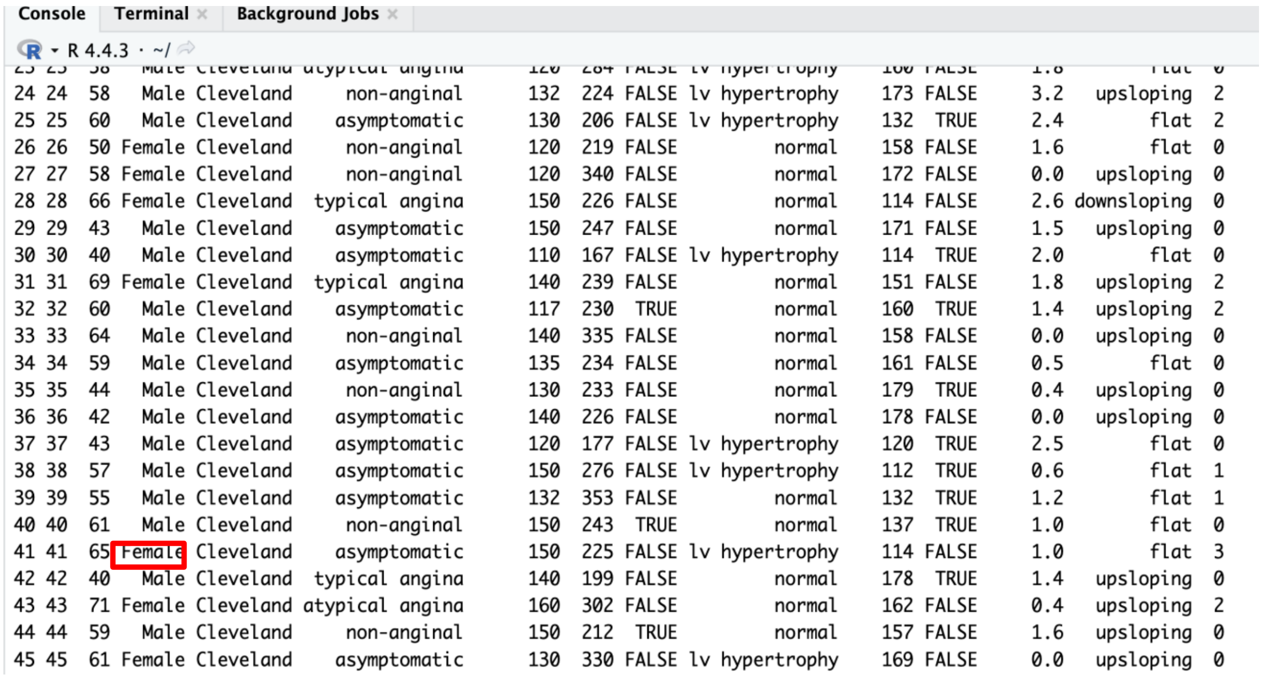
```
Dataset_mid$sex
```



```
 [12] "Female" "Male"   "Male"   "Male"   "Male"   "Male"   "Male"   "Female" "Male"   "Male"   "Female"
 [23] "Male"   "Male"   "Male"   "Female" "Female" "Female" "Male"   "Male"   "Female" "Male"   "Male"  
 [34] "Male"   "Male"   "Male"   "Male"   "Male"   "Male"   "Male"   "F"      "Male"   "Female" "Male"  
 [45] "Female" "Male"   "Male"   "Male"   "Female" "Male"   "Female" "Male"   "Male"   "Male"   "Male"  
 [56] "Male"   "Male"   "Male"   "Male"   "Male"   "Female" "Female" "Male"   "Female" "Male"   "Male"  
 [67] "Male"   "Male"   "Male"   "Male"   "Female" "Male"   "Male"   "Male"   "Male"   "Female" "Male"  
 [78] "Female" "Male"   "Male"   "Male"   "Female" "Male"   "Male"   "Male"   "Male"   "Male"   "Female"
 [89] "Female" "Female" "Male"   "Female" "Male"   "Female" "Female" "Male"   "Male"   "Female" "Male"  
[100] "Male"   "Male"   "Male"   "Female" "Female" "Male"   "Male"   "Male"   "Male"   "Male"   "Male"  
[111] "Female" "Male"   "Male"   "Female" "Female" "Male"   "Male"   "Female" "Male"   "Male"   "Male"  
[122] "Female" "Male"   "Male"   "Male"   "Female" "Female" "Male"   "Male"   "Female" "Male"   "Male"  
[133] "Male"   "Male"   "Female" "Female" "Male"   "Male"   "Male"   "Male"   "Male"   "Male"   "Male"  
[144] "Male"   "Male"   "Male"   "Male"   "Male"   "Male"   "Female" "Male"   "Female" "Female" "Male"  
[155] "Male"   "Male"   "Male"   "Male"   "Male"   "Male"   "Male"   "Male"   "Female" "Female" "Male"  
[166] "Male"   "Male"   "Female" "Male"   "Female" "Male"   "Male"   "Female" "Female" "Male"   "Male"  
[177] "Male"   "Male"   "Male"   "Male"   "Male"   "Female" "Male"   "Male"   "Female" "Female" "Male"  
```

OUTPUT CODE:

```
invalid_indices<-grep("F",Dataset_mid$sex)
Dataset_mid$sex[invalid_indices]<-"Female"
Dataset_mid
```



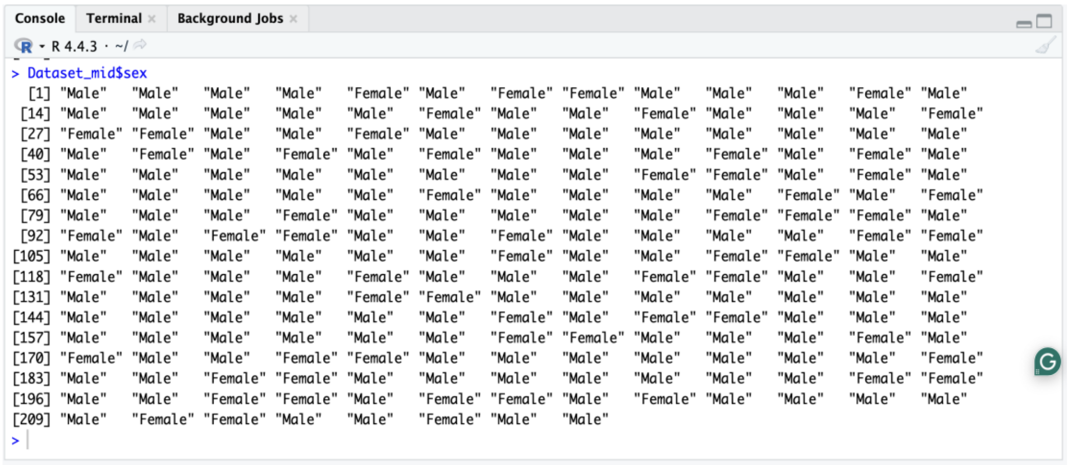
```
23 23  58   Male Cleveland atypical angina     120  284 FALSE lv hypertrophy    160 FALSE     1.8        flat  0
24 24  58   Male Cleveland     non-anginal     132  224 FALSE lv hypertrophy    173 FALSE     3.2   upsloping  2
25 25  60   Male Cleveland    asymptomatic     130  206 FALSE lv hypertrophy    132  TRUE     2.4        flat  2
26 26  50 Female Cleveland     non-anginal     120  219 FALSE         normal    158 FALSE     1.6        flat  0
27 27  58 Female Cleveland     non-anginal     120  340 FALSE         normal    172 FALSE     0.0   upsloping  0
28 28  66 Female Cleveland  typical angina     150  226 FALSE         normal    114 FALSE     2.6 downsloping  0
29 29  43   Male Cleveland    asymptomatic     150  247 FALSE         normal    171 FALSE     1.5   upsloping  0
30 30  40   Male Cleveland    asymptomatic     110  167 FALSE lv hypertrophy    114  TRUE     2.0        flat  0
31 31  69 Female Cleveland  typical angina     140  239 FALSE         normal    151 FALSE     1.8   upsloping  2
32 32  60   Male Cleveland    asymptomatic     117  230  TRUE         normal    160  TRUE     1.4   upsloping  2
33 33  64   Male Cleveland     non-anginal     140  335 FALSE         normal    158 FALSE     0.0   upsloping  0
34 34  59   Male Cleveland    asymptomatic     135  234 FALSE         normal    161 FALSE     0.5        flat  0
35 35  44   Male Cleveland     non-anginal     130  233 FALSE         normal    179  TRUE     0.4   upsloping  0
36 36  42   Male Cleveland    asymptomatic     140  226 FALSE         normal    178 FALSE     0.0   upsloping  0
37 37  43   Male Cleveland    asymptomatic     120  177 FALSE lv hypertrophy    120  TRUE     2.5        flat  0
38 38  57   Male Cleveland    asymptomatic     150  276 FALSE lv hypertrophy    112  TRUE     0.6        flat  1
39 39  55   Male Cleveland    asymptomatic     132  353 FALSE         normal    132  TRUE     1.2        flat  1
40 40  61   Male Cleveland     non-anginal     150  243  TRUE         normal    137  TRUE     1.0        flat  0
41 41  65 Female Cleveland    asymptomatic     150  225 FALSE lv hypertrophy    114 FALSE     1.0        flat  3
42 42  40   Male Cleveland  typical angina     140  199 FALSE         normal    178  TRUE     1.4   upsloping  0
43 43  71 Female Cleveland atypical angina     160  302 FALSE         normal    162 FALSE     0.4   upsloping  2
44 44  59   Male Cleveland     non-anginal     150  212  TRUE         normal    157 FALSE     1.6   upsloping  0
45 45  61 Female Cleveland    asymptomatic     130  330 FALSE lv hypertrophy    169 FALSE     0.0   upsloping  0
```

5. <u>Categorical to Numeric Conversion:</u>

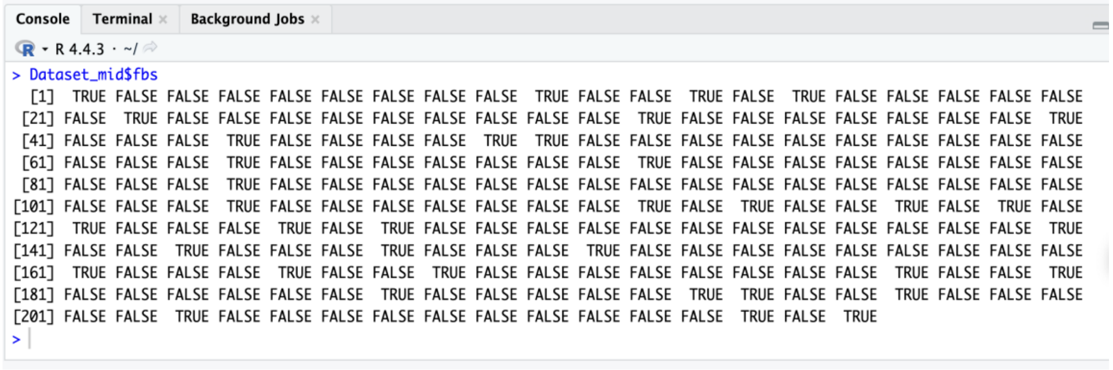
In the dataset, the columns sex, fbs, and exang originally contained categorical values. To prepare them for analysis, we used the **factor() method** to change them into numbers.. For sex, we assigned 0 to Female and 1 to Male. For both fbs and exang, we assigned 0 to FALSE and 1 to TRUE.

<u>CODE:</u>

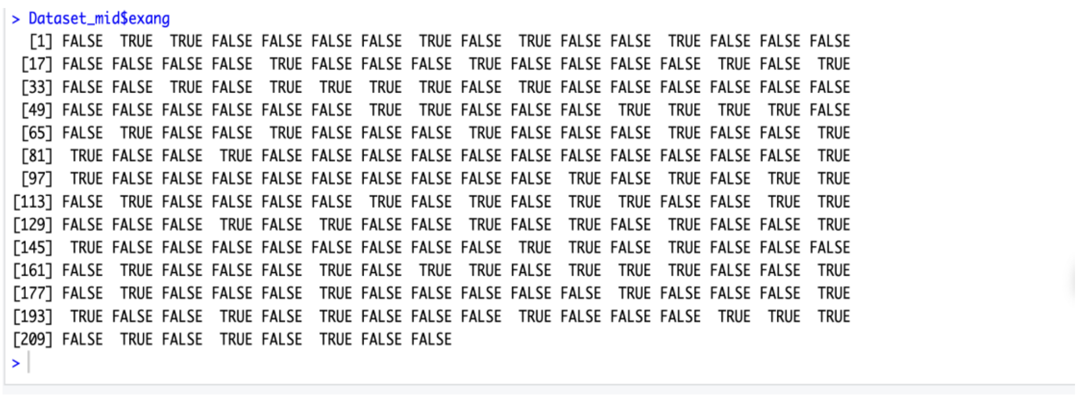
```
Dataset_mid$sex
Dataset_mid$fbs
Dataset_mid$exang
```



```
> Dataset_mid$sex
  [1] "Male"   "Male"   "Male"   "Male"   "Female" "Male"   "Female" "Female" "Male"   "Male"   "Male"   "Female" "Male"  
 [14] "Male"   "Male"   "Male"   "Male"   "Male"   "Female" "Male"   "Male"   "Female" "Male"   "Male"   "Male"   "Female"
 [27] "Female" "Female" "Male"   "Male"   "Female" "Male"   "Male"   "Male"   "Male"   "Male"   "Male"   "Male"   "Male"  
 [40] "Male"   "Female" "Male"   "Female" "Male"   "Female" "Male"   "Male"   "Male"   "Female" "Male"   "Female" "Male"  
 [53] "Male"   "Male"   "Male"   "Male"   "Male"   "Male"   "Male"   "Male"   "Female" "Female" "Male"   "Female" "Male"  
 [66] "Male"   "Male"   "Male"   "Male"   "Male"   "Female" "Male"   "Male"   "Male"   "Male"   "Female" "Male"   "Female"
 [79] "Male"   "Male"   "Male"   "Female" "Male"   "Male"   "Male"   "Male"   "Male"   "Female" "Female" "Female" "Male"  
 [92] "Female" "Male"   "Female" "Female" "Male"   "Male"   "Female" "Male"   "Male"   "Male"   "Male"   "Female" "Female"
[105] "Male"   "Male"   "Male"   "Male"   "Male"   "Male"   "Female" "Male"   "Male"   "Female" "Female" "Male"   "Male"  
[118] "Female" "Male"   "Male"   "Male"   "Female" "Male"   "Male"   "Male"   "Female" "Female" "Male"   "Male"   "Female"
[131] "Male"   "Male"   "Male"   "Male"   "Female" "Female" "Male"   "Male"   "Male"   "Male"   "Male"   "Male"   "Male"  
[144] "Male"   "Male"   "Male"   "Male"   "Male"   "Male"   "Female" "Male"   "Female" "Female" "Male"   "Male"   "Male"  
[157] "Male"   "Male"   "Male"   "Male"   "Male"   "Male"   "Female" "Female" "Male"   "Male"   "Male"   "Female" "Male"  
[170] "Female" "Male"   "Male"   "Female" "Female" "Male"   "Male"   "Male"   "Male"   "Male"   "Male"   "Male"   "Female"
[183] "Male"   "Male"   "Female" "Female" "Male"   "Male"   "Male"   "Male"   "Male"   "Male"   "Male"   "Female" "Female"
[196] "Male"   "Male"   "Female" "Female" "Male"   "Female" "Female" "Male"   "Female" "Male"   "Male"   "Male"   "Male"  
[209] "Male"   "Female" "Female" "Male"   "Male"   "Female" "Male"   "Male"  
> 
```



```
> Dataset_mid$fbs
  [1]  TRUE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE  TRUE FALSE FALSE  TRUE FALSE  TRUE FALSE FALSE FALSE FALSE FALSE
 [21] FALSE  TRUE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE  TRUE FALSE FALSE FALSE FALSE FALSE FALSE FALSE  TRUE
 [41] FALSE FALSE FALSE  TRUE FALSE FALSE FALSE FALSE  TRUE  TRUE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
 [61] FALSE FALSE FALSE  TRUE FALSE FALSE FALSE FALSE FALSE FALSE FALSE  TRUE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
 [81] FALSE FALSE FALSE  TRUE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
[101] FALSE FALSE FALSE  TRUE FALSE FALSE FALSE FALSE FALSE FALSE FALSE  TRUE FALSE  TRUE FALSE FALSE  TRUE FALSE  TRUE FALSE
[121]  TRUE FALSE FALSE FALSE  TRUE FALSE  TRUE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE  TRUE
[141] FALSE FALSE  TRUE FALSE FALSE FALSE  TRUE FALSE FALSE FALSE  TRUE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
[161]  TRUE FALSE FALSE FALSE  TRUE FALSE FALSE  TRUE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE  TRUE FALSE FALSE  TRUE
[181] FALSE FALSE FALSE FALSE FALSE FALSE  TRUE FALSE FALSE FALSE FALSE FALSE  TRUE  TRUE FALSE FALSE  TRUE FALSE FALSE FALSE
[201] FALSE FALSE  TRUE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE  TRUE FALSE  TRUE
> 
```



```
> Dataset_mid$exang
  [1] FALSE  TRUE  TRUE FALSE FALSE FALSE FALSE  TRUE FALSE  TRUE FALSE FALSE  TRUE FALSE FALSE FALSE
 [17] FALSE FALSE FALSE FALSE  TRUE FALSE FALSE FALSE  TRUE FALSE FALSE FALSE FALSE  TRUE FALSE  TRUE
 [33] FALSE FALSE  TRUE FALSE  TRUE  TRUE  TRUE  TRUE FALSE  TRUE FALSE FALSE FALSE FALSE FALSE FALSE
 [49] FALSE FALSE FALSE FALSE FALSE FALSE  TRUE  TRUE FALSE FALSE FALSE  TRUE  TRUE  TRUE  TRUE FALSE
 [65] FALSE  TRUE FALSE FALSE  TRUE FALSE FALSE FALSE  TRUE FALSE FALSE FALSE  TRUE FALSE FALSE  TRUE
 [81]  TRUE FALSE FALSE  TRUE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE  TRUE
 [97]  TRUE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE  TRUE FALSE  TRUE FALSE  TRUE  TRUE
[113] FALSE  TRUE FALSE FALSE FALSE FALSE  TRUE FALSE  TRUE FALSE  TRUE  TRUE FALSE FALSE  TRUE  TRUE
[129] FALSE FALSE FALSE  TRUE FALSE  TRUE FALSE FALSE  TRUE FALSE  TRUE FALSE  TRUE FALSE FALSE  TRUE
[145]  TRUE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE  TRUE  TRUE FALSE  TRUE FALSE FALSE FALSE
[161] FALSE  TRUE FALSE FALSE FALSE  TRUE FALSE  TRUE  TRUE FALSE  TRUE  TRUE  TRUE FALSE FALSE  TRUE
[177] FALSE  TRUE FALSE FALSE FALSE  TRUE FALSE FALSE FALSE FALSE FALSE  TRUE FALSE FALSE FALSE  TRUE
[193]  TRUE FALSE FALSE  TRUE FALSE  TRUE FALSE FALSE FALSE  TRUE FALSE FALSE FALSE  TRUE  TRUE  TRUE
[209] FALSE  TRUE FALSE  TRUE FALSE  TRUE FALSE FALSE
> 
```

OUTPUT CODE:

Dataset_mid$sex <- factor(Dataset_mid$sex, levels = c("Female", "Male"), labels = c(0, 1))

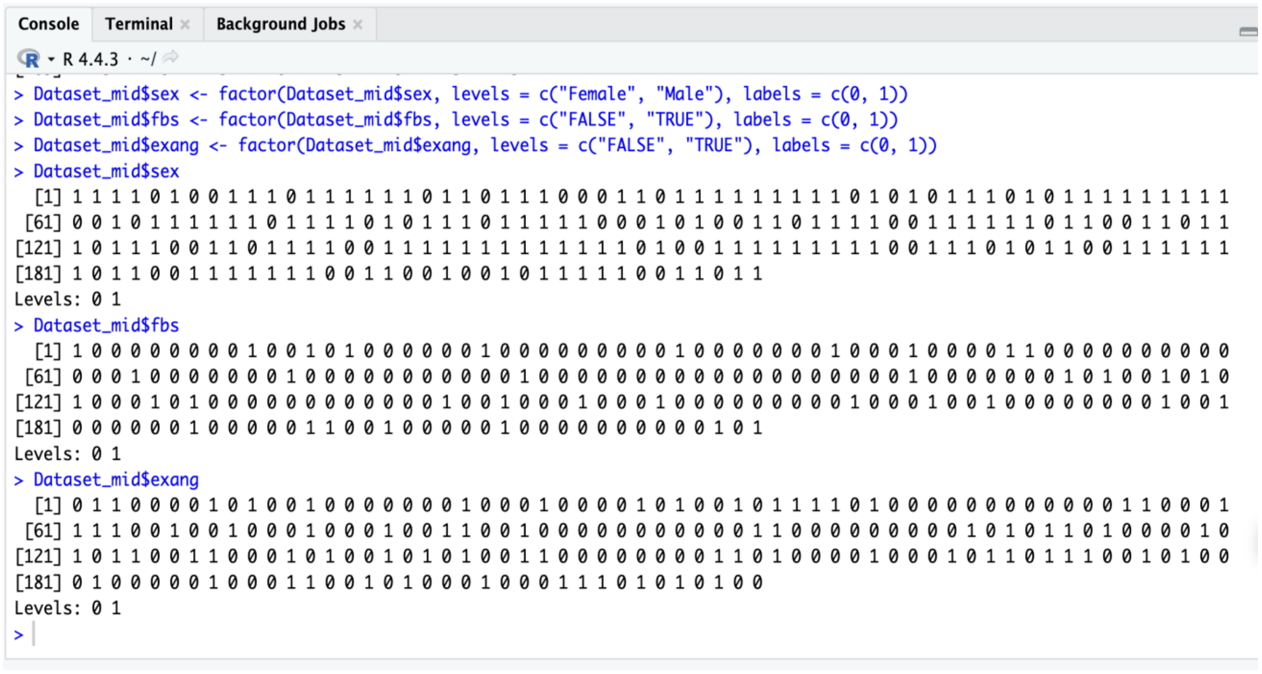
Dataset_mid$fbs <- factor(Dataset_mid$fbs, levels = c("FALSE", "TRUE"), labels = c(0, 1))

Dataset_mid$exang <- factor(Dataset_mid$exang, levels = c("FALSE", "TRUE"), labels = c(0, 1))

Dataset_mid$sex

Dataset_mid$fbs

Dataset_mid$exang

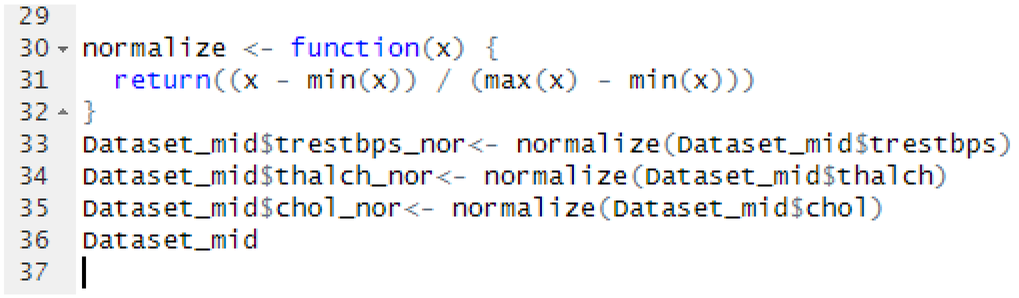


```
> Dataset_mid$sex <- factor(Dataset_mid$sex, levels = c("Female", "Male"), labels = c(0, 1))
> Dataset_mid$fbs <- factor(Dataset_mid$fbs, levels = c("FALSE", "TRUE"), labels = c(0, 1))
> Dataset_mid$exang <- factor(Dataset_mid$exang, levels = c("FALSE", "TRUE"), labels = c(0, 1))
> Dataset_mid$sex
  [1] 1 1 1 1 0 1 0 0 1 1 1 0 1 1 1 1 1 1 0 1 1 0 1 1 1 0 0 0 1 1 0 1 1 1 1 1 1 1 1 1 0 1 0 1 0 1 1 1 0 1 0 1 1 1 1 1 1 1 1 1
 [61] 0 0 1 0 1 1 1 1 1 1 0 1 1 1 1 0 1 0 1 1 1 0 1 1 1 1 1 0 0 0 1 0 1 0 0 1 1 0 1 1 1 1 0 0 1 1 1 1 1 1 0 1 1 0 0 1 1 0 1 1
[121] 1 0 1 1 1 0 0 1 1 0 1 1 1 1 0 0 1 1 1 1 1 1 1 1 1 1 1 1 0 1 0 0 1 1 1 1 1 1 1 1 1 0 0 1 1 1 0 1 0 1 1 0 0 1 1 1 1 1 1 1
[181] 1 0 1 1 0 0 1 1 1 1 1 1 1 0 0 1 1 0 0 1 0 0 1 0 1 1 1 1 1 1 0 0 1 1 0 1 1
Levels: 0 1
> Dataset_mid$fbs
  [1] 1 0 0 0 0 0 0 0 0 1 0 0 1 0 1 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 1 0 0 0 1 0 0 0 0 1 1 0 0 0 0 0 0 0 0 0 0 0
 [61] 0 0 0 1 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 1 0 1 0 0 1 0 1 0
[121] 1 0 0 0 1 0 1 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 1 0 0 0 1 0 0 0 1 0 0 0 0 0 0 0 0 0 1 0 0 0 1 0 0 1 0 0 0 0 0 0 0 0 1 0 0 1
[181] 0 0 0 0 0 0 1 0 0 0 0 0 1 1 0 0 1 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 1 0 1
Levels: 0 1
> Dataset_mid$exang
  [1] 0 1 1 0 0 0 0 1 0 1 0 0 1 0 0 0 0 0 0 0 1 0 0 0 1 0 0 0 0 1 0 1 0 0 1 0 1 1 1 1 0 1 0 0 0 0 0 0 0 0 0 0 0 0 1 1 0 0 0 1
 [61] 1 1 1 0 0 1 0 0 1 0 0 0 1 0 0 0 1 0 0 1 1 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 1 1 0 0 0 0 0 0 0 0 0 0 1 0 1 0 1 1 0 1 0 0 0 0 1 0
[121] 1 0 1 1 0 0 1 1 0 0 0 1 0 1 0 0 1 0 1 0 1 0 0 1 1 0 0 0 0 0 0 0 0 1 1 0 1 0 0 0 0 1 0 0 0 1 0 1 1 0 1 1 1 0 0 1 0 1 0 0
[181] 0 1 0 0 0 0 0 1 0 0 0 1 1 0 0 1 0 1 0 0 0 1 0 0 0 1 1 1 0 1 0 1 0 1 0 0
Levels: 0 1
> 
```

6. Normalization for Continuous attribute:

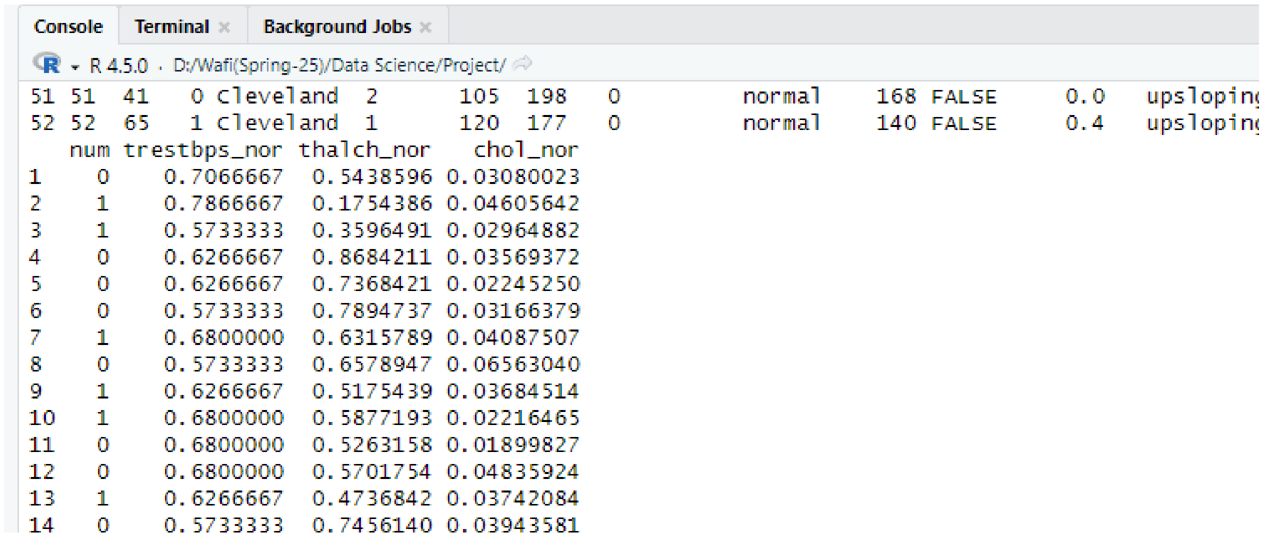
Continuous attributes often have different ranges in data analysis and machine learning, which can impact model performance .To solve this, normalization is applied in the dataset to scale all values between 0 and 1. In this case, a normalize() function has been used to apply Min-Max normalization to the “trestbps”,” thalch”, and “chol” columns. This ensures that all continuous features contribute equally during analysis and helps to improve model performance by preventing attributes with larger values from dominating the results.

CODE:



```r

normalize <- function(x) {
  return((x - min(x)) / (max(x) - min(x)))
}
Dataset_mid$trestbps_nor<- normalize(Dataset_mid$trestbps)
Dataset_mid$thalch_nor<- normalize(Dataset_mid$thalch)
Dataset_mid$chol_nor<- normalize(Dataset_mid$chol)
Dataset_mid

```

OUTPUT:

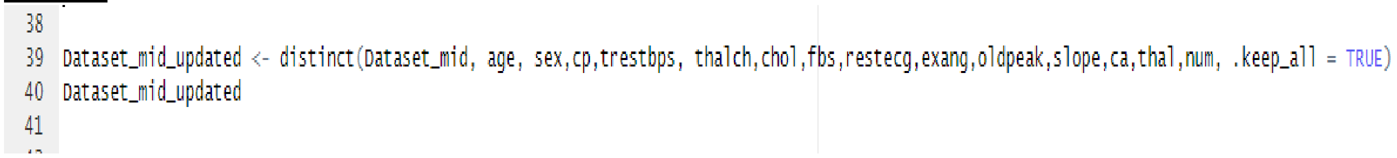


```
51 51  41    0 Cleveland  2       105  198    0          normal     168  FALSE      0.0    upslopin
52 52  65    1 Cleveland  1       120  177    0          normal     140  FALSE      0.4    upslopin
   num trestbps_nor thalch_nor   chol_nor
1    0    0.7066667  0.5438596 0.03080023
2    1    0.7866667  0.1754386 0.04605642
3    1    0.5733333  0.3596491 0.02964882
4    0    0.6266667  0.8684211 0.03569372
5    0    0.6266667  0.7368421 0.02245250
6    0    0.5733333  0.7894737 0.03166379
7    1    0.6800000  0.6315789 0.04087507
8    0    0.5733333  0.6578947 0.06563040
9    1    0.6266667  0.5175439 0.03684514
10   1    0.6800000  0.5877193 0.02216465
11   0    0.6800000  0.5263158 0.01899827
12   0    0.6800000  0.5701754 0.04835924
13   1    0.6266667  0.4736842 0.03742084
14   0    0.5733333  0.7456140 0.03943581
```

7. Removing Duplicates values:

In data analysis, removing duplicate records is an essential step to ensure data quality and accuracy. By using the distinct() function, duplicate rows has been removed from the dataset based on specific columns: age, sex, cp, trestbps, thalch, chol, fbs, restecg, exang, oldpeak, slope, ca, thal, and num. The ".keep_all = TRUE" argument ensures that all columns are retained in the final dataset, keeping only the first row of each unique combination of values across the specified columns.

CODE:

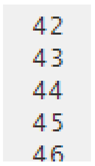


```r

Dataset_mid_updated <- distinct(Dataset_mid, age, sex,cp,trestbps, thalch,chol,fbs,restecg,exang,oldpeak,slope,ca,thal,num, .keep_all = TRUE)
Dataset_mid_updated

```

8. Filtering Method:

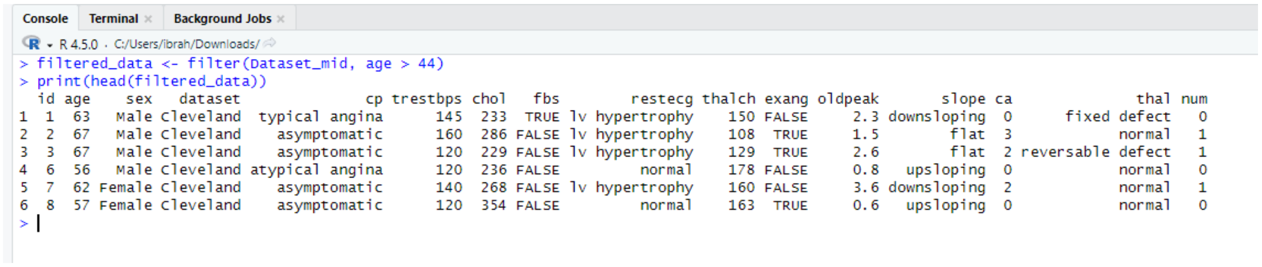
The filter() function is used to produce the subset of the data that satisfies the conditions specified in the filter() method.By using the filter() function, the dataset shows only those patients whose age is greater than 44.

CODE:



```r
filtered_data <- filter(Dataset_mid, age > 44)
print(head(filtered_data))
```

OUTPUT:

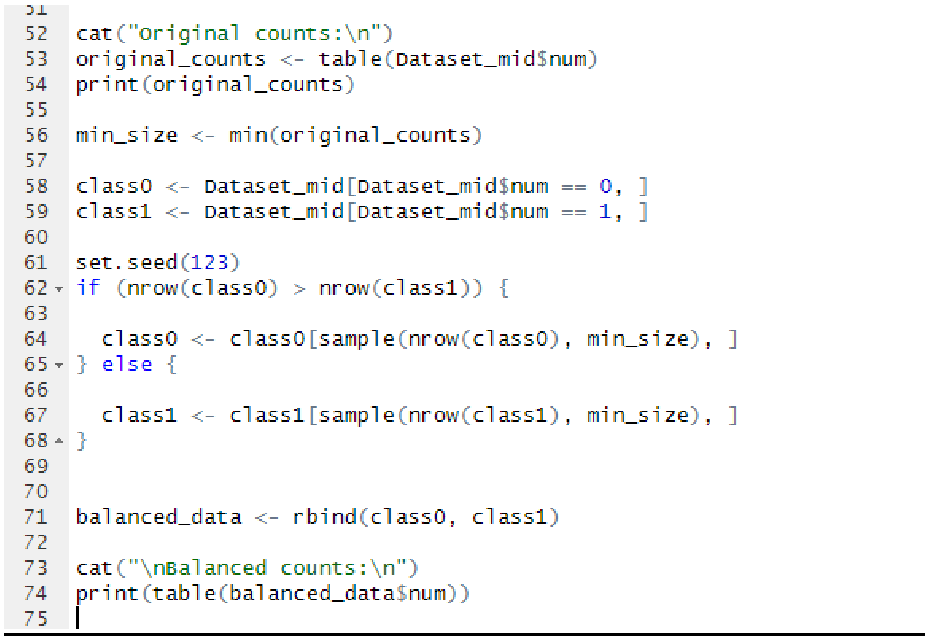


```
> filtered_data <- filter(Dataset_mid, age > 44)
> print(head(filtered_data))
  id age    sex  dataset              cp trestbps chol   fbs        restecg thalch exang oldpeak       slope ca              thal num
1  1  63   Male Cleveland  typical angina      145  233  TRUE lv hypertrophy    150 FALSE     2.3 downsloping  0      fixed defect   0
2  2  67   Male Cleveland    asymptomatic      160  286 FALSE lv hypertrophy    108  TRUE     1.5        flat  3            normal   1
3  3  67   Male Cleveland    asymptomatic      120  229 FALSE lv hypertrophy    129  TRUE     2.6        flat  2 reversable defect   1
4  6  56   Male Cleveland atypical angina      120  236 FALSE         normal    178 FALSE     0.8   upsloping  0            normal   0
5  7  62 Female Cleveland    asymptomatic      140  268 FALSE lv hypertrophy    160 FALSE     3.6 downsloping  2            normal   1
6  8  57 Female Cleveland    asymptomatic      120  354 FALSE         normal    163  TRUE     0.6   upsloping  0            normal   0
> 
```

9. Convert Imbalanced to Balanced Dataset:

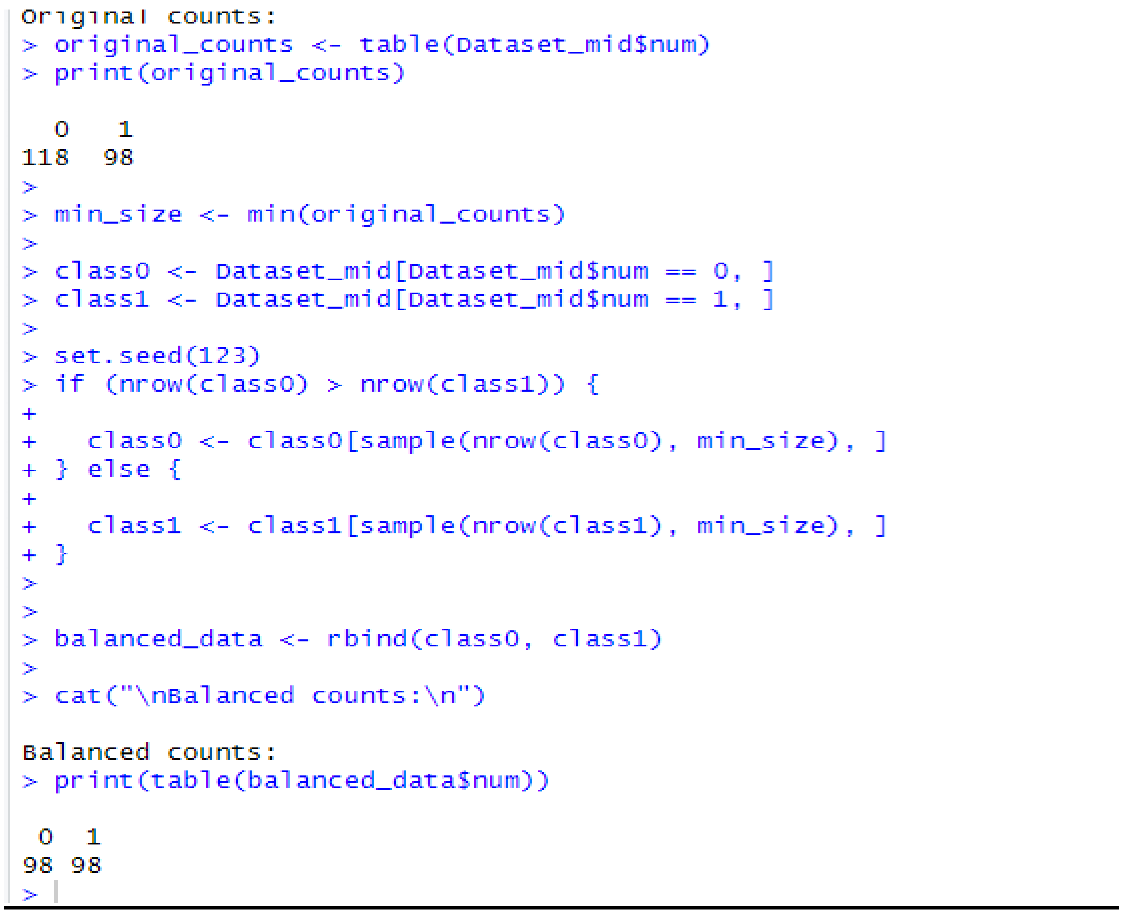
The dataset is balanced by randomly reducing the larger class. We checked the counts of heart disease and non-heart disease cases by using table (). Then randomly selected cases from disease group using sample () to match the non-disease group. Then we combined them with rbind ().

CODE:



```r

cat("Original counts:\n")
original_counts <- table(Dataset_mid$num)
print(original_counts)

min_size <- min(original_counts)

class0 <- Dataset_mid[Dataset_mid$num == 0, ]
class1 <- Dataset_mid[Dataset_mid$num == 1, ]

set.seed(123)
if (nrow(class0) > nrow(class1)) {

  class0 <- class0[sample(nrow(class0), min_size), ]
} else {

  class1 <- class1[sample(nrow(class1), min_size), ]
}


balanced_data <- rbind(class0, class1)

cat("\nBalanced counts:\n")
print(table(balanced_data$num))

```

OUTPUT:

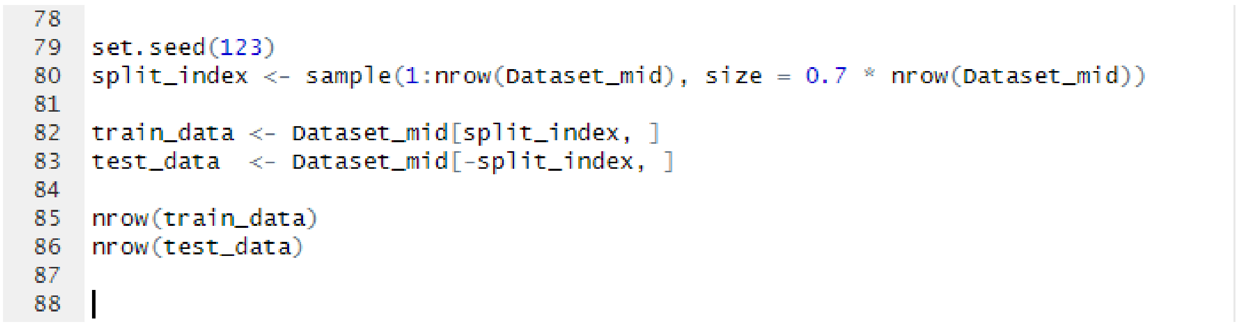


```
Original counts:
> original_counts <- table(Dataset_mid$num)
> print(original_counts)

  0   1 
118  98 
> 
> min_size <- min(original_counts)
> 
> class0 <- Dataset_mid[Dataset_mid$num == 0, ]
> class1 <- Dataset_mid[Dataset_mid$num == 1, ]
> 
> set.seed(123)
> if (nrow(class0) > nrow(class1)) {
+ 
+   class0 <- class0[sample(nrow(class0), min_size), ]
+ } else {
+ 
+   class1 <- class1[sample(nrow(class1), min_size), ]
+ }
> 
> 
> balanced_data <- rbind(class0, class1)
> 
> cat("\nBalanced counts:\n")

Balanced counts:
> print(table(balanced_data$num))

 0  1 
98 98 
> 
```

10. Split the Dataset for Training and Testing

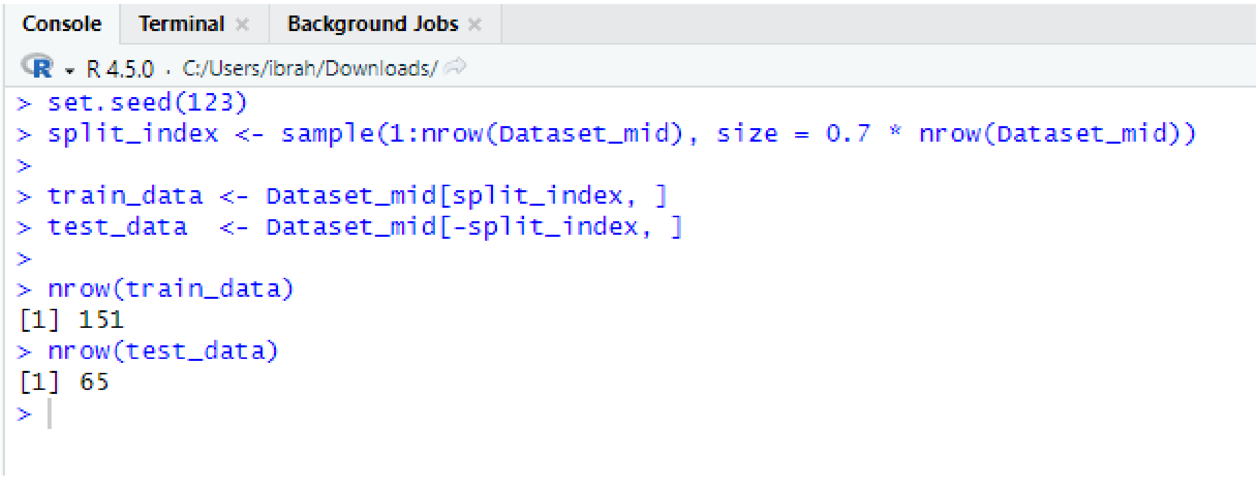
When the dataset is split into training and test sets to analyze the ability of a machine learning algorithm to generalize to new unseen data. The training set will be used to train the algorithm by letting it learn from the patterns and relationships in the data it's been given. After the algorithm has been trained, we then use the test set to determine its ability to make accurate predictions on data it has never encountered. it's critical to create a Training Set and a Test Set from the dataset. This allows the algorithm to learn from one subset of the data and to be tested on unseen data. For this data, the dataset was divided at random using the sample() function. Around 70% of the data were placed in the training set and 30% in the test set. The set.seed() function makes the results repeatable so the same split will be created every time the code runs.

CODE:



```

set.seed(123)
split_index <- sample(1:nrow(Dataset_mid), size = 0.7 * nrow(Dataset_mid))

train_data <- Dataset_mid[split_index, ]
test_data  <- Dataset_mid[-split_index, ]

nrow(train_data)
nrow(test_data)


```

OUTPUT:

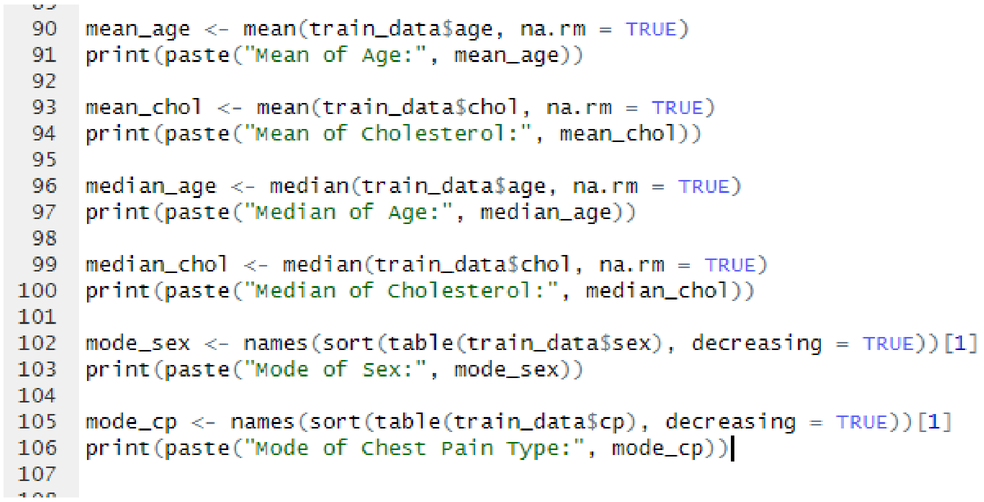


```
> set.seed(123)
> split_index <- sample(1:nrow(Dataset_mid), size = 0.7 * nrow(Dataset_mid))
>
> train_data <- Dataset_mid[split_index, ]
> test_data  <- Dataset_mid[-split_index, ]
>
> nrow(train_data)
[1] 151
> nrow(test_data)
[1] 65
>
```

11. Compute the Central Tendencies (Mean, Median, Mode)

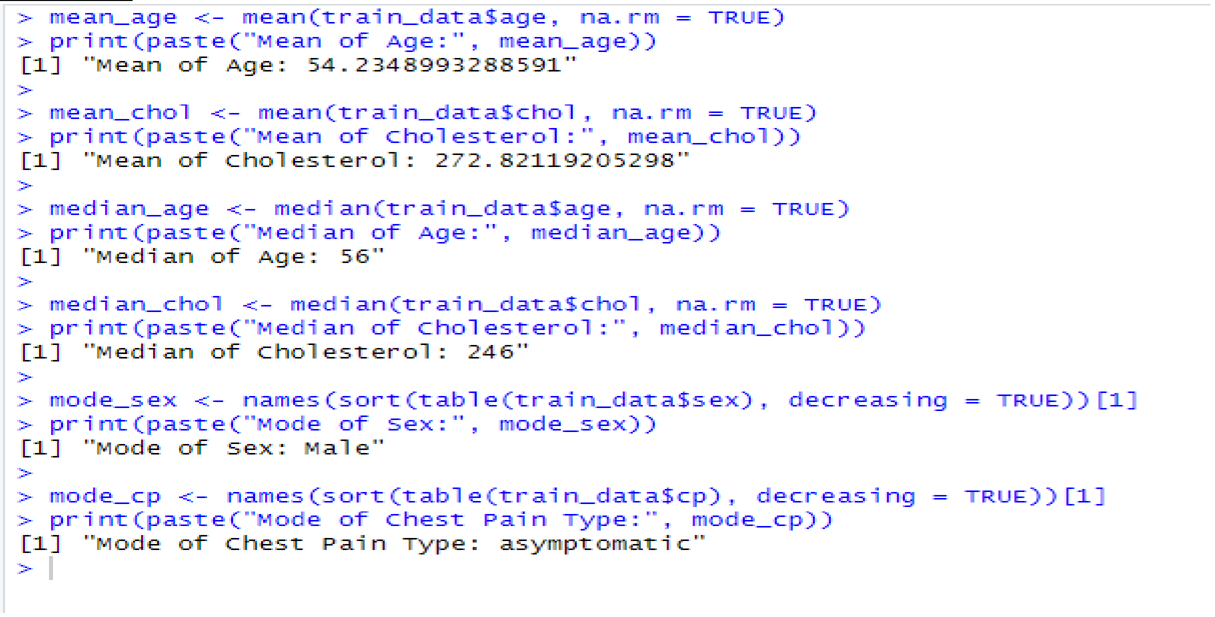
Central tendency measures help us understand what a typical or average value looks like in a dataset. These include the mean which is the average of all values, the median which is the middle value when all values are sorted, and the mode which is the value that appears most often. In this project, we chose two number-based attributes—age and cholesterol level (chol)—and two category-based attributes—sex and chest pain type (cp). For the numeric ones, we calculated the mean and median to see what an average patient's age and cholesterol levels are. For the categorical ones, we found the mode to see which sex and chest pain type were most common among the patients. This gives us a simple idea of the general characteristics of the patient group in the dataset.

CODE:



```r
mean_age <- mean(train_data$age, na.rm = TRUE)
print(paste("Mean of Age:", mean_age))

mean_chol <- mean(train_data$chol, na.rm = TRUE)
print(paste("Mean of Cholesterol:", mean_chol))

median_age <- median(train_data$age, na.rm = TRUE)
print(paste("Median of Age:", median_age))

median_chol <- median(train_data$chol, na.rm = TRUE)
print(paste("Median of Cholesterol:", median_chol))

mode_sex <- names(sort(table(train_data$sex), decreasing = TRUE))[1]
print(paste("Mode of Sex:", mode_sex))

mode_cp <- names(sort(table(train_data$cp), decreasing = TRUE))[1]
print(paste("Mode of Chest Pain Type:", mode_cp))
```

OUTPUT:

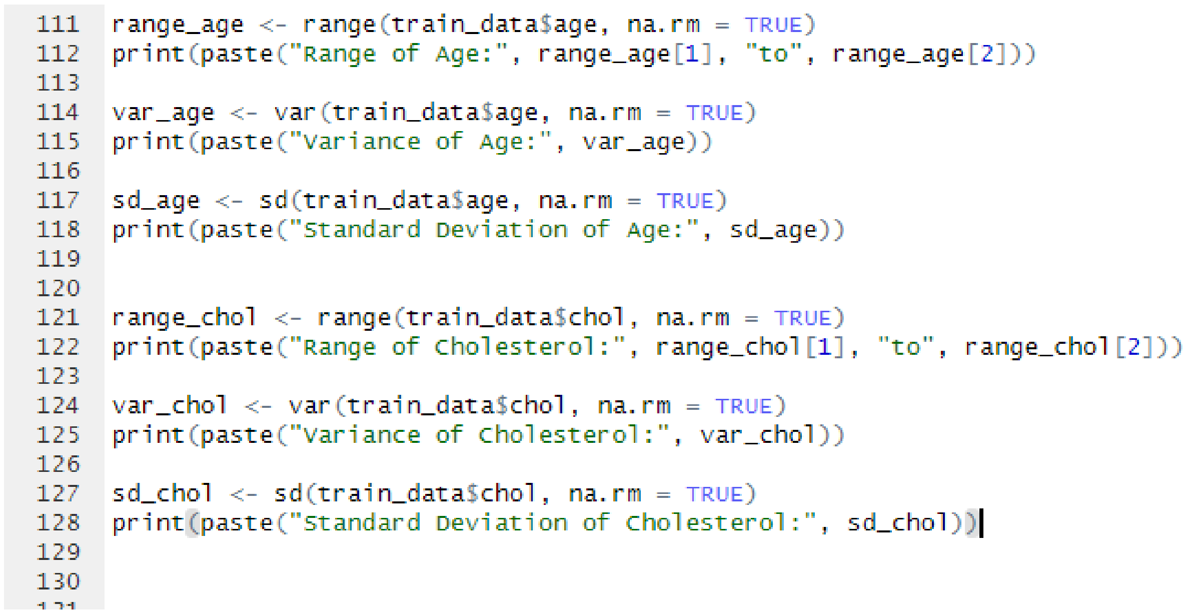


```
> mean_age <- mean(train_data$age, na.rm = TRUE)
> print(paste("Mean of Age:", mean_age))
[1] "Mean of Age: 54.2348993288591"
>
> mean_chol <- mean(train_data$chol, na.rm = TRUE)
> print(paste("Mean of Cholesterol:", mean_chol))
[1] "Mean of Cholesterol: 272.82119205298"
>
> median_age <- median(train_data$age, na.rm = TRUE)
> print(paste("Median of Age:", median_age))
[1] "Median of Age: 56"
>
> median_chol <- median(train_data$chol, na.rm = TRUE)
> print(paste("Median of Cholesterol:", median_chol))
[1] "Median of Cholesterol: 246"
>
> mode_sex <- names(sort(table(train_data$sex), decreasing = TRUE))[1]
> print(paste("Mode of Sex:", mode_sex))
[1] "Mode of Sex: Male"
>
> mode_cp <- names(sort(table(train_data$cp), decreasing = TRUE))[1]
> print(paste("Mode of Chest Pain Type:", mode_cp))
[1] "Mode of Chest Pain Type: asymptomatic"
>
```

12. Spread (Range, Variance, Standard Deviation) computation:

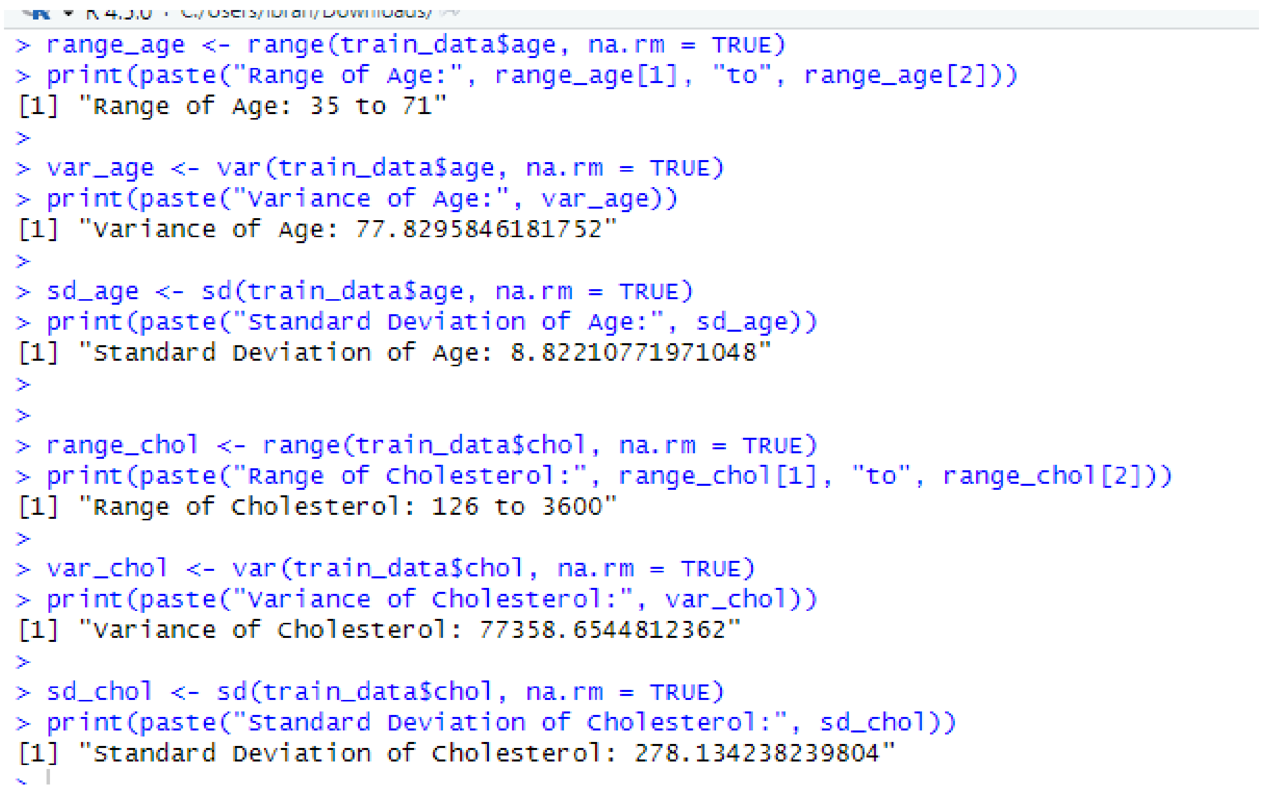
Spread measures give us insight into how the data values are distributed. In this case, age and chol were again used to calculate the range, variance, and standard deviation. These metrics shows us how much variation exists in age and cholesterol levels within the dataset. The range indicates the minimum and the maximum values, the variance shows how far the values spread from the mean, and the standard deviation quantifies the average amount of variation.

CODE:



```r
range_age <- range(train_data$age, na.rm = TRUE)
print(paste("Range of Age:", range_age[1], "to", range_age[2]))

var_age <- var(train_data$age, na.rm = TRUE)
print(paste("Variance of Age:", var_age))

sd_age <- sd(train_data$age, na.rm = TRUE)
print(paste("Standard Deviation of Age:", sd_age))


range_chol <- range(train_data$chol, na.rm = TRUE)
print(paste("Range of Cholesterol:", range_chol[1], "to", range_chol[2]))

var_chol <- var(train_data$chol, na.rm = TRUE)
print(paste("Variance of Cholesterol:", var_chol))

sd_chol <- sd(train_data$chol, na.rm = TRUE)
print(paste("Standard Deviation of Cholesterol:", sd_chol))
```

OUTPUT:



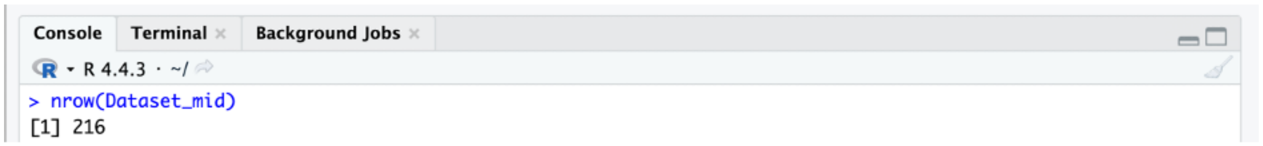
```
> range_age <- range(train_data$age, na.rm = TRUE)
> print(paste("Range of Age:", range_age[1], "to", range_age[2]))
[1] "Range of Age: 35 to 71"
>
> var_age <- var(train_data$age, na.rm = TRUE)
> print(paste("Variance of Age:", var_age))
[1] "Variance of Age: 77.8295846181752"
>
> sd_age <- sd(train_data$age, na.rm = TRUE)
> print(paste("Standard Deviation of Age:", sd_age))
[1] "Standard Deviation of Age: 8.82210771971048"
>
>
> range_chol <- range(train_data$chol, na.rm = TRUE)
> print(paste("Range of Cholesterol:", range_chol[1], "to", range_chol[2]))
[1] "Range of Cholesterol: 126 to 3600"
>
> var_chol <- var(train_data$chol, na.rm = TRUE)
> print(paste("Variance of Cholesterol:", var_chol))
[1] "Variance of Cholesterol: 77358.6544812362"
>
> sd_chol <- sd(train_data$chol, na.rm = TRUE)
> print(paste("Standard Deviation of Cholesterol:", sd_chol))
[1] "Standard Deviation of Cholesterol: 278.134238239804"
```

13. Removing Outliers:

We checked the number of rows and removed outliers from trestbps and chol using the IQR method. This helped remove extreme values. After filtering, the number of rows decreased and we used summary() to view the updated data.

CODE:

nrow(Dataset_mid)



```
> nrow(Dataset_mid)
[1] 216
```

OUTPUT CODE:

- **Remove outliers from trestbps attribute:**
  Q1 <- quantile(Dataset_mid$trestbps, 0.25)
  Q3 <- quantile(Dataset_mid$trestbps, 0.75)
  IQR <- Q3 - Q1
  lower <- Q1 - 1.5 * IQR
  upper <- Q3 + 1.5 * IQR
  Dataset_mid <- Dataset_mid[Dataset_mid$trestbps >= lower & Dataset_mid$trestbps <= upper,]

- <u>**Remove outliers from chol attribute**</u>

Q1 <- quantile(Dataset_mid$chol, 0.25)

Q3 <- quantile(Dataset_mid$chol, 0.75)

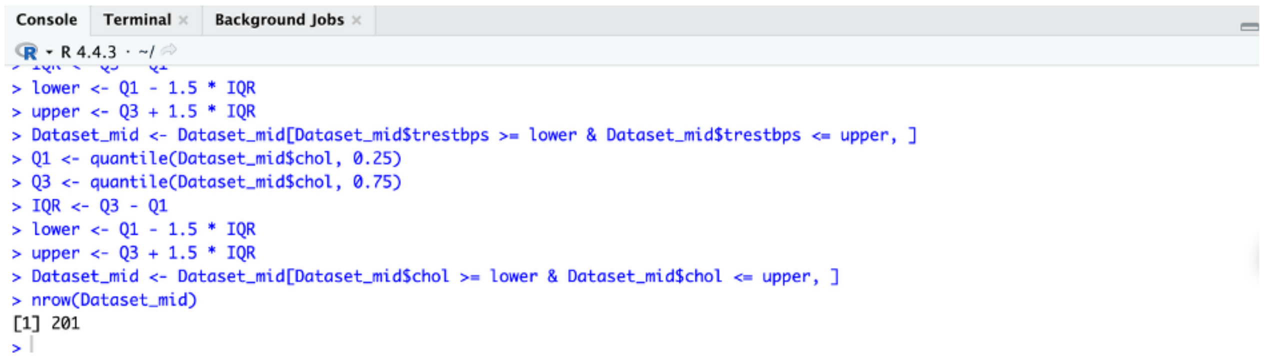
IQR <- Q3 - Q1

lower <- Q1 - 1.5 * IQR

upper <- Q3 + 1.5 * IQR

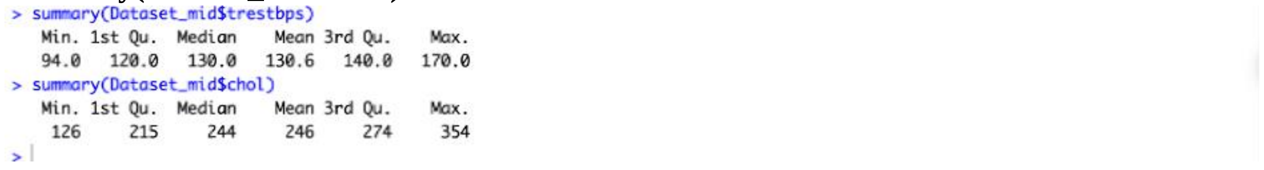
Dataset_mid <- Dataset_mid[Dataset_mid$chol >= lower & Dataset_mid$chol <= upper,]

nrow(Dataset_mid)



```
> lower <- Q1 - 1.5 * IQR
> upper <- Q3 + 1.5 * IQR
> Dataset_mid <- Dataset_mid[Dataset_mid$trestbps >= lower & Dataset_mid$trestbps <= upper, ]
> Q1 <- quantile(Dataset_mid$chol, 0.25)
> Q3 <- quantile(Dataset_mid$chol, 0.75)
> IQR <- Q3 - Q1
> lower <- Q1 - 1.5 * IQR
> upper <- Q3 + 1.5 * IQR
> Dataset_mid <- Dataset_mid[Dataset_mid$chol >= lower & Dataset_mid$chol <= upper, ]
> nrow(Dataset_mid)
[1] 201
>
```

summary(Dataset_mid$age)
summary(Dataset_mid$chol)



```
> summary(Dataset_mid$trestbps)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
   94.0   120.0   130.0   130.6   140.0   170.0 
> summary(Dataset_mid$chol)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
    126     215     244     246     274     354 
>
```

Conclusion:

With the help of this project, the heart disease dataset was effectively preprocessed by handling missing values, transforming categorical data to numerical, normalizing continuous variables, and eliminating outliers and duplicates. After that, the dataset was divided into testing and training sets. The distribution of important characteristics, such as age and cholesterol levels, was examined using central tendency and spread methods. To improve patient outcomes and anticipate the risk of heart disease, these procedures make sure the dataset is balanced, clean, and prepared for additional research.