AMERICAN INTERNATIONAL UNIVERSITY–BANGLADESH (AIUB)

FACULTY OF SCIENCE & TECHNOLOGY

INTRODUCTION TO DATA SCIENCE

SPRING 23-24

Section: C

Group: 09

PROJECT REPORT ON

Applying Data Pre-processing on a Dataset

Supervised By

TOHEDUL ISLAM

Submitted By

Name	ID
TOUHID ALAM	22-46330-1
SUMAIYA SARKAR SHIMLA	22-46172-1

Date of Submission: **March 18, 2024.**

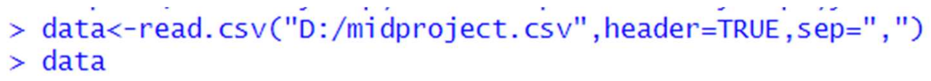
Introduction:

Data pre-processing is a phase in the data analysis and data mining process where raw data is taken and converted into a clean format that computers and machine learning models can understand and analysis. The dataset in the project demonstrates the significant risk factors for maternal mortality, which is one of the main concerns of SDG of UN. The data was collected from different hospitals, community clinics and maternal health cares from the rural areas of Bangladesh through the IoT based risk monitoring system. The raw data may include inaccuracies and mistakes. It must be cleaned to be used for the intended purpose.

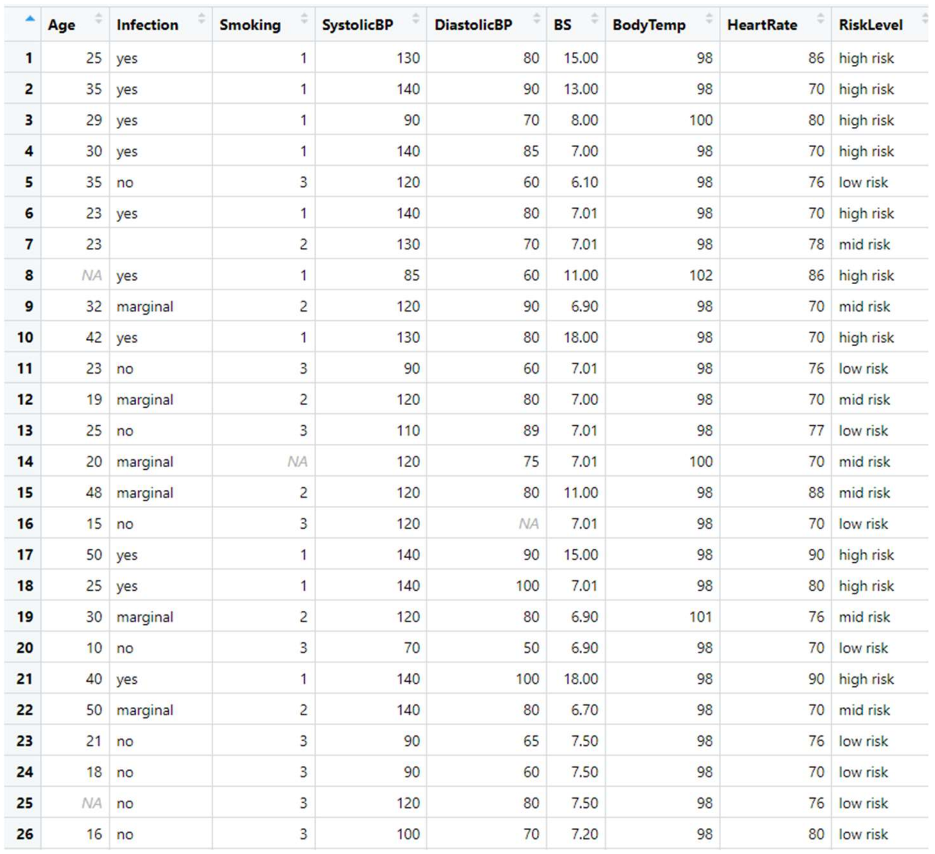
Data Exploration:

Import the dataset:

The dataset is saved as the midproject.csv file. To begin preprocessing data in R, we must first import the file,



```
> data<-read.csv("D:/midproject.csv",header=TRUE,sep=",")
> data
```



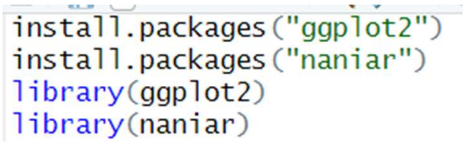
	Age	Infection	Smoking	SystolicBP	DiastolicBP	BS	BodyTemp	HeartRate	RiskLevel
1	25	yes	1	130	80	15.00	98	86	high risk
2	35	yes	1	140	90	13.00	98	70	high risk
3	29	yes	1	90	70	8.00	100	80	high risk
4	30	yes	1	140	85	7.00	98	70	high risk
5	35	no	3	120	60	6.10	98	76	low risk
6	23	yes	1	140	80	7.01	98	70	high risk
7	23		2	130	70	7.01	98	78	mid risk
8	NA	yes	1	85	60	11.00	102	86	high risk
9	32	marginal	2	120	90	6.90	98	70	mid risk
10	42	yes	1	130	80	18.00	98	70	high risk
11	23	no	3	90	60	7.01	98	76	low risk
12	19	marginal	2	120	80	7.00	98	70	mid risk
13	25	no	3	110	89	7.01	98	77	low risk
14	20	marginal	NA	120	75	7.01	100	70	mid risk
15	48	marginal	2	120	80	11.00	98	88	mid risk
16	15	no	3	120	NA	7.01	98	70	low risk
17	50	yes	1	140	90	15.00	98	90	high risk
18	25	yes	1	140	100	7.01	98	80	high risk
19	30	marginal	2	120	80	6.90	101	76	mid risk
20	10	no	3	70	50	6.90	98	70	low risk
21	40	yes	1	140	100	18.00	98	90	high risk
22	50	marginal	2	140	80	6.70	98	70	mid risk
23	21	no	3	90	65	7.50	98	76	low risk
24	18	no	3	90	60	7.50	98	70	low risk
25	NA	no	3	120	80	7.50	98	76	low risk
26	16	no	3	100	70	7.20	98	80	low risk

Figure 01: Imported Raw Dataset.

After importing the dataset, the data is converted into a R data-frame and is stored in the data variable. Let's see the overall summary of the dataset.

Required Library:

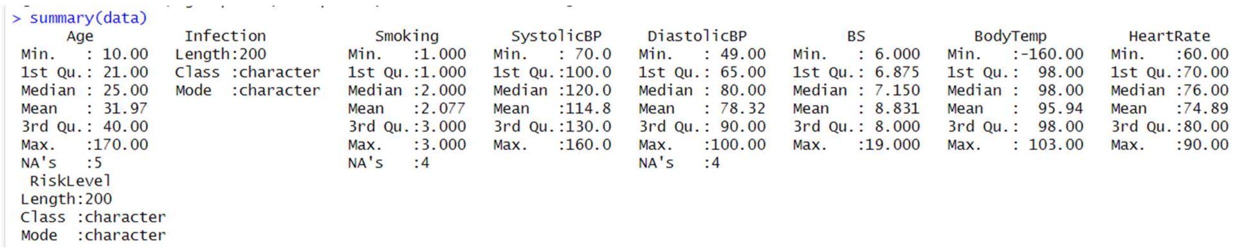
For our data visualization purpose, we will use ggplot2 and narniar library,



```
install.packages("ggplot2")
install.packages("naniar")
library(ggplot2)
library(naniar)
```

Summary of the dataset:

summary(data)

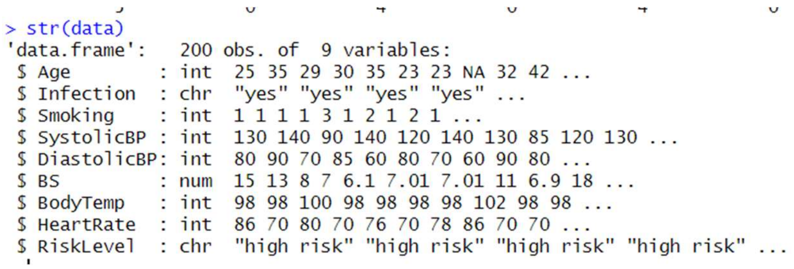


```
> summary(data)
      Age          Infection            Smoking        SystolicBP      DiastolicBP           BS            BodyTemp         HeartRate
 Min.   : 10.00   Length:200         Min.   :1.000   Min.   : 70.0   Min.   : 49.00   Min.   : 6.000   Min.   :-160.00   Min.   :60.00
 1st Qu.: 21.00   Class :character   1st Qu.:1.000   1st Qu.:100.0   1st Qu.: 65.00   1st Qu.: 6.875   1st Qu.:  98.00   1st Qu.:70.00
 Median : 25.00   Mode  :character   Median :2.000   Median :120.0   Median : 80.00   Median : 7.150   Median :  98.00   Median :76.00
 Mean   : 31.97                      Mean   :2.077   Mean   :114.8   Mean   : 78.32   Mean   : 8.831   Mean   :  95.94   Mean   :74.89
 3rd Qu.: 40.00                      3rd Qu.:3.000   3rd Qu.:130.0   3rd Qu.: 90.00   3rd Qu.: 8.000   3rd Qu.:  98.00   3rd Qu.:80.00
 Max.   :170.00                      Max.   :3.000   Max.   :160.0   Max.   :100.00   Max.   :19.000   Max.   : 103.00   Max.   :90.00
 NA's   :5                           NA's   :4                       NA's   :4
  RiskLevel
 Length:200
 Class :character
 Mode  :character
```

Figure 2: Summary of the raw dataset.

In the summary we can see the dataset has 9 attributes where some attributes have missing values.

To show the data types of the dataset,

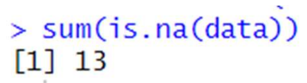


```
> str(data)
'data.frame':   200 obs. of  9 variables:
 $ Age        : int  25 35 29 30 35 23 23 NA 32 42 ...
 $ Infection  : chr  "yes" "yes" "yes" "yes" ...
 $ Smoking    : int  1 1 1 1 3 1 2 1 2 1 ...
 $ SystolicBP : int  130 140 90 140 120 140 130 85 120 130 ...
 $ DiastolicBP: int  80 90 70 85 60 80 70 60 90 80 ...
 $ BS         : num  15 13 8 7 6.1 7.01 7.01 11 6.9 18 ...
 $ BodyTemp   : int  98 98 100 98 98 98 98 102 98 98 ...
 $ HeartRate  : int  86 70 80 70 76 70 78 86 70 70 ...
 $ RiskLevel  : chr  "high risk" "high risk" "high risk" "high risk" ...
```

Figure 3: Attribute types and values of the datatype.

We can see there are 200 instances in the dataset. The Age, SystolicBP, DiastolicBP, BodyTemp and HeartRate attribute is integer, the BS attribute is in numerical form and the Infection and RiskLevel attribute is character, the Smoking attribute is converted from character to integers.

To show the total number of the missing values



```
> sum(is.na(data))
[1] 13
```

The dataset has 13 missing values distributed in the whole dataset. We need to recover them while preprocessing the dataset.

To show the number of the missing values in graph attribute wish:

gg_miss_upset(data)

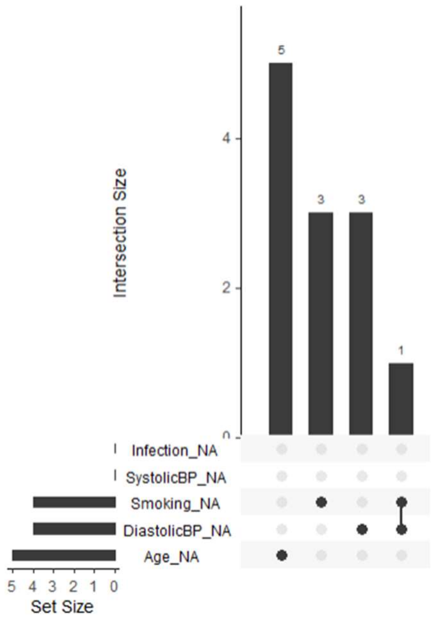


Figure 4: Missing values of the dataset.

Project Solution Design:

The dataset shows there are missing values in a couple attributes. We must deal with them. Also, there may be some invalid values. These values must be recovered properly. Moreover, if there are any outliers in particular attribute, we must recover them using proper method. Finally, we also need to convert attributes to their proper data type. For example, we need to convert the Smoking attribute to character datatype. For recovering missing values, invalid values, or outliers we may use average (mean) or most frequent value (mode) depending on the datatype of the attribute.

Data Pre-processing:

Age Attribute:

The age column is an integer type attribute in years when a woman during pregnant.

```
> class(data$Age)
[1] "integer"
```

Also, there is no invalid data type in the age column.

```
> any(!as.numeric(data$Age))
[1] FALSE
```

Let's remove rows with empty values in the "Age" column along with all the columns in the dataset to remove the missing values.

```
> data<-data[!is.na(data$Age),]
> data
```

Now, let's see if there are any outliers in the Age column using boxplot.

```
> boxplot(data$Age,main="Boxplot of Age",ylab="Values")
```

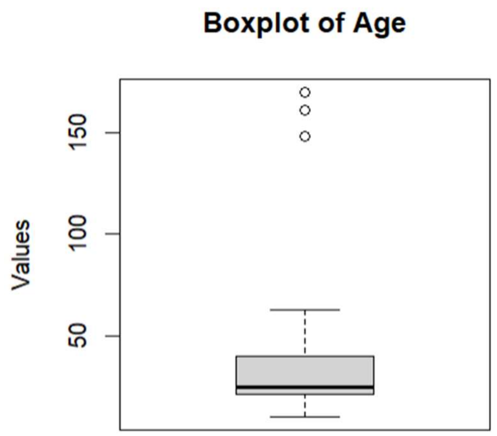
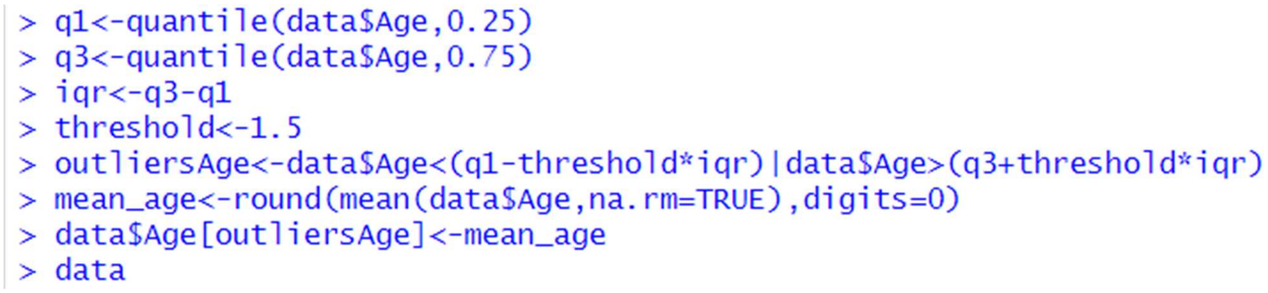


Figure 5: Boxplot of Age attribute.

These values are:

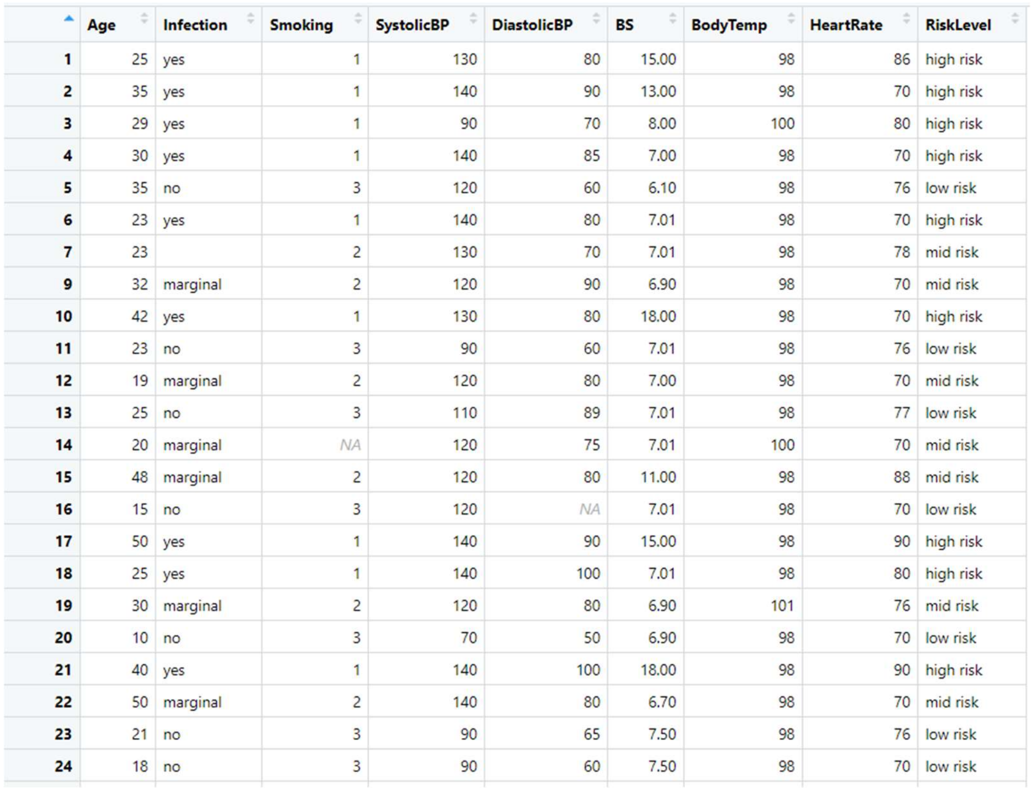
```
> boxplot.stats(data$Age)$out
[1] 148 161 170
```

Let’s recover these values using IQR method:



```
> q1<-quantile(data$Age,0.25)
> q3<-quantile(data$Age,0.75)
> iqr<-q3-q1
> threshold<-1.5
> outliersAge<-data$Age<(q1-threshold*iqr)|data$Age>(q3+threshold*iqr)
> mean_age<-round(mean(data$Age,na.rm=TRUE),digits=0)
> data$Age[outliersAge]<-mean_age
> data
```

Now, the dataset looks like this.



	Age	Infection	Smoking	SystolicBP	DiastolicBP	BS	BodyTemp	HeartRate	RiskLevel
1	25	yes	1	130	80	15.00	98	86	high risk
2	35	yes	1	140	90	13.00	98	70	high risk
3	29	yes	1	90	70	8.00	100	80	high risk
4	30	yes	1	140	85	7.00	98	70	high risk
5	35	no	3	120	60	6.10	98	76	low risk
6	23	yes	1	140	80	7.01	98	70	high risk
7	23		2	130	70	7.01	98	78	mid risk
9	32	marginal	2	120	90	6.90	98	70	mid risk
10	42	yes	1	130	80	18.00	98	70	high risk
11	23	no	3	90	60	7.01	98	76	low risk
12	19	marginal	2	120	80	7.00	98	70	mid risk
13	25	no	3	110	89	7.01	98	77	low risk
14	20	marginal	NA	120	75	7.01	100	70	mid risk
15	48	marginal	2	120	80	11.00	98	88	mid risk
16	15	no	3	120	NA	7.01	98	70	low risk
17	50	yes	1	140	90	15.00	98	90	high risk
18	25	yes	1	140	100	7.01	98	80	high risk
19	30	marginal	2	120	80	6.90	101	76	mid risk
20	10	no	3	70	50	6.90	98	70	low risk
21	40	yes	1	140	100	18.00	98	90	high risk
22	50	marginal	2	140	80	6.70	98	70	mid risk
23	21	no	3	90	65	7.50	98	76	low risk
24	18	no	3	90	60	7.50	98	70	low risk

Figure 6: Dataset after recovering outliers of Age column.

The Age attribute is now clean and ready to use.

Infection Attribute:

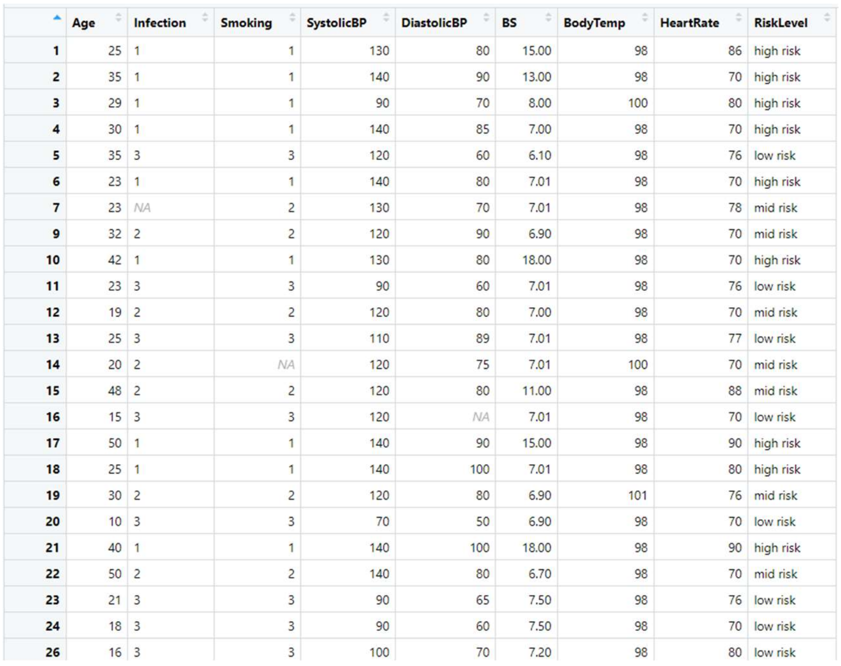
Firstly, let’s check what values are there in the Infection Column.

```
> unique(data$Infection)
[1] "yes"      "no"       ""         "marginal" "yesss"    "yoo"
```

Here, three valid values are yes, marginal, and no and the value “yesss” and “yoo” and missing values are invalid. As the Infection column is categorical, we will convert it in numerical values to recover any missing values or invalid values.

```
> data$Infection[data$Infection=="yes"]<- 1
> data$Infection[data$Infection=="marginal"]<- 2
> data$Infection[data$Infection=="no"]<- 3
> data$Infection[!(data$Infection%in%c(1,2,3))]<-NA
> data$Infection<-as.numeric(data$Infection)
```

Now, the Infection column looks like this.



	Age	Infection	Smoking	SystolicBP	DiastolicBP	BS	BodyTemp	HeartRate	RiskLevel
1	25	1	1	130	80	15.00	98	86	high risk
2	35	1	1	140	90	13.00	98	70	high risk
3	29	1	1	90	70	8.00	100	80	high risk
4	30	1	1	140	85	7.00	98	70	high risk
5	35	3	3	120	60	6.10	98	76	low risk
6	23	1	1	140	80	7.01	98	70	high risk
7	23	NA	2	130	70	7.01	98	78	mid risk
9	32	2	2	120	90	6.90	98	70	mid risk
10	42	1	1	130	80	18.00	98	70	high risk
11	23	3	3	90	60	7.01	98	76	low risk
12	19	2	2	120	80	7.00	98	70	mid risk
13	25	3	3	110	89	7.01	98	77	low risk
14	20	2	NA	120	75	7.01	100	70	mid risk
15	48	2	2	120	80	11.00	98	88	mid risk
16	15	3	3	120	NA	7.01	98	70	low risk
17	50	1	1	140	90	15.00	98	90	high risk
18	25	1	1	140	100	7.01	98	80	high risk
19	30	2	2	120	80	6.90	101	76	mid risk
20	10	3	3	70	50	6.90	98	70	low risk
21	40	1	1	140	100	18.00	98	90	high risk
22	50	2	2	140	80	6.70	98	70	mid risk
23	21	3	3	90	65	7.50	98	76	low risk
24	18	3	3	90	60	7.50	98	70	low risk
26	16	3	3	100	70	7.20	98	80	low risk

Figure 7: Dataset after convering the Infection column in numeric type.

Now, we can see there are 10 missing values in the Infection column.

```
> sum(is.na(data$Infection))
[1] 10
```

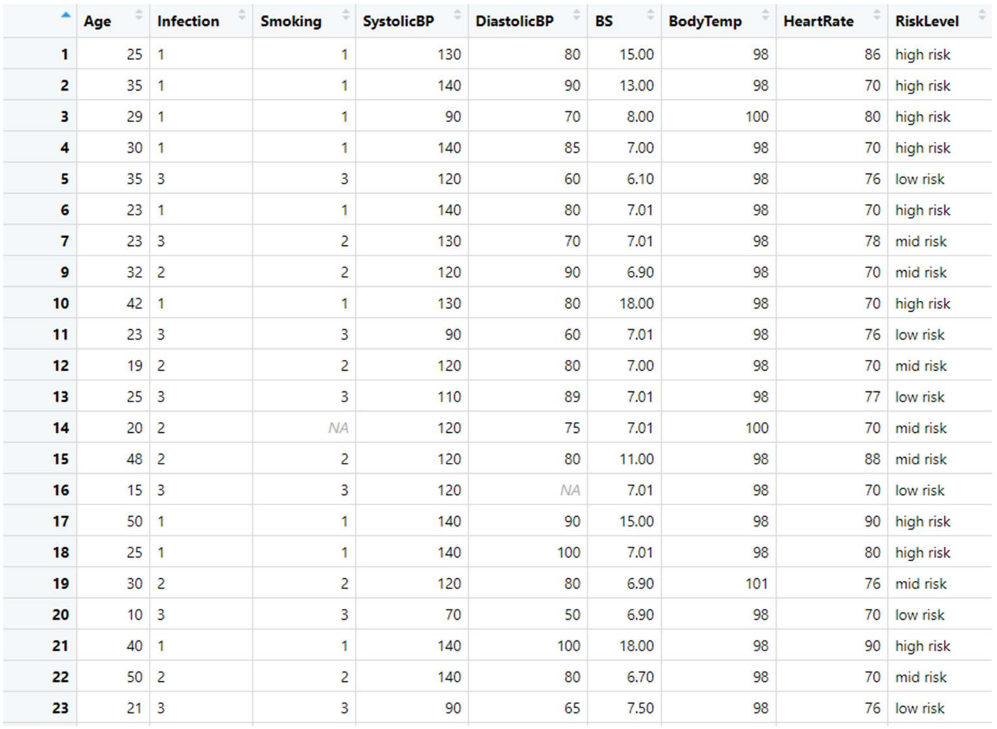
As the infection column is categorical, let’s recover these missing values using the most frequent(mode) value.

```
> names(sort(table(data$Infection),decreasing=TRUE))[1]
[1] "3"
```

So, the most frequent value of the Infection column is 3 (no).

```
> data$Infection[is.na(data$Infection)]<-names(sort(table(data$Infection),decreasing=TRUE))[1]
```

Now, the dataset looks like this.



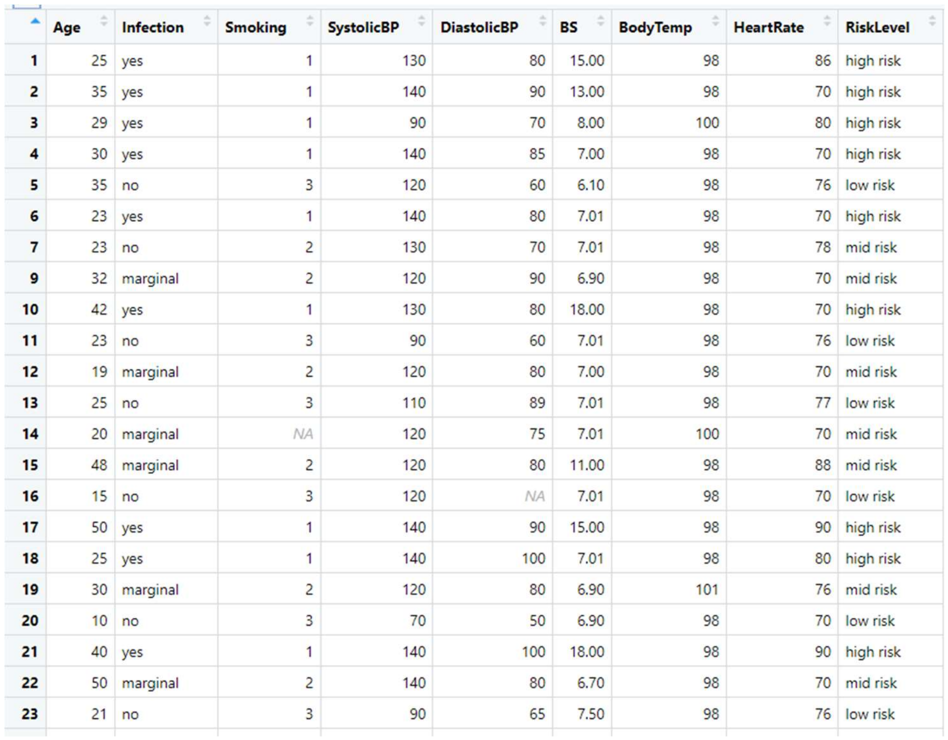
	Age	Infection	Smoking	SystolicBP	DiastolicBP	BS	BodyTemp	HeartRate	RiskLevel
1	25	1	1	130	80	15.00	98	86	high risk
2	35	1	1	140	90	13.00	98	70	high risk
3	29	1	1	90	70	8.00	100	80	high risk
4	30	1	1	140	85	7.00	98	70	high risk
5	35	3	3	120	60	6.10	98	76	low risk
6	23	1	1	140	80	7.01	98	70	high risk
7	23	3	2	130	70	7.01	98	78	mid risk
9	32	2	2	120	90	6.90	98	70	mid risk
10	42	1	1	130	80	18.00	98	70	high risk
11	23	3	3	90	60	7.01	98	76	low risk
12	19	2	2	120	80	7.00	98	70	mid risk
13	25	3	3	110	89	7.01	98	77	low risk
14	20	2	NA	120	75	7.01	100	70	mid risk
15	48	2	2	120	80	11.00	98	88	mid risk
16	15	3	3	120	NA	7.01	98	70	low risk
17	50	1	1	140	90	15.00	98	90	high risk
18	25	1	1	140	100	7.01	98	80	high risk
19	30	2	2	120	80	6.90	101	76	mid risk
20	10	3	3	70	50	6.90	98	70	low risk
21	40	1	1	140	100	18.00	98	90	high risk
22	50	2	2	140	80	6.70	98	70	mid risk
23	21	3	3	90	65	7.50	98	76	low risk

Figure 8: Dataset after recovering the missing values with mode value of Infection attribute.

Let's convert the Infection attribute's values back to the original values "yes", "marginal" and "no".

```
> data$Infection<-as.character(data$Infection)
> data$Infection[data$Infection=="1"]<- "yes"
> data$Infection[data$Infection=="2"]<- "marginal"
> data$Infection[data$Infection=="3"]<- "no"
```

Now, the dataset looks like this,



	Age	Infection	Smoking	SystolicBP	DiastolicBP	BS	BodyTemp	HeartRate	RiskLevel
1	25	yes	1	130	80	15.00	98	86	high risk
2	35	yes	1	140	90	13.00	98	70	high risk
3	29	yes	1	90	70	8.00	100	80	high risk
4	30	yes	1	140	85	7.00	98	70	high risk
5	35	no	3	120	60	6.10	98	76	low risk
6	23	yes	1	140	80	7.01	98	70	high risk
7	23	no	2	130	70	7.01	98	78	mid risk
9	32	marginal	2	120	90	6.90	98	70	mid risk
10	42	yes	1	130	80	18.00	98	70	high risk
11	23	no	3	90	60	7.01	98	76	low risk
12	19	marginal	2	120	80	7.00	98	70	mid risk
13	25	no	3	110	89	7.01	98	77	low risk
14	20	marginal	NA	120	75	7.01	100	70	mid risk
15	48	marginal	2	120	80	11.00	98	88	mid risk
16	15	no	3	120	NA	7.01	98	70	low risk
17	50	yes	1	140	90	15.00	98	90	high risk
18	25	yes	1	140	100	7.01	98	80	high risk
19	30	marginal	2	120	80	6.90	101	76	mid risk
20	10	no	3	70	50	6.90	98	70	low risk
21	40	yes	1	140	100	18.00	98	90	high risk
22	50	marginal	2	140	80	6.70	98	70	mid risk
23	21	no	3	90	65	7.50	98	76	low risk

Figure 9: Dataset after converting the Infection column in categorical type.

Smoking Attribute:

Let's see the datatype of Smoking attribute.

```
> class(data$Smoking)
[1] "integer"
```

We can see the Smoking attribute is Integer type but as the dataset, here 1 means yes, 2 means sometimes and 3 means no. So, the smoking attribute is categorical but converted into integer and there are no invalid values.

```
> unique(data$Smoking)
[1]  1  3  2 NA
```

Let's see how many missing values the smoking column has.

```
> sum(is.na(data$Smoking))
[1] 4
```

Let's recover them using the most frequent(mode) value as the smoking attribute is in categorical type.

```
> smoking_mode<-names(sort(table(data$Smoking),decreasing=TRUE))[1]
> smoking_mode
[1] "3"
```

So, the most frequent value of Smoking attribute is "3" which means no.

Let’s recover the missing values.

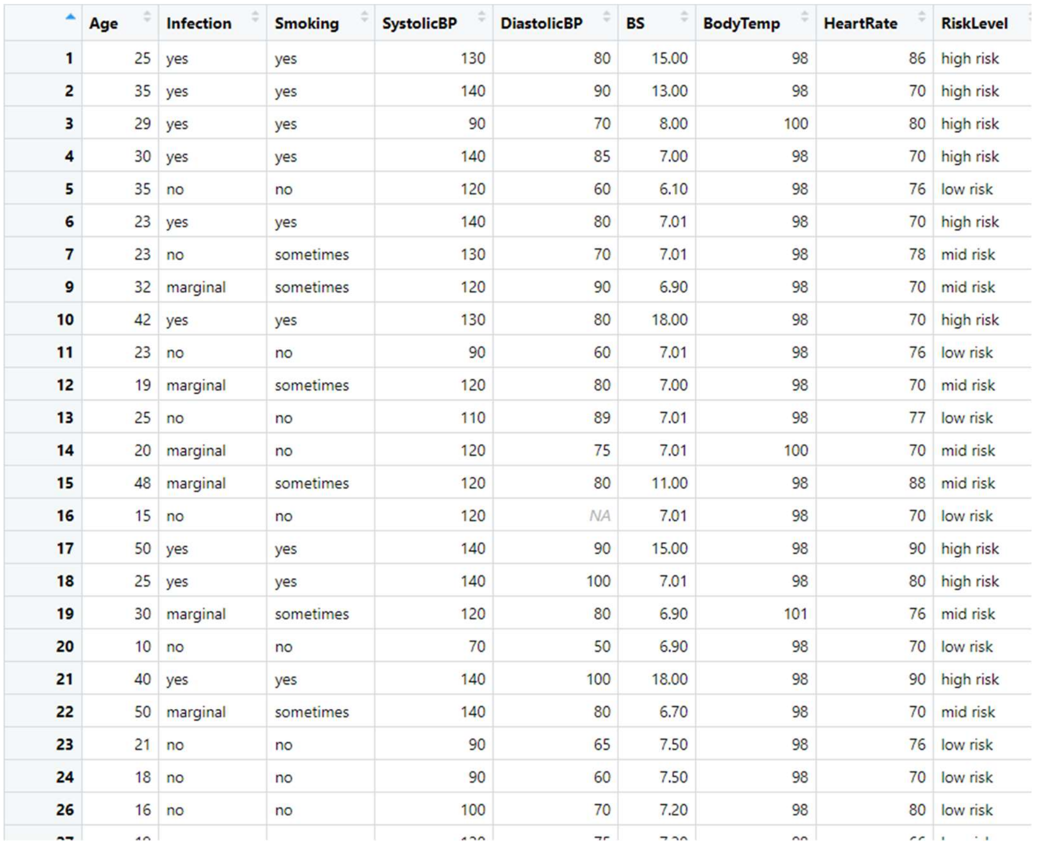
```
> data$Smoking[is.na(data$Smoking)]<-smoking_mode
> sum(is.na(data$Smoking))
[1] 0
```

We can see, now Smoking attribute does not have any missing values.

Let’s convert the Smoking attribute back to original values where 1 means yes, 2 means sometimes and 3 means no.

```
> data$Smoking<-as.character(data$Smoking)
> data$Smoking[data$Smoking=="1"]<-"yes"
> data$Smoking[data$Smoking=="2"]<-"sometimes"
> data$Smoking[data$Smoking=="3"]<-"no"
```

Now, the dataset looks like this,



	Age	Infection	Smoking	SystolicBP	DiastolicBP	BS	BodyTemp	HeartRate	RiskLevel
1	25	yes	yes	130	80	15.00	98	86	high risk
2	35	yes	yes	140	90	13.00	98	70	high risk
3	29	yes	yes	90	70	8.00	100	80	high risk
4	30	yes	yes	140	85	7.00	98	70	high risk
5	35	no	no	120	60	6.10	98	76	low risk
6	23	yes	yes	140	80	7.01	98	70	high risk
7	23	no	sometimes	130	70	7.01	98	78	mid risk
9	32	marginal	sometimes	120	90	6.90	98	70	mid risk
10	42	yes	yes	130	80	18.00	98	70	high risk
11	23	no	no	90	60	7.01	98	76	low risk
12	19	marginal	sometimes	120	80	7.00	98	70	mid risk
13	25	no	no	110	89	7.01	98	77	low risk
14	20	marginal	no	120	75	7.01	100	70	mid risk
15	48	marginal	sometimes	120	80	11.00	98	88	mid risk
16	15	no	no	120	NA	7.01	98	70	low risk
17	50	yes	yes	140	90	15.00	98	90	high risk
18	25	yes	yes	140	100	7.01	98	80	high risk
19	30	marginal	sometimes	120	80	6.90	101	76	mid risk
20	10	no	no	70	50	6.90	98	70	low risk
21	40	yes	yes	140	100	18.00	98	90	high risk
22	50	marginal	sometimes	140	80	6.70	98	70	mid risk
23	21	no	no	90	65	7.50	98	76	low risk
24	18	no	no	90	60	7.50	98	70	low risk
26	16	no	no	100	70	7.20	98	80	low risk

Figure 10: Dataset after converting the Smoking attribute in categorical type.

SystolicBP Attribute:

The systolic is an integer type attribute representing the highest level of pressure exerted by the blood against the walls of the arteries when the heart beats.

```
> class(data$SystolicBP)
[1] "integer"
```

We can see there are no missing values in the SystolicBP attribute.

```
> sum(is.na(data$SystolicBP))
[1] 0
```

There are no invalid values in the SystolicBP attribute.

```
> any(is.na(as.numeric(data$SystolicBP)))
[1] FALSE
```

Also, using the boxplot, we can see systolicBP attribute does not have any outliers.

```
> boxplot(data$SystolicBP,main="Boxplot of systolicBP",ylab="values")
```

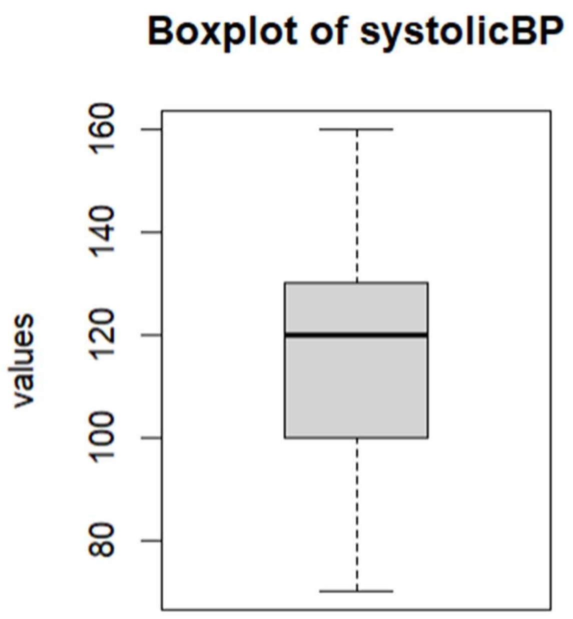


Figure 11: Boxplot of SystolicBP column.

So, the SystolicBP column is complete and ready to use from previously.

DiastolicBP Attribute:

DiastolicBP attribute is a numeric column representing lower number in a blood pressure reading and represents the pressure in the arteries when the heart is resting between beats, specifically during diastole (the relaxation phase of the cardiac cycle).

```
> class(data$DiastolicBP)
[1] "integer"
```

We can see, in the DiastolicBP attribute, there are some missing values.

```
> sum(is.na(data$DiastolicBP))
[1] 4
```

Let's recover them using the mean value of the DiastolicBP attribute.

```
> mean_diasolic<-round(mean(data$DiastolicBP,na.rm=TRUE),digits=0)
> mean_diastolic
[1] 78
> data$DiastolicBP[is.na(data$DiastolicBP)]<-mean_diastolic
```

There are no invalid value in the DiastolicBP attribute.

```
> any(is.na(as.numeric(data$DiastolicBP)))
[1] FALSE
```

We can see there are no outliers in the DiastolicBP attribute using boxploting.

```
> boxplot(data$DiastolicBP,main="Boxplot of DiastolicBP",ylab="Values")
```

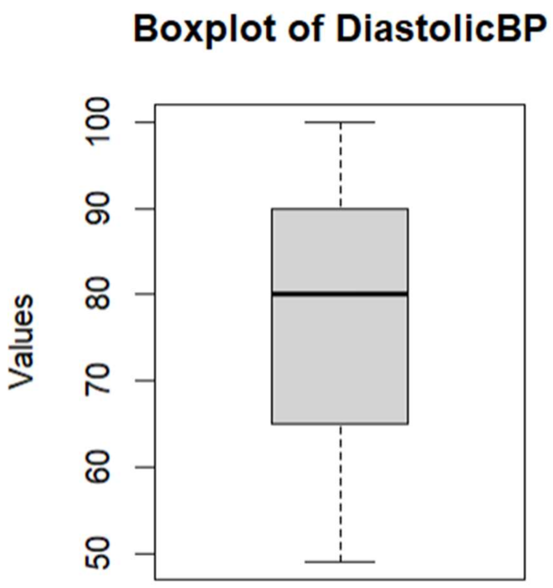


Figure 12: Boxplot of the DiastolicBP attribute.

BS Attribute:

The BS attribute is an integer type attribute which represents the blood glucose levels in terms of molar concentration(mmol/L)

```
> class(data$BS)
[1] "numeric"
```

There is no missing value in the BS attribute.

```
> sum(is.na(data$BodyTemp))
[1] 0
```

There is also no invalid value in the BS attribute.

```
> any(is.na(as.numeric(data$BS)))
[1] FALSE
```

The range of the BS attribute

```
> range(data$BS)
[1]  6 19
```

Using boxplot, we can see there are several outlier values. But we will consider them extreme values as we know blood sugar level can be at that high level range.

```
> boxplot(data$BS,main="Boxplot of BS",ylab="Values")
```

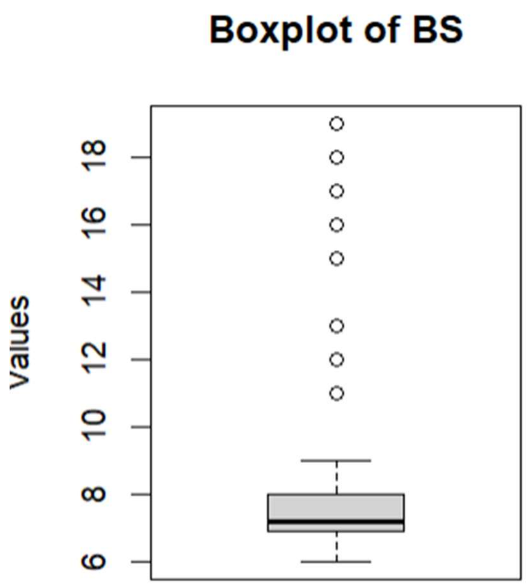
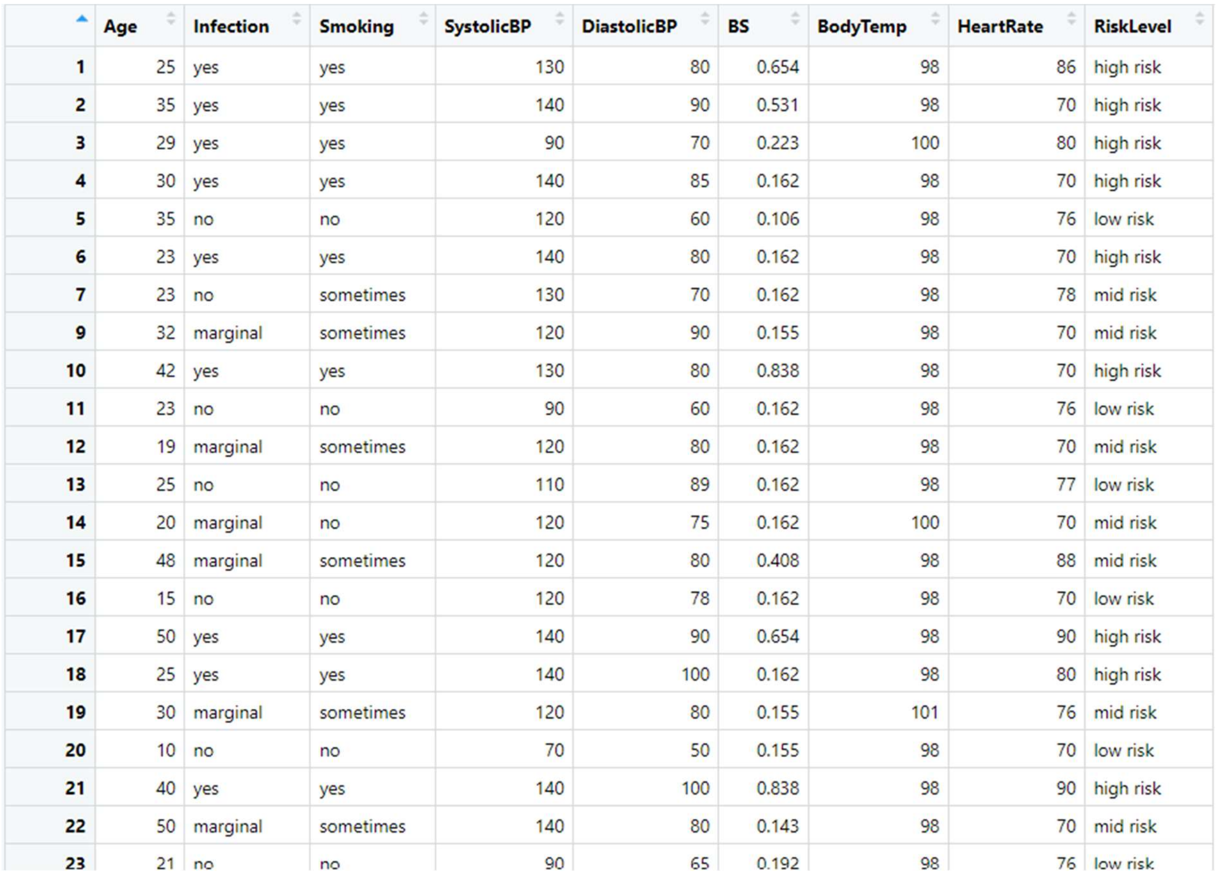


Figure 13: Boxplot of BS column.

We can use Normalization on BS attribute to prevent dominating the distance results because of large values. Let's convert each value of A, say a, to 0.1+(0.9-0.1) (a − min)/(max − min) to range them within 0.1 to 0.9.

```
> min_BS<-min(data$BS)
> max_BS<-max(data$BS)
> custom_min<-0.1
> custom_max<-0.9
> data$BS<-custom_min+(custom_max-custom_min)*(data$BS-min_BS)/(max_BS-min_BS)
> data$BS<-round(data$BS,5)
```

Now, the dataset looks like this,



	Age	Infection	Smoking	SystolicBP	DiastolicBP	BS	BodyTemp	HeartRate	RiskLevel
1	25	yes	yes	130	80	0.654	98	86	high risk
2	35	yes	yes	140	90	0.531	98	70	high risk
3	29	yes	yes	90	70	0.223	100	80	high risk
4	30	yes	yes	140	85	0.162	98	70	high risk
5	35	no	no	120	60	0.106	98	76	low risk
6	23	yes	yes	140	80	0.162	98	70	high risk
7	23	no	sometimes	130	70	0.162	98	78	mid risk
9	32	marginal	sometimes	120	90	0.155	98	70	mid risk
10	42	yes	yes	130	80	0.838	98	70	high risk
11	23	no	no	90	60	0.162	98	76	low risk
12	19	marginal	sometimes	120	80	0.162	98	70	mid risk
13	25	no	no	110	89	0.162	98	77	low risk
14	20	marginal	no	120	75	0.162	100	70	mid risk
15	48	marginal	sometimes	120	80	0.408	98	88	mid risk
16	15	no	no	120	78	0.162	98	70	low risk
17	50	yes	yes	140	90	0.654	98	90	high risk
18	25	yes	yes	140	100	0.162	98	80	high risk
19	30	marginal	sometimes	120	80	0.155	101	76	mid risk
20	10	no	no	70	50	0.155	98	70	low risk
21	40	yes	yes	140	100	0.838	98	90	high risk
22	50	marginal	sometimes	140	80	0.143	98	70	mid risk
23	21	no	no	90	65	0.192	98	76	low risk

Figure 14: Dataset after normalizing the BS attribute.

BodyTemp Attribute:

BodyTemp is a integer attribute which refers to the average temperature of the human body. It is typically measured in Fahrenheit (°F)

```
> class(data$BodyTemp)
[1] "integer"
```

We can see BodyTemp attribute does not have any missing values.

```
> sum(is.na(data$BodyTemp))
[1] 0
```

Using Boxplot, we can see BodyTemp attribute has come outliers which are negative values. We know the body temperature of a human cannot be negative.

```
> boxplot(data$BodyTemp,main="Boxplot of BodyTemp",ylab="Values")
```

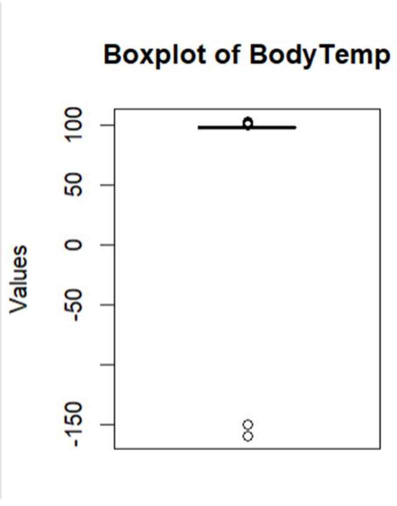


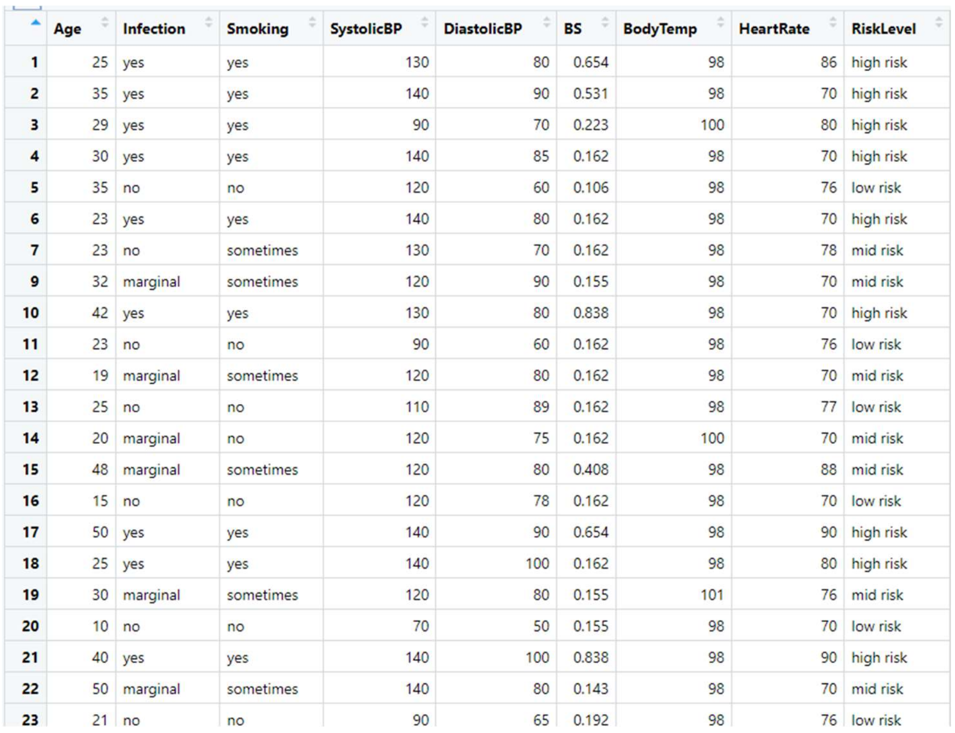
Figure 15: Boxplot of BodyTemp attribute.

Let’s recover the negative values using the mean value of the BodyTemp attribute.

```
> mean_bodytemp<-round(mean(data$BodyTemp[data$BodyTemp>=0]))
> mean_bodytemp
[1] 98
```

```
> data$BodyTemp[data$Body<0]<-mean_bodytemp
```

Now, the dataset looks like this,



	Age	Infection	Smoking	SystolicBP	DiastolicBP	BS	BodyTemp	HeartRate	RiskLevel
1	25	yes	yes	130	80	0.654	98	86	high risk
2	35	yes	yes	140	90	0.531	98	70	high risk
3	29	yes	yes	90	70	0.223	100	80	high risk
4	30	yes	yes	140	85	0.162	98	70	high risk
5	35	no	no	120	60	0.106	98	76	low risk
6	23	yes	yes	140	80	0.162	98	70	high risk
7	23	no	sometimes	130	70	0.162	98	78	mid risk
9	32	marginal	sometimes	120	90	0.155	98	70	mid risk
10	42	yes	yes	130	80	0.838	98	70	high risk
11	23	no	no	90	60	0.162	98	76	low risk
12	19	marginal	sometimes	120	80	0.162	98	70	mid risk
13	25	no	no	110	89	0.162	98	77	low risk
14	20	marginal	no	120	75	0.162	100	70	mid risk
15	48	marginal	sometimes	120	80	0.408	98	88	mid risk
16	15	no	no	120	78	0.162	98	70	low risk
17	50	yes	yes	140	90	0.654	98	90	high risk
18	25	yes	yes	140	100	0.162	98	80	high risk
19	30	marginal	sometimes	120	80	0.155	101	76	mid risk
20	10	no	no	70	50	0.155	98	70	low risk
21	40	yes	yes	140	100	0.838	98	90	high risk
22	50	marginal	sometimes	140	80	0.143	98	70	mid risk
23	21	no	no	90	65	0.192	98	76	low risk

Figure 16: Dataset after recovering the negative values of BodyTemp attribute.

HeartRate Attribute:

HeartRate attribute is an integer column which represents A normal resting heart rate in bpm.

```
> class(data$HeartRate)
[1] "integer"
```

We can see there are no missing data in HeartRate attribute.

```
> sum(is.na(data$HeartRate))
[1] 0
```

The HeartRate attribute does not have any invalid value.

```
> any(is.na(as.numeric(data$HeartRate)))
[1] FALSE
```

Using boxplot, we can see HeartRate attribute does not have any outliers.

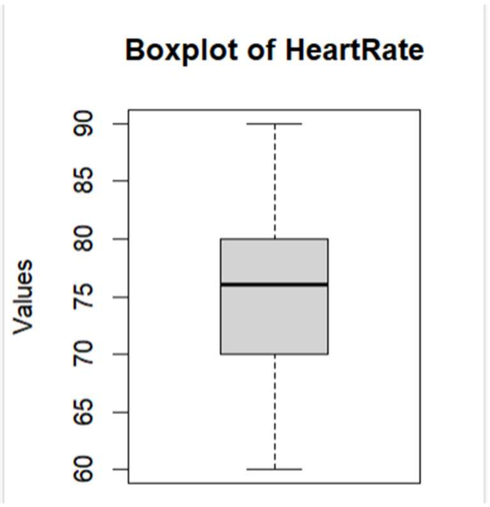


Figure 17: Boxplot of HeartRate attribute.

So, HeartRate attribute is complete and ready from previous to use.

RiskLevel Attribute:

The RiskLevel Attribute is a categorical attribute which Predicts Risk Intensity Level during pregnancy considering the previous attribute.

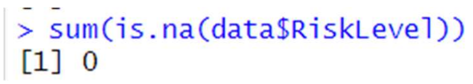
```
> class(data$RiskLevel)
[1] "character"
```

We can see the values in RiskLevel attribute are “high risk”, “low risk” and “mid risk”

```
> unique(data$RiskLevel)
[1] "high risk" "low risk"  "mid risk"
```

There are also no invalid values in RiskLevel attribute.

Futhermore, there are no missing value or invalid value in the RiskLevel attribute.

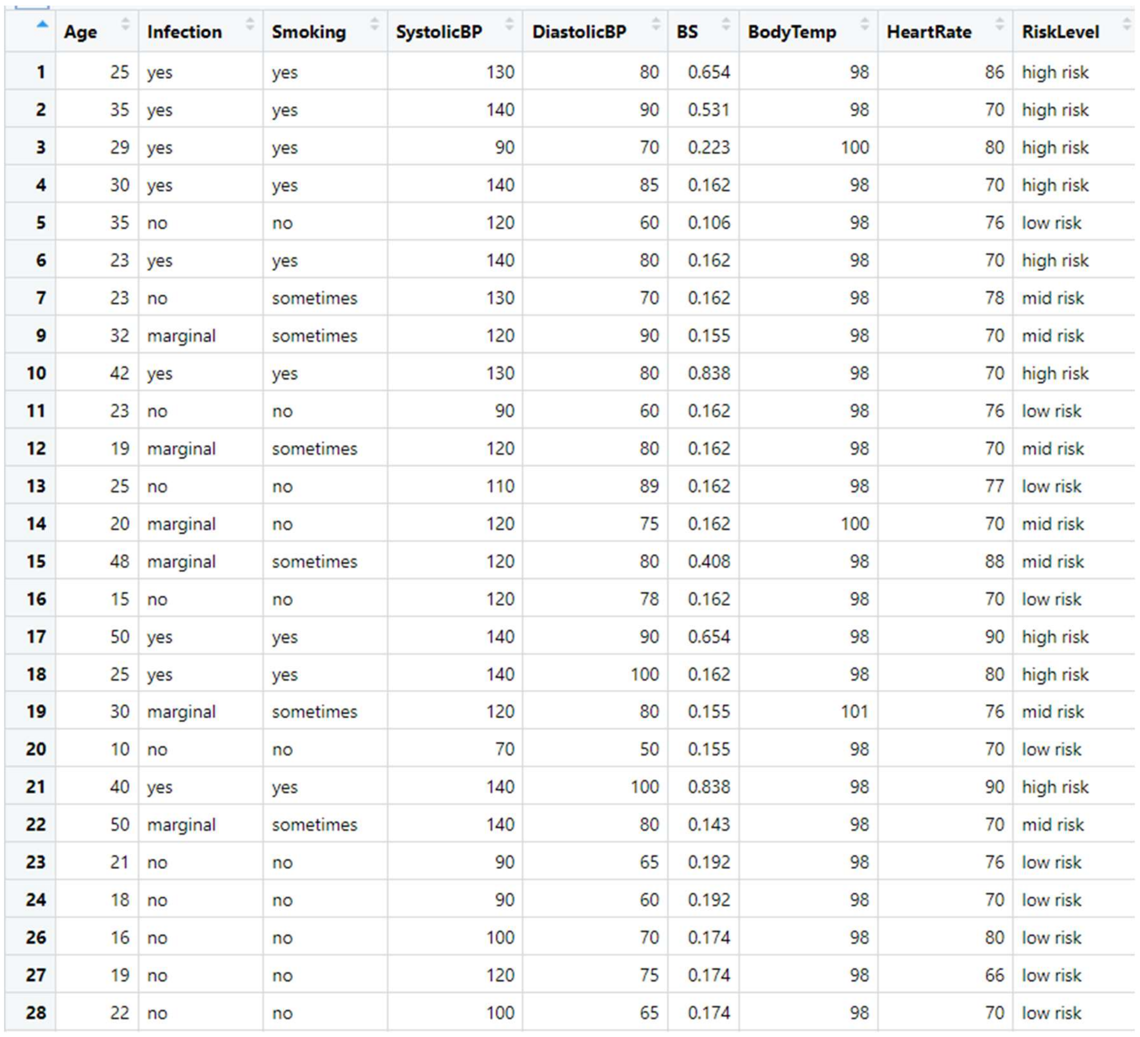


```
> sum(is.na(data$RiskLevel))
[1] 0
```

So, the RiskLevel attribute is clean and complete from previously.

Discussion:

At the beginning, we were given a dataset which had null values, invalid values and converted attribute in Figure 1. After pre-processing, we get a clean dataset. The dataset looks like this now,



	Age	Infection	Smoking	SystolicBP	DiastolicBP	BS	BodyTemp	HeartRate	RiskLevel
1	25	yes	yes	130	80	0.654	98	86	high risk
2	35	yes	yes	140	90	0.531	98	70	high risk
3	29	yes	yes	90	70	0.223	100	80	high risk
4	30	yes	yes	140	85	0.162	98	70	high risk
5	35	no	no	120	60	0.106	98	76	low risk
6	23	yes	yes	140	80	0.162	98	70	high risk
7	23	no	sometimes	130	70	0.162	98	78	mid risk
9	32	marginal	sometimes	120	90	0.155	98	70	mid risk
10	42	yes	yes	130	80	0.838	98	70	high risk
11	23	no	no	90	60	0.162	98	76	low risk
12	19	marginal	sometimes	120	80	0.162	98	70	mid risk
13	25	no	no	110	89	0.162	98	77	low risk
14	20	marginal	no	120	75	0.162	100	70	mid risk
15	48	marginal	sometimes	120	80	0.408	98	88	mid risk
16	15	no	no	120	78	0.162	98	70	low risk
17	50	yes	yes	140	90	0.654	98	90	high risk
18	25	yes	yes	140	100	0.162	98	80	high risk
19	30	marginal	sometimes	120	80	0.155	101	76	mid risk
20	10	no	no	70	50	0.155	98	70	low risk
21	40	yes	yes	140	100	0.838	98	90	high risk
22	50	marginal	sometimes	140	80	0.143	98	70	mid risk
23	21	no	no	90	65	0.192	98	76	low risk
24	18	no	no	90	60	0.192	98	70	low risk
26	16	no	no	100	70	0.174	98	80	low risk
27	19	no	no	120	75	0.174	98	66	low risk
28	22	no	no	100	65	0.174	98	70	low risk

Figure 18: Dataset after pre-processing.

Let’s view the data distribution of the pre-processed dataset.

```
> ggplot(data,aes(x=RiskLevel))+geom_bar()
> ggplot(data,aes(x=Smoking))+geom_bar()
> ggplot(data,aes(x=Infection))+geom_bar()
> ggplot(data,aes(x=RiskLevel))+geom_bar()
> for(col in names(data)[sapply(data,is.numeric)] ){
+   hist(data[[col]], main=paste(col,"Distribution"),xlab=col)
+ }
```

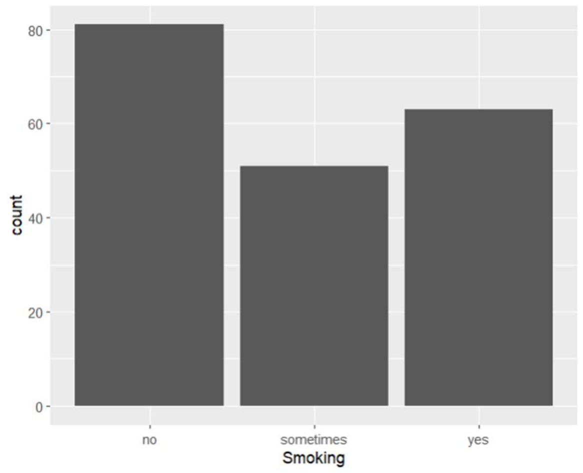


Figure 19: Data distribution of Smoking Attribute.

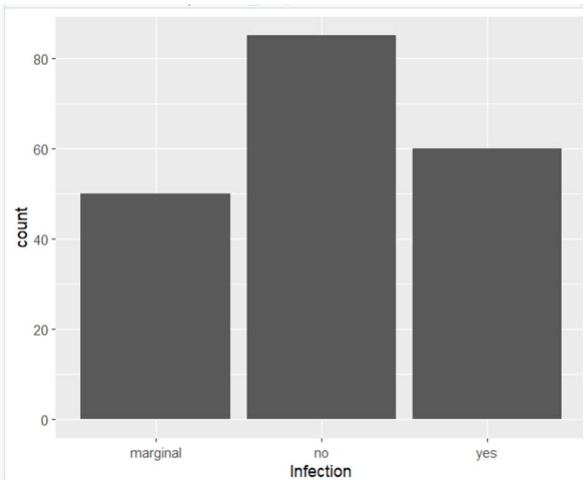


Figure 20: Data distribution of Infection Attribute.

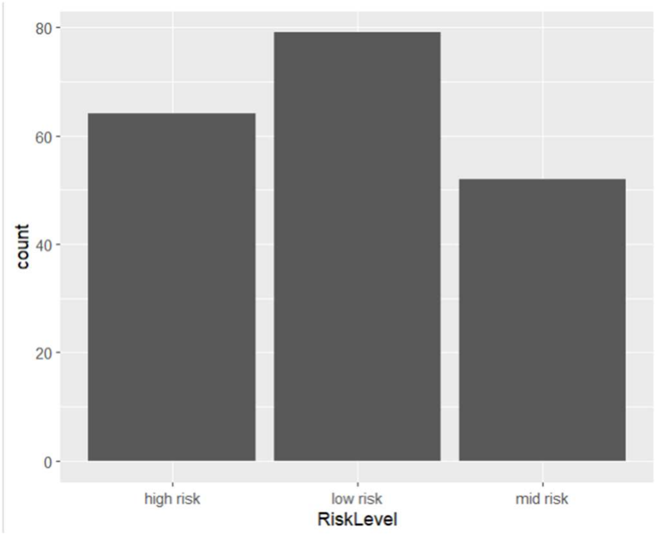


Figure 21: Data distribution of RiskLevel attribute.

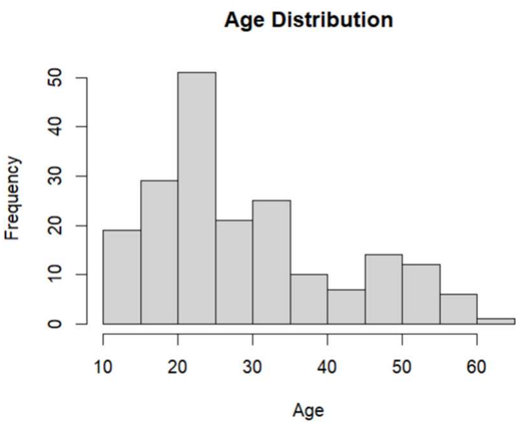


Figure 22: Data distribution of Age Attribute.

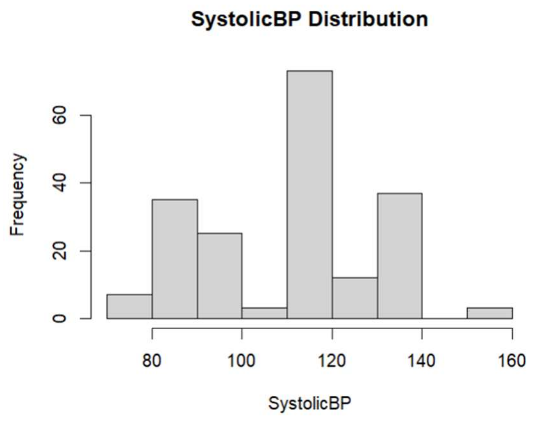


Figure 23: Data distribution of SystolicBP Attribute.

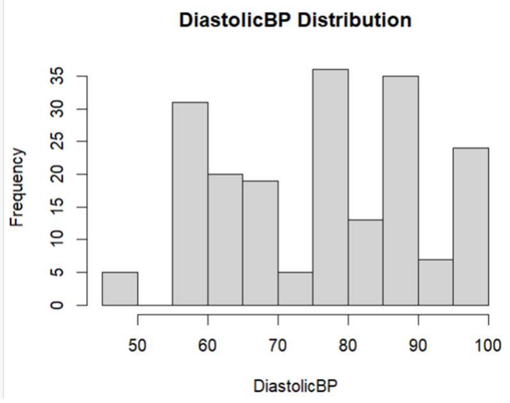


Figure 24: Data distribution of DiastolicBP Attribute.

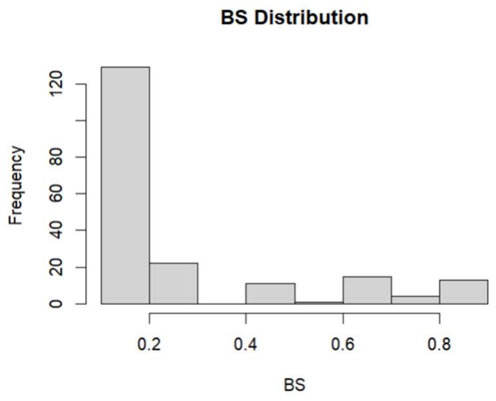


Figure 25: Data distribution of BS Attribute.

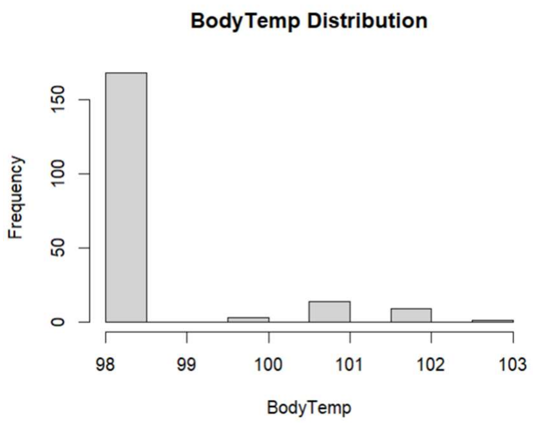


Figure 26: Data distribution of BodyTemp Attribute.

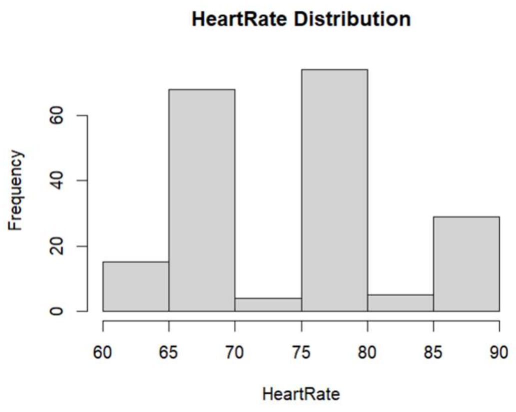


Figure 27: Data distribution of HeartRate Attribute.