Diabetes Prediction and Visualization

Mini Project Report submitted in partial fulfillment.

of the requirement for the degree of

B. E. (Information Technology)

Submitted By

Tanaya Desai - 18101B0059
Shraddha Babar - 18101B0065
Siddhi Suryavanshi - 18101B0071

Under the Guidance of

Prof. Shruti Agrawal

Department of Information Technology

Vidyalankar Institute of Technology

Wadala(E), Mumbai 400 037

University of Mumbai

2021-22

CERTIFICATE OF APPROVAL

For

Mini Project Report

On

R Programming Lab

This is to Certify that

Tanaya Desai
Shraddha Babar
Siddhi Suryavanshi

Have successfully carried out Mini Project entitled

"**Diabetes Prediction and Visualization**"

In partial fulfillment of degree course in

Information Technology

As laid down by University of Mumbai during the academic year 2020-21

Under the Guidance of
Prof. Shruti Agrawal

Signature of Guide   Head of Department

Examiner 1   Examiner 2

ACKNOWLEGEMENT

We would like to express our deepest appreciation to all those who provided us the possibility to complete this report. We express our profound gratitude we give to our **Prof. Shruti Agrawal** Ma’am, our respectable project guide, for her gigantic support and guidance. Without her counseling our project would not have seen the light of the day.

We extend our sincere thanks to **Dr. Vipul Dalal**, Head of the Department of Information Technology for offering valuable advice at every stage of this undertaking. We would like to thank all the staff members who willingly helped us. We are grateful to VIDYALANKAR INSTITUTE OF TECHNOLOGY for giving us this opportunity.

The days we have spent in the institute will always be remembered and also bereckoned as guiding in our career.

1. **Tanaya Desai**
2. **Shraddha Babar**
3. **Siddhi Suryavanshi**

Table of Contents

Abstract

Over time, having too much glucose in blood can cause health issues. Diabetes occurs when your blood glucose or sugar is too high. One in six people with diabetes in the world is from India. With the development of standards of living, diabetes is gradually increasing in people. Diabetes is one of the major international health problems.

World Health Organization reports says that around 422 million people have diabetes worldwide. Data mining plays a huge role in predicting diabetes in the healthcare industry. Therefore, a convenient way to predict diabetics is essential so that necessary protocols can be taken beforehand.

In this project, a diabetes prediction and visualization is implemented for predicting diabetes which comprises of some external features for diabetes alongside regular features like Insulin, Glucose, BMI, Age, etc. The prediction model, which produces highly accurate results, applied and compared various algorithms like Decision Tree, Linear Regression and Naive Bayes to determineDiabetes.

There are many algorithms developed for prediction of diabetes. But most of the algorithms failed in case of the accuracy estimation. Also, there is a need to automate the overall process of diabetes prediction. This automation of diabetic database helps in identification of impact of diabetes on various human organs. More the accuracy of prediction, more the chances of accurate severity estimation. Therefore, this project concentrated on providing different prediction methods of diabetes.

1. Introduction

As the advancement is pushing, contraptions are making tremendous proportion of data each. There is an overall emission in the openness of data for researchers. The complexity, colossal size and heterogeneity of data anticipate that one should look, find and take on new programming mechanical assemblies and parts to successfully make due, inspect, and picture the data. Medical data examination needs an advancement that helps with playing out a continuous assessment on the colossal dataset. In Medical data industry the use of perceptive assessment are basically high. Assumptions can be made concerning patients, which patients, districts or geographic will be affected by some infection. As a result of these applications in clinical benefits industry data investigation has gotten a huge proportion of income from investigators in past two or three years.

Diabetes is an amazingly typical infection. According to the National Diabetes Statics Report, beginning at 2015, 30.3 million people in the United States had diabetes, which infers one of ten people in the United States is encountering diabetes. Also, one out of ten of them don't understand they have the disease. It is similarly a steady disease affecting the individual fulfillment conflictingly, since most patients ought to oversee diabetes reliably and it can provoke issues impacting basically everybody structures. In this manner, it is vital to hinder and examine the disease. In any case, the clinical resource is confined, and experts can make investigate for explicit number of patients in the limited time.

In this project, a diabetes prediction and visualization is implemented for predicting diabetes which comprises of some external features for diabetes alongside regular features like Insulin, Glucose, BMI, Age, etc. The prediction model, which produces highly accurate results, applied and compared various algorithms like Decision Tree, Linear Regression and Naive Bayes to determine Diabetes.

Decision Tree

The Supervised learning strategy, which is utilized for taking care of arrangement issues. Decision tree is a method which iteratively breaks the given dataset into at least two example information. The objective of the technique is to foresee the class worth of the objective variable. The decision tree will assist with isolating the informational collection and fabricates the choice model to anticipate the obscure class marks. A contribution to the Decision tree is a dataset, comprising of a few credits and occasions esteems and result will be the choice model. Issues confronted while building a choice model are choosing the parting characteristic, parts, halting rules, and pruning, preparing test, quality and amount, the request for parts and so on.

Naive Bayes

Naive Bayes is a Supervised Machine Learning algorithm based on the Bayes Theorem that is used to solve classification problems by following a probabilistic approach. It is based on the idea that the predictor variables in a Machine Learning model are independent of each other. Meaning that the outcome of a model depends on a set of independent variables that have nothing to do with each other.

Linear regression

It is a regression model that uses a straight line to describe the relationship between variables. It finds the line of best fit through your data by searching for the value of the regression coefficient(s) that minimizes the total error of the model.

2. Aim and Objectives

The aim of the project is to predict diabetes by implementing various algorithms like Decision Tree, Linear Regression and Naive Bayes which comprises of some external features for diabetes alongside regular features like Insulin, Glucose, BMI, Age, etc. The prediction model, which produces highly accurate results, applied and compared to determine Diabetes.

The core objectives are

1. Predict if person is diabetes patient or not
2. Find most indicative features of diabetes
3. Try different classification methods to find highest accuracy

3. Problem Definition

Diabetes is an amazingly typical infection. According to the National Diabetes Statics Report, beginning at 2015, 30.3 million people in the United States had diabetes, which infers one of ten people in the United States is encountering diabetes. Also, one out of ten of them don't understand they have the disease. It is similarly a steady disease affecting the individual fulfillment conflictingly, since most patients ought to oversee diabetes reliably and it can provoke issues impacting basically everybody structures. In this manner, it is vital to hinder and examine the disease. In any case, the clinical resource is confined, and experts can make investigate for explicit number of patients in the limited time.

Over time, having too much glucose in blood can cause health issues. Diabetes occurs when your blood glucose or sugar is too high. One in six people with diabetes in the world is from India. With the development of standards of living, diabetes is gradually increasing in people. Diabetes is one of the major international health problems. So, we decided to predict diabetes using R.

In this project, a diabetes prediction and visualization is implemented for predicting diabetes which comprises of some external features for diabetes alongside regular features like Insulin, Glucose, BMI, Age, etc. The prediction model, which produces highly accurate results, applied and compared various algorithms like Decision Tree, Linear Regression and Naive Bayes to determine Diabetes.

4. Dataset Used

This dataset is originally from the National Institute of Diabetes and Digestive and Kidney Diseases. The objective is to predict based on diagnostic measurements whether a patient has diabetes.

The data set is taken from UCI machine learning repository. The data set consists of 9 attributes: number of times pregnant, plasma glucose concentration, diastolic blood pressure, triceps skin folds thickness, serum insulin, body mass index, pedigree type, age and class. Here, the class label is binary classification. It has two values

- Tested positive (1) which means diabetic
- Tested negative (0) which saysnondiabetic

Several constraints were placed on the selection of these instances from a larger database. In particular, all patients here are females at least 21 years old of Pima Indian heritage.

- Pregnancies: Number of times pregnant
- Glucose: Plasma glucose concentration a 2 hours in an oral glucose tolerance test
- BloodPressure: Diastolic blood pressure (mm Hg)
- SkinThickness: Triceps skin fold thickness (mm)
- Insulin: 2-Hour serum insulin (mu U/ml)
- BMI: Body mass index (weight in kg/(height in m)^2)
- DiabetesPedigreeFunction: Diabetes pedigree function
- Age: Age (years)
- Outcome: Class variable (0 or 1)

5. Components

5.1 Hardware

5.1.1 Laptop / Desktop

A laptop computer is smaller than a desktop computer, generally less than three inches thick, and weigh less than desktop computers. The laptop's size makes it convenient for transportation in briefcases, backpacks, and other bags.

The device derives its name from being able to be used by resting on a person's lap without the need for a desk or other surface. Laptop computers may also be referred to as notebook computers, though a notebook computer usually describes a computer that is smaller and lighter than a laptop computer.

5.2 Software

5.2.1 RStudio

RStudio is a powerful and easy way to interact with R programming, considered as Integrated Development Environment (IDE) that provides a one-stop solution for all the statistical computing and graphics. The RStudio is a more advanced version of R that comes with a multi-pane window setup that provides access to all primary things on a single screen (such as source, console, environment & history, files, photos, graphs, etc).

5.2.2 Libraries Used

1. **Library ggplot2**

ggplot2 is an R package which is designed especially for data visualization and providing best exploratory data analysis. It provides beautiful, hassle-free plots that take care of minute details like drawing legends and representing them. The plots can be created iteratively and edited later. This package is designed to work in a layered fashion, starting with a layer showing the raw data collected during exploratory data analysis with R then adding layers of annotations and statistical summaries.

2. **Library dplyr**

R has a library called dplyr to help in data transformation. The dplyr library is fundamentally created around four functions to manipulate the data and five verbs to clean the data. After that, we can use the ggplot library to analyze and visualize the data. dplyr is a grammar of data manipulation, providing a consistent set of verbs that help you solve the most common data manipulation challenges:

mutate() : adds new variables that are functions of existing variables
select() : picks variables based on their names.
filter() : picks cases based on their values.
summarise() : reduces multiple values down to a single summary.
arrange() : changes the ordering of the rows.

3. **Library grid**

grid package uses functions circleGrob, linesGrob, polygonGrob, rasterGrob, rectGrob, segmentsGrob, legendGrob, xaxisGrob, and yaxisGrob to create graphical objects (grobs). This package is a basis for all other higher graphical functions used in other packages such as lattice, ggplot2, etc. Also, it can manipulate the lattice outputs.

4. **Library gridExtra**

The gridExtra package provides useful extensions to the grid system, with an emphasis on higher-level functions to work with grid graphic objects, rather than the lower-level utilities in the grid package that are used to create and edit specific lower-level elements of a plot. This package has particularly useful functions that allow you to arrange and write multiple grobs to a graphics device and to include tables in grid graphics objects.

5. Library corrplot

R package corrplot provides a visual exploratory tool on correlation matrix that supports automatic variable reordering to help detect hidden patterns among variables.
corrplot is very easy to use and provides a rich array of plotting options in visualization method, graphic layout, color, legend, text labels, etc. It also provides p-values and confidence intervals to help users determine the statistical significance of the correlations.

6. Library caret

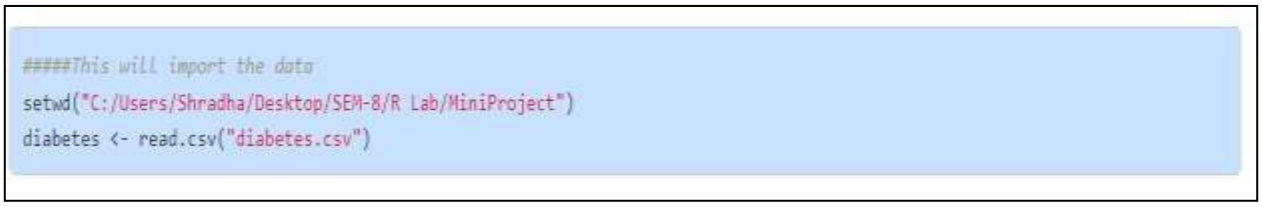
Caret stands for classification and regression training and is arguably the biggest project in R. This package is sufficient to solve almost any classification or regression machine learning problem. It supports approximately 200 machine learning algorithms and makes it easy to perform critical tasks such as data preparation, data cleaning, feature selection, and model validation.

7. Library e1071

e1071 is a package for R programming that provides functions for statistic and probabilistic algorithms like a fuzzy classifier, naive Bayes classifier, bagged clustering, short-time Fourier transform, support vector machine, etc. When it comes to SVM, there are many packages available in R to implement it. However, e1071 is the most intuitive package for this purpose.The svm() function of the e1071 package provides a robust interface in the form of the libsvm. This interface makes implementing SVM's very quick and simple. It also facilitates probabilistic classification by using the kernel trick. It provides the most common kernels like linear, RBF, sigmoid, and polynomial.

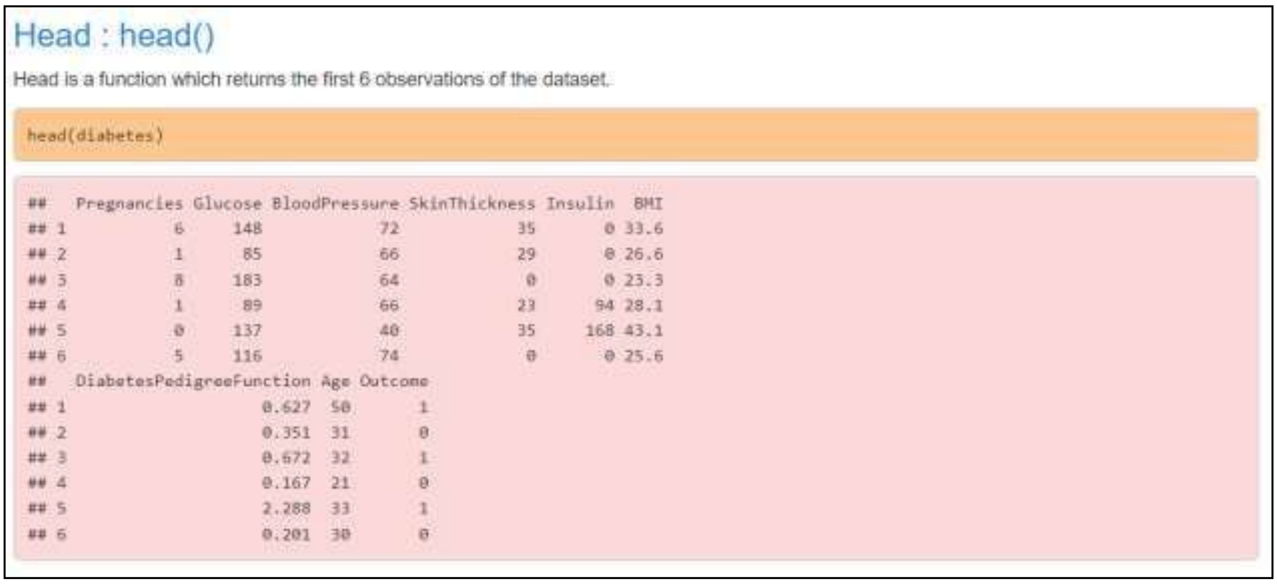
6. Code & Output

1. Importing Diabetes Data



```
#####This will import the data
setwd("C:/Users/Shradha/Desktop/SEM-8/R Lab/MiniProject")
diabetes <- read.csv("diabetes.csv")
```

2. Exploratory Data Analysis on Diabetes Data

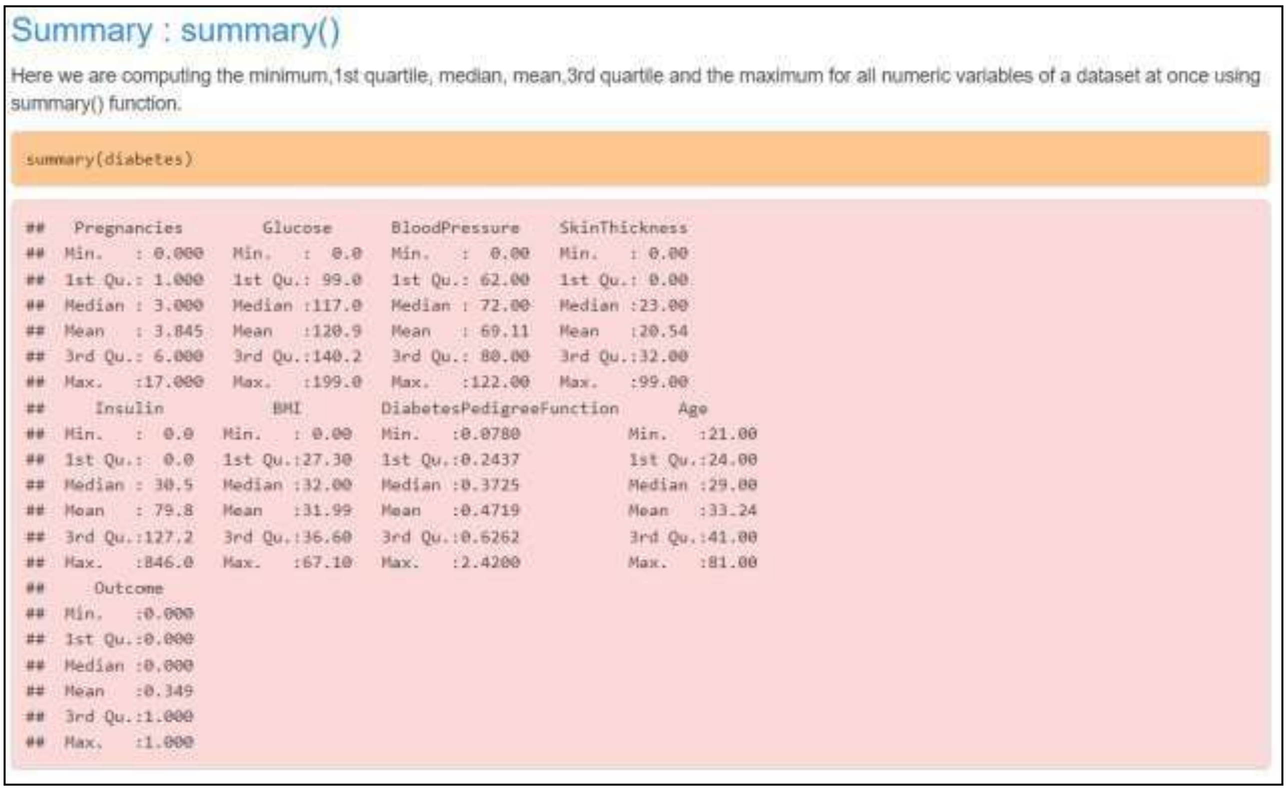


Head : head()

Head is a function which returns the first 6 observations of the dataset.

```
head(diabetes)
```

```
##   Pregnancies Glucose BloodPressure SkinThickness Insulin  BMI
## 1           6     148            72            35       0 33.6
## 2           1      85            66            29       0 26.6
## 3           8     183            64             0       0 23.3
## 4           1      89            66            23      94 28.1
## 5           0     137            40            35     168 43.1
## 6           5     116            74             0       0 25.6
##   DiabetesPedigreeFunction Age Outcome
## 1                    0.627  50       1
## 2                    0.351  31       0
## 3                    0.672  32       1
## 4                    0.167  21       0
## 5                    2.288  33       1
## 6                    0.201  30       0
```

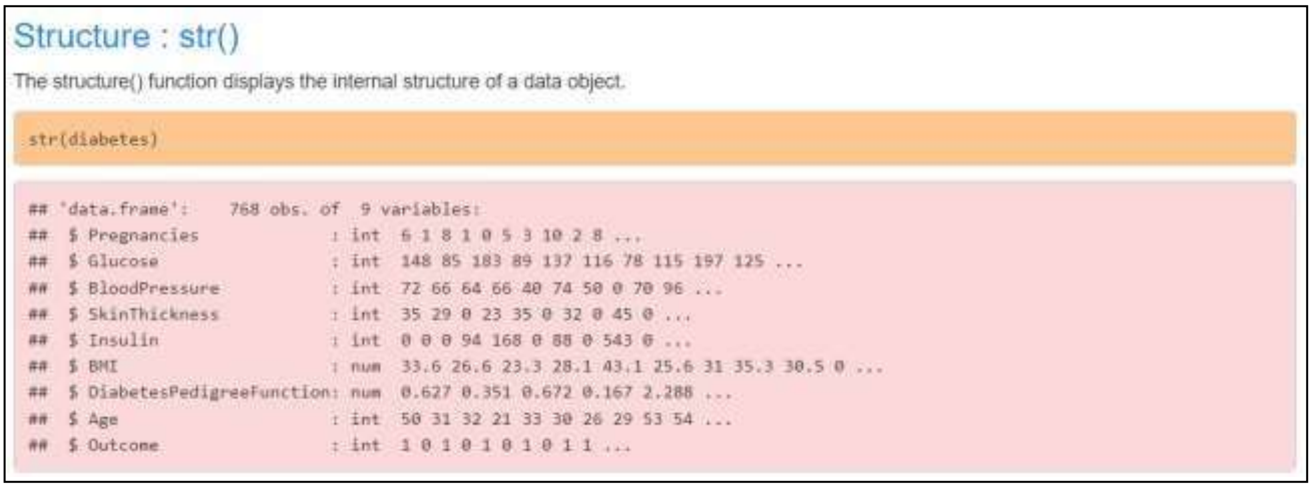


Summary : summary()

Here we are computing the minimum,1st quartile, median, mean,3rd quartile and the maximum for all numeric variables of a dataset at once using summary() function.

```
summary(diabetes)
```

```
##   Pregnancies        Glucose      BloodPressure    SkinThickness
##  Min.   : 0.000   Min.   :  0.0   Min.   :  0.00   Min.   : 0.00
##  1st Qu.: 1.000   1st Qu.: 99.0   1st Qu.: 62.00   1st Qu.: 0.00
##  Median : 3.000   Median :117.0   Median : 72.00   Median :23.00
##  Mean   : 3.845   Mean   :120.9   Mean   : 69.11   Mean   :20.54
##  3rd Qu.: 6.000   3rd Qu.:140.2   3rd Qu.: 80.00   3rd Qu.:32.00
##  Max.   :17.000   Max.   :199.0   Max.   :122.00   Max.   :99.00
##     Insulin           BMI        DiabetesPedigreeFunction      Age
##  Min.   :  0.0   Min.   : 0.00   Min.   :0.0780           Min.   :21.00
##  1st Qu.:  0.0   1st Qu.:27.30   1st Qu.:0.2437           1st Qu.:24.00
##  Median : 30.5   Median :32.00   Median :0.3725           Median :29.00
##  Mean   : 79.8   Mean   :31.99   Mean   :0.4719           Mean   :33.24
##  3rd Qu.:127.2   3rd Qu.:36.60   3rd Qu.:0.6262           3rd Qu.:41.00
##  Max.   :846.0   Max.   :67.10   Max.   :2.4200           Max.   :81.00
##     Outcome
##  Min.   :0.000
##  1st Qu.:0.000
##  Median :0.000
##  Mean   :0.349
##  3rd Qu.:1.000
##  Max.   :1.000
```

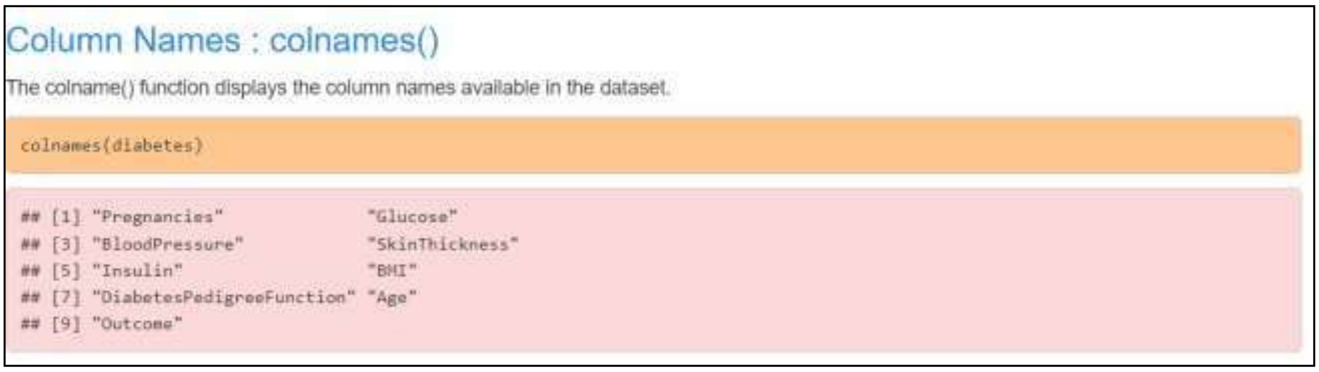


Structure : str()

The structure() function displays the internal structure of a data object.

```
str(diabetes)
```

```
## 'data.frame':    768 obs. of  9 variables:
##  $ Pregnancies             : int  6 1 8 1 0 5 3 10 2 8 ...
##  $ Glucose                 : int  148 85 183 89 137 116 78 115 197 125 ...
##  $ BloodPressure           : int  72 66 64 66 40 74 50 0 70 96 ...
##  $ SkinThickness           : int  35 29 0 23 35 0 32 0 45 0 ...
##  $ Insulin                 : int  0 0 0 94 168 0 88 0 543 0 ...
##  $ BMI                     : num  33.6 26.6 23.3 28.1 43.1 25.6 31 35.3 30.5 0 ...
##  $ DiabetesPedigreeFunction: num  0.627 0.351 0.672 0.167 2.288 ...
##  $ Age                     : int  50 31 32 21 33 30 26 29 53 54 ...
##  $ Outcome                 : int  1 0 1 0 1 0 1 0 1 1 ...
```

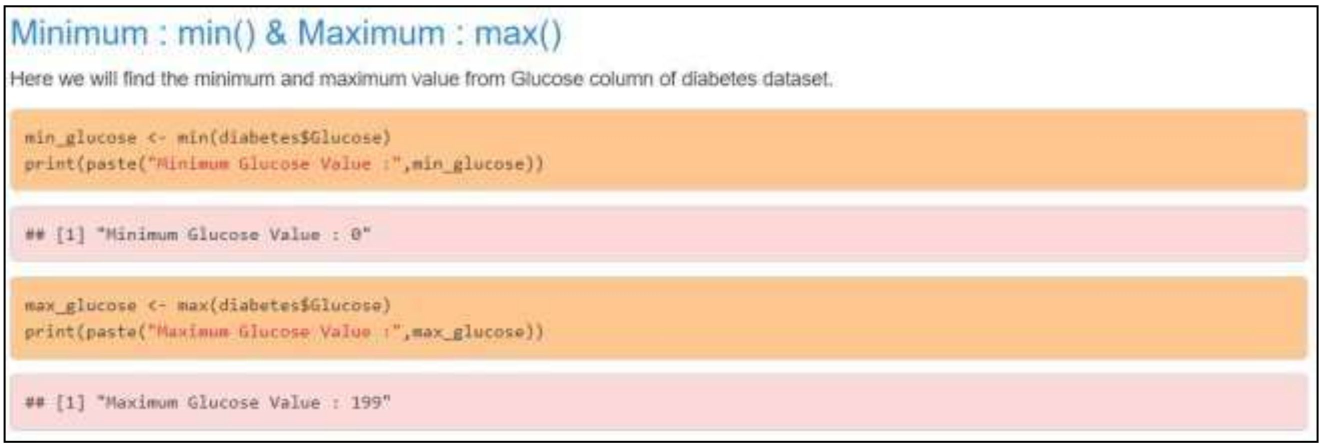


Column Names : colnames()

The colname() function displays the column names available in the dataset.

```
colnames(diabetes)
```

```
## [1] "Pregnancies"              "Glucose"
## [3] "BloodPressure"            "SkinThickness"
## [5] "Insulin"                  "BMI"
## [7] "DiabetesPedigreeFunction" "Age"
## [9] "Outcome"
```



Minimum : min() & Maximum : max()

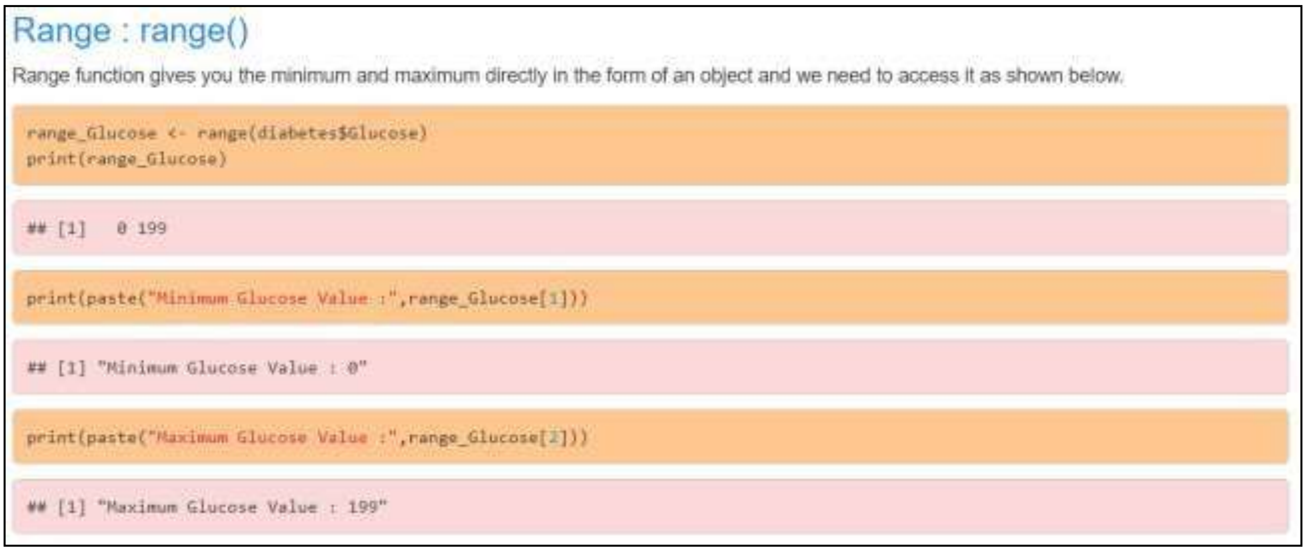
Here we will find the minimum and maximum value from Glucose column of diabetes dataset.

```
min_glucose <- min(diabetes$Glucose)
print(paste("Minimum Glucose Value :",min_glucose))
```

```
## [1] "Minimum Glucose Value : 0"
```

```
max_glucose <- max(diabetes$Glucose)
print(paste("Maximum Glucose Value :",max_glucose))
```

```
## [1] "Maximum Glucose Value : 199"
```



Range : range()

Range function gives you the minimum and maximum directly in the form of an object and we need to access it as shown below.

```
range_Glucose <- range(diabetes$Glucose)
print(range_Glucose)
```

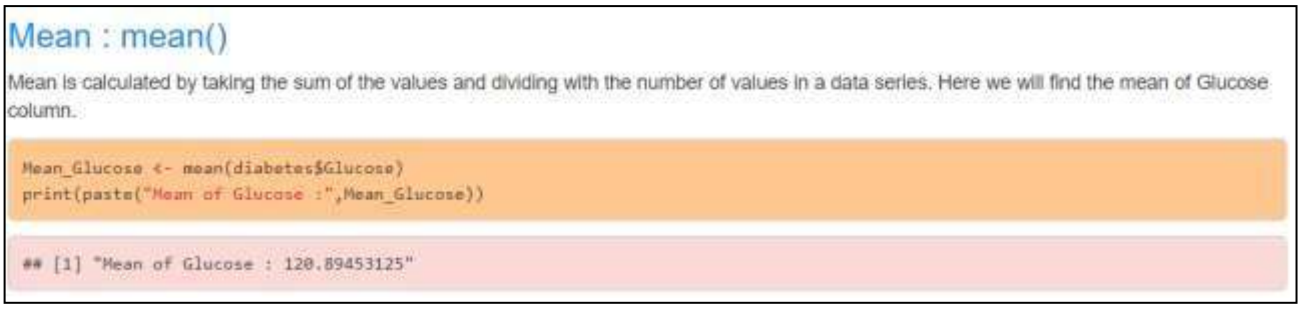
```
## [1]   0 199
```

```
print(paste("Minimum Glucose Value :",range_Glucose[1]))
```

```
## [1] "Minimum Glucose Value : 0"
```

```
print(paste("Maximum Glucose Value :",range_Glucose[2]))
```

```
## [1] "Maximum Glucose Value : 199"
```

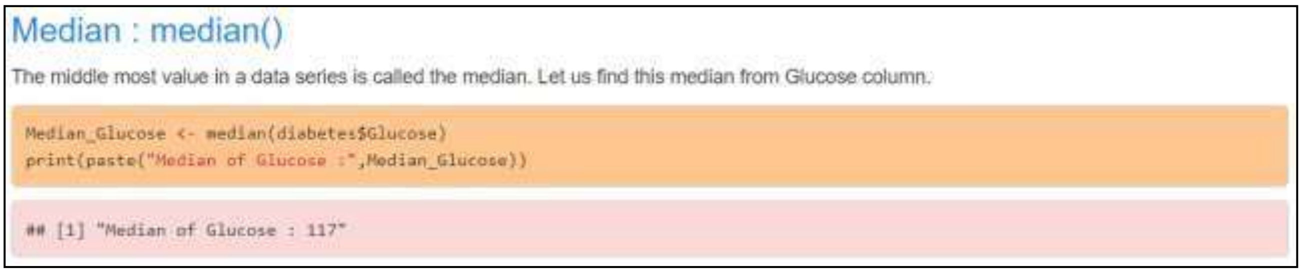


Mean : mean()

Mean is calculated by taking the sum of the values and dividing with the number of values in a data series. Here we will find the mean of Glucose column.

```
Mean_Glucose <- mean(diabetes$Glucose)
print(paste("Mean of Glucose :",Mean_Glucose))
```

```
## [1] "Mean of Glucose : 120.89453125"
```

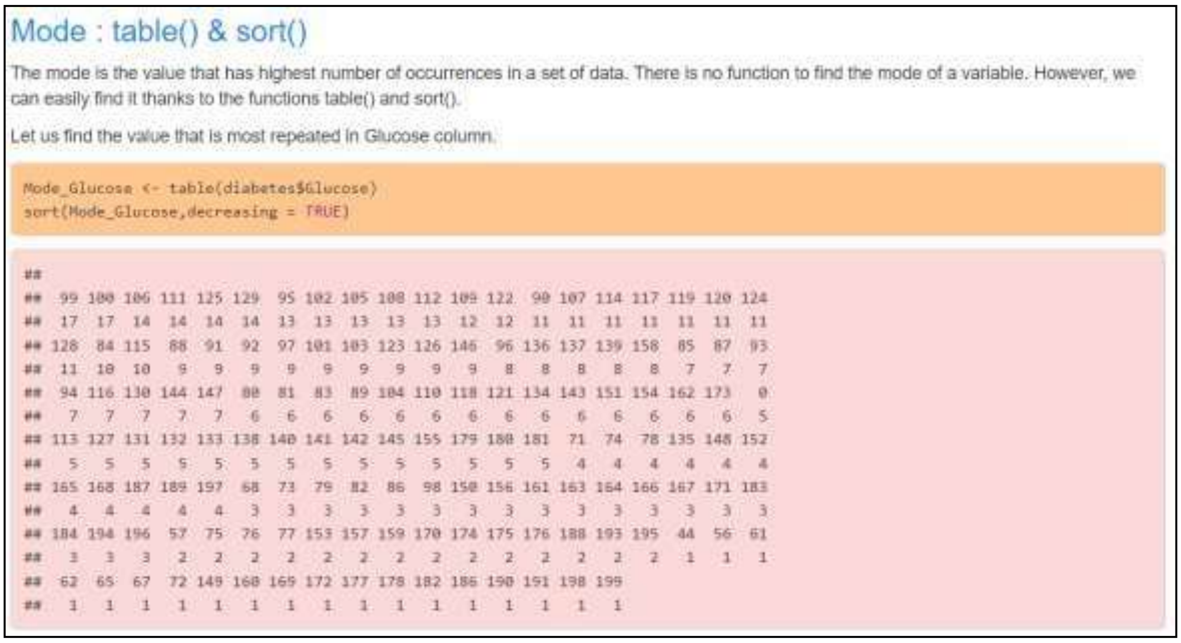


Median : median()

The middle most value in a data series is called the median. Let us find this median from Glucose column.

```
Median_Glucose <- median(diabetes$Glucose)
print(paste("Median of Glucose :",Median_Glucose))
```

```
## [1] "Median of Glucose : 117"
```



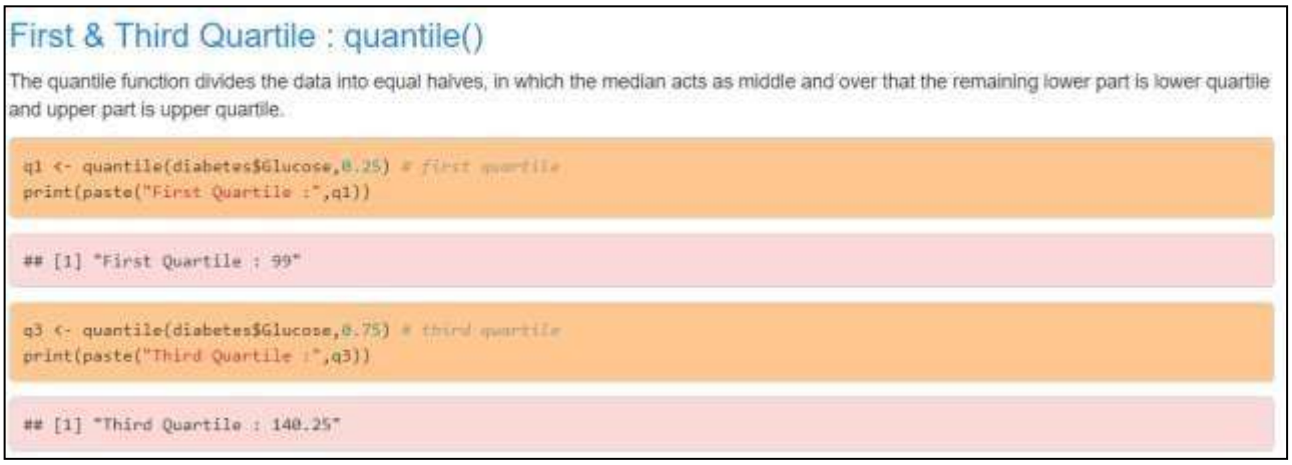
Mode : table() & sort()

The mode is the value that has highest number of occurrences in a set of data. There is no function to find the mode of a variable. However, we can easily find it thanks to the functions table() and sort().

Let us find the value that is most repeated in Glucose column.

```
Mode_Glucose <- table(diabetes$Glucose)
sort(Mode_Glucose,decreasing = TRUE)
```

```
##
##  99 100 106 111 125 129  95 102 105 108 112 109 122  90 107 114 117 119 120 124
##  17  17  14  14  14  14  13  13  13  13  13  12  12  11  11  11  11  11  11  11
## 128  84 115  88  91  92  97 101 103 123 126 146  96 136 137 139 158  85  87  93
##  11  10  10   9   9   9   9   9   9   9   9   9   8   8   8   8   8   7   7   7
##  94 116 130 144 147  80  81  83  89 104 110 118 121 134 143 151 154 162 173   0
##   7   7   7   7   7   6   6   6   6   6   6   6   6   6   6   6   6   6   6   5
## 113 127 131 132 133 138 140 141 142 145 155 179 180 181  71  74  78 135 148 152
##   5   5   5   5   5   5   5   5   5   5   5   5   5   5   4   4   4   4   4   4
## 165 168 187 189 197  68  73  79  82  86  98 150 156 161 163 164 166 167 171 183
##   4   4   4   4   4   3   3   3   3   3   3   3   3   3   3   3   3   3   3   3
## 184 194 196  57  75  76  77 153 157 159 170 174 175 176 188 193 195  44  56  61
##   3   3   3   2   2   2   2   2   2   2   2   2   2   2   2   2   2   1   1   1
##  62  65  67  72 149 160 169 172 177 178 182 186 190 191 198 199
##   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1
```



First & Third Quartile : quantile()

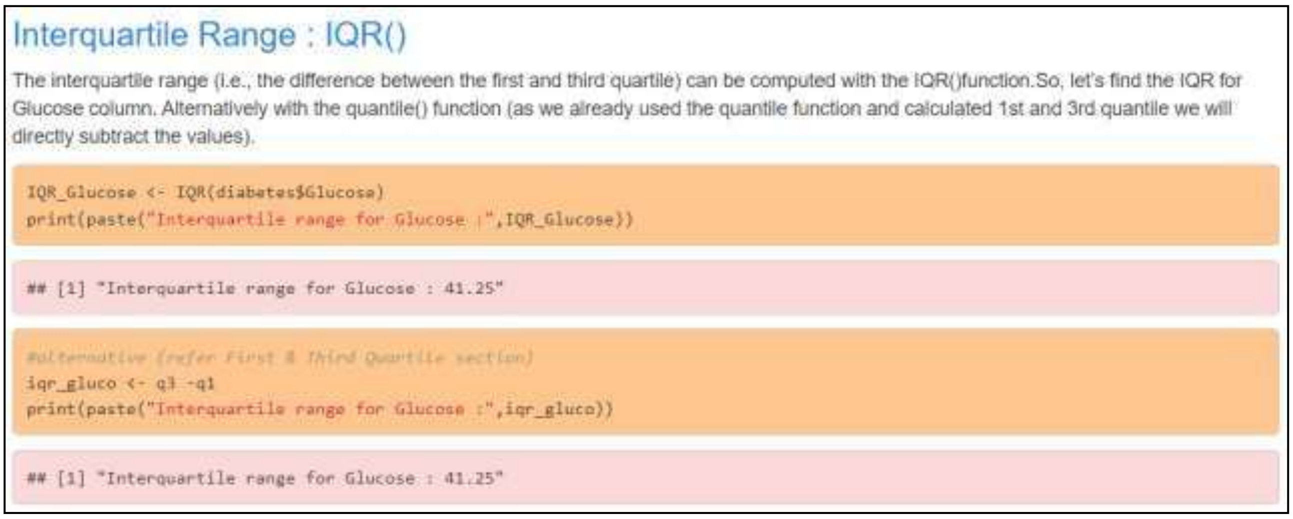
The quantile function divides the data into equal halves, in which the median acts as middle and over that the remaining lower part is lower quartile and upper part is upper quartile.

```
q1 <- quantile(diabetes$Glucose,0.25) # first quartile
print(paste("First Quartile :",q1))
```

```
## [1] "First Quartile : 99"
```

```
q3 <- quantile(diabetes$Glucose,0.75) # third quartile
print(paste("Third Quartile :",q3))
```

```
## [1] "Third Quartile : 140.25"
```



Interquartile Range : IQR()

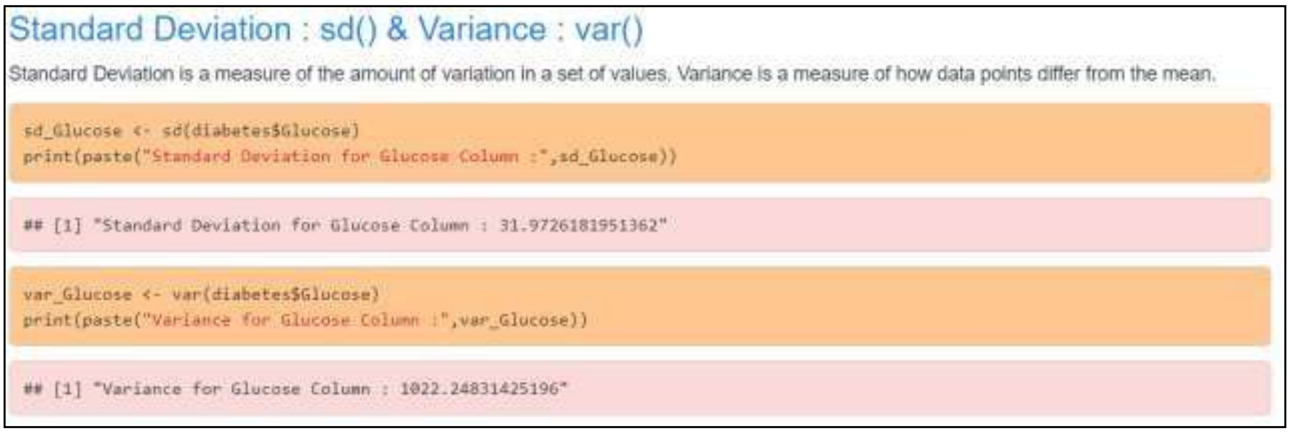
The interquartile range (i.e., the difference between the first and third quartile) can be computed with the IQR()function.So, let's find the IQR for Glucose column. Alternatively with the quantile() function (as we already used the quantile function and calculated 1st and 3rd quantile we will directly subtract the values).

```
IQR_Glucose <- IQR(diabetes$Glucose)
print(paste("Interquartile range for Glucose :",IQR_Glucose))
```

```
## [1] "Interquartile range for Glucose : 41.25"
```

```
#alternative (refer First & Third Quartile section)
iqr_gluco <- q3 -q1
print(paste("Interquartile range for Glucose :",iqr_gluco))
```

```
## [1] "Interquartile range for Glucose : 41.25"
```



Standard Deviation : sd() & Variance : var()

Standard Deviation is a measure of the amount of variation in a set of values. Variance is a measure of how data points differ from the mean.

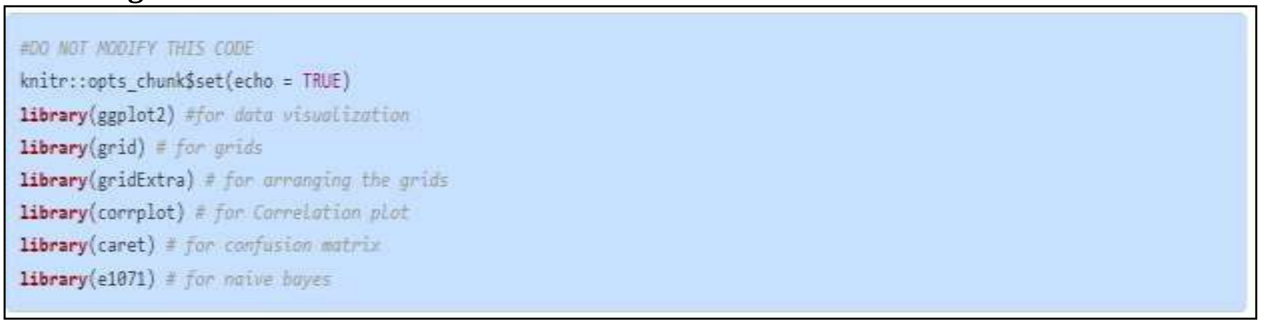
```
sd_Glucose <- sd(diabetes$Glucose)
print(paste("Standard Deviation for Glucose Column :",sd_Glucose))
```

```
## [1] "Standard Deviation for Glucose Column : 31.9726181951362"
```

```
var_Glucose <- var(diabetes$Glucose)
print(paste("Variance for Glucose Column :",var_Glucose))
```

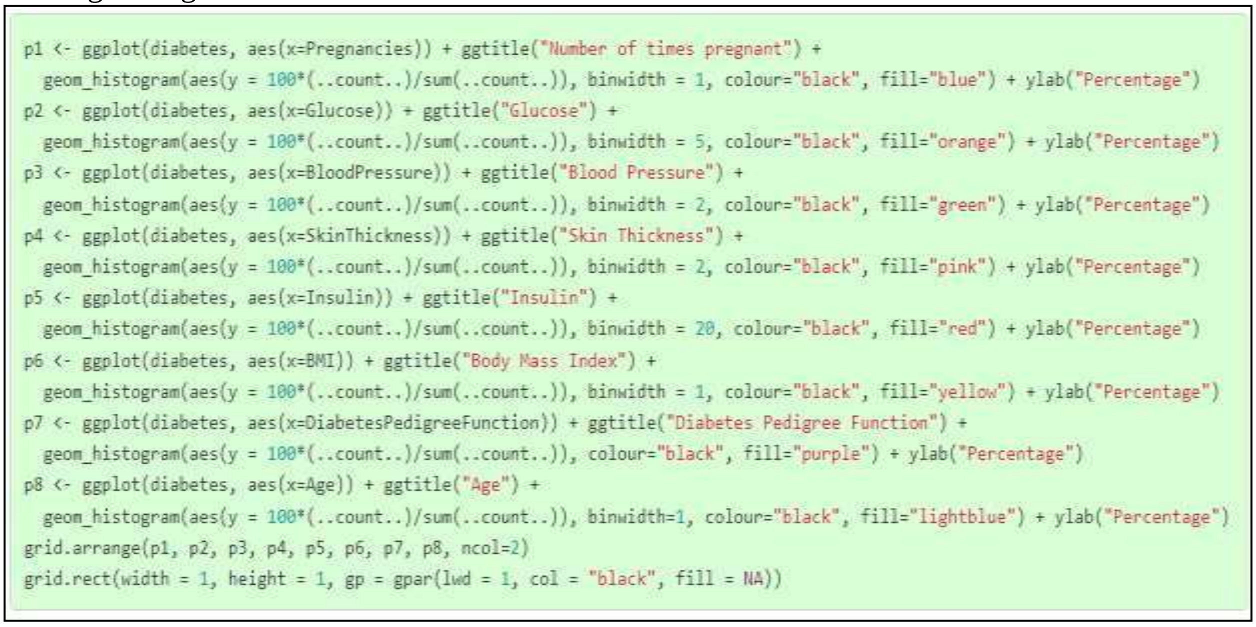
```
## [1] "Variance for Glucose Column : 1022.24831425196"
```

3. Predicting Diabetes

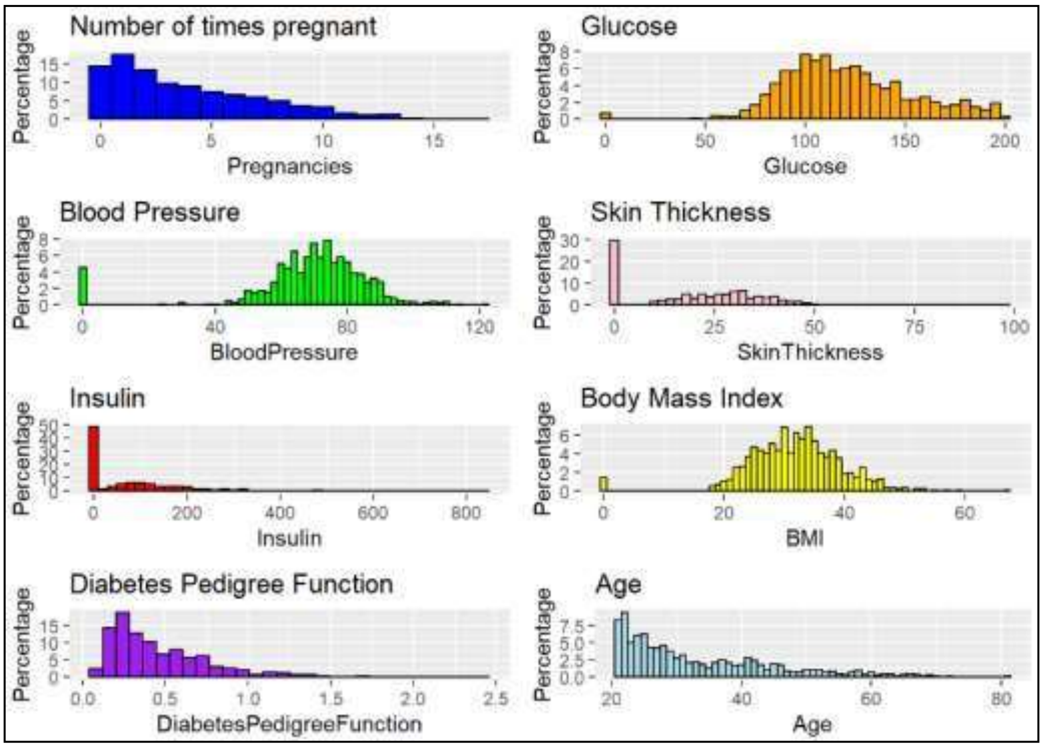


```
#DO NOT MODIFY THIS CODE
knitr::opts_chunk$set(echo = TRUE)
library(ggplot2) #for data visualization
library(grid) # for grids
library(gridExtra) # for arranging the grids
library(corrplot) # for Correlation plot
library(caret) # for confusion matrix
library(e1071) # for naive bayes
```

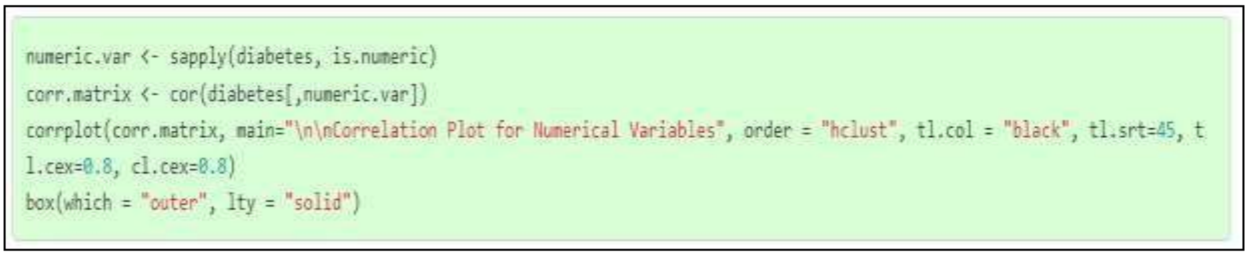
➢ Plotting Histograms of Numeric Values



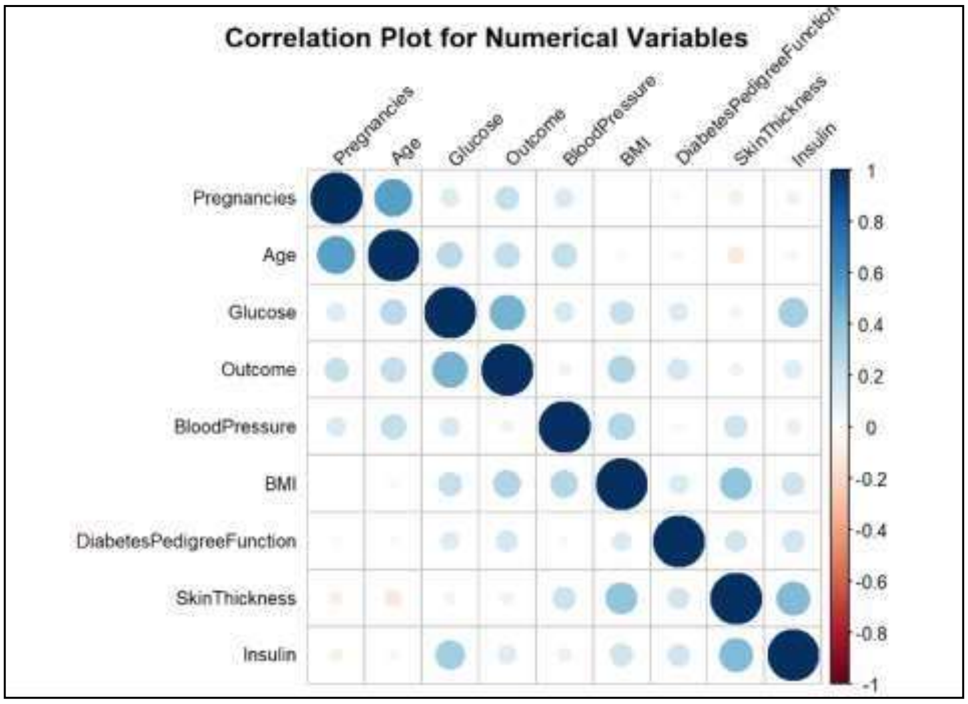
```
p1 <- ggplot(diabetes, aes(x=Pregnancies)) + ggtitle("Number of times pregnant") +
  geom_histogram(aes(y = 100*(..count..)/sum(..count..)), binwidth = 1, colour="black", fill="blue") + ylab("Percentage")
p2 <- ggplot(diabetes, aes(x=Glucose)) + ggtitle("Glucose") +
  geom_histogram(aes(y = 100*(..count..)/sum(..count..)), binwidth = 5, colour="black", fill="orange") + ylab("Percentage")
p3 <- ggplot(diabetes, aes(x=BloodPressure)) + ggtitle("Blood Pressure") +
  geom_histogram(aes(y = 100*(..count..)/sum(..count..)), binwidth = 2, colour="black", fill="green") + ylab("Percentage")
p4 <- ggplot(diabetes, aes(x=SkinThickness)) + ggtitle("Skin Thickness") +
  geom_histogram(aes(y = 100*(..count..)/sum(..count..)), binwidth = 2, colour="black", fill="pink") + ylab("Percentage")
p5 <- ggplot(diabetes, aes(x=Insulin)) + ggtitle("Insulin") +
  geom_histogram(aes(y = 100*(..count..)/sum(..count..)), binwidth = 20, colour="black", fill="red") + ylab("Percentage")
p6 <- ggplot(diabetes, aes(x=BMI)) + ggtitle("Body Mass Index") +
  geom_histogram(aes(y = 100*(..count..)/sum(..count..)), binwidth = 1, colour="black", fill="yellow") + ylab("Percentage")
p7 <- ggplot(diabetes, aes(x=DiabetesPedigreeFunction)) + ggtitle("Diabetes Pedigree Function") +
  geom_histogram(aes(y = 100*(..count..)/sum(..count..)), colour="black", fill="purple") + ylab("Percentage")
p8 <- ggplot(diabetes, aes(x=Age)) + ggtitle("Age") +
  geom_histogram(aes(y = 100*(..count..)/sum(..count..)), binwidth=1, colour="black", fill="lightblue") + ylab("Percentage")
grid.arrange(p1, p2, p3, p4, p5, p6, p7, p8, ncol=2)
grid.rect(width = 1, height = 1, gp = gpar(lwd = 1, col = "black", fill = NA))
```



➢ Correlation between Numeric Variables

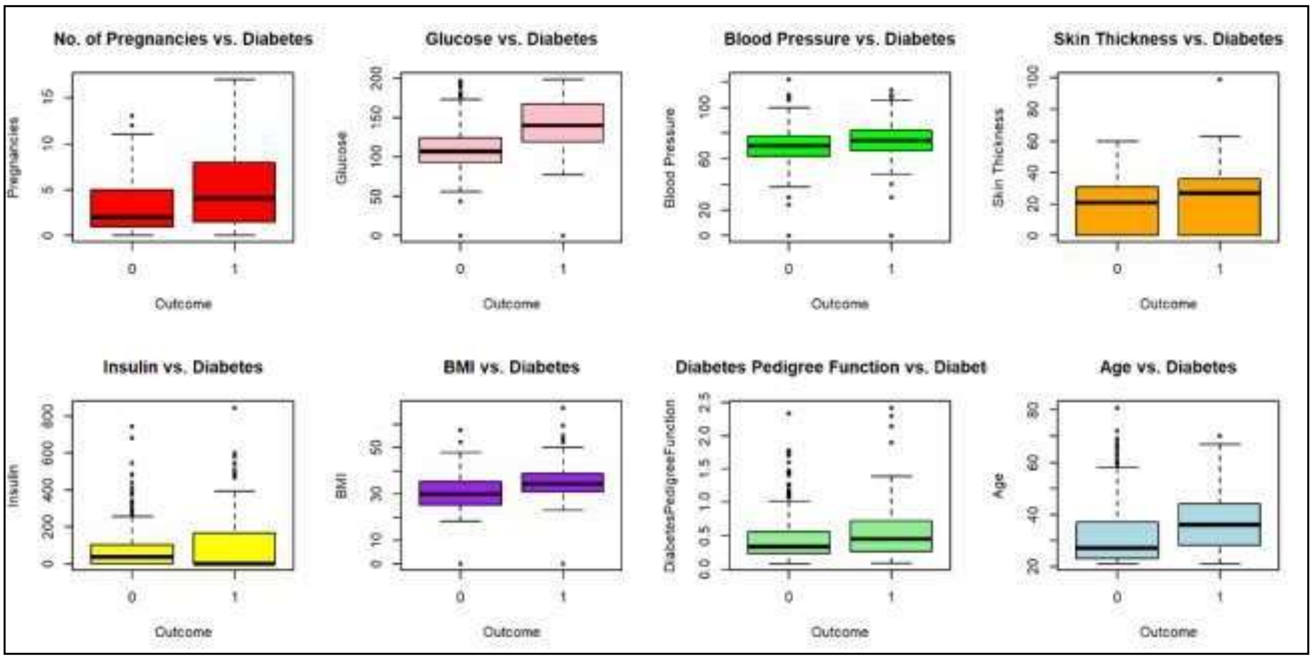


```
numeric.var <- sapply(diabetes, is.numeric)
corr.matrix <- cor(diabetes[,numeric.var])
corrplot(corr.matrix, main="\n\nCorrelation Plot for Numerical Variables", order = "hclust", tl.col = "black", tl.srt=45, t
l.cex=0.8, cl.cex=0.8)
box(which = "outer", lty = "solid")
```



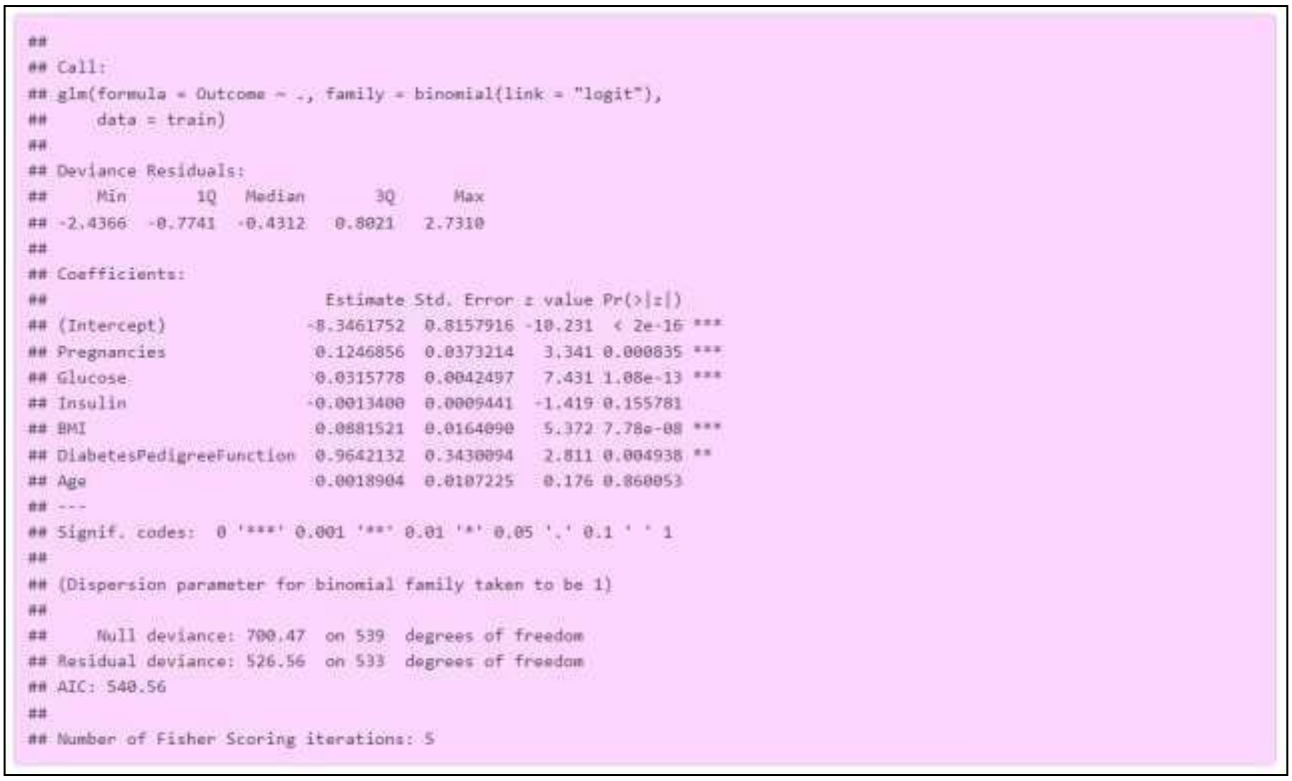
➢ Correlation between Numeric Variables and Outcomes

```
attach(diabetes)
par(mfrow=c(2,4))
boxplot(Pregnancies~Outcome, main="No. of Pregnancies vs. Diabetes",
        xlab="Outcome", ylab="Pregnancies",col="red")
boxplot(Glucose~Outcome, main="Glucose vs. Diabetes",
        xlab="Outcome", ylab="Glucose",col="pink")
boxplot(BloodPressure~Outcome, main="Blood Pressure vs. Diabetes",
        xlab="Outcome", ylab="Blood Pressure",col="green")
boxplot(SkinThickness~Outcome, main="Skin Thickness vs. Diabetes",
        xlab="Outcome", ylab="Skin Thickness",col="orange")
boxplot(Insulin~Outcome, main="Insulin vs. Diabetes",
        xlab="Outcome", ylab="Insulin",col="yellow")
boxplot(BMI~Outcome, main="BMI vs. Diabetes",
        xlab="Outcome", ylab="BMI",col="purple")
boxplot(DiabetesPedigreeFunction~Outcome, main="Diabetes Pedigree Function vs. Diabetes", xlab="Outcome", ylab="DiabetesPedi
greeFunction",col="lightgreen")
boxplot(Age~Outcome, main="Age vs. Diabetes",
        xlab="Outcome", ylab="Age",col="lightblue")
box(which = "outer", lty = "solid")
```



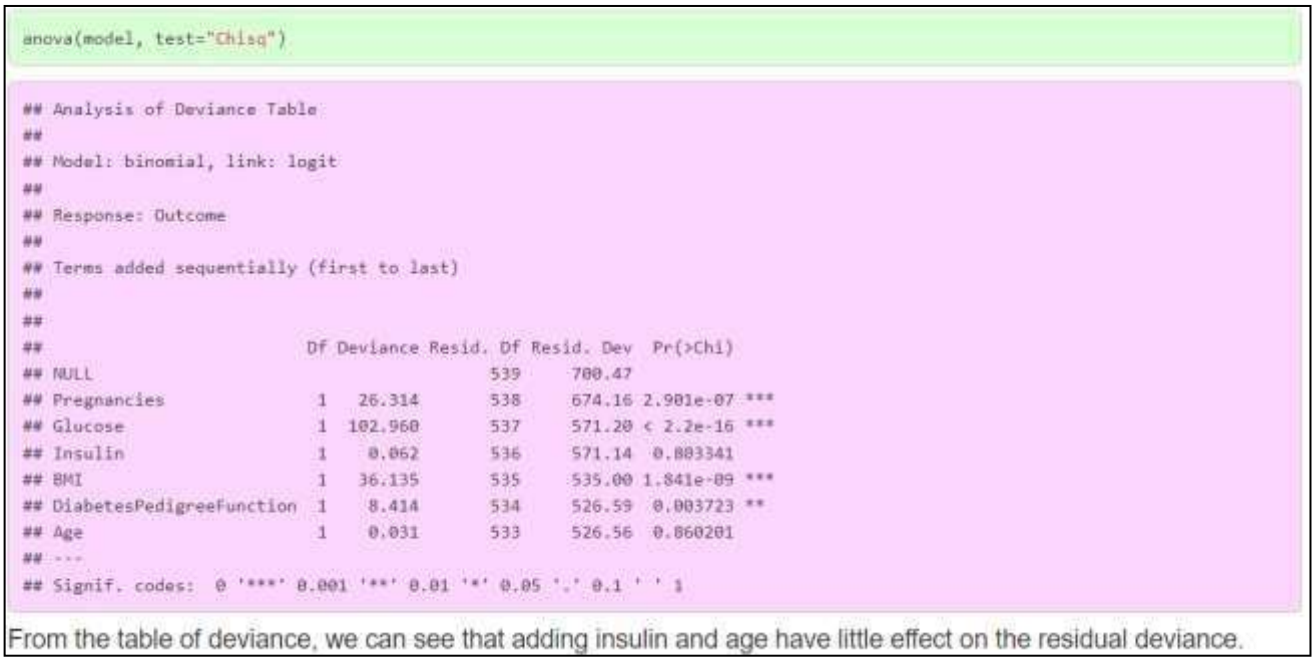
➢ Linear Regression

```
diabetes$BloodPressure <- NULL
diabetes$SkinThickness <- NULL
train <- diabetes[1:540,]
test <- diabetes[541:768,]
model <-glm(Outcome ~.,family=binomial(link='logit'),data=train)
summary(model)
```



```
##
## Call:
## glm(formula = Outcome ~ ., family = binomial(link = "logit"),
##     data = train)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.4366  -0.7741  -0.4312   0.8021   2.7310
##
## Coefficients:
##                           Estimate Std. Error z value Pr(>|z|)
## (Intercept)              -8.3461752  0.8157916 -10.231  < 2e-16 ***
## Pregnancies               0.1246856  0.0373214   3.341 0.000835 ***
## Glucose                   0.0315778  0.0042497   7.431 1.08e-13 ***
## Insulin                  -0.0013400  0.0009441  -1.419 0.155781
## BMI                       0.0881521  0.0164090   5.372 7.78e-08 ***
## DiabetesPedigreeFunction  0.9642132  0.3430094   2.811 0.004938 **
## Age                       0.0018904  0.0107225   0.176 0.860053
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 700.47  on 539  degrees of freedom
## Residual deviance: 526.56  on 533  degrees of freedom
## AIC: 540.56
##
## Number of Fisher Scoring iterations: 5
```

The top three most relevant features are “Glucose”, “BMI” and “Number of times pregnant” because of the low p-values.
“Insulin” and “Age” appear not statistically significant.



```
anova(model, test="Chisq")
```

```
## Analysis of Deviance Table
##
## Model: binomial, link: logit
##
## Response: Outcome
##
## Terms added sequentially (first to last)
##
##
##                          Df Deviance Resid. Df Resid. Dev  Pr(>Chi)
## NULL                                       539     700.47
## Pregnancies               1   26.314       538     674.16 2.901e-07 ***
## Glucose                   1  102.960       537     571.20 < 2.2e-16 ***
## Insulin                   1    0.062       536     571.14  0.803341
## BMI                       1   36.135       535     535.00 1.841e-09 ***
## DiabetesPedigreeFunction  1    8.414       534     526.59  0.003723 **
## Age                       1    0.031       533     526.56  0.860201
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

From the table of deviance, we can see that adding insulin and age have little effect on the residual deviance.

➢ Cross Validation

```
fitted.results <- predict(model,newdata=test,type='response')
fitted.results <- ifelse(fitted.results > 0.5,1,0)
(conf_matrix_logi<-table(fitted.results, test$Outcome))
```

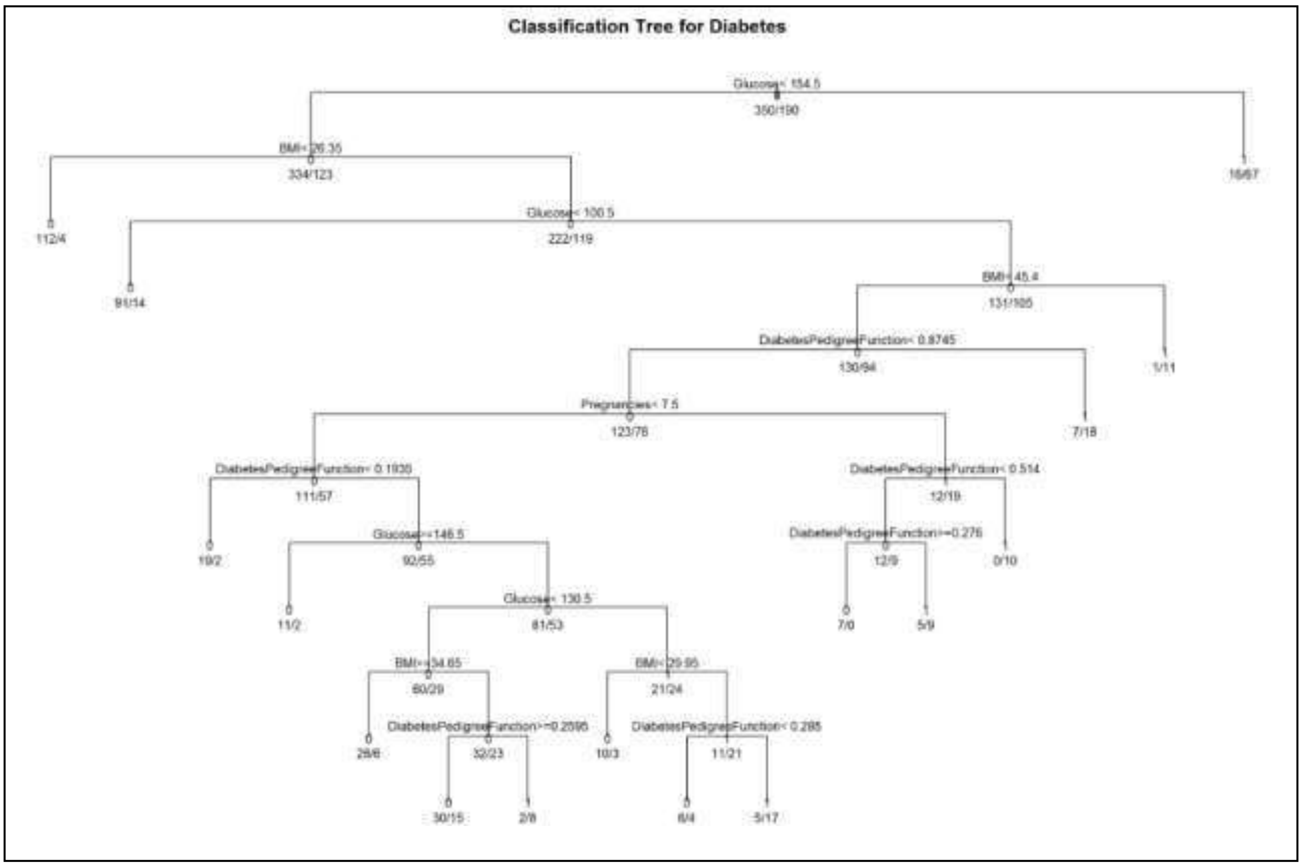
```
##
## fitted.results   0   1
##              0 136  34
##              1  14  44
```

```
misClasificError <- mean(fitted.results != test$Outcome)
print(paste('Accuracy',1-misClasificError))
```

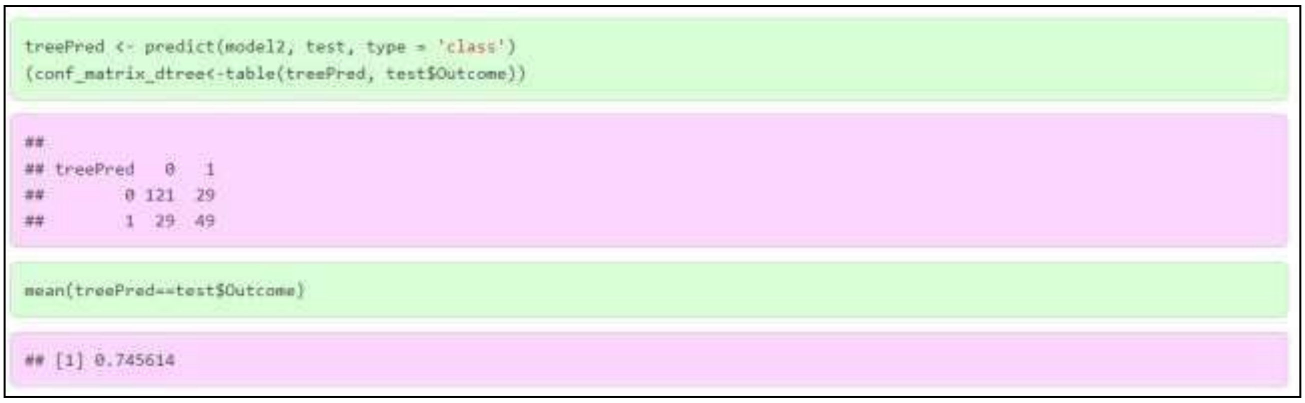
```
## [1] "Accuracy 0.789473684210526"
```

➢ Decision Tree

```
library(rpart)
model2 <- rpart(Outcome ~ Pregnancies + Glucose + BMI + DiabetesPedigreeFunction, data=train,method="class")
plot(model2, uniform=TRUE,
    main="Classification Tree for Diabetes")
text(model2, use.n=TRUE, all=TRUE, cex=.8)
box(which = "outer", lty = "solid")
```



➢ Confusion Table and Accuracy



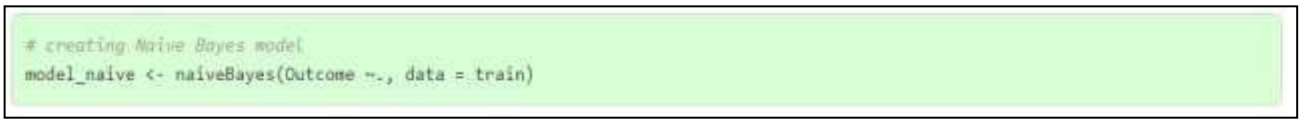
```
treePred <- predict(model2, test, type = 'class')
(conf_matrix_dtree<-table(treePred, test$Outcome))
```

```
##
## treePred   0   1
##        0 121  29
##        1  29  49
```

```
mean(treePred==test$Outcome)
```

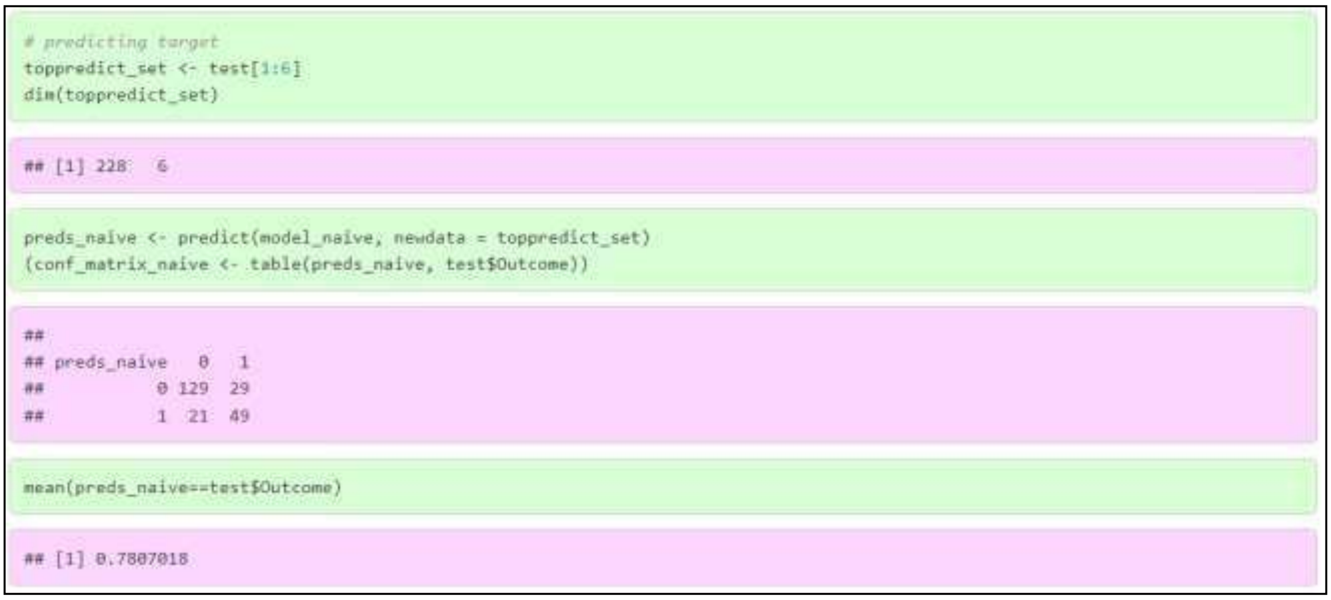
```
## [1] 0.745614
```

➢ Naive Bayes



```
# creating Naive Bayes model
model_naive <- naiveBayes(Outcome ~., data = train)
```

➢ Confusion Table and Accuracy



```
# predicting target
toppredict_set <- test[1:6]
dim(toppredict_set)
```

```
## [1] 228   6
```

```
preds_naive <- predict(model_naive, newdata = toppredict_set)
(conf_matrix_naive <- table(preds_naive, test$Outcome))
```

```
##
## preds_naive   0   1
##           0 129  29
##           1  21  49
```

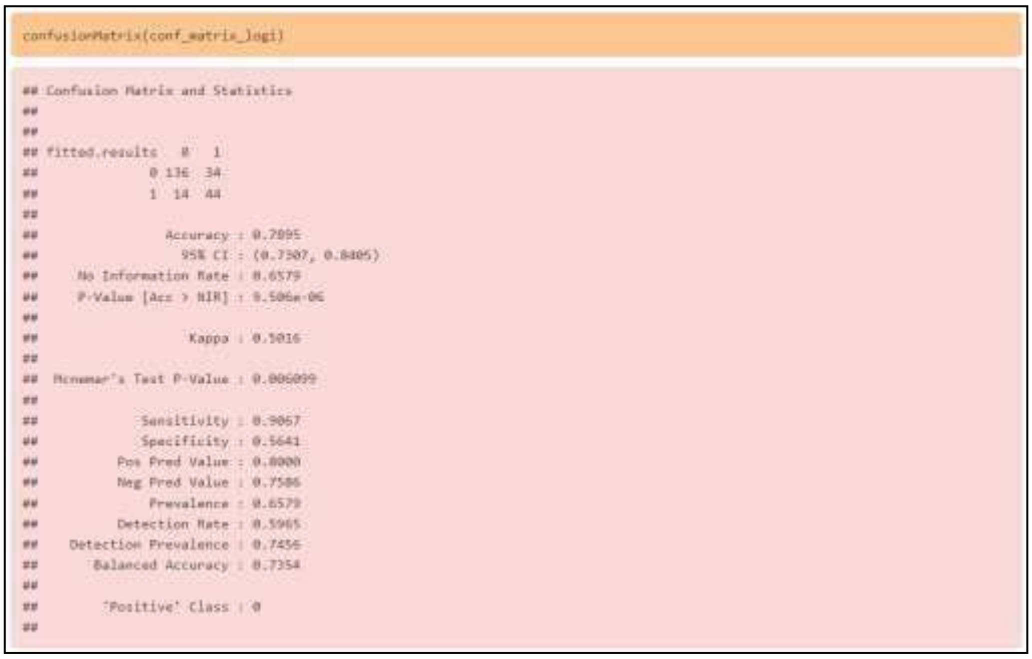
```
mean(preds_naive==test$Outcome)
```

```
## [1] 0.7807018
```

7. Conclusion

If we compare accuracy and sensitivity level of our models to see the highest value, we can summarise as followed :

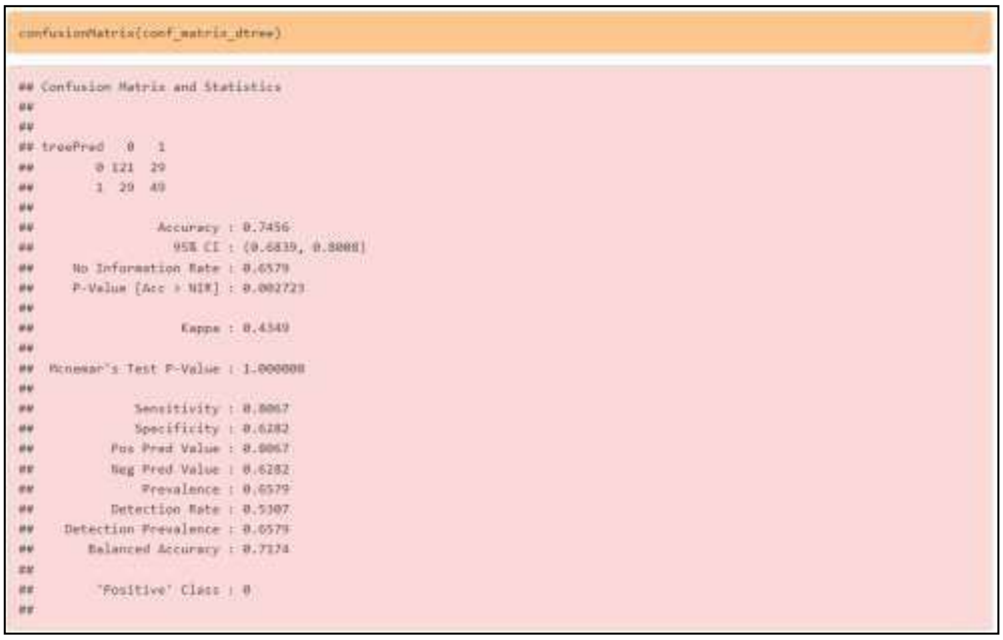
Linear Regression :



```
confusionMatrix(conf_matrix_logi)
```

```
## Confusion Matrix and Statistics
##
##
## fitted.results   0   1
##              0 136  34
##              1  14  44
##
##                Accuracy : 0.7895
##                  95% CI : (0.7307, 0.8405)
##     No Information Rate : 0.6579
##     P-Value [Acc > NIR] : 9.506e-06
##
##                   Kappa : 0.5016
##
##  Mcnemar's Test P-Value : 0.006099
##
##             Sensitivity : 0.9067
##             Specificity : 0.5641
##          Pos Pred Value : 0.8000
##          Neg Pred Value : 0.7586
##              Prevalence : 0.6579
##          Detection Rate : 0.5965
##    Detection Prevalence : 0.7456
##       Balanced Accuracy : 0.7354
##
##        'Positive' Class : 0
##
```

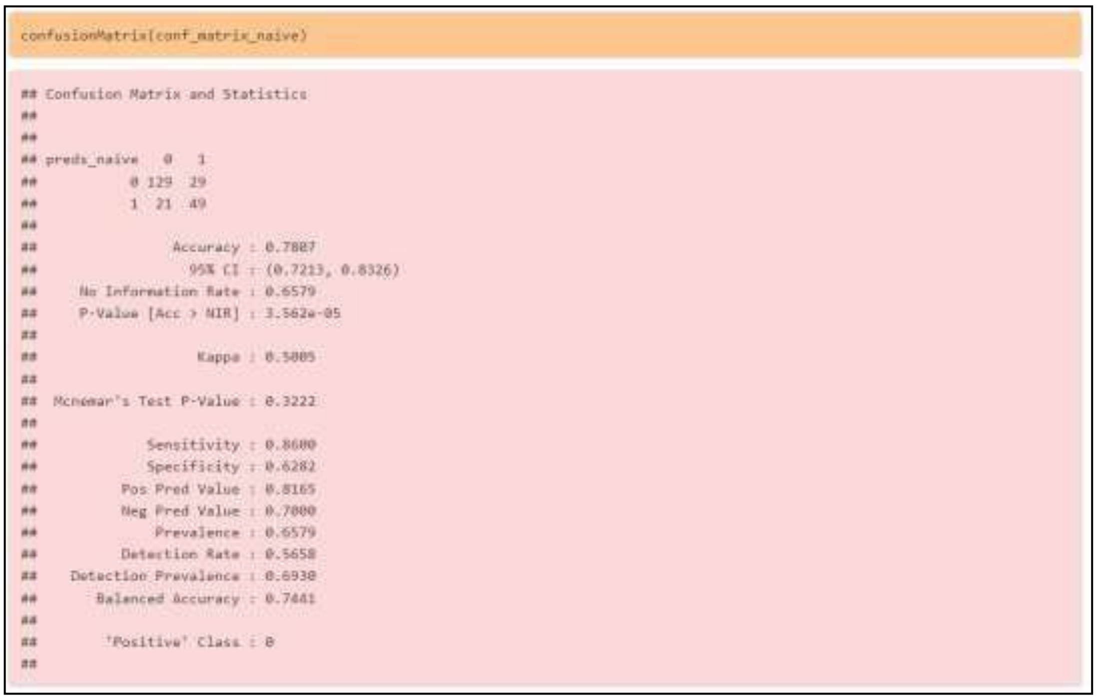
Decision Tree :



```
confusionMatrix(conf_matrix_dtree)
```

```
## Confusion Matrix and Statistics
##
##
## treePred   0   1
##        0 121  29
##        1  29  49
##
##                Accuracy : 0.7456
##                  95% CI : (0.6839, 0.8008)
##     No Information Rate : 0.6579
##     P-Value [Acc > NIR] : 0.002723
##
##                   Kappa : 0.4349
##
##  Mcnemar's Test P-Value : 1.000000
##
##             Sensitivity : 0.8067
##             Specificity : 0.6282
##          Pos Pred Value : 0.8067
##          Neg Pred Value : 0.6282
##              Prevalence : 0.6579
##          Detection Rate : 0.5307
##    Detection Prevalence : 0.6579
##       Balanced Accuracy : 0.7174
##
##        'Positive' Class : 0
##
```

Naive Bayes :



```
confusionMatrix(conf_matrix_naive)
```

```
## Confusion Matrix and Statistics
##
##
## preds_naive   0   1
##           0 129  29
##           1  21  49
##
##                Accuracy : 0.7807
##                  95% CI : (0.7213, 0.8326)
##     No Information Rate : 0.6579
##     P-Value [Acc > NIR] : 3.562e-05
##
##                   Kappa : 0.5005
##
##  Mcnemar's Test P-Value : 0.3222
##
##             Sensitivity : 0.8600
##             Specificity : 0.6282
##          Pos Pred Value : 0.8165
##          Neg Pred Value : 0.7000
##              Prevalence : 0.6579
##          Detection Rate : 0.5658
##    Detection Prevalence : 0.6930
##       Balanced Accuracy : 0.7441
##
##        'Positive' Class : 0
##
```

In this project, we compared the performance of Linear Regression, Decision Tree and Naive Bayes algorithms and found that Linear Regression performed better on this standard, unaltered dataset. After, Linear Regression there comes Naive Bayes algorithm with more accuracy than the Decision Tree. Accuracy given by Linear Regression was 79%, Decision Tree was 74% and Naive Bayes was 78%.

8. References

[1] https://susanli2016.github.io/Predict-Diabetes/

[2] https://rpubs.com/meinari/diabetes-prediction

GITHUB LINK :

https://github.com/Shraddha2004/Diabetes-Prediction-using-R

WEBSITE LINK :

https://rpubs.com/Shraddha20/Diabetes_Prediction_using_R