Healthcare Diabetes Project

Swathi Hebbar

The datasets consists of several medical predictor variables and one target variable, Outcome. Predictor variables includes the number of pregnancies the patient has had, their BMI, insulin level, age, and so on. Pregnancies: Number of times pregnant

(1)Glucose: Plasma glucose concentration a 2 hours in an oral glucose tolerance test (2)BloodPressure: Diastolic blood pressure (mm Hg) (3)SkinThickness: Triceps skin fold thickness (mm) (4)Insulin: 2-Hour serum insulin (mu U/ml) (5)BMI: Body mass index (weight in kg/(height in m)^2) (6)DiabetesPedigreeFunction: Diabetes pedigree function (7)Age: Age (years) (8)Outcome: Class variable (0 or 1) 268 of 768 are 1, the others are 0

Build a model to accurately predict whether the patients in the dataset have diabetes or not?

1.Perform Descriptive analysis

```
#Loading the data
library("readxl")
healthcare_data <- read.csv("C:\\Users\\admin\\Desktop\\Healthcare - Diabetes\\health care diabetes.csv
View(healthcare_data)
str(healthcare_data)
```

```
## '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 ...
```

First we load the data and then will have a look at the datatype of all variables.We check which variables can be made as a facor variable.

```
#Checking the variables which can be done as factor
table(healthcare_data$Pregnancies)
```

```
##
##   0   1   2   3   4   5   6   7   8   9  10  11  12  13  14  15  17
## 111 135 103  75  68  57  50  45  38  28  24  11   9  10   2   1   1
```

```
table(healthcare_data$Outcome)
```

```
##
##   0   1
## 500 268
```

```
healthcare_data$Pregnancies[healthcare_data$Pregnancies>10] <- 10

#Converting continous variable into categorical variable
```

```
healthcare_data$Outcome <- as.factor(healthcare_data$Outcome)
healthcare_data$Pregnancies <- as.factor(healthcare_data$Pregnancies)
str(healthcare_data)
```

```
## 'data.frame':    768 obs. of  9 variables:
##  $ Pregnancies             : Factor w/ 11 levels "0","1","2","3",..: 7 2 9 2 1 6 4 11 3 9 ...
##  $ 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                 : Factor w/ 2 levels "0","1": 2 1 2 1 2 1 2 1 2 2 ...
```

After converting several variables as factor, we check with null values. We count the number of null values in a data frame.

```
#Checking for any NA
sum(is.na(healthcare_data))
```

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

```
df<- healthcare_data
```

We observe, in the beginning the dataframe did not had any null values. But if we look into the data properly, several variables are assigned 0. And 0 does not give any meaning to some variables like Insulin,Glucose,Skin thickness,etc. So we count the the number of zeroes in the dataframe.

```
#Counting the number of zeroes in dataframe
library(plyr)
nonzero <- function(x) sum(x == 0)
numcolwise(nonzero)(df)
```

```
##   Glucose BloodPressure SkinThickness Insulin BMI DiabetesPedigreeFunction Age
## 1       5            35           227     374  11                        0   0
```

Since 0 in Glucose,BP and BMI gives no meaning, and the count is very small, we omit the data which has 0 in it.

```
# deleting the rows whcih contains 0's in Glucose,BP and BMI as the count is very small
library(dplyr)
```

```
##
## Attaching package: 'dplyr'
```

```
## The following objects are masked from 'package:plyr':
##
##     arrange, count, desc, failwith, id, mutate, rename, summarise,
##     summarize
```

```
## The following objects are masked from 'package:stats':
##
##     filter, lag
```

```
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
df <- filter(df,df$Glucose > 0)
df <- filter(df,df$BloodPressure>0)
df <- filter(df,df$BMI>0)
```

We count the number of zeroes once again.Now we can't delete the data which contains 0 in it as the count is more. So we convert our 0's into NA. Because we have several ways to deal with the NA's.

```
#counting the number of zeroes again
numcolwise(nonzero)(df)
```

```
##   Glucose BloodPressure SkinThickness Insulin BMI DiabetesPedigreeFunction Age
## 1       0             0           192     332   0                        0   0
```

```
#converting all zeroes to NA
df1 <- df[, 4:5][df[, 4:5] == 0] <- NA
```

To deal with the NA's in dataframe, we need to know what % of data is null.

```
#Now counting the number of NA's
sum(is.na(df))
```

```
## [1] 524
```

```
sum(is.na(df$SkinThickness))
```

```
## [1] 192
```

```
sum(is.na(df$Insulin))
```

```
## [1] 332
```

```
#Checking the %of data whcih is missing
sum(is.na(df$SkinThickness))/nrow(df)*100
```

```
## [1] 26.51934
```

```
sum(is.na(df$Insulin))/nrow(df)*100
```

```
## [1] 45.85635
```

Now as we can see, the amount of data which is null is quite large to omit. And if we treat NA with mean/median then the complete meaning of data might change. We can't apply mode method to treat NA as all the variables are continuous. So we use the method of prediction to find the NA's in a dataframe. We can predict the variables which ever has NA but we should have knowledge about the other variables whcih are correlated to it.So first we find the variables which has influence over these variables Glucose and Insulin.

```
#Replacing those NA's by the method of prediction
#Checking which variable is correlated to Insulin and SkinThickness
summary(lm(Insulin~.,data=df))
```

```
##
## Call:
## lm(formula = Insulin ~ ., data = df)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -329.84  -49.92  -12.97   31.38  521.09
##
## Coefficients:
##                           Estimate Std. Error t value Pr(>|t|)
## (Intercept)             -184.4651    43.7000  -4.221 3.05e-05 ***
```

```
## Pregnancies1               12.4567    17.1188   0.728   0.4673
## Pregnancies2               17.5637    18.3439   0.957   0.3389
## Pregnancies3               -9.6489    20.0266  -0.482   0.6302
## Pregnancies4               18.9208    23.4759   0.806   0.4208
## Pregnancies5               -5.7563    26.0097  -0.221   0.8250
## Pregnancies6                8.0872    27.9489   0.289   0.7725
## Pregnancies7               -1.7209    28.4124  -0.061   0.9517
## Pregnancies8               42.6946    32.4561   1.315   0.1892
## Pregnancies9              -35.7118    35.3968  -1.009   0.3137
## Pregnancies10             -27.8539    29.2562  -0.952   0.3417
## Glucose                     2.1646     0.1928  11.228  < 2e-16 ***
## BloodPressure              -0.4960     0.4400  -1.127   0.2604
## SkinThickness              -0.1701     0.6405  -0.266   0.7907
## BMI                         2.3724     1.0043   2.362   0.0187 *
## DiabetesPedigreeFunction   13.2132    14.7059   0.898   0.3695
## Age                         0.8715     0.7320   1.191   0.2346
## Outcome1                   -6.6025    13.0170  -0.507   0.6123
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 96.33 on 374 degrees of freedom
##   (332 observations deleted due to missingness)
## Multiple R-squared:  0.3715, Adjusted R-squared:  0.3429
## F-statistic:    13 on 17 and 374 DF,  p-value: < 2.2e-16
```

```
summary(lm(SkinThickness~.,data=df))
```

```
##
## Call:
## lm(formula = SkinThickness ~ ., data = df)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -21.6642  -4.9047  -0.6693   5.1524  21.0493
##
## Coefficients:
##                            Estimate Std. Error t value Pr(>|t|)
## (Intercept)              -7.206001   3.591562  -2.006   0.0455 *
## Pregnancies1              2.207776   1.378213   1.602   0.1100
## Pregnancies2              0.887605   1.481954   0.599   0.5496
## Pregnancies3             -0.224620   1.617148  -0.139   0.8896
## Pregnancies4              0.028888   1.896792   0.015   0.9879
## Pregnancies5              3.938773   2.089924   1.885   0.0603 .
## Pregnancies6              1.193794   2.255642   0.529   0.5969
## Pregnancies7              3.281230   2.287382   1.434   0.1523
## Pregnancies8              6.037525   2.607527   2.315   0.0211 *
## Pregnancies9              4.421201   2.852228   1.550   0.1220
## Pregnancies10             2.467142   2.361188   1.045   0.2968
## Glucose                   0.006629   0.017992   0.368   0.7128
## BloodPressure            -0.003032   0.035582  -0.085   0.9321
## Insulin                  -0.001108   0.004174  -0.266   0.7907
## BMI                       0.974804   0.064265  15.169   <2e-16 ***
## DiabetesPedigreeFunction  1.524152   1.185827   1.285   0.1995
## Age                       0.036096   0.059172   0.610   0.5422
## Outcome1                  0.610819   1.050711   0.581   0.5614
```

```
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 7.777 on 374 degrees of freedom
##   (332 observations deleted due to missingness)
## Multiple R-squared:  0.477,  Adjusted R-squared:  0.4532
## F-statistic: 20.06 on 17 and 374 DF,  p-value: < 2.2e-16
```

Now after we get to know about the correlated variable, we try predicting the Glucose and SkinThickness. Then we assign the predicted set to a dataframe.

```
#Predicting the Insulin Values with the help of correlated variable
library(rpart)
anova_mod <- rpart(Insulin ~ . - Glucose,
                   data=df[!is.na(df$Insulin), ],
                   method="anova", na.action=na.omit)
Insulin_pred <- predict(anova_mod, df[is.na(df$Insulin), ])
Insulin_pred <- data.frame(Insulin_pred)
```

Now we have a predicted set. Now we replace the NA's in dataframe with the predicted dataset what we obtained.

```
#Replacing those predicted insulin values with NA's
s <- 1
for (val in 1:nrow(df)){
  if (is.na(df[val,5])) {
    df[val,5] <- Insulin_pred[s,1]
    s <- s+1
  }
}
```

Now we can count the NA's in the df$Insulin, we can observe that NA's are replaced with the predicted value. NA's will be zero.

```
#Now we can check that NA count is 0
sum(is.na(df$Insulin))
```

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

Now we repeat the same with another variable Skin Thickness.

```
#Predicting SkinThcikness with the help of correlated variable
anova_mod2 <- rpart(SkinThickness ~ . - BMI,
                    data=df[!is.na(df$SkinThickness), ],
                    method="anova", na.action=na.omit)
Skin_pred <- predict(anova_mod2, df[is.na(df$SkinThickness), ])
Skin_pred <- data.frame(Skin_pred)


#Replacing NA's with predicted values
s <- 1
for (val in 1:nrow(df)){
  if (is.na(df[val,4])) {
    df[val,4] <- Skin_pred[s,1]
    s <- s+1
  }
}
```

```
#Now the count of NA is 0
sum(is.na(df$SkinThickness))
```

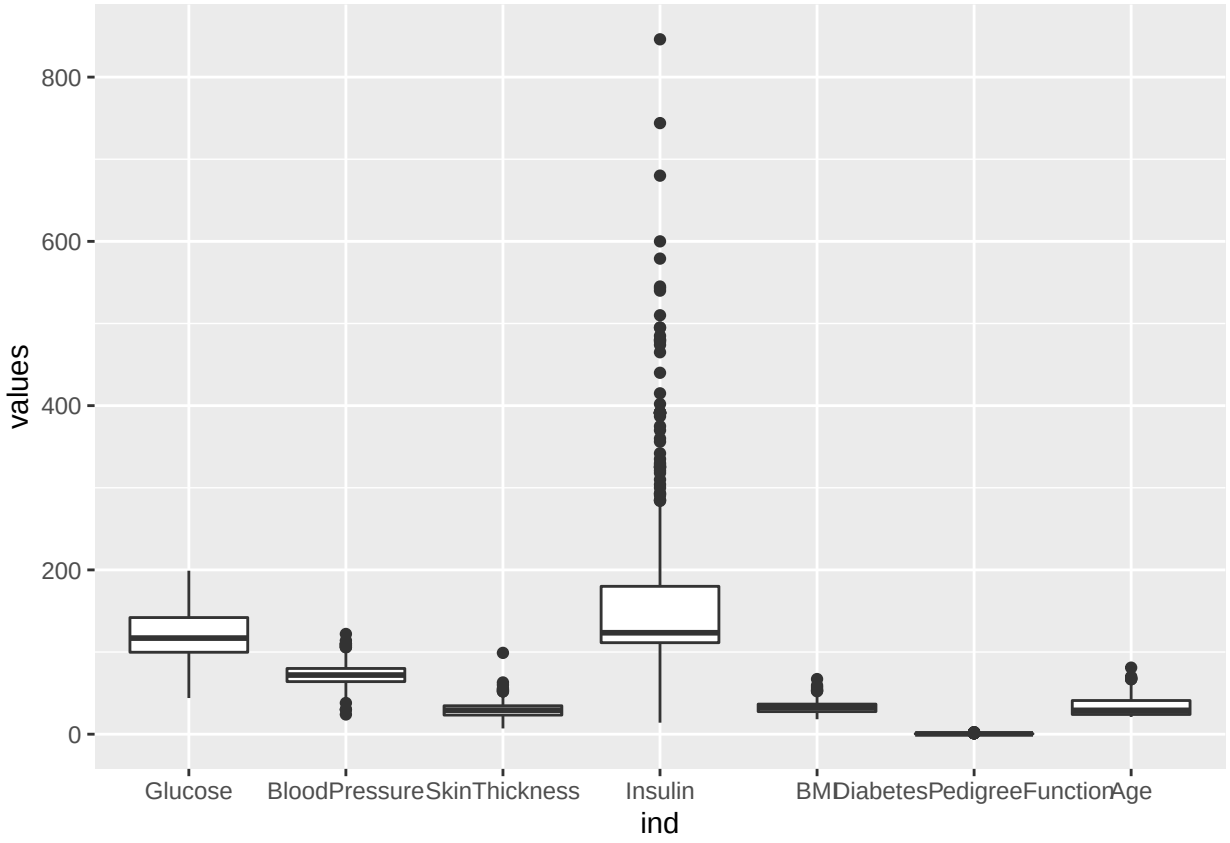
```
## [1] 0
```

```
df_new <- df
```

Now the NA's are treated, we deal with the outliers.

```
#Boxplot to detect the outliers
library(ggplot2)
ggplot(stack(df), aes(x = ind, y = values)) +
  geom_boxplot()
```

```
## Warning in stack.data.frame(df): non-vector columns will be ignored
```




```
#To find the outliers
#Defining the quantiles
summary(df$BloodPressure)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    24.0    64.0    72.0    72.4    80.0   122.0
```

```
quantile(df$BloodPressure,probs = c(0,.25,.50,.75,.90,1))
```

```
##   0%  25%  50%  75%  90% 100%
##   24   64   72   80   88  122
```

```
summary(df$BMI)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   18.20   27.50   32.40   32.47   36.60   67.10
```

```
quantile(df$BMI,probs = c(0,.25,.50,.5,.75,.90,1))
```

```
##    0%   25%   50%   50%   75%   90%  100%
## 18.20 27.50 32.40 32.40 36.60 41.44 67.10
```

```
summary(df$DiabetesPedigreeFunction)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  0.0780  0.2450  0.3790  0.4748  0.6275  2.4200
```

```
quantile(df$DiabetesPedigreeFunction,probs = c(0,.25,.50,.75,.85,.90,.95,1))
```

```
##      0%     25%     50%     75%     85%     90%     95%    100%
## 0.07800 0.24500 0.37900 0.62750 0.75295 0.88070 1.13770 2.42000
```

```
summary(df$Age)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   21.00   24.00   29.00   33.35   41.00   81.00
```

```
quantile(df$Age,probs = c(0,.25,.50,.75,.90,1))
```

```
##   0%  25%  50%  75%  90% 100%
##   21   24   29   41   51   81
```

```
summary(df$SkinThickness)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    7.00   23.26   29.25   29.39   34.58   99.00
```

```
quantile(df$SkinThickness,probs = c(0,.25,.50,.60,.75,.90,1))
```

```
##       0%      25%      50%      60%      75%      90%     100%
##  7.00000 23.25735 29.25000 30.00000 34.58333 41.00000 99.00000
```

```
summary(df$Insulin)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    14.0   111.4   123.5   154.7   180.0   846.0
```

```
quantile(df$Insulin,probs = c(0,.25,.50,.60,.75,.90,1))
```

```
##       0%      25%      50%      60%      75%      90%     100%
##  14.0000 111.4186 123.5000 151.6000 180.0000 265.0000 846.0000
```

```
#to remove the outliers
outliers <- boxplot(df$BloodPressure,plot=FALSE)$out
df<-df[-which(df$BloodPressure %in% outliers), ]
#boxplot(df$BloodPressure)

outliers <- boxplot(df$SkinThickness,plot=FALSE)$out
df<-df[-which(df$SkinThickness %in% outliers), ]
#boxplot(df$SkinThickness)

outliers <- boxplot(df$Insulin,plot=FALSE)$out
df<-df[-which(df$Insulin %in% outliers), ]
#boxplot(df$Insulin)
```

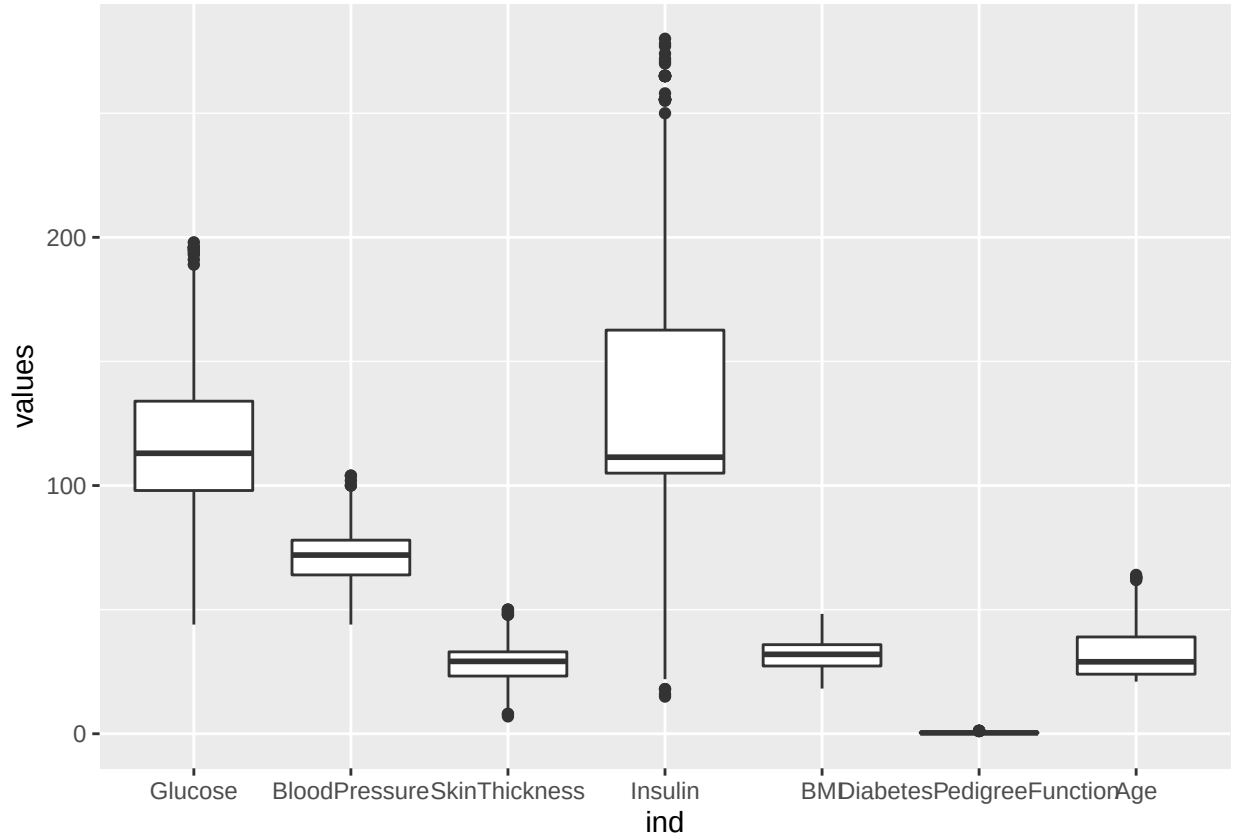
```
outliers <- boxplot(df$BMI,plot=FALSE)$out
df<-df[-which(df$BMI %in% outliers), ]
#boxplot(df$BMI)

outliers <- boxplot(df$DiabetesPedigreeFunction,plot=FALSE)$out
df<-df[-which(df$DiabetesPedigreeFunction %in% outliers), ]
#boxplot(df$DiabetesPedigreeFunction)

outliers <- boxplot(df$Age,plot=FALSE)$out
df<-df[-which(df$Age %in% outliers), ]
#boxplot(df$Age)

#Plotting the boxplot again
ggplot(stack(df), aes(x = ind, y = values)) +
  geom_boxplot()
```

```
## Warning in stack.data.frame(df): non-vector columns will be ignored
```



Now the data is treated with NA's, outliers. And several variables are categorised to factor variables. The data processisng is done for now.So we write the dataframe to local disc

```
#Writing the final data to local disc after data processing
df_new <- df
write.csv(df_new,"C:\\Users\\admin\\Desktop\\Healthcare - Diabetes\\data.csv", row.names = FALSE)
```

2.Visualization of data distribution using Histogram

```
#Histogram to check the data distribution
#install.packages("Hmisc")
library(Hmisc)
```

```
## Warning: package 'Hmisc' was built under R version 4.0.5
```

```
## Loading required package: lattice
```

```
## Loading required package: survival
```

```
## Loading required package: Formula
```

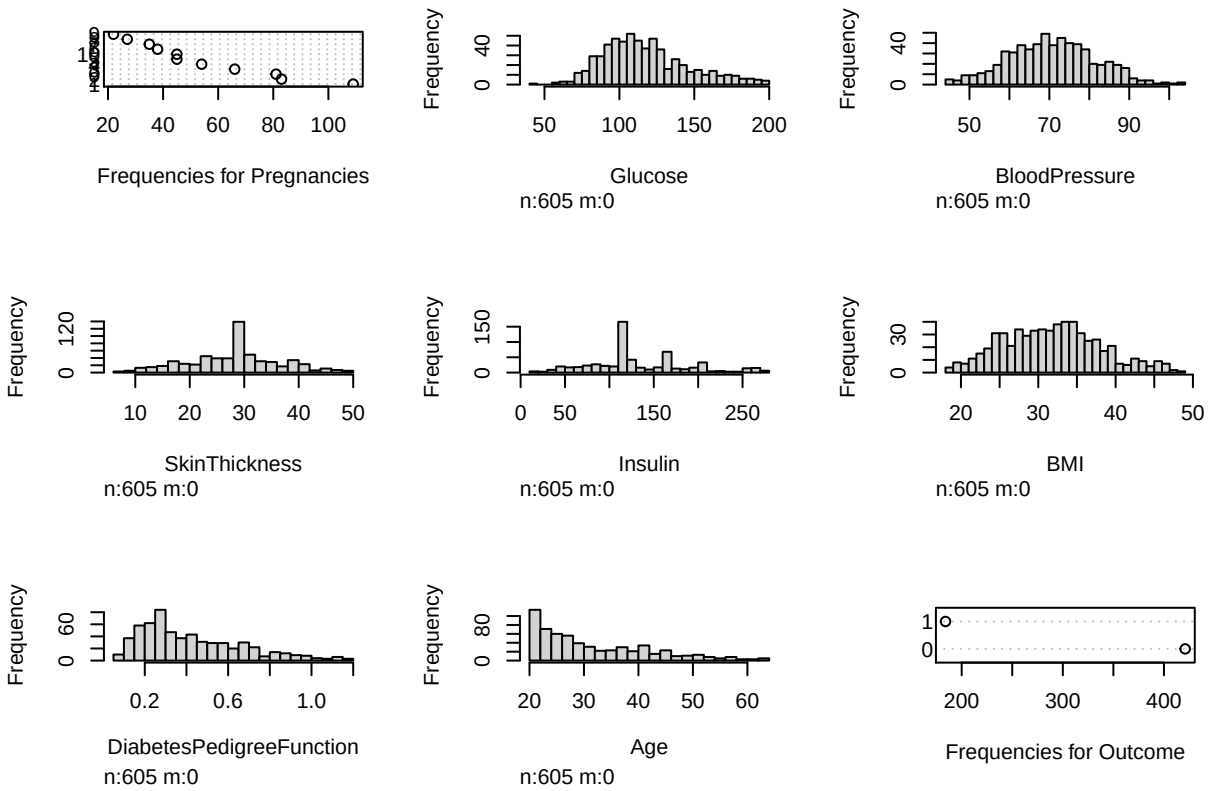
```
##
## Attaching package: 'Hmisc'
```

```
## The following objects are masked from 'package:dplyr':
##
##     src, summarize
```

```
## The following objects are masked from 'package:plyr':
##
##     is.discrete, summarize
```

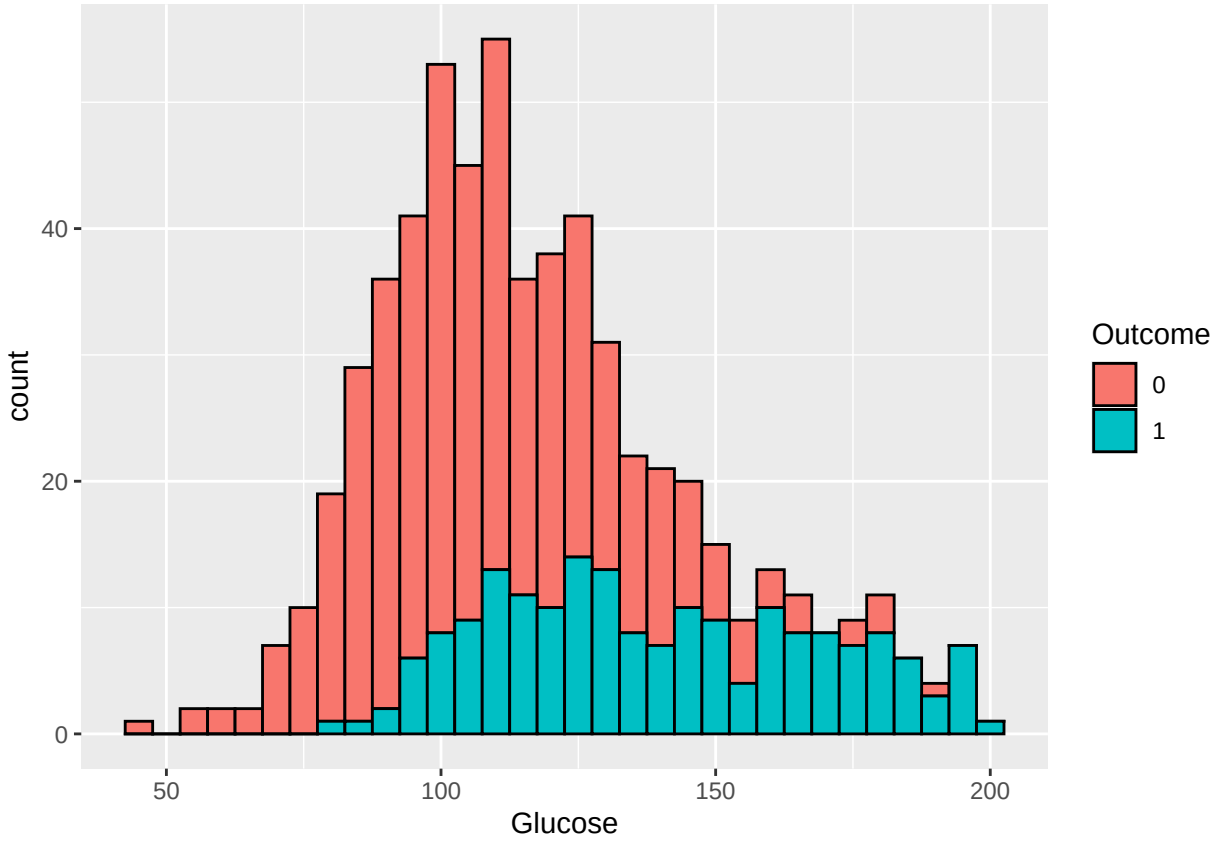
```
## The following objects are masked from 'package:base':
##
##     format.pval, units
```

```
hist.data.frame(df)
```



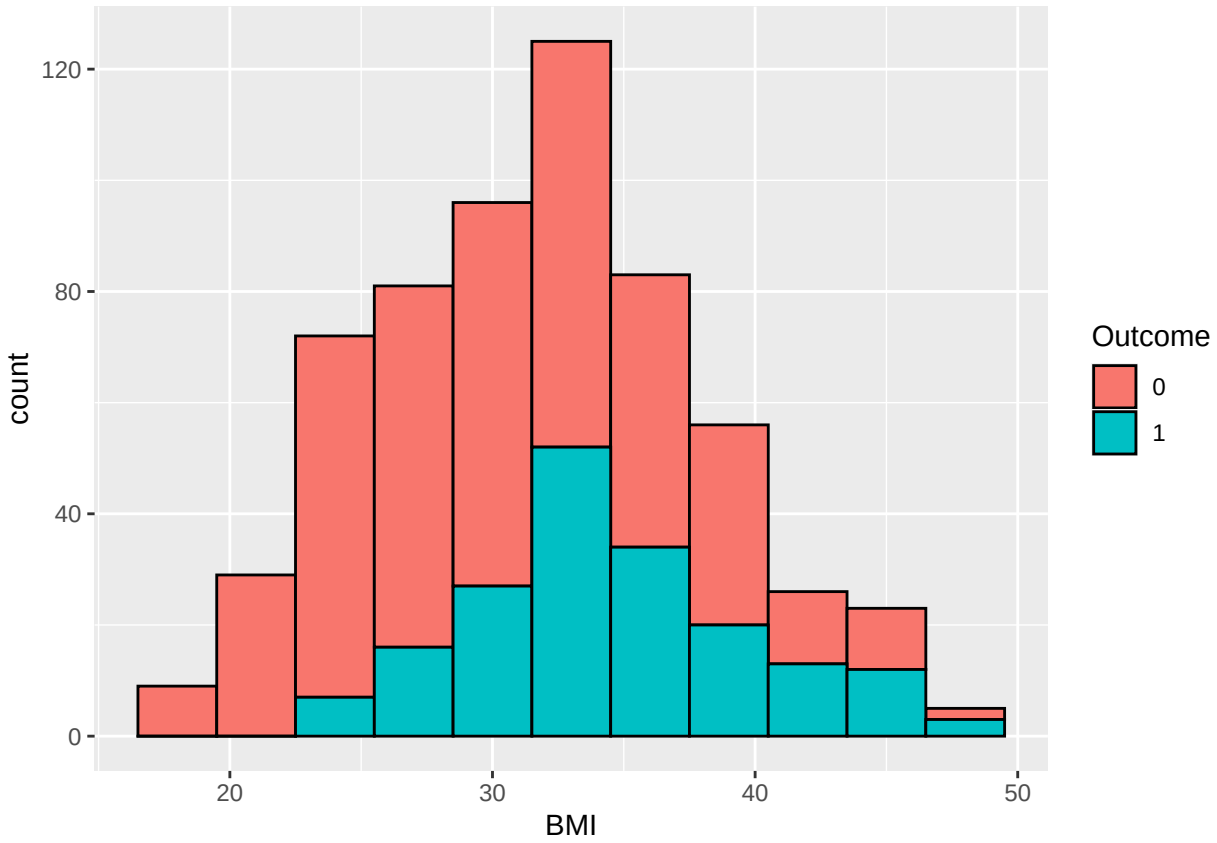
3. and 4. Creating a plot to describe the frequency and by their outcome and to know the datatypes.

```
#Plot with the count of outcomes on all variables
ggplot(data=df) +
  geom_histogram(mapping=aes(x=Glucose,fill=Outcome),color="black",binwidth = 5)
```



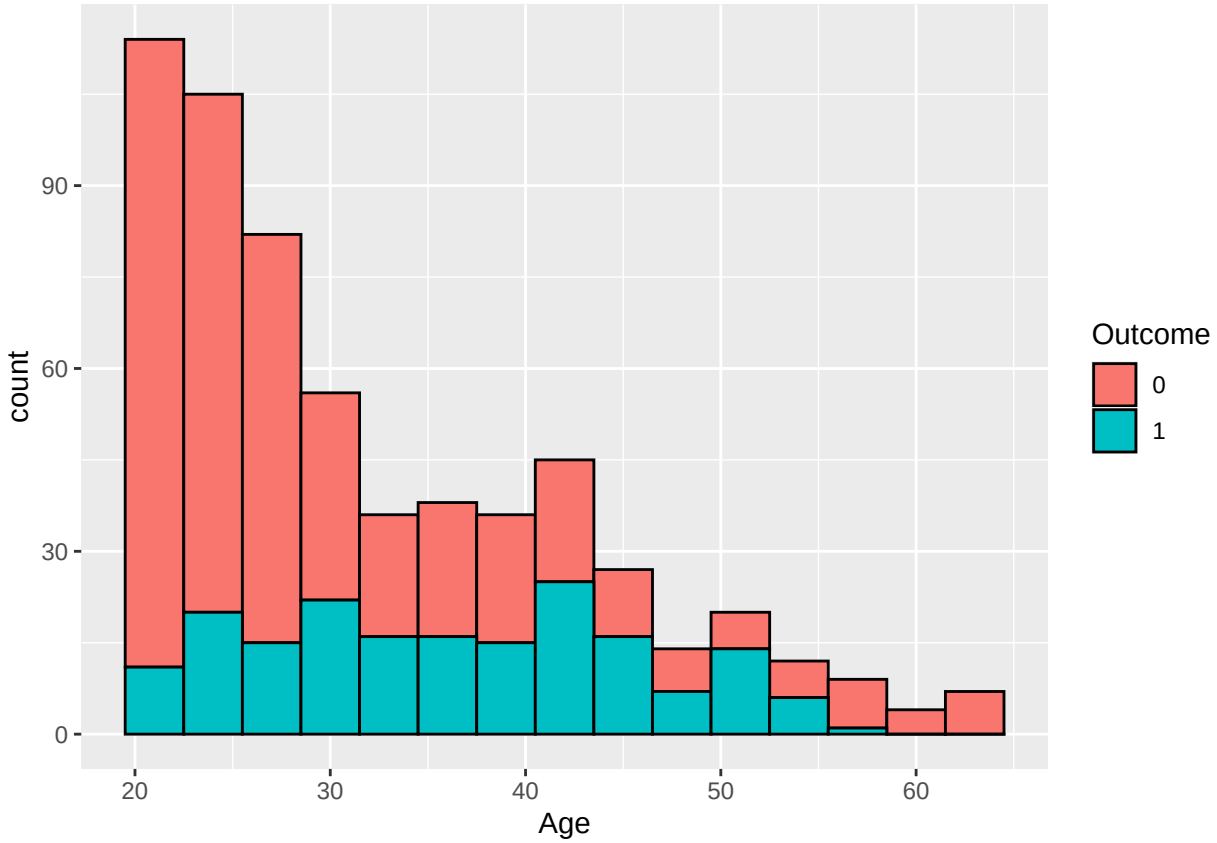
Here 0 means non diabetic and 1 means diabetic. And we observe that as the glucose level starts increasing from 100, the number of diabetic patients kept increasing.

```
ggplot(data=df) +
  geom_histogram(mapping=aes(x=BMI,fill=Outcome), color = "black",binwidth =3)
```



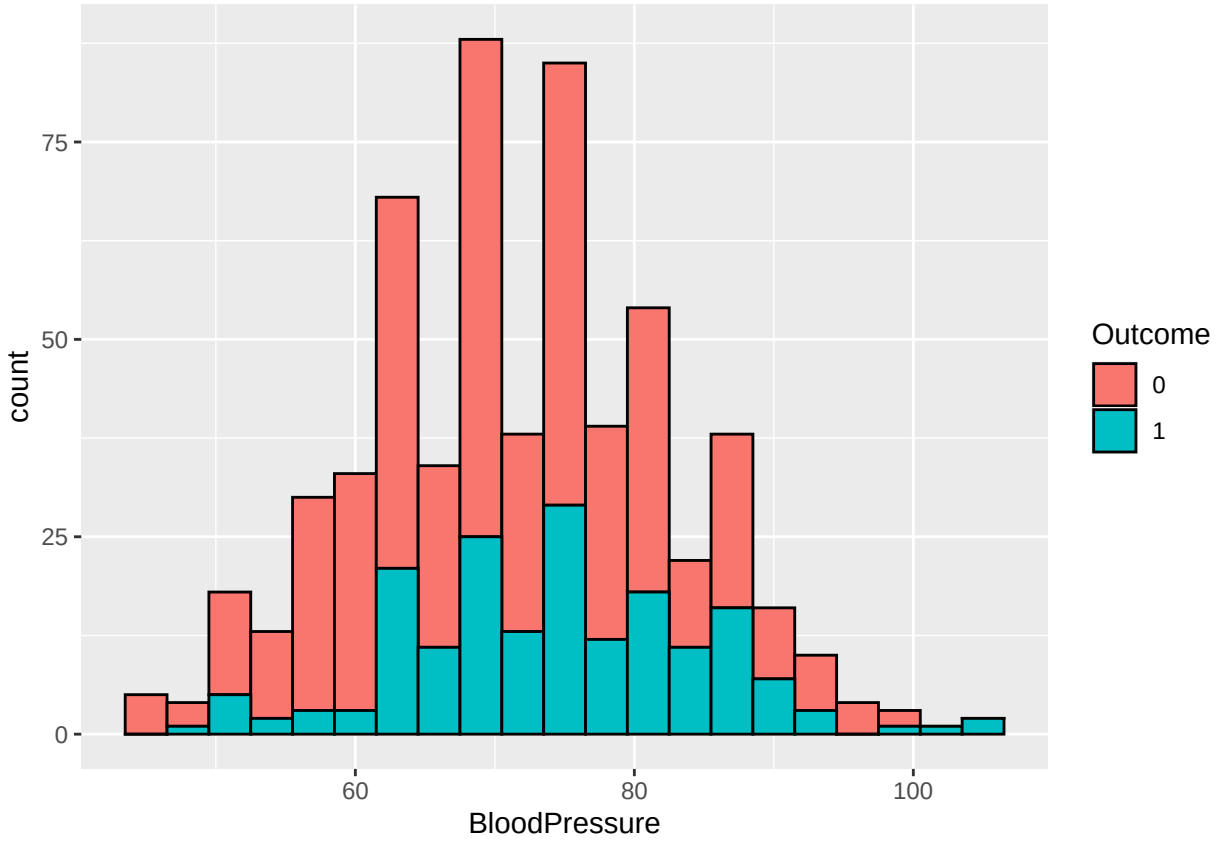
Here we can observe that Obese class i.e., people who have BMI more than 25 are affected by diabetes. No one who had low BMI were affected by diabetes. So we can say that even BMI plays an important role in Diabetes.

```
ggplot(data=df) +
  geom_histogram(mapping=aes(x=Age,fill=Outcome),color = "black",binwidth =3)
```



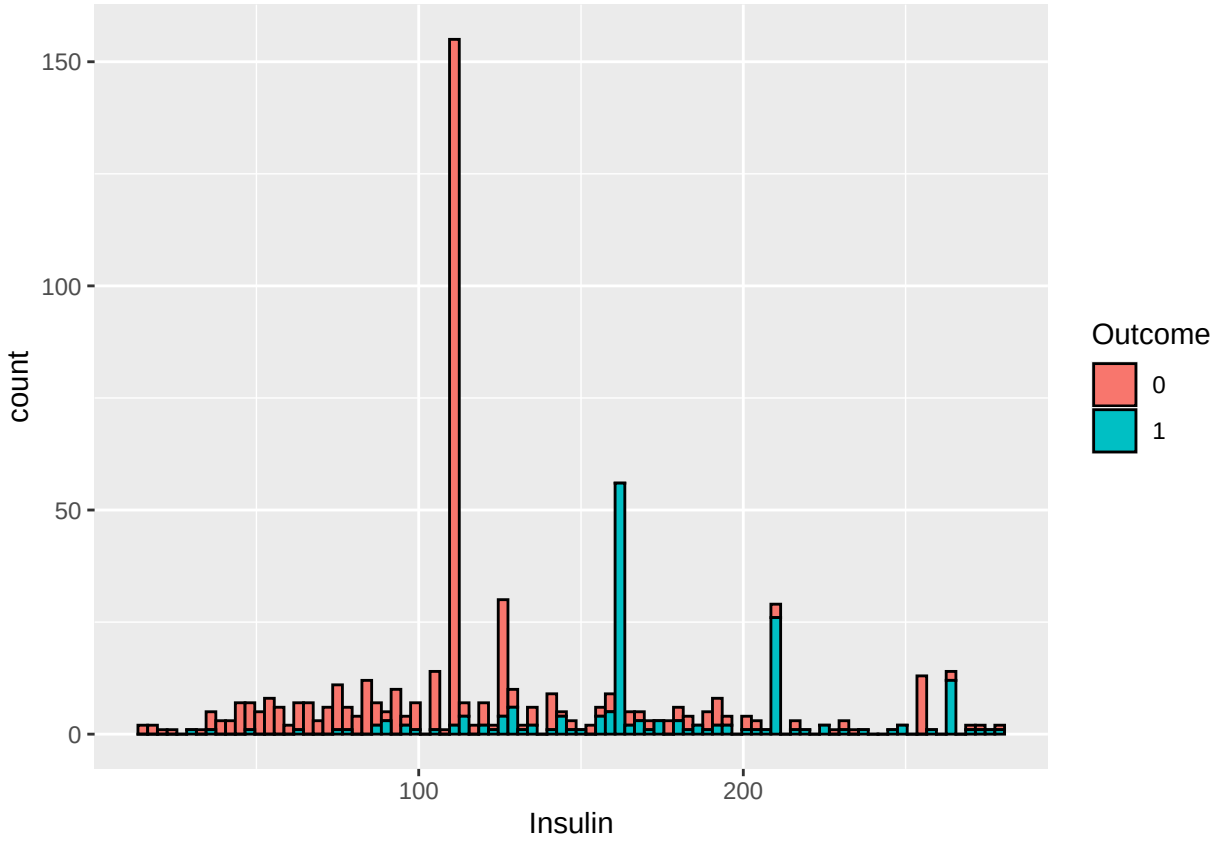
Clearly Age did not had much impact on the Diabetes. We can observe the number of diabetic patients remains almost same in all age category.

```
ggplot(data=df) +
  geom_histogram(mapping=aes(x=BloodPressure,fill=Outcome), color = "black",binwidth =3)
```



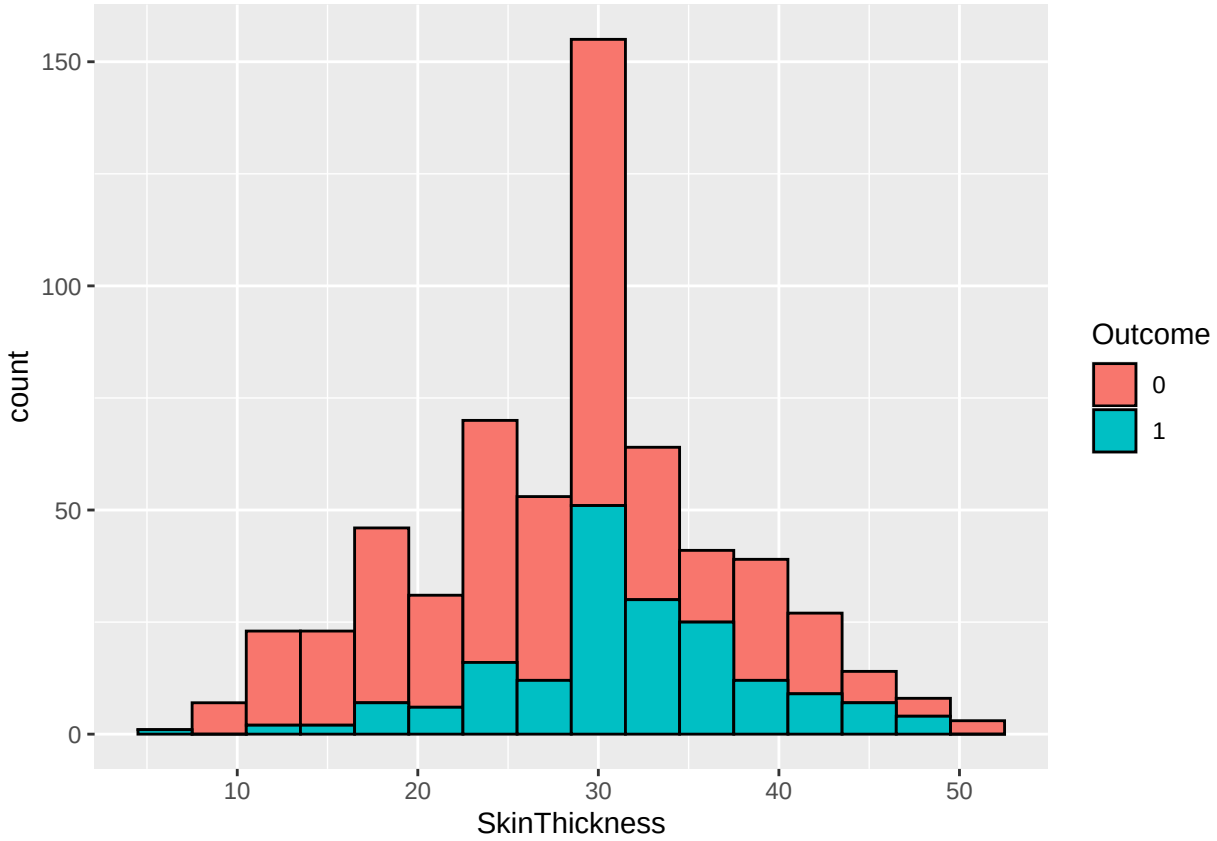
Here we know that the normal Blood pressure lies between 80-100. The people who are diabetic are having low Blood pressure as well, high Bloodpressure as well and normal Blood pressure as well. So we can conclude that it does not impact much to Diabetes. Irrespective of Age and Blood pressure people are getting affected by Diabetes.

```
ggplot(data=df) +
  geom_histogram(mapping=aes(x=Insulin,fill=Outcome), color = "black",binwidth =3)
```



Here, we can see a spike in Insulin near 100-200. Its because, the normal Insulin level is within 110. And most of the people are falling to that category, so there is a spike in between 100-200. And clearly more the Insulin, the patients are at more risk of getting Diabetes.

```
ggplot(data=df) +
  geom_histogram(mapping=aes(x=SkinThickness,fill=Outcome), color = "black",binwidth =3)
```



Skin Thickness is directly related to the BMI. So even here more the Skin Thickness, more the patients are at risk of diabetes.

```
#To know the data type of all the varibales
str(df)
```

```
## 'data.frame':    605 obs. of  9 variables:
##  $ Pregnancies             : Factor w/ 11 levels "0","1","2","3",..: 7 2 9 2 6 4 5 11 6 1 ...
##  $ Glucose                 : int  148 85 183 89 116 78 110 168 166 118 ...
##  $ BloodPressure           : int  72 66 64 66 74 50 92 74 72 84 ...
##  $ SkinThickness           : num  35 29 29.5 23 29.2 ...
##  $ Insulin                 : num  163 111 210 94 111 ...
##  $ BMI                     : num  33.6 26.6 23.3 28.1 25.6 31 37.6 38 25.8 45.8 ...
##  $ DiabetesPedigreeFunction: num  0.627 0.351 0.672 0.167 0.201 0.248 0.191 0.537 0.587 0.551 ...
##  $ Age                     : int  50 31 32 21 30 26 30 34 51 31 ...
##  $ Outcome                 : Factor w/ 2 levels "0","1": 2 1 2 1 1 2 1 2 2 2 ...
```

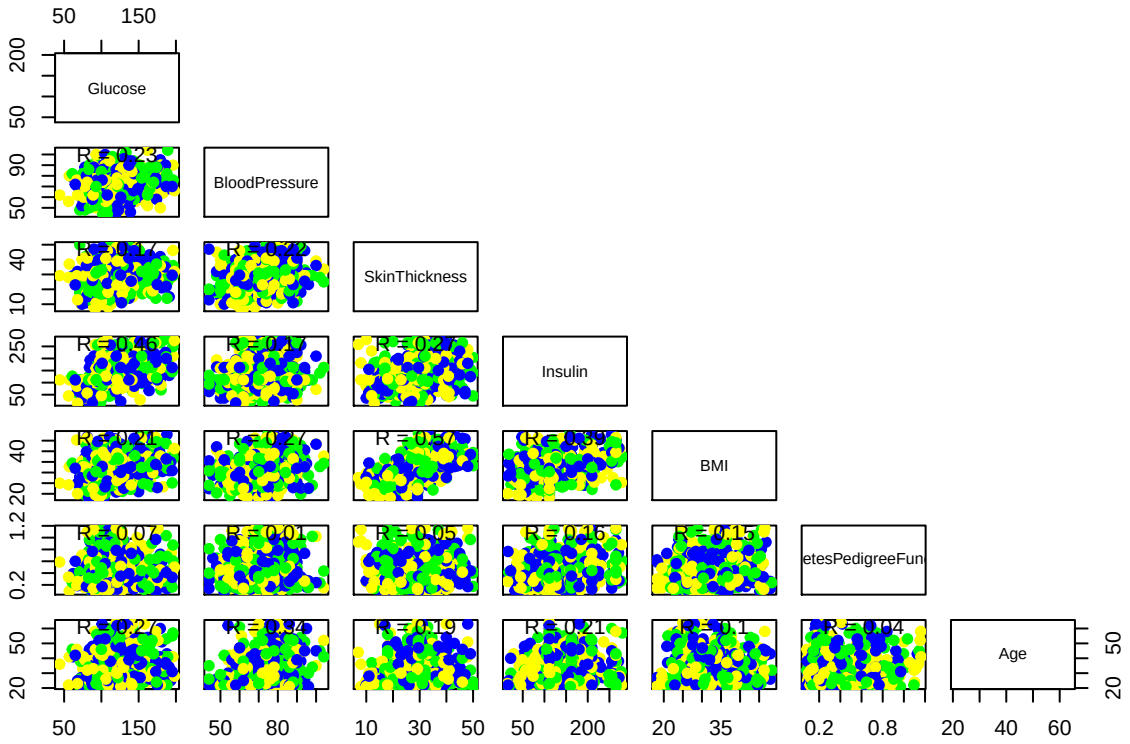
We can have a look at all the data types of variables.

5. Creating scatter plots between all pairs of variables

```
#SCATTER PLOT

lower.panel<-function(x, y){
  points(x,y, pch=19, col=c("blue", "green", "yellow"))
  r <- round(cor(x, y), digits=2)
  txt <- paste0("R = ", r)
  usr <- par("usr"); on.exit(par(usr))
  par(usr = c(0, 1, 0, 1))
```

```
  text(0.5, 0.9, txt)
}
pairs(df[2:8], lower.panel = lower.panel,
      upper.panel = NULL)
```



Even here we can observe the correlation due to R value.If R value is positive and equal to 1 then more is the correlation. If R value is negative and its almost to equal to -1 then its likely inversely related. If R=0 then its not related.

6. Heatmap with the correlation matrix

```
#HEATMAP to analyse the correlation

df_graph <- data.matrix(df)
#install.packages("corrplot")
library(corrplot)
```

```
## corrplot 0.84 loaded
```

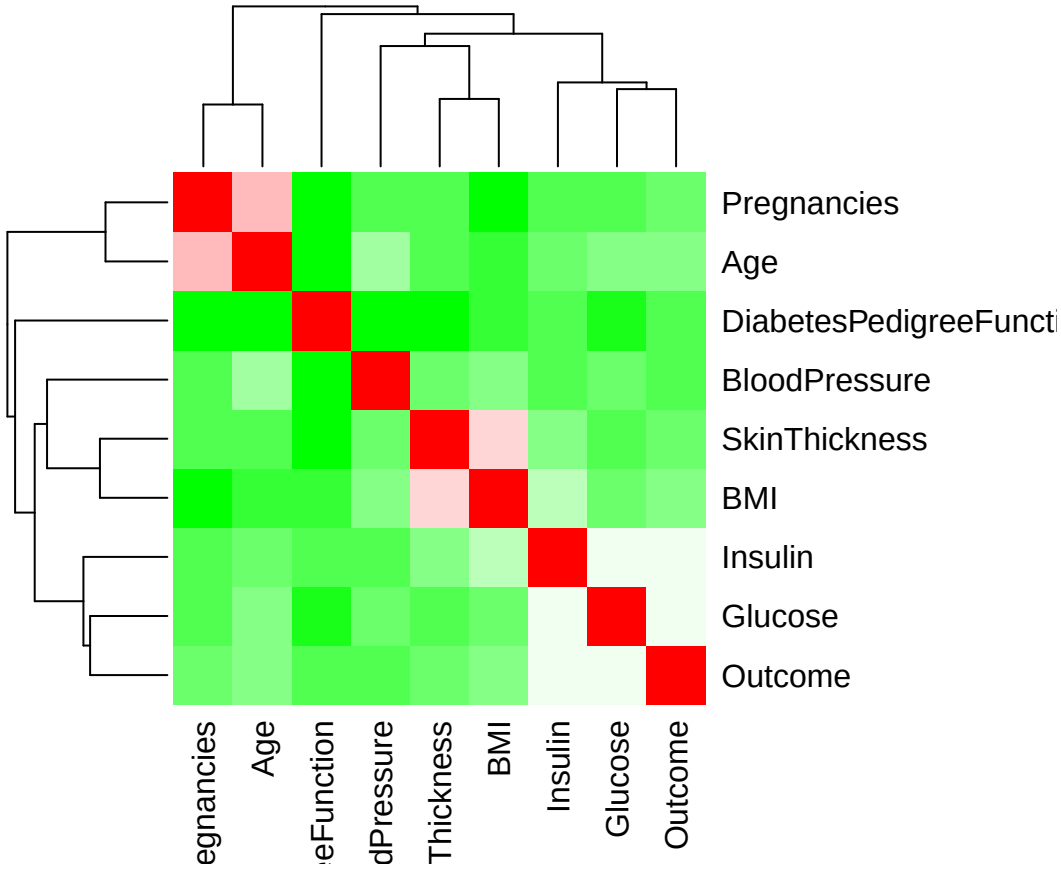
```
df.cor<-cor(df_graph)
df.cor
```

```
##                          Pregnancies    Glucose BloodPressure SkinThickness
## Pregnancies              1.000000000 0.15406033    0.18366801    0.16330039
## Glucose                  0.154060329 1.00000000    0.23144307    0.16825278
## BloodPressure            0.183668013 0.23144307    1.00000000    0.22327022
## SkinThickness            0.163300387 0.16825278    0.22327022    1.00000000
## Insulin                  0.174454284 0.45635911    0.16862086    0.27023271
## BMI                      0.051706497 0.20602834    0.27053596    0.57085609
```

```
## DiabetesPedigreeFunction 0.002385062 0.06795306    0.01105309    0.04822386
## Age                      0.605011943 0.26705122    0.33512448    0.19296685
## Outcome                  0.251215468 0.49202172    0.16350476    0.22256158
##                            Insulin       BMI DiabetesPedigreeFunction
## Pregnancies              0.1744543 0.0517065              0.002385062
## Glucose                  0.4563591 0.2060283              0.067953059
## BloodPressure            0.1686209 0.2705360              0.011053089
## SkinThickness            0.2702327 0.5708561              0.048223857
## Insulin                  1.0000000 0.3926701              0.162177294
## BMI                      0.3926701 1.0000000              0.145296089
## DiabetesPedigreeFunction 0.1621773 0.1452961              1.000000000
## Age                      0.2127109 0.1044152              0.043747084
## Outcome                  0.4714434 0.2901103              0.179900988
##                                 Age   Outcome
## Pregnancies              0.60501194 0.2512155
## Glucose                  0.26705122 0.4920217
## BloodPressure            0.33512448 0.1635048
## SkinThickness            0.19296685 0.2225616
## Insulin                  0.21271088 0.4714434
## BMI                      0.10441518 0.2901103
## DiabetesPedigreeFunction 0.04374708 0.1799010
## Age                      1.00000000 0.2591060
## Outcome                  0.25910599 1.0000000
```

Correlation matrix

```
palette = colorRampPalette(c("green", "white", "red")) (20)
heatmap(x = df.cor, col = palette, symm = TRUE)
```



7., 8. and 9. Building several models to predict the outcome and verifying the classification report, ROC with all models.

```
#Model building

#Defining train and test data
set.seed(124)
train_indices <- sample(1:nrow(df), 0.70 * nrow(df))
train <- df[train_indices, ]
test <- df[-train_indices,]
#View(test)

#Having a look at the data distribution in the dataframe along with train and test data
prop.table(table(df$Outcome))
```

```
##
##         0         1
## 0.6958678 0.3041322
```

```
prop.table(table(train$Outcome))
```

```
##
##         0         1
## 0.6903073 0.3096927
```

```
prop.table(table(test$Outcome))
```

```
##
```

```
##         0         1
## 0.7087912 0.2912088
```

We divide the data frame given into test and train data. We apply the model to train and predict the outcome in the test data. (1) Logistic regression :

```
###Logistic regression

# Fit the model
library(class)
```

```
## Warning: package 'class' was built under R version 4.0.5
```

```
log_reg <- glm(Outcome ~ . , data = train,family = 'binomial')
summary(log_reg)
```

```
##
## Call:
## glm(formula = Outcome ~ ., family = "binomial", data = train)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -3.2020  -0.5922  -0.2794   0.3796   2.7923
##
## Coefficients:
##                            Estimate Std. Error z value Pr(>|z|)
## (Intercept)              -10.223235   1.494324  -6.841 7.84e-12 ***
## Pregnancies1              -0.148765   0.560099  -0.266  0.79054
## Pregnancies2              -0.541952   0.598907  -0.905  0.36552
## Pregnancies3               0.378800   0.599926   0.631  0.52777
## Pregnancies4              -0.259889   0.609871  -0.426  0.67001
## Pregnancies5               0.448373   0.627211   0.715  0.47469
## Pregnancies6              -0.871975   0.785530  -1.110  0.26698
## Pregnancies7               0.233589   0.717607   0.326  0.74479
## Pregnancies8               0.806139   0.841841   0.958  0.33827
## Pregnancies9               2.199401   0.931988   2.360  0.01828 *
## Pregnancies10              0.626614   0.687723   0.911  0.36222
## Glucose                    0.035887   0.005940   6.042 1.53e-09 ***
## BloodPressure             -0.013274   0.014603  -0.909  0.36337
## SkinThickness              0.011565   0.020962   0.552  0.58113
## Insulin                    0.014084   0.003076   4.578 4.69e-06 ***
## BMI                        0.061670   0.030820   2.001  0.04540 *
## DiabetesPedigreeFunction   1.559675   0.592860   2.631  0.00852 **
## Age                        0.024012   0.017931   1.339  0.18053
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 523.55  on 422  degrees of freedom
## Residual deviance: 324.58  on 405  degrees of freedom
## AIC: 360.58
##
## Number of Fisher Scoring iterations: 5
```

```r
#To improvise the model and prevent the loss of data
library(MASS)
```

```
##
## Attaching package: 'MASS'
```

```
## The following object is masked from 'package:dplyr':
##
##     select
```

```r
new_mod=stepAIC(log_reg)
```

```
## Start:  AIC=360.58
## Outcome ~ Pregnancies + Glucose + BloodPressure + SkinThickness +
##     Insulin + BMI + DiabetesPedigreeFunction + Age
##
##                            Df Deviance    AIC
## - Pregnancies              10   341.00 357.00
## - SkinThickness             1   324.89 358.89
## - BloodPressure             1   325.42 359.42
## - Age                       1   326.36 360.36
## <none>                          324.58 360.58
## - BMI                       1   328.65 362.65
## - DiabetesPedigreeFunction  1   331.59 365.59
## - Insulin                   1   347.62 381.62
## - Glucose                   1   369.00 403.00
##
## Step:  AIC=357
## Outcome ~ Glucose + BloodPressure + SkinThickness + Insulin +
##     BMI + DiabetesPedigreeFunction + Age
##
##                            Df Deviance    AIC
## - BloodPressure             1   341.29 355.29
## - SkinThickness             1   341.98 355.98
## <none>                          341.00 357.00
## - BMI                       1   343.97 357.97
## - DiabetesPedigreeFunction  1   348.59 362.59
## - Age                       1   350.28 364.28
## - Insulin                   1   364.54 378.54
## - Glucose                   1   387.50 401.50
##
## Step:  AIC=355.29
## Outcome ~ Glucose + SkinThickness + Insulin + BMI + DiabetesPedigreeFunction +
##     Age
##
##                            Df Deviance    AIC
## - SkinThickness             1   342.21 354.21
## <none>                          341.29 355.29
## - BMI                       1   344.05 356.05
## - DiabetesPedigreeFunction  1   349.18 361.18
## - Age                       1   350.38 362.38
## - Insulin                   1   365.01 377.01
## - Glucose                   1   387.70 399.70
##
## Step:  AIC=354.21
```

```
## Outcome ~ Glucose + Insulin + BMI + DiabetesPedigreeFunction +
##     Age
##
##                            Df Deviance    AIC
## <none>                          342.21 354.21
## - BMI                       1   348.08 358.08
## - DiabetesPedigreeFunction  1   350.08 360.08
## - Age                       1   352.41 362.41
## - Insulin                   1   365.41 375.41
## - Glucose                   1   388.76 398.76
```

```
summary(new_mod)
```

```
##
## Call:
## glm(formula = Outcome ~ Glucose + Insulin + BMI + DiabetesPedigreeFunction +
##     Age, family = "binomial", data = train)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -3.3394  -0.6064  -0.3248   0.5389   2.7558
##
## Coefficients:
##                            Estimate Std. Error z value Pr(>|z|)
## (Intercept)              -11.017419   1.233192  -8.934  < 2e-16 ***
## Glucose                    0.034456   0.005509   6.254 3.99e-10 ***
## Insulin                    0.013432   0.002906   4.623 3.78e-06 ***
## BMI                        0.061768   0.025737   2.400  0.01640 *
## DiabetesPedigreeFunction   1.556857   0.558692   2.787  0.00533 **
## Age                        0.041249   0.012913   3.194  0.00140 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 523.55  on 422  degrees of freedom
## Residual deviance: 342.21  on 417  degrees of freedom
## AIC: 354.21
##
## Number of Fisher Scoring iterations: 5
```

```
new_mod$formula
```

```
## Outcome ~ Glucose + Insulin + BMI + DiabetesPedigreeFunction +
##     Age
```

```
#We use the variables which ever is having more impact on the outcome
new_mod <- glm(new_mod$formula,data = train,family = 'binomial')
summary(new_mod)
```

```
##
## Call:
## glm(formula = new_mod$formula, family = "binomial", data = train)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
```

```
## -3.3394  -0.6064  -0.3248   0.5389   2.7558
##
## Coefficients:
##                             Estimate Std. Error z value Pr(>|z|)
## (Intercept)              -11.017419   1.233192  -8.934  < 2e-16 ***
## Glucose                    0.034456   0.005509   6.254 3.99e-10 ***
## Insulin                    0.013432   0.002906   4.623 3.78e-06 ***
## BMI                        0.061768   0.025737   2.400  0.01640 *
## DiabetesPedigreeFunction   1.556857   0.558692   2.787  0.00533 **
## Age                        0.041249   0.012913   3.194  0.00140 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 523.55  on 422  degrees of freedom
## Residual deviance: 342.21  on 417  degrees of freedom
## AIC: 354.21
##
## Number of Fisher Scoring iterations: 5
```

```r
#Predicting the probability of having the outcome here
pred_prob <- predict(new_mod, newdata = test,type = 'response')
#View(pred_prob)

#Factorising the predictions depending on the threshhold 0.5
pred_class <- as.factor(ifelse(pred_prob >= 0.5, 1, 0))
#if the threshhold/probability is greater than 0.5 then it will be taken as 1 or else 0
#Confusion matrix
t<-table(test$Outcome,pred_class)
```

Here by the help of feature reduction we get a better model without loss of much data in it.

```r
#Classification report


#Defining the functions for finding the accuracy, error rate, prescion and f1_score
prediction_accuracy <- function(new_df){
  accuracy = (t[2,2] + t[1,1]) / (t[2,2] + t[1,2] + t[1,1] + t[2,1])
  accuracy = round(accuracy, 2)
  paste("The prediction accuracy is: ", accuracy)
}

classification_error_rate <- function(new_df){
  classification_error_rate = (t[1,2] + t[2,1]) / (t[2,2] + t[1,2] + t[1,1] + t[2,1])
  classification_error_rate = round(classification_error_rate, 2)
  paste0('The classification error rate is: ',classification_error_rate)
}

precision <- function(new_df){
  p       = (t[2,2] / (t[2,2] + t[1,2]))
  p       = round(p, 2)
  paste0('The precision is: ', p)
}
```

```
f1.score <- function(new_df){
  p = (t[2,2] / (t[2,2] + t[1,2]))
  s = t[2,2] / (t[2,2] + t[2,1])
  f1 = (2*p*s) / (p + s)
  f1 = round(f1, 2)
  paste0("The F1 score is: ", f1)
}
```

We created a function for calculating several results which will be called whenever needed.

```
#Detailed classification report
caret::confusionMatrix(pred_class, test$Outcome,positive='0')
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 112  23
##          1  17  30
##
##                Accuracy : 0.7802
##                  95% CI : (0.713, 0.8381)
##     No Information Rate : 0.7088
##     P-Value [Acc > NIR] : 0.01869
##
##                   Kappa : 0.4492
##
##  Mcnemar's Test P-Value : 0.42920
##
##             Sensitivity : 0.8682
##             Specificity : 0.5660
##          Pos Pred Value : 0.8296
##          Neg Pred Value : 0.6383
##              Prevalence : 0.7088
##          Detection Rate : 0.6154
##    Detection Prevalence : 0.7418
##       Balanced Accuracy : 0.7171
##
##        'Positive' Class : 0
##
```

```
prediction_accuracy(test)
```

```
## [1] "The prediction accuracy is:  0.78"
```

```
classification_error_rate(test)
```

```
## [1] "The classification error rate is: 0.22"
```

```
precision(test)
```

```
## [1] "The precision is: 0.64"
```

```
f1.score(test)
```

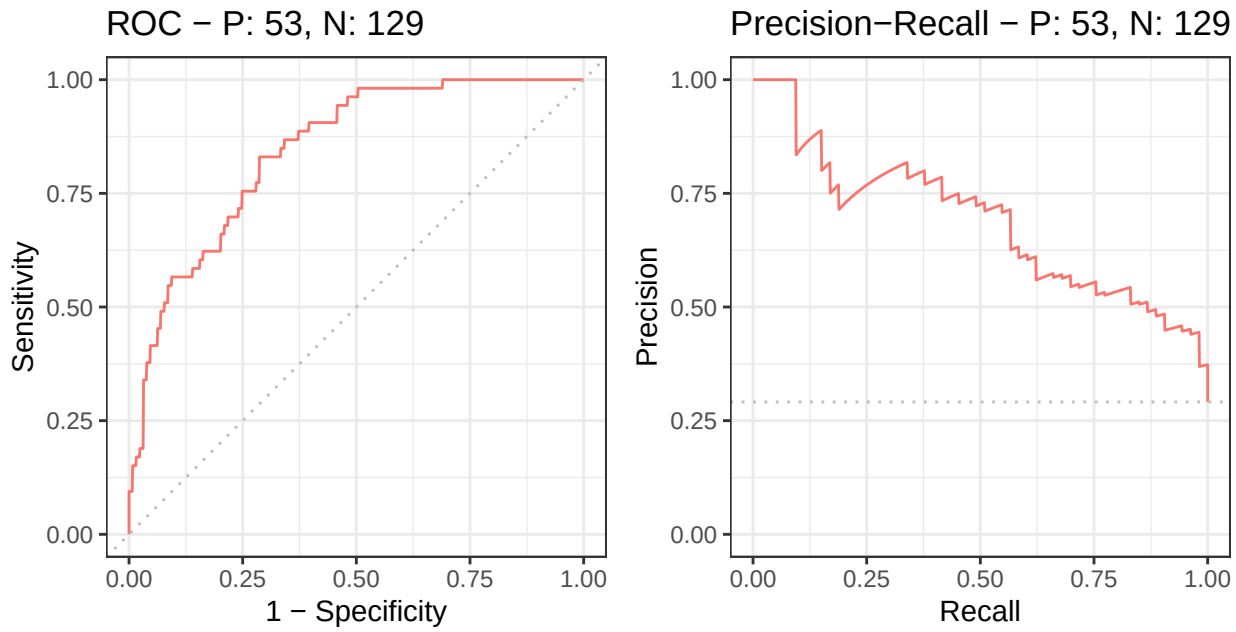
```
## [1] "The F1 score is: 0.6"
```

From the help of improvised Logistic regression, we were able to predict the outcome with 78% accuracy in it.

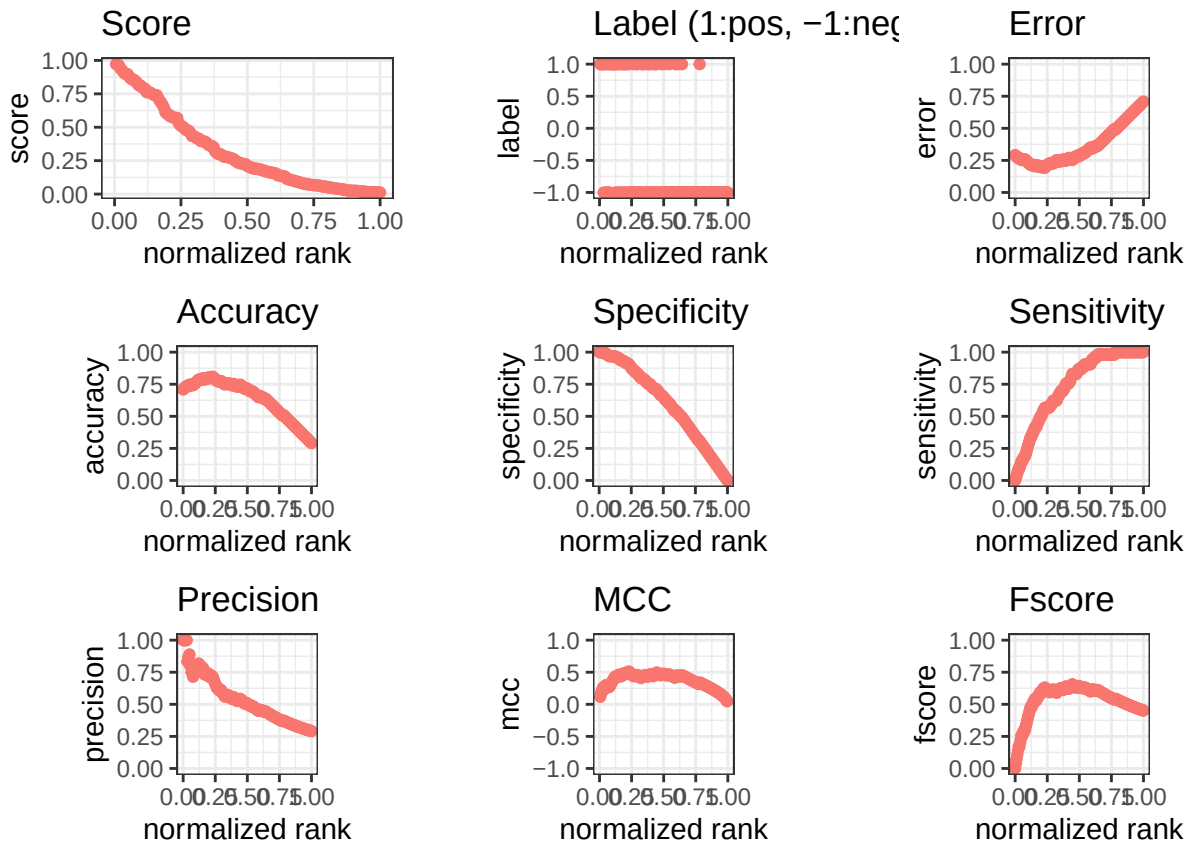
```
#ROC curve
#install.packages("precrec")
library(precrec)
```

```
## Warning: package 'precrec' was built under R version 4.0.5
```

```
precrec_obj <- evalmod(scores = pred_prob, labels = test$Outcome)
autoplot(precrec_obj)
```




```
#All possible plots of various model
precrec_obj <- evalmod(scores = pred_prob, labels = test$Outcome,mode="basic")
autoplot(precrec_obj)
```




```
##########################
```

(2)Naive Bayes :

```
###Naive Bayes


# set.seed(100)
#
# train_indices <- sample(1:nrow(df), 0.70 * nrow(df))
# train <- df[train_indices, ]
# test <- df[-train_indices,]

library(e1071)
```

```
## Warning: package 'e1071' was built under R version 4.0.5
```

```
##
## Attaching package: 'e1071'
```

```
## The following object is masked from 'package:Hmisc':
##
##     impute
```

```
pred_3<-naiveBayes(Outcome~.,data=train)
pred_3_prob<-predict(pred_3,type="raw",newdata=test)
pred_3_class<-as.factor(ifelse(pred_3_prob[,2]>.5,1,0))

#Confusion matrix
```

```
t<-table(test$Outcome,pred_3_class)


#Classification Report
caret::confusionMatrix(pred_3_class, test$Outcome,positive='0')
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 106  18
##          1  23  35
##
##                Accuracy : 0.7747
##                  95% CI : (0.7071, 0.8332)
##     No Information Rate : 0.7088
##     P-Value [Acc > NIR] : 0.0281
##
##                   Kappa : 0.469
##
##  Mcnemar's Test P-Value : 0.5322
##
##             Sensitivity : 0.8217
##             Specificity : 0.6604
##          Pos Pred Value : 0.8548
##          Neg Pred Value : 0.6034
##              Prevalence : 0.7088
##          Detection Rate : 0.5824
##    Detection Prevalence : 0.6813
##       Balanced Accuracy : 0.7410
##
##        'Positive' Class : 0
##
```

```
prediction_accuracy(test)
```

```
## [1] "The prediction accuracy is:  0.77"
```

```
classification_error_rate(test)
```

```
## [1] "The classification error rate is: 0.23"
```

```
precision(test)
```

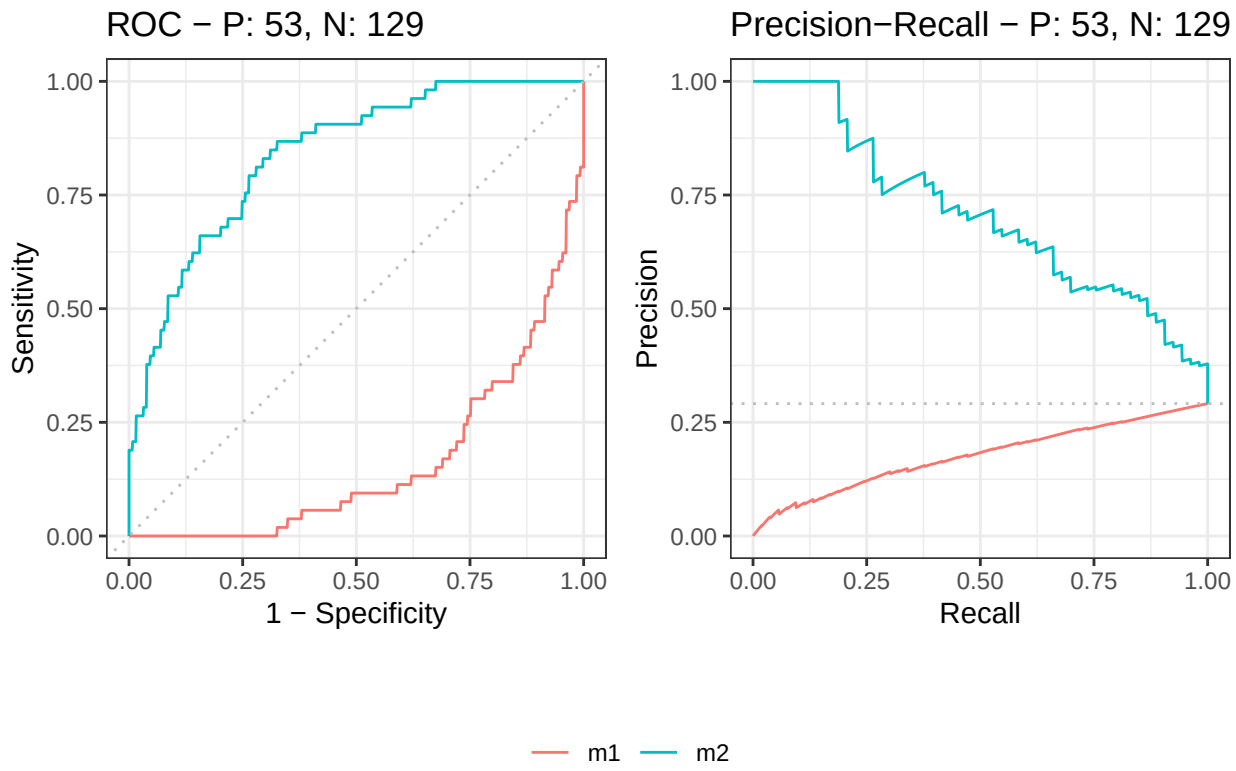
```
## [1] "The precision is: 0.6"
```

```
f1.score(test)
```

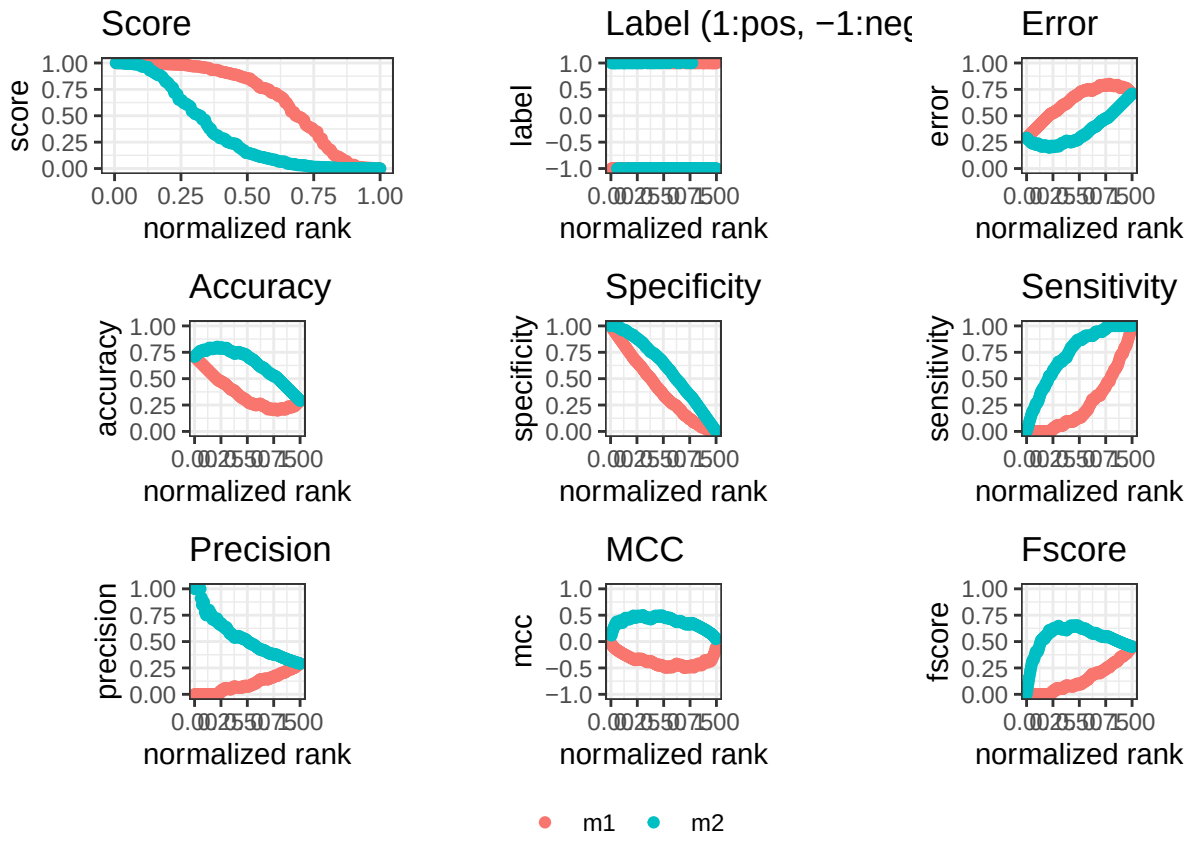
```
## [1] "The F1 score is: 0.63"
```

Here with the help of NAive Bayes model, we got 77% accuracy.

```
#ROC curve
precrec_obj <- evalmod(scores = pred_3_prob, labels = test$Outcome)
autoplot(precrec_obj)
```




```
precrec_obj <- evalmod(scores = pred_3_prob, labels = test$Outcome,mode="basic")
autoplot(precrec_obj)
```




```
############################
```

(3) Support Vector Machine (SVM)

```
##SVM


# set.seed(100)
# train_indices <- sample(1:nrow(df), 0.70 * nrow(df))
# train <- df[train_indices, ]
# test <- df[-train_indices,]


library(e1071)
classifier = svm(formula = Outcome ~ .,
                 data = train,
                 type = 'C-classification',
                 kernel = 'linear')
pred_4 <- predict(classifier,newdata=test,type="response")
#Confusion matrix
t <- table(test$Outcome,pred_4)


#Classification report
caret::confusionMatrix(pred_4, test$Outcome,positive='0')
```

```
## Confusion Matrix and Statistics
```

```
##
##           Reference
## Prediction   0   1
##          0 118  26
##          1  11  27
##
##                Accuracy : 0.7967
##                  95% CI : (0.7308, 0.8526)
##     No Information Rate : 0.7088
##     P-Value [Acc > NIR] : 0.004616
##
##                   Kappa : 0.4627
##
##  Mcnemar's Test P-Value : 0.021359
##
##             Sensitivity : 0.9147
##             Specificity : 0.5094
##          Pos Pred Value : 0.8194
##          Neg Pred Value : 0.7105
##              Prevalence : 0.7088
##          Detection Rate : 0.6484
##    Detection Prevalence : 0.7912
##       Balanced Accuracy : 0.7121
##
##        'Positive' Class : 0
##
```

```
prediction_accuracy(test)
```

```
## [1] "The prediction accuracy is:  0.8"
```

```
classification_error_rate(test)
```

```
## [1] "The classification error rate is: 0.2"
```

```
precision(test)
```

```
## [1] "The precision is: 0.71"
```

```
f1.score(test)
```

```
## [1] "The F1 score is: 0.59"
```

Here by SVM model, we got the predictions with 79% accuracy. We can observe that this is better than Logistic regression and Naive Bayes model.

```
#ROC curve
library(pROC)
```

```
## Warning: package 'pROC' was built under R version 4.0.5
```

```
## Type 'citation("pROC")' for a citation.
```

```
##
## Attaching package: 'pROC'
```

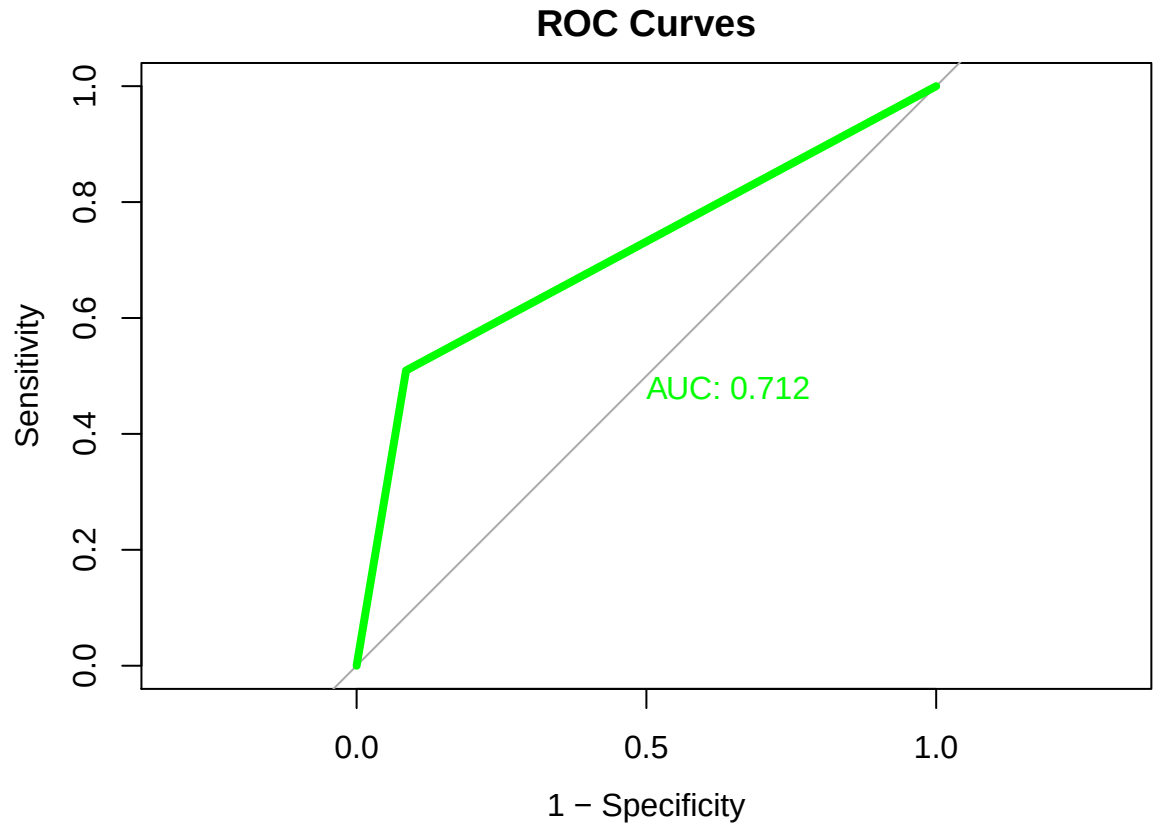
```
## The following object is masked from 'package:precrec':
##
##     auc
```

```
## The following objects are masked from 'package:stats':
##
##     cov, smooth, var
```

```
pred_4 <- as.vector(pred_4, mode = "numeric")
lrROC <- roc(test$Outcome ~
               pred_4,plot=TRUE,print.auc=TRUE,col="green",
             lwd =4,legacy.axes=TRUE,main="ROC Curves")
```

```
## Setting levels: control = 0, case = 1
```

```
## Setting direction: controls < cases
```




```
##################
```

(4) K-Nearest Neighbours (KNN) :

```
#KNN

# set.seed(100)
#
# train_indices <- sample(1:nrow(df), 0.70 * nrow(df))
# train <- df[train_indices, ]
# test <- df[-train_indices,]
# nrow(train)
# nrow(test)
library(class)
pred_2<- knn(train,test,train$Outcome,k=5)
#Confusion matrix
```

```r
t<-table(test$Outcome,pred_2)

#Classification Report

caret::confusionMatrix(pred_2, test$Outcome,positive='0')
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 120  18
##          1   9  35
##
##                Accuracy : 0.8516
##                  95% CI : (0.7915, 0.8999)
##     No Information Rate : 0.7088
##     P-Value [Acc > NIR] : 4.966e-06
##
##                   Kappa : 0.6217
##
##  Mcnemar's Test P-Value : 0.1237
##
##             Sensitivity : 0.9302
##             Specificity : 0.6604
##          Pos Pred Value : 0.8696
##          Neg Pred Value : 0.7955
##              Prevalence : 0.7088
##          Detection Rate : 0.6593
##    Detection Prevalence : 0.7582
##       Balanced Accuracy : 0.7953
##
##        'Positive' Class : 0
##
```

```r
prediction_accuracy(test)
```

```
## [1] "The prediction accuracy is:  0.85"
```

```r
classification_error_rate(test)
```

```
## [1] "The classification error rate is: 0.15"
```

```r
precision(test)
```

```
## [1] "The precision is: 0.8"
```

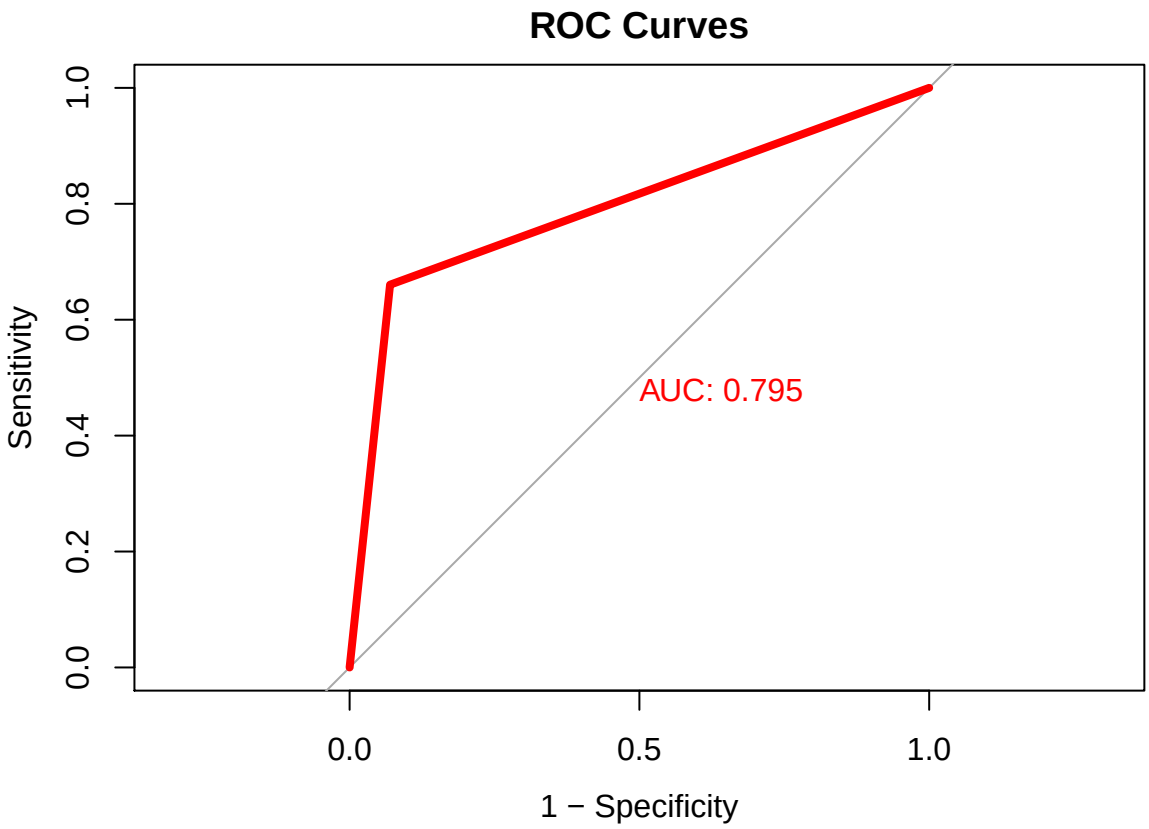
```r
f1.score(test)
```

```
## [1] "The F1 score is: 0.72"
```

```r
#ROC curve
library(pROC)
pred_2 <- as.vector(pred_2, mode = "numeric")
lrROC <- roc(test$Outcome ~
               pred_2,plot=TRUE,print.auc=TRUE,col="red",
             lwd =4,legacy.axes=TRUE,main="ROC Curves")
```

```
## Setting levels: control = 0, case = 1
```

```
## Setting direction: controls < cases
```



Now from all these models, we can observe that KNN gives the best accuracy with 85.16% with only 27 false values out of 182 results.

Finally we observe the variables : Glucose, Insulin affects Diabetes. Even BMI and skin thickness matters a bit for the patients of Diabetes.