Final Project: Diabities in Women

Jyoti Phogat

2022-12-17

```
#rm(list = ls())
data <- read.csv("diabetes.csv", header = T)
#View(data)
```

Load Required Libraries

```
library(ggplot2)
library(car)
```

```
## Loading required package: carData
```

```
library(caret)
```

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

```
library(class)
install.packages("devtools")
```

```
## Installing package into '/cloud/lib/x86_64-pc-linux-gnu-library/4.2'
## (as 'lib' is unspecified)
```

```
library(devtools)
```

```
## Loading required package: usethis
```

```
library(e1071)
library(Hmisc)
```

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

```
##
## Attaching package: 'survival'
```

```
## The following object is masked from 'package:caret':
##
##     cluster
```

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

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

```
## The following object is masked from 'package:e1071':
##
##     impute
```

```
## The following objects are masked from 'package:base':
##
```

```
##     format.pval, units
```

```
install.packages("lmtest")
```

```
## Installing package into '/cloud/lib/x86_64-pc-linux-gnu-library/4.2'
## (as 'lib' is unspecified)
```

```
library(lmtest)
```

```
## Loading required package: zoo
```

```
##
## Attaching package: 'zoo'
```

```
## The following objects are masked from 'package:base':
##
##     as.Date, as.Date.numeric
```

```
library(plyr)
```

```
##
## Attaching package: 'plyr'
```

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

```
library(pROC)
```

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

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

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

```
library(psych)
```

```
##
## Attaching package: 'psych'
```

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

```
## The following object is masked from 'package:car':
##
##     logit
```

```
## The following objects are masked from 'package:ggplot2':
##
##     %+%, alpha
```

```
library(ROCR)
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:Hmisc':
##
##     src, summarize

## The following object is masked from 'package:car':
##
##     recode

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

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

```
library(corrplot)
```

```
## corrplot 0.92 loaded
```

```
library(caTools)
install.packages("DataExplorer")
```

```
## Installing package into '/cloud/lib/x86_64-pc-linux-gnu-library/4.2'
## (as 'lib' is unspecified)
```

```
library(DataExplorer)
```

```
names(data) <- c("NoPreg", "PlaGluConc", "DiastolicBP", "TSkinThick", "Test", "BMI", "DiabPediFunc", "Ag
dim(data)
```

```
## [1] 2000    9
```

```
str(data)
```

```
## 'data.frame':    2000 obs. of  9 variables:
##  $ NoPreg      : int  2 0 0 0 1 0 4 8 2 2 ...
##  $ PlaGluConc  : int  138 84 145 135 139 173 99 194 83 89 ...
##  $ DiastolicBP : int  62 82 0 68 62 78 72 80 65 90 ...
##  $ TSkinThick  : int  35 31 0 42 41 32 17 0 28 30 ...
##  $ Test        : int  0 125 0 250 480 265 0 0 66 0 ...
##  $ BMI         : num  33.6 38.2 44.2 42.3 40.7 46.5 25.6 26.1 36.8 33.5 ...
##  $ DiabPediFunc: num  0.127 0.233 0.63 0.365 0.536 ...
##  $ Age         : int  47 23 31 24 21 58 28 67 24 42 ...
##  $ Class       : int  1 0 1 1 0 0 0 0 0 0 ...
```

```
summary(data)
```

```
##      NoPreg         PlaGluConc     DiastolicBP       TSkinThick
##  Min.   : 0.000   Min.   :  0.0   Min.   :  0.00   Min.   :  0.00
##  1st Qu.: 1.000   1st Qu.: 99.0   1st Qu.: 63.50   1st Qu.:  0.00
##  Median : 3.000   Median :117.0   Median : 72.00   Median : 23.00
##  Mean   : 3.704   Mean   :121.2   Mean   : 69.15   Mean   : 20.93
##  3rd Qu.: 6.000   3rd Qu.:141.0   3rd Qu.: 80.00   3rd Qu.: 32.00
##  Max.   :17.000   Max.   :199.0   Max.   :122.00   Max.   :110.00
##       Test             BMI         DiabPediFunc         Age
##  Min.   :  0.00   Min.   : 0.00   Min.   :0.0780   Min.   :21.00
```

```
##  1st Qu.:  0.00   1st Qu.:27.38   1st Qu.:0.2440   1st Qu.:24.00
##  Median : 40.00   Median :32.30   Median :0.3760   Median :29.00
##  Mean   : 80.25   Mean   :32.19   Mean   :0.4709   Mean   :33.09
##  3rd Qu.:130.00   3rd Qu.:36.80   3rd Qu.:0.6240   3rd Qu.:40.00
##  Max.   :744.00   Max.   :80.60   Max.   :2.4200   Max.   :81.00
##      Class
##  Min.   :0.000
##  1st Qu.:0.000
##  Median :0.000
##  Mean   :0.342
##  3rd Qu.:1.000
##  Max.   :1.000
```

```
  describe(data)
```

```
##              vars    n   mean     sd median trimmed   mad   min    max  range
## NoPreg          1 2000   3.70   3.31   3.00    3.30  2.97  0.00  17.00  17.00
## PlaGluConc      2 2000 121.18  32.07 117.00  119.71 29.65  0.00 199.00 199.00
## DiastolicBP     3 2000  69.15  19.19  72.00   71.33 11.86  0.00 122.00 122.00
## TSkinThick      4 2000  20.93  16.10  23.00   20.35 17.79  0.00 110.00 110.00
## Test            5 2000  80.25 111.18  40.00   58.43 59.30  0.00 744.00 744.00
## BMI             6 2000  32.19   8.15  32.30   32.07  6.97  0.00  80.60  80.60
## DiabPediFunc    7 2000   0.47   0.32   0.38    0.42  0.25  0.08   2.42   2.34
## Age             8 2000  33.09  11.79  29.00   31.32 10.38 21.00  81.00  60.00
## Class           9 2000   0.34   0.47   0.00    0.30  0.00  0.00   1.00   1.00
##                skew kurtosis   se
## NoPreg         0.98     0.40 0.07
## PlaGluConc     0.16     0.55 0.72
## DiastolicBP   -1.85     5.30 0.43
## TSkinThick     0.21     0.15 0.36
## Test           1.99     5.10 2.49
## BMI           -0.09     4.11 0.18
## DiabPediFunc   1.81     4.98 0.01
## Age            1.18     0.82 0.26
## Class          0.67    -1.56 0.01
```

```
attach(data)

prop.table(table(NoPreg,Class),1)*100
```

```
##       Class
## NoPreg         0         1
##     0   66.77741  33.22259
##     1   77.80899  22.19101
##     2   83.09859  16.90141
##     3   64.10256  35.89744
##     4   64.92147  35.07853
##     5   63.12057  36.87943
##     6   66.41221  33.58779
##     7   44.00000  56.00000
##     8   44.79167  55.20833
##     9   40.00000  60.00000
##     10  59.25926  40.74074
##     11  37.50000  62.50000
##     12  56.52174  43.47826
```

```
##     13  36.36364  63.63636
##     14   0.00000 100.00000
##     15   0.00000 100.00000
##     17   0.00000 100.00000
```

```
table(Class)
```

```
## Class
##    0    1
## 1316  684
```

```
268/(500+268)
```

```
## [1] 0.3489583
```

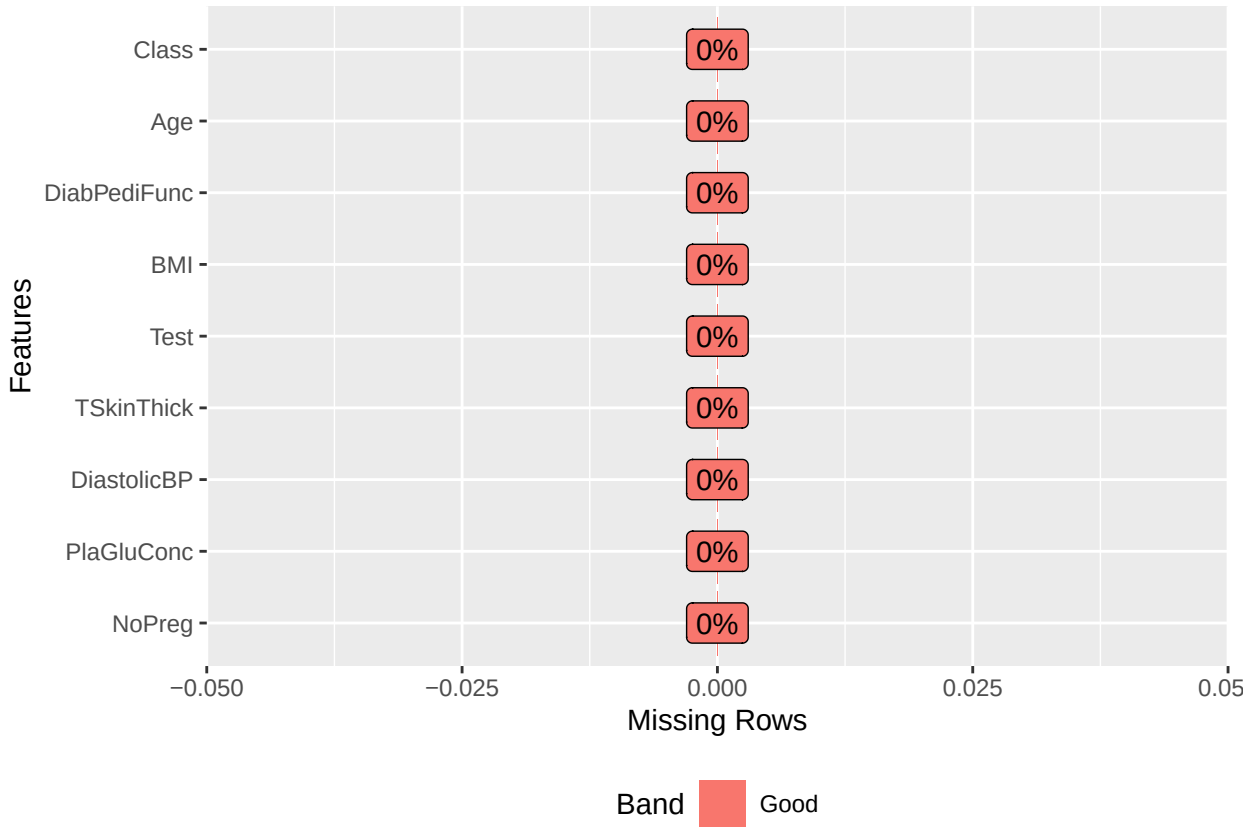
```
mystats <- function(x)
{
  nmiss<-sum(is.na(x))   #to calculate the missing values
  a <- x[!is.na(x)]
  m <- mean(a)           #to calculate the mean
  n <- length(a)         #the length
  s <- sd(a)             #the standard devistion
  min <- min(a)          #the minimum value
  q1<-quantile(a,0.01)   #the different percentiles
  q5<-quantile(a,0.05)
  q95<-quantile(a,0.95)
  q99<-quantile(a,0.99)
  max <- max(a)          #the max value
  UC <- m+3*s            #the upper limit
  LC <- m-3*s            #the lower limit
  outlier_flag<- max>UC | min<LC #mark the variable/data with outlierflag, if it is above Upper cut-off
  return(c(n=n, nmiss=nmiss, outlier_flag=outlier_flag, mean=m, stdev=s,min = min, q1=q1,q5=q5,q95=q95,
}
```

```
#select the variables from the dataset, on which the calculations are to be performed.
diag_stats<-t(data.frame(apply(data[,c(1:9)], 2, mystats)))
#View(diag_stats)
```

```
# Checking null data
sapply(data,function(x) sum(is.na(x)))
```

```
##       NoPreg   PlaGluConc  DiastolicBP   TSkinThick         Test          BMI
##            0            0            0            0            0            0
## DiabPediFunc          Age        Class
##            0            0            0
```

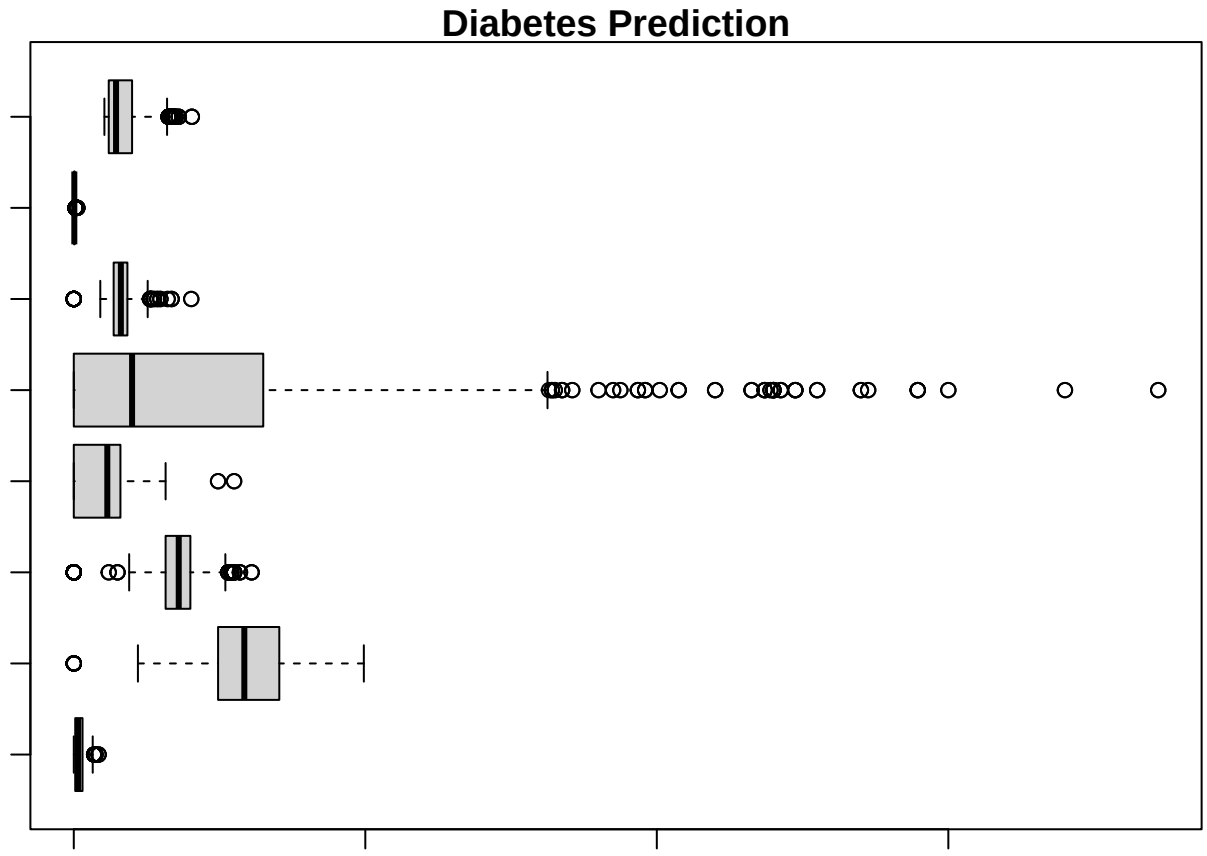
```
plot_missing(data)
```




```
# Checking for Outliers
par("mar")
```

```
## [1] 5.1 4.1 4.1 2.1
```

```
par(mar=c(1,1,1,1))
boxplot(data[1:8], horizontal=TRUE,las=1, main="Diabetes Prediction")
```




```
# Checking # of unique values in each column
sapply(data,function(x) length(unique(x)))
```

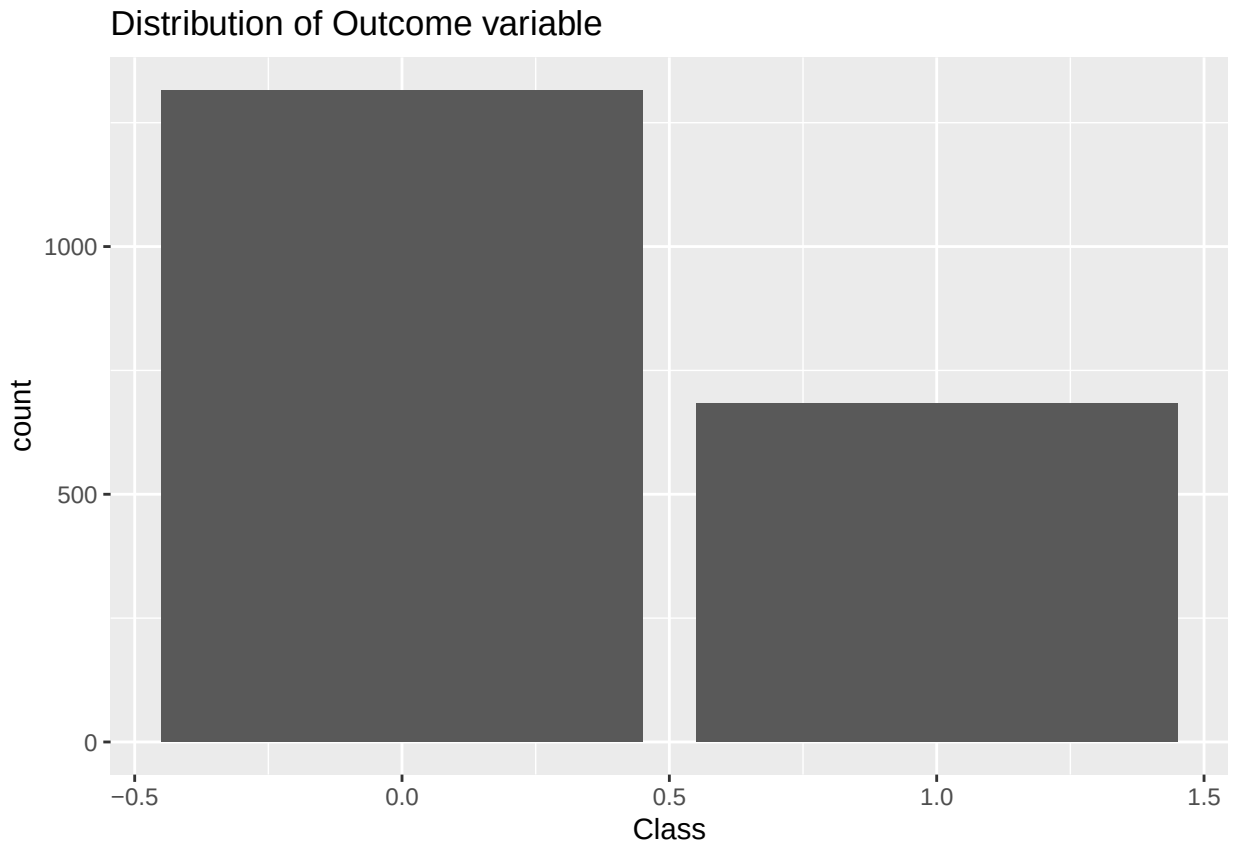
```
##       NoPreg   PlaGluConc  DiastolicBP   TSkinThick         Test          BMI
##           17          136           47           53          182          247
## DiabPediFunc          Age        Class
##          505           52            2
```

```
# Target variable
length(which(data$Class=="1"))*100/nrow(data)
```

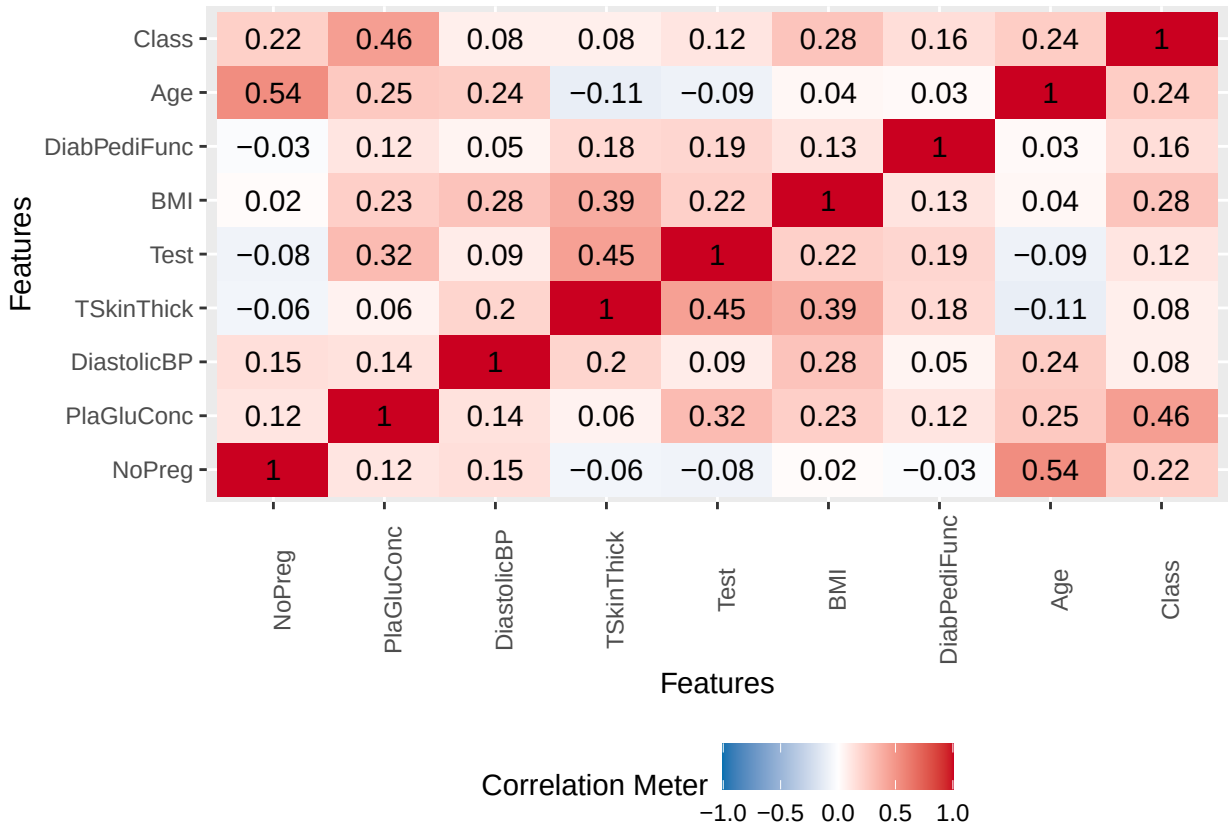
```
## [1] 34.2
```

```
ggplot(data,aes(Class,fill = Class)) +
  geom_bar() +
  ggtitle("Distribution of Outcome variable")
```

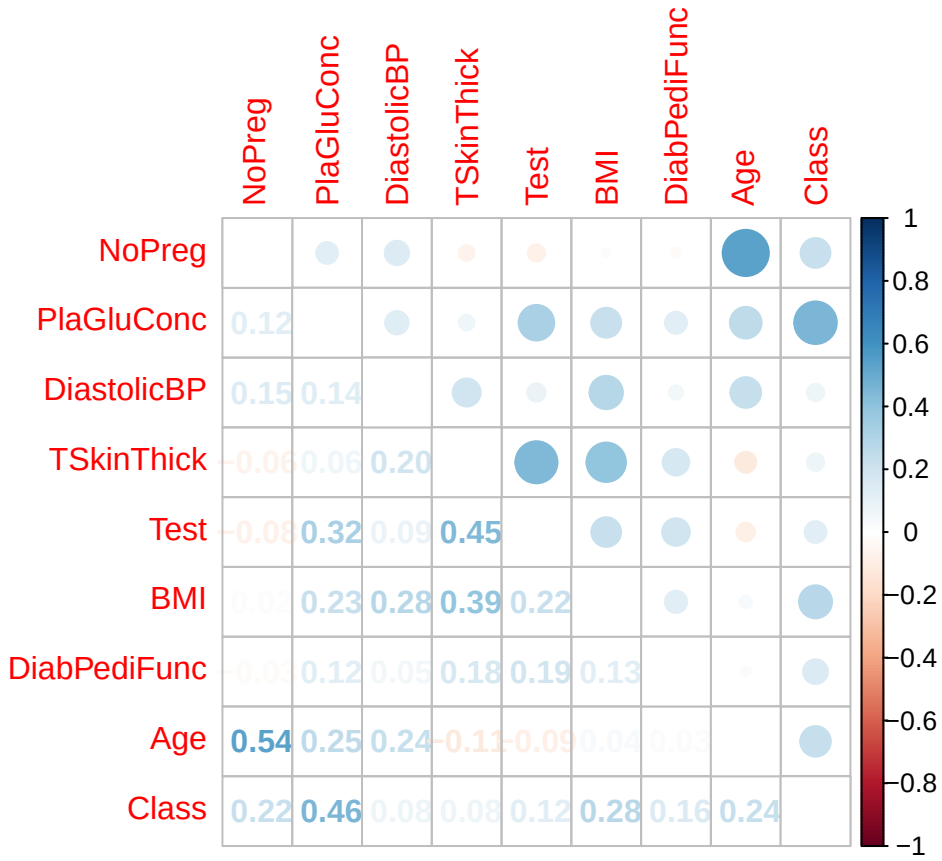
```
## Warning: The following aesthetics were dropped during statistical transformation: fill
## i This can happen when ggplot fails to infer the correct grouping structure in
##   the data.
## i Did you forget to specify a `group` aesthetic or to convert a numerical
##   variable into a factor?
```



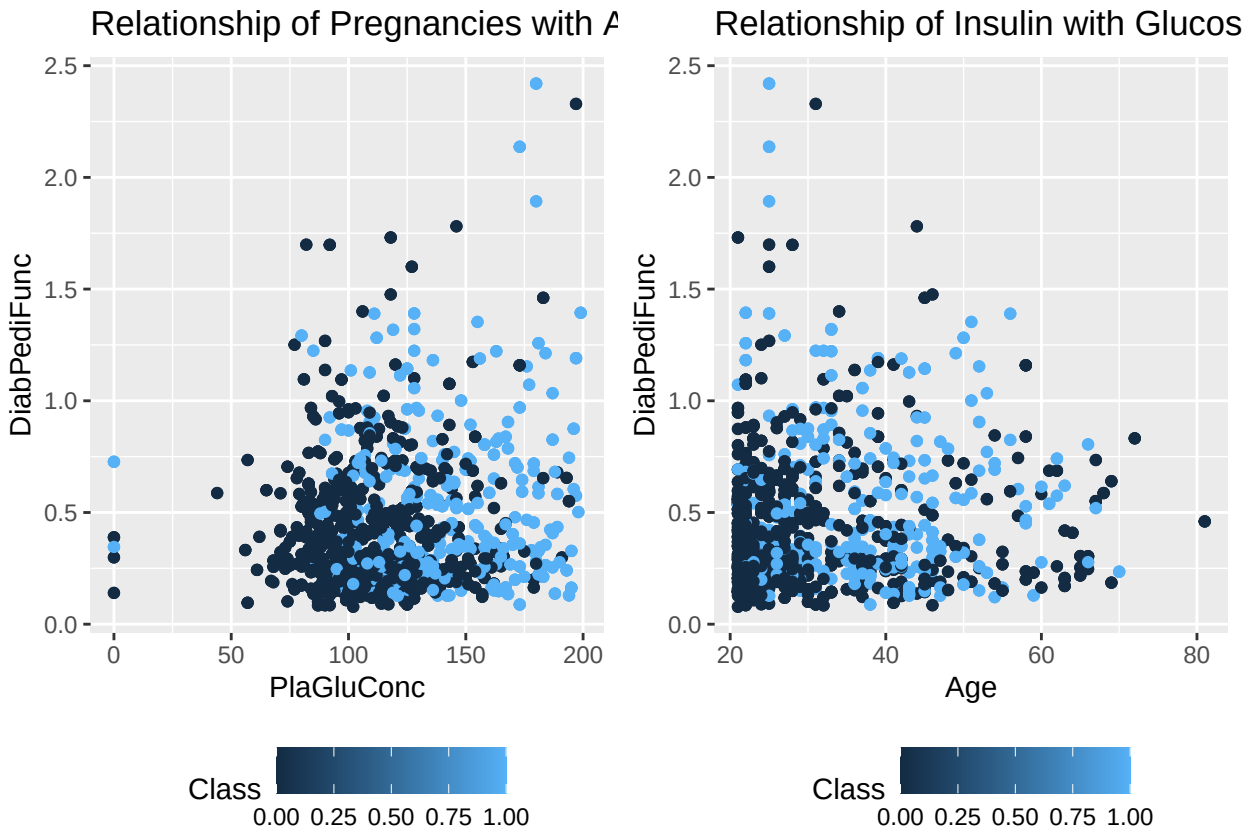

```
# Correlation check
#dev.off()
library(corrplot)
cormat <- cor(data[,c(1:9)])
#cor.plot(cormat)
diag (cormat) = 0 #Remove self correlations
plot_correlation(data[,c(1:9)])
```



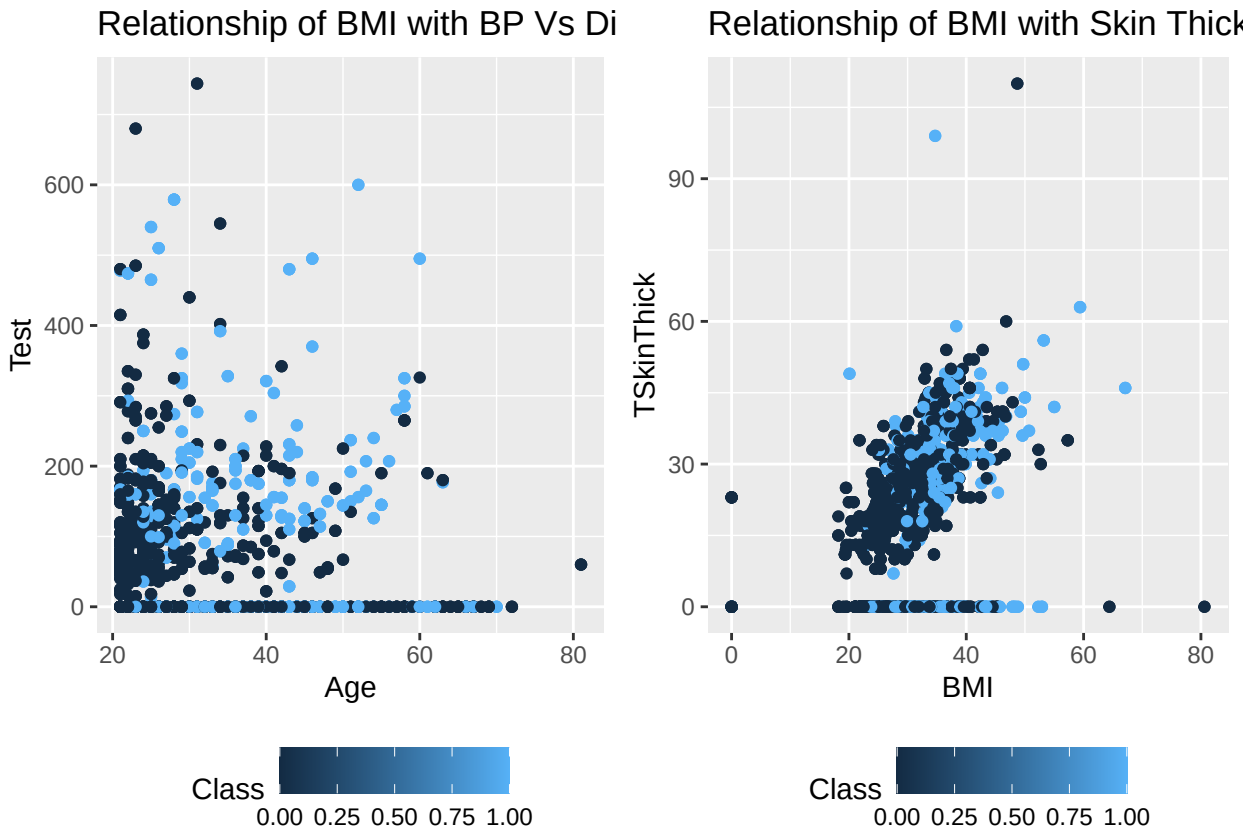

```
corrplot.mixed(cormat,tl.pos = "lt")
```



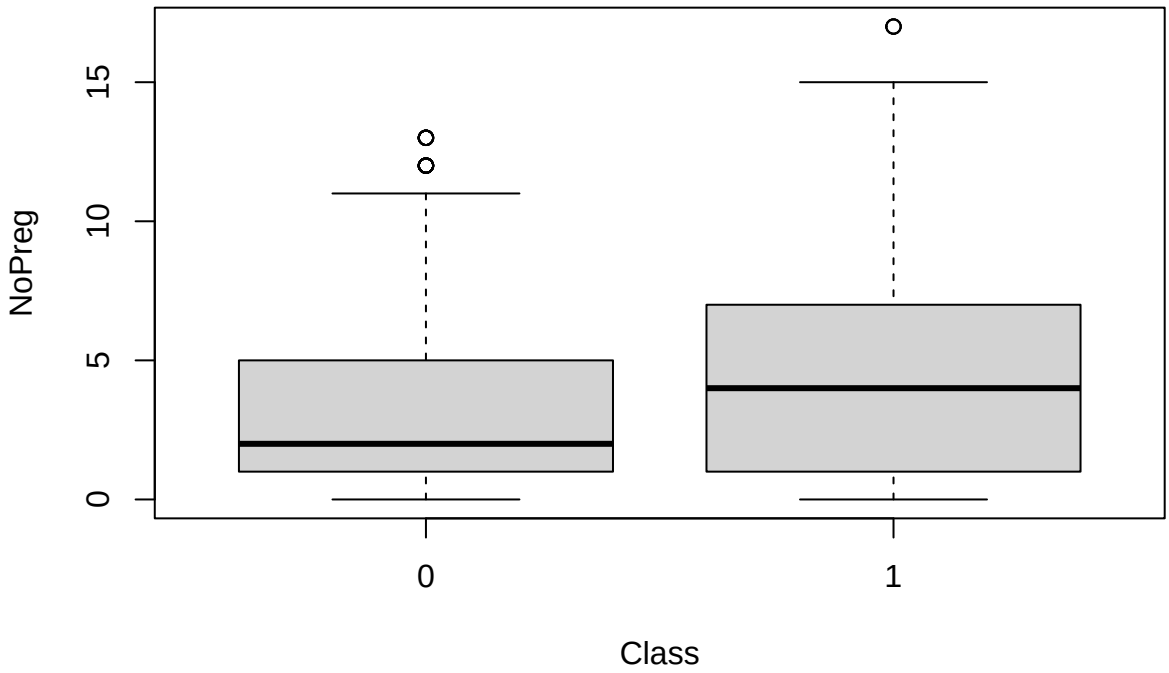

```
#PLOT - Independent Variables vs Dependent Variable
##1. SCATTER PLOT
p1 <- ggplot(data, aes(x = PlaGluConc, y = DiabPediFunc)) +
  geom_point(aes(color=Class)) +
  theme(legend.position = "bottom") +
  ggtitle("Relationship of Pregnancies with Age Vs Diabetes")

p2 <- ggplot(data,aes(x=Age, y=DiabPediFunc))+
  geom_point(aes(color=Class))+
  theme(legend.position = "bottom") +
  ggtitle("Relationship of Insulin with Glucose Vs Diabetes")

gridExtra::grid.arrange(p1, p2, ncol = 2)
```



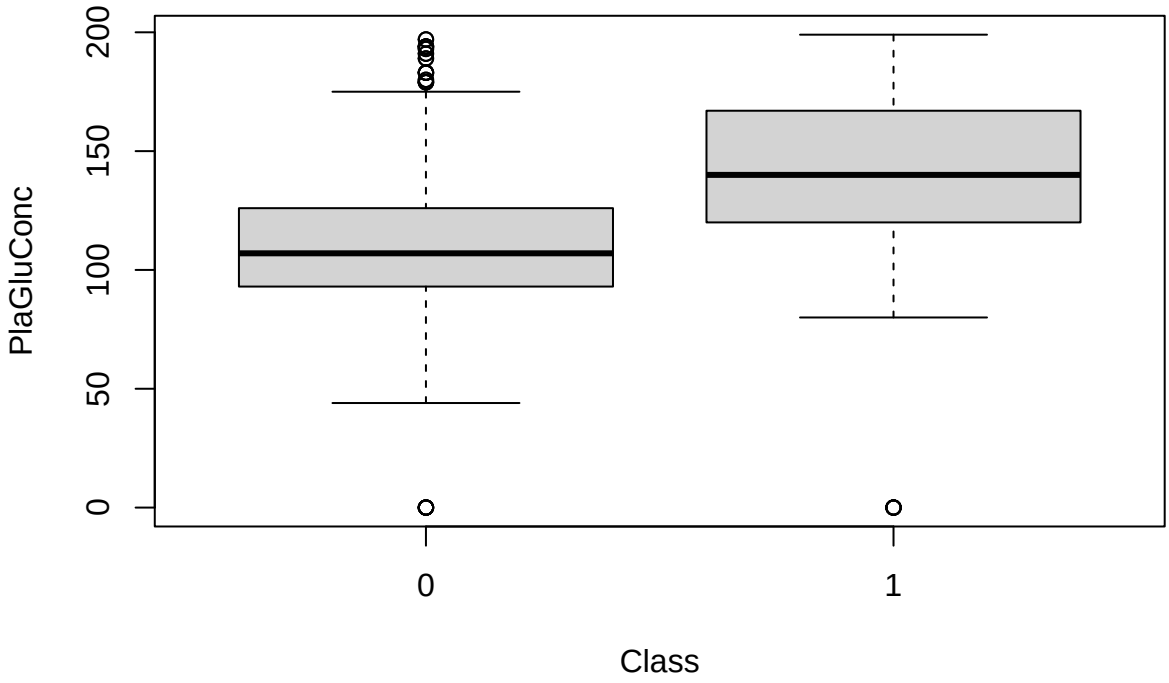

```
## 2. SCATTER PLOT
p1 <- ggplot(data,aes(x=Age,y=Test))+
  geom_point(aes(color=Class))+
  theme(legend.position = "bottom") +
  ggtitle("Relationship of BMI with BP Vs Diabetes")

p2 <- ggplot(data,aes(x=BMI,y=TSkinThick))+
  geom_point(aes(color=Class))+
  theme(legend.position = "bottom") +
  ggtitle("Relationship of BMI with Skin Thickness Vs Diabetes")

gridExtra::grid.arrange(p1, p2, ncol = 2)
```



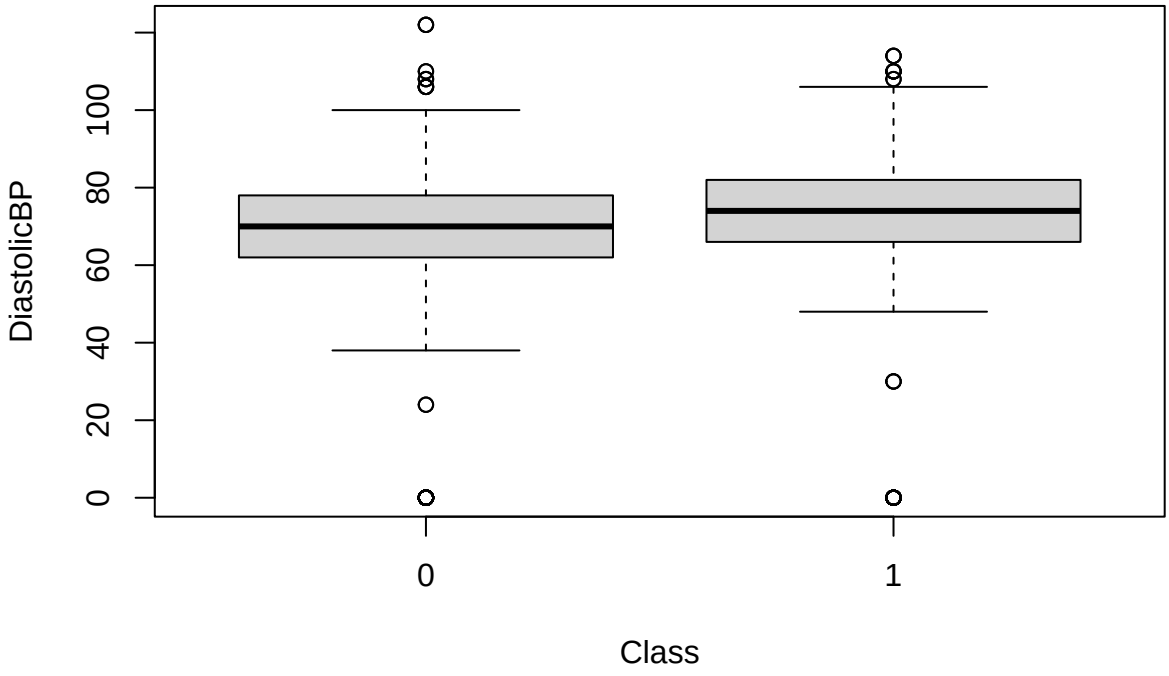

```
## 3. BOXPLOT
boxplot(NoPreg~Class)
```



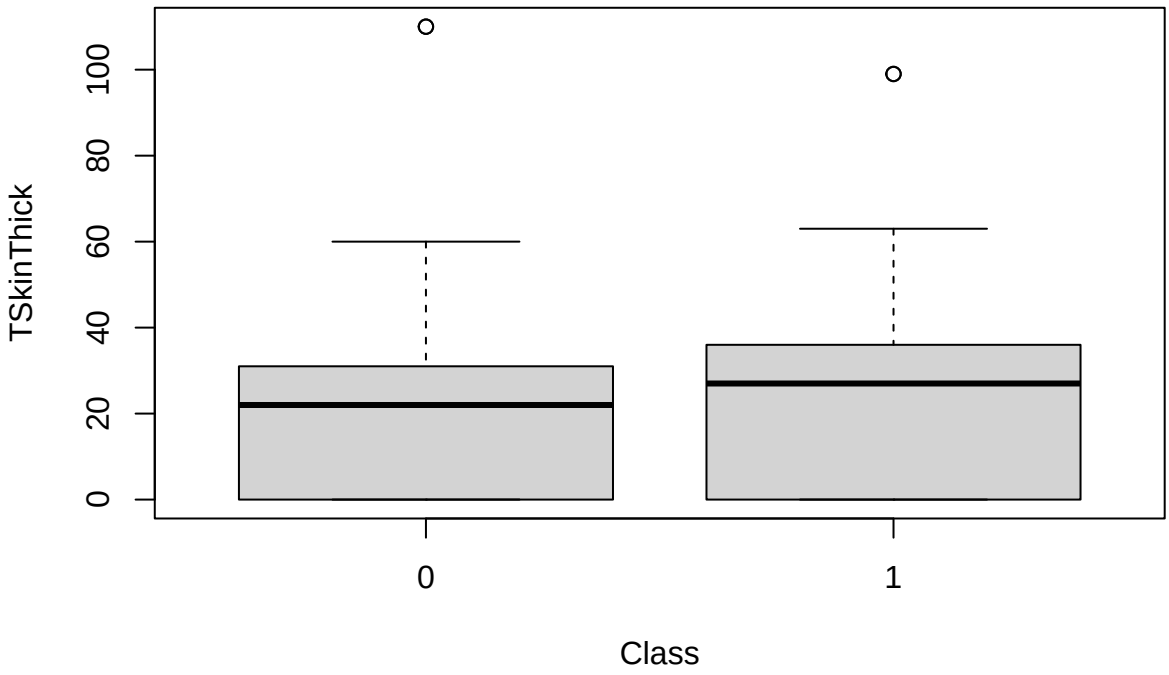

```
boxplot(PlaGluConc~Class)
```



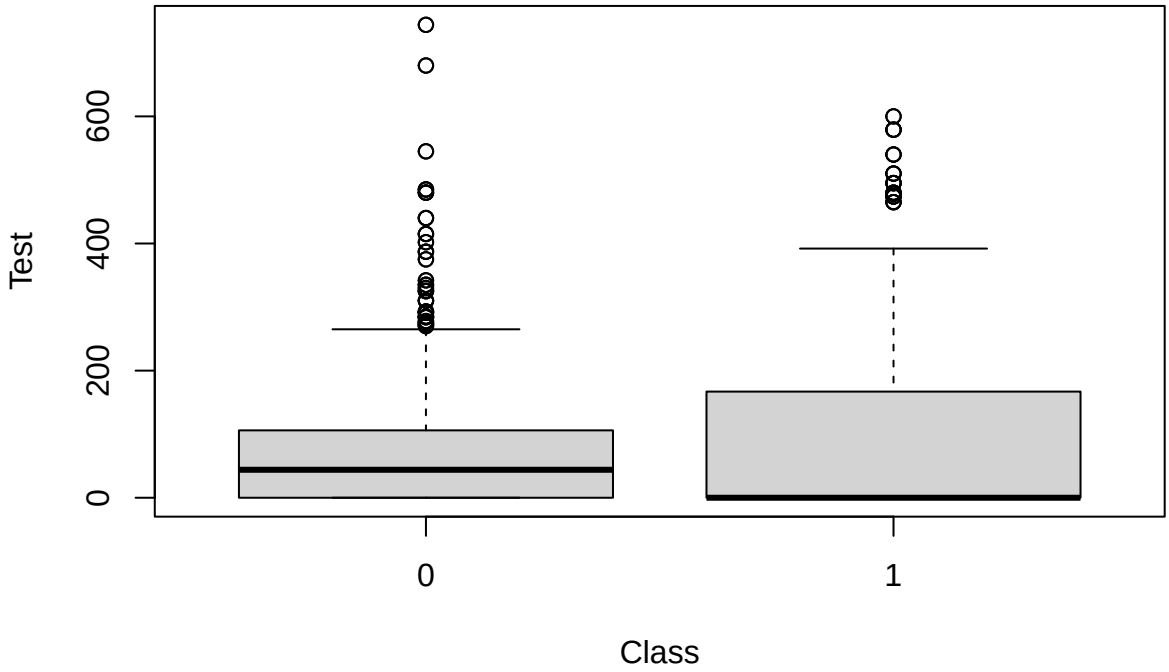

```
boxplot(DiastolicBP~Class)
```



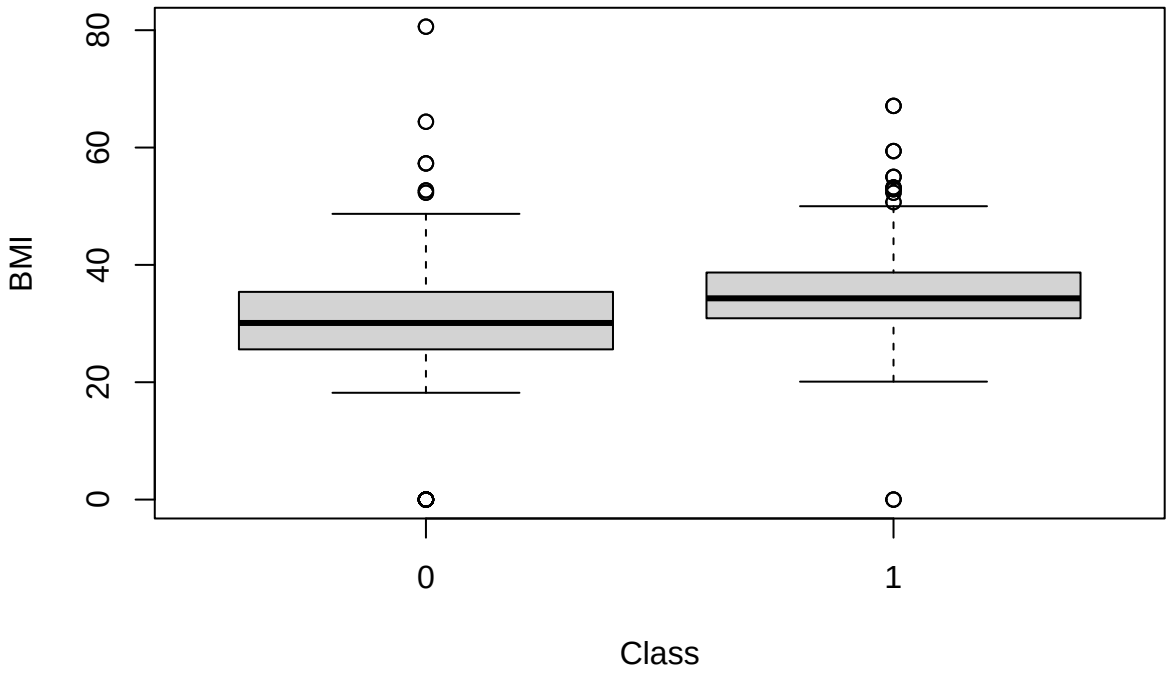

```
boxplot(TSkinThick~Class)
```



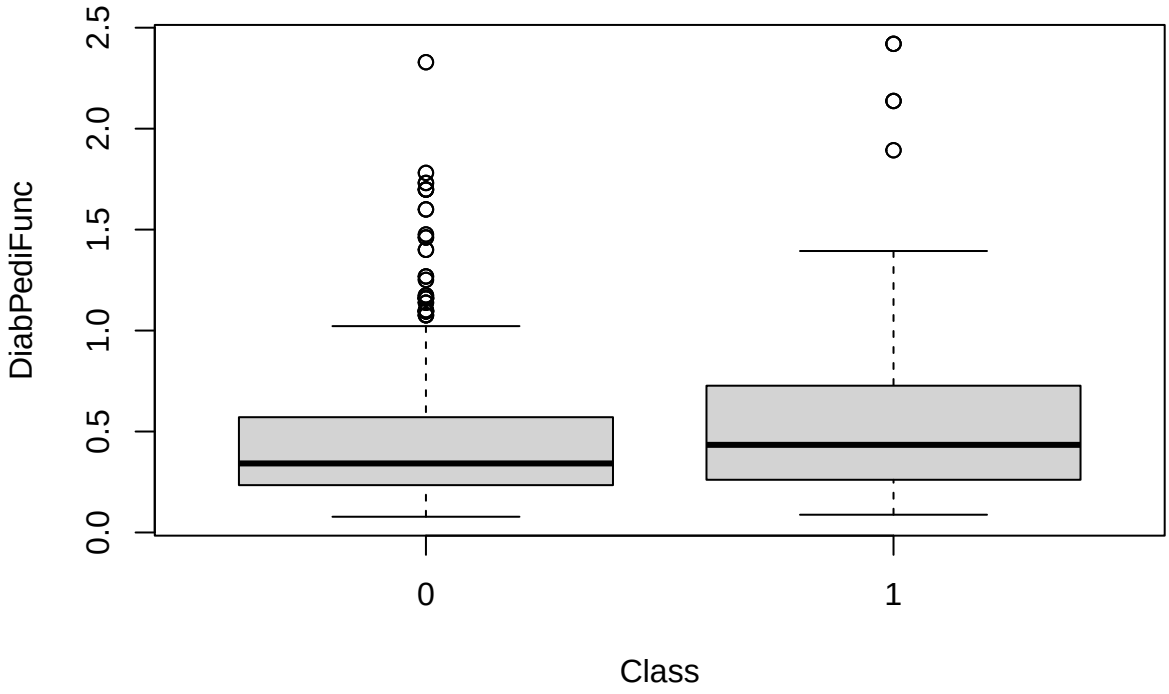

```
boxplot(Test~Class)
```



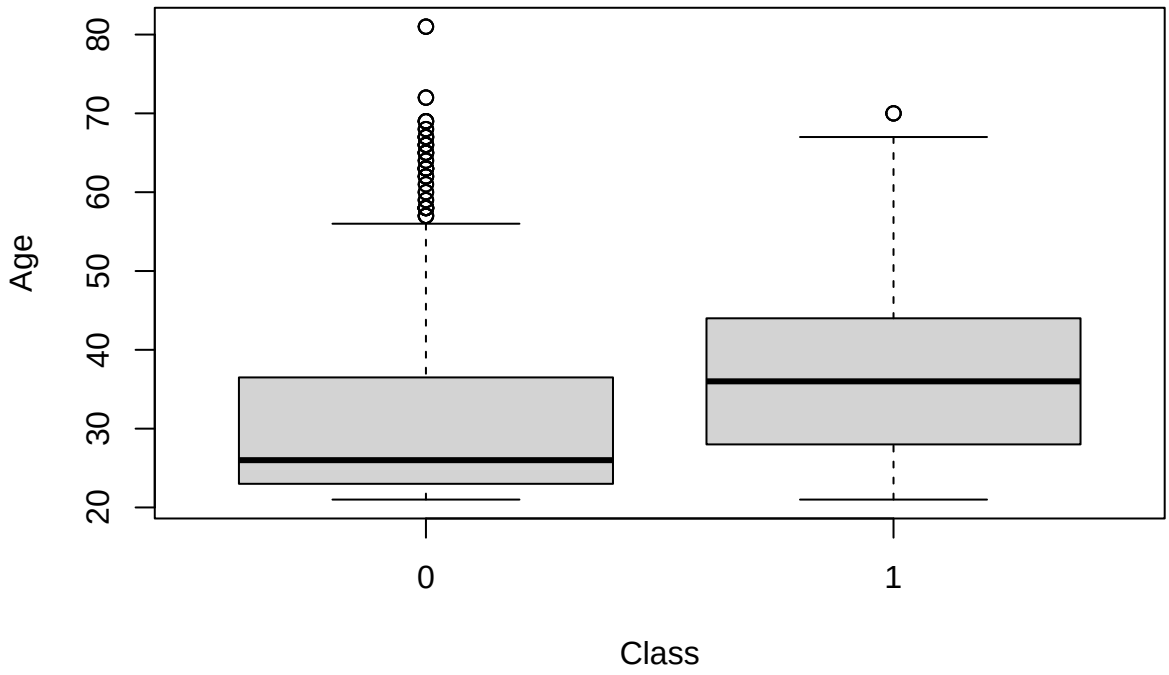

```
boxplot(BMI~Class)
```




```
boxplot(DiabPediFunc~Class)
```




```
boxplot(Age~Class)
```



Data Preparation

.....

Split the Data

```
library(caret)
set.seed(1234)
```

```
pd <- sample(2,nrow(data), replace = T, prob = c(0.7,0.3))
train_data <- data[pd==1,]
test_data <- data[pd==2,]

prop.table(table(data$Class))
```

```
##
##     0     1
## 0.658 0.342
```

```
prop.table(table(train_data$Class))
```

```
##
##       0       1
## 0.66363 0.33637
```

```
prop.table(table(test_data$Class))
```

```
##
##         0         1
## 0.6439791 0.3560209
```

```
# Binary variables needs to be converted into factor variables
train_data$Class <- as.factor(train_data$Class)
test_data$Class <- as.factor(test_data$Class)
```

```
dim(train_data)
```

```
## [1] 1427    9
```

```
dim(test_data)
```

```
## [1] 573   9
```

```
### Model Building - Logistic regression

logit_model1 = glm(Class ~ ., data = train_data,
                   family = binomial)

summary(logit_model1)
```

```
##
## Call:
## glm(formula = Class ~ ., family = binomial, data = train_data)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -3.1103  -0.7373  -0.4607   0.7812   2.8762
##
## Coefficients:
##                Estimate Std. Error z value Pr(>|z|)
## (Intercept)  -7.4602960  0.4894048 -15.244  < 2e-16 ***
## NoPreg        0.1360500  0.0235833   5.769 7.98e-09 ***
## PlaGluConc    0.0312222  0.0025248  12.366  < 2e-16 ***
## DiastolicBP  -0.0088899  0.0036527  -2.434   0.0149 *
## TSkinThick   -0.0027199  0.0049024  -0.555   0.5790
## Test          0.0002717  0.0006821   0.398   0.6904
## BMI           0.0706478  0.0098468   7.175 7.25e-13 ***
## DiabPediFunc  0.9241246  0.2183691   4.232 2.32e-05 ***
## Age           0.0065337  0.0068228   0.958   0.3382
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 1822.6  on 1426  degrees of freedom
## Residual deviance: 1388.3  on 1418  degrees of freedom
## AIC: 1406.3
##
## Number of Fisher Scoring iterations: 5
```

Check for multicollinearity

```
library(car)
vif(logit_model1)
```

```
##       NoPreg   PlaGluConc  DiastolicBP   TSkinThick         Test          BMI
##     1.455453     1.160117     1.142782     1.502630     1.428064     1.212398
## DiabPediFunc          Age
##     1.026722     1.524172
```

#After removing Insignificant Variables

```
logit_model2 <- glm(Class ~ NoPreg+PlaGluConc+BMI+DiabPediFunc+DiastolicBP, data = train_data,
                    family = binomial)

summary(logit_model2)
```

```
##
## Call:
## glm(formula = Class ~ NoPreg + PlaGluConc + BMI + DiabPediFunc +
##     DiastolicBP, family = binomial, data = train_data)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -3.0490  -0.7293  -0.4610   0.7792   2.8979
##
## Coefficients:
##               Estimate Std. Error z value Pr(>|z|)
## (Intercept)  -7.353383   0.467951 -15.714  < 2e-16 ***
## NoPreg        0.147939   0.020267   7.300 2.88e-13 ***
## PlaGluConc    0.031988   0.002363  13.536  < 2e-16 ***
## BMI           0.068442   0.009263   7.388 1.49e-13 ***
## DiabPediFunc  0.923499   0.216052   4.274 1.92e-05 ***
## DiastolicBP  -0.008698   0.003595  -2.420   0.0155 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 1822.6  on 1426  degrees of freedom
## Residual deviance: 1389.5  on 1421  degrees of freedom
## AIC: 1401.5
##
## Number of Fisher Scoring iterations: 5
```

#Verify the best AIC and model

```
step_model <- step(logit_model1)
```

```
## Start:  AIC=1406.25
## Class ~ NoPreg + PlaGluConc + DiastolicBP + TSkinThick + Test +
##     BMI + DiabPediFunc + Age
##
##                Df Deviance    AIC
## - Test          1   1388.4 1404.4
## - TSkinThick    1   1388.6 1404.6
## - Age           1   1389.2 1405.2
## <none>              1388.2 1406.2
## - DiastolicBP   1   1394.2 1410.2
## - DiabPediFunc  1   1406.5 1422.5
## - NoPreg        1   1422.9 1438.9
## - BMI           1   1444.9 1460.9
## - PlaGluConc    1   1574.3 1590.3
##
## Step:  AIC=1404.41
## Class ~ NoPreg + PlaGluConc + DiastolicBP + TSkinThick + BMI +
```

```
##     DiabPediFunc + Age
##
##                Df Deviance    AIC
## - TSkinThick    1   1388.6 1402.6
## - Age           1   1389.3 1403.3
## <none>              1388.4 1404.4
## - DiastolicBP   1   1394.4 1408.4
## - DiabPediFunc  1   1406.9 1420.9
## - NoPreg        1   1423.0 1437.0
## - BMI           1   1445.0 1459.0
## - PlaGluConc    1   1602.5 1616.5
##
## Step:  AIC=1402.59
## Class ~ NoPreg + PlaGluConc + DiastolicBP + BMI + DiabPediFunc +
##     Age
##
##                Df Deviance    AIC
## - Age           1   1389.5 1401.5
## <none>              1388.6 1402.6
## - DiastolicBP   1   1394.9 1406.9
## - DiabPediFunc  1   1406.9 1418.9
## - NoPreg        1   1423.1 1435.1
## - BMI           1   1449.9 1461.9
## - PlaGluConc    1   1602.6 1614.6
##
## Step:  AIC=1401.54
## Class ~ NoPreg + PlaGluConc + DiastolicBP + BMI + DiabPediFunc
##
##                Df Deviance    AIC
## <none>              1389.5 1401.5
## - DiastolicBP   1   1395.4 1405.4
## - DiabPediFunc  1   1408.1 1418.1
## - NoPreg        1   1445.0 1455.0
## - BMI           1   1450.0 1460.0
## - PlaGluConc    1   1618.7 1628.7
```

#Goodness of Fit

```
anova(logit_model1, test = "Chisq")
```

```
## Analysis of Deviance Table
##
## Model: binomial, link: logit
##
## Response: Class
##
## Terms added sequentially (first to last)
##
##
##              Df Deviance Resid. Df Resid. Dev  Pr(>Chi)
## NULL                          1426     1822.6
## NoPreg        1   67.372      1425     1755.2 2.249e-16 ***
## PlaGluConc    1  282.387      1424     1472.8 < 2.2e-16 ***
## DiastolicBP   1    0.202      1423     1472.6  0.653433
## TSkinThick    1    7.654      1422     1465.0  0.005666 **
```

```
## Test          1    0.207      1421     1464.7  0.649157
## BMI           1   57.023      1420     1407.7 4.307e-14 ***
## DiabPediFunc  1   18.549      1419     1389.2 1.656e-05 ***
## Age           1    0.912      1418     1388.2  0.339717
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Likelihood ratio test

```
library(lmtest)
lrtest(logit_model2)
```

```
## Likelihood ratio test
##
## Model 1: Class ~ NoPreg + PlaGluConc + BMI + DiabPediFunc + DiastolicBP
## Model 2: Class ~ 1
##   #Df  LogLik Df  Chisq Pr(>Chisq)
## 1   6 -694.77
## 2   1 -911.28 -5 433.02  < 2.2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Pseudo R-square

```
install.packages("pscl")
```

```
## Installing package into '/cloud/lib/x86_64-pc-linux-gnu-library/4.2'
## (as 'lib' is unspecified)
```

```
library(pscl)
```

```
## Classes and Methods for R developed in the
## Political Science Computational Laboratory
## Department of Political Science
## Stanford University
## Simon Jackman
## hurdle and zeroinfl functions by Achim Zeileis
```

```
pR2(logit_model1)
```

```
## fitting null model for pseudo-r2
```

```
##          llh      llhNull           G2     McFadden         r2ML         r2CU
## -694.1274479 -911.2797930  434.3046902    0.2382938    0.2623959    0.3638414
```

```
1-(257.4688426/341.9401912)
```

```
## [1] 0.2470354
```

Odds Ratio

```
exp(coef(logit_model1))
```

```
##  (Intercept)       NoPreg   PlaGluConc  DiastolicBP   TSkinThick         Test
```

```
## 0.0005754858 1.1457392228 1.0317147155 0.9911494782 0.9972837933 1.0002717299
##          BMI DiabPediFunc          Age
## 1.0732031718 2.5196616536 1.0065551409
```

Probability

```
exp(coef(logit_model1))/(1+exp(coef(logit_model1)))
```

```
##  (Intercept)       NoPreg   PlaGluConc  DiastolicBP   TSkinThick         Test
## 0.0005751548 0.5339601433 0.5078049136 0.4977775345 0.4993200249 0.5000679233
##          BMI DiabPediFunc          Age
## 0.5176546064 0.7158817811 0.5016334315
```

Accuracy | Base Line Model

```
nrow(train_data[train_data$Class == 0,])/nrow(train_data)
```

```
## [1] 0.66363
```

Performance metrics (train sample)

```
pred = predict(logit_model2, data=train_data, type="response")
y_pred_num = ifelse(pred>0.5,1,0)
y_pred = factor(y_pred_num, levels=c(0,1))
y_actual = train_data$Class
logi <-confusionMatrix(y_pred,train_data$Class,positive="1")
logi
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 846 217
##          1 101 263
##
##                Accuracy : 0.7772
##                  95% CI : (0.7547, 0.7985)
##     No Information Rate : 0.6636
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.4692
##
##  Mcnemar's Test P-Value : 1.127e-10
##
##             Sensitivity : 0.5479
##             Specificity : 0.8933
##          Pos Pred Value : 0.7225
##          Neg Pred Value : 0.7959
##              Prevalence : 0.3364
##          Detection Rate : 0.1843
##    Detection Prevalence : 0.2551
##       Balanced Accuracy : 0.7206
```

```
##
##        'Positive' Class : 1
##
```

Calibrating thresold levels to increase sensitivity

```
pred = predict(logit_model2, data=train_data, type="response")
y_pred_num = ifelse(pred>0.35,1,0)
y_pred = factor(y_pred_num, levels=c(0,1))
y_actual = train_data$Class
confusionMatrix(y_pred,y_actual,positive="1")
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 757 130
##          1 190 350
##
##                Accuracy : 0.7758
##                  95% CI : (0.7532, 0.7972)
##     No Information Rate : 0.6636
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.5127
##
##  Mcnemar's Test P-Value : 0.0009731
##
##             Sensitivity : 0.7292
##             Specificity : 0.7994
##          Pos Pred Value : 0.6481
##          Neg Pred Value : 0.8534
##              Prevalence : 0.3364
##          Detection Rate : 0.2453
##    Detection Prevalence : 0.3784
##       Balanced Accuracy : 0.7643
##
##        'Positive' Class : 1
##
```

Performance metrics (test sample)

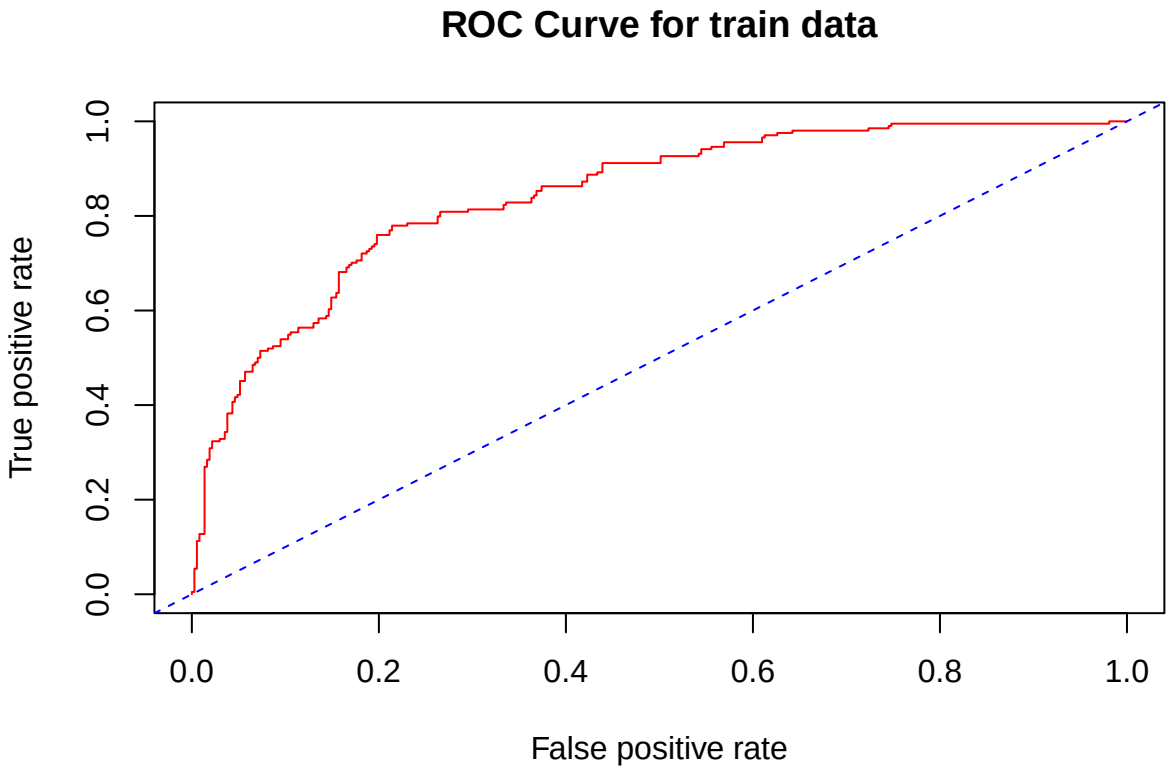
```
pred = predict(logit_model2, newdata=test_data, type="response")
y_pred_num = ifelse(pred>0.35,1,0)
y_pred = factor(y_pred_num, levels=c(0,1))
y_actual = test_data$Class
confusionMatrix(y_pred,y_actual,positive="1")
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
```

```
##          0 299  56
##          1  70 148
##
##                Accuracy : 0.7801
##                  95% CI : (0.7439, 0.8134)
##     No Information Rate : 0.644
##     P-Value [Acc > NIR] : 1.118e-12
##
##                   Kappa : 0.5277
##
##  Mcnemar's Test P-Value : 0.2468
##
##             Sensitivity : 0.7255
##             Specificity : 0.8103
##          Pos Pred Value : 0.6789
##          Neg Pred Value : 0.8423
##              Prevalence : 0.3560
##          Detection Rate : 0.2583
##    Detection Prevalence : 0.3805
##       Balanced Accuracy : 0.7679
##
##        'Positive' Class : 1
##
```

ROC plot

```
library(ROCR)
train_roc <- prediction(pred, test_data$Class)
#dev.off()
plot(performance(train_roc, "tpr", "fpr"),
     col = "red", main = "ROC Curve for train data")
abline(0, 1, lty = 8, col = "blue")
```



#

AUC

```
train_auc = performance(train_roc, "auc")
train_area = as.numeric(slot(train_auc, "y.values"))
train_area
```

```
## [1] 0.8436686
```

KS

```
ks_train <- performance(train_roc, "tpr", "fpr")
train_ks <- max(attr(ks_train, "y.values")[[1]] - (attr(ks_train, "x.values")[[1]]))
train_ks
```

```
## [1] 0.5653196
```

Gini

```
train_gini = (2 * train_area) - 1
train_gini
```

```
## [1] 0.6873373
```

KNN

Normalize variables

```r
scale = preProcess(train_data, method = "range")

train.norm.data = predict(scale, train_data)
test.norm.data = predict(scale, test_data)

knn_fit = train(Class ~., data = train.norm.data, method = "knn",
                trControl = trainControl(method = "cv", number = 3),
                tuneLength = 10)

knn_fit
```

```
## k-Nearest Neighbors
##
## 1427 samples
##    8 predictor
##    2 classes: '0', '1'
##
## No pre-processing
## Resampling: Cross-Validated (3 fold)
## Summary of sample sizes: 951, 951, 952
## Resampling results across tuning parameters:
##
##   k   Accuracy   Kappa
##    5  0.7673434  0.4579496
##    7  0.7743403  0.4753495
##    9  0.7617264  0.4436247
##   11  0.7610305  0.4410737
##   13  0.7610217  0.4389614
##   15  0.7547192  0.4187191
##   17  0.7519121  0.4106028
##   19  0.7575217  0.4239402
##   21  0.7533201  0.4134007
##   23  0.7519151  0.4068717
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was k = 7.
```

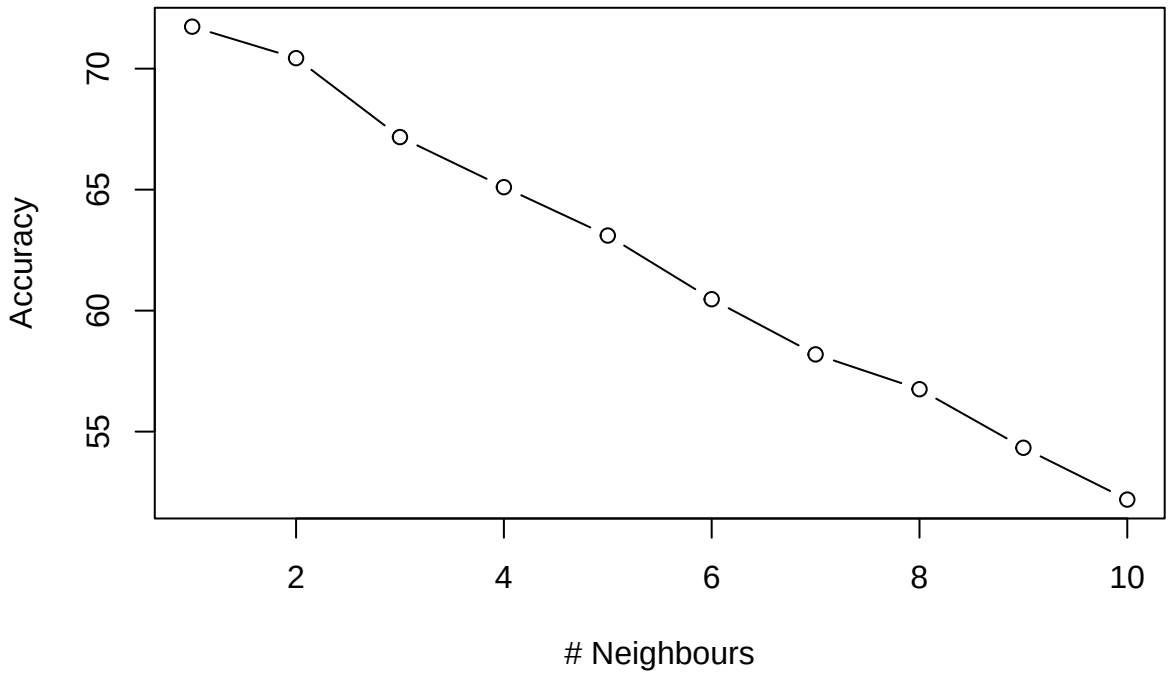
```r
knn_fit$bestTune$k
```

```
## [1] 7
```

#knn_fit*results*Accuracy

#Plotting the k vs accuracy

```r
plot((knn_fit$results$Accuracy)*100 - knn_fit$results$k, type = 'b', xlab = "# Neighbours", ylab = "Accu
```



Performance metrics (train sample)

```
pred = predict(knn_fit, data = train.norm.data[-9], type = "raw")
knnc <- confusionMatrix(pred,train.norm.data$Class,positive="1")
knnc
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 857 150
##          1  90 330
##
##                Accuracy : 0.8318
##                  95% CI : (0.8114, 0.8509)
##     No Information Rate : 0.6636
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.6113
##
##  Mcnemar's Test P-Value : 0.0001398
##
##             Sensitivity : 0.6875
##             Specificity : 0.9050
##          Pos Pred Value : 0.7857
##          Neg Pred Value : 0.8510
##              Prevalence : 0.3364
##          Detection Rate : 0.2313
##    Detection Prevalence : 0.2943
##       Balanced Accuracy : 0.7962
##
##        'Positive' Class : 1
##
```

Performance metrics (test sample)

```
pred = predict(knn_fit, newdata = test.norm.data[-9], type = "raw")
confusionMatrix(pred,test.norm.data$Class,positive="1")
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 331  70
##          1  38 134
##
##                Accuracy : 0.8115
##                  95% CI : (0.777, 0.8427)
##     No Information Rate : 0.644
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.574
##
##  Mcnemar's Test P-Value : 0.002855
##
##             Sensitivity : 0.6569
##             Specificity : 0.8970
##          Pos Pred Value : 0.7791
##          Neg Pred Value : 0.8254
##              Prevalence : 0.3560
##          Detection Rate : 0.2339
##    Detection Prevalence : 0.3002
##       Balanced Accuracy : 0.7769
##
##        'Positive' Class : 1
##
```

Model Building - NB

```
library(e1071)
NB = naiveBayes(x=train.norm.data[-c(9)], y=train.norm.data$Class)
```

Performance metrics (train sample)

```
pred_NB_train = predict(NB, newdata = train.norm.data[-9])
nbc <-confusionMatrix(pred_NB_train, train.norm.data$Class,positive="1")
nbc
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 812 206
##          1 135 274
##
##                Accuracy : 0.761
##                  95% CI : (0.738, 0.783)
```

```
##     No Information Rate : 0.6636
##     P-Value [Acc > NIR] : 7.037e-16
##
##                   Kappa : 0.4445
##
##  Mcnemar's Test P-Value : 0.0001502
##
##             Sensitivity : 0.5708
##             Specificity : 0.8574
##          Pos Pred Value : 0.6699
##          Neg Pred Value : 0.7976
##              Prevalence : 0.3364
##          Detection Rate : 0.1920
##    Detection Prevalence : 0.2866
##       Balanced Accuracy : 0.7141
##
##        'Positive' Class : 1
##
```

Performance metrics (test sample)

```
pred_NB_test = predict(NB, newdata = test.norm.data[-9])
confusionMatrix(pred_NB_test,test.norm.data$Class,positive="1")
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 315  93
##          1  54 111
##
##                Accuracy : 0.7435
##                  95% CI : (0.7056, 0.7788)
##     No Information Rate : 0.644
##     P-Value [Acc > NIR] : 2.185e-07
##
##                   Kappa : 0.4155
##
##  Mcnemar's Test P-Value : 0.001723
##
##             Sensitivity : 0.5441
##             Specificity : 0.8537
##          Pos Pred Value : 0.6727
##          Neg Pred Value : 0.7721
##              Prevalence : 0.3560
##          Detection Rate : 0.1937
##    Detection Prevalence : 0.2880
##       Balanced Accuracy : 0.6989
##
##        'Positive' Class : 1
##
```

#Compare Accuracy

```
knnc$overall['Accuracy']
```

```
## Accuracy
## 0.831815
```

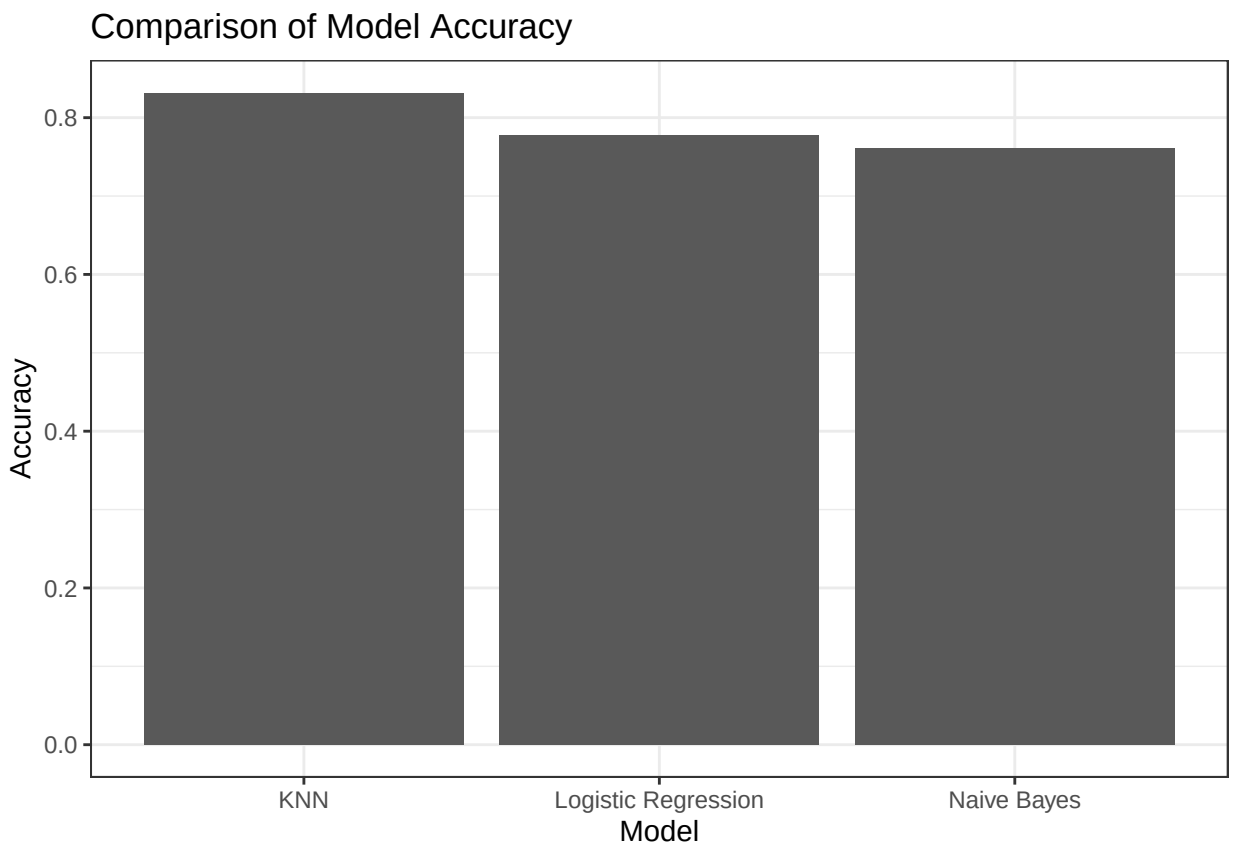
```
logi$overall['Accuracy']
```

```
##  Accuracy
## 0.7771549
```

```
nbc$overall['Accuracy']
```

```
##  Accuracy
## 0.7610371
```

```
Model=c("Logistic Regression","KNN","Naive Bayes")
Accuracy<- c(logi$overall['Accuracy'], knnc$overall['Accuracy'], nbc$overall['Accuracy'])
accuracy <- data.frame(Model, Accuracy)
Accuracy<- c(logi, knnc, nbc)

ggplot(accuracy,aes(x=Model,y=Accuracy)) + geom_bar(stat='identity') + theme_bw() + ggtitle('Comparison
```




```
names(data)
```

```
## [1] "NoPreg"       "PlaGluConc"   "DiastolicBP"  "TSkinThick"   "Test"
## [6] "BMI"          "DiabPediFunc" "Age"          "Class"
```

As the result shows KNN is the best model for this kind of data analysis.