Risk of Diabetes- Statistical analysis of comorbidity and leading physiological and demographic factors

Samriddhi Soni

2023-12-15

Loading relevant libraries

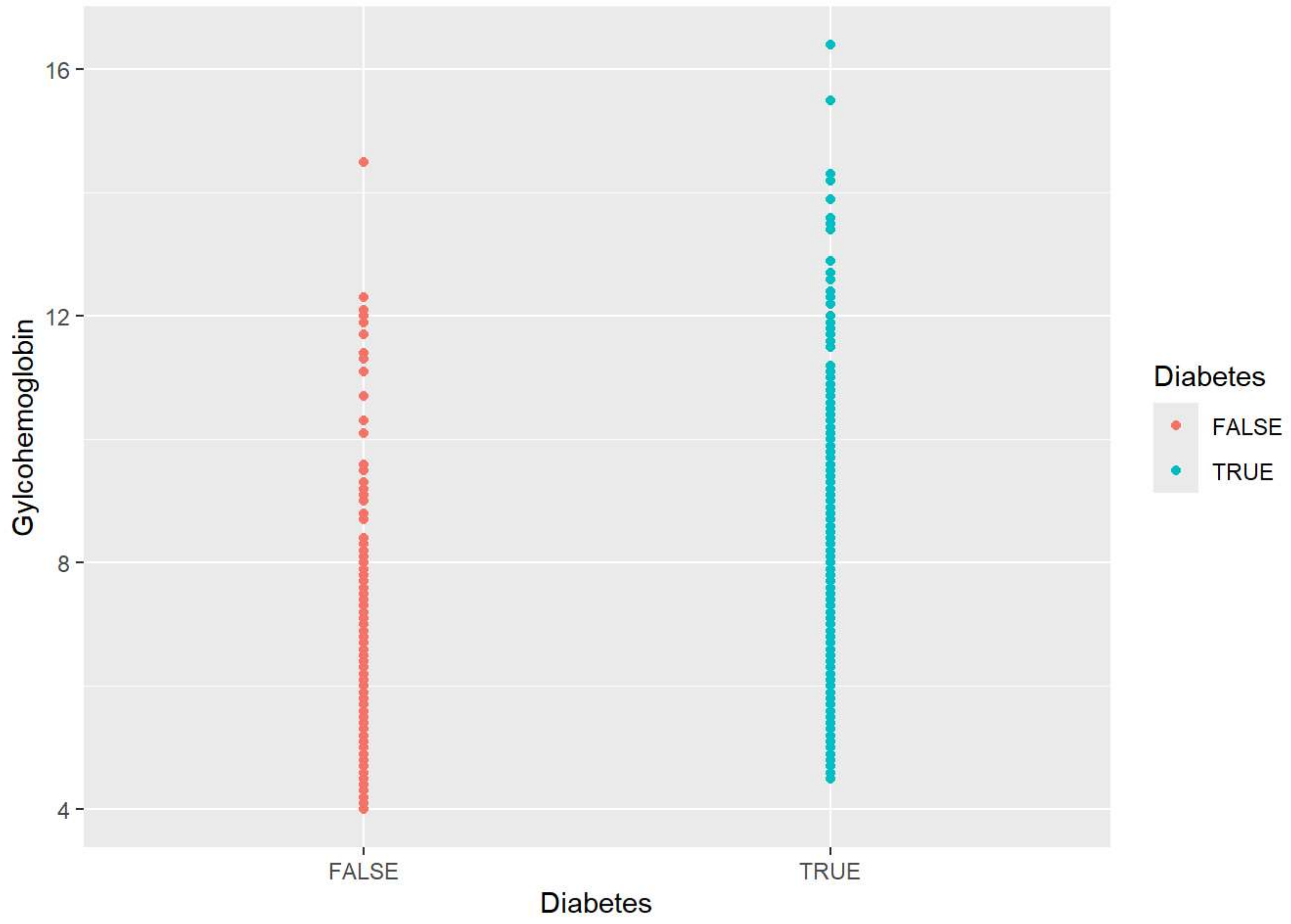
```
library("ggplot2")
library("dplyr")
library("caret")
library("rpart")
library("rpart.plot")
library("randomForest")
library("modelr")
library("data.table")
library("randomForest")
library("corrplot")
```

Loading data

```
data <- read.csv(file="C:/Users/ss6557/Desktop/Semester 3/ORLA-6541-Data Science for organizatio
n and leadership/Final Paper/data (1).csv/data_new.csv")
data <- na.omit(data)
data <- data[,-1]

# Factor the variables
data$Gender <- as.factor(data$Gender)
data$Ethnicity <- as.factor(data$Ethnicity)
```

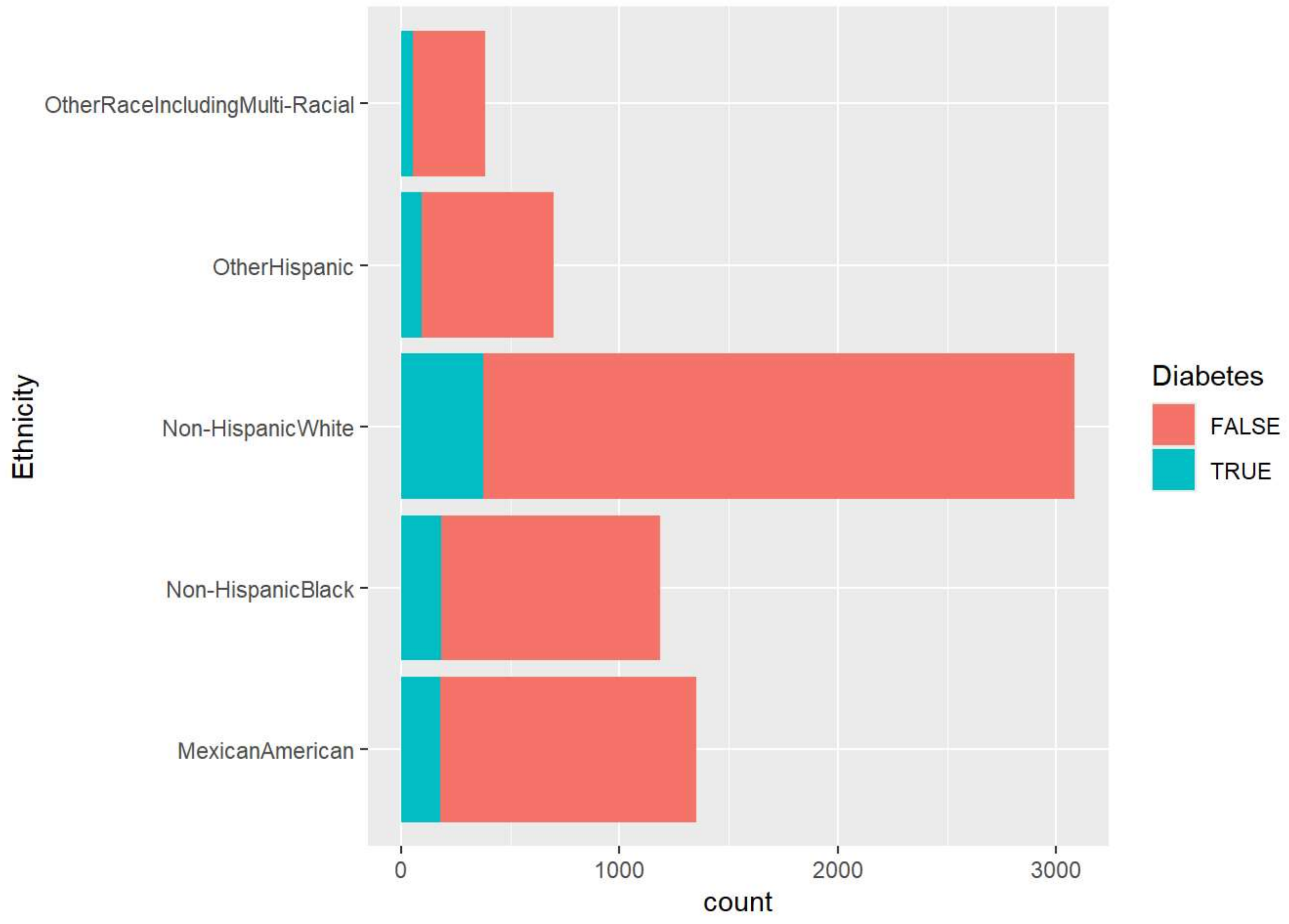
Visualizations

```
par(mfrow=c(2,2))
#1
ggplot(
  data = data,
  mapping = aes(x = Diabetes, y = Glycohemoglobin)
) +
  geom_point(aes(color = Diabetes)) +
  labs(
    x = "Diabetes", y = "Gylcohemoglobin",
    color = "Diabetes", shape = "Diabetes"
  )
```



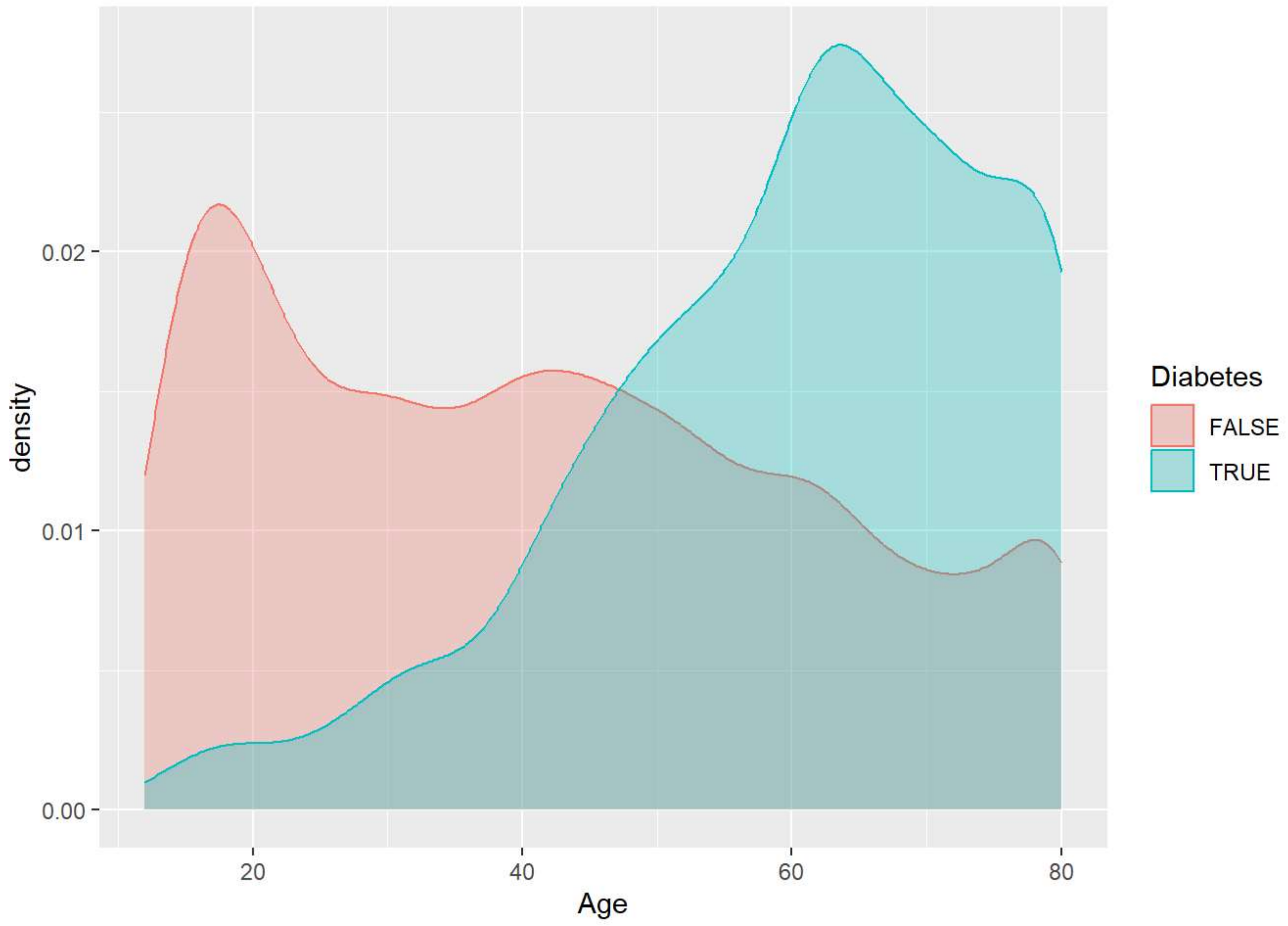
Ethnicity vs Diabetes

```
ggplot(data, aes(x=Ethnicity, fill=Diabetes)) + geom_bar() + coord_flip()
```



Age vs diabetes

```
ggplot(data,
       aes(x = Age,
           color = Diabetes,
           fill = Diabetes))+
  geom_density(alpha = 0.3,
               na.rm = TRUE)
```



#HCA Heatmaps

```
library(hopach)
```

```
## Loading required package: cluster
```

```
## Loading required package: Biobase
```

```
## Loading required package: BiocGenerics
```

```
##
## Attaching package: 'BiocGenerics'
```

```
## The following object is masked from 'package:randomForest':
##
##     combine
```

```
## The following objects are masked from 'package:dplyr':
##
##     combine, intersect, setdiff, union
## The following objects are masked from 'package:stats':
##
##     IQR, mad, sd, var, xtabs
```

```
## The following objects are masked from 'package:base':
##
##     anyDuplicated, aperm, append, as.data.frame, basename, cbind,
##     colnames, dirname, do.call, duplicated, eval, evalq, Filter, Find,
##     get, grep, grepl, intersect, is.unsorted, lapply, Map, mapply,
##     match, mget, order, paste, pmax, pmax.int, pmin, pmin.int,
##     Position, rank, rbind, Reduce, rownames, sapply, setdiff, sort,
##     table, tapply, union, unique, unsplit, which.max, which.min
```

```
## Welcome to Bioconductor
##
##     Vignettes contain introductory material; view with
##     'browseVignettes()'. To cite Bioconductor, see
##     'citation("Biobase")', and for packages 'citation("pkgname")'.
```

```
##
## Attaching package: 'hopach'
```

```
## The following object is masked from 'package:rpart':
##
##     prune
```

```
library(ComplexHeatmap)
```

```
## Loading required package: grid
## ========================================
## ComplexHeatmap version 2.14.0
## Bioconductor page: http://bioconductor.org/packages/ComplexHeatmap/
## Github page: https://github.com/jokergoo/ComplexHeatmap
## Documentation: http://jokergoo.github.io/ComplexHeatmap-reference
##
## If you use it in published research, please cite either one:
## - Gu, Z. Complex Heatmap Visualization. iMeta 2022.
## - Gu, Z. Complex heatmaps reveal patterns and correlations in multidimensional
##     genomic data. Bioinformatics 2016.
##
##
## The new InteractiveComplexHeatmap package can directly export static
## complex heatmaps into an interactive Shiny app with zero effort. Have a try!
##
## This message can be suppressed by:
##   suppressPackageStartupMessages(library(ComplexHeatmap))
## ========================================
```

```
library(circlize)
```

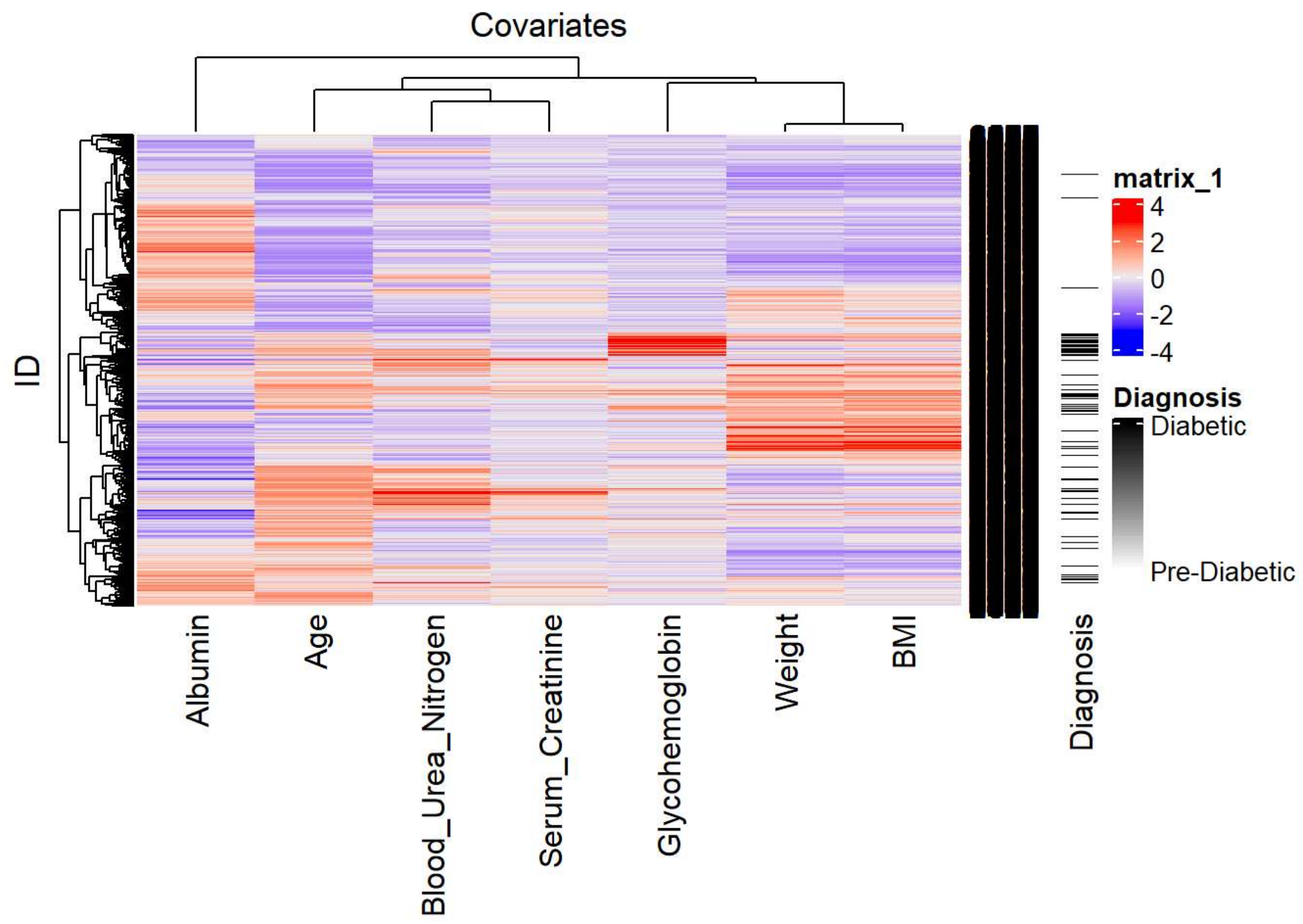
```
## Warning: package 'circlize' was built under R version 4.2.3
```

```
## ========================================
## circlize version 0.4.15
## CRAN page: https://cran.r-project.org/package=circlize
## Github page: https://github.com/jokergoo/circlize
## Documentation: https://jokergoo.github.io/circlize_book/book/
##
## If you use it in published research, please cite:
## Gu, Z. circlize implements and enhances circular visualization
##   in R. Bioinformatics 2014.
##
## This message can be suppressed by:
##   suppressPackageStartupMessages(library(circlize))
## ========================================
```

```
suppressPackageStartupMessages(library(circlize))

data_num <- data[,c(2,6,7,8,9,10,11)]
data_scale <- scale(data_num)
```

Clustering

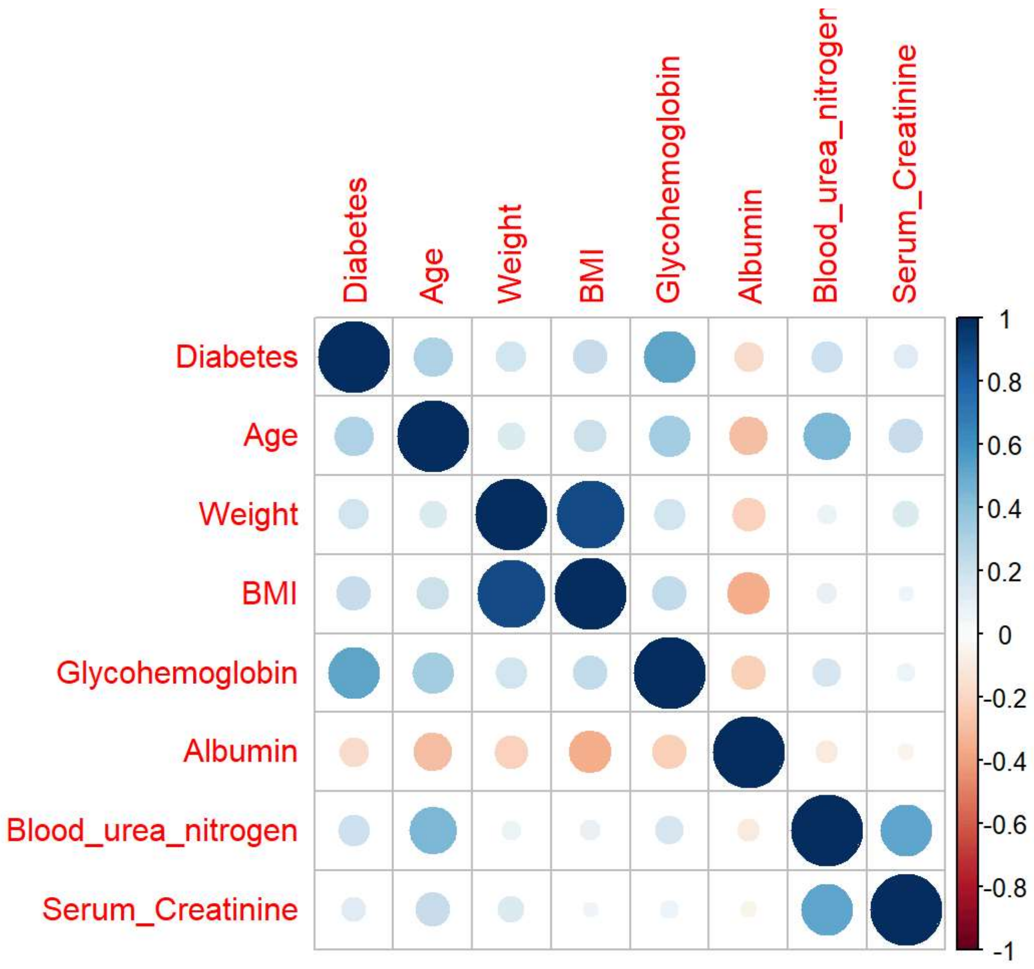
```
uncenter.dist <- function(m) {
as.dist(as.matrix(distancematrix(m, d="cosangle")))
}
row.clus<-hclust(uncenter.dist(data_scale), method = "ave")
col.clus<-hclust(uncenter.dist(t(data_scale)), method = "ave")

suppressMessages(ht_main <- Heatmap(data_scale, cluster_rows=row.clus, cluster_columns=col.clus,
row_title = "ID", column_title = "Covariates"))

ht_diabetes<-Heatmap(data[,5], name = "Diagnosis",
col = colorRamp2(c(FALSE,TRUE),c("white", "black")),
heatmap_legend_param = list(at = c(0,1),
labels = c("Pre-Diabetic", "Diabetic")),
width = unit(0.5,"cm"))

draw(ht_main+ht_diabetes, auto_adjust = FALSE)
```



```
data_num <- cbind(data$Diabetes, data_num)
colnames(data_num) <- c("Diabetes","Age","Weight","BMI","Glycohemoglobin","Albumin","Blood_urea_
nitrogen","Serum_Creatinine")
```

Correlation plot

```
corr <- cor(data_num)
corrplot(corr)
```



Splitting the data into training and testing set

```
data$Diabetes <- as.factor(data$Diabetes)
data_num$Diabetes <- as.factor(data_num$Diabetes)
```

```
set.seed(4321)

# Split the data into training and testing set
index <- createDataPartition(data_num$Diabetes, p=0.7, list = FALSE)
train_dat <- data_num[index, ]
test_dat <- data_num[-index, ]
```

Logistic Regression

```
library(pROC)
```

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

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

```
## The following object is masked from 'package:BiocGenerics':
##
##     var
```

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

```
logmodel <- glm(Diabetes~., data = train_dat, family = "binomial")
summary(logmodel)
```

```
##
## Call:
## glm(formula = Diabetes ~ ., family = "binomial", data = train_dat)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -3.8596  -0.4346  -0.2563  -0.1594   3.1200
##
## Coefficients:
##                      Estimate Std. Error z value Pr(>|z|)
## (Intercept)        -13.474087   1.049139 -12.843  < 2e-16 ***
## Age                  0.027105   0.003614   7.499 6.43e-14 ***
## Weight              -0.003376   0.005173  -0.653    0.514
## BMI                  0.066655   0.016835   3.959 7.51e-05 ***
## Glycohemoglobin      1.436196   0.078965  18.188  < 2e-16 ***
## Albumin             -0.097049   0.189226  -0.513    0.608
## Blood_urea_nitrogen   0.019547   0.009752   2.004    0.045 *
## Serum_Creatinine     0.104091   0.146009   0.713    0.476
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 3664.7  on 4694  degrees of freedom
## Residual deviance: 2403.1  on 4687  degrees of freedom
## AIC: 2419.1
##
## Number of Fisher Scoring iterations: 6
```

```
predictions <- predict(logmodel, test_dat, type = "response")

roc_curve <- roc(test_dat$Diabetes, predictions)
```

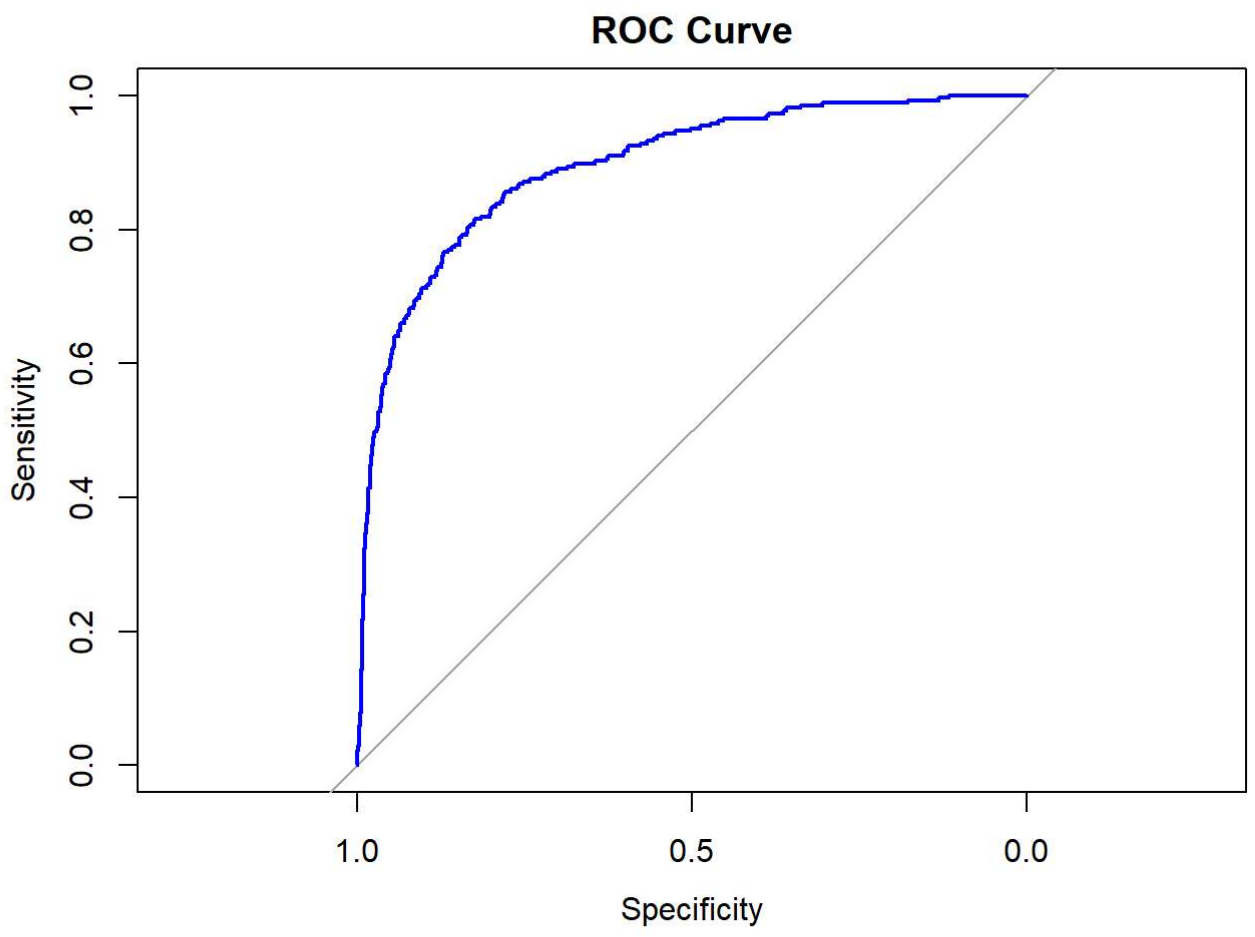
```
## Setting levels: control = FALSE, case = TRUE
```

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

```
auc_value <- auc(roc_curve)

# Print AUC value
cat("AUC:", auc_value, "\n")
```

```
## AUC: 0.8945385
```

```
# Plot the ROC AUC curve
plot(roc_curve, main = "ROC Curve", col = "blue", lwd = 2, xlim=c(1,0))
```



Confusion matrix and prediction accuracy

```
predictions <- predict(logmodel, test_dat) %>% as.data.frame()
colnames(predictions) <- c("Diabetes")
predictions$Diabetes <- exp(predictions$Diabetes)
head(predictions)
```

```
##        Diabetes
## 1    0.03073168
## 7    0.98813375
## 8    0.01294696
## 10 408.81551064
## 16   0.16055259
## 23   0.02953699
```

```
# Create a confusion matrix
predictions <- mutate(predictions,
  Diabetes = as.factor(ifelse(predictions > 0.5, TRUE, FALSE))
  ) %>%
  select(Diabetes)

# Confusion matrix
confusionMatrix(predictions$Diabetes, test_dat$Diabetes)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction FALSE TRUE
##      FALSE  1672  111
##      TRUE     74  154
##
##                Accuracy : 0.908
##                  95% CI : (0.8945, 0.9203)
##     No Information Rate : 0.8682
##     P-Value [Acc > NIR] : 2.006e-08
##
##                   Kappa : 0.5727
##
##  Mcnemar's Test P-Value : 0.008126
##
##             Sensitivity : 0.9576
##             Specificity : 0.5811
##          Pos Pred Value : 0.9377
##          Neg Pred Value : 0.6754
##              Prevalence : 0.8682
##          Detection Rate : 0.8314
##    Detection Prevalence : 0.8866
##       Balanced Accuracy : 0.7694
##
##        'Positive' Class : FALSE
##
```

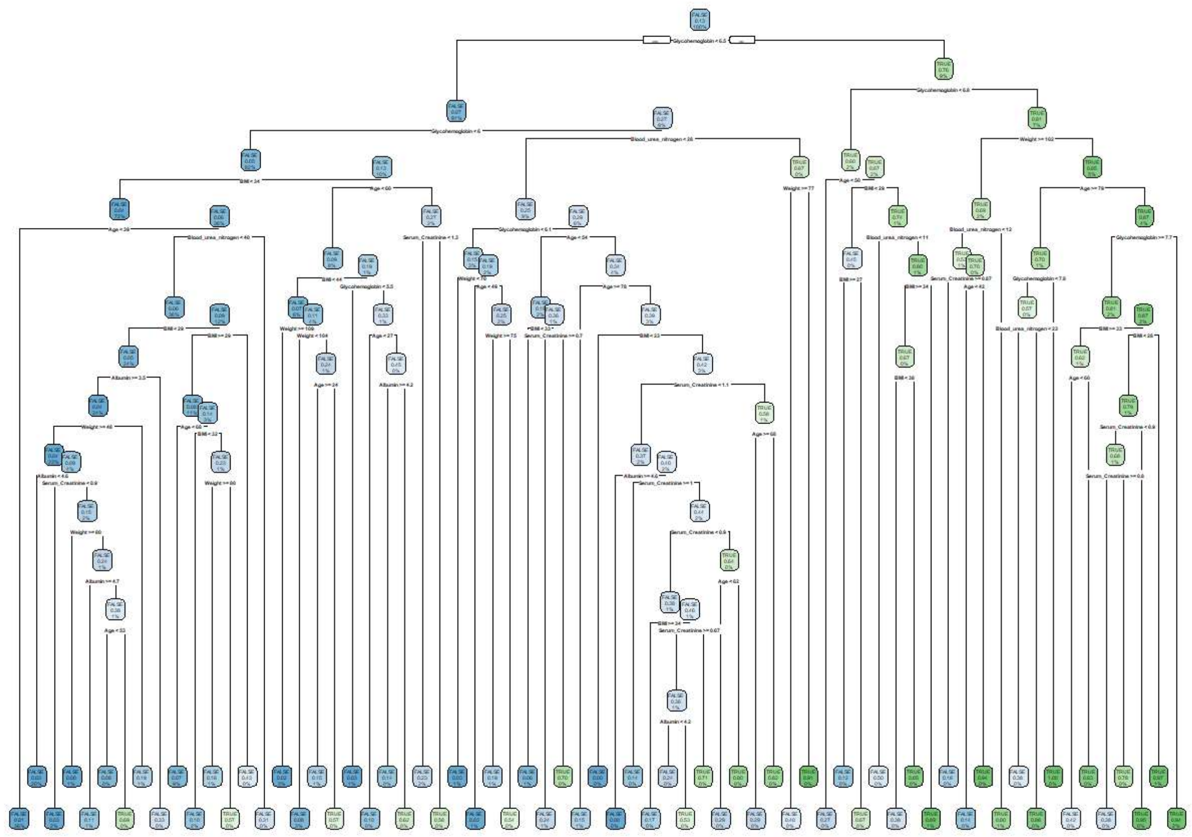
Decision tree and Random Forest

```
library("caret")
library("rpart")
library("rpart.plot")
library("randomForest")
library("modelr")
library("data.table")
library("randomForest")
```

Fitting decision tree without pruning

```
dt_fit1 <- rpart(formula= train_dat$Diabetes~.,
                 data=train_dat,
                 method="class",
                 control= rpart.control(minsplit=20,
                                        cp=0,
                                        xval=0),
                 parms = list(split="gini"))
rpart.plot(dt_fit1)
```

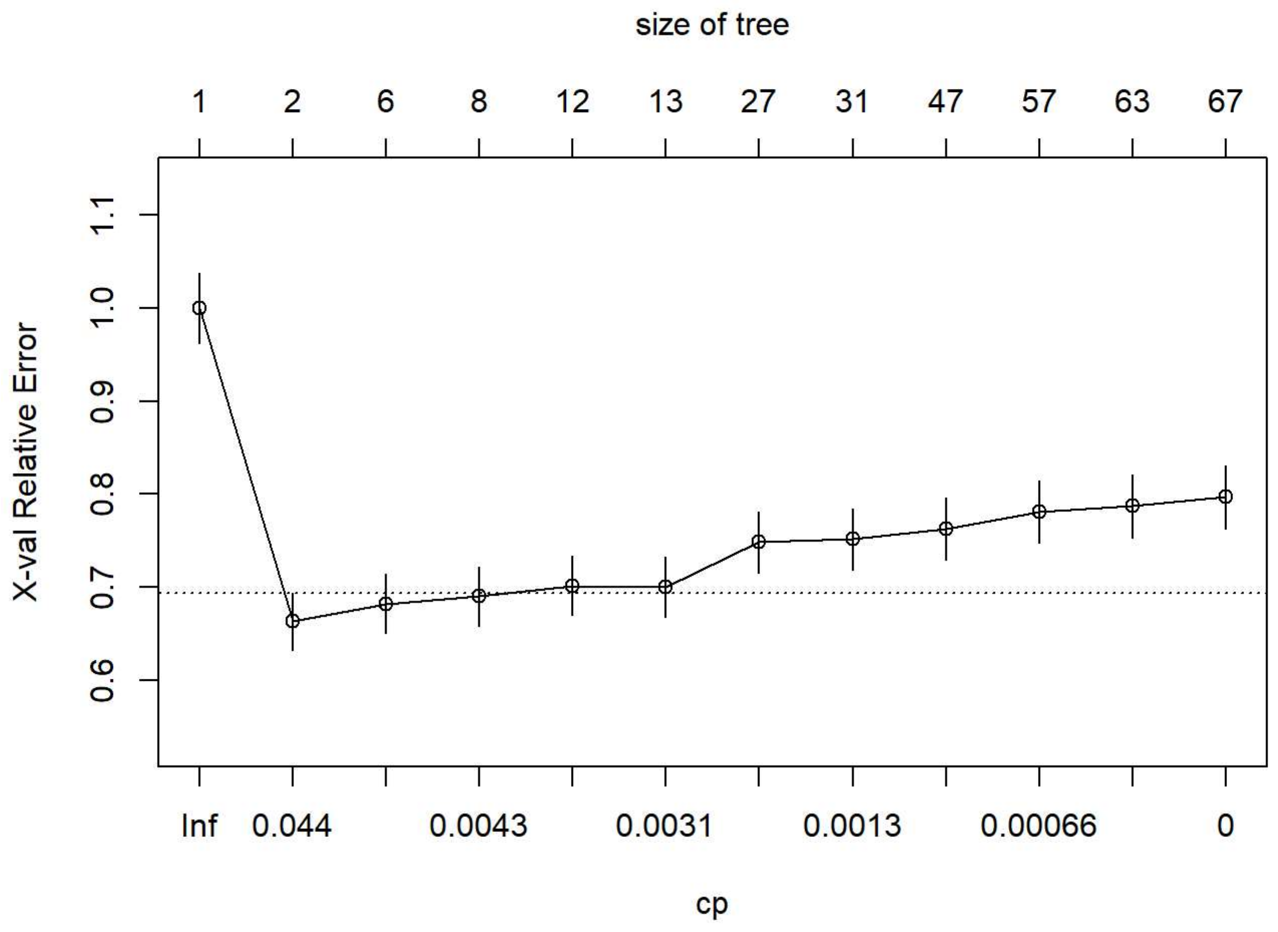
```
## Warning: labs do not fit even at cex 0.15, there may be some overplotting
```



Overfitting- requires pruning

Cross validation for optimal value of cp

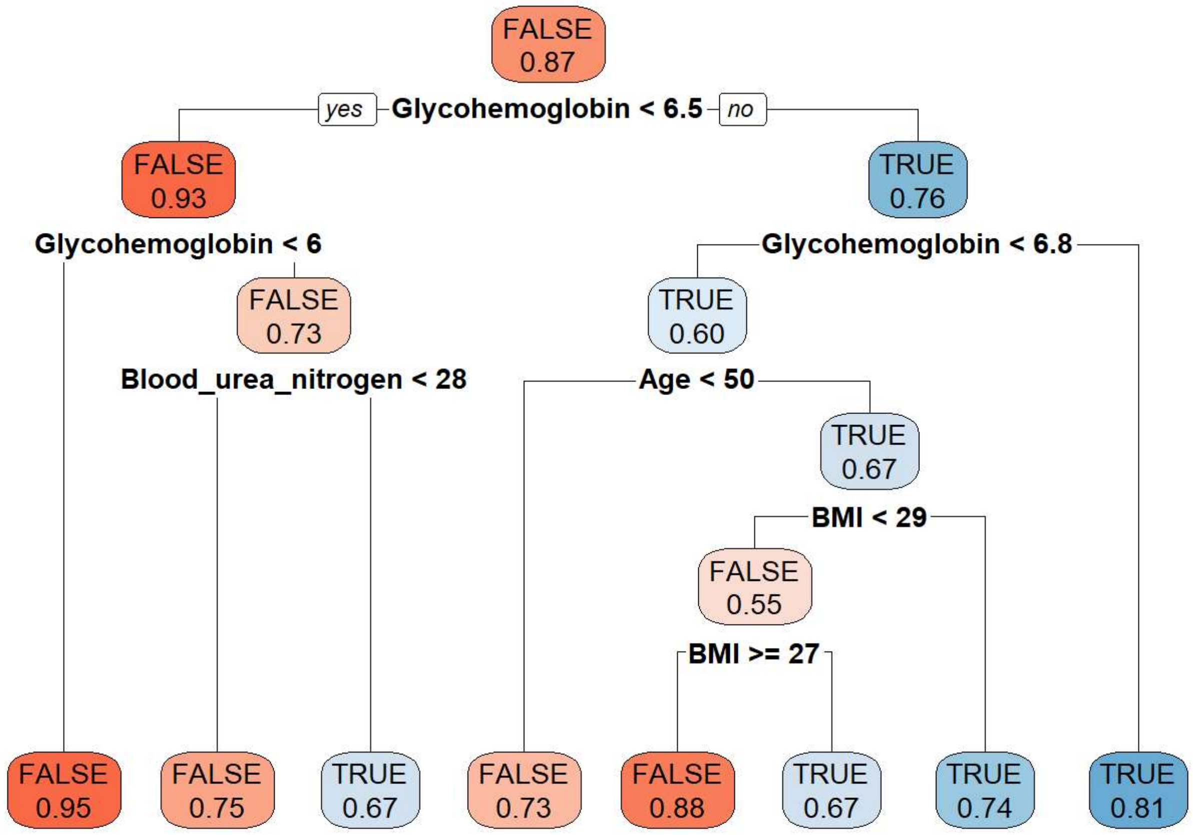
```
dt_fit4 <- rpart(formula = train_dat$Diabetes~.,
                 data = train_dat,
                 method = "class",
                 control = rpart.control(minsplit = 20,
                                         cp = 0,
                                         xval = 10),
                parms = list(split = "gini"))
plotcp(dt_fit4)
```



0.005- too simplified model 0.0031- overfitted(too complex model) 0.0043- optimal

Pruning the tree

```
dt_fit2 <- rpart(formula= train_dat$Diabetes~.,
                 data=train_dat,
                 method="class",
                 control= rpart.control(minsplit=20,
                                        cp=0.0043,
                                        xval=0),
                 parms = list(split="gini"))
rpart.plot(dt_fit2,
           extra=8,
           box.palette="RdBu")
```



Printing the output of decision tree

```
printcp(dt_fit2)
```

```
##
## Classification tree:
## rpart(formula = train_dat$Diabetes ~ ., data = train_dat, method = "class",
##     parms = list(split = "gini"), control = rpart.control(minsplit = 20,
##         cp = 0.0043, xval = 0))
##
## Variables actually used in tree construction:
## [1] Age                 Blood_urea_nitrogen BMI
## [4] Glycohemoglobin
##
## Root node error: 620/4695 = 0.13206
##
## n= 4695
##
##          CP nsplit rel error
## 1 0.3500000      0   1.00000
## 2 0.0056452      1   0.65000
## 3 0.0048387      5   0.62742
## 4 0.0043000      7   0.61774
```

```
varImp(dt_fit2)
```

```
##                        Overall
## Age                 122.010891
## Albumin               3.370485
## Blood_urea_nitrogen  51.753646
## BMI                  69.288645
## Glycohemoglobin     409.271225
## Serum_Creatinine     19.469247
## Weight               32.995068
```

Checking the classificaion accuracy using the test data

```
dt_pred <- predict(dt_fit2, test_dat) %>% as.data.frame()
head(dt_pred)
```

```
##        FALSE       TRUE
## 1  0.9518104 0.04818963
## 7  0.1931464 0.80685358
## 8  0.9518104 0.04818963
## 10 0.1931464 0.80685358
## 16 0.9518104 0.04818963
## 23 0.9518104 0.04818963
```

```
dt_pred <- mutate(dt_pred,
  Diabetes = as.factor(ifelse(dt_pred$`FALSE` >= 0.5, FALSE, TRUE))
  ) %>%
  select(Diabetes)

# Confusion matrix
confusionMatrix(dt_pred$Diabetes, test_dat$Diabetes)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction FALSE TRUE
##      FALSE  1700  123
##      TRUE     46  142
##
##                Accuracy : 0.916
##                  95% CI : (0.903, 0.9277)
##     No Information Rate : 0.8682
##     P-Value [Acc > NIR] : 1.258e-11
##
##                   Kappa : 0.5811
##
##  Mcnemar's Test P-Value : 5.031e-09
##
##             Sensitivity : 0.9737
##             Specificity : 0.5358
##          Pos Pred Value : 0.9325
##          Neg Pred Value : 0.7553
##              Prevalence : 0.8682
##          Detection Rate : 0.8454
##    Detection Prevalence : 0.9065
##       Balanced Accuracy : 0.7548
##
##        'Positive' Class : FALSE
##
```

The output shows that the overall accuracy is around 91.6%, sensitivity is 97.37 % and specificity is 53.58%