563project

Yan Liu

11/9/2021

Read data and save a clean dataset after removing missing

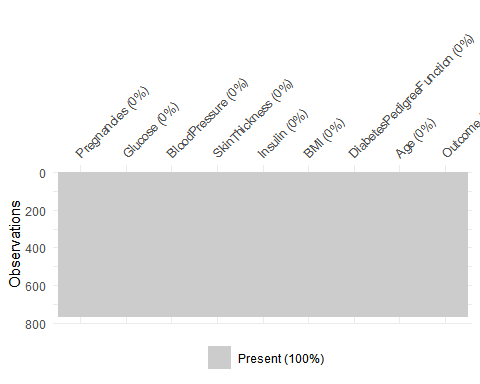
library(tidyverse)  
  
data\_whole<-read\_csv("diabetes\_good.csv")  
names(data\_whole)

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

#visualize missing data  
#data\_whole[,2:8][data\_whole[,2:8]==0]<- NA  
  
sum(complete.cases(data\_whole))

## [1] 768

#visualize missing data  
library(naniar)  
vis\_miss(data\_whole)



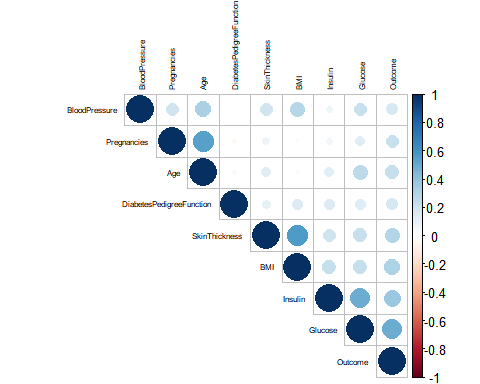
#give NAs to 0s which are impossible  
#data\_clean<-na.omit(data\_whole)  
  
  
#quantile(data\_clean$Insulin, prob=c(.40,.5,.65))  
#mean(data\_clean$Insulin)  
#sd(data\_clean$Insulin)

Split data Split the data into a training (70% of the data) and test set (30% of the data)

library(caret)  
library(rsample)  
set.seed(14)  
index <- initial\_split(data\_whole,  
 prop = 0.7)  
train <- training(index)  
test <- testing(index)

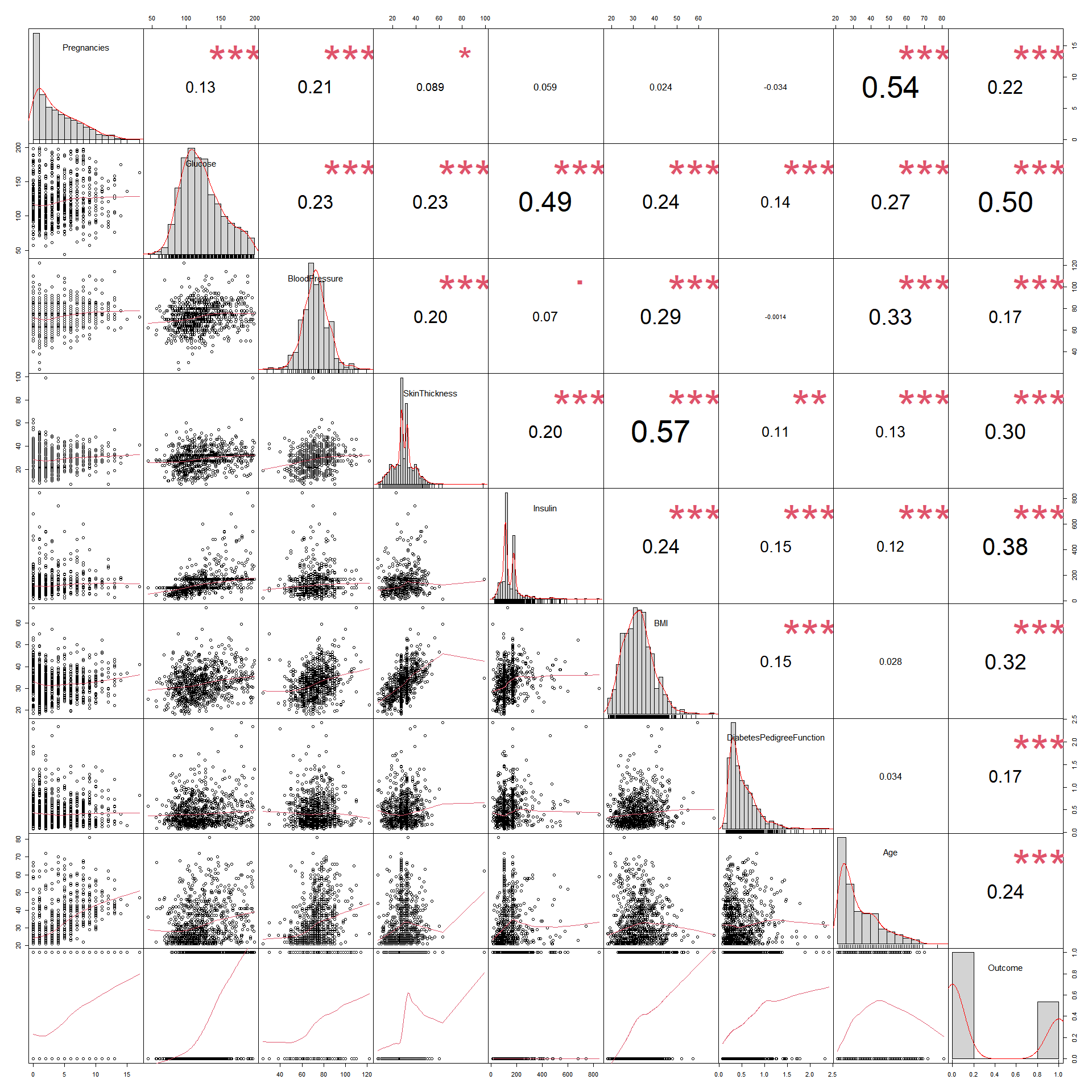
Data Plots

library(corrplot)  
res <- cor(data\_whole)  
  
#Plot the correlation matrix values by cluster  
corrplot(res, type = "upper", order = "hclust",  
 tl.col = "black", tl.cex = 0.5)



#table(data\_clean$Outcome)  
#mean(data\_clean$Outcome)#NIR(66.8%), outcome has 262 "0" (66.8%) and 130 "1" (33.2%)  
  
#table(data\_clean$Outcome)/nrow(data\_clean)

library(PerformanceAnalytics)  
chart.Correlation(data\_whole, histogram = TRUE, method = "pearson")



change Outcome to factor

train <- train %>%   
 mutate(Outcome = as.factor(Outcome))  
test <- test %>%   
 mutate(Outcome = as.factor(Outcome))  
  
  
table(test$Outcome)

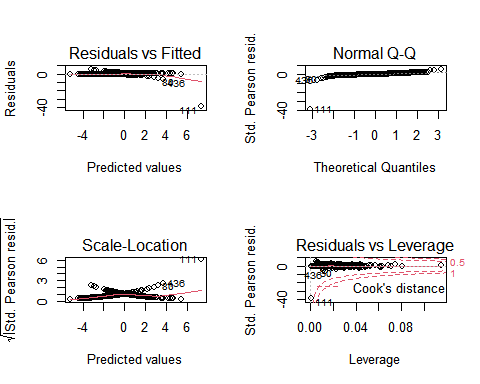
##   
## 0 1   
## 154 77

logistical regression. No clear collinearity according to VIF.

glm<- glm(Outcome~.,train,family="binomial")  
summary(glm)

##   
## Call:  
## glm(formula = Outcome ~ ., family = "binomial", data = train)  
##   
## Deviance Residuals:   
## Min 1Q Median 3Q Max   
## -3.8165 -0.6900 -0.3424 0.7090 2.5421   
##   
## Coefficients:  
## Estimate Std. Error z value Pr(>|z|)   
## (Intercept) -9.495376 1.017191 -9.335 < 2e-16 \*\*\*  
## Pregnancies 0.095314 0.038342 2.486 0.012923 \*   
## Glucose 0.028779 0.004648 6.192 5.94e-10 \*\*\*  
## BloodPressure -0.009863 0.010427 -0.946 0.344202   
## SkinThickness 0.034557 0.017575 1.966 0.049273 \*   
## Insulin 0.006228 0.001876 3.319 0.000903 \*\*\*  
## BMI 0.076619 0.022426 3.417 0.000634 \*\*\*  
## DiabetesPedigreeFunction 0.856056 0.369599 2.316 0.020549 \*   
## Age 0.020333 0.011601 1.753 0.079651 .   
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## (Dispersion parameter for binomial family taken to be 1)  
##   
## Null deviance: 699.06 on 536 degrees of freedom  
## Residual deviance: 477.07 on 528 degrees of freedom  
## AIC: 495.07  
##   
## Number of Fisher Scoring iterations: 5

par(mfrow=c(2,2))  
plot(glm)



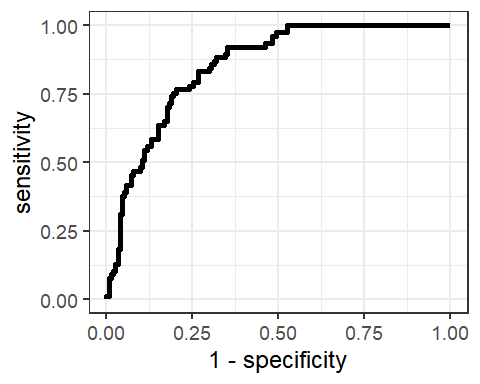
car::vif(glm)# a VIF value larger than 5 or 10 indicates a problematic amount of multicollinearity, No multicollinearity

## Pregnancies Glucose BloodPressure   
## 1.339699 1.183797 1.201077   
## SkinThickness Insulin BMI   
## 1.377819 1.124433 1.483154   
## DiabetesPedigreeFunction Age   
## 1.023364 1.499789

# predicted probability of glm  
glm\_prob <- predict(glm, test,type = "response")  
# predicted outcome of glm  
glm\_pred <- rep(0, length(glm\_prob))  
glm\_pred[glm\_prob > 0.5] <- 1  
accuracy\_glm<-mean(glm\_pred == test$Outcome)  
  
glm\_conf<-confusionMatrix(data =factor(glm\_pred) , reference = test$Outcome) #?No Information Rate : 0.7542   
glm\_conf$overall[[1]]

## [1] 0.7748918

#ROC and its AUC  
library(pROC)  
library(ggplot2)  
roccurve\_glm <- roc(response = test$Outcome, predictor = glm\_prob)  
ggroc(roccurve\_glm, legacy.axes = TRUE, lwd=2) +theme\_bw(base\_size = 18)



auc\_glm<-auc(roccurve\_glm)

backward selection

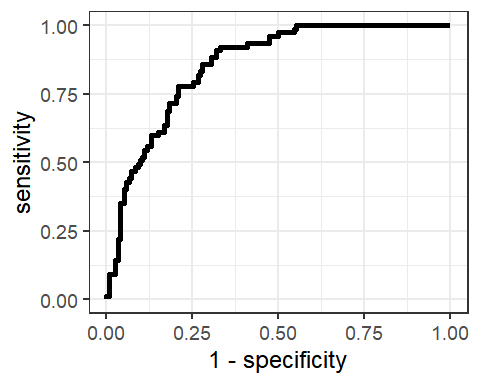
back<-step(glm) #backward selection based on AIC 4 variables

## Start: AIC=495.07  
## Outcome ~ Pregnancies + Glucose + BloodPressure + SkinThickness +   
## Insulin + BMI + DiabetesPedigreeFunction + Age  
##   
## Df Deviance AIC  
## - BloodPressure 1 477.97 493.97  
## <none> 477.07 495.07  
## - Age 1 480.12 496.12  
## - SkinThickness 1 481.08 497.08  
## - DiabetesPedigreeFunction 1 482.58 498.58  
## - Pregnancies 1 483.39 499.39  
## - BMI 1 489.19 505.19  
## - Insulin 1 490.65 506.65  
## - Glucose 1 520.34 536.34  
##   
## Step: AIC=493.97  
## Outcome ~ Pregnancies + Glucose + SkinThickness + Insulin + BMI +   
## DiabetesPedigreeFunction + Age  
##   
## Df Deviance AIC  
## <none> 477.97 493.97  
## - Age 1 480.37 494.37  
## - SkinThickness 1 481.95 495.95  
## - DiabetesPedigreeFunction 1 483.64 497.64  
## - Pregnancies 1 484.22 498.22  
## - BMI 1 489.23 503.23  
## - Insulin 1 491.99 505.99  
## - Glucose 1 520.37 534.37

back

##   
## Call: glm(formula = Outcome ~ Pregnancies + Glucose + SkinThickness +   
## Insulin + BMI + DiabetesPedigreeFunction + Age, family = "binomial",   
## data = train)  
##   
## Coefficients:  
## (Intercept) Pregnancies Glucose   
## -9.908517 0.094587 0.028236   
## SkinThickness Insulin BMI   
## 0.034470 0.006291 0.072310   
## DiabetesPedigreeFunction Age   
## 0.862199 0.017357   
##   
## Degrees of Freedom: 536 Total (i.e. Null); 529 Residual  
## Null Deviance: 699.1   
## Residual Deviance: 478 AIC: 494

# predicted probability of back  
back\_prob <- predict(back, test,type = "response")  
# ROC and its auc  
roccurve\_back <- roc(response = test$Outcome, predictor = back\_prob)  
ggroc(roccurve\_back, legacy.axes = TRUE, lwd=2) +theme\_bw(base\_size = 18)



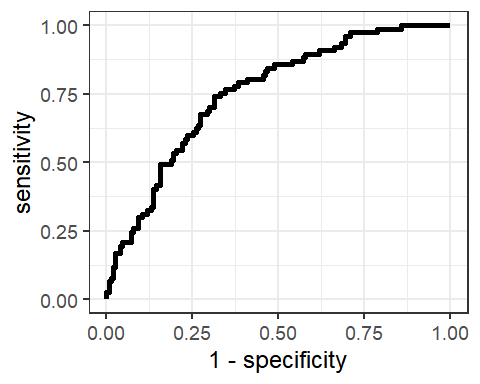
auc\_back<-auc(roccurve\_back)  
  
# predicted outcome of glm  
back\_pred <- rep(0, length(glm\_prob))  
back\_pred[back\_prob > 0.5] <- 1  
accuracy\_back<-mean(back\_pred == test$Outcome)

best selection

train\_back<-as.data.frame(model.matrix(Outcome~.-1, data = train))  
library(bestglm)  
best<-bestglm(train\_back, IC="AIC")  
best$BestModel

##   
## Call:  
## lm(formula = y ~ ., data = data.frame(Xy[, c(bestset[-1], FALSE),   
## drop = FALSE], y = y))  
##   
## Coefficients:  
## (Intercept) Pregnancies Glucose   
## 5.27146 1.59887 0.08097   
## BloodPressure SkinThickness BMI   
## 0.21302 0.11945 -0.24204   
## DiabetesPedigreeFunction   
## 1.93320

# predicted probability of glm  
best\_prob <- predict(best$BestModel, test,type = "response")  
# ROC and its auc  
roccurve\_best <- roc(response = test$Outcome, predictor = best\_prob)  
ggroc(roccurve\_best, legacy.axes = TRUE, lwd=2) +theme\_bw(base\_size = 18)



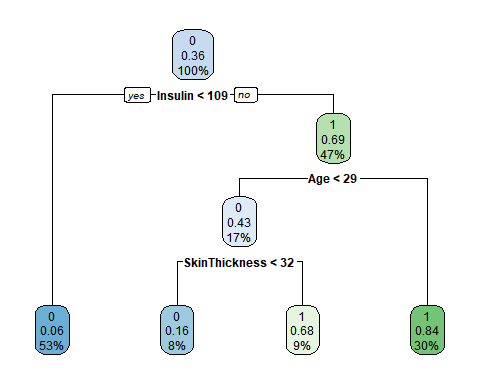
auc\_best<-auc(roccurve\_best)  
  
# predicted outcome of glm  
best\_pred <- rep(0, length(glm\_prob))  
best\_pred[glm\_prob > 0.5] <- 1  
accuracy\_best<-mean(best\_pred == test$Outcome)

classification tree

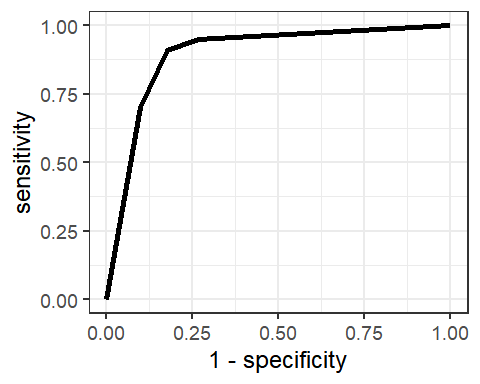
library(rpart)  
library(rpart.plot)  
tree\_class <- rpart(  
 Outcome ~ .,  
 data = train,  
 method = 'class',  
 parms = list(split = "information"),  
 control = rpart.control(  
 xval = 10,  
 minbucket = 2,  
 cp = 0  
 )  
)  
printcp(tree\_class)

##   
## Classification tree:  
## rpart(formula = Outcome ~ ., data = train, method = "class",   
## parms = list(split = "information"), control = rpart.control(xval = 10,   
## minbucket = 2, cp = 0))  
##   
## Variables actually used in tree construction:  
## [1] Age BloodPressure BMI   
## [4] DiabetesPedigreeFunction Glucose Insulin   
## [7] Pregnancies SkinThickness   
##   
## Root node error: 191/537 = 0.35568  
##   
## n= 537   
##   
## CP nsplit rel error xerror xstd  
## 1 0.5130890 0 1.000000 1.00000 0.058081  
## 2 0.0759162 1 0.486911 0.49738 0.046297  
## 3 0.0183246 3 0.335079 0.40838 0.042750  
## 4 0.0130890 5 0.298429 0.39791 0.042290  
## 5 0.0104712 8 0.256545 0.38220 0.041581  
## 6 0.0078534 14 0.193717 0.40838 0.042750  
## 7 0.0069808 18 0.162304 0.41885 0.043200  
## 8 0.0052356 22 0.125654 0.41885 0.043200  
## 9 0.0000000 32 0.073298 0.44503 0.044285

cp <- tree\_class$cptable  
tree\_class\_final <- prune(tree\_class, cp = cp[3,1])#used minimum  
rpart.plot(tree\_class\_final)



# test error rate using min cp  
tree\_pred <- predict(tree\_class\_final, newdata=test, type = "class")  
  
accuracy\_tree<-mean(tree\_pred == test$Outcome)  
  
tree\_prob <- predict(tree\_class\_final, newdata=test, type = "prob")  
  
roccurve\_tree <- roc(response = test$Outcome, predictor = tree\_prob[,2])  
ggroc(roccurve\_tree, legacy.axes = TRUE, lwd=2) +theme\_bw(base\_size = 18)



auc\_tree<-auc(roccurve\_tree)  
  
klaR::errormatrix(true = test$Outcome, predicted = tree\_pred, relative = TRUE)

## predicted  
## true 0 1 -SUM-  
## 0 0.82467532 0.1753247 0.17532468  
## 1 0.09090909 0.9090909 0.09090909  
## -SUM- 0.20588235 0.7941176 0.14718615

confusionMatrix(tree\_pred,test$Outcome)

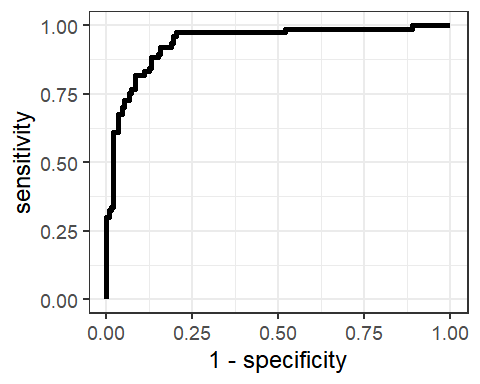
## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction 0 1  
## 0 127 7  
## 1 27 70  
##   
## Accuracy : 0.8528   
## 95% CI : (0.8004, 0.8959)  
## No Information Rate : 0.6667   
## P-Value [Acc > NIR] : 1.121e-10   
##   
## Kappa : 0.689   
##   
## Mcnemar's Test P-Value : 0.00112   
##   
## Sensitivity : 0.8247   
## Specificity : 0.9091   
## Pos Pred Value : 0.9478   
## Neg Pred Value : 0.7216   
## Prevalence : 0.6667   
## Detection Rate : 0.5498   
## Detection Prevalence : 0.5801   
## Balanced Accuracy : 0.8669   
##   
## 'Positive' Class : 0   
##

boost tree

cvcontrol <- trainControl(method = "repeatedcv",  
 number = 5,  
 allowParallel = TRUE)  
grid <- expand.grid(  
 n.trees = c(10, 50, 100, 500, 1000),  
 interaction.depth = c(1:3),  
 shrinkage = c(0.01, 0.05, 0.1),  
 n.minobsinnode = c(5,10)  
)  
capture <- capture.output(  
 train.gbm <- train(  
 Outcome ~ .,  
 data = train,  
 method = "gbm",  
 trControl = cvcontrol,  
 tuneGrid = grid  
 )  
)  
train.gbm

## Stochastic Gradient Boosting   
##   
## 537 samples  
## 8 predictor  
## 2 classes: '0', '1'   
##   
## No pre-processing  
## Resampling: Cross-Validated (5 fold, repeated 1 times)   
## Summary of sample sizes: 430, 429, 430, 429, 430   
## Resampling results across tuning parameters:  
##   
## shrinkage interaction.depth n.minobsinnode n.trees Accuracy Kappa   
## 0.01 1 5 10 0.6443233 0.0000000  
## 0.01 1 5 50 0.7692627 0.4493806  
## 0.01 1 5 100 0.8209934 0.6192513  
## 0.01 1 5 500 0.8583766 0.6911406  
## 0.01 1 5 1000 0.8732953 0.7219138  
## 0.01 1 10 10 0.6443233 0.0000000  
## 0.01 1 10 50 0.7785566 0.4756793  
## 0.01 1 10 100 0.8210280 0.6163915  
## 0.01 1 10 500 0.8583593 0.6904444  
## 0.01 1 10 1000 0.8788854 0.7335139  
## 0.01 2 5 10 0.6443233 0.0000000  
## 0.01 2 5 50 0.8213049 0.5706524  
## 0.01 2 5 100 0.8565247 0.6819015  
## 0.01 2 5 500 0.8751990 0.7241918  
## 0.01 2 5 1000 0.8788854 0.7322176  
## 0.01 2 10 10 0.6443233 0.0000000  
## 0.01 2 10 50 0.8268432 0.5811533  
## 0.01 2 10 100 0.8620803 0.6950264  
## 0.01 2 10 500 0.8770509 0.7278168  
## 0.01 2 10 1000 0.8807546 0.7370543  
## 0.01 3 5 10 0.6443233 0.0000000  
## 0.01 3 5 50 0.8436310 0.6311651  
## 0.01 3 5 100 0.8732780 0.7178002  
## 0.01 3 5 500 0.8732953 0.7216234  
## 0.01 3 5 1000 0.8770509 0.7299073  
## 0.01 3 10 10 0.6443233 0.0000000  
## 0.01 3 10 50 0.8380582 0.6191023  
## 0.01 3 10 100 0.8695396 0.7108583  
## 0.01 3 10 500 0.8826237 0.7410743  
## 0.01 3 10 1000 0.8825891 0.7419558  
## 0.05 1 5 10 0.7598996 0.4222229  
## 0.05 1 5 50 0.8396850 0.6517610  
## 0.05 1 5 100 0.8527864 0.6768314  
## 0.05 1 5 500 0.8826237 0.7415827  
## 0.05 1 5 1000 0.8732433 0.7205705  
## 0.05 1 10 10 0.7524230 0.4088410  
## 0.05 1 10 50 0.8489789 0.6751226  
## 0.05 1 10 100 0.8676878 0.7104613  
## 0.05 1 10 500 0.8825718 0.7390772  
## 0.05 1 10 1000 0.8732433 0.7191177  
## 0.05 2 5 10 0.8175493 0.5676304  
## 0.05 2 5 50 0.8620630 0.6982812  
## 0.05 2 5 100 0.8695916 0.7135584  
## 0.05 2 5 500 0.8788681 0.7327314  
## 0.05 2 5 1000 0.8825545 0.7413259  
## 0.05 2 10 10 0.8249394 0.5828757  
## 0.05 2 10 50 0.8676532 0.7080596  
## 0.05 2 10 100 0.8788854 0.7325072  
## 0.05 2 10 500 0.8825891 0.7394120  
## 0.05 2 10 1000 0.8825372 0.7380485  
## 0.05 3 5 10 0.8305815 0.6008666  
## 0.05 3 5 50 0.8714434 0.7169213  
## 0.05 3 5 100 0.8882139 0.7547606  
## 0.05 3 5 500 0.8918830 0.7622799  
## 0.05 3 5 1000 0.8881447 0.7533389  
## 0.05 3 10 10 0.8491173 0.6498361  
## 0.05 3 10 50 0.8732433 0.7226735  
## 0.05 3 10 100 0.8714088 0.7168108  
## 0.05 3 10 500 0.8825545 0.7392425  
## 0.05 3 10 1000 0.8844237 0.7435697  
## 0.10 1 5 10 0.8284528 0.6354458  
## 0.10 1 5 50 0.8564728 0.6888445  
## 0.10 1 5 100 0.8751471 0.7271783  
## 0.10 1 5 500 0.8732433 0.7185586  
## 0.10 1 5 1000 0.8695396 0.7106841  
## 0.10 1 10 10 0.8247664 0.6234044  
## 0.10 1 10 50 0.8639322 0.7011262  
## 0.10 1 10 100 0.8695223 0.7120395  
## 0.10 1 10 500 0.8769990 0.7270552  
## 0.10 1 10 1000 0.8639668 0.6989414  
## 0.10 2 5 10 0.8527345 0.6720024  
## 0.10 2 5 50 0.8733126 0.7204575  
## 0.10 2 5 100 0.8677051 0.7066889  
## 0.10 2 5 500 0.8825718 0.7410575  
## 0.10 2 5 1000 0.8825372 0.7404001  
## 0.10 2 10 10 0.8490481 0.6682773  
## 0.10 2 10 50 0.8770163 0.7297977  
## 0.10 2 10 100 0.8770163 0.7304063  
## 0.10 2 10 500 0.8807027 0.7368754  
## 0.10 2 10 1000 0.8788508 0.7324899  
## 0.10 3 5 10 0.8489962 0.6649509  
## 0.10 3 5 50 0.8658532 0.7062443  
## 0.10 3 5 100 0.8751644 0.7261204  
## 0.10 3 5 500 0.8789027 0.7329976  
## 0.10 3 5 1000 0.8807373 0.7370536  
## 0.10 3 10 10 0.8602631 0.6898576  
## 0.10 3 10 50 0.8788681 0.7325567  
## 0.10 3 10 100 0.8807373 0.7359299  
## 0.10 3 10 500 0.8788335 0.7322700  
## 0.10 3 10 1000 0.8750952 0.7236954  
##   
## Accuracy was used to select the optimal model using the largest value.  
## The final values used for the model were n.trees = 500, interaction.depth =  
## 3, shrinkage = 0.05 and n.minobsinnode = 5.

boost\_pred <- predict(train.gbm, newdata=test, type = "raw")  
accuracy\_boost<-mean(boost\_pred == test$Outcome)  
  
  
boost\_prob <- predict(train.gbm, newdata=test, type = "prob")  
  
roccurve\_tree <- roc(response = test$Outcome, predictor = boost\_prob[,2])  
ggroc(roccurve\_tree, legacy.axes = TRUE, lwd=2) +theme\_bw(base\_size = 18)



auc\_boost<-auc(roccurve\_tree)  
  
klaR::errormatrix(true = test$Outcome, predicted = boost\_pred, relative = TRUE)

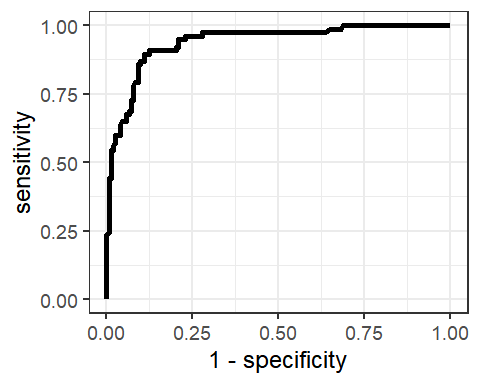
## predicted  
## true 0 1 -SUM-  
## 0 0.9155844 0.08441558 0.08441558  
## 1 0.1818182 0.81818182 0.18181818  
## -SUM- 0.5185185 0.48148148 0.11688312

random forests model

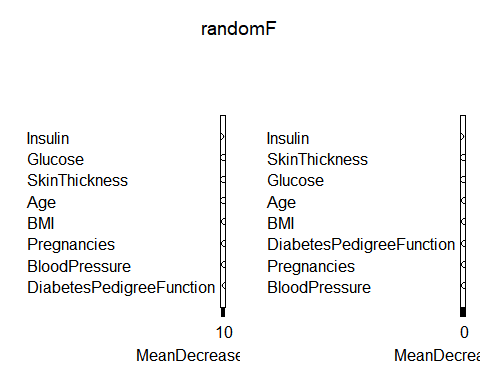
library(randomForest)  
randomF<-randomForest(Outcome~., data=train, mtry=3, importance=TRUE)  
randomF

##   
## Call:  
## randomForest(formula = Outcome ~ ., data = train, mtry = 3, importance = TRUE)   
## Type of random forest: classification  
## Number of trees: 500  
## No. of variables tried at each split: 3  
##   
## OOB estimate of error rate: 11.92%  
## Confusion matrix:  
## 0 1 class.error  
## 0 316 30 0.0867052  
## 1 34 157 0.1780105

randomF\_pred <- predict(randomF, newdata=test, type = "response")  
accuracy\_randomF<-mean(randomF\_pred == test$Outcome)  
  
  
randomF\_prob <- predict(randomF, newdata=test, type = "prob")  
  
roccurve\_randomF <- roc(response = test$Outcome, predictor = randomF\_prob[,2])  
ggroc(roccurve\_randomF, legacy.axes = TRUE, lwd=2) +theme\_bw(base\_size = 18)



auc\_randomF<-auc(roccurve\_randomF)  
  
varImpPlot(randomF)#Glucose is the most important one.

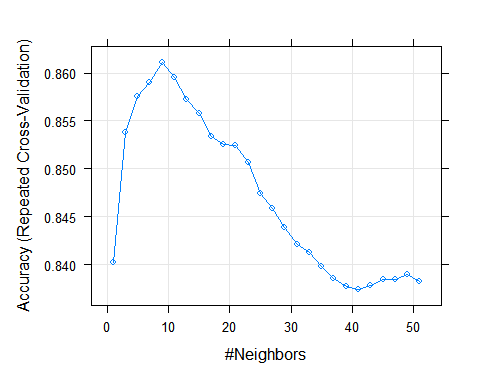


klaR::errormatrix(true = test$Outcome, predicted = randomF\_pred, relative = TRUE)

## predicted  
## true 0 1 -SUM-  
## 0 0.9025974 0.0974026 0.0974026  
## 1 0.1428571 0.8571429 0.1428571  
## -SUM- 0.4230769 0.5769231 0.1125541

Knn

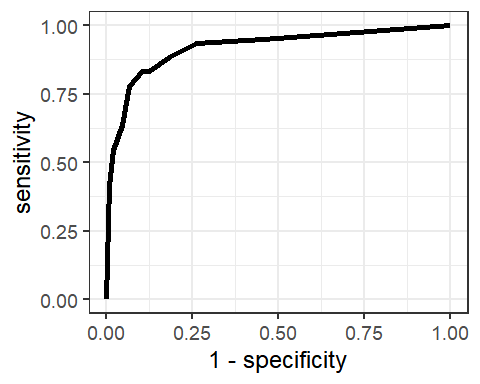
## K values for tuning  
kgrid <- expand.grid(k = seq(1, 51, by = 2))  
## LOOCV tuning  
tr <- trainControl(method = "repeatedcv",  
 number = 5,  
 repeats = 50)  
## Train the classifier  
knn <- train(  
 Outcome ~ .,  
 data = train,  
 method = "knn",  
 tuneGrid = kgrid,  
 trControl = tr  
)  
plot(knn)



knn$bestTune$k

## [1] 9

tuned\_knn <- train(  
 Outcome ~ .,  
 data = train,  
 method = "knn",  
 tuneGrid = expand.grid(k = knn$bestTune$k),  
 trControl = trainControl(method = "none")  
)  
  
knn\_pred <- predict(tuned\_knn,  
 newdata = test,  
 type = "raw")  
accuracy\_knn<-mean(knn\_pred == test$Outcome)  
  
knn\_prob <- predict(tuned\_knn,  
 newdata = test,  
 type = "prob")  
roccurve\_knn <- roc(response = test$Outcome, predictor = knn\_prob[,2])  
ggroc(roccurve\_knn, legacy.axes = TRUE, lwd=2) +theme\_bw(base\_size = 18)



auc\_knn<-auc(roccurve\_knn)