Classification Trees and Ensemble Methods

```
library(mlbench)
library(caret)
library(rpart)
library(rpart.plot)
library(party)
library(partykit)
library(randomForest)
library(ranger)
library(gbm)
library(plotmo)
library(pdp)
library(pROC)
library(lime)
```

We use the Pima Indians Diabetes Database (used in L6.Rmd) for illustration. The data contain 768 observations and 9 variables. The outcome is a binary variable `diabetes`.

```
data(PimaIndiansDiabetes)
dat <- PimaIndiansDiabetes
dat$diabetes <- factor(dat$diabetes, c("pos", "neg"))

set.seed(1)
rowTrain <- createDataPartition(y = dat$diabetes,
                                p = 0.75,
                                list = FALSE)
```

Classification trees

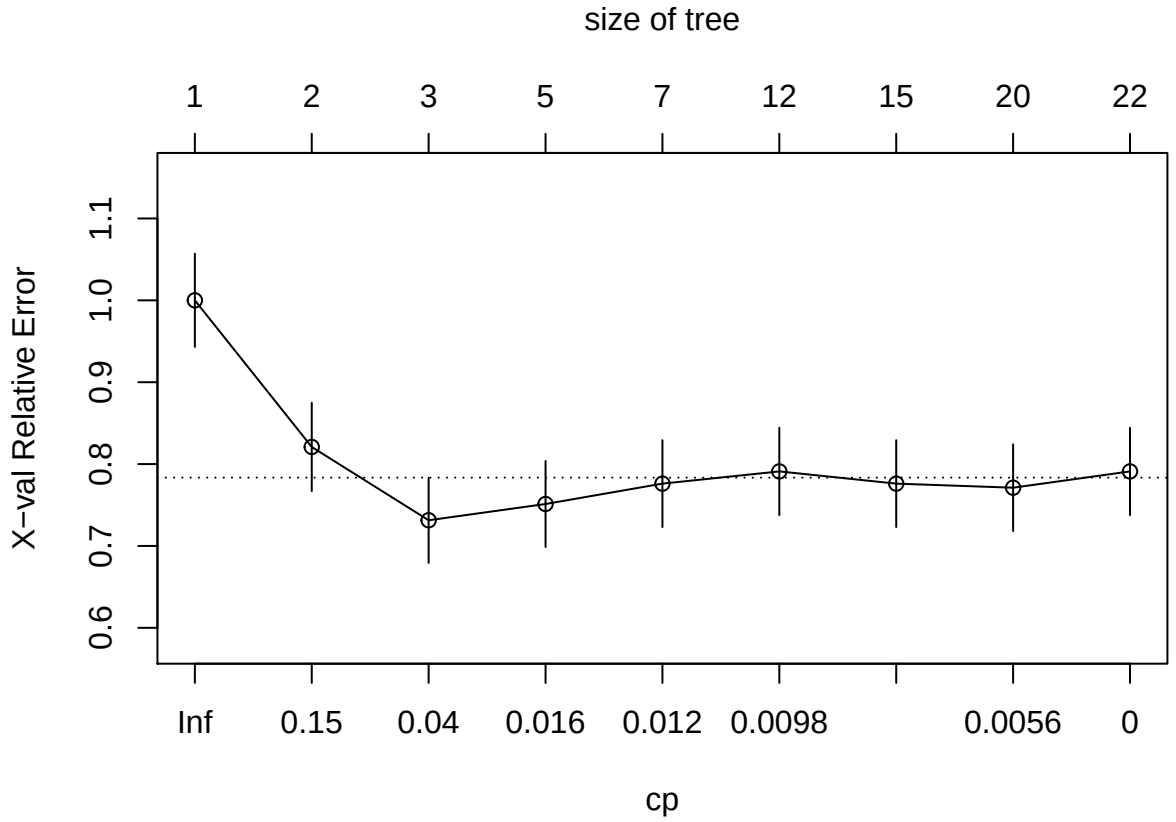
Using `rpart`

```
set.seed(1)
tree1 <- rpart(formula = diabetes~., data = dat,
               subset = rowTrain,
               control = rpart.control(cp = 0))

cpTable <- printcp(tree1)
```

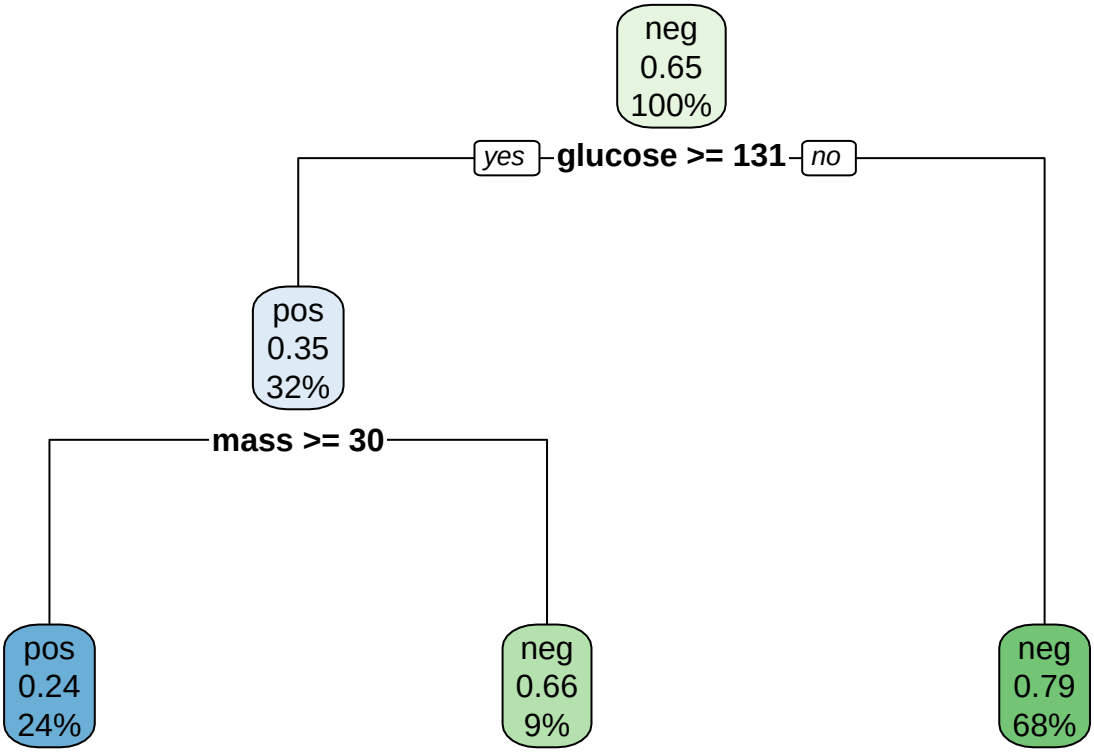
```
##
## Classification tree:
## rpart(formula = diabetes ~ ., data = dat, subset = rowTrain,
##     control = rpart.control(cp = 0))
##
## Variables actually used in tree construction:
## [1] age      glucose  insulin  mass     pedigree pregnant pressure
##
## Root node error: 201/576 = 0.34896
##
## n= 576
##
```

```
##          CP nsplit rel error  xerror     xstd
## 1 0.2736318      0   1.00000 1.00000 0.056912
## 2 0.0796020      1   0.72637 0.82090 0.053983
## 3 0.0199005      2   0.64677 0.73134 0.052057
## 4 0.0124378      4   0.60697 0.75124 0.052514
## 5 0.0116086      6   0.58209 0.77612 0.053062
## 6 0.0082919     11   0.50746 0.79104 0.053378
## 7 0.0062189     14   0.48259 0.77612 0.053062
## 8 0.0049751     19   0.43781 0.77114 0.052954
## 9 0.0000000     21   0.42786 0.79104 0.053378
```

```
plotcp(tree1)
```



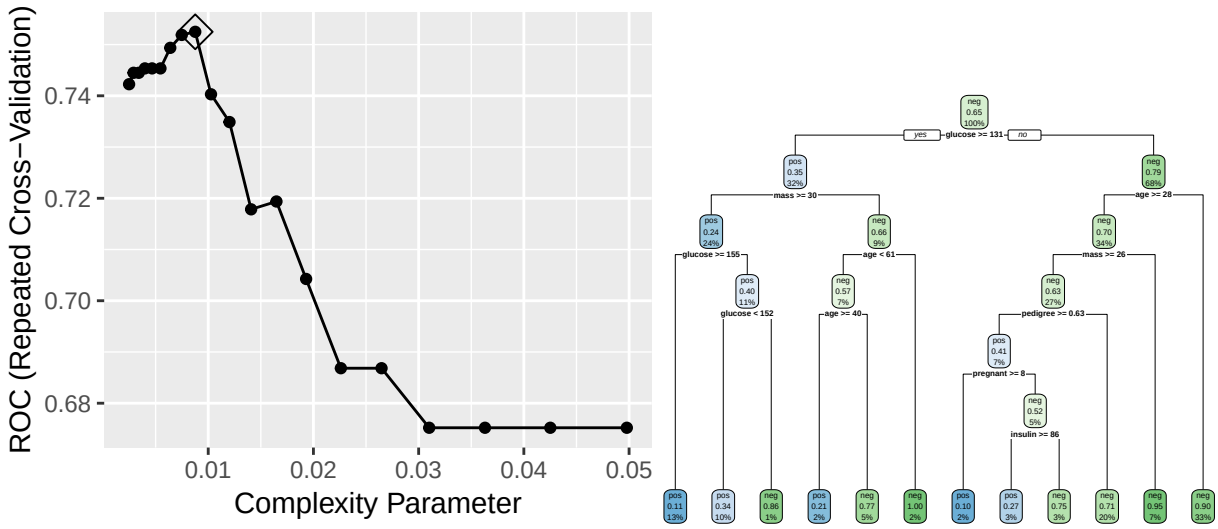

```
minErr <- which.min(cpTable[,4])

# minimum cross-validation error
tree2 <- prune(tree1, cp = cpTable[minErr,1])
rpart.plot(tree2)
```



Using `caret`

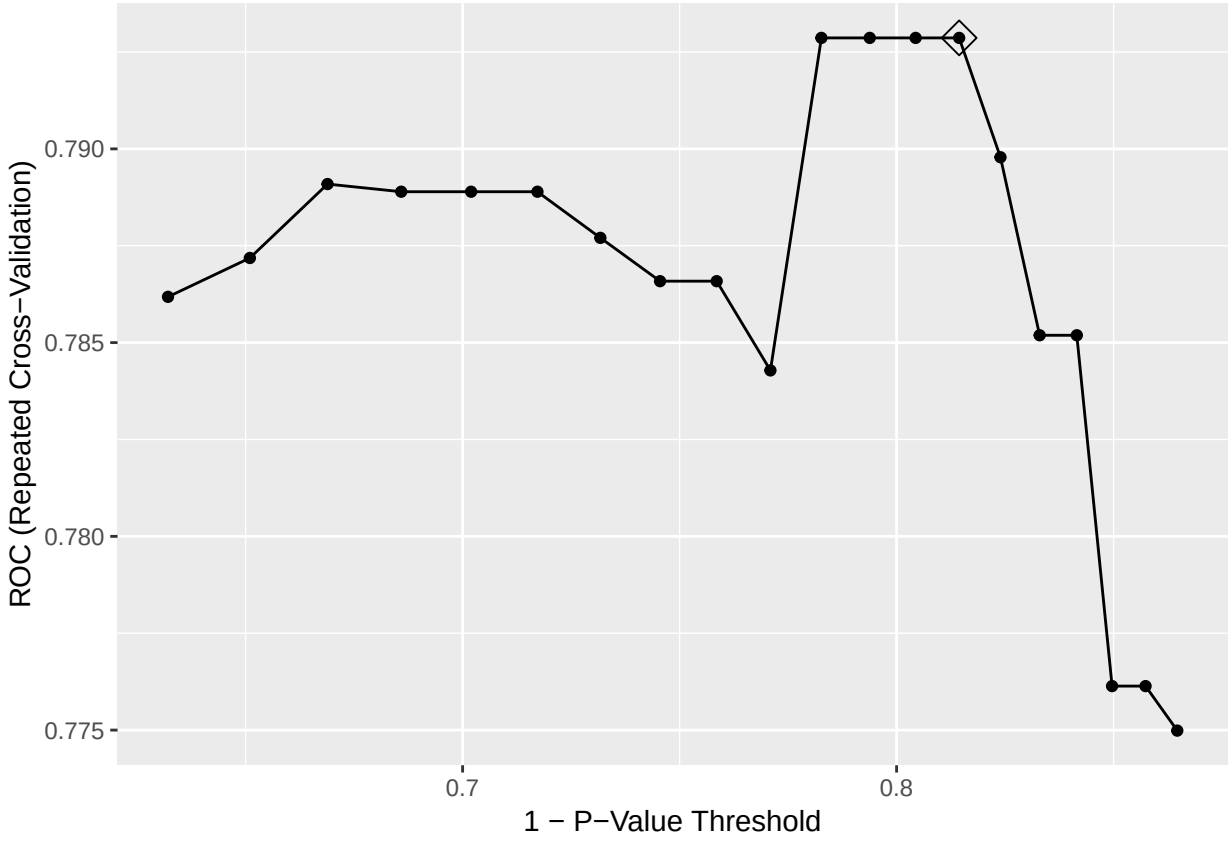
CART

```
ctrl <- trainControl(method = "repeatedcv",
                     summaryFunction = twoClassSummary,
                     classProbs = TRUE)

set.seed(1)
rpart.fit <- train(diabetes~., dat,
                   subset = rowTrain,
                   method = "rpart",
                   tuneGrid = data.frame(cp = exp(seq(-6,-3, len = 20))),
                   trControl = ctrl,
                   metric = "ROC")
ggplot(rpart.fit, highlight = TRUE)
rpart.plot(rpart.fit$finalModel)
```

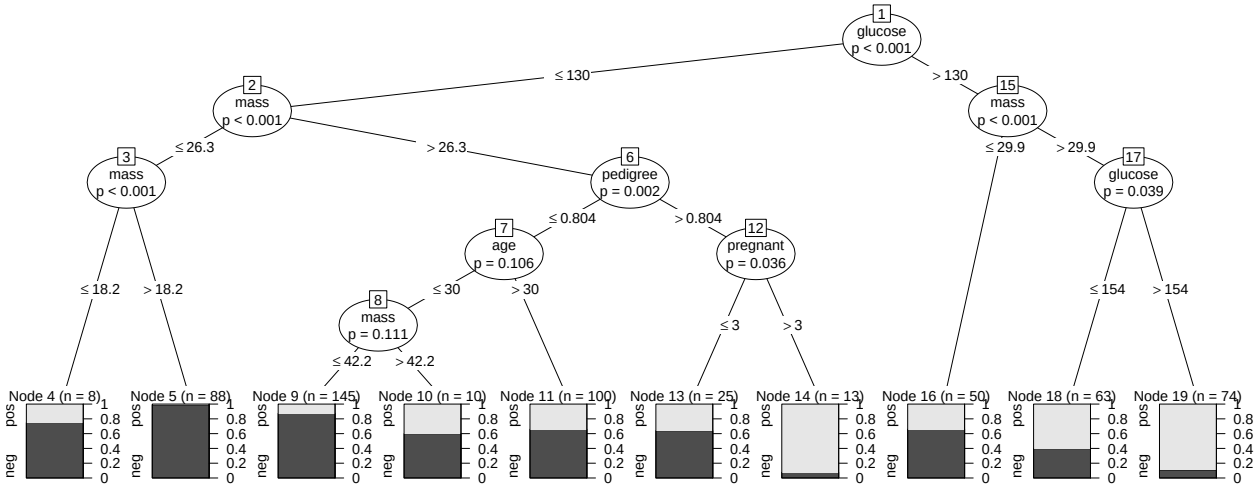


CIT

```
set.seed(1)
ctree.fit <- train(diabetes~., dat,
                   subset = rowTrain,
                   method = "ctree",
                   tuneGrid = data.frame(mincriterion = 1-exp(seq(-2, -1, length = 20))),
                   metric = "ROC",
                   trControl = ctrl)
ggplot(ctree.fit, highlight = TRUE)
```




```
plot(ctree.fit$finalModel)
```

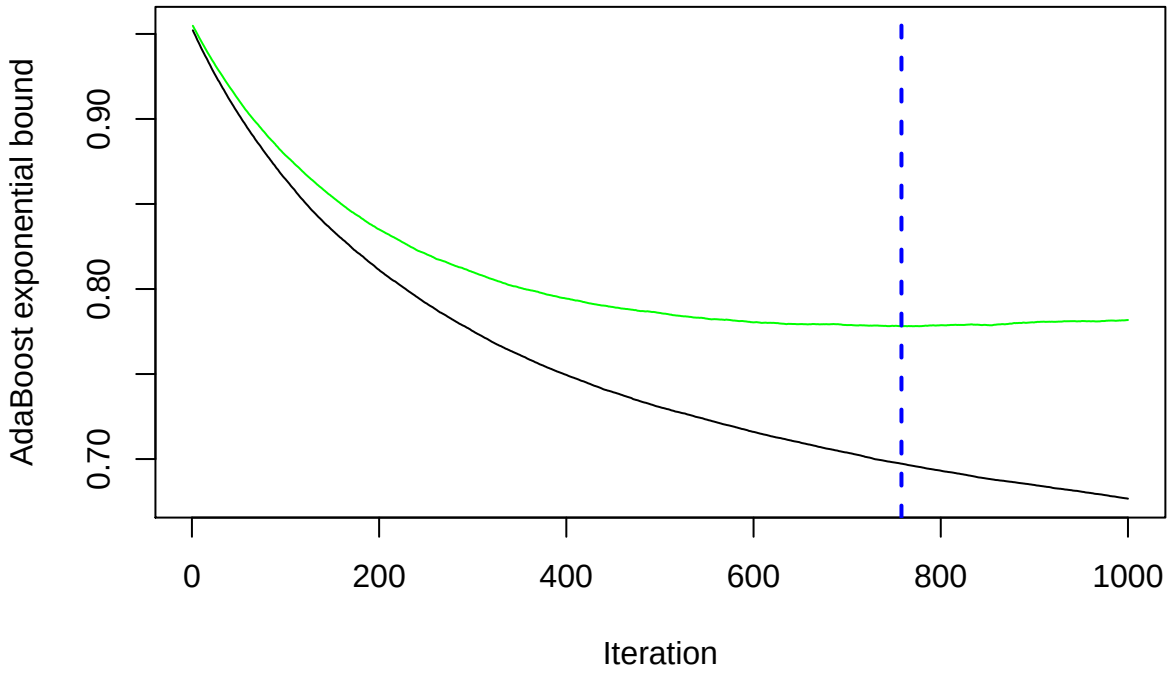



```
rpart.pred <- predict(tree2, newdata = dat[-rowTrain,])[,1]

rpartc.pred <- predict(rpart.fit, newdata = dat[-rowTrain,],
                       type = "prob")[,1]
ctree.pred <- predict(ctree.fit, newdata = dat[-rowTrain,],
                       type = "prob")[,1]
```

Random forests and boosting

```
set.seed(1)
bagging <- randomForest(diabetes~., dat[rowTrain,],
                        mtry = 8)

set.seed(1)
rf <- randomForest(diabetes~., dat[rowTrain,],
                   mtry = 3)

# fast implementation
set.seed(1)
rf2 <- ranger(diabetes~., dat[rowTrain,],
              mtry = 3, probability = TRUE)


rf.pred <- predict(rf, newdata = dat[-rowTrain,], type = "prob")[,1]
rf2.pred <- predict(rf2, data = dat[-rowTrain,], type = "response")$predictions[,1]
```

```
dat2 <- dat
dat2$diabetes <- as.numeric(dat$diabetes == "pos")

set.seed(1)
bst <- gbm(diabetes~., dat2[rowTrain,],
           distribution = "adaboost",
           n.trees = 1000,
           interaction.depth = 2,
           shrinkage = 0.005,
           cv.folds = 10)

nt <- gbm.perf(bst, method = "cv")
```



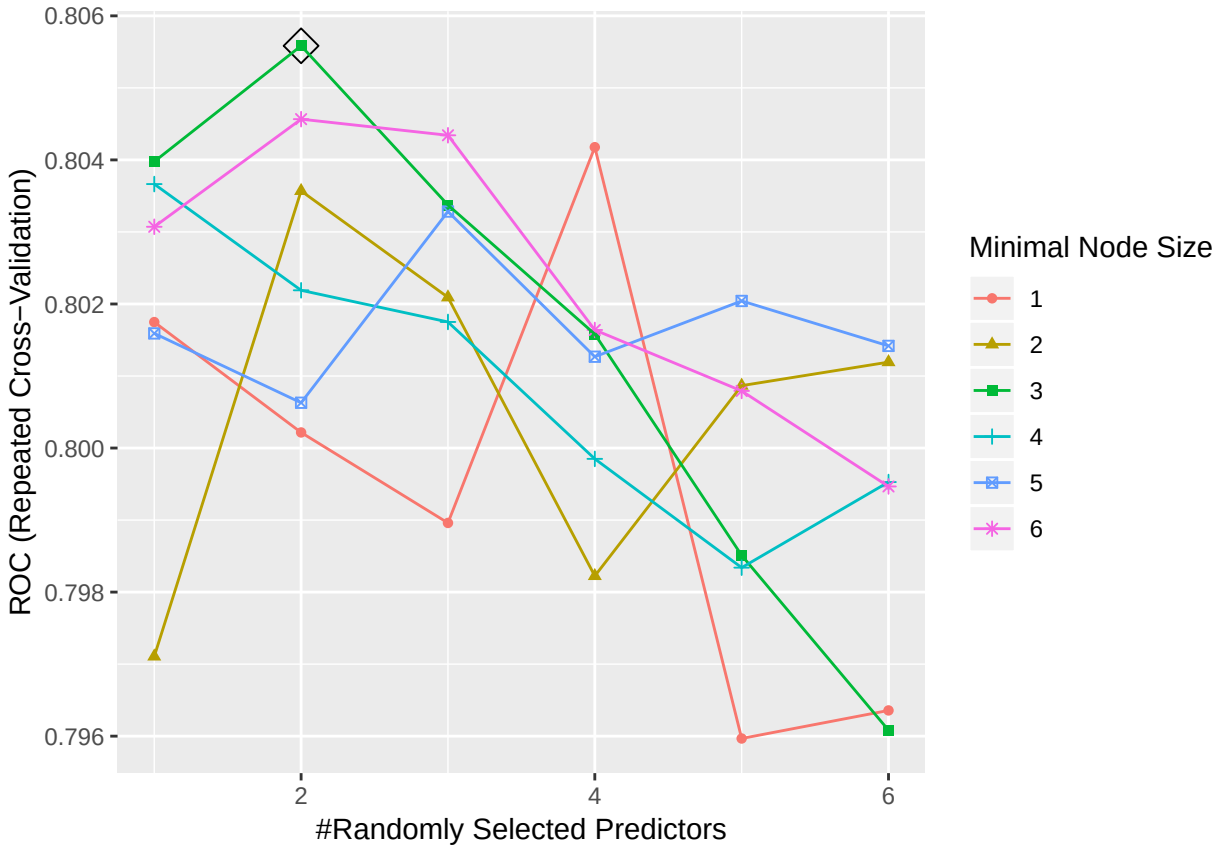
```
nt
```

```
## [1] 758
```

Grid search using `caret`

Random forests

```
# Try more if possible
rf.grid <- expand.grid(mtry = 1:6,
                       splitrule = "gini",
                       min.node.size = 1:6)
set.seed(1)
rf.fit <- train(diabetes~., dat,
                subset = rowTrain,
                method = "ranger",
                tuneGrid = rf.grid,
                metric = "ROC",
                trControl = ctrl)

ggplot(rf.fit, highlight = TRUE)
```

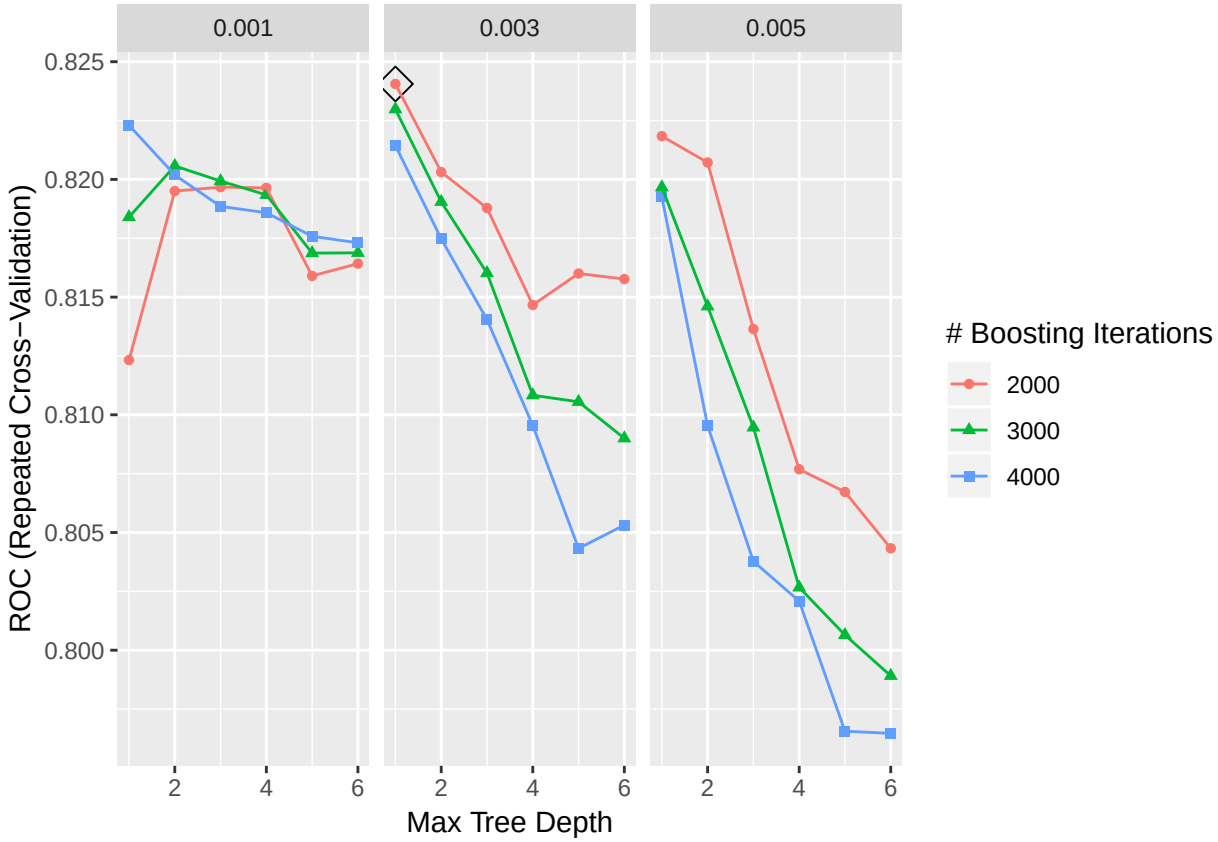



```
rf.pred <- predict(rf.fit, newdata = dat[-rowTrain,], type = "prob")[,1]
```

Boosting

Binomial loss

```
gbmB.grid <- expand.grid(n.trees = c(2000,3000,4000),
                         interaction.depth = 1:6,
                         shrinkage = c(0.001,0.003,0.005),
                         n.minobsinnode = 1)
set.seed(1)
# Binomial loss function
gbmB.fit <- train(diabetes~., dat,
                  subset = rowTrain,
                  tuneGrid = gbmB.grid,
                  trControl = ctrl,
                  method = "gbm",
                  distribution = "bernoulli",
                  metric = "ROC",
                  verbose = FALSE)

ggplot(gbmB.fit, highlight = TRUE)
```

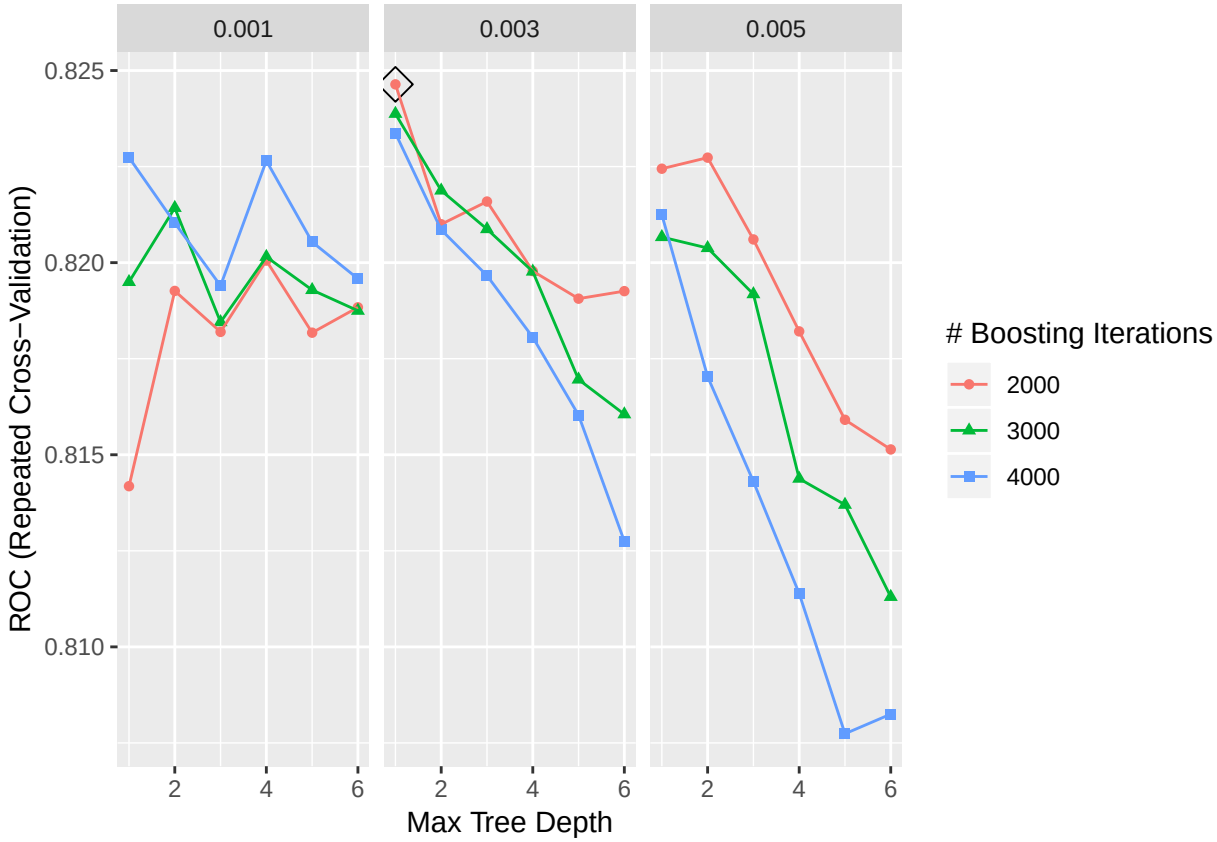



```
gbmB.pred <- predict(gbmB.fit, newdata = dat[-rowTrain,], type = "prob")[,1]
```

AdaBoost

```
gbmA.grid <- expand.grid(n.trees = c(2000,3000,4000),
                         interaction.depth = 1:6,
```

```
                        shrinkage = c(0.001,0.003,0.005),
                        n.minobsinnode = 1)
set.seed(1)
# Adaboost loss function
gbmA.fit <- train(diabetes~., dat,
                  subset = rowTrain,
                  tuneGrid = gbmA.grid,
                  trControl = ctrl,
                  method = "gbm",
                  distribution = "adaboost",
                  metric = "ROC",
                  verbose = FALSE)

ggplot(gbmA.fit, highlight = TRUE)
```




```
gbmA.pred <- predict(gbmA.fit, newdata = dat[-rowTrain,], type = "prob")[,1]
```

```
resamp <- resamples(list(rf = rf.fit,
                         gbmA = gbmA.fit,
                         gbmB = gbmB.fit,
                         rpart = rpart.fit,
                         ctree = ctree.fit))
summary(resamp)
```

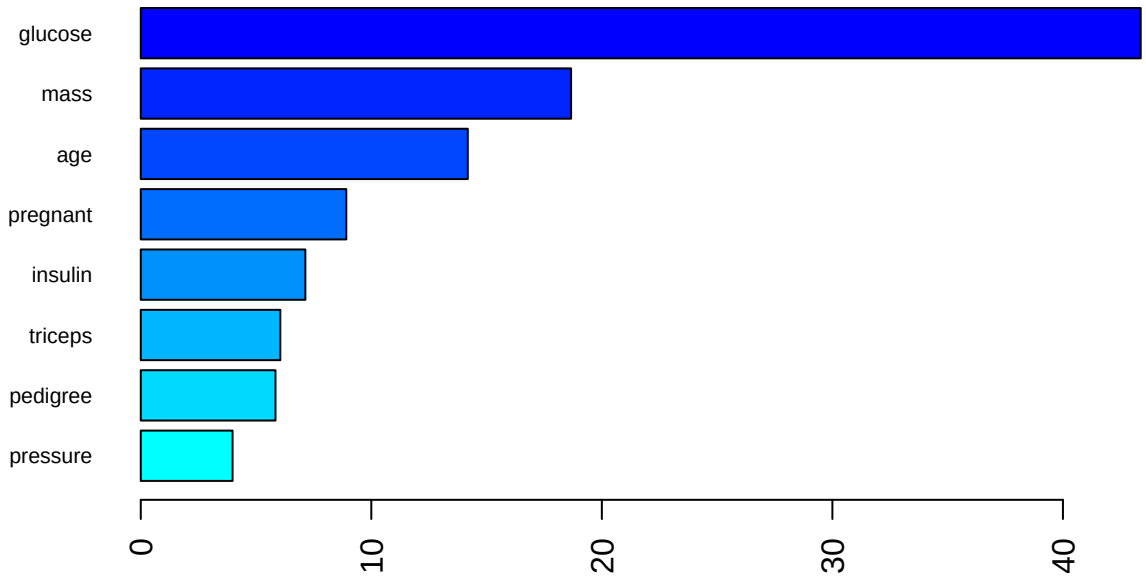
```
##
## Call:
## summary.resamples(object = resamp)
```

```
##
## Models: rf, gbmA, gbmB, rpart, ctree
## Number of resamples: 10
##
## ROC
##            Min.   1st Qu.    Median      Mean   3rd Qu.      Max. NA's
## rf    0.7302632 0.7526583 0.8002134 0.8055841 0.8395270 0.9223058    0
## gbmA  0.7618421 0.7891447 0.8209459 0.8246400 0.8537162 0.9197995    0
## gbmB  0.7578947 0.7914474 0.8141892 0.8240547 0.8537162 0.9273183    0
## rpart 0.6519737 0.6934877 0.7577703 0.7524912 0.7858108 0.8790727    0
## ctree 0.7105263 0.7697635 0.7918919 0.7928637 0.8065789 0.8934837    0
##
## Sens
##       Min. 1st Qu. Median      Mean 3rd Qu.      Max. NA's
## rf    0.35   0.425  0.550 0.5359524  0.5875 0.8095238    0
## gbmA  0.20   0.450  0.550 0.5261905  0.6250 0.7619048    0
## gbmB  0.20   0.500  0.525 0.5407143  0.6250 0.8571429    0
## rpart 0.30   0.550  0.600 0.5711905  0.6375 0.7619048    0
## ctree 0.50   0.550  0.550 0.5861905  0.6250 0.7619048    0
##
## Spec
##            Min.   1st Qu.    Median      Mean   3rd Qu.      Max. NA's
## rf    0.7631579 0.8378378 0.8399716 0.8453770 0.8618421 0.9189189    0
## gbmA  0.7894737 0.8389047 0.8783784 0.8773115 0.9144737 0.9473684    0
## gbmB  0.7894737 0.8648649 0.8801565 0.8825747 0.9331437 0.9473684    0
## rpart 0.7105263 0.7852063 0.8133001 0.8137269 0.8581081 0.9189189    0
## ctree 0.7894737 0.8120555 0.8648649 0.8453770 0.8684211 0.8918919    0
```

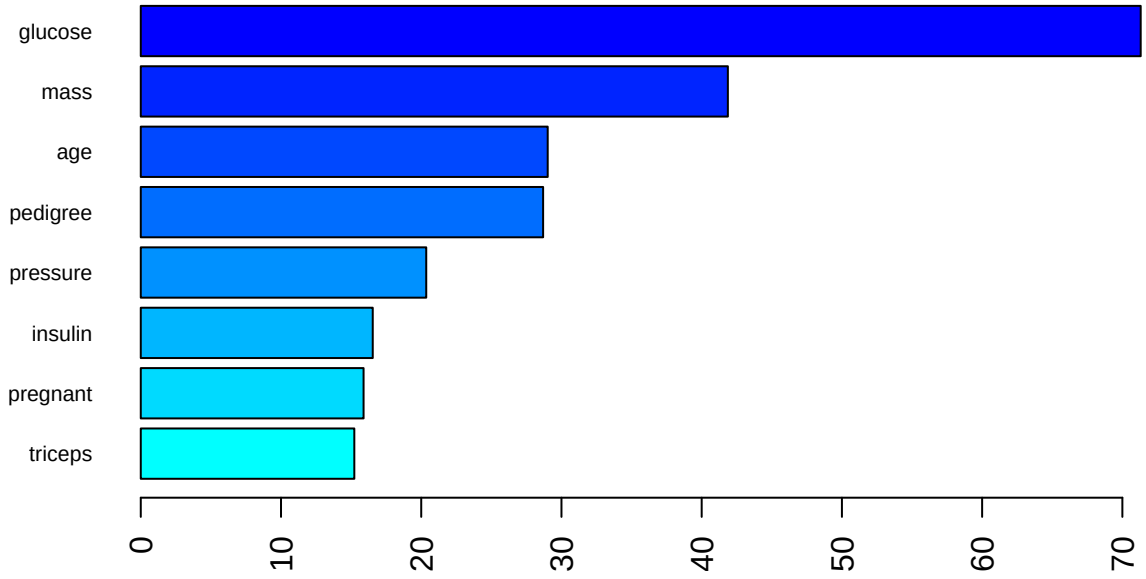
Understanding your models

Variable importance

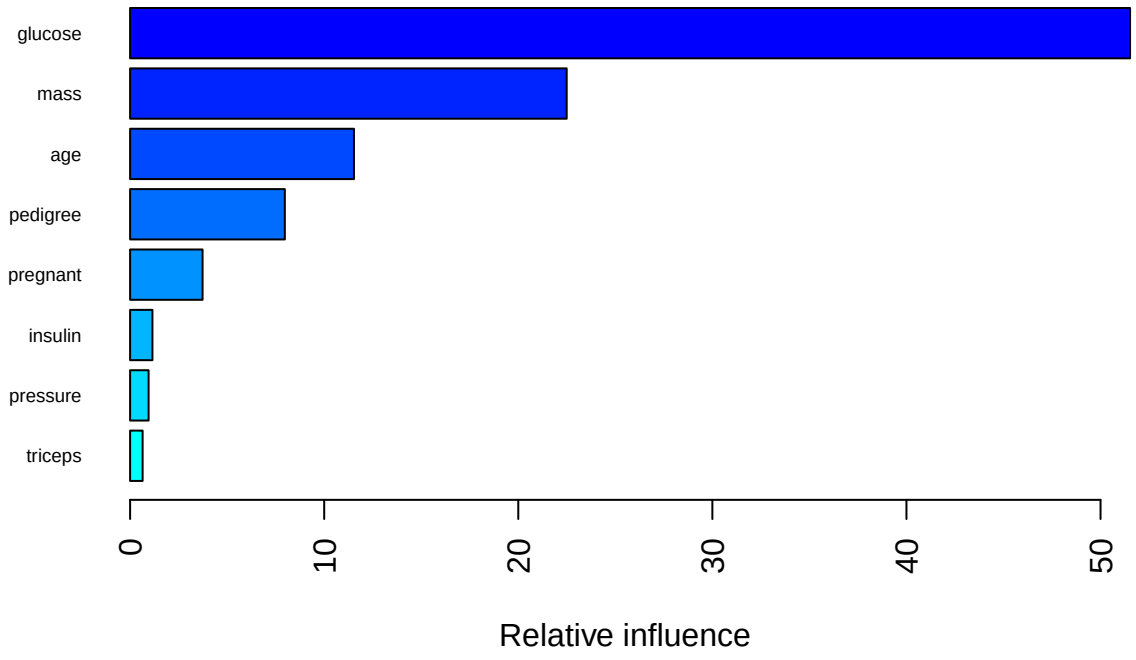
```
set.seed(1)
rf2.final.per <- ranger(diabetes~., dat[rowTrain,],
                        mtry = 3,
                        min.node.size = 5,
                        splitrule = "gini",
                        importance = "permutation",
                        scale.permutation.importance = TRUE)

barplot(sort(ranger::importance(rf2.final.per), decreasing = FALSE),
        las = 2, horiz = TRUE, cex.names = 0.7,
        col = colorRampPalette(colors = c("cyan","blue"))(8))
```




```
set.seed(1)
rf2.final.imp <- ranger(diabetes~., dat[rowTrain,],
                        mtry = 3, splitrule = "gini",
                        min.node.size = 5,
                        importance = "impurity")

barplot(sort(ranger::importance(rf2.final.imp), decreasing = FALSE),
        las = 2, horiz = TRUE, cex.names = 0.7,
        col = colorRampPalette(colors = c("cyan","blue"))(8))
```



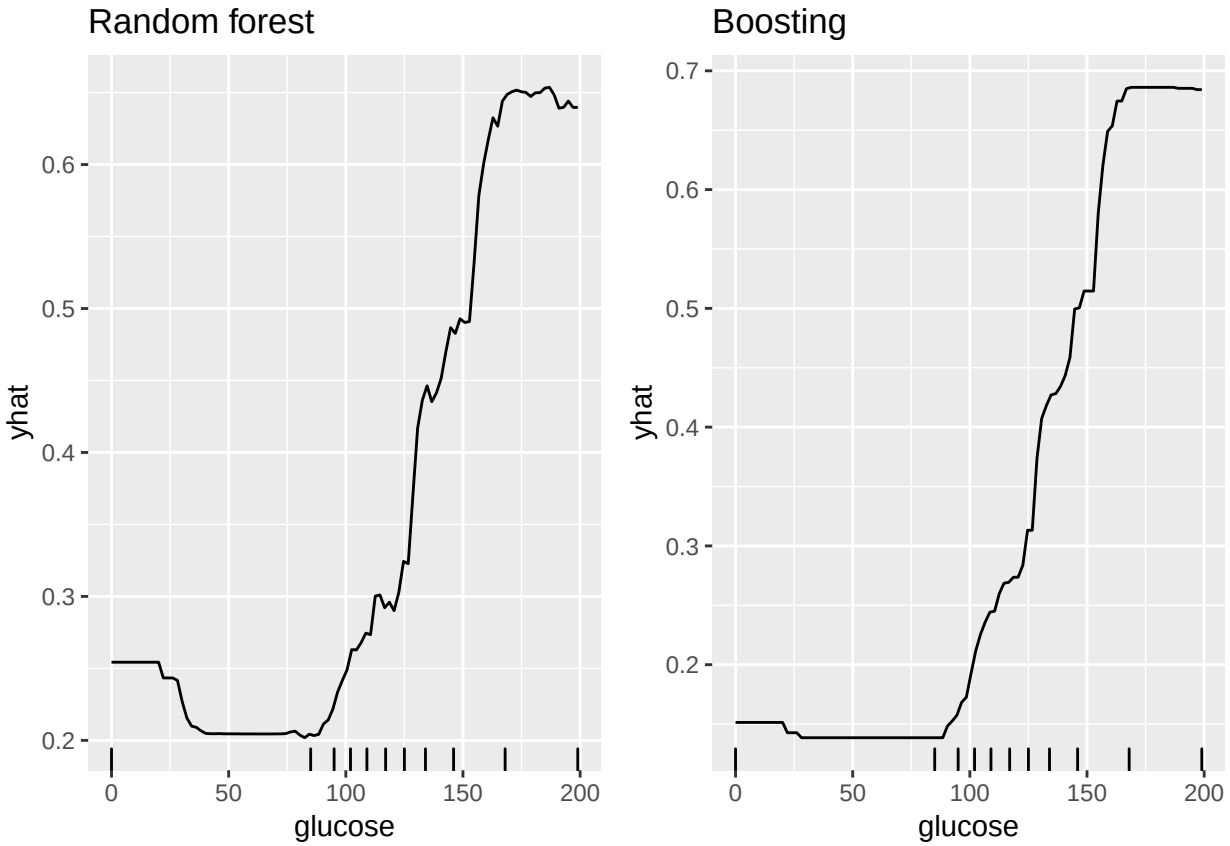

```
summary(gbmA.fit$finalModel, las = 2, cBars = 19, cex.names = 0.6)
```




```
##                var    rel.inf
## glucose   glucose 51.5213810
## mass         mass 22.4945309
## age           age 11.5328648
## pedigree pedigree  7.9719727
## pregnant pregnant  3.7327827
## insulin   insulin  1.1502190
## pressure pressure  0.9524305
## triceps   triceps  0.6438184
```

PDP

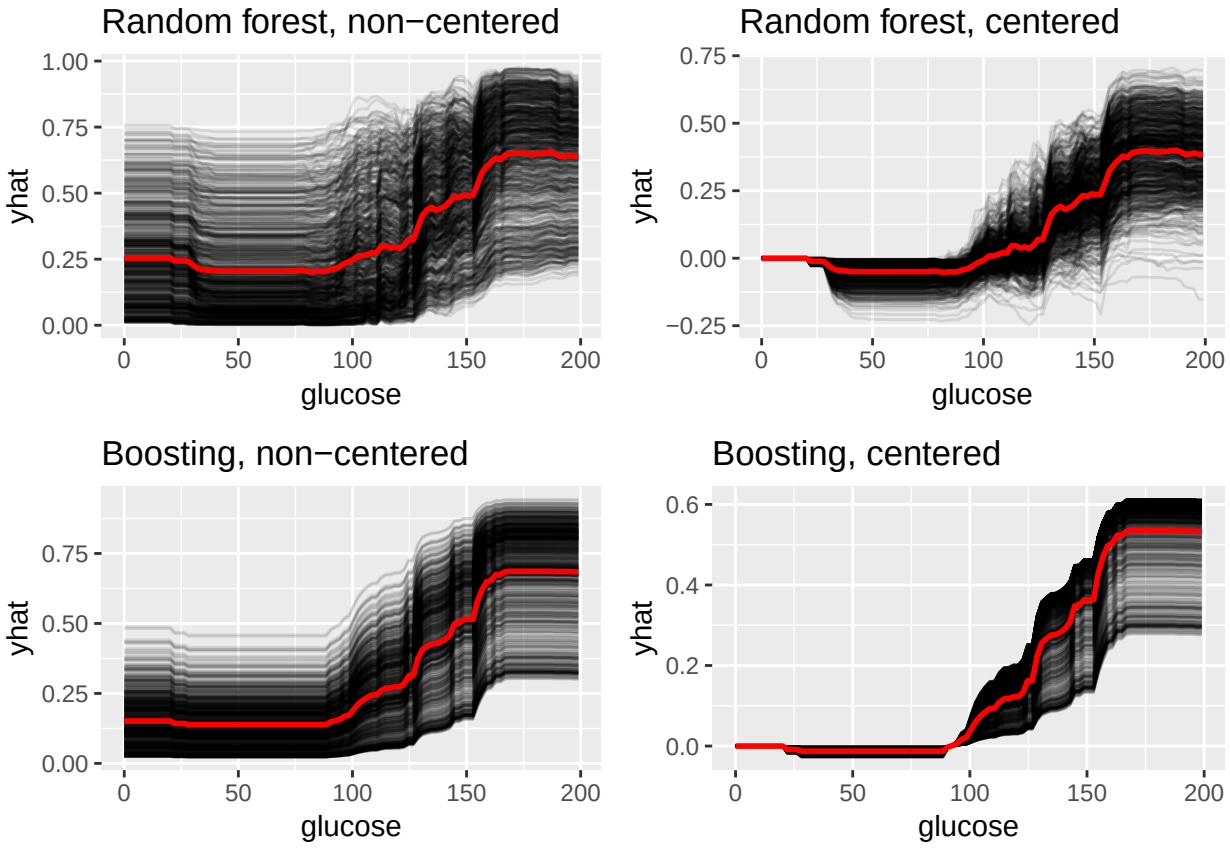
```
pdp.rf <- rf.fit %>%
  partial(pred.var = "glucose",
          grid.resolution = 100,
          prob = TRUE) %>%
  autoplot(rug = TRUE, train = dat[rowTrain,]) +
  ggtitle("Random forest")

pdp.gbm <- gbmA.fit %>%
  partial(pred.var = "glucose",
          grid.resolution = 100,
          prob = TRUE) %>%
  autoplot(rug = TRUE, train = dat[rowTrain,]) +
  ggtitle("Boosting")

grid.arrange(pdp.rf, pdp.gbm, nrow = 1)
```



ICE

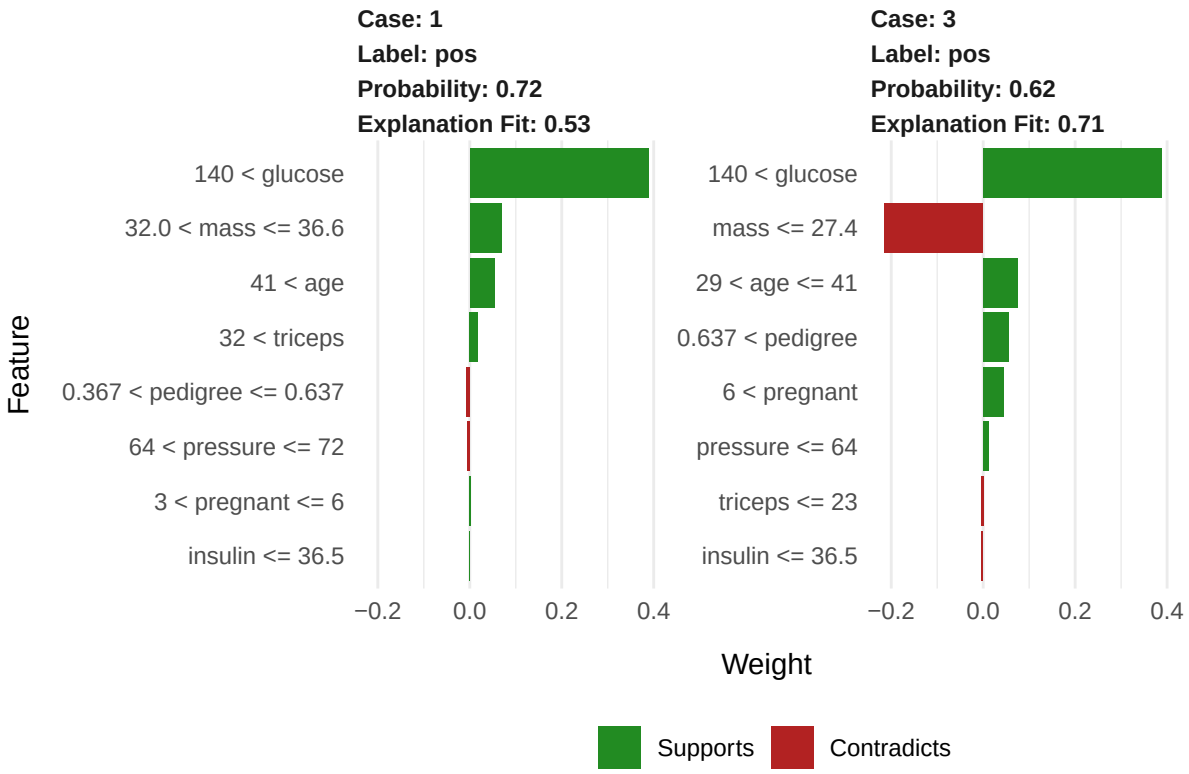
```
ice1.rf <- rf.fit %>%
  partial(pred.var = "glucose",
          grid.resolution = 100,
          ice = TRUE,
          prob = TRUE) %>%
  autoplot(train = dat[rowTrain,], alpha = .1) +
  ggtitle("Random forest, non-centered")

ice2.rf <- rf.fit %>%
  partial(pred.var = "glucose",
          grid.resolution = 100,
          ice = TRUE,
          prob = TRUE) %>%
  autoplot(train = dat[rowTrain,], alpha = .1,
           center = TRUE) +
  ggtitle("Random forest, centered")

ice1.gbm <- gbmA.fit %>%
  partial(pred.var = "glucose",
          grid.resolution = 100,
          ice = TRUE,
          prob = TRUE) %>%
  autoplot(train = dat[rowTrain,], alpha = .1) +
```

```
  ggtitle("Boosting, non-centered")

ice2.gbm <- gbmA.fit %>%
  partial(pred.var = "glucose",
          grid.resolution = 100,
          ice = TRUE,
          prob = TRUE) %>%
  autoplot(train = dat[rowTrain,], alpha = .1,
           center = TRUE) +
  ggtitle("Boosting, centered")

grid.arrange(ice1.rf, ice2.rf, ice1.gbm, ice2.gbm,
             nrow = 2, ncol = 2)
```

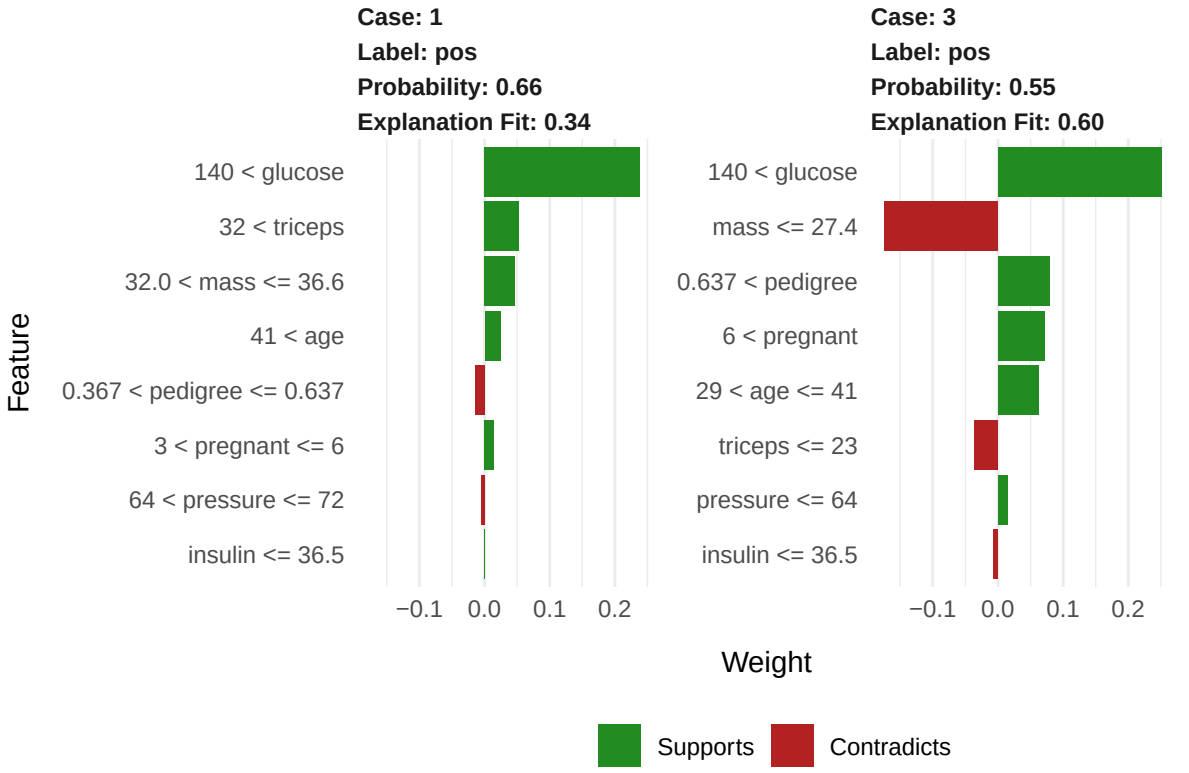


Explain your prediction

```
new_obs <- dat[-rowTrain,-9][1:2,]
explainer.gbm <- lime(dat[rowTrain,-9], gbmA.fit)
explanation.gbm <- explain(new_obs, explainer.gbm, n_features = 8,
                           labels = "pos")
plot_features(explanation.gbm)
```




```
explainer.rf <- lime(dat[rowTrain,-9], rf.fit)
explanation.rf <- explain(new_obs, explainer.rf, n_features = 8,
                          labels = "pos")
plot_features(explanation.rf)
```



Test data performance

```
roc.rpart <- roc(dat$diabetes[-rowTrain], rpart.pred)
roc.rpartc <- roc(dat$diabetes[-rowTrain], rpartc.pred)
roc.ctree <- roc(dat$diabetes[-rowTrain], ctree.pred)
roc.rf <- roc(dat$diabetes[-rowTrain], rf.pred)
roc.gbmA <- roc(dat$diabetes[-rowTrain], gbmA.pred)
roc.gbmB <- roc(dat$diabetes[-rowTrain], gbmB.pred)


plot(roc.rpart)
plot(roc.rpartc, add = TRUE, col = 2)
plot(roc.ctree, add = TRUE, col = 3)
plot(roc.rf, add = TRUE, col = 4)
plot(roc.gbmA, add = TRUE, col = 5)
plot(roc.gbmB, add = TRUE, col = 6)


auc <- c(roc.rpart$auc[1], roc.rpartc$auc[1], roc.ctree$auc[1],
         roc.rf$auc[1], roc.gbmA$auc[1], roc.gbmB$auc[1])

modelNames <- c("rpart","rpart_caret","ctree","rf","gbmA","gbmB")
legend("bottomright", legend = paste0(modelNames, ": ", round(auc,3)),
       col = 1:6, lwd = 2)
```