8106final_code

Ze Li

```r
# load libraries
library(dplyr)
library(tidyverse)
library(corrplot)
library(gridExtra)
library(ggplot2)
library(patchwork)

library(MASS)
library(mgcv)
library(earth)
library(Formula)
library(plotmo)
library(plotrix)
library(TeachingDemos)

library(caret)
library(glmnet)
library(tidymodels)
library(mlbench)
library(pROC)
library(pdp)
library(vip)
library(AppliedPredictiveModeling)
library(rsample)
library(klaR)
library(rpart)
library(rpart.plot)
library(party)
library(partykit)
library(randomForest)
library(ranger)
library(gbm)
library(e1071)
library(kernlab)
library(ggrepel)
```

```r
# load data
load("severity_training.RData")
load("severity_test.RData")
train <- as.data.frame(training_data)
test <- as.data.frame(test_data)
head(train)
```

```
##   id age gender race smoking height weight  bmi diabetes hypertension SBP LDL
## 1  1  59      0    1       1  170.3   74.7 25.8        0            0 120  95
## 2  2  54      1    1       1  170.8   75.7 26.0        1            1 133  87
## 3  3  55      1    3       1  172.7   89.5 30.0        0            0 123 139
## 4  4  59      0    1       0  171.7   74.6 25.3        0            0 121 126
## 6  6  64      1    1       0  168.8   87.9 30.8        0            1 132  99
## 9  9  67      0    1       0  168.5   76.5 27.0        0            1 138  97
##   vaccine depression severity
## 1       1          5        0
## 2       0          2        1
## 3       1          5        0
## 4       1          4        0
## 6       1          9        0
## 9       0          8        1
```

```
head(test)
```

```
##    id age gender race smoking height weight  bmi diabetes hypertension SBP LDL
## 5   5  62      1    1       0  160.2   75.8 29.5        1            0 122 107
## 7   7  64      0    1       0  172.4   80.9 27.2        0            0 122  99
## 8   8  62      1    4       1  175.6   77.4 25.1        0            0 119 123
## 16 16  62      1    1       0  174.7   92.4 30.3        0            1 146 108
## 20 20  56      0    1       0  172.3   74.2 25.0        0            0 130 146
## 23 23  66      1    1       1  173.2   68.3 22.8        0            1 133 106
##    vaccine depression severity
## 5        1          9        0
## 7        0          7        0
## 8        0          6        0
## 16       0          4        1
## 20       1          7        0
## 23       0          8        1
```

```
train.raw =
  train[, -1] %>%
  mutate(
    gender = as.factor(gender),
    race = as.factor(race),
    smoking = as.factor(smoking),
    hypertension = as.factor(hypertension),
    diabetes = as.factor(diabetes),
    vaccine = as.factor(vaccine),
    severity = as.factor(severity)
    )

test.raw =
  test[, -1] %>%
  mutate(
    gender = as.factor(gender),
    race = as.factor(race),
    smoking = as.factor(smoking),
    hypertension = as.factor(hypertension),
    diabetes = as.factor(diabetes),
    vaccine = as.factor(vaccine),
```

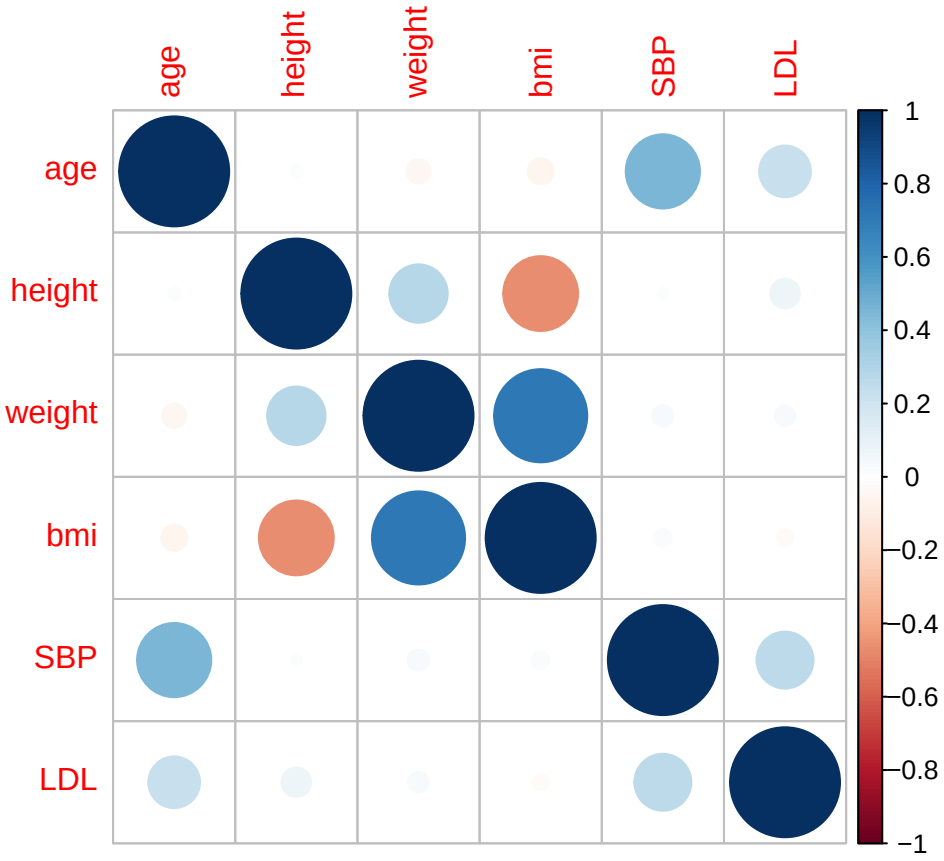
```
    severity = as.factor(severity)
    )

x_train <- model.matrix(severity ~ . , train.raw)[, -1]
y_train <- train.raw$severity
x_test <- model.matrix(severity ~ . , test.raw)[, -1]
y_test <- test.raw$severity
head(train.raw)
```

```
##   age gender race smoking height weight  bmi diabetes hypertension SBP LDL
## 1  59      0    1       1  170.3   74.7 25.8        0            0 120  95
## 2  54      1    1       1  170.8   75.7 26.0        1            1 133  87
## 3  55      1    3       1  172.7   89.5 30.0        0            0 123 139
## 4  59      0    1       0  171.7   74.6 25.3        0            0 121 126
## 6  64      1    1       0  168.8   87.9 30.8        0            1 132  99
## 9  67      0    1       0  168.5   76.5 27.0        0            1 138  97
##   vaccine depression severity
## 1       1          5        0
## 2       0          2        1
## 3       1          5        0
## 4       1          4        0
## 6       1          9        0
## 9       0          8        1
```

Exploratary Data Analysis

Correlation plot

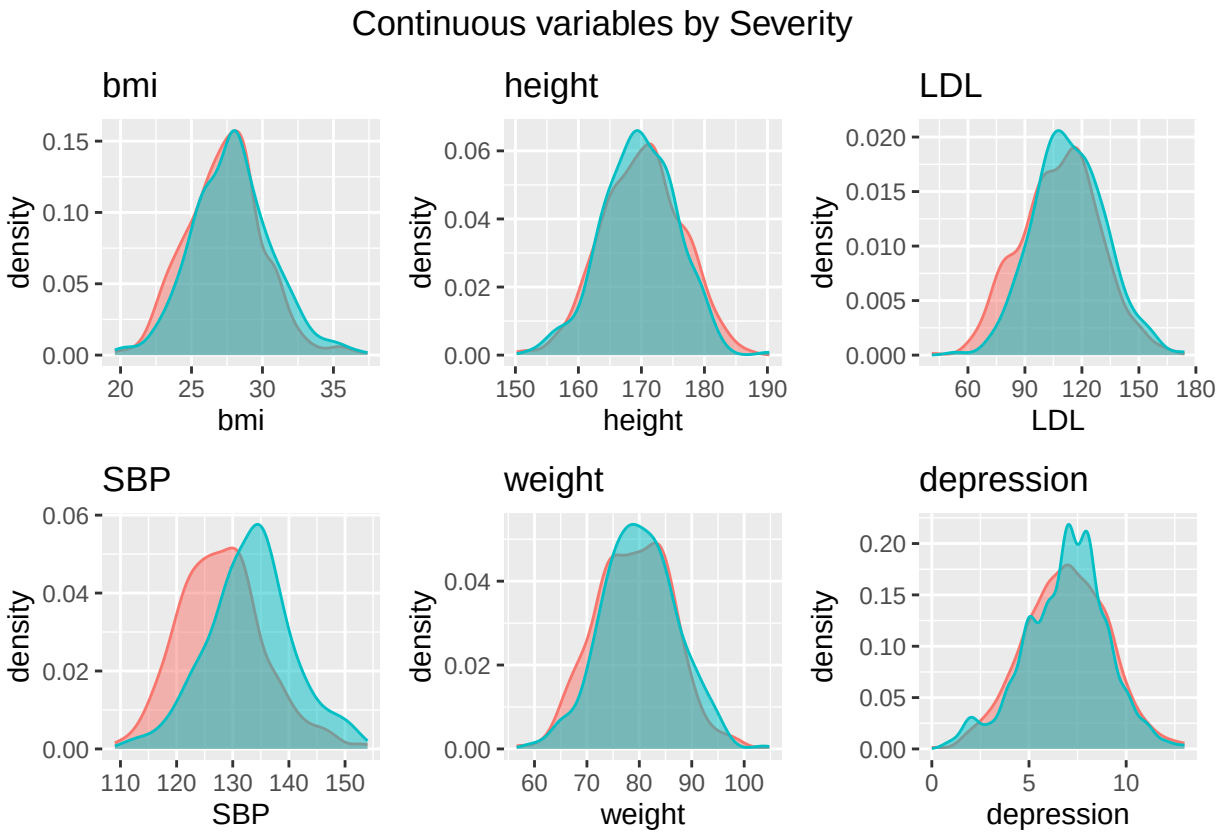
```
corr_dat = train.raw %>%
  dplyr::select('age', 'height', 'weight', 'bmi','SBP', 'LDL')
corrplot(cor(corr_dat), method = "circle", type = "full")
```



Continuous Variables

```
p_bmi <- ggplot(train.raw, aes(x = bmi, fill = as.factor(severity),
                               color = as.factor(severity))) +
  geom_density(alpha = 0.5) +
  labs(title = "bmi") +
  theme(legend.position = "none")

p_height <- ggplot(train.raw, aes(x = height, fill = as.factor(severity),
                                  color = as.factor(severity))) +
  geom_density(alpha = 0.5) +
  labs(title = "height") +
  theme(legend.position = "none")

p_ldl <- ggplot(train.raw, aes(x = LDL, fill = as.factor(severity),
                               color = as.factor(severity))) +
  geom_density(alpha = 0.5) +
  labs(title = "LDL") +
  theme(legend.position = "none")

p_sbp <- ggplot(train.raw, aes(x = SBP, fill = as.factor(severity),
                               color = as.factor(severity))) +
  geom_density(alpha = 0.5) +
  labs(title = "SBP") +
```

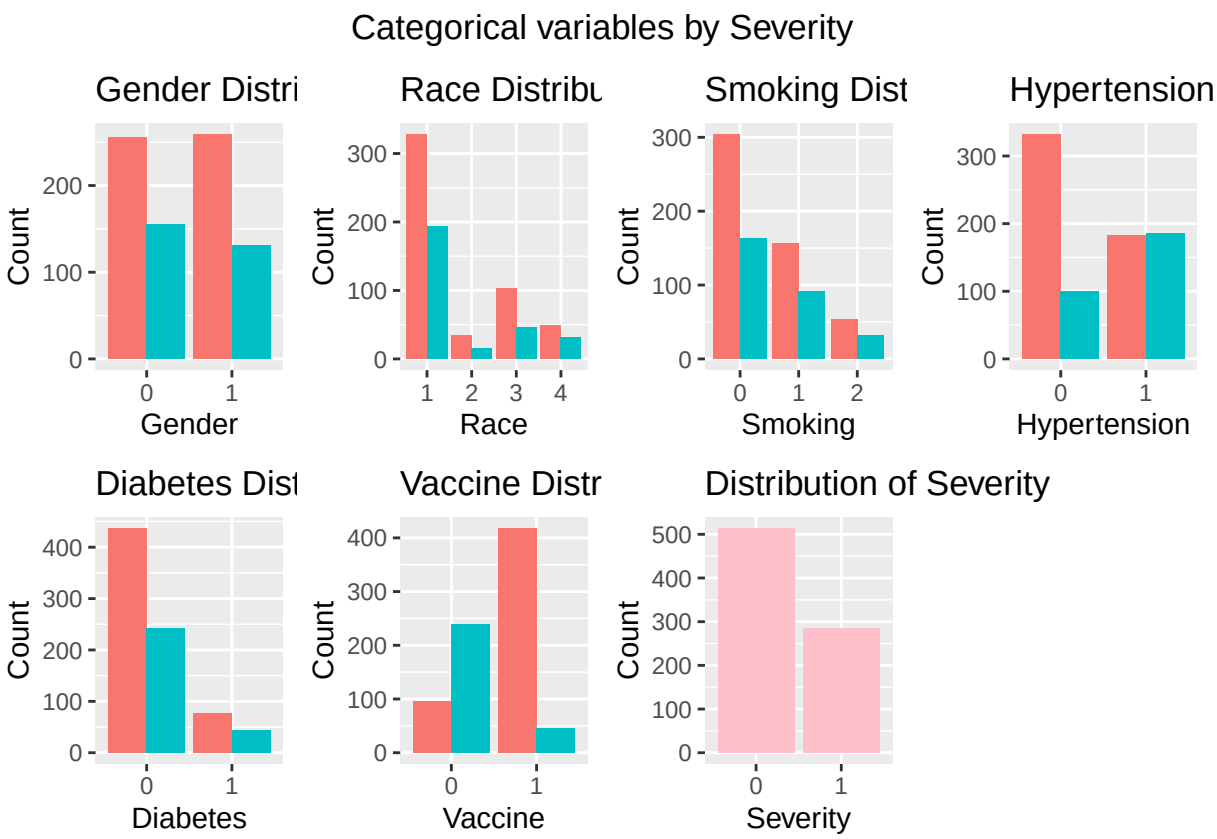
```
  theme(legend.position = "none")

p_weight <- ggplot(train.raw, aes(x = weight, fill = as.factor(severity),
                                  color = as.factor(severity))) +
  geom_density(alpha = 0.5) +
  labs(title = "weight") +
  theme(legend.position = "none")

p_depression <- ggplot(train.raw, aes(x = depression, fill = as.factor(severity),
                                      color = as.factor(severity))) +
  geom_density(alpha = 0.5) +
  labs(title = "depression") +
  theme(legend.position = "none")

plot_grid <- p_bmi + p_height + p_ldl + p_sbp + p_weight + p_depression +
  plot_layout(ncol = 3, byrow = TRUE) +
  plot_annotation(title="Continuous variables by Severity", # Adding the title here
                  theme = theme(plot.title = element_text(hjust = 0.5)))  # Center the title)

plot_grid
```



Categorical variables

```
# Bar Plot for Gender
```

```r
p11 <- ggplot(train.raw, aes(x = as.factor(gender), fill = as.factor(severity))) +
  geom_bar(position = "dodge") +
  labs(title = "Gender Distribution", x = "Gender", y = "Count") +
  theme(legend.position = "none")

# Bar Plot for Race
p12 <- ggplot(train.raw, aes(x = as.factor(race), fill = as.factor(severity))) +
  geom_bar(position = "dodge") +
  labs(title = "Race Distribution", x = "Race", y = "Count") +
  theme(legend.position = "none")

# Bar Plot for Smoking Status
p13 <- ggplot(train.raw, aes(x = as.factor(smoking), fill = as.factor(severity))) +
  geom_bar(position = "dodge") +
  labs(title = "Smoking Distribution", x = "Smoking", y = "Count") +
  theme(legend.position = "none")

# Bar Plot for Hypertension
p14 <- ggplot(train.raw, aes(x = as.factor(hypertension), fill = as.factor(severity))) +
  geom_bar(position = "dodge") +
  labs(title = "Hypertension Distribution", x = "Hypertension", y = "Count") +
  theme(legend.position = "none")

# Bar Plot for Diabetes
p15 <- ggplot(train.raw, aes(x = as.factor(diabetes), fill = as.factor(severity))) +
  geom_bar(position = "dodge") +
  labs(title = "Diabetes Distribution", x = "Diabetes", y = "Count") +
  theme(legend.position = "none")

# Bar plot for Vaccine
p16 <- ggplot(train.raw, aes(x = as.factor(vaccine), fill = as.factor(severity))) +
  geom_bar(position = "dodge") +
  labs(title = "Vaccine Distribution", x = "Vaccine", y = "Count") +
  theme(legend.position = "none")

# Bar plot for Severity
p17 <- ggplot(train.raw, aes(x = severity)) +
  geom_bar(fill = "#FFC0CB") +
  labs(title = "Distribution of Severity", x = "Severity", y = "Count")

# Combine the plots into a 2x4 grid
plot_grid2 <- p11 + p12 + p13 + p14 + p15 + p16 + p17 +
  plot_layout(ncol = 4, byrow = TRUE) +
  plot_annotation(title="Categorical variables by Severity", # Adding the title here
                  theme = theme(plot.title = element_text(hjust = 0.5)))  # Center the title)

# Display the combined plot
plot_grid2
```



Model fitting

logistic regression

```
# Using caret
ctrl <- trainControl(method = "cv", number = 10,
                     summaryFunction = twoClassSummary,
                     classProbs = TRUE)
levels(y_train) <- make.names(levels(y_train))
set.seed(83)
model.glm <- train(x = x_train,
                   y = y_train,
                   method = "glm",
                   metric = "ROC",
                   trControl = ctrl)

summary(model.glm)
```

glm

```
##
## Call:
## NULL
##
```

```
## Coefficients:
##                  Estimate Std. Error z value Pr(>|z|)
## (Intercept)   -36.142676  30.259726  -1.194  0.23232
## age             0.064795   0.028105   2.305  0.02114 *
## gender1        -0.409132   0.211384  -1.935  0.05293 .
## race2          -0.202620   0.441920  -0.458  0.64659
## race3           0.017372   0.288969   0.060  0.95206
## race4          -0.174620   0.352767  -0.495  0.62060
## smoking1        0.024966   0.234778   0.106  0.91531
## smoking2        0.492400   0.352148   1.398  0.16203
## height          0.111718   0.178298   0.627  0.53093
## weight         -0.133375   0.190301  -0.701  0.48339
## bmi             0.537585   0.541321   0.993  0.32066
## diabetes1       0.253028   0.292635   0.865  0.38723
## hypertension1   0.380927   0.340032   1.120  0.26260
## SBP             0.070811   0.023346   3.033  0.00242 **
## LDL             0.010022   0.005828   1.720  0.08548 .
## vaccine1       -3.617987   0.241871 -14.958  < 2e-16 ***
## depression     -0.037969   0.050059  -0.758  0.44816
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 1043.15  on 799  degrees of freedom
## Residual deviance:  593.41  on 783  degrees of freedom
## AIC: 627.41
##
## Number of Fisher Scoring iterations: 5
```

```
## test error
glm.pred <- predict(model.glm, newdata = x_test, type = "prob")[,2]
roc.glm <- roc(y_test, glm.pred)
```

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

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

```
1-roc.glm$auc[1]
```

```
## [1] 0.1039316
```

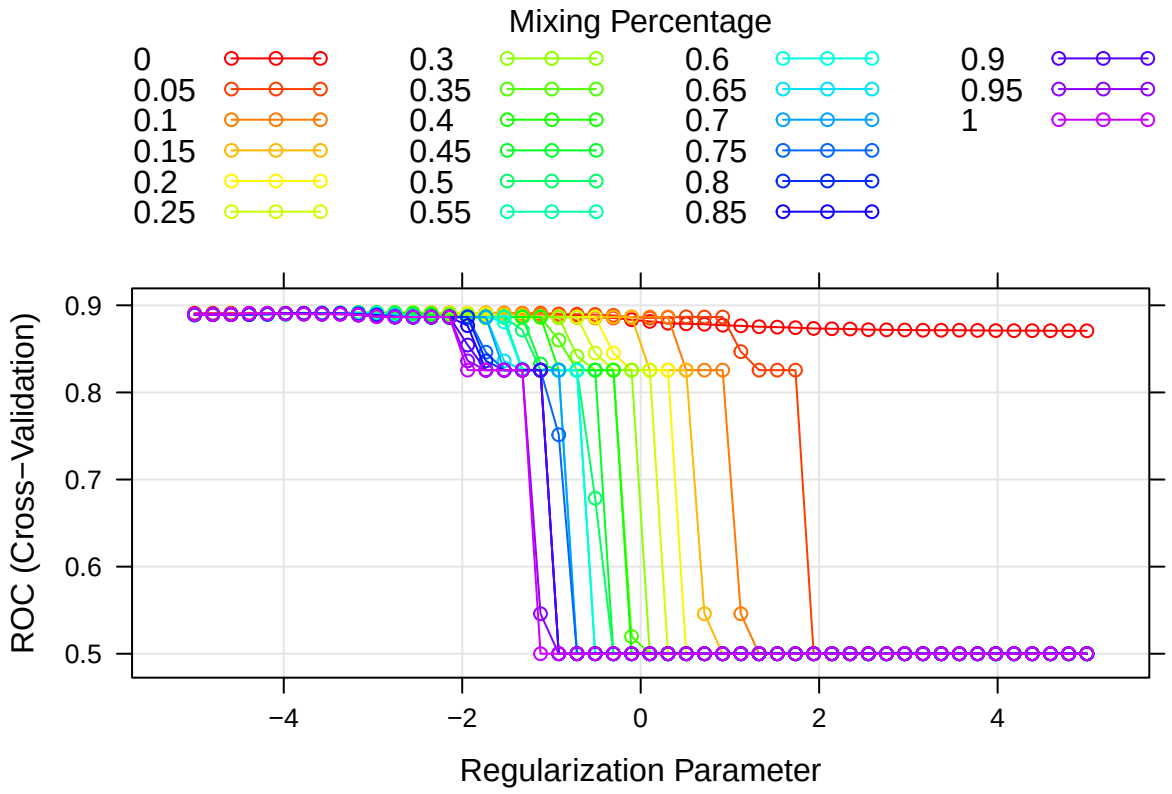
```
glmnGrid <- expand.grid(.alpha = seq(0, 1, length = 21),
                        .lambda = exp(seq(-5, 5, length = 50)))
set.seed(83)
model.glmn <- train(x = x_train,
                    y = y_train,
                    method = "glmnet",
                    tuneGrid = glmnGrid,
```

```
                    metric = "ROC",
                    trControl = ctrl)

# glmn best tune
model.glmn$bestTune
```

Penalized logistic regression

```
##     alpha     lambda
## 263  0.25 0.07800203
```

```
myCol <- rainbow(25)
myPar <- list(superpose.symbol = list(col = myCol),
              superpose.line = list(col = myCol))

plot(model.glmn, par.settings = myPar, xTrans = function(x) log(x))
```




```
# test error
glmn.pred <- predict(model.glmn, newdata = x_test, type = "prob")[,2]
roc.glmn <- roc(y_test, glmn.pred)
```

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

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

```r
1-roc.glmn$auc[1]
```

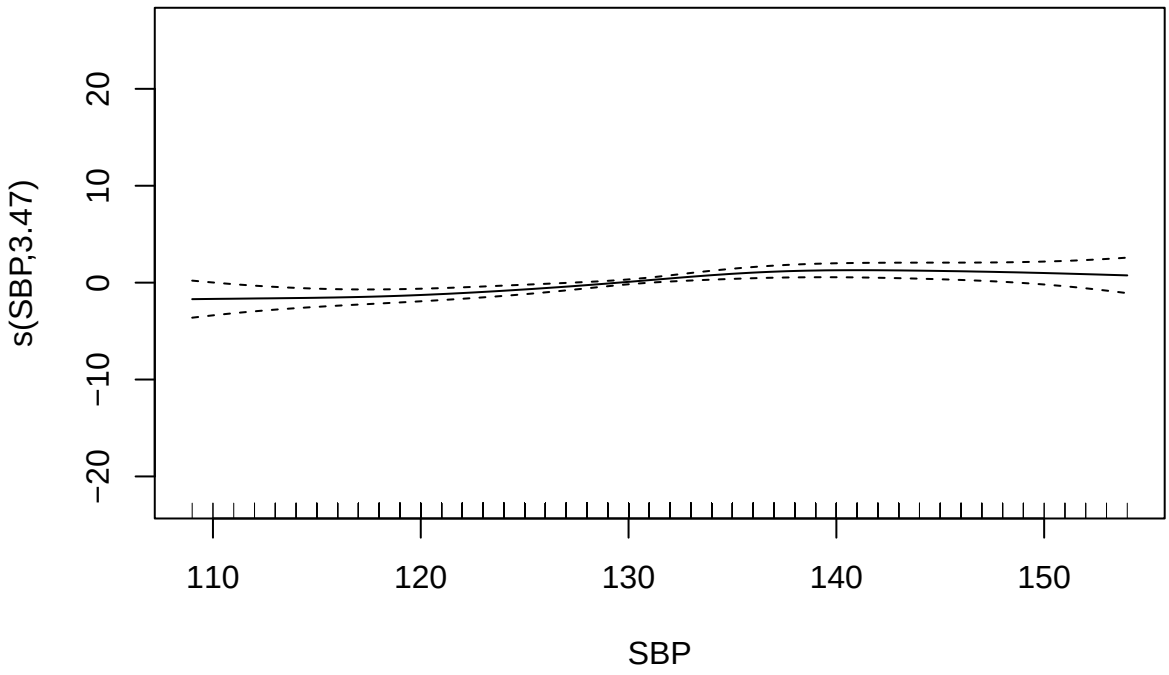
```
## [1] 0.1039316
```

```r
set.seed(83)
model.gam <- train(x = x_train,
                   y = y_train,
                   method = "gam",
                   metric = "ROC",
                   trControl = ctrl)

# gam best tune
model.gam$finalModel
```

GAM

```
##
## Family: binomial
## Link function: logit
##
## Formula:
## .outcome ~ gender1 + race2 + race3 + race4 + smoking1 + smoking2 +
##     diabetes1 + hypertension1 + vaccine1 + s(depression) + s(age) +
##     s(SBP) + s(LDL) + s(bmi) + s(height) + s(weight)
##
## Estimated degrees of freedom:
## 1.00 1.06 3.47 1.74 1.00 7.69 1.00
##  total = 26.97
##
## UBRE score: -0.2345662
```

```r
plot(model.gam$finalModel, select = 3)
```




```
# test error
gam.pred <- predict(model.gam, newdata = x_test, type = "prob")[,2]
roc.gam <- roc(y_test, gam.pred)
```

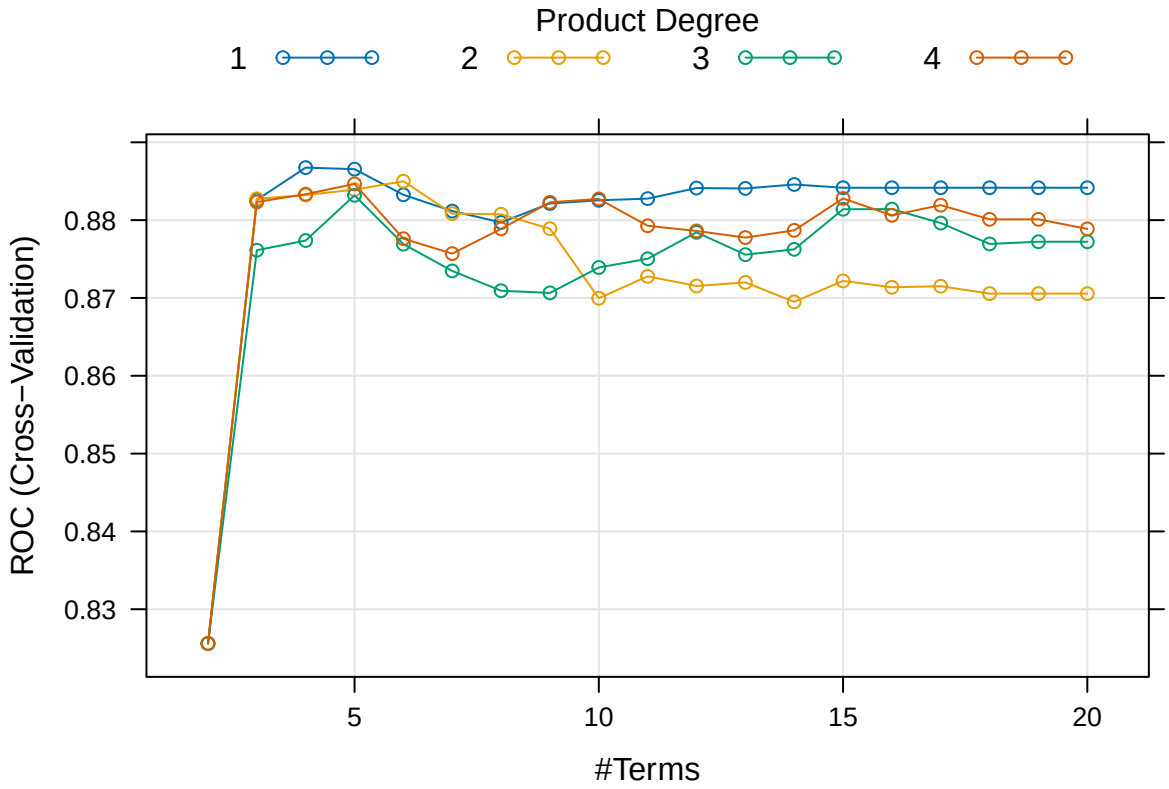
```
## Setting levels: control = 0, case = 1
```

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

```
1-roc.gam$auc[1]
```

```
## [1] 0.102792
```

```
set.seed(83)
model.mars <- train(x = x_train,
                    y = y_train,
                    method = "earth",
                    tuneGrid = expand.grid(degree = 1:4,
                                           nprune = 2:20),
                    metric = "ROC",
                    trControl = ctrl)
plot(model.mars)
```



MARS

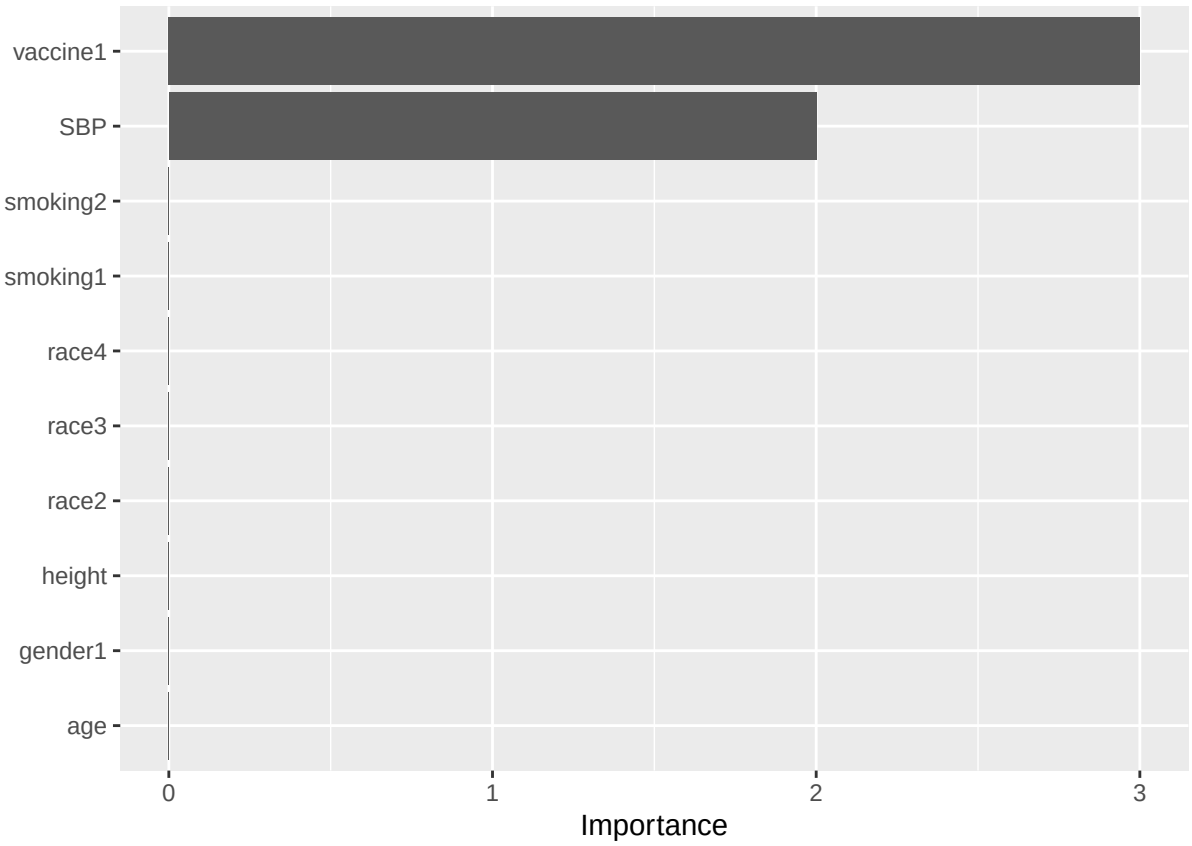
```
model.mars$bestTune
```

```
##   nprune degree
## 3      4      1
```

```
coef(model.mars$finalModel)
```

```
##  (Intercept)     vaccine1   h(SBP-139)   h(139-SBP)
##  2.193120089 -3.323844997 -0.002886632 -0.129783879
```

```
#pdp::partial(model.mars, pred.var = c("age"), grid.resolution = 200) %>% autoplot()

vip(model.mars$finalModel, type = "nsubsets")
```




```
# test error
glm.pred <- predict(model.glm, newdata = x_test, type = "prob")[,2]
glmn.pred <- predict(model.glmn, newdata = x_test, type = "prob")[,2]
gam.pred <- predict(model.gam, newdata = x_test, type = "prob")[,2]
mars.pred <- predict(model.mars, newdata = x_test, type = "prob")[,2]

roc.glm <- roc(y_test, glm.pred)
```

logistic regression test performance

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

## Setting direction: controls < cases
```

```
roc.glmn <- roc(y_test, glmn.pred)
```

```
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
```

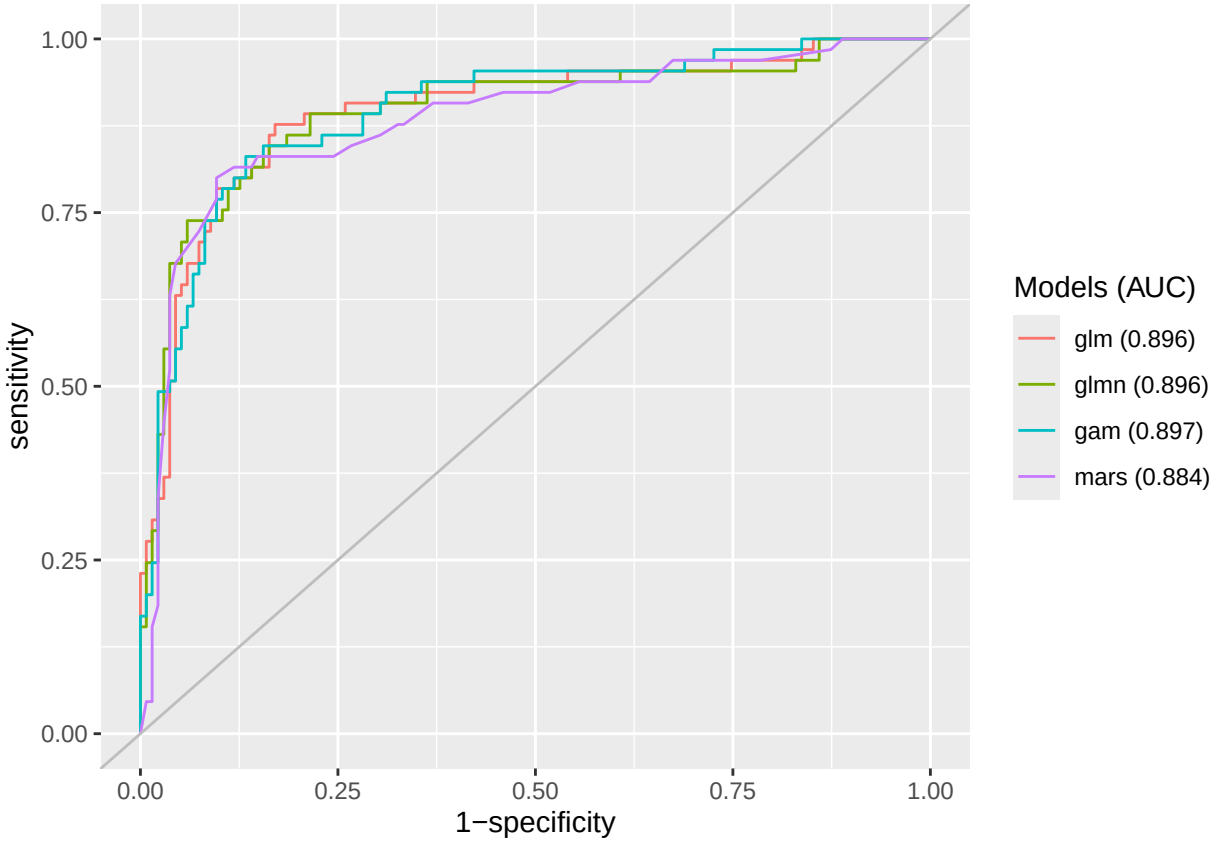
```
roc.gam <- roc(y_test, gam.pred)
```

```
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
```

```
roc.mars <- roc(y_test, mars.pred)
```

```
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
```

```
auc <- c(roc.glm$auc[1], roc.glmn$auc[1],
         roc.gam$auc[1], roc.mars$auc[1])

modelNames <- c("glm","glmn","gam","mars")

ggroc(list(roc.glm, roc.glmn, roc.gam, roc.mars), legacy.axes = TRUE) +
  scale_color_discrete(labels = paste0(modelNames, " (", round(auc,3),")"),
                       name = "Models (AUC)") +
  geom_abline(intercept = 0, slope = 1, color = "grey")
```



LDA

```
set.seed(83)
model.lda <- train(x = x_train,
                   y = y_train,
                   method = "lda",
                   metric = "ROC",
                   trControl = ctrl)
```

```r
lda.pred2 <- predict(model.lda, newdata = x_test, type = "prob")
head(lda.pred2)
```

```
##            X0         X1
## 5  0.97224793 0.02775207
## 7  0.23742088 0.76257912
## 8  0.47116613 0.52883387
## 16 0.03673728 0.96326272
## 20 0.97430555 0.02569445
## 23 0.17362395 0.82637605
```

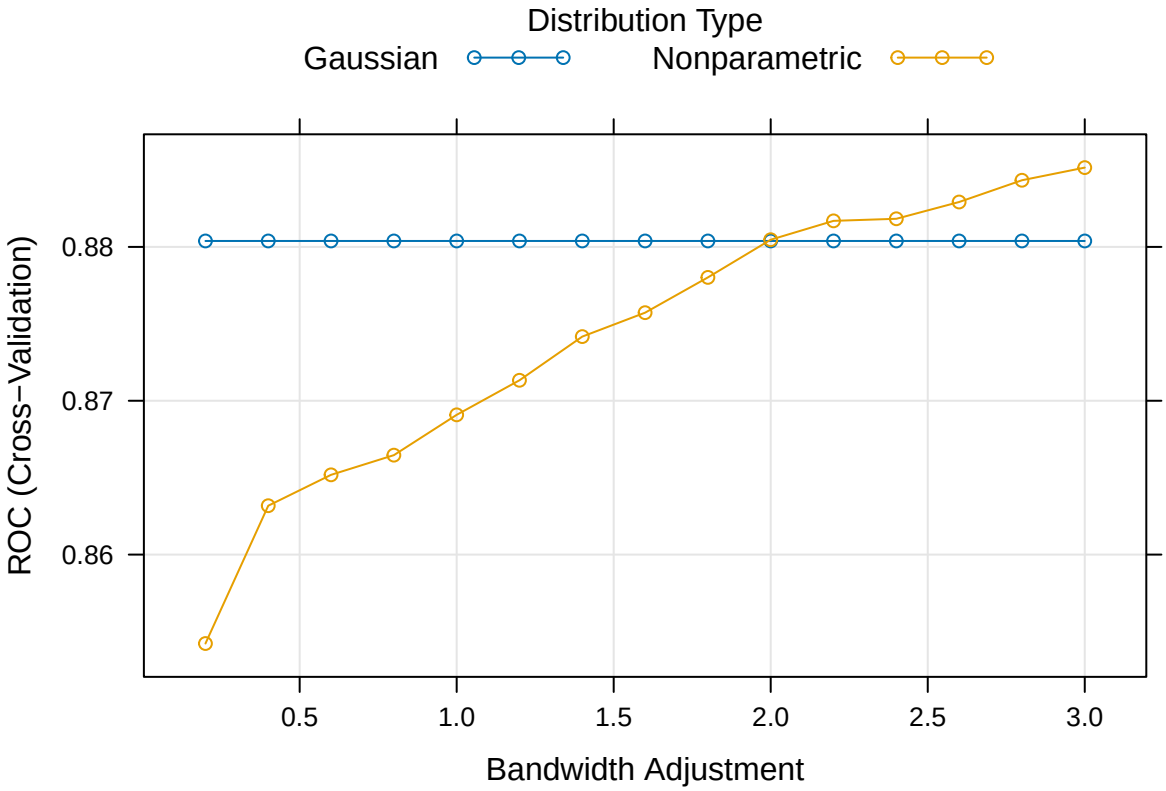
QDA

```r
set.seed(83)
model.qda <- train(x = x_train,
                   y = y_train,
                   method = "qda",
                   metric = "ROC",
                   trControl = ctrl)
qda.pred2 <- predict(model.qda, newdata = x_test, type = "prob")
head(qda.pred2)
```

```
##            X0         X1
## 5  0.98826264 0.01173736
## 7  0.11045382 0.88954618
## 8  0.43666109 0.56333891
## 16 0.02962259 0.97037741
## 20 0.97869427 0.02130573
## 23 0.06623623 0.93376377
```

naive bayers NB

```r
nbGrid <- expand.grid(usekernel = c(FALSE, TRUE), fL = 1,
                      adjust = seq(.2, 3, by = .2))
set.seed(83)
model.nb <- train(x = x_train,
                  y = y_train,
                  method = "nb",
                  tuneGrid = nbGrid,
                  metric = "ROC",
                  trControl = ctrl)
plot(model.nb)
```




```
# test performance
roc.lda <- roc(y_test, lda.pred2[,2])
```

lda qda test perforamce

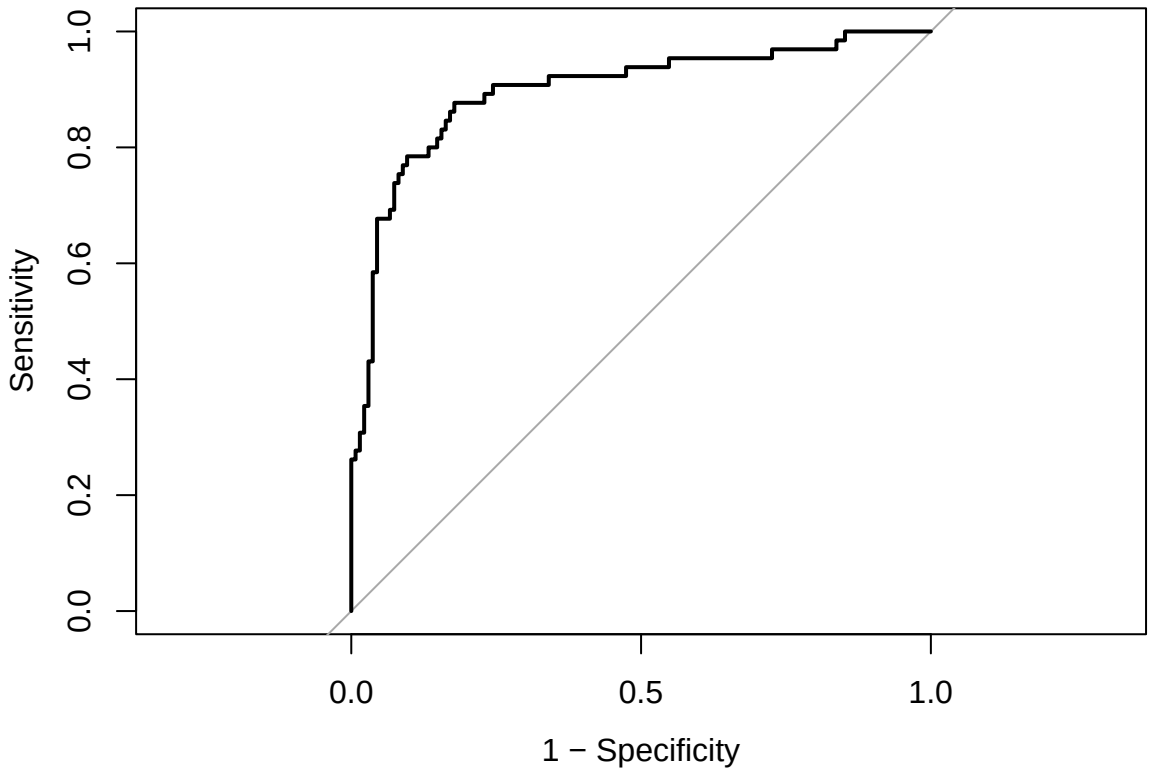
```
## Setting levels: control = 0, case = 1
```

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

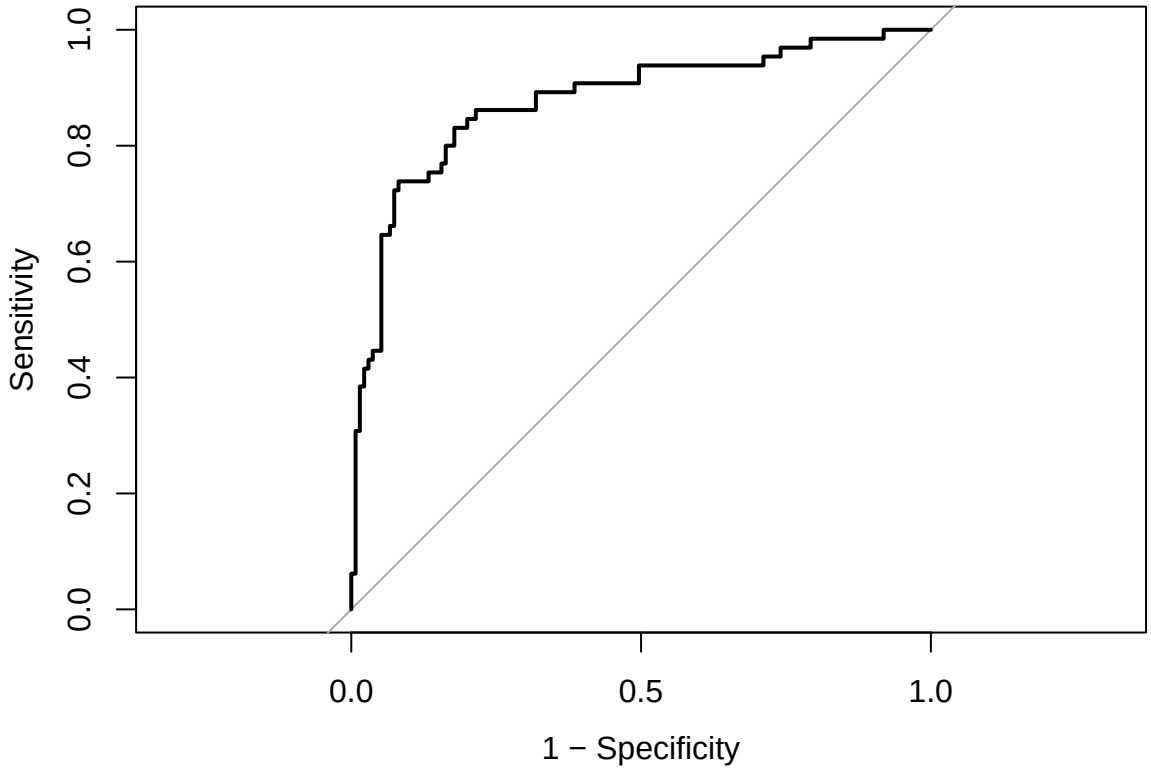
```
roc.qda <- roc(y_test, qda.pred2[,2])
```

```
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
```

```
plot(roc.lda, legacy.axes = TRUE)
```

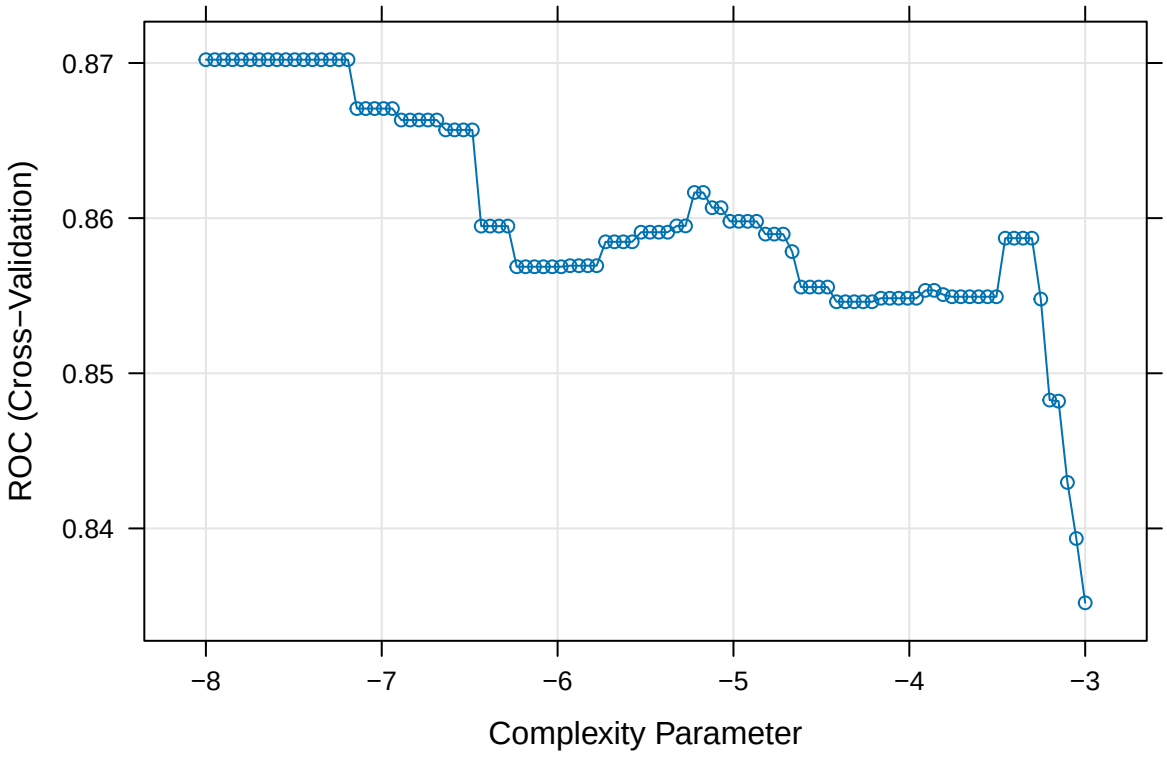



```
plot(roc.qda, legacy.axes = TRUE)
```



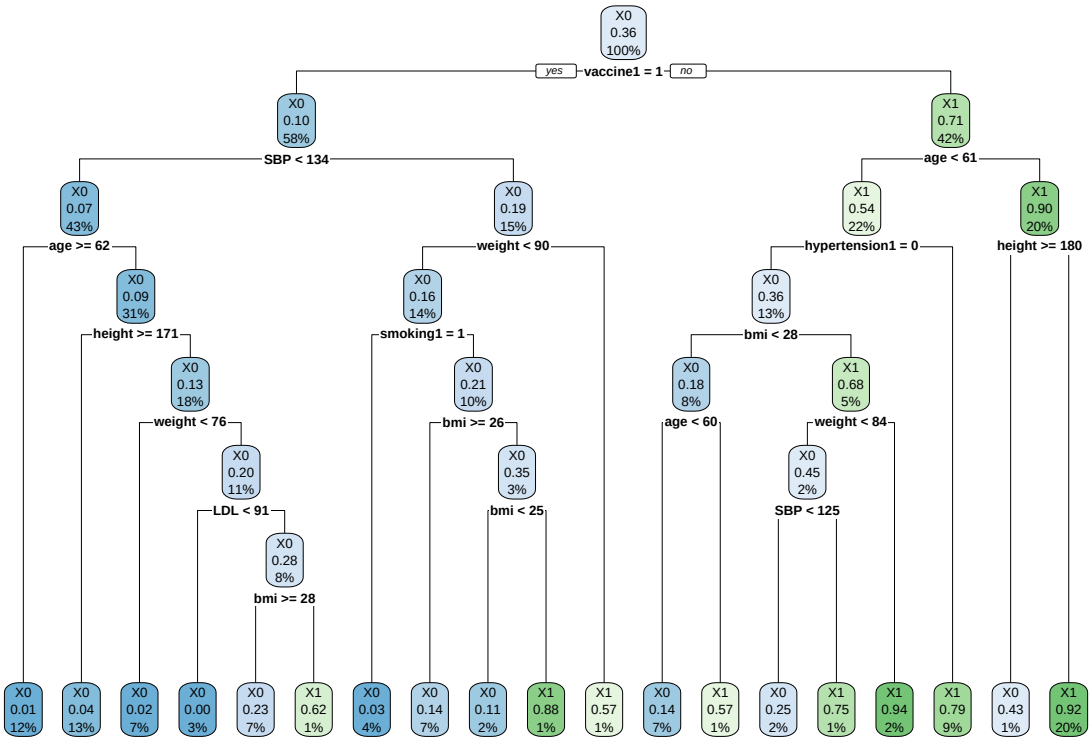
classification tree

```
#ctrl2 <- trainControl(method = "cv",
#                       summaryFunction = twoClassSummary,
#                       classProbs = TRUE)

levels(train.raw$severity) <- make.names(levels(train.raw$severity))
set.seed(83)
rpart.fit <- train(severity ~ . ,
                   train.raw,
                   method = "rpart",
                   tuneGrid = data.frame(cp = exp(seq(-8,-3, len = 100))),
                   trControl = ctrl,
                   metric = "ROC")
plot(rpart.fit, xTrans = log)
```

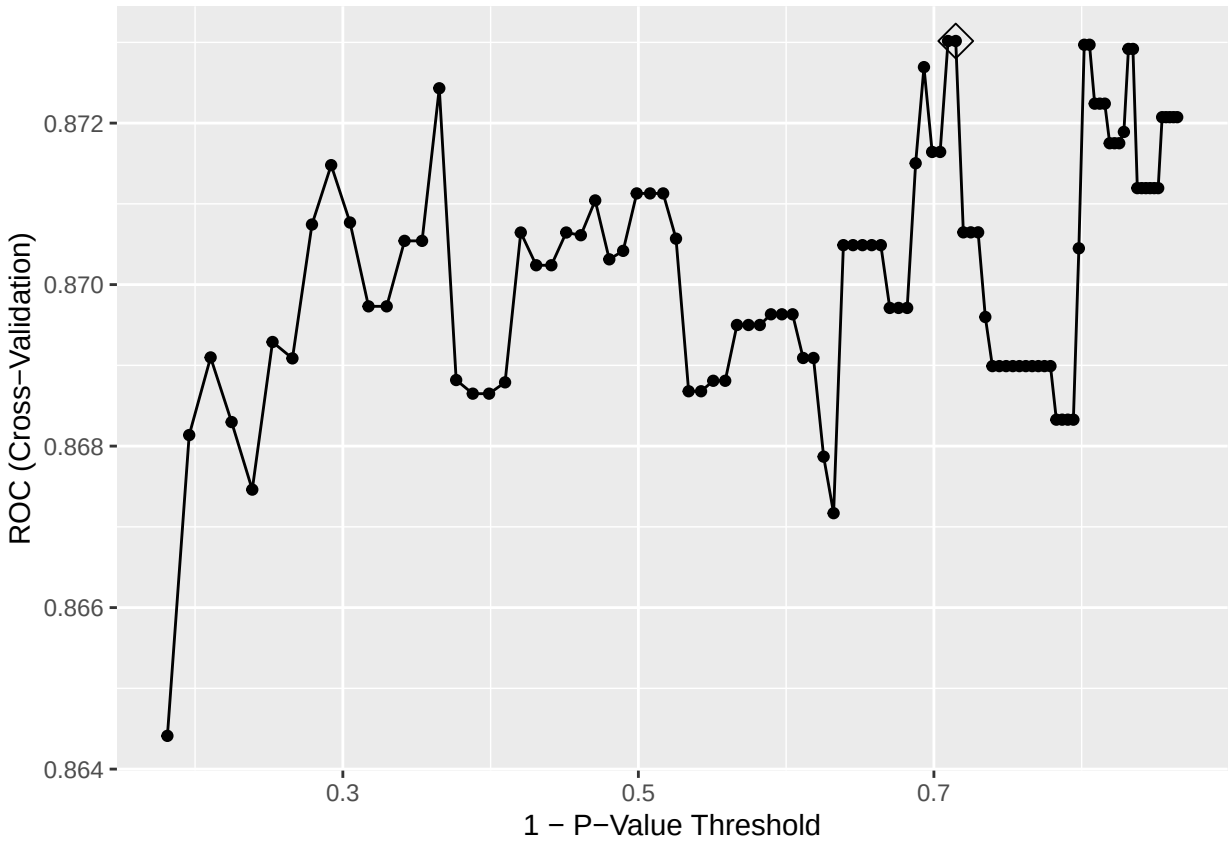


rpart

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

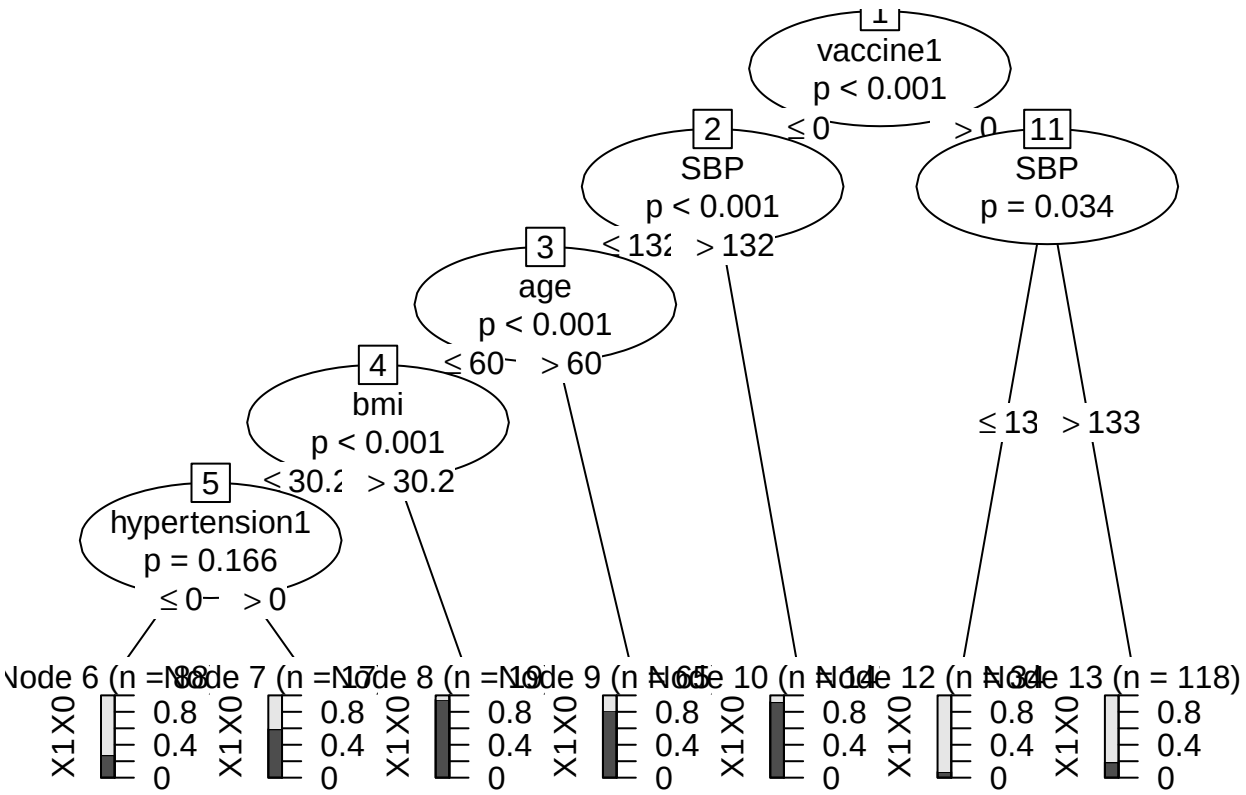



```r
# CIT
set.seed(83)
ctree.fit <- train(severity ~ . ,
                   train.raw,
                   method = "ctree",
                   tuneGrid = data.frame(mincriterion = 1-exp(seq(-2, -0.2, length = 100))),
                   metric = "ROC",
                   trControl = ctrl)
ggplot(ctree.fit, highlight = TRUE)
```



ctree

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



```
ctree.fit$bestTune
```

```
##    mincriterion
## 59    0.7147945
```

```
summary(resamples(list(rpart.fit, ctree.fit)))
```

classification tree test performance

```
##
## Call:
## summary.resamples(object = resamples(list(rpart.fit, ctree.fit)))
##
## Models: Model1, Model2
## Number of resamples: 10
##
## ROC
##             Min.   1st Qu.    Median      Mean   3rd Qu.      Max. NA's
## Model1 0.7990196 0.8544806 0.8699532 0.8702149 0.8843845 0.9240716    0
## Model2 0.7650560 0.8563843 0.8715717 0.8730165 0.8932019 0.9512599    0
##
## Sens
##             Min.   1st Qu.    Median      Mean   3rd Qu.      Max. NA's
## Model1 0.8039216 0.8486991 0.9038462 0.8909502 0.9411765 0.9607843    0
## Model2 0.8039216 0.8829186 0.9125189 0.9045626 0.9366516 0.9803922    0
##
## Spec
##             Min.   1st Qu.    Median      Mean   3rd Qu.      Max. NA's
## Model1 0.4642857 0.6813424 0.7019704 0.7051724 0.7844828 0.8275862    0
## Model2 0.5357143 0.7142857 0.7586207 0.7399015 0.8189655 0.8620690    0
```

```
rpart.pred2 <- predict(rpart.fit, newdata = test.raw,
                       type = "prob")[,1]
ctree.pred <- predict(ctree.fit, newdata = test.raw, type = "prob")[,1]
roc.rpart <- roc(y_test, rpart.pred2)
```

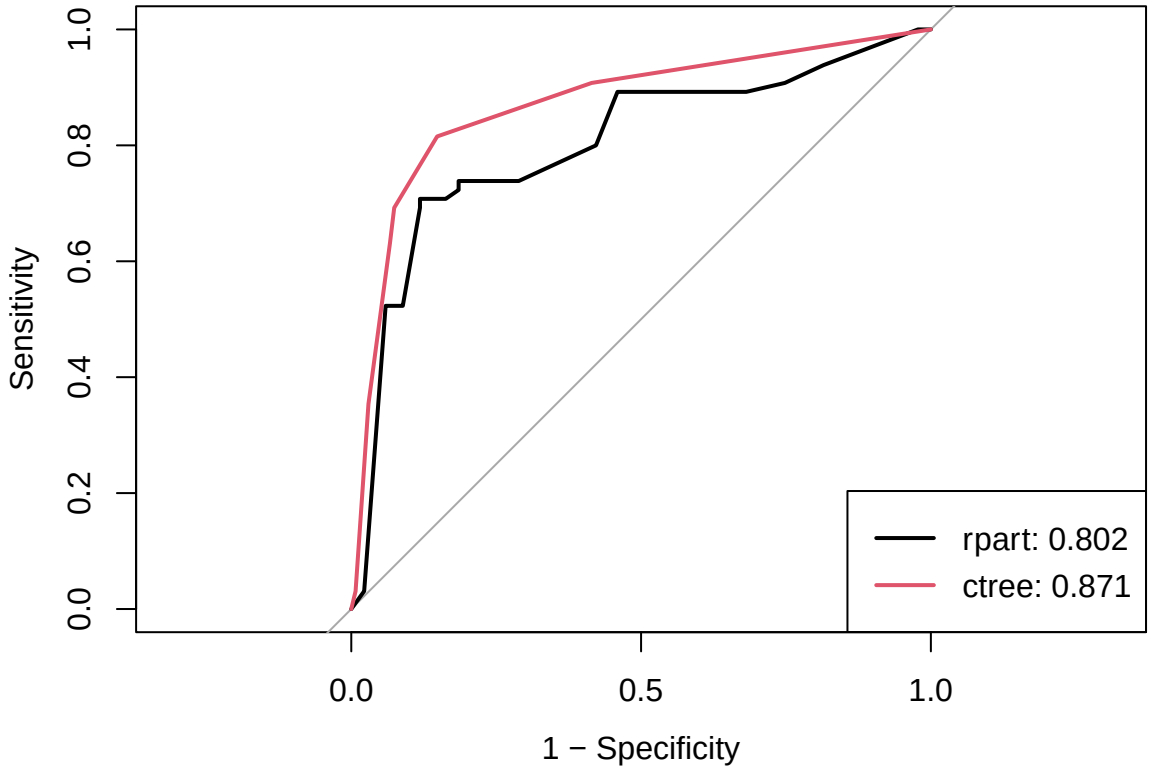
```
## Setting levels: control = 0, case = 1
```

```
## Setting direction: controls > cases
```

```
roc.ctree <- roc(y_test, ctree.pred)
```

```
## Setting levels: control = 0, case = 1
## Setting direction: controls > cases
```

```r
auc <- c(roc.rpart$auc[1], roc.ctree$auc[1])
plot(roc.rpart, legacy.axes = TRUE)
plot(roc.ctree, col = 2, add = TRUE)
modelNames <- c("rpart","ctree")
legend("bottomright", legend = paste0(modelNames, ": ", round(auc,3)),
       col = 1:2, lwd = 2)
```



random forest

```r
rf.grid <- expand.grid(mtry = 1:8, splitrule = "gini",
                       min.node.size = seq(from = 2, to = 16, by = 2))

set.seed(83)
rf.fit <- train(severity ~ . ,
                data = train.raw,
                method = "ranger",
                tuneGrid = rf.grid,
                trControl = ctrl)
```

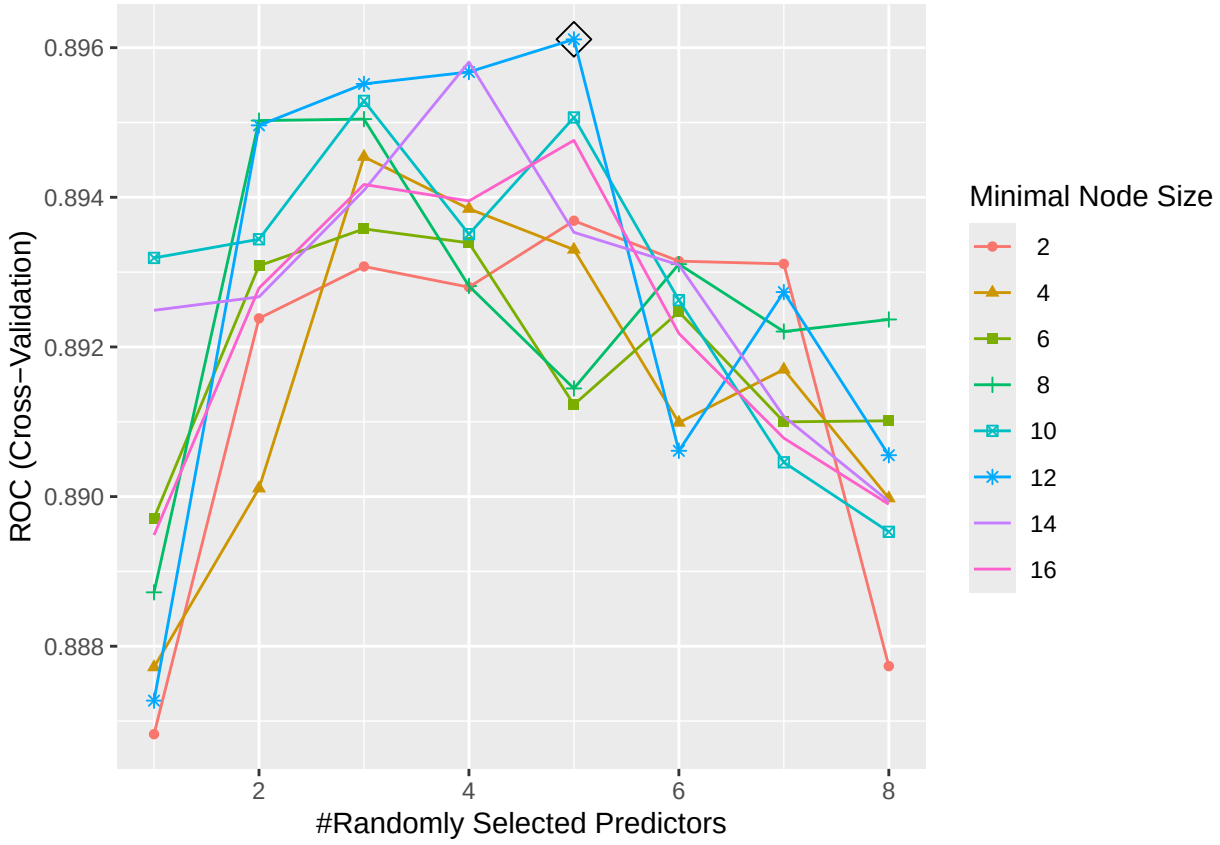
ranger

```
## Warning in train.default(x, y, weights = w, ...): The metric "Accuracy" was not
## in the result set. ROC will be used instead.
```

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

```
## Warning: The shape palette can deal with a maximum of 6 discrete values because more
## than 6 becomes difficult to discriminate
## i you have requested 8 values. Consider specifying shapes manually if you need
##   that many have them.
```

```
## Warning: Removed 16 rows containing missing values or values outside the scale range
## (`geom_point()`).
```




```
rf.fit$bestTune
```

```
##    mtry splitrule min.node.size
## 38    5      gini            12
```

```
# test
rf.pred <- predict(rf.fit, newdata = test.raw, type = "prob")[,1]
roc.rf <- roc(test.raw$severity, rf.pred)
```

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

```
## Setting direction: controls > cases
```

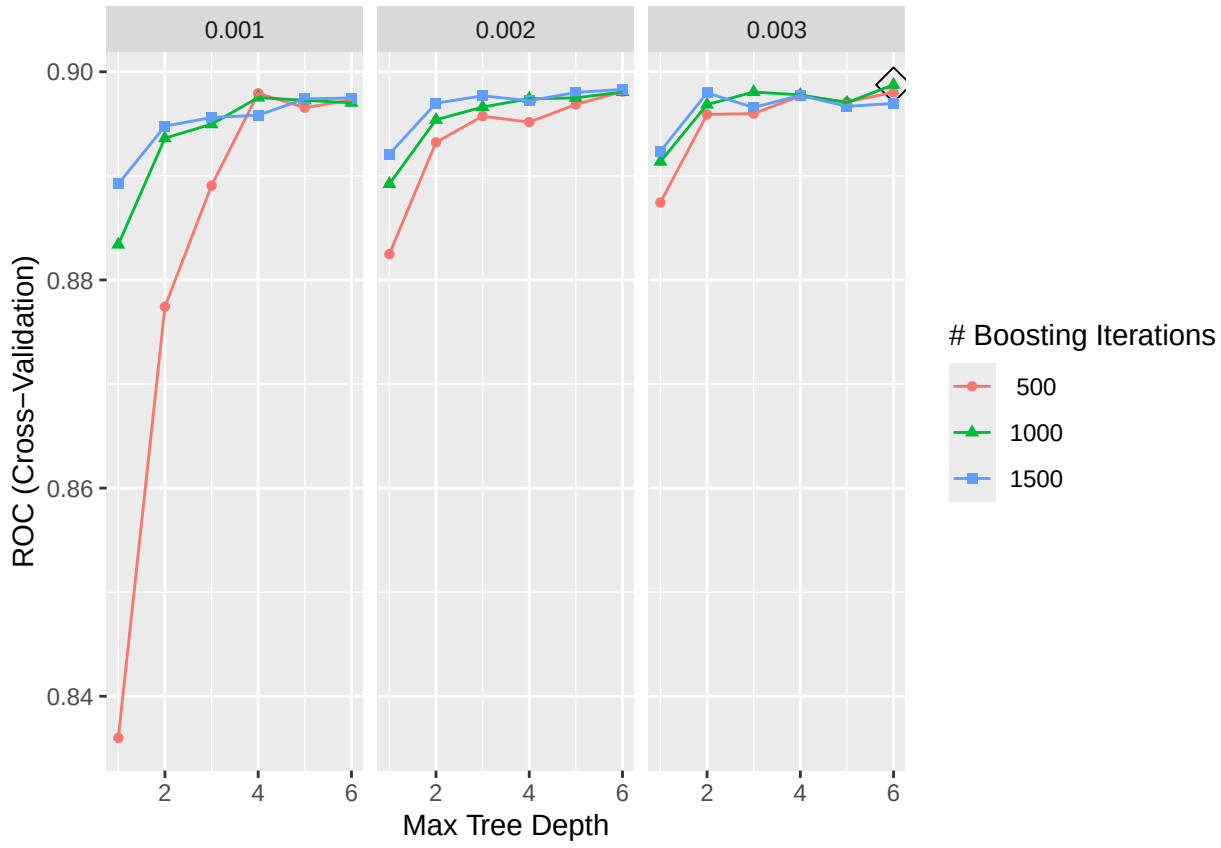
```
roc.rf
```

```
##
## Call:
## roc.default(response = test.raw$severity, predictor = rf.pred)
##
## Data: rf.pred in 135 controls (test.raw$severity 0) > 65 cases (test.raw$severity 1).
## Area under the curve: 0.8683
```

```
1-roc.rf$auc[1]
```

```
## [1] 0.1317379
```

```
gbmA.grid <- expand.grid(n.trees = c(500, 1000, 1500),
                         interaction.depth = 1:6,
                         shrinkage = c(0.001, 0.002, 0.003),
                         n.minobsinnode = 1)
set.seed(83)
gbmA.fit <- train(severity ~ . ,
                  train.raw,
                  tuneGrid = gbmA.grid,
                  trControl = ctrl,
                  method = "gbm",
                  distribution = "adaboost",
                  metric = "ROC",
                  verbose = FALSE)

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



AdaBoost

```
# test
gbmA.pred <- predict(gbmA.fit, newdata = test.raw, type = "prob")[,1]
roc.gbmA <- roc(test.raw$severity, gbmA.pred)
```

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

```
## Setting direction: controls > cases
```

```
1-roc.gbmA$auc[1]
```

```
## [1] 0.1120228
```

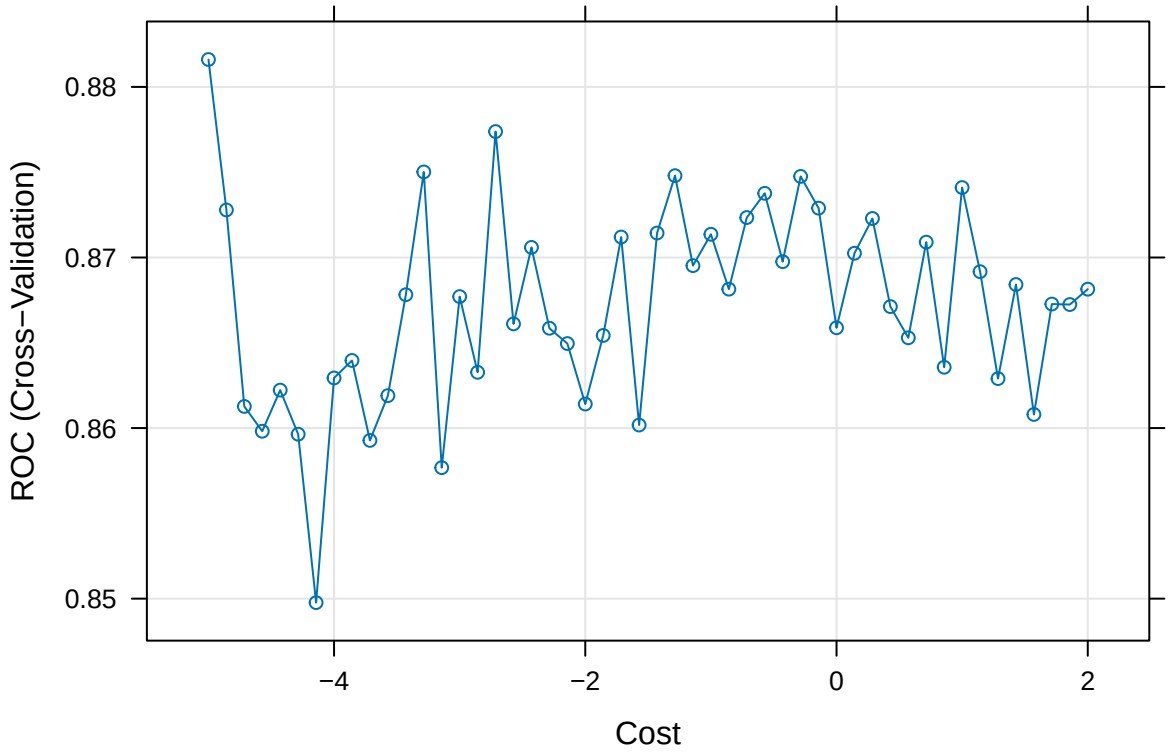
svm

```
# kernlab
set.seed(83)
svml.fit <- train(severity ~ . ,
                  data = train.raw,
                  method = "svmLinear",
                  tuneGrid = data.frame(C = exp(seq(-5, 2, len = 50))),
                  trControl = ctrl)
```

svml

```
## Warning in train.default(x, y, weights = w, ...): The metric "Accuracy" was not
## in the result set. ROC will be used instead.
```

```
plot(svml.fit, highlight = TRUE, xTrans = log)
```



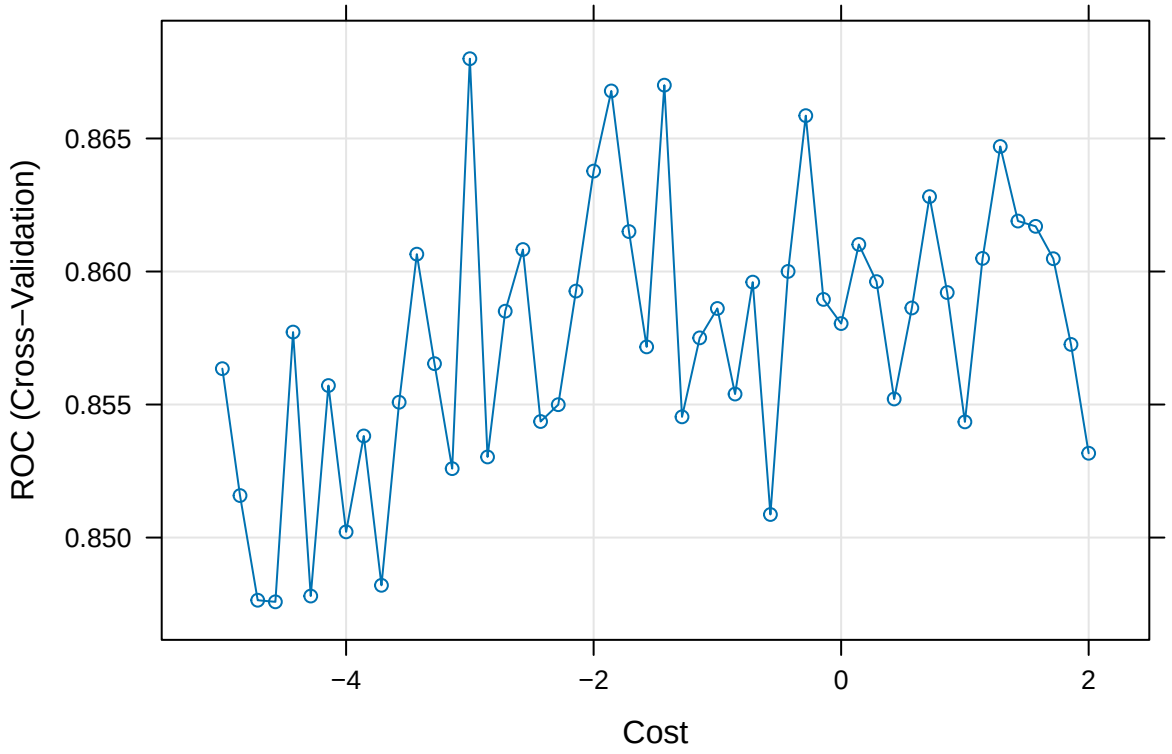

```
svml.fit$bestTune
```

```
##             C
## 1 0.006737947
```

```
# e1071
set.seed(83)
svml.fit2 <- train(severity ~ . ,
                   data = train.raw,
                   method = "svmLinear2",
                   tuneGrid = data.frame(cost = exp(seq(-5, 2, len = 50))),
                   trControl = ctrl)
```

```
## Warning in train.default(x, y, weights = w, ...): The metric "Accuracy" was not
## in the result set. ROC will be used instead.
```

```
plot(svml.fit2, highlight = TRUE, xTrans = log)
```

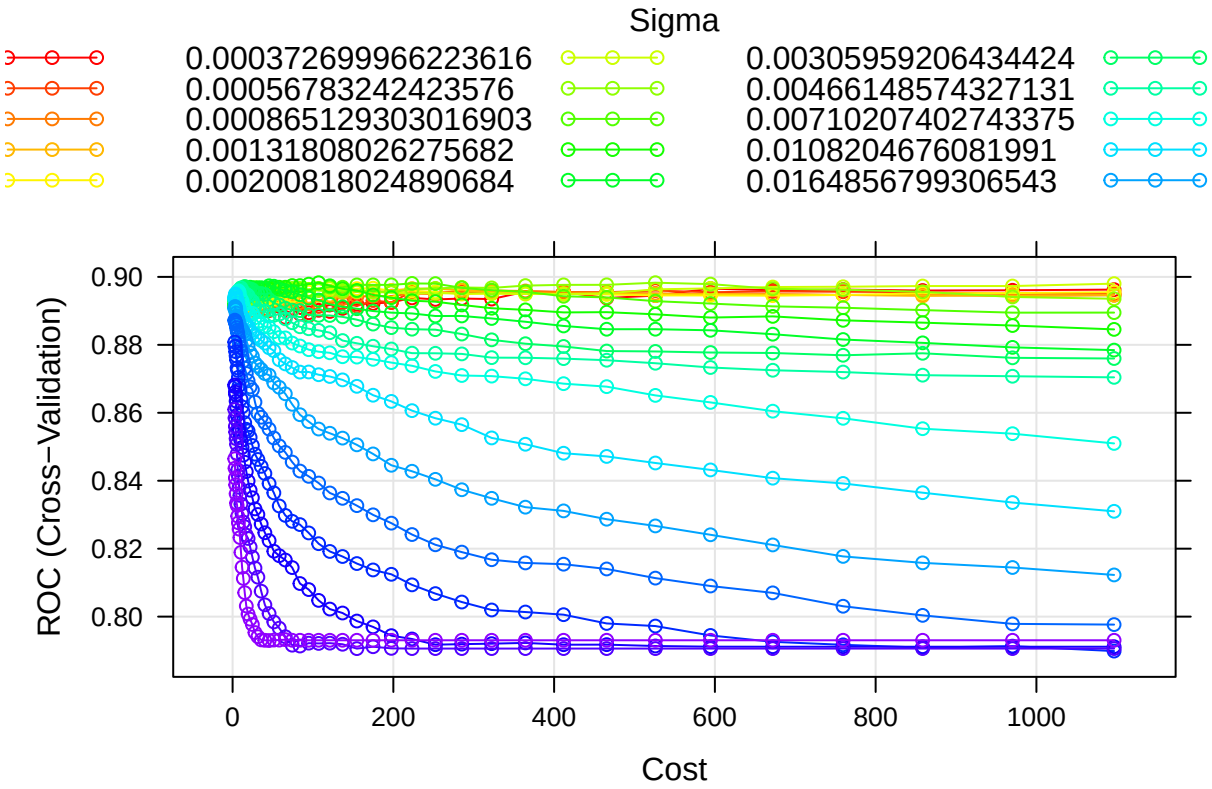



```
svmr.grid <- expand.grid(C = exp(seq(1, 7, len = 50)),
                         sigma = exp(seq(-10, -2, len = 20)))
# tunes over both cost and sigma
set.seed(83)
svmr.fit <- train(severity ~ . , data = train.raw,
                  method = "svmRadialSigma",
                  tuneGrid = svmr.grid,
                  trControl = ctrl)
```

svmr

```
## line search fails -0.6832979 0.3122083 1.048723e-05 -3.011969e-06 -3.814446e-08 1.354287e-08 -4.4082
```

```
myCol <- rainbow(25)
myPar <- list(superpose.symbol = list(col = myCol),
              superpose.line = list(col = myCol))
plot(svmr.fit, highlight = TRUE, par.settings = myPar)
```




```
svmr.fit$bestTune
```

```
##            sigma        C
## 867 0.0005678324 526.0026
```

```
# tune over cost and uses a single value of sigma based on kernlab's sigma function
set.seed(83)
svmr.fit2 <- train(severity ~ . , data = train.raw,
                   method = "svmRadialCost",
                   tuneGrid = data.frame(C = exp(seq(-3, 3, len = 20))),
                   trControl = ctrl)
```

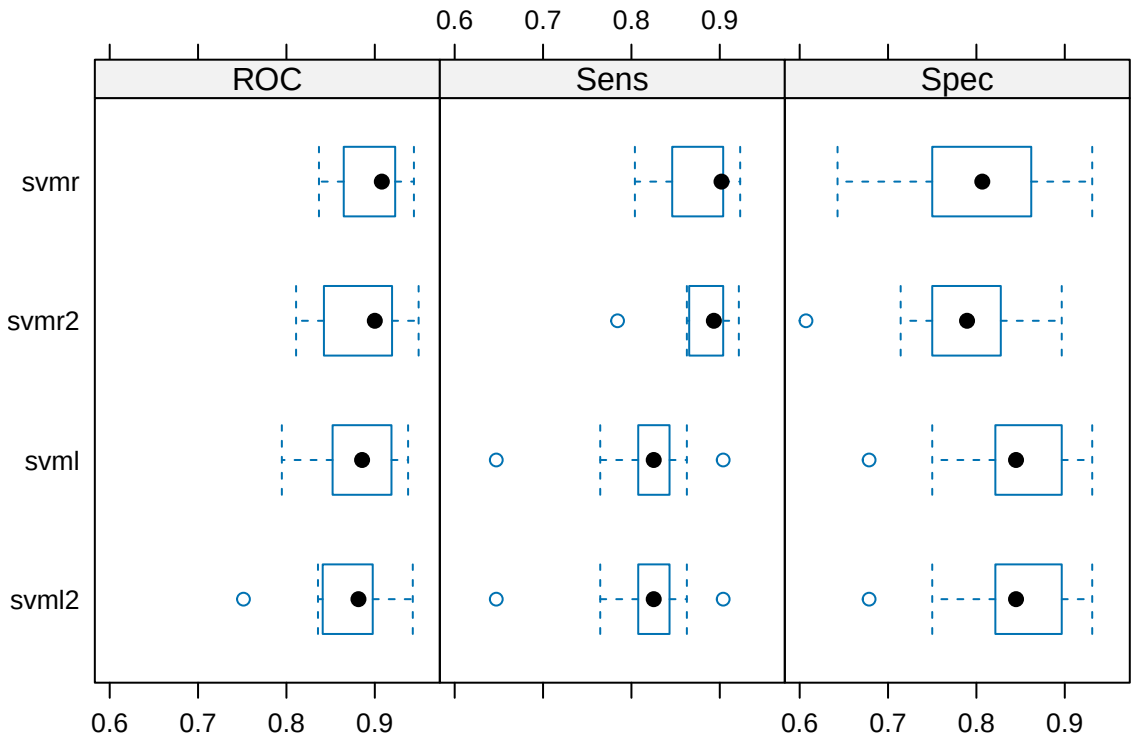
```
resamp <- resamples(list(svmr = svmr.fit, svmr2 = svmr.fit2,
                         svml = svml.fit, svml2 = svml.fit2))
summary(resamp)
```

model comparison

```
##
## Call:
## summary.resamples(object = resamp)
##
## Models: svmr, svmr2, svml, svml2
## Number of resamples: 10
```

```
##
## ROC
##            Min.   1st Qu.    Median      Mean   3rd Qu.      Max. NA's
## svmr  0.8368347 0.8719272 0.9079615 0.8982876 0.9211786 0.9442971    0
## svmr2 0.8109244 0.8503876 0.8999974 0.8863974 0.9173337 0.9496021    0
## svml  0.7948179 0.8566873 0.8856247 0.8816084 0.9149763 0.9376658    0
## svml2 0.7514006 0.8419118 0.8815013 0.8679987 0.8968795 0.9429708    0
##
## Sens
##            Min.   1st Qu.    Median      Mean   3rd Qu.      Max. NA's
## svmr  0.8039216 0.8552036 0.9019608 0.8812217 0.9033748 0.9230769    0
## svmr2 0.7843137 0.8696267 0.8932881 0.8812594 0.9033748 0.9215686    0
## svml  0.6470588 0.8116516 0.8252262 0.8130090 0.8390837 0.9038462    0
## svml2 0.6470588 0.8116516 0.8252262 0.8130090 0.8390837 0.9038462    0
##
## Spec
##            Min.   1st Qu.    Median      Mean   3rd Qu.      Max. NA's
## svmr  0.6428571 0.7521552 0.8066502 0.8032020 0.8534483 0.9310345    0
## svmr2 0.6071429 0.7521552 0.7894089 0.7788177 0.8189655 0.8965517    0
## svml  0.6785714 0.8214286 0.8448276 0.8381773 0.8879310 0.9310345    0
## svml2 0.6785714 0.8214286 0.8448276 0.8381773 0.8879310 0.9310345    0
```

```r
bwplot(resamp)
```




```r
# test error
pred.svml <- predict(svml.fit, newdata = test.raw)
pred.svmr <- predict(svmr.fit, newdata = test.raw)
levels(pred.svml) <- levels(test.raw$severity)
confusionMatrix(data = pred.svml, reference = test.raw$severity)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 115  12
##          1  20  53
##
##                Accuracy : 0.84
##                  95% CI : (0.7817, 0.8879)
##     No Information Rate : 0.675
##     P-Value [Acc > NIR] : 9.736e-08
##
##                   Kappa : 0.6466
##
##  Mcnemar's Test P-Value : 0.2159
##
##             Sensitivity : 0.8519
##             Specificity : 0.8154
##          Pos Pred Value : 0.9055
##          Neg Pred Value : 0.7260
##              Prevalence : 0.6750
##          Detection Rate : 0.5750
##    Detection Prevalence : 0.6350
##       Balanced Accuracy : 0.8336
##
##        'Positive' Class : 0
##
```

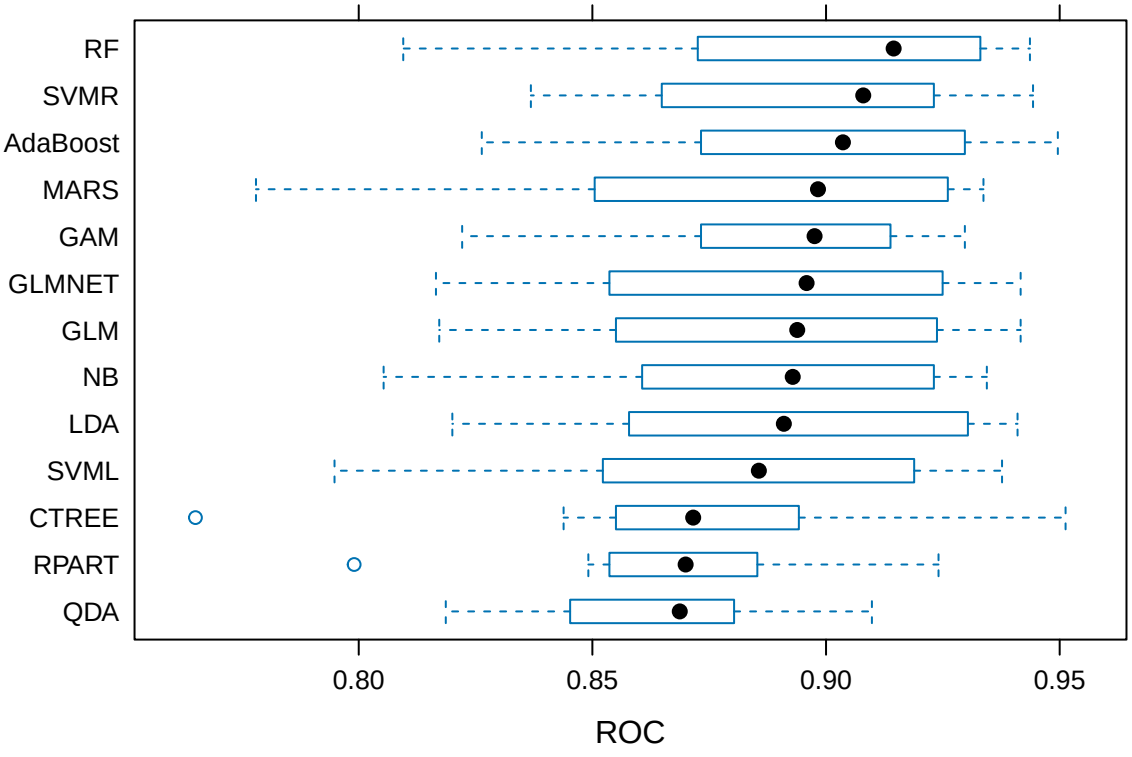
```
levels(pred.svmr) <- levels(test_data$severity)
confusionMatrix(data = pred.svmr, reference = test.raw$severity)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 120  16
##          1  15  49
##
##                Accuracy : 0.845
##                  95% CI : (0.7873, 0.8922)
##     No Information Rate : 0.675
##     P-Value [Acc > NIR] : 3.744e-08
##
##                   Kappa : 0.6453
##
##  Mcnemar's Test P-Value : 1
##
##             Sensitivity : 0.8889
##             Specificity : 0.7538
##          Pos Pred Value : 0.8824
##          Neg Pred Value : 0.7656
##              Prevalence : 0.6750
##          Detection Rate : 0.6000
##    Detection Prevalence : 0.6800
```

```
##        Balanced Accuracy : 0.8214
##
##         'Positive' Class : 0
##
```

total model comparison

```
model_list <- list( GLM = model.glm,
                    GLMNET = model.glmn,
                    GAM = model.gam,
                    MARS = model.mars,
                    LDA = model.lda,
                    QDA = model.qda,
                    NB = model.nb,
                    RPART = rpart.fit,
                    CTREE = ctree.fit,
                    RF = rf.fit,
                    AdaBoost = gbmA.fit,
                    SVML = svml.fit,
                    SVMR = svmr.fit
                    )
res <- resamples(model_list)
bwplot(res, metric = "ROC")
```



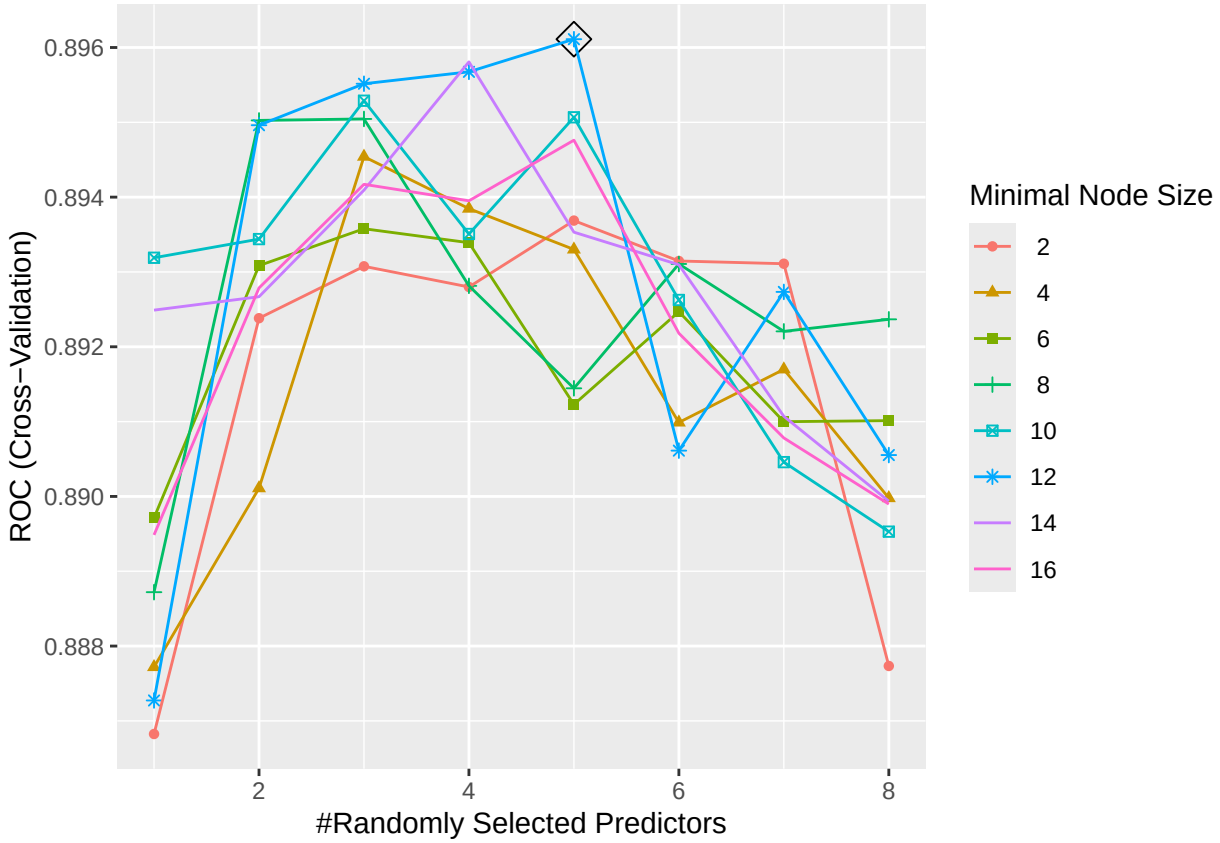
Final model

RF

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

```
## Warning: The shape palette can deal with a maximum of 6 discrete values because more
## than 6 becomes difficult to discriminate
## i you have requested 8 values. Consider specifying shapes manually if you need
##   that many have them.
```

```
## Warning: Removed 16 rows containing missing values or values outside the scale range
## (`geom_point()`).
```




```
rf.fit$bestTune
```

```
##    mtry splitrule min.node.size
## 38    5      gini            12
```

```
# test
rf.pred <- predict(rf.fit, newdata = test.raw, type = "prob")[,1]
roc.rf <- roc(test.raw$severity, rf.pred)
```

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

```
## Setting direction: controls > cases
```

```
roc.rf
```

```
##
## Call:
## roc.default(response = test.raw$severity, predictor = rf.pred)
##
## Data: rf.pred in 135 controls (test.raw$severity 0) > 65 cases (test.raw$severity 1).
## Area under the curve: 0.8683
```

```
1-roc.rf$auc[1]
```

```
## [1] 0.1317379
```

SVMR

```
# variable importance
svmr_importance <- varImp(svmr.fit, scale = TRUE)
plot(svmr_importance)
```