Supervised

2019-5-15

```r
heart_disease = read_csv("..\\data\\heart.csv") %>%
    mutate(target = ifelse(target==1, 0, 1)) %>%
    mutate(target=as.factor(target)) %>%
    mutate(target=as.factor(ifelse(target==0, "absence", "presence")))
```

```
## Parsed with column specification:
## cols(
##   age = col_double(),
##   sex = col_double(),
##   cp = col_double(),
##   trestbps = col_double(),
##   chol = col_double(),
##   fbs = col_double(),
##   restecg = col_double(),
##   thalach = col_double(),
##   exang = col_double(),
##   oldpeak = col_double(),
##   slope = col_double(),
##   ca = col_double(),
##   thal = col_double(),
##   target = col_double()
## )
```

```r
# %>%  mutate(target = relevel(target, "absence"))
#  %>%   arrange(-as.numeric(target))
set.seed(1)
#trRows = createDataPartition(heart_disease$target, p = .75, list = FALSE)
#train = heart_disease[trRows,]
#test = heart_disease[-trRows,]

# heart_disease2 = read_csv("..\\data\\heart.csv") %>%
#     mutate(target = ifelse(target==1, 0, 1)) %>%
#     mutate(target=as.factor(heart_disease$target))

heart_disease = heart_disease %>%
    filter(thal != 0) %>%
    mutate(sex=as.factor(sex),
           cp=as.factor(cp),
           fbs=as.factor(fbs),
           restecg=as.factor(restecg),
           exang=as.factor(exang),
           slope=as.factor(slope),
           thal=as.factor(thal))
model.x <- model.matrix(target~.,heart_disease)[,-1]
model.y <- heart_disease$target

# test = test %>%
#     mutate(sex=as.factor(sex),
```

```r
#              cp=as.factor(cp),
#              fbs=as.factor(fbs),
#              restecg=as.factor(restecg),
#              exang=as.factor(exang),
#              slope=as.factor(slope),
#              thal=as.factor(thal))
# test.x <- model.matrix(target~.,test)[,-1]
# test.y <- test$target
```

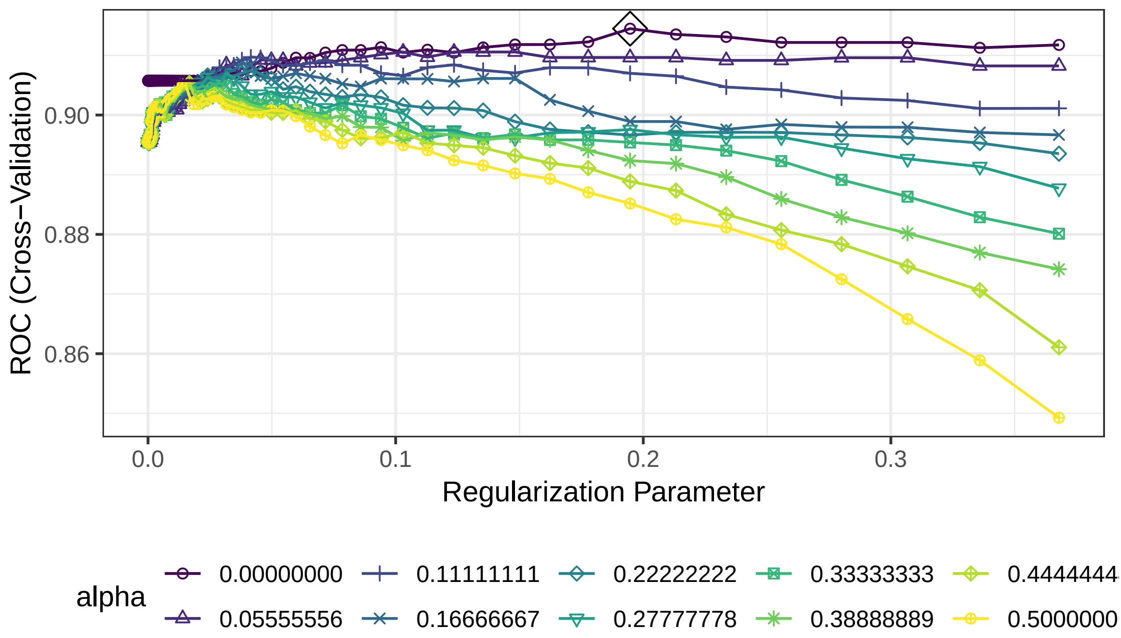
Regularized logistic

```r
ctrl = trainControl(method = "cv",
                    classProbs = TRUE,
                    summaryFunction = twoClassSummary)


glmnGrid <- expand.grid(.alpha = seq(0, 0.5, length = 10),
                        .lambda = exp(seq(-10,-1, length = 100)))
set.seed(1)
model.glm <- train(x = model.x,
                   y = model.y,
                   method = "glmnet",
                   tuneGrid = glmnGrid,
                   metric = "ROC",
                   trControl = ctrl)
```

```r
ggplot(model.glm, highlight = T)  +
    viridis::scale_color_viridis(discrete = TRUE) +
    scale_shape_manual(values = seq(1,10))
```

```
## Scale for 'colour' is already present. Adding another scale for
## 'colour', which will replace the existing scale.
```

```
## Scale for 'shape' is already present. Adding another scale for 'shape',
## which will replace the existing scale.
```




```
model.glm$bestTune
```

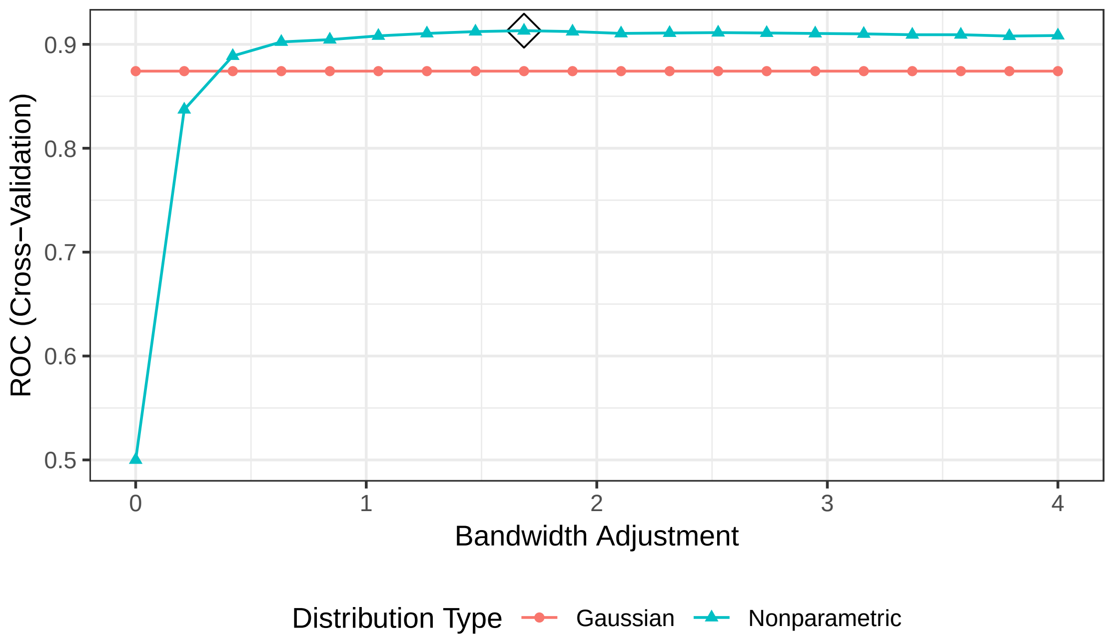
```
##    alpha    lambda
## 93     0 0.1946867
```

LDA

```
set.seed(1)
model.lda = train(x = model.x,
                   y = model.y,
                  method = "lda",
                  metric = "ROC",
                  trControl = ctrl)
```

Naive bayes

```
set.seed(1)
nbGrid = expand.grid(usekernel = c(FALSE,TRUE),
                     fL = 1, adjust = seq(0, 4, length = 20))
model.bayes = train(x = model.x,
                    y = model.y,
                    method = "nb",
                    tuneGrid = nbGrid,
                    metric = "ROC",
                    trControl = ctrl)

ggplot(model.bayes, highlight = T)
```

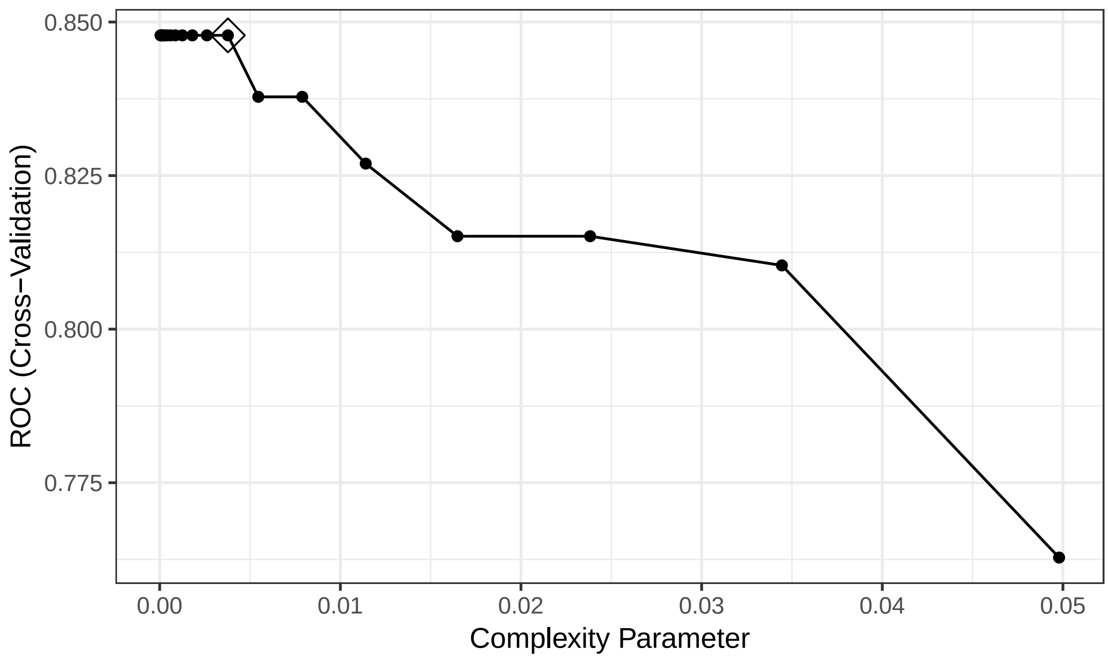



```
model.bayes$bestTune
```

```
##    fL usekernel   adjust
## 29  1      TRUE 1.684211
```

Tree

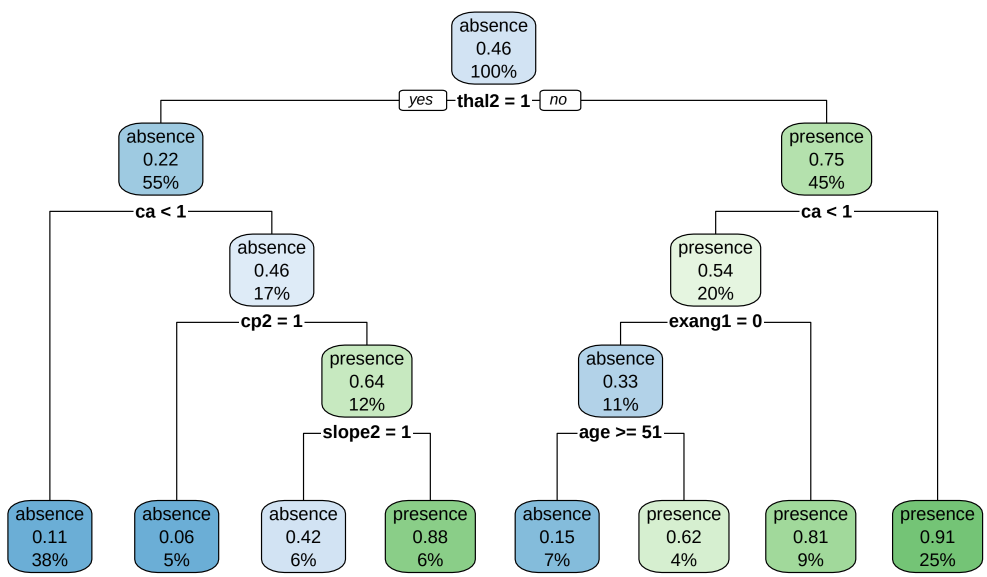
```
set.seed(1)
tree.class <- train(model.x, model.y,
                    method = "rpart",
                    tuneGrid = data.frame(cp = exp(seq(-10,-3, len = 20))),
                    trControl = ctrl,
                    metric = "ROC")
ggplot(tree.class, highlight = TRUE)
```




```
tree.class$bestTune
```

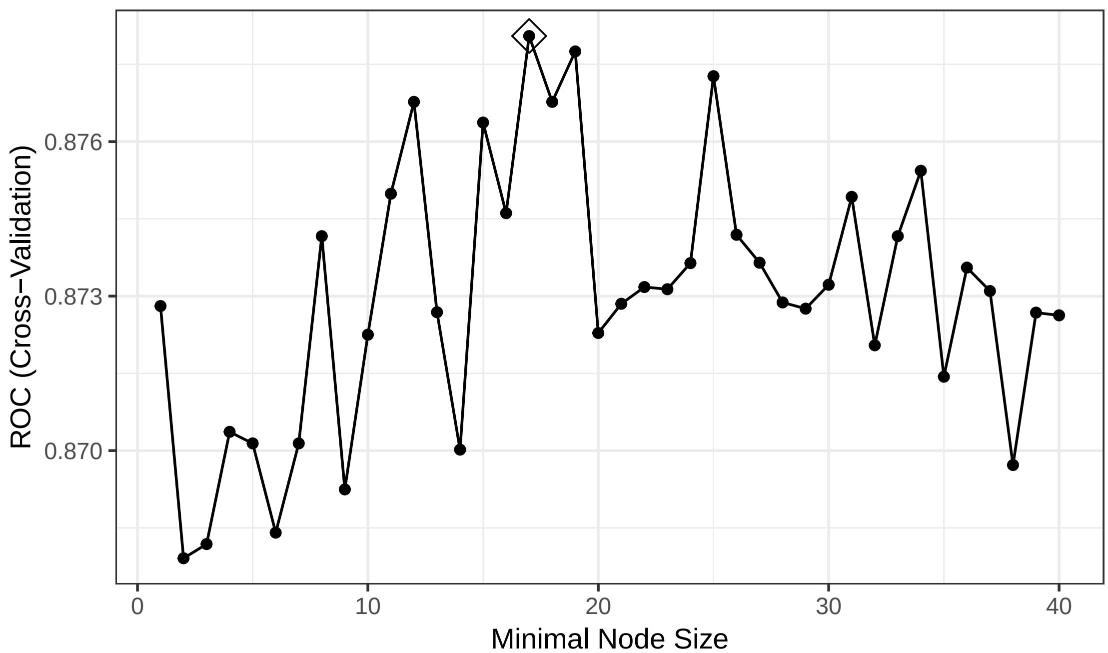
```
##             cp
## 13 0.003776539
```

```
rpart.plot(tree.class$finalModel)
```



Bagging

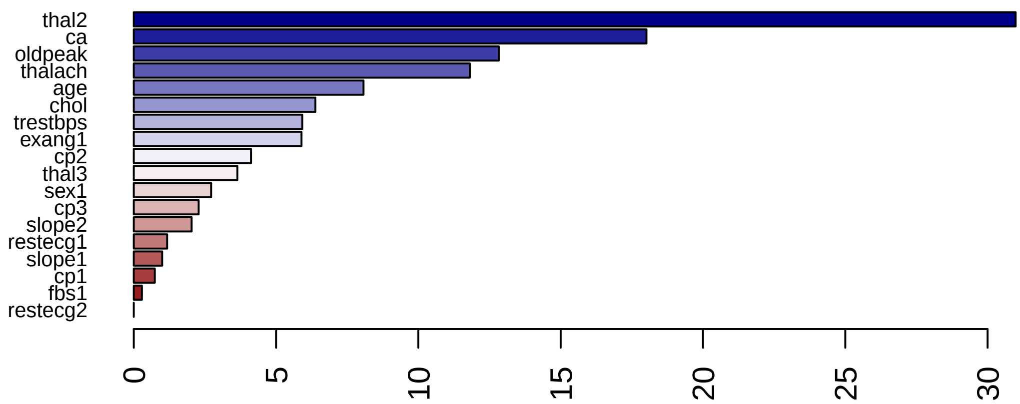
```r
bagging.grid <- expand.grid(mtry = 18,
                            splitrule = "gini",
                            min.node.size = 1:40)

set.seed(1)
bagging.class <- train(model.x, model.y,
                       method = "ranger",
                       tuneGrid = bagging.grid,
                       metric = "ROC",
                       trControl = ctrl,
                       importance = "impurity")
ggplot(bagging.class, highlight = TRUE)
```




```r
bagging.class$bestTune
```

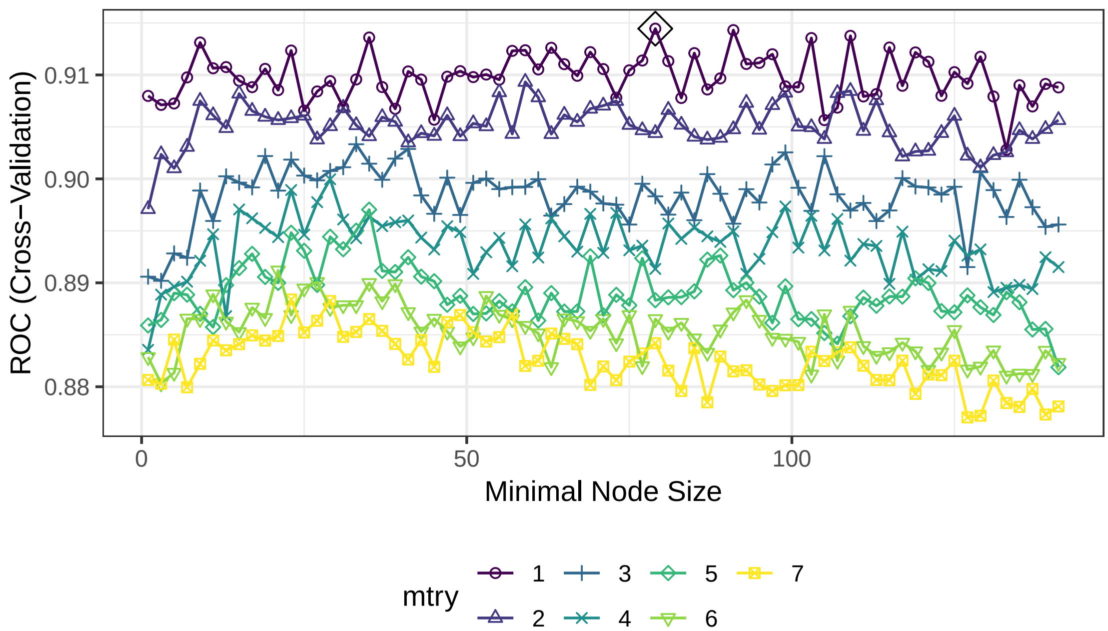
```
##    mtry splitrule min.node.size
## 17   18      gini            17
```

```r
barplot(sort(ranger::importance(bagging.class$finalModel),
             decreasing = FALSE),
las = 2, horiz = TRUE, cex.names = 0.7,
col = colorRampPalette(colors = c("darkred","white","darkblue"))(18))
```

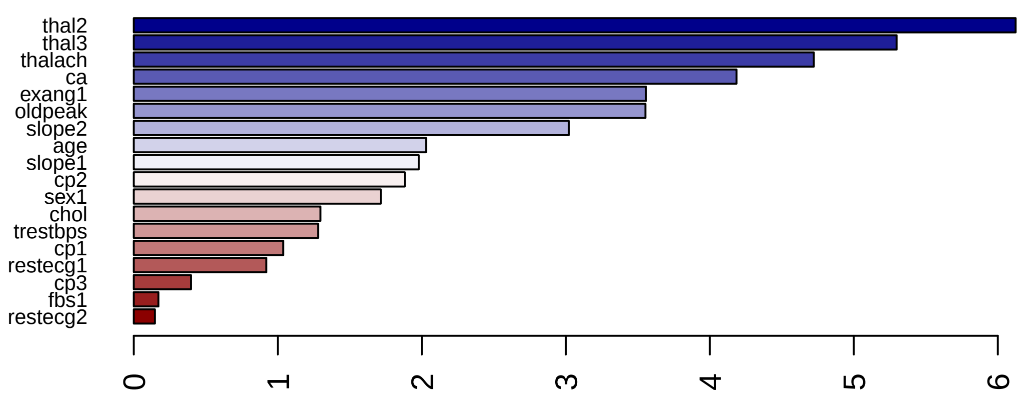


Random Forest

```
rf.grid <- expand.grid(mtry = 1:7,
                       splitrule = "gini",
                       min.node.size = seq(1,141, by = 2))

set.seed(1)
rf.class <- train(model.x, model.y,
                  method = "ranger",
                  tuneGrid = rf.grid,
                  metric = "ROC",
                  trControl = ctrl,
                  importance = "impurity")


ggplot(rf.class, highlight = TRUE) +
    viridis::scale_color_viridis(discrete = TRUE) +
    scale_shape_manual(values = seq(1,7))
```




```
barplot(sort(ranger::importance(rf.class$finalModel), decreasing = FALSE),
las = 2, horiz = TRUE, cex.names = 0.7,
col = colorRampPalette(colors = c("darkred","white","darkblue"))(18))
```



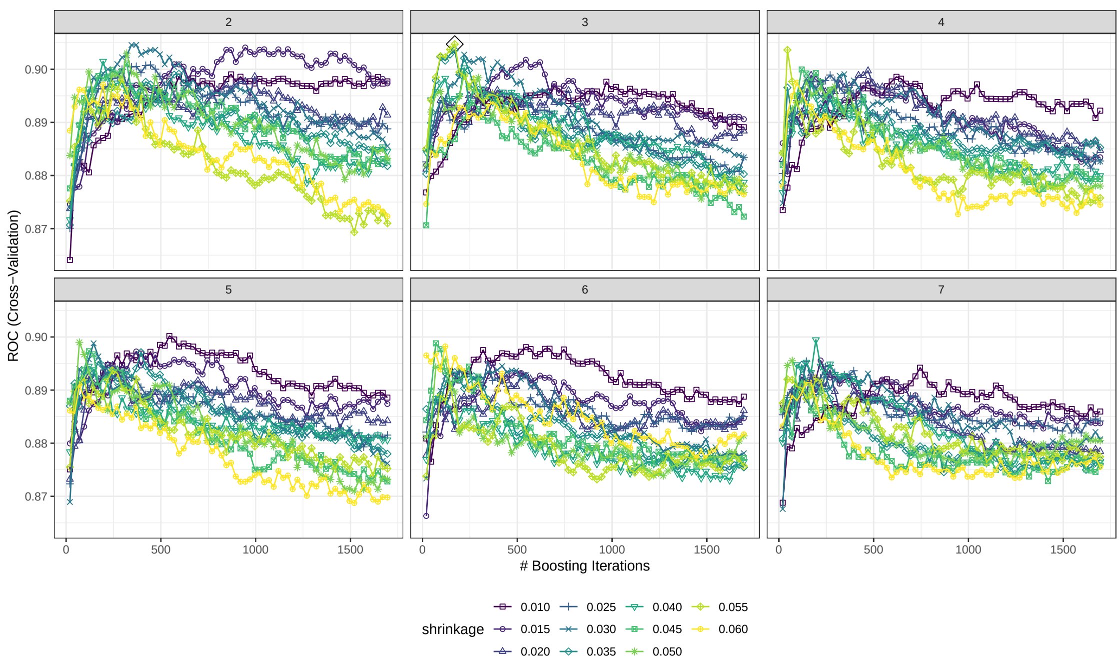

```
a = predict(rf.class, type="raw")
```

Boosting

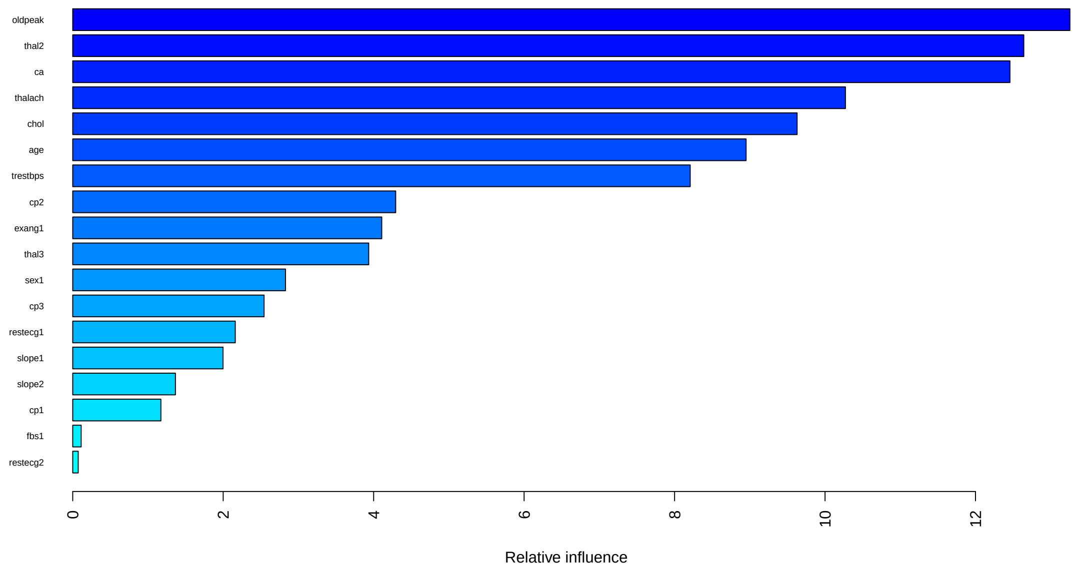
```
boost.grid <- expand.grid(n.trees = seq(20, 1700, by = 25),
                          interaction.depth = 2:7,
                          shrinkage =  seq(0.01, 0.06, by = 0.005),
                          n.minobsinnode = 1)

set.seed(1)
# Adaboost loss function
boost.class = train(model.x, model.y,
                    tuneGrid = boost.grid,
                    trControl = ctrl,
                    method = "gbm",
                    distribution = "adaboost",
                    metric = "ROC",
                    verbose = FALSE)

ggplot(boost.class, highlight = TRUE) +
    viridis::scale_color_viridis(discrete = TRUE) +
    scale_shape_manual(values = seq(0,10))
```

```
## Scale for 'colour' is already present. Adding another scale for
## 'colour', which will replace the existing scale.
```

```
## Scale for 'shape' is already present. Adding another scale for 'shape',
## which will replace the existing scale.
```




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

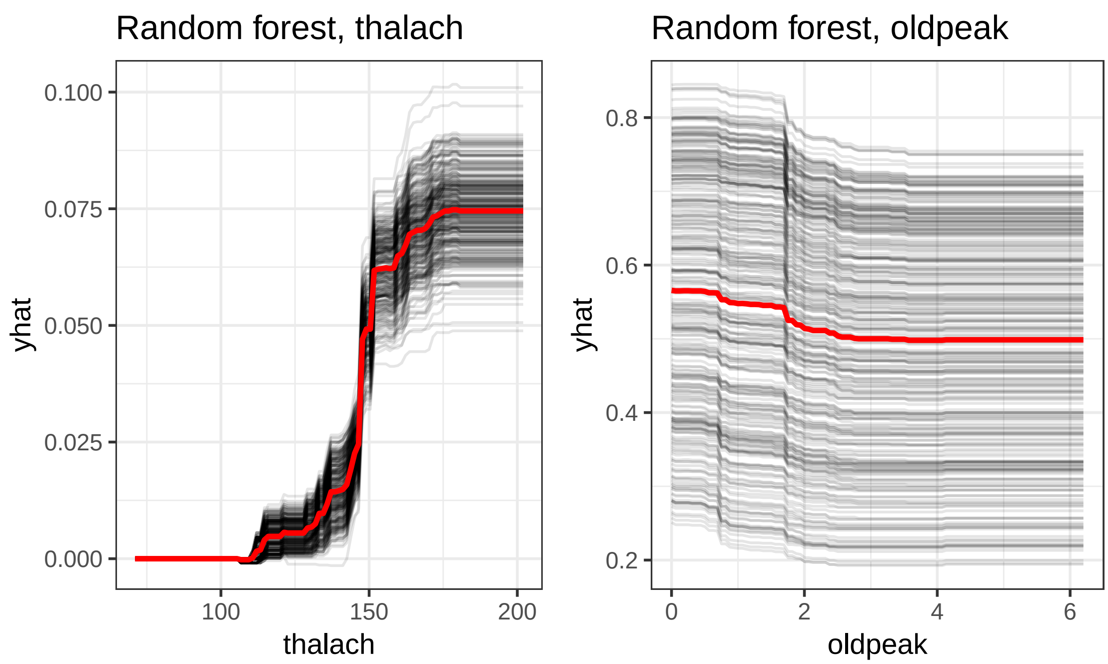



```
##                 var     rel.inf
## oldpeak   oldpeak 13.25454921
## thal2       thal2 12.64189418
## ca             ca 12.45748937
## thalach   thalach 10.27116560
## chol         chol  9.62876235
## age           age  8.95061424
## trestbps trestbps  8.20801677
## cp2           cp2  4.29299200
## exang1     exang1  4.10768934
## thal3       thal3  3.93422601
## sex1         sex1  2.82822808
## cp3           cp3  2.54336672
## restecg1 restecg1  2.16062178
## slope1     slope1  1.99785656
## slope2     slope2  1.36531004
## cp1           cp1  1.17180420
## fbs1         fbs1  0.11237993
## restecg2 restecg2  0.07303361
```

centered ICE

```
ice_ca.rf = rf.class %>%
    pdp::partial(pred.var = "thalach",
            grid.resolution = 100,
            ice = TRUE,
            prob = TRUE) %>%
    autoplot(train = heart_disease, alpha = .1, center = TRUE) +
    ggtitle("Random forest, thalach")
```

```
ice_oldpeak.rf =  rf.class %>%
    partial(pred.var = "oldpeak",
            grid.resolution = 100,
            ice = TRUE,
            prob = TRUE) %>%
    autoplot(train = heart_disease, alpha = .1) +
    ggtitle("Random forest, oldpeak")

grid.arrange(ice_ca.rf, ice_oldpeak.rf, nrow = 1)
```

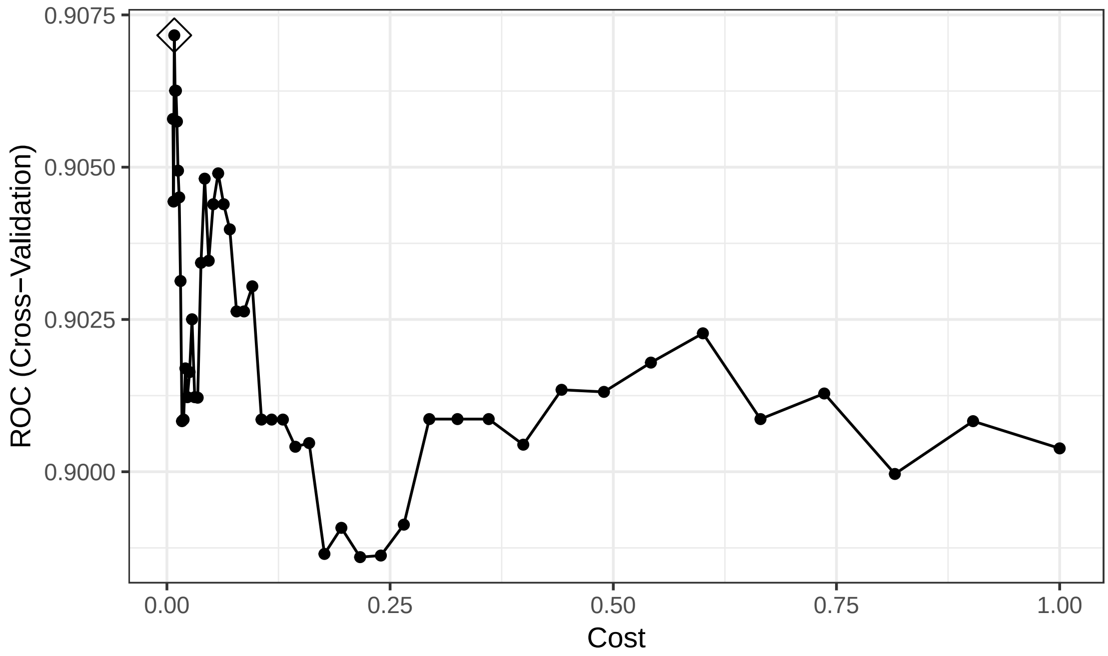


SVM

```
## linear boundary
set.seed(1)
svml.fit <- train(target~.,
                  data = heart_disease,
                  method = "svmLinear2",
                  preProcess = c("center", "scale"),
                  tuneGrid = data.frame(cost = exp(seq(-5,0,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.
```

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



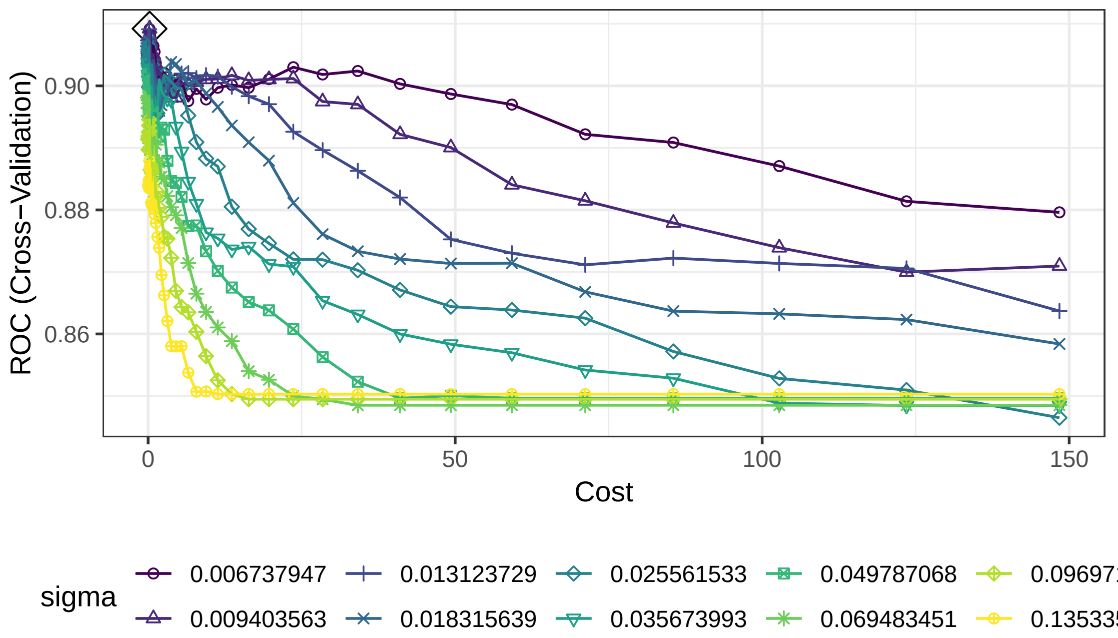

```r
svml.fit$bestTune
```

```
##           cost
## 3 0.008263406
```

```r
## radial kernel
svmr.grid <- expand.grid(C = exp(seq(-4,5,len=50)),
                         sigma = exp(seq(-5,-2,len=10)))
set.seed(1)
svmr.fit <- train(target~.,
                  data = heart_disease,
                  method = "svmRadial",
                  preProcess = c("center", "scale"),
                  tuneGrid = svmr.grid,
                  trControl = ctrl)
```

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

```r
ggplot(svmr.fit, highlight = TRUE) +
    viridis::scale_color_viridis(discrete = TRUE) +
    scale_shape_manual(values = seq(1,10))
```



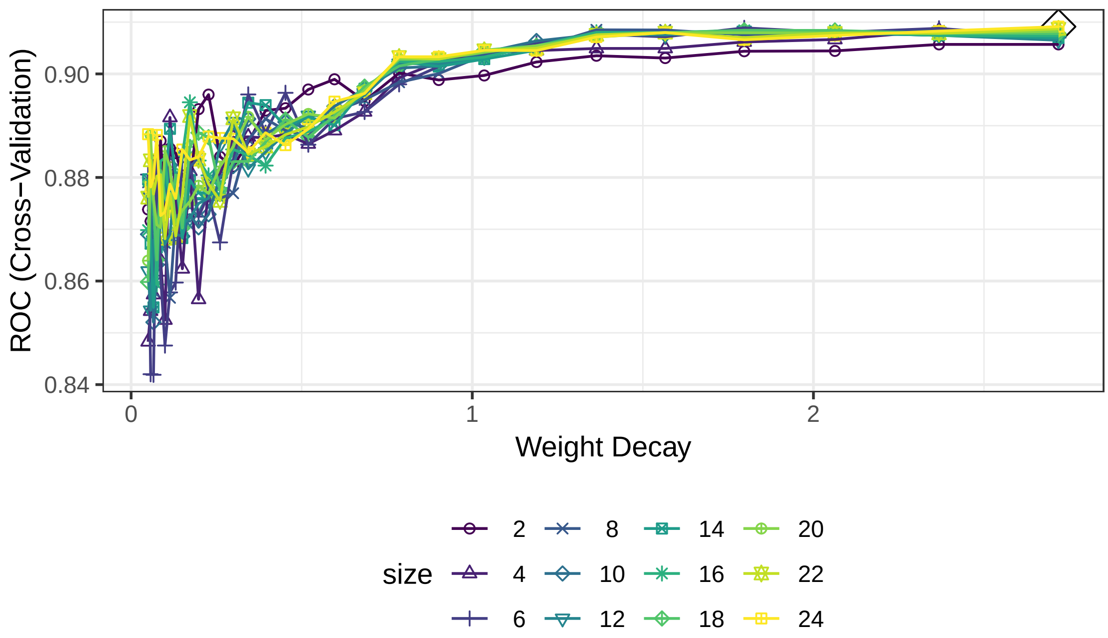

```
svmr.fit$bestTune
```

```
##           sigma        C
## 142 0.009403563 0.239651
```

```
#resamp <- resamples(list(svmr = svmr.fit,
#                          svml = svml.fit))
#bwplot(resamp)
#summary(resamp)
```

Neural network

```
nnetGrid <- expand.grid(size = seq(from = 2, to = 24, by = 2),
                        decay = exp(seq(from = -3, to = 1, length = 30)))

set.seed(1)
cnnet.fit <- train(target~.,
                   heart_disease,
                   method = "nnet",
                   tuneGrid = nnetGrid,
                   preProcess = c("center","scale"),
                   trControl = ctrl,
                   metric = "ROC",
                   trace = FALSE)

ggplot(cnnet.fit, highlight = TRUE) +
    viridis::scale_color_viridis(discrete = TRUE) +
    scale_shape_manual(values = seq(1,13))
```

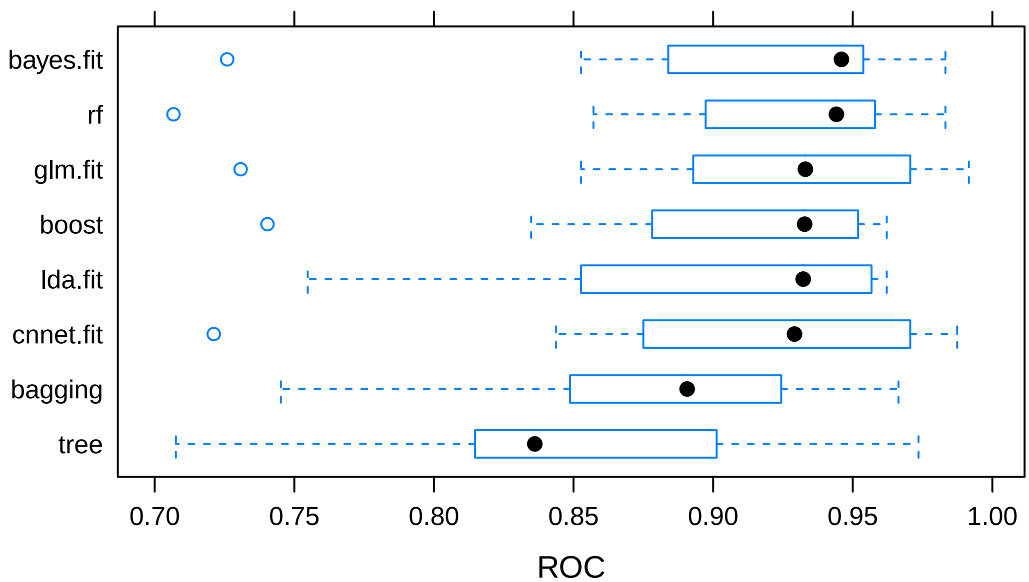



```r
resamp = resamples(list(
                        glm.fit = model.glm,
                        lda.fit = model.lda,
                        bayes.fit = model.bayes,
                        boost = boost.class,
                        rf = rf.class,
                        bagging = bagging.class,
                        tree = tree.class,
                        cnnet.fit = cnnet.fit
                        ))
summary(resamp)
```

```
##
## Call:
## summary.resamples(object = resamp)
##
## Models: glm.fit, lda.fit, bayes.fit, boost, rf, bagging, tree, cnnet.fit
## Number of resamples: 10
##
## ROC
##                Min.   1st Qu.    Median      Mean   3rd Qu.      Max. NA's
## glm.fit   0.7307692 0.8962054 0.9330357 0.9144756 0.9674370 0.9915966    0
## lda.fit   0.7548077 0.8642988 0.9322479 0.9052986 0.9567308 0.9621849    0
## bayes.fit 0.7259615 0.8917411 0.9459438 0.9131545 0.9537815 0.9831933    0
## boost     0.7403846 0.8851759 0.9327731 0.9047229 0.9505495 0.9621849    0
## rf        0.7067308 0.9040179 0.9441459 0.9144514 0.9569328 0.9831933    0
## bagging   0.7451923 0.8497243 0.8906957 0.8780503 0.9231880 0.9663866    0
## tree      0.7075893 0.8175223 0.8361345 0.8478325 0.8964983 0.9735577    0
## cnnet.fit 0.7211538 0.8816964 0.9291370 0.9091286 0.9684874 0.9873950    0
##
## Sens
```

```
##               Min.  1st Qu.    Median      Mean   3rd Qu.      Max. NA's
## glm.fit   0.7500 0.812500 0.9099265 0.8830882 0.9411765 1.0000000    0
## lda.fit   0.8125 0.875000 0.8823529 0.8959559 0.9402574 1.0000000    0
## bayes.fit 0.5625 0.765625 0.8786765 0.8393382 0.9402574 0.9411765    0
## boost     0.7500 0.812500 0.8750000 0.8643382 0.9264706 0.9411765    0
## rf        0.6875 0.828125 0.9099265 0.8827206 0.9411765 1.0000000    0
## bagging   0.6250 0.812500 0.8180147 0.8216912 0.8805147 0.9411765    0
## tree      0.6875 0.765625 0.8786765 0.8411765 0.9237132 0.9411765    0
## cnnet.fit 0.7500 0.812500 0.9099265 0.8830882 0.9411765 1.0000000    0
##
## Spec
##                Min.   1st Qu.    Median      Mean   3rd Qu.      Max. NA's
## glm.fit   0.5384615 0.7321429 0.8214286 0.8027473 0.9065934 0.9285714    0
## lda.fit   0.5384615 0.7142857 0.7142857 0.7598901 0.8887363 0.9285714    0
## bayes.fit 0.5384615 0.7321429 0.7857143 0.7956044 0.9230769 0.9285714    0
## boost     0.3846154 0.7321429 0.7857143 0.7653846 0.8543956 0.9230769    0
## rf        0.3846154 0.6607143 0.7857143 0.7373626 0.8392857 0.9230769    0
## bagging   0.4615385 0.7142857 0.7857143 0.7516484 0.8310440 0.9230769    0
## tree      0.5384615 0.7142857 0.7417582 0.7516484 0.7857143 0.9285714    0
## cnnet.fit 0.5384615 0.7321429 0.8214286 0.7956044 0.9065934 0.9285714    0
```

```r
bwplot(resamp, metric = "ROC")
```



Comparing accuracy

Regularized logistic