data_test_Weijia

Weijia Xiong

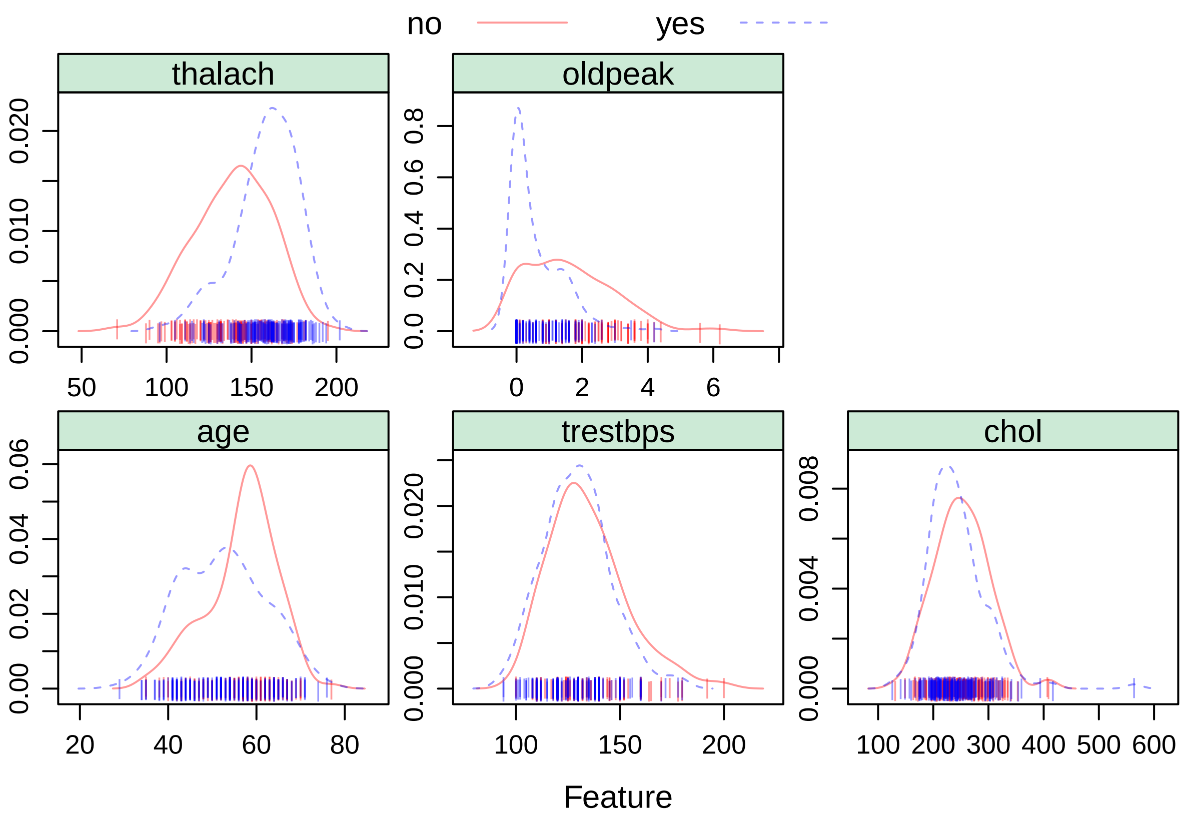
5/15/2020

Load data

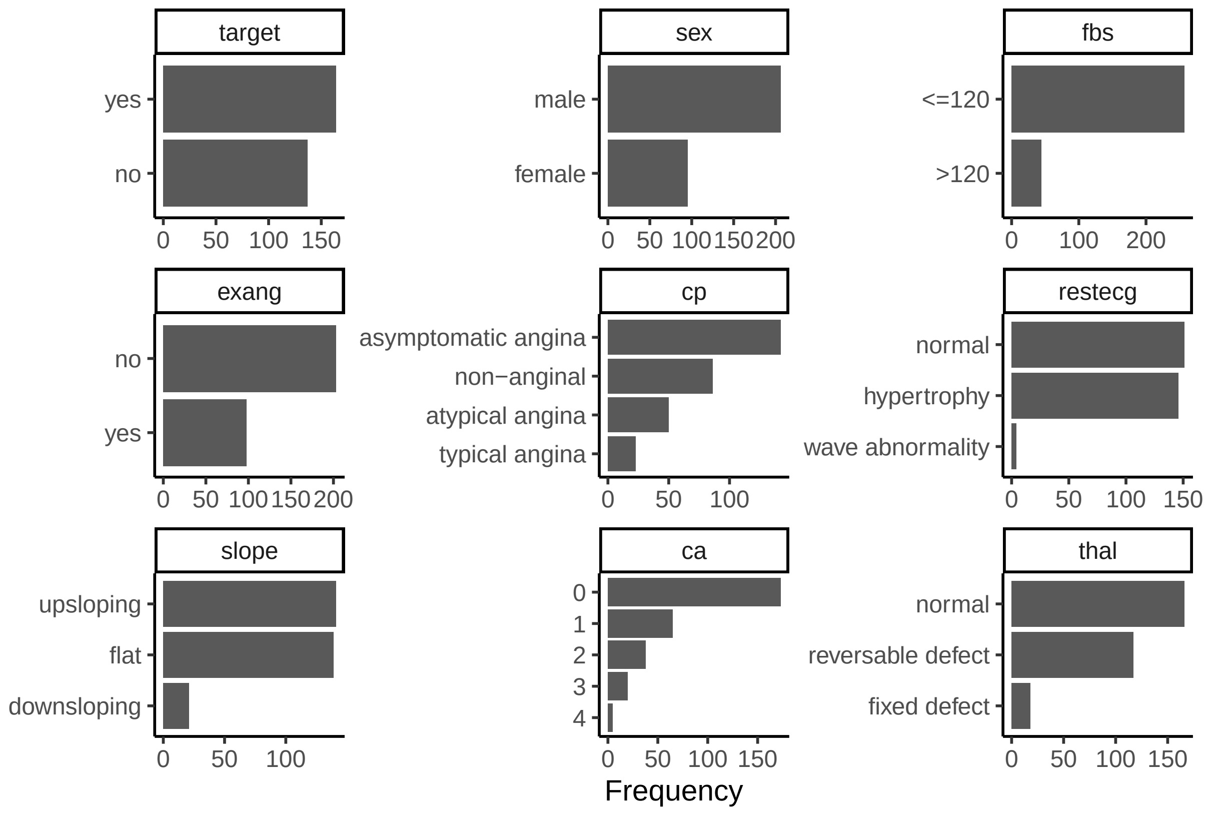
```r
# import data
heart = read_csv("heart.csv")

data = heart %>%
  mutate(sex = if_else(sex == 1, "male", "female"),
         fbs = if_else(fbs == 1, ">120", "<=120"),
         exang = if_else(exang == 1, "yes" ,"no"),
         target = if_else(target == 1, "yes", "no"),
         cp = case_when(
           cp == 3 ~ "typical angina",
           cp == 1 ~ "atypical angina",
           cp == 2 ~ "non-anginal",
           cp == 0 ~ "asymptomatic angina"
                        ),
         restecg = case_when(
           restecg == 0 ~ "hypertrophy",
           restecg == 1 ~ "normal",
           restecg == 2 ~ "wave abnormality"
                              ),
        slope = case_when(
          slope == 2 ~ "upsloping",
          slope == 1 ~ "flat",
          slope == 0 ~ "downsloping"
        ),
        thal = case_when(
          thal == 1 ~ "fixed defect",
          thal == 2 ~ "normal",
          thal == 3 ~ "reversable defect"
        ),
         cp = as.factor(cp),
         restecg = as.factor(restecg),
         slope = as.factor(slope),
         ca = as.factor(ca),
         thal = as.factor(thal)
         ) %>%
  mutate_if(is.character, as.factor) %>%
  dplyr::select(target, sex, fbs, exang, cp, restecg, slope, ca, thal, everything()) %>%
  na.omit()
```

Exploratory analysis/visualization

```
theme1 <- transparentTheme(trans = .4)
theme1$strip.background$col <- rgb(.0, .6, .2, .2)
trellis.par.set(theme1)

featurePlot(x = data[, 10:14],
            y = data$target,
            scales = list(x=list(relation="free"),
                          y=list(relation="free")),
            plot = "density", pch = "|",
            auto.key = list(columns = 2))
```



```
#ggpairs(data[,1:9])
plot_bar(data,ggtheme = theme_classic())
```

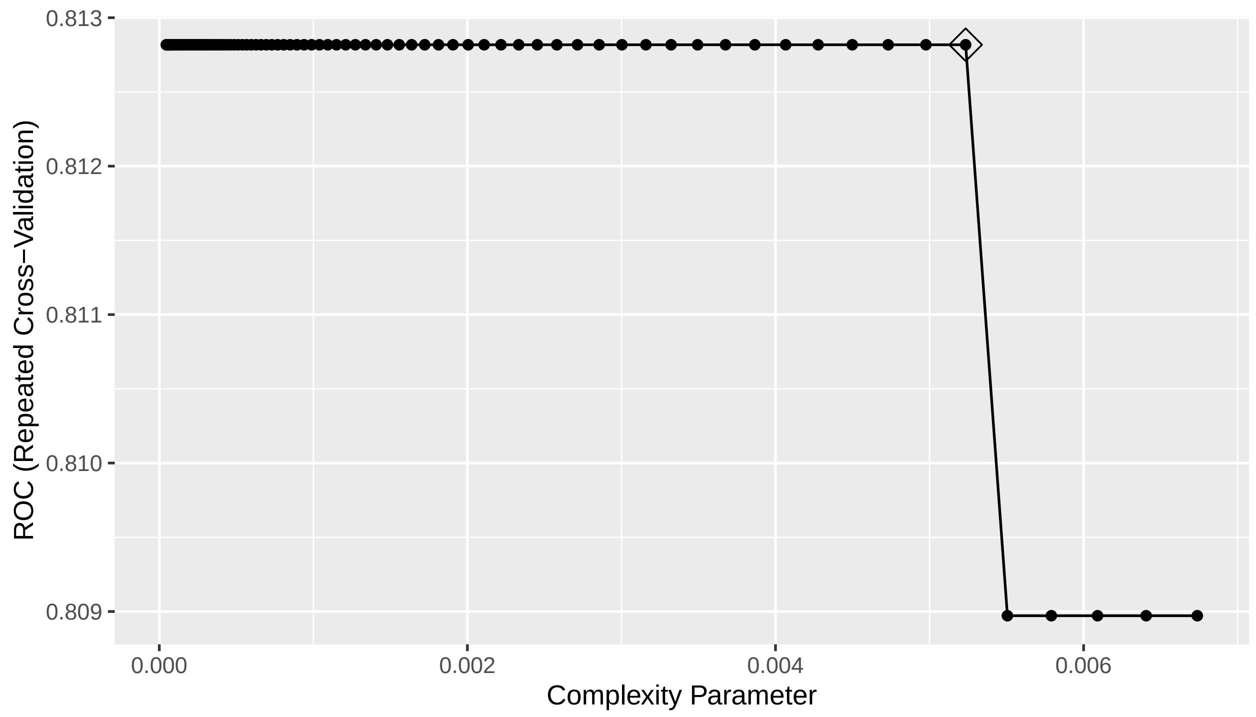


Models

```
# train test set partition
set.seed(123)
rowTrain <- createDataPartition(y = data$target,
                                p = 0.75,
                                list = FALSE)
```

Tree

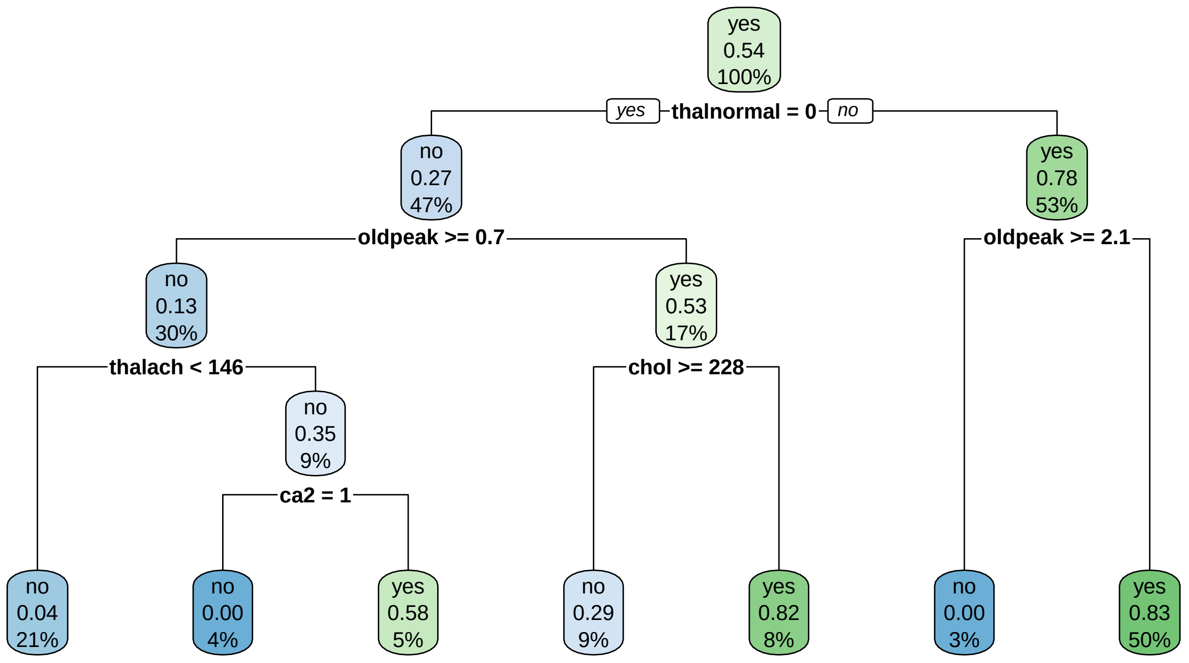
```
set.seed(123)
ctrl <- trainControl(method = "repeatedcv", summaryFunction = twoClassSummary, classProbs = TRUE)

rpart.fit.c <- train(target~., data=data[rowTrain,],
                     method = "rpart",
                     tuneGrid = data.frame(cp = exp(seq(-10, -5, len = 100))),
                     trControl = ctrl,
                     metric = "ROC")
ggplot(rpart.fit.c, highlight = TRUE)
```



```
rpart.fit.c$finalModel$cptable
```

```
##            CP nsplit rel error
## 1 0.466019417      0 1.0000000
## 2 0.067961165      1 0.5339806
## 3 0.053398058      2 0.4660194
## 4 0.009708738      4 0.3592233
## 5 0.005234284      6 0.3398058
```

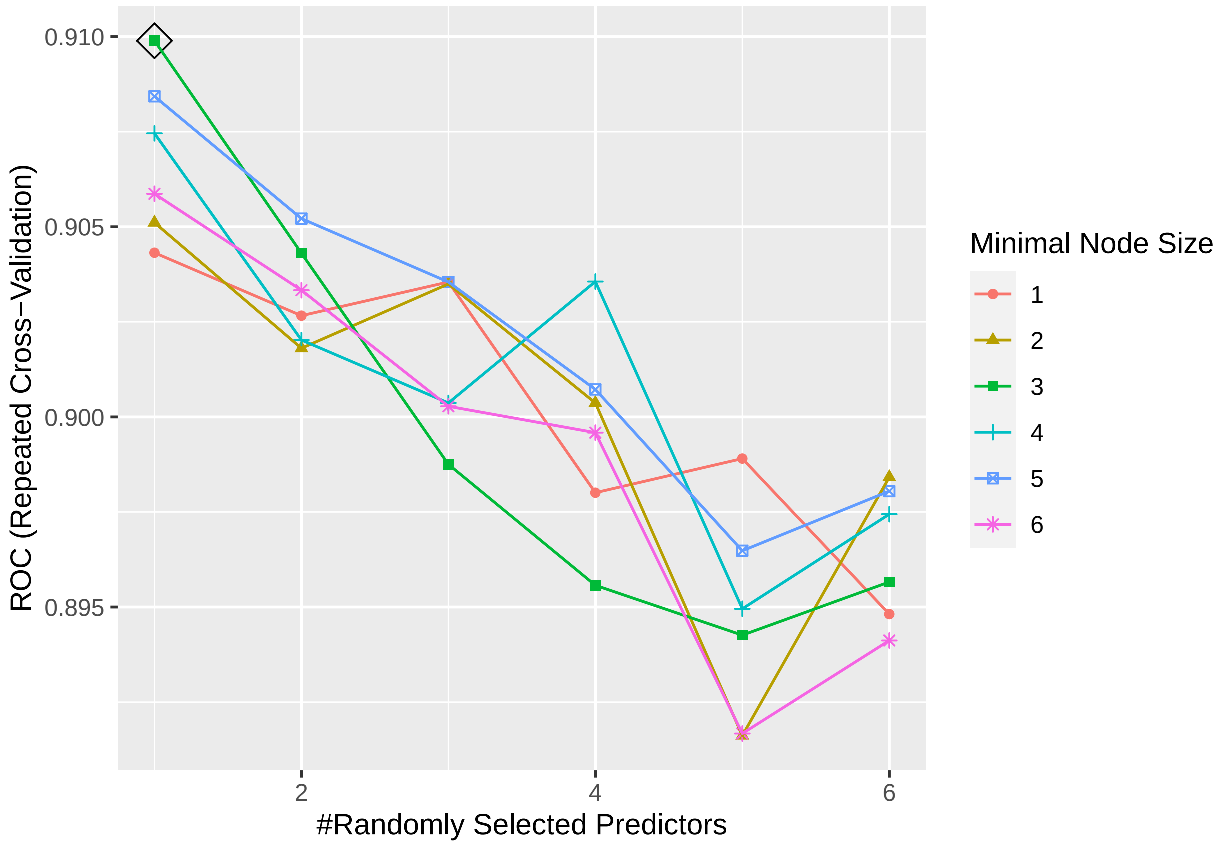
```
rpart.plot(rpart.fit.c$finalModel)
```



```r
# error rate
tree_error_rate = mean(data[-rowTrain,]$target != predict(rpart.fit.c, newdata = data[-rowTrain,], type
```

Random Forest

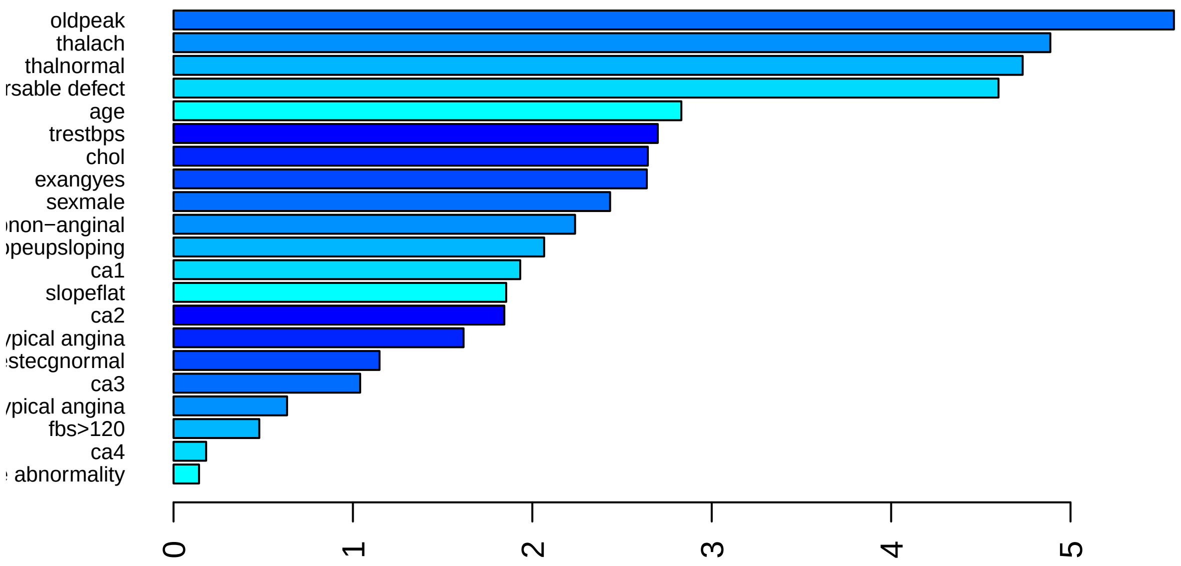
```r
rf.grid <- expand.grid(mtry = 1:6,
                       splitrule = "gini",
                       min.node.size = 1:6)
set.seed(123)
rf.fit.c <- train(target~., data=data[rowTrain,],
                method = "ranger",
                tuneGrid = rf.grid,
                metric = "ROC",
                trControl = ctrl,
                importance = "impurity")
ggplot(rf.fit.c, highlight = TRUE)
```



```r
rf.fit.c$bestTune
```

```
##   mtry splitrule min.node.size
## 3    1      gini             3
```

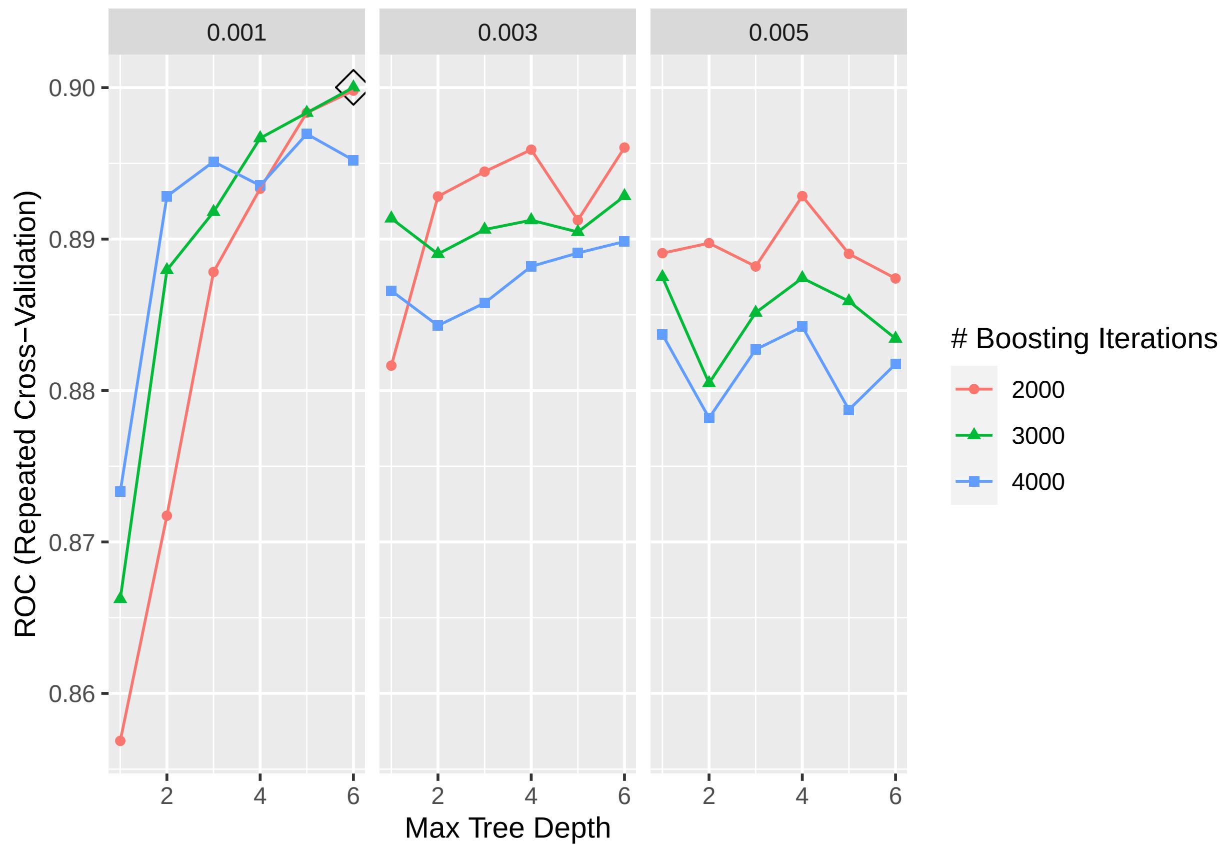
```r
# variable importance
barplot(sort(ranger::importance(rf.fit.c$finalModel), decreasing = FALSE),
        las = 2, horiz = TRUE, cex.names = 0.7,
        col = colorRampPalette(colors = c("cyan","blue"))(8))
```



```r
# error rate
rf_error_rate = mean(data[-rowTrain,]$target != predict(rf.fit.c, newdata = data[-rowTrain,], type = "ra
```

Boosting

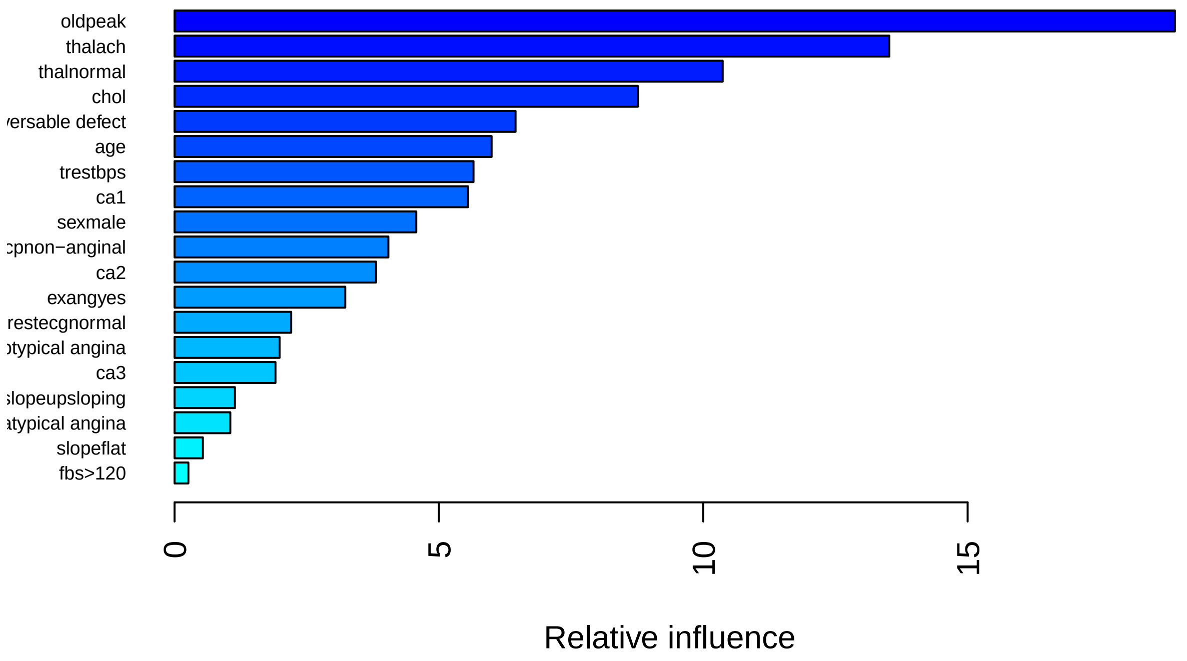
```r
gbmB.grid <- expand.grid(n.trees = c(2000,3000,4000),
                         interaction.depth = 1:6,
                         shrinkage = c(0.001,0.003,0.005),
                         n.minobsinnode = 1)
set.seed(123)
gbmB.fit <- train(target~., data=data[rowTrain,],
                  tuneGrid = gbmB.grid,
                  trControl = ctrl,
                  method = "gbm",
                  distribution = "bernoulli",
                  metric = "ROC",
                  verbose = FALSE)
ggplot(gbmB.fit, highlight = TRUE)
```



```r
gbmB.fit$bestTune
```

```
##    n.trees interaction.depth shrinkage n.minobsinnode
## 17    3000                 6     0.001              1
```

```r
# variable importance
summary(gbmB.fit$finalModel, las = 2, cBars = 19, cex.names = 0.6)
```



```
##                                                var     rel.inf
## oldpeak                                    oldpeak 18.91958101
## thalach                                    thalach 13.51904469
## thalnormal                              thalnormal 10.36704176
## chol                                          chol  8.76230743
## thalreversable defect     thalreversable defect  6.44987160
## age                                            age  5.99660020
## trestbps                                  trestbps  5.65487628
## ca1                                            ca1  5.55117966
## sexmale                                    sexmale  4.57210418
## cpnon-anginal                        cpnon-anginal  4.04442705
## ca2                                            ca2  3.81144617
## exangyes                                  exangyes  3.22988398
## restecgnormal                        restecgnormal  2.20678228
## cptypical angina                  cptypical angina  1.98658330
## ca3                                            ca3  1.90934173
## slopeupsloping                      slopeupsloping  1.14213877
## cpatypical angina                cpatypical angina  1.05536264
## slopeflat                                slopeflat  0.53604546
## fbs>120                                    fbs>120  0.26235260
## ca4                                            ca4  0.02302921
## restecgwave abnormality restecgwave abnormality  0.00000000
```

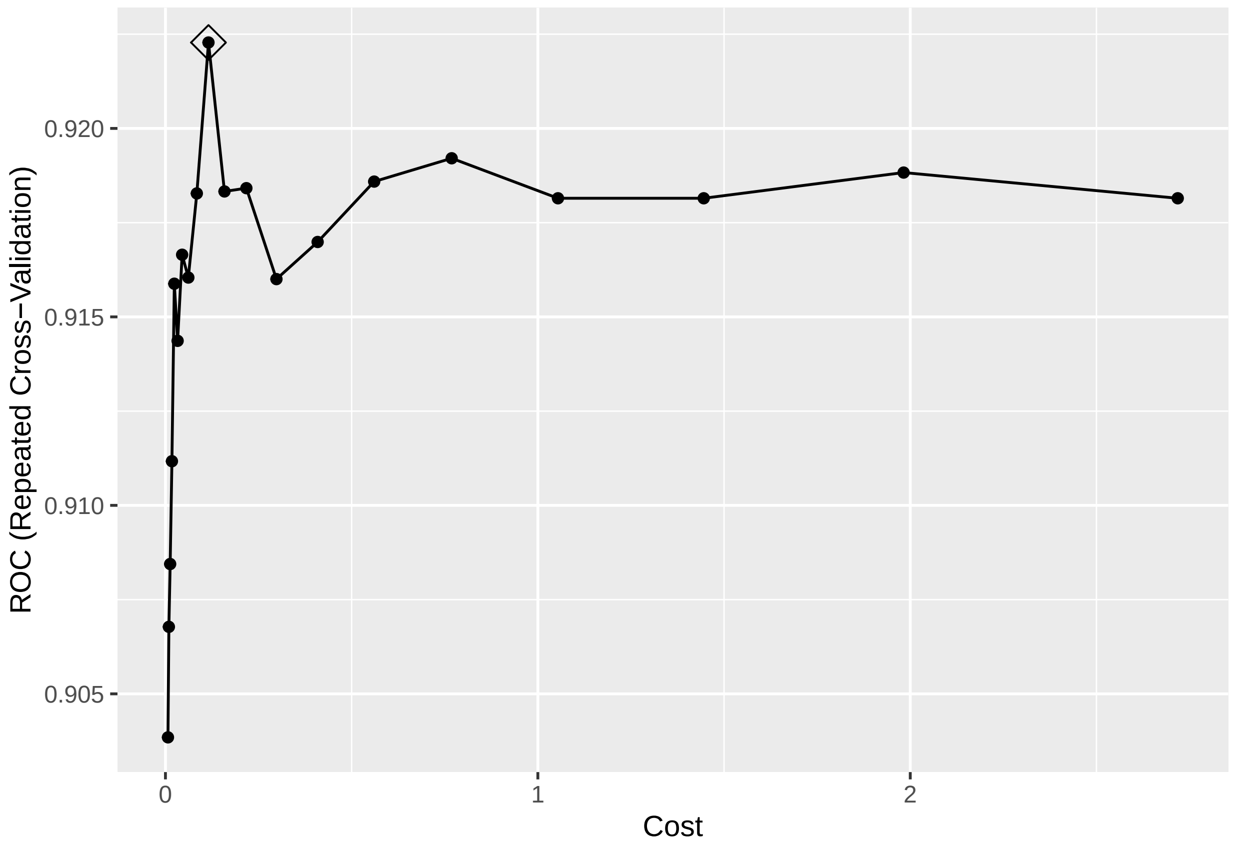
```
# error rate
boost_error_rate = mean(data[-rowTrain,]$target != predict(gbmB.fit, newdata = data[-rowTrain,], type =
```

Support vector machine

Linear kernel

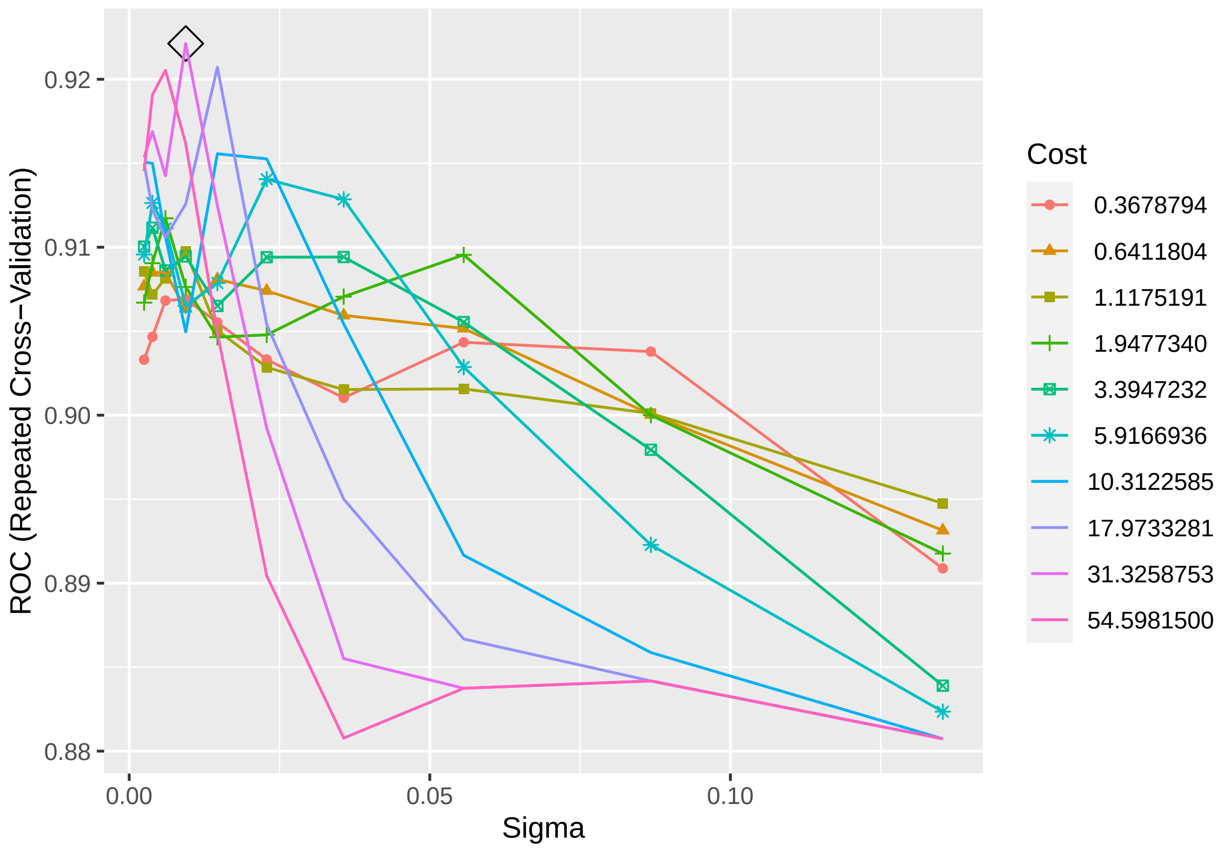
```
set.seed(123)
svml.fit <- train(target~.,
                  data = data[rowTrain,],
```

```r
                  method = "svmLinear2",
                  preProcess = c("center", "scale"),
                  tuneGrid = data.frame(cost = exp(seq(-5,1,len=20))),
                  trControl = ctrl)
ggplot(svml.fit, highlight = TRUE)
```



Radial kernel

```r
svmr.grid <- expand.grid(C = exp(seq(-1,4,len=10)),
                         sigma = exp(seq(-6,-2,len=10)))
set.seed(123)
svmr.fit <- train(target~.,
                  data = data,
                  subset = rowTrain,
                  method = "svmRadial",
                  preProcess = c("center", "scale"), tuneGrid = svmr.grid,
                  trControl = ctrl)
ggplot(svmr.fit, highlight = TRUE)
```



Test model performance in test data

```
pred.svmr <- predict(svmr.fit, newdata = data[-rowTrain,])
pred.svml <- predict(svml.fit, newdata = data[-rowTrain,])
matr.svmr = confusionMatrix(data = pred.svmr,
                reference = data$target[-rowTrain])
matr.svml = confusionMatrix(data = pred.svml,
                reference = data$target[-rowTrain])
matr.svmr
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction no yes
##        no  26  10
##        yes  8  31
##
##                Accuracy : 0.76
##                  95% CI : (0.6475, 0.8511)
##     No Information Rate : 0.5467
##     P-Value [Acc > NIR] : 0.0001097
##
##                   Kappa : 0.5182
##
##  Mcnemar's Test P-Value : 0.8136637
##
##             Sensitivity : 0.7647
##             Specificity : 0.7561
##          Pos Pred Value : 0.7222
##          Neg Pred Value : 0.7949
##              Prevalence : 0.4533
##          Detection Rate : 0.3467
##    Detection Prevalence : 0.4800
##       Balanced Accuracy : 0.7604
##
##        'Positive' Class : no
##
```

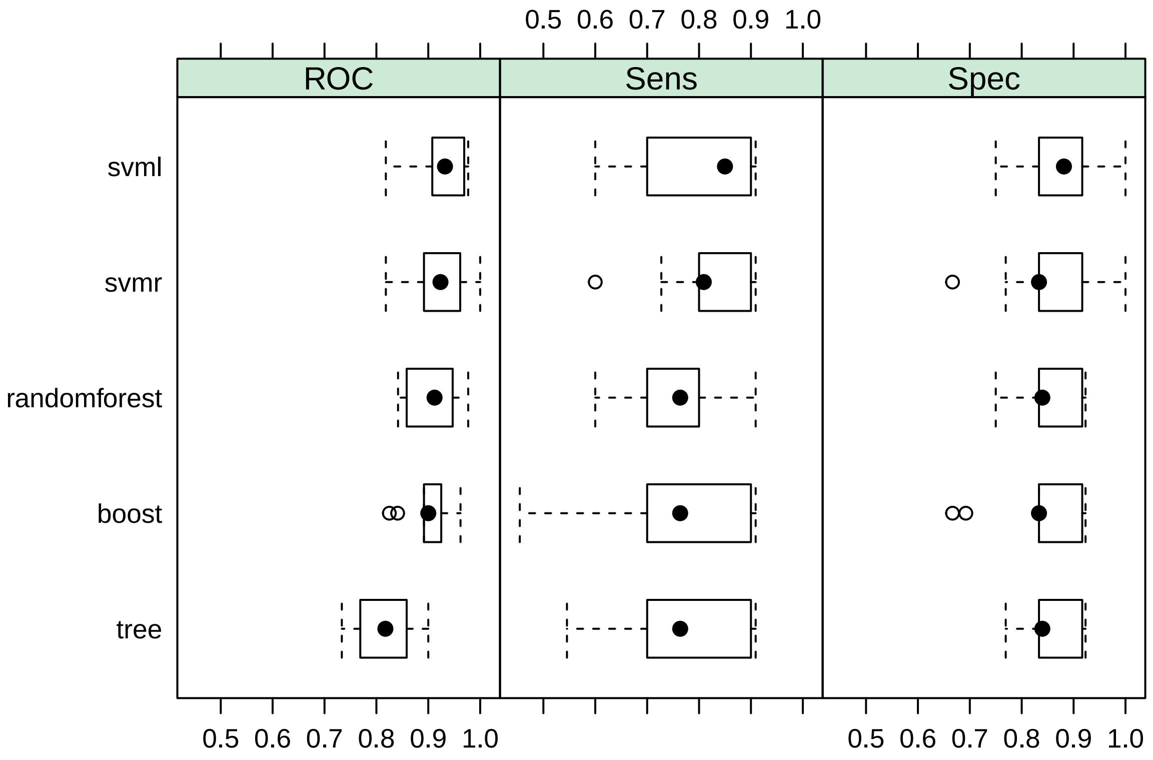
```
matr.svml
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction no yes
##        no  27   7
##        yes  7  34
##
##                Accuracy : 0.8133
##                  95% CI : (0.7067, 0.894)
##     No Information Rate : 0.5467
##     P-Value [Acc > NIR] : 1.183e-06
##
##                   Kappa : 0.6234
##
##  Mcnemar's Test P-Value : 1
##
##             Sensitivity : 0.7941
##             Specificity : 0.8293
##          Pos Pred Value : 0.7941
##          Neg Pred Value : 0.8293
##              Prevalence : 0.4533
##          Detection Rate : 0.3600
##    Detection Prevalence : 0.4533
##       Balanced Accuracy : 0.8117
##
##        'Positive' Class : no
##
```

```
svmr_error_rate = mean(data[-rowTrain,]$target != pred.svmr, type = "raw")
svml_error_rate = mean(data[-rowTrain,]$target != pred.svml, type = "raw")
```

Comparison of models

```
set.seed(123)
resamp <- resamples(list(svml = svml.fit, svmr = svmr.fit, boost = gbmB.fit,randomforest = rf.fit.c, tre
bwplot(resamp)
```



The support vector machine model with linear kernel performs better with higher accuracy and kappa when checking their predictive ability with test data.

```
#save.image(file='test_Weijia.RData')
```