Introduction to classification and regression trees, random forests and model-based recursive partitioning in R

Day 2: Stability and tree ensembles

Exercise 6: Boston housing data (2)

Use the `gbm()` function from the package `gbm` to Fit a boosted ensemble on the Boston housing training data. Probably, you will first need to install the package `gbm`. Specify `distribution = "gaussian"`, as the outcome variable is continuous.

a) Use 5 or 10-fold cross validation on the training data to find the optimal value of the number of trees (`n.trees`). If you feel like it, also determine the optimal shrinkage parameter or learning rate by cross validation. Specify a sensible value for `interaction.depth` (or, alternatively, also determine the optimal value by cross-validation).

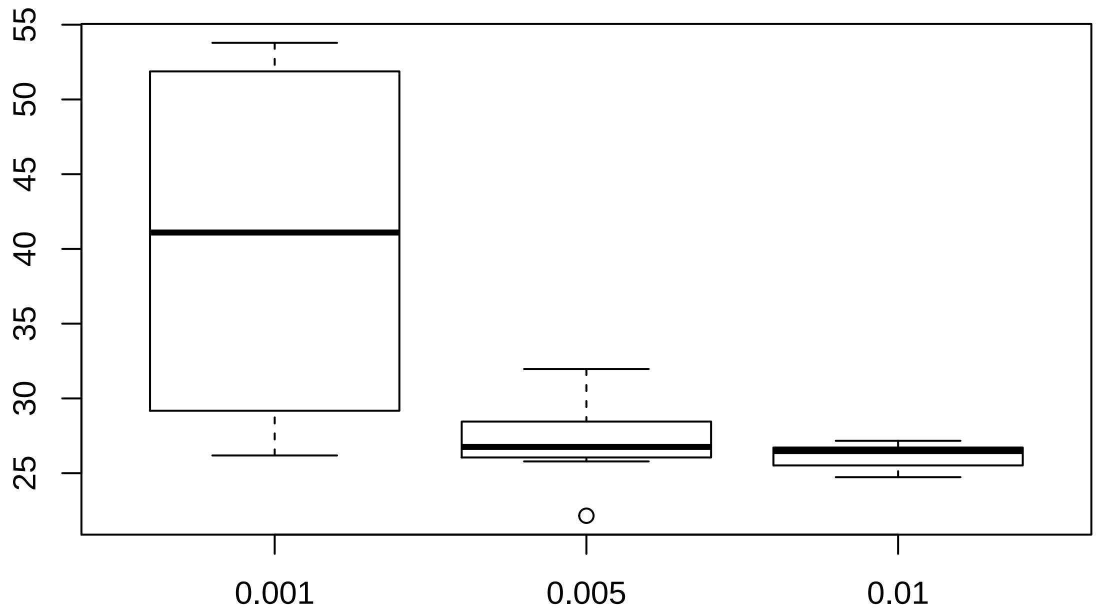
```r
data("Boston", package = "MASS")
library(gbm)
```

```
## Loading required package: survival

## Loading required package: lattice

## Loading required package: splines

## Loading required package: parallel

## Loaded gbm 2.1.3
```

```r
set.seed(1)
train <- sample(1:nrow(Boston), nrow(Boston)/4)
x <- Boston[train, -14]
y <- Boston[train, 14]
xtest <- Boston[-train, -14]
ytest <- Boston[-train, 14]
```

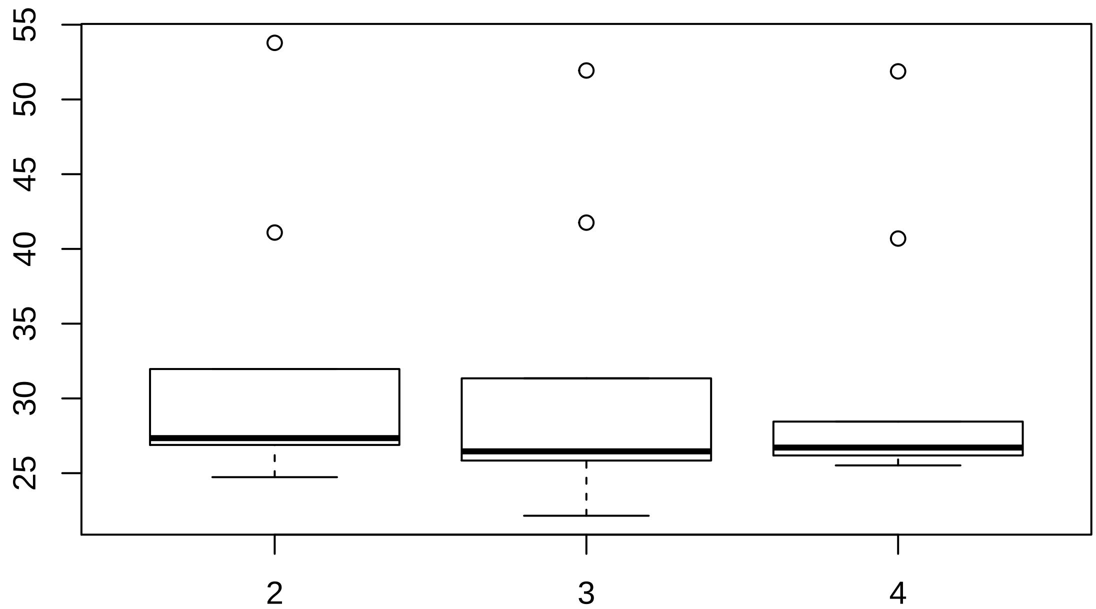
```r
# Use CV to determine best values of shrinkage, interaction.depth and n.trees:
set.seed(825)

# create function to get cross-validated error estimates:
cv.boosting.params <- function(formula = NULL, data = NULL, shrinkage = .001,
                               interaction.depth = 1, bag.fraction = .5, cv.folds = 10,
                               distribution = "gaussian", n.trees = 500) {
  boosted.cv.res <- list()
  counter <- 0
  for (i in 1:length(shrinkage)) {
    for (j in 1:length(interaction.depth)) {
      for (k in 1:length(bag.fraction)) {
        for (l in 1:length(n.trees)) {
          counter <- counter + 1
          boosted.fit <- gbm(formula = formula, data = data,
                             distribution = distribution,
                             bag.fraction = bag.fraction[k],
                             n.trees = n.trees[l], shrinkage = shrinkage[i],
                             interaction.depth = interaction.depth[j],
                             cv.folds = cv.folds)
          boosted.cv.res[[counter]] <- list(boosted.fit = boosted.fit,
```

```
                                              best.n.tree = which(boosted.fit$cv.error ==
                                                                  min(boosted.fit$cv.error)),
                                              best.n.tree.error = min(boosted.fit$cv.error),
                                              ensemble.error = boosted.fit$cv.error[n.trees],
                                              shrinkage = shrinkage[i],
                                              n.trees = n.trees[l],
                                              interaction.depth = interaction.depth[i],
                                              bag.fraction = bag.fraction[k])
        }
      }
    }
  }
  return(boosted.cv.res)
}

# specify values of shrinkage, interaction.depth and n.trees
boosted.cv <- cv.boosting.params(formula = medv ~ ., data = Boston[train,],
                          shrinkage = c(.001, .005, .01),
                          interaction.depth = 2:4, distribution = "gaussian",
                          n.trees = c(500, 1000, 5000))

# Extract cross validation results:
opt_enssize <- list()
cv_opt_error <- list()
for(i in 1:length(boosted.cv)) {
  opt_enssize[[i]] <- which(boosted.cv[[i]]$boosted.fit$cv.error ==
                              min(boosted.cv[[i]]$boosted.fit$cv.error))
  cv_opt_error[[i]] <- min(boosted.cv[[i]]$boosted.fit$cv.error)
}

# plot cross validation results:
cv.results <- data.frame(shrinkage = rep(c(.001, .005, .01), each = 9),
                         interaction.depth = rep(rep(2:4, each = 3), times = 3),
                         n.trees = rep(c(500, 1000, 5000), times = 9),
                         cv_error = unlist(cv_opt_error))

boxplot(cv_error ~ shrinkage, data = cv.results)
```



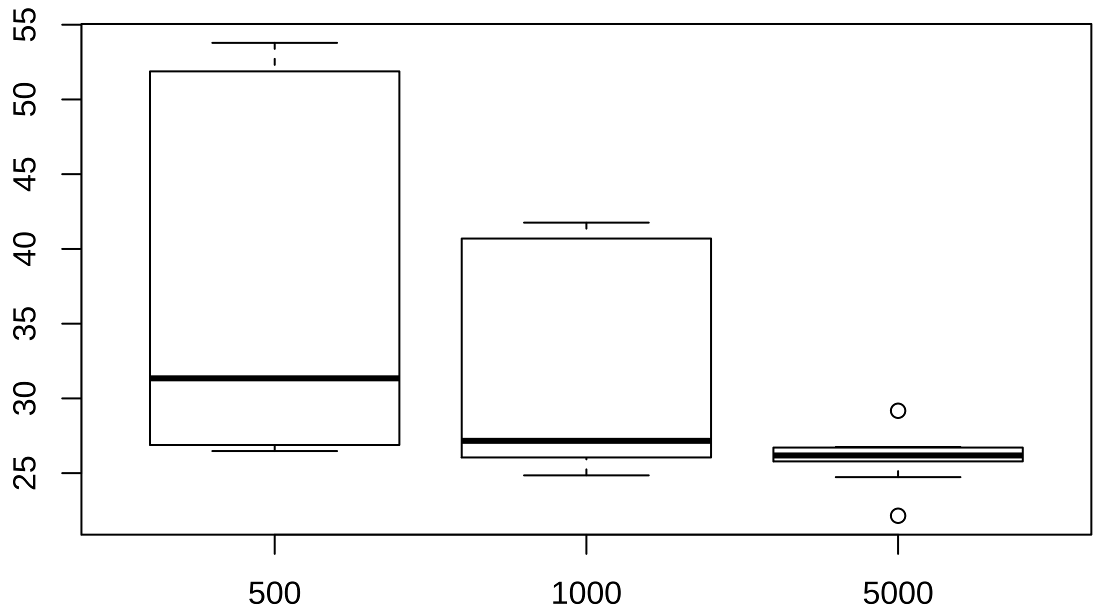

```r
# A learning rate of .01 seems to perform best, overall.
```

```r
boxplot(cv_error ~ interaction.depth, data = cv.results)
```




```r
# An interaction depth of 4 seems to perform best, overall.
```

```r
boxplot(cv_error ~ n.trees, data = cv.results)
```




```
# An ensemble of 5,000 trees seems to perform best, overall.
```

b) Fit a boosted tree ensemble on the training data, using the optimal parameter values you found above.

```
# Now fit a boosted tree model with the tuned parameters:
set.seed(23456)
Boston.boost <- gbm(medv ~., data = Boston[train,], n.trees = 5000, shrinkage = .01,
                    interaction.depth = 4, distribution = "gaussian")
mean((Boston[-train,"medv"] - predict(Boston.boost, newdata = Boston[-train,], n.trees= 5000))^2)
```

```
## [1] 17.35978
```

c) Compare the test error of the boosted tree ensemble with that of the random forest and bagged ensemble you fitted above.

```
library(randomForest)
```

```
## randomForest 4.6-12
```

```
## Type rfNews() to see new features/changes/bug fixes.
```

```
set.seed(1)
rf.boston <- randomForest(x = x, y = y, mtry = sqrt(13), ntree = 750, data = Boston)
mean(((Boston[-train,"medv"] - predict(rf.boston, newdata = xtest))^2))
```

```
## [1] 14.59218
```

```
bag.boston <- randomForest(x = x, y = y, mtry = 13, ntree = 750, data = Boston)
mean(((Boston[-train,"medv"] - predict(bag.boston, newdata = xtest))^2))
```

```
## [1] 14.59535
```

```
# The boosted ensemble is outperformed by the random forest and bagged ensemble,
# on this occasion.
```