HW8 Solutions

Dante Wu

2023-03-10

1. Split the dataset

```
boston = read.csv("/Users/wugaoyu/Desktop/PhD/TA/Winter 2023/Boston.csv")

set.seed(189)
train.index = sample(1:nrow(boston), (1/2)*nrow(boston)+1)

data.train = boston[train.index, ]
data.test = boston[-train.index, ]
```

2. A single regression tree

Using all predictors

```
library(ISLR)
library(MASS)
library(tree)
```

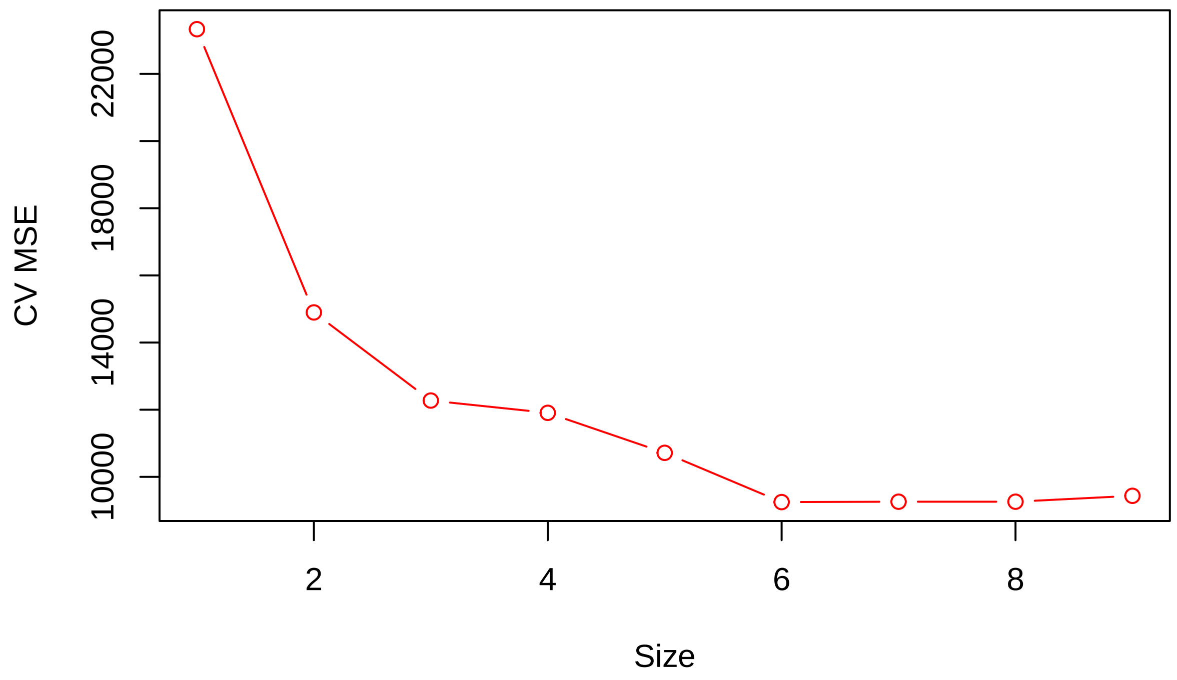
```
## Registered S3 method overwritten by 'tree':
##   method     from
##   print.tree cli
```

```
tree.boston = tree(medv~., data = data.train)
tree.boston.summary = summary(tree.boston)
tree.boston.summary
```

```
##
## Regression tree:
## tree(formula = medv ~ ., data = data.train)
## Variables actually used in tree construction:
## [1] "lstat"   "dis"     "rm"      "ptratio" "crim"
## Number of terminal nodes:  9
## Residual mean deviance:  16.35 = 4005 / 245
## Distribution of residuals:
##       Min.    1st Qu.     Median       Mean    3rd Qu.       Max.
## -16.320000  -2.060000   0.006796   0.000000   2.482000  11.660000
```

Cross validation and pruning

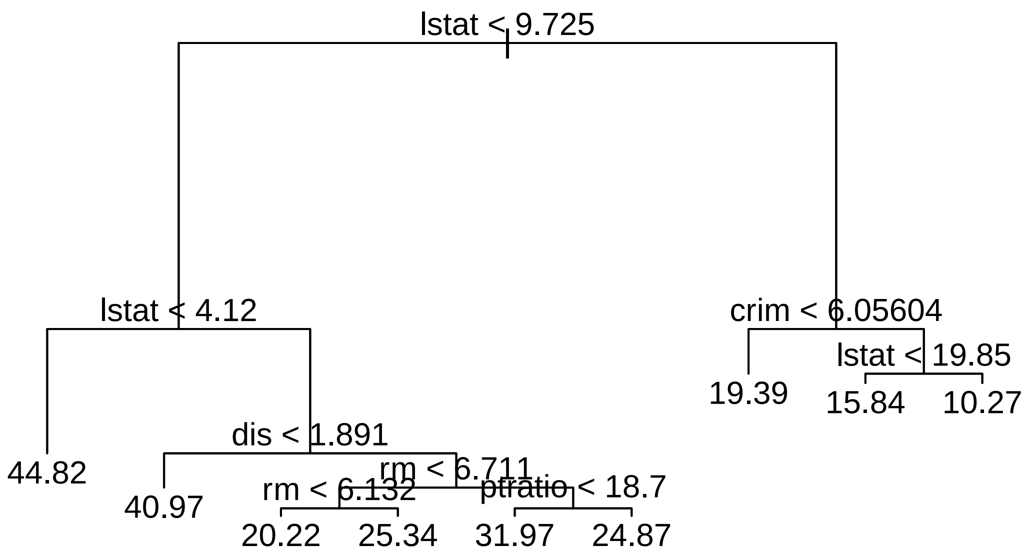
```
cv.boston = cv.tree(tree.boston, K=10)
plot(cv.boston$size, cv.boston$dev, type="b", xlab="Size", ylab="CV MSE", col="red")
```



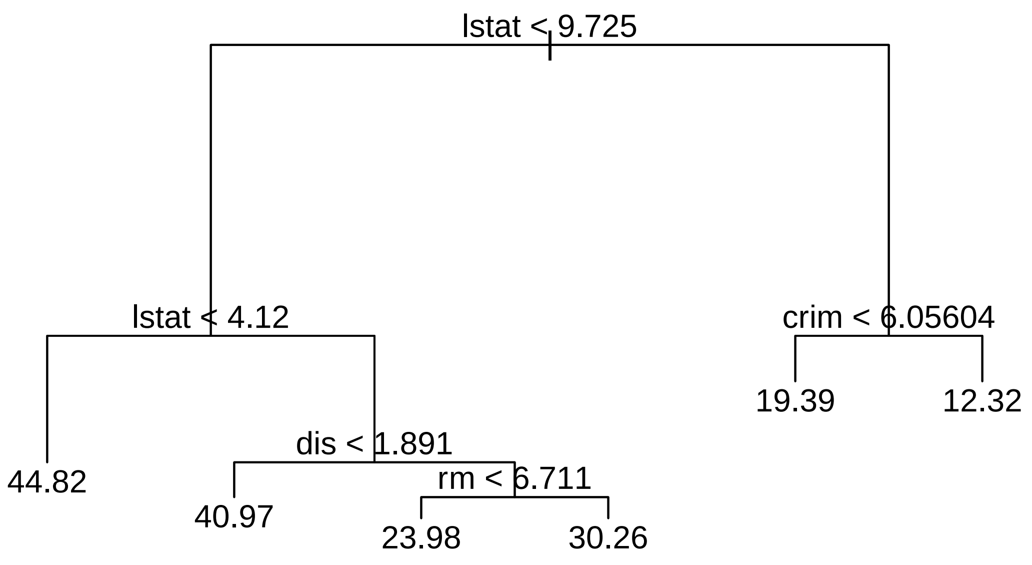
```
cv.size = cv.boston$size[which.min(cv.boston$dev)]

#prune
prune.boston = prune.tree(tree.boston, best=cv.size)
```

Plot both trees

```
plot(tree.boston)
text(tree.boston, pretty=0)
```



```
plot(prune.boston)
text(prune.boston, pretty = 0)
```



Calculating the test errors

```
y.test = data.test$medv

yhat.single = predict(tree.boston, newdata = data.test)
mse.single = mean((yhat.single - y.test)^2)

yhat.prune = predict(prune.boston, newdata = data.test)
mse.prune = mean((yhat.prune - y.test)^2)

sprintf("MSE for a single tree: %0.2f.", mse.single)
```

```
## [1] "MSE for a single tree: 27.41."
```

```
sprintf("MSE for a pruned tree: %0.2f.", mse.prune)
```

```
## [1] "MSE for a pruned tree: 30.74."
```

While the pruned tree has a slightly larger test error, the structure is simpler.

Remark: You may also have the same trees, which gives the same test errors. This is normal because the main objective is to minimize the MSE but not having restrictions on the size of the tree.

3. Bagging and Random Forest

Bagging

```
library(randomForest)
```

```
## randomForest 4.7-1.1
```

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

```
bagging.boston = randomForest(medv~., data = data.train,
                              mtry=(ncol(boston)-1), importance=TRUE, ntree=100)
bagging.boston
```

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

```
##  randomForest(formula = medv ~ ., data = data.train, mtry = (ncol(boston) -      1), importance = TRU
##                Type of random forest: regression
##                      Number of trees: 100
## No. of variables tried at each split: 14
##
##           Mean of squared residuals: 15.42806
##                     % Var explained: 82.97
```

```r
#calculating MSE
yhat.bagging = predict(bagging.boston, newdata = data.test)
mse.bagging = mean((yhat.bagging - y.test)^2)
sprintf("MSE for bagging: %0.2f.", mse.bagging)
```

```
## [1] "MSE for bagging: 9.48."
```

Random Forest

```r
RF.boston = randomForest(medv~., data = data.train, mtry=4, importance=TRUE, ntree=100)
RF.boston
```

```
##
## Call:
##  randomForest(formula = medv ~ ., data = data.train, mtry = 4,      importance = TRUE, ntree = 100)
##                Type of random forest: regression
##                      Number of trees: 100
## No. of variables tried at each split: 4
##
##           Mean of squared residuals: 15.57938
##                     % Var explained: 82.8
```

```r
#calculating MSE
yhat.RF = predict(RF.boston, newdata = data.test)
mse.RF = mean((yhat.RF - y.test)^2)
sprintf("MSE for random forest: %0.2f.", mse.RF)
```

```
## [1] "MSE for random forest: 8.82."
```

Make comparisions

```r
table.tree = matrix(c(mse.single, mse.prune, mse.bagging, mse.RF), 1, 4)

colnames(table.tree) = c("Single", "Prune", "Bagging", "RF")

knitr::kable(table.tree, align = c(rep('c', 4)), digits = 2)
```

Single	Prune	Bagging	RF
27.41	30.74	9.48	8.82

From the results above, bagging and random forest both reduce the test errors dramatically, meaning that they outperform a single tree or a pruned tree.

Remark: Random forest is not necessarily better than bagging as the choice of m can be improper.