ExampleBostonDataset

alexaoh

3/9/2021

Example: Boston data set

(Taken from Stefanie Muff's lecture on Trees: Module 8 in TMA4268, spring 2021). Thank you! ;))

(ISLR book, Sections 8.3.2 to 8.3.4.)

Remember the data set: The aim is to predict the median value of owner-occupied homes (in 1000$)

We first run through trees, bagging and random forests - before arriving at boosting.

```r
library(MASS)
set.seed(1)
train = sample(1:nrow(Boston), nrow(Boston)/2)
head(Boston)
```

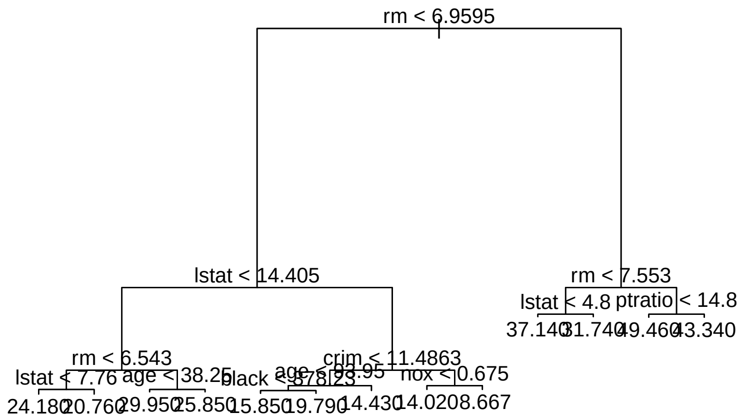
```
##      crim zn indus chas   nox    rm  age    dis rad tax ptratio  black lstat
## 1 0.00632 18  2.31    0 0.538 6.575 65.2 4.0900   1 296    15.3 396.90  4.98
## 2 0.02731  0  7.07    0 0.469 6.421 78.9 4.9671   2 242    17.8 396.90  9.14
## 3 0.02729  0  7.07    0 0.469 7.185 61.1 4.9671   2 242    17.8 392.83  4.03
## 4 0.03237  0  2.18    0 0.458 6.998 45.8 6.0622   3 222    18.7 394.63  2.94
## 5 0.06905  0  2.18    0 0.458 7.147 54.2 6.0622   3 222    18.7 396.90  5.33
## 6 0.02985  0  2.18    0 0.458 6.430 58.7 6.0622   3 222    18.7 394.12  5.21
##   medv
## 1 24.0
## 2 21.6
## 3 34.7
## 4 33.4
## 5 36.2
## 6 28.7
```

Regression tree

```r
library(tree)
tree.boston=tree(medv~.,Boston,subset=train,control = tree.control(nrow(Boston), mindev = 0.005))
summary(tree.boston)
```

```
##
## Regression tree:
## tree(formula = medv ~ ., data = Boston, subset = train, control = tree.control(nrow(Boston),
##     mindev = 0.005))
## Variables actually used in tree construction:
## [1] "rm"      "lstat"   "age"     "crim"    "black"   "nox"     "ptratio"
## Number of terminal nodes:  13
## Residual mean deviance:  7.353 = 1765 / 240
## Distribution of residuals:
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
## -8.1400 -1.4360 -0.1615  0.0000  1.4640 12.8600
```

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

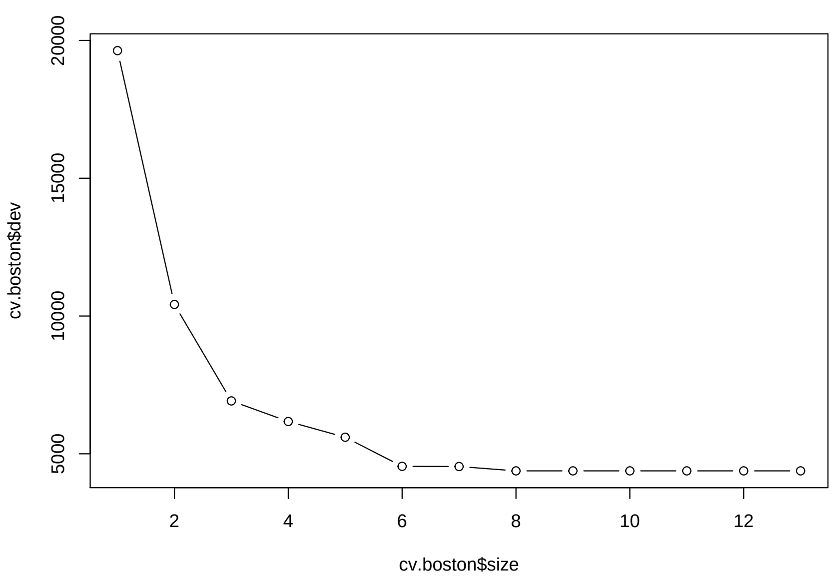


Remember:

- The `tree()` function has a built-in default stopping criterion.
- You can change this with the `control` option, for example by setting `control = tree.control(mincut = 2, minsize = 4, mindev = 0.001)`. Here we used `mindev=0.005`.

Need to prune?

```
cv.boston=cv.tree(tree.boston)
plot(cv.boston$size,cv.boston$dev,type='b')
```

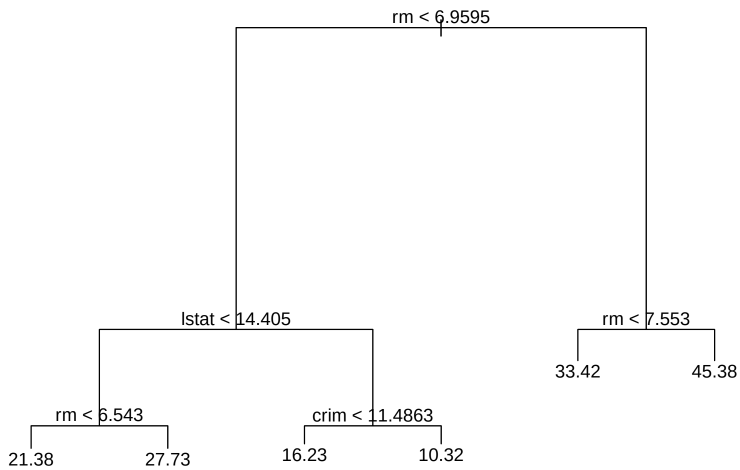


It looks like a tree with 6 leaves would work well.

Pruning

So we are pruning to a 6-node tree here:

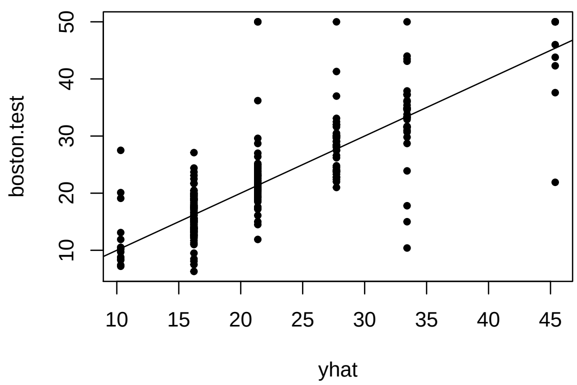
```
prune.boston=prune.tree(tree.boston,best=6)
plot(prune.boston)
text(prune.boston,pretty=0)
```



Test error for full tree

We calculate the test error for the pruned tree:

```
yhat=predict(prune.boston,newdata=Boston[-train,])
boston.test=Boston[-train,"medv"]
plot(yhat,boston.test, pch=20)
abline(0,1)
```




```
mean((yhat-boston.test)^2) # Calculate test MSE.
```

```
## [1] 35.16439
```

Bagging

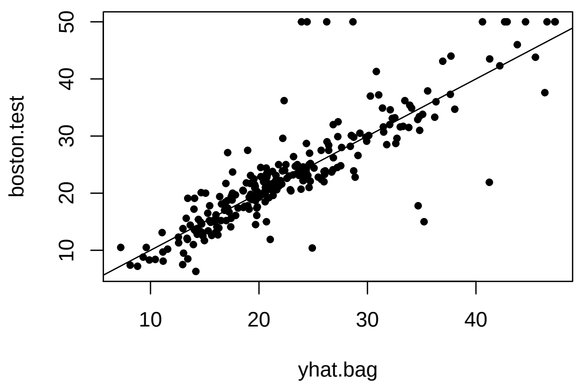
Remember: For bagging you can use the `randomForest()` function, but include all variables (here `mtry=13`).

```
library(randomForest)
set.seed(1)
bag.boston=randomForest(medv~.,data=Boston,subset=train,mtry=13,importance=TRUE)
bag.boston
```

```
##
## Call:
##  randomForest(formula = medv ~ ., data = Boston, mtry = 13, importance = TRUE,      subset = train)
##                Type of random forest: regression
##                      Number of trees: 500
## No. of variables tried at each split: 13
##
##           Mean of squared residuals: 11.39601
##                     % Var explained: 85.17
```

Test error for bagged tree

```
yhat.bag = predict(bag.boston,newdata=Boston[-train,])
plot(yhat.bag, boston.test,pch=20)
abline(0,1)
```




```
bag.boston=randomForest(medv~.,data=Boston,subset=train,mtry=13,ntree=25)
yhat.bag = predict(bag.boston,newdata=Boston[-train,])
mean((yhat.bag-boston.test)^2)
```

```
## [1] 23.66716
```

Random forest

Let's go from bagging to a random forest[1], using 6 randomly selected predictors for each tree:

```
set.seed(1)
rf.boston=randomForest(medv~.,data=Boston,subset=train,mtry=6,importance=TRUE)
yhat.rf = predict(rf.boston,newdata=Boston[-train,])
mean((yhat.rf-boston.test)^2)
```

```
## [1] 19.62021
```

It's interesting to see how the prediction error further decreased with respect to simple bagging.

Variable importance

```
importance(rf.boston)
```

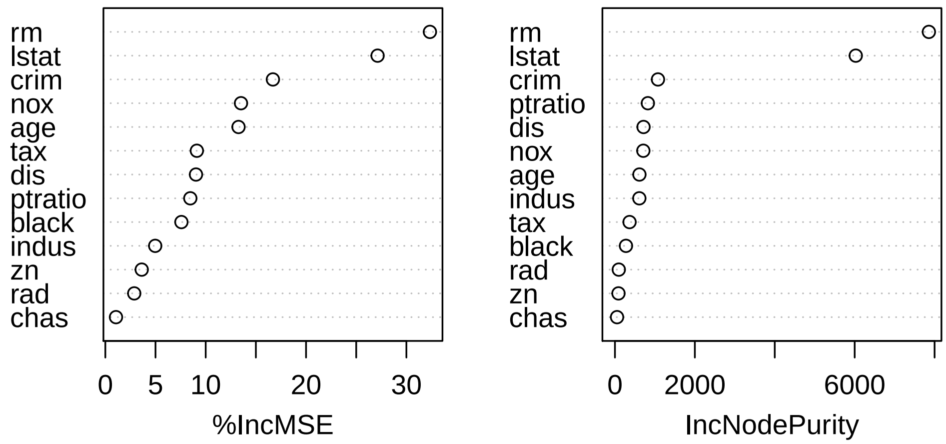
```
##           %IncMSE IncNodePurity
## crim    16.697017    1076.08786
## zn       3.625784      88.35342
## indus    4.968621     609.53356
## chas     1.061432      52.21793
## nox     13.518179     709.87339
## rm      32.343305    7857.65451
## age     13.272498     612.21424
## dis      9.032477     714.94674
## rad      2.878434      95.80598
## tax      9.118801     364.92479
## ptratio  8.467062     823.93341
## black    7.579482     275.62272
## lstat   27.129817    6027.63740
```

Interpretation?

And the variable importance plots

[1] n.b., why are we now speaking of a forest and no longer of a tree?

```r
varImpPlot(rf.boston,main="")
```



To understand what this means, please check again the meaning of the variables by typing `?Boston`.

Boosting

And finally, we are boosing the Boston trees! We boost with 5000 trees and allow the interaction depth (number of splits per tree) to be of degree 4:

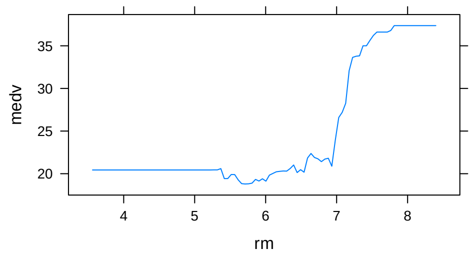
```r
library(gbm)
set.seed(1)
boost.boston=gbm(medv~.,data=Boston[train,],
                 distribution="gaussian",
                 n.trees=5000,interaction.depth=4)
summary(boost.boston,plotit=FALSE)
```

```
##             var    rel.inf
## rm           rm 43.9919329
## lstat     lstat 33.1216941
## crim       crim  4.2604167
## dis         dis  4.0111090
## nox         nox  3.4353017
## black     black  2.8267554
## age         age  2.6113938
## ptratio ptratio  2.5403035
## tax         tax  1.4565654
## indus     indus  0.8008740
## rad         rad  0.6546400
## zn           zn  0.1446149
## chas       chas  0.1443986
```

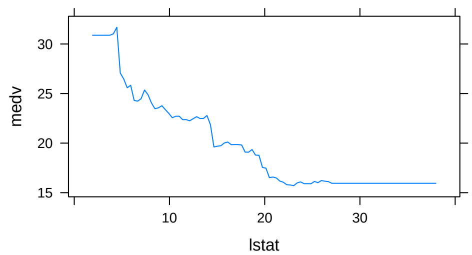
Partial dependency plots - integrating out other variables

`rm` (number of rooms) and `lstat` (% of lower status population) are the most important predictors. Partial dependency plots show the effect of individual predictors, integrated over the other predictors see @hastie_etal2009, Section 10.13.2.

```r
plot(boost.boston,i="rm",ylab="medv")
```




```
plot(boost.boston,i="lstat",ylab="medv")
```



Prediction on test set

- Calculate the MSE on the test set, first for the model with $\lambda = 0.001$ (default), then with $\lambda = 0.2$.
- We could have done cross-validation to find the best λ over a grid, but it seems not to make a big difference.

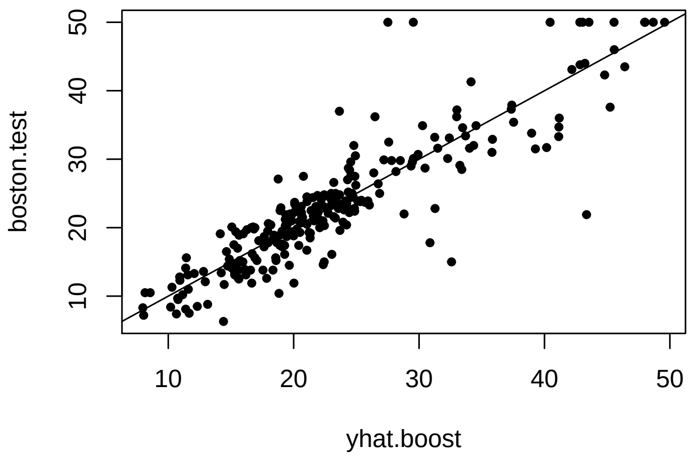
```
yhat.boost=predict(boost.boston,newdata=Boston[-train,],n.trees=5000)
mean((yhat.boost-boston.test)^2)
```

```
## [1] 18.84709
```

```
boost.boston=gbm(medv~.,data=Boston[train,],distribution="gaussian",
                 n.trees=5000,interaction.depth=4,shrinkage=0.2,verbose=F)
yhat.boost=predict(boost.boston,newdata=Boston[-train,],n.trees=5000)
mean((yhat.boost-boston.test)^2)
```

```
## [1] 18.33455
```

```
plot(yhat.boost,boston.test,pch=20)
abline(0,1)
```



Further reading

- Videoes on YouTube by the authors of ISL, Chapter 8, and corresponding slides.
- Solutions to exercises in the book, chapter 8

References