Statistics 452: Statistical Learning and Prediction

Chapter 8, Part 4: Regression Trees Lab

Brad McNeney

2018-11-02

Boston Data

- Recall the `Boston` dataset in which the response is the median house price in $1000 and and there are 13 predictors.
- As on the midterm, I've replaced the variable `black` by `predAA`, an indicator that takes value 1 if the town is predominantly African American and 0 otherwise.

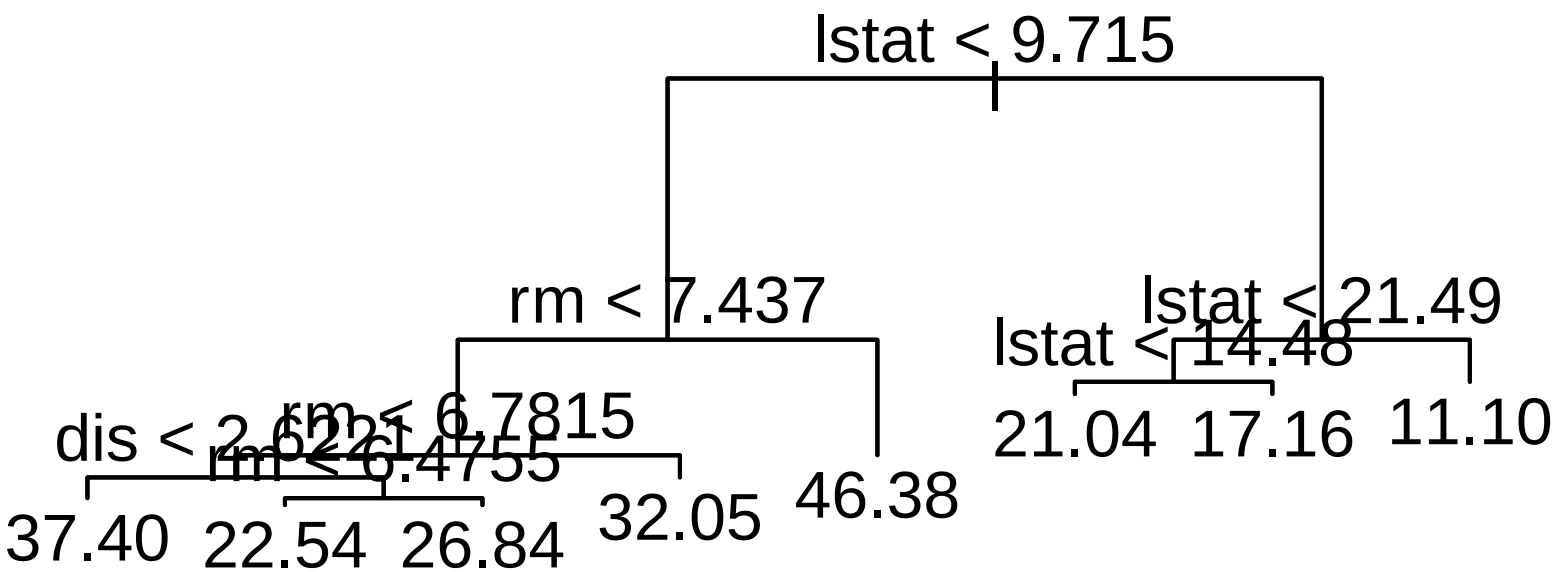
Training and Test Data

- Split the data in half for training and testing.

```
set.seed(1)
train <- sample(1:nrow(Boston),nrow(Boston)/2)
```

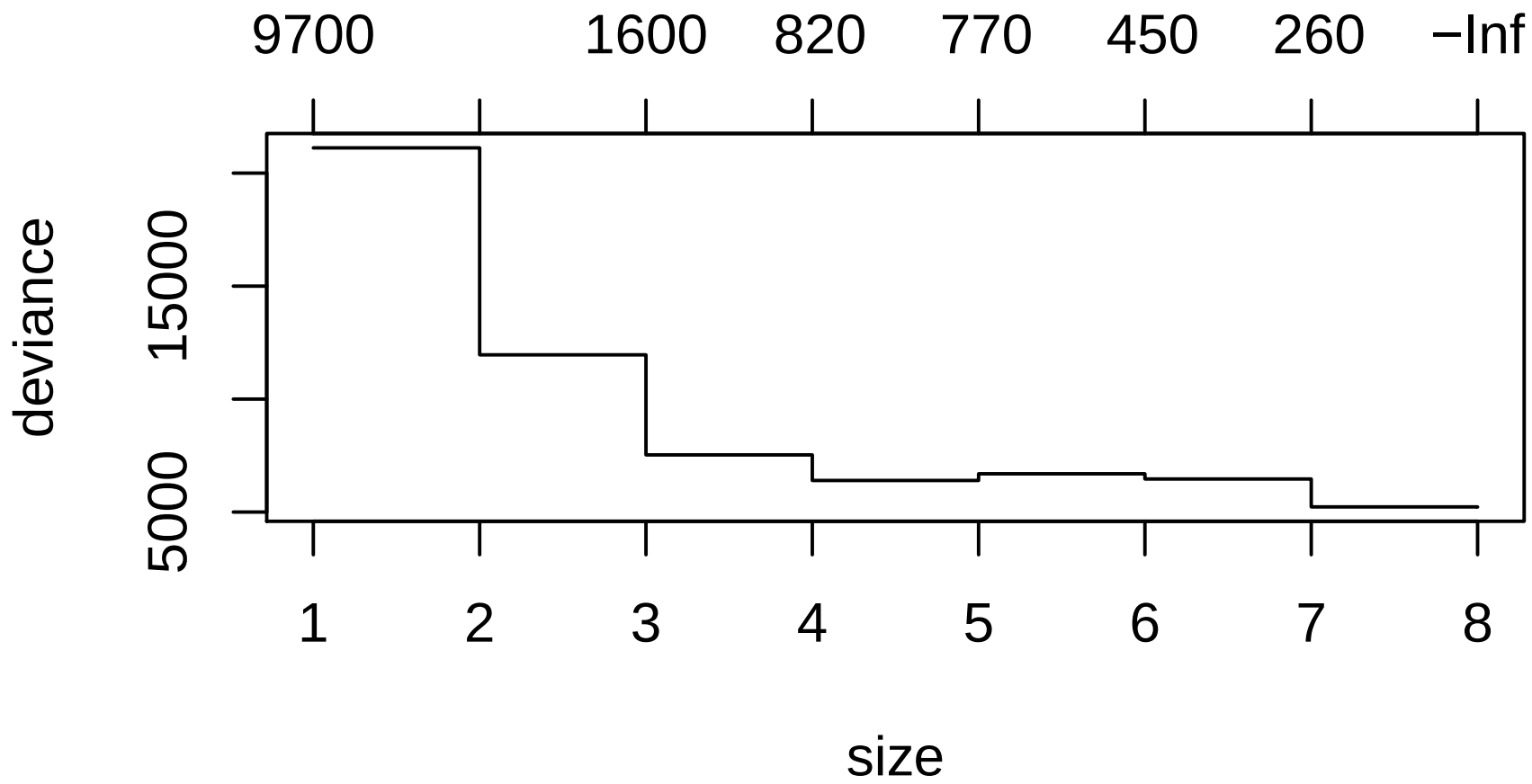
Regression Tree

```
library(tree)
tt <- tree(medv ~ ., data=Boston, subset=train)
plot(tt) # only rm, lsat, tax and dis used
text(tt)
```



Cross-Validation to Prune Tree

```
cvt <- cv.tree(tt)
plot(cvt) # No pruning required -- could use size 5
```




```
          # based on parsimony
```

Test Set Error

- Use the unpruned tree

```
yhat <- predict(tt,newdata=Boston[-train,])
y <- Boston[-train,"medv"]
mean((y-yhat)^2)
```

```
## [1] 25.04559
```

Bagging

```
library(randomForest)
set.seed(1) # for bootstrapping
btt <- randomForest(medv ~ ., data=Boston,subset=train,
                    mtry=13) # mtry=p for bagging
yhat <- predict(btt,newdata=Boston[-train,])
mean((y-yhat)^2)
```

```
## [1] 12.93789
```

Random Forest

```
set.seed(1) # for bootstrapping
rtt <- randomForest(medv ~ ., data=Boston,subset=train,
                    mtry=sqrt(13),importance=TRUE)
yhat <- predict(rtt,newdata=Boston[-train,])
mean((y-yhat)^2)
```

```
## [1] 11.5727
```

```
importance(rtt)
```

```
##           %IncMSE IncNodePurity
## crim    13.624844    1302.46550
## zn       3.063279     130.58952
## indus   10.751141    1233.05949
## chas     2.449582     115.44442
## nox     13.646660    1447.61798
## rm      28.120867    5476.38837
## age      9.398114     715.14472
## dis     13.575117    1382.47645
## rad      5.028803     162.43799
## tax      9.512444     642.01260
## ptratio 11.263839    1315.85311
## lstat   26.179457    6210.01315
## predAA   7.248321      60.13087
```

Boosting

```
library(gbm)
bott <- gbm(medv ~ ., data=Boston[train,],
                      distribution="gaussian",n.trees=1000) #M
yhat <- predict(bott,newdata=Boston[-train,],n.trees=1000)
mean((y-yhat)^2)
```

```
## [1] 13.81823
```

Boosting with Greater Interaction Depth

```
bott <- gbm(medv ~ ., data=Boston[train,], interaction.depth=4,
                      distribution="gaussian",n.trees=1000)
yhat <- predict(bott,newdata=Boston[-train,],n.trees=1000)
mean((y-yhat)^2)
```

```
## [1] 10.05349
```

Boosting with Less Shrinkage

```
bott <- gbm(medv ~ ., data=Boston[train,], interaction.depth=4,
            shrinkage=0.001, # decrease from default 0.1
            distribution="gaussian",n.trees=1000)
yhat <- predict(bott,newdata=Boston[-train,],n.trees=1000)
mean((y-yhat)^2)
```

```
## [1] 29.02575
```

Using the Test Set for Tuning

- Note: Using the test set to tune the boosting algorithm does not lead to valid test set estimates.
  - We are essentially fitting the test data.
- If we require a test set for tuning, it should not be the one we use to evaluate the tuned algorithm.
  - We should split the data into three parts: (i) training (can be the largest part), (ii) tuning test set, and (iii) test set.