Decision Trees for Regression

Karen Mazidi

Try linear regression on Boston

We get a correlation of 0.8 and a rmse of 5.36. So, the median home value was off by about $5,360.

```
library(tree)
library(MASS)
names(Boston)
```

```
##  [1] "crim"    "zn"      "indus"   "chas"    "nox"     "rm"      "age"
##  [8] "dis"     "rad"     "tax"     "ptratio" "black"   "lstat"   "medv"
```

```
# divide into train and test
set.seed(1234)
i <- sample(nrow(Boston), 0.8*nrow(Boston), replace = FALSE)
train <- Boston[i,]
test <- Boston[-i,]
lm1 <- lm(medv~., data=train)
summary(lm1)
```

```
##
## Call:
## lm(formula = medv ~ ., data = train)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -15.2049  -2.5360  -0.6665   1.8159  26.6255
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)  32.411618   5.713098   5.673 2.74e-08 ***
## crim         -0.110754   0.033940  -3.263 0.001199 **
## zn            0.045643   0.014616   3.123 0.001925 **
## indus        -0.021751   0.066244  -0.328 0.742828
## chas          2.878843   0.925365   3.111 0.002001 **
## nox         -17.056324   4.108702  -4.151 4.06e-05 ***
## rm            4.263719   0.469918   9.073  < 2e-16 ***
## age          -0.014451   0.014672  -0.985 0.325241
## dis          -1.485880   0.217730  -6.824 3.39e-11 ***
## rad           0.275831   0.071664   3.849 0.000139 ***
## tax          -0.012435   0.004041  -3.077 0.002235 **
## ptratio      -0.892542   0.144594  -6.173 1.68e-09 ***
## black         0.009214   0.002961   3.111 0.001999 **
## lstat        -0.405280   0.056926  -7.119 5.25e-12 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 4.618 on 390 degrees of freedom
## Multiple R-squared:  0.7589, Adjusted R-squared:  0.7509
## F-statistic: 94.43 on 13 and 390 DF,  p-value: < 2.2e-16
```

```r
pred <- predict(lm1, newdata=test)
cor_lm <- cor(pred, test$medv)
print(paste("cor = ", cor_lm))
```

```
## [1] "cor =  0.804577519382442"
```

```r
rmse_lm <- sqrt(mean((pred-test$medv)^2))
print(paste("rmse = ", rmse_lm))
```

```
## [1] "rmse =  5.3663515302298"
```

Using tree

Correlation was 0.88 and rmse was 4.19. The tree performed better than the linear regression model.

```r
tree1 <- tree(medv~., data=train)
summary(tree1)
```

```
##
## Regression tree:
## tree(formula = medv ~ ., data = train)
## Variables actually used in tree construction:
## [1] "lstat"   "rm"      "crim"    "ptratio" "nox"
## Number of terminal nodes:  11
## Residual mean deviance:  12.91 = 5073 / 393
## Distribution of residuals:
##      Min.   1st Qu.    Median      Mean   3rd Qu.      Max.
## -14.58000  -1.98500  -0.06153   0.00000   1.79600  16.84000
```

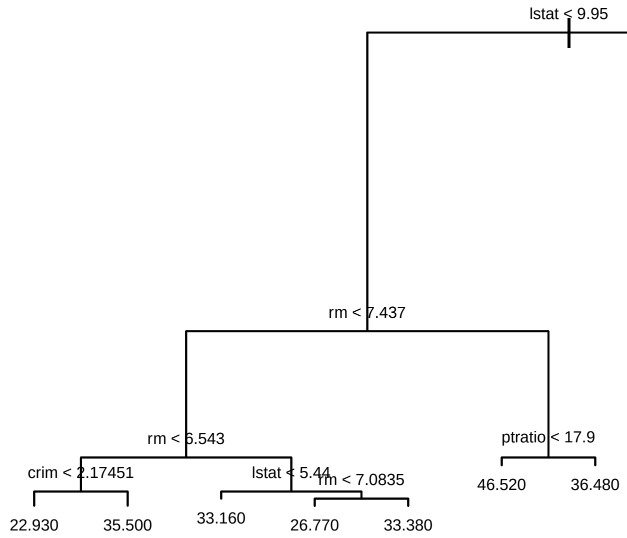
```r
pred <- predict(tree1, newdata=test)
cor_tree <- cor(pred, test$medv)
print(paste('correlation:', cor_tree))
```

```
## [1] "correlation: 0.88477699936987"
```

```r
rmse_tree <- sqrt(mean((pred-test$medv)^2))
print(paste('rmse:', rmse_tree))
```

```
## [1] "rmse: 4.1984948374158"
```

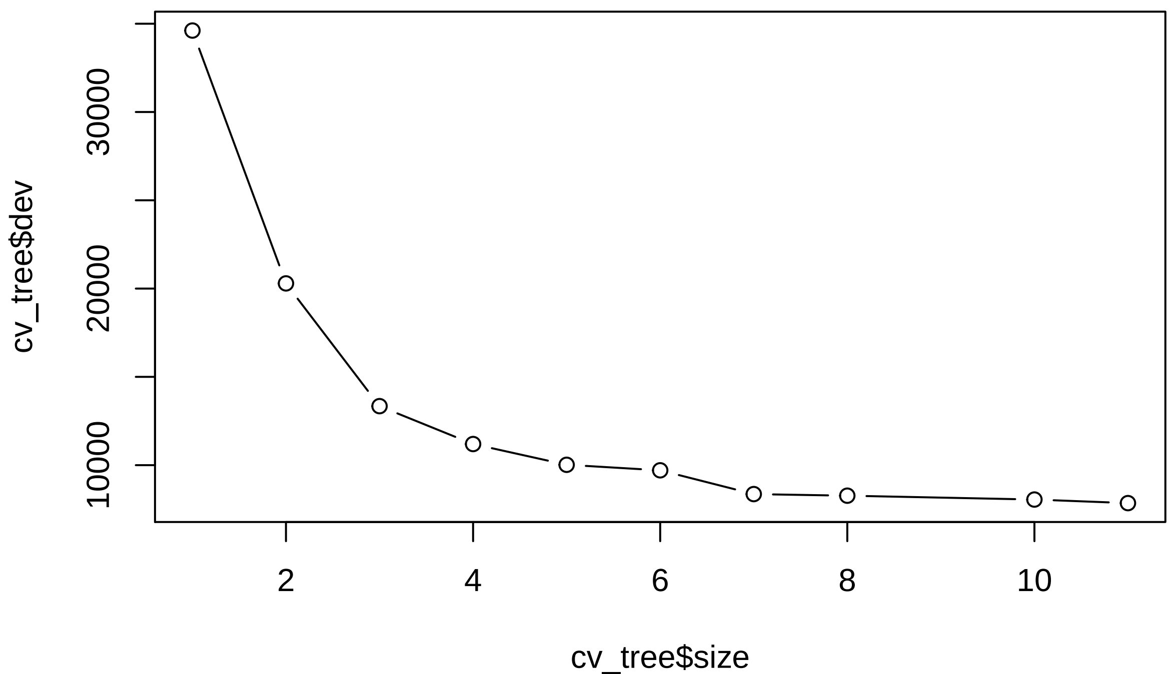
```r
plot(tree1)
text(tree1, cex=0.5, pretty=0)
```



cross validation

The plot shows the deviance for various tree sizes. The full tree with 11 terminal (leaf) nodes has the smallest deviance, but it might overfit the data. At the other extreme, a tree with one node has the highest deviance. A happy medium is somewhere in the bend of the curve.

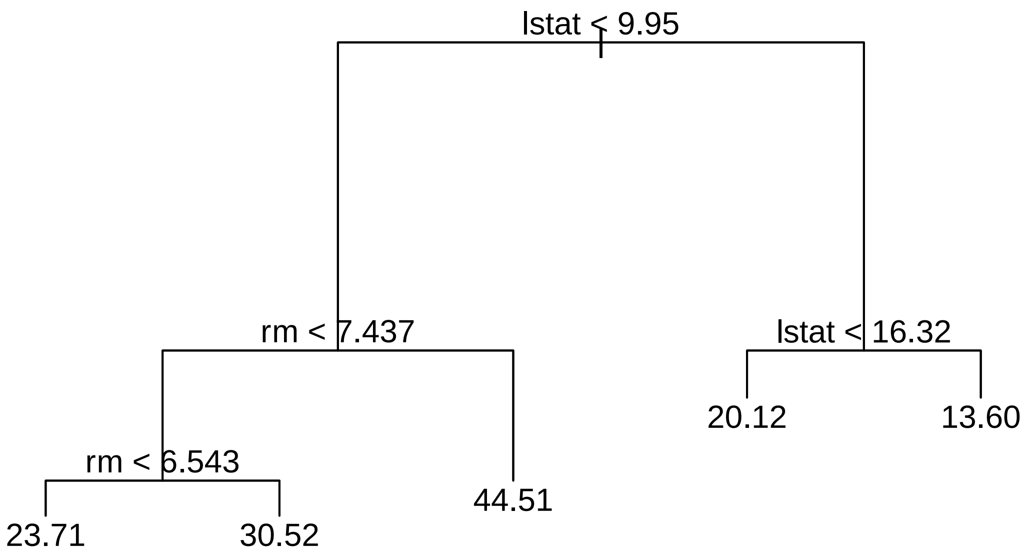
```
cv_tree <- cv.tree(tree1)
plot(cv_tree$size, cv_tree$dev, type='b')
```



prune the tree

The tree is pruned to 5 terminal nodes, and then plotted.

```
tree_pruned <- prune.tree(tree1, best=5)
plot(tree_pruned)
text(tree_pruned, pretty=0)
```



test on the pruned tree

The correlation and rmse are not as good as the unpruned tree but still slightly better than the linear regression model. In this case pruning did not improve results on the test data but the tree is simpler and easier to interpret.

```
pred_pruned <- predict(tree_pruned, newdata=test)
cor_pruned <- cor(pred_pruned, test$medv)
rmse_pruned <- sqrt(mean((pred_pruned-test$medv)^2))
print(paste("cor of pruned tree = ", cor_pruned))
```

```
## [1] "cor of pruned tree =  0.814280213110326"
```

```
print(paste("rmse of pruned tree = ", rmse_pruned))
```

```
## [1] "rmse of pruned tree =  5.27185392365713"
```

Random Forest

The importance=TRUE argument tells the algorithm to consider the importance of predictors.

```
library(randomForest)
```

```
## randomForest 4.6-14

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

```
set.seed(1234)
rf <- randomForest(medv~., data=train, importance=TRUE)
rf
```

```
##
## Call:
##  randomForest(formula = medv ~ ., data = train, importance = TRUE)
##                Type of random forest: regression
##                      Number of trees: 500
## No. of variables tried at each split: 4
##
##           Mean of squared residuals: 10.66999
##                     % Var explained: 87.5
```

predict on the random forest

The random forest model got improved results over any of the previous models in this notebook.

```r
pred_rf <- predict(rf, newdata=test)
cor_rf <- cor(pred_rf, test$medv)
print(paste('corr:', cor_rf))
```

```
## [1] "corr: 0.931673264971012"
```

```r
rmse_rf <- sqrt(mean((pred_rf-test$medv)^2))
print(paste('rmse:', rmse_rf))
```

```
## [1] "rmse: 3.43580269934469"
```

bagging

Setting mtry to the number of predictors, p, will result in bagging

```r
bag <- randomForest(medv~., data=train, mtry=13)
bag
```

```
##
## Call:
##  randomForest(formula = medv ~ ., data = train, mtry = 13)
##                Type of random forest: regression
##                      Number of trees: 500
## No. of variables tried at each split: 13
##
##           Mean of squared residuals: 9.849697
##                     % Var explained: 88.46
```

predict

Our results for bagging were slightly lower than for the random forest.

```r
pred_bag <- predict(bag, newdata=test)
cor_bag <- cor(pred_bag, test$medv)
print(paste('corr:', cor_bag))
```

```
## [1] "corr: 0.924091397871912"
```

```r
rmse_bag <- sqrt(mean((pred_bag-test$medv)^2))
print(paste('rmse:', rmse_bag))
```

```
## [1] "rmse: 3.49118767038579"
```