NBA 4920/6921 Lecture 16

Tree Methods: Regression Application

Murat Unal

10/26/2021

```r
rm(list=ls())
options(digits = 3, scipen = 999)
library(tidyverse)
library(ISLR)
library(cowplot)
library(stargazer)
library(lmtest)
library(sandwich)
library(MASS)
library(jtools)
library(caret)
library(rpart)
library(rpart.plot)
library(ROCR)
set.seed(2)
```

```r
Hitters <- ISLR::Hitters
Hitters <- na.omit(Hitters)
```

- Seperate train and test data

```r
train = sample(1:nrow(Hitters), 0.7*nrow(Hitters))
```

Regression Trees

- We can fit a regression tree using `rpart` and then visualize it using `rpart.plot`
- we need to set `method = "anova"`, for classification set `method="class"`
- The main tuning parameter is `cp`, the complexity parameter

```r
hit_tree = rpart(Salary ~ .,data = Hitters[train,],
                 method="anova")
```

```
hit_tree
```

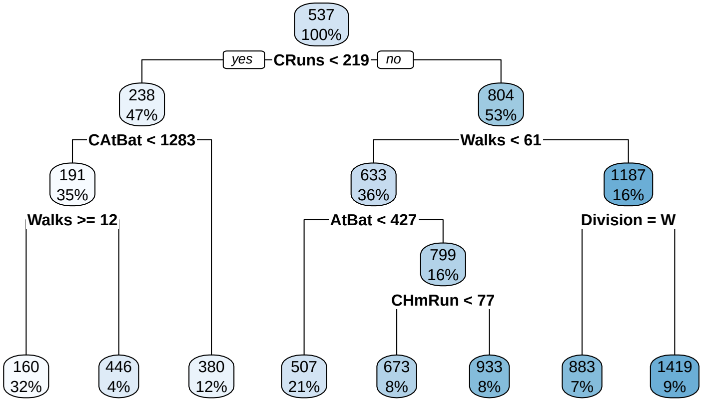
```
n= 184

node), split, n, deviance, yval
      * denotes terminal node

 1) root 184 40400000  537
   2) CRuns< 218 87  5140000  238
     4) CAtBat< 1.28e+03 65  4150000  191
       8) Walks>=11.5 58   311000  160 *
       9) Walks< 11.5 7  3330000  446 *
     5) CAtBat>=1.28e+03 22   401000  380 *
   3) CRuns>=218 97 20500000  804
     6) Walks< 61 67  5700000  633
      12) AtBat< 426 38  1720000  507 *
      13) AtBat>=426 29  2570000  799
        26) CHmRun< 76.5 15   335000  673 *
        27) CHmRun>=76.5 14  1750000  933 *
     7) Walks>=61 30  8480000 1190
```

- We can visualize our tree model with `rpart.plot()`

```
rpart.plot(hit_tree)
```



- Behind the scenes `rpart()` is automatically applying a range of cost complexity α values to prune the tree.
- To compare the error for each α value, it performs a 10-fold CV (by default)

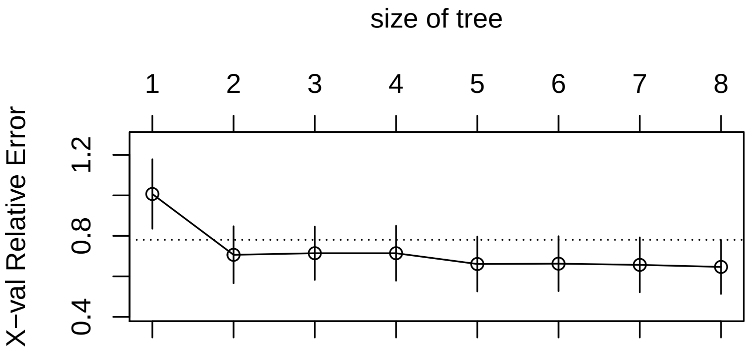
```
hit_tree$cptable
```

```
      CP nsplit rel error xerror  xstd
1 0.3639      0     1.000  1.007 0.172
2 0.1576      1     0.636  0.707 0.141
3 0.0524      2     0.479  0.714 0.132
4 0.0348      3     0.426  0.714 0.136
5 0.0145      4     0.391  0.661 0.136
6 0.0126      5     0.377  0.663 0.136
7 0.0122      6     0.364  0.657 0.136
8 0.0100      7     0.352  0.647 0.134
```

- Here we don't find much improvement after 2 terminal nodes

- Notice the dashed line which goes through the point $|T| = 2$.
- It's common to instead use the smallest tree within 1 standard error (SE) of the minimum CV error (this is called the 1-SE rule).
- Thus, we could use a tree with just 2 terminal nodes and reasonably expect to experience similar results within a small margin of error.

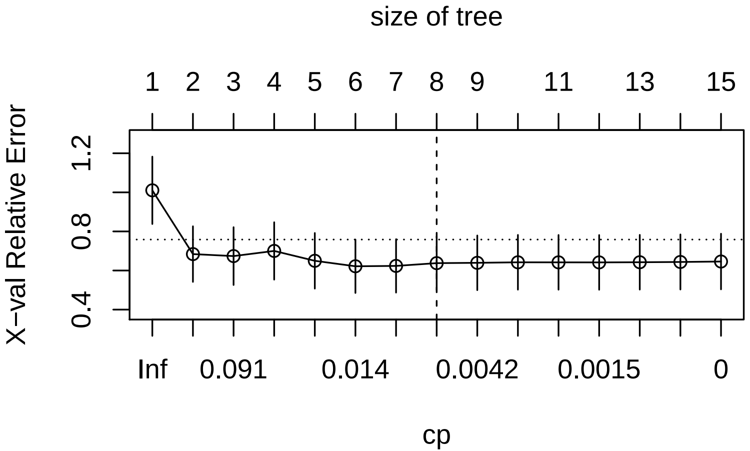
```
plotcp(hit_tree)
```



- To illustrate the point of selecting a tree with 8 terminal nodes (or 2 if you go by the 1-SE rule), we can force `rpart()` to generate a full tree by setting `cp = 0` (no penalty results in a fully grown tree).

```
hit_tree2 = rpart(Salary ~ .,data = Hitters[train,],
          method="anova",control = list(cp=0,xval=10))
```

```r
plotcp(hit_tree2)
abline(v = 8, lty = "dashed")
```



- Make predictions on the test data with best cp

```
pred.tree <- predict(hit_tree, Hitters[-train,])

rmse.min.cp <- sqrt(mean((Hitters[-train,"Salary"]-
                          pred.tree)^2))

rmse.min.cp
```

```
[1] 309
```

- `maxdepth`: is the maximum number of internal nodes between the root node and the terminal nodes.
- We could obtain a tree with 2 terminal nodes by setting `maxdepth=1`.

```
hit_tree1se <- rpart(Salary ~ .,data = Hitters[train,],
          method="anova",maxdepth=1)

hit_tree1se
```

```
n= 184

node), split, n, deviance, yval
      * denotes terminal node

1) root 184 40400000 537
  2) CRuns< 218 87  5140000 238 *
  3) CRuns>=218 97 20500000 804 *
```

- Predictions and rmse

```
pred.tree1se <- predict(hit_tree1se, Hitters[-train,])
rmse.1se.cp <- sqrt(mean((Hitters[-train,"Salary"]-
                           pred.tree1se)^2))
rmse.1se.cp
```

```
[1] 323
```

Exercise

- Train a regression tree to predict the crime rates (`crim`) in the Boston dataset.

```
data_test <- read.csv("boston_test.csv")
data_train <- read.csv("boston_train.csv")
```

- Obtain the tree

```
bos_tree = rpart(crim ~ .,data = data_train,
                 method="anova")

bos_tree
```

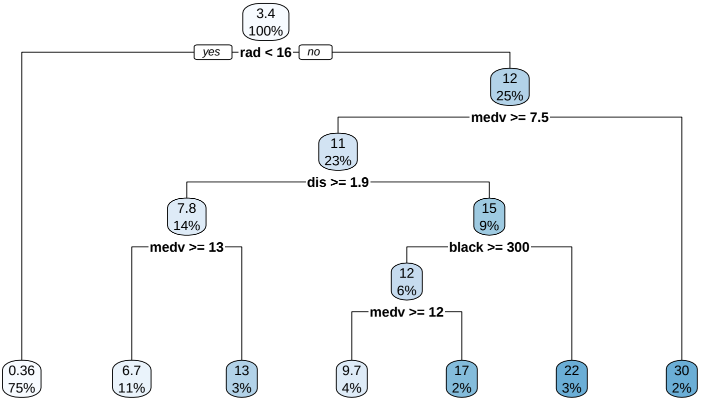
```
n= 405

node), split, n, deviance, yval
      * denotes terminal node

 1) root 405 21800.0  3.37
   2) rad< 16 303    91.0  0.36 *
   3) rad>=16 102 10800.0 12.30
     6) medv>=7.45 94  5400.0 10.80
      12) dis>=1.87 56   750.0  7.82
        24) medv>=12.6 45   378.0  6.66 *
        25) medv< 12.6 11    64.3 12.60 *
      13) dis< 1.87 38  3410.0 15.20
        26) black>=300 26  1010.0 12.30
```

- Visualize the tree

```
rpart.plot(bos_tree)
```



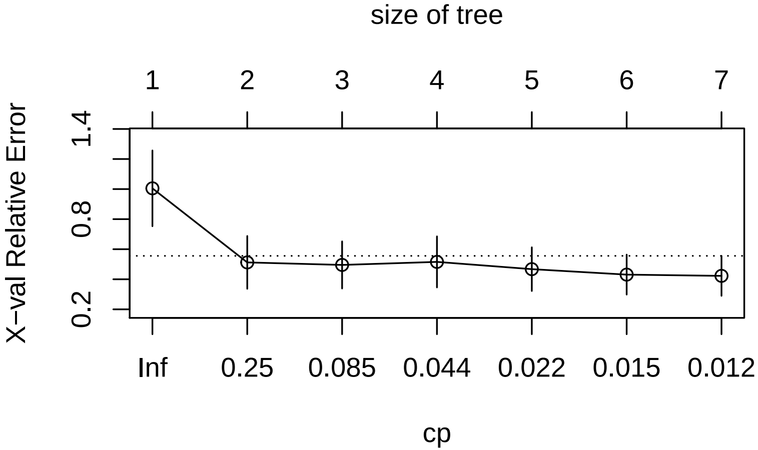
- Examine the cross validation errors against cp

```
bos_tree$cptable
```

```
      CP nsplit rel error xerror  xstd
1 0.5010      0     1.000  1.005 0.252
2 0.1253      1     0.499  0.512 0.176
3 0.0571      2     0.374  0.495 0.157
4 0.0337      3     0.317  0.516 0.170
5 0.0150      4     0.283  0.467 0.145
6 0.0141      5     0.268  0.431 0.133
7 0.0100      6     0.254  0.423 0.133
```

- Plot `cp`

```
plotcp(bos_tree)
```



- Obtain the 1se tree

```
bos_tree1se = rpart(crim ~ .,data = data_train,
                    method="anova", maxdepth=1)
bos_tree1se
```

```
n= 405

node), split, n, deviance, yval
      * denotes terminal node

1) root 405 21800  3.37
  2) rad< 16 303    91  0.36 *
  3) rad>=16 102 10800 12.30 *
```

- Predictions and rmse for min cp and 1se cp

```
pred.bos.tree <- predict(bos_tree, data_test)
rmse.min.cp <- sqrt(mean((data_test$crim-
                            pred.bos.tree)^2))
rmse.min.cp
```

```
[1] 9.48
```

```
pred.bos.tree1se <- predict(bos_tree1se, data_test)
rmse.1se.cp <- sqrt(mean((data_test$crim-
                            pred.bos.tree1se)^2))
rmse.1se.cp
```

```
[1] 10.7
```