SurvMeth 640 Assignment 3

Cheng, Chia Wen

2023-03-13

Setup

```
library(mlbench)
library(rpart)
library(partykit)
library(caret)
```

Data

In this notebook, we use the Boston Housing data set (again). "This dataset contains information collected by the U.S Census Service concerning housing in the area of Boston Mass. It was obtained from the StatLib archive (http://lib.stat.cmu.edu/datasets/boston), and has been used extensively throughout the literature to benchmark algorithms."

Source: https://www.cs.toronto.edu/~delve/data/boston/bostonDetail.html

```
data(BostonHousing2)
head(BostonHousing2)
```

```
##         town tract      lon     lat medv cmedv    crim zn indus chas   nox
## 1     Nahant  2011 -70.9550 42.2550 24.0  24.0 0.00632 18  2.31    0 0.538
## 2 Swampscott  2021 -70.9500 42.2875 21.6  21.6 0.02731  0  7.07    0 0.469
## 3 Swampscott  2022 -70.9360 42.2830 34.7  34.7 0.02729  0  7.07    0 0.469
## 4 Marblehead  2031 -70.9280 42.2930 33.4  33.4 0.03237  0  2.18    0 0.458
## 5 Marblehead  2032 -70.9220 42.2980 36.2  36.2 0.06905  0  2.18    0 0.458
## 6 Marblehead  2033 -70.9165 42.3040 28.7  28.7 0.02985  0  2.18    0 0.458
##      rm  age    dis rad tax ptratio      b lstat
## 1 6.575 65.2 4.0900   1 296    15.3 396.90  4.98
## 2 6.421 78.9 4.9671   2 242    17.8 396.90  9.14
## 3 7.185 61.1 4.9671   2 242    17.8 392.83  4.03
## 4 6.998 45.8 6.0622   3 222    18.7 394.63  2.94
## 5 7.147 54.2 6.0622   3 222    18.7 396.90  5.33
## 6 6.430 58.7 6.0622   3 222    18.7 394.12  5.21
```

```
names(BostonHousing2)
```

```
##  [1] "town"    "tract"   "lon"     "lat"     "medv"    "cmedv"   "crim"
##  [8] "zn"      "indus"   "chas"    "nox"     "rm"      "age"     "dis"
## [15] "rad"     "tax"     "ptratio" "b"       "lstat"
```

Drop some variables that are not needed.

```
BostonHousing2$town <- NULL
BostonHousing2$tract <- NULL
BostonHousing2$cmedv <- NULL
```

Split the data into a train and test set.

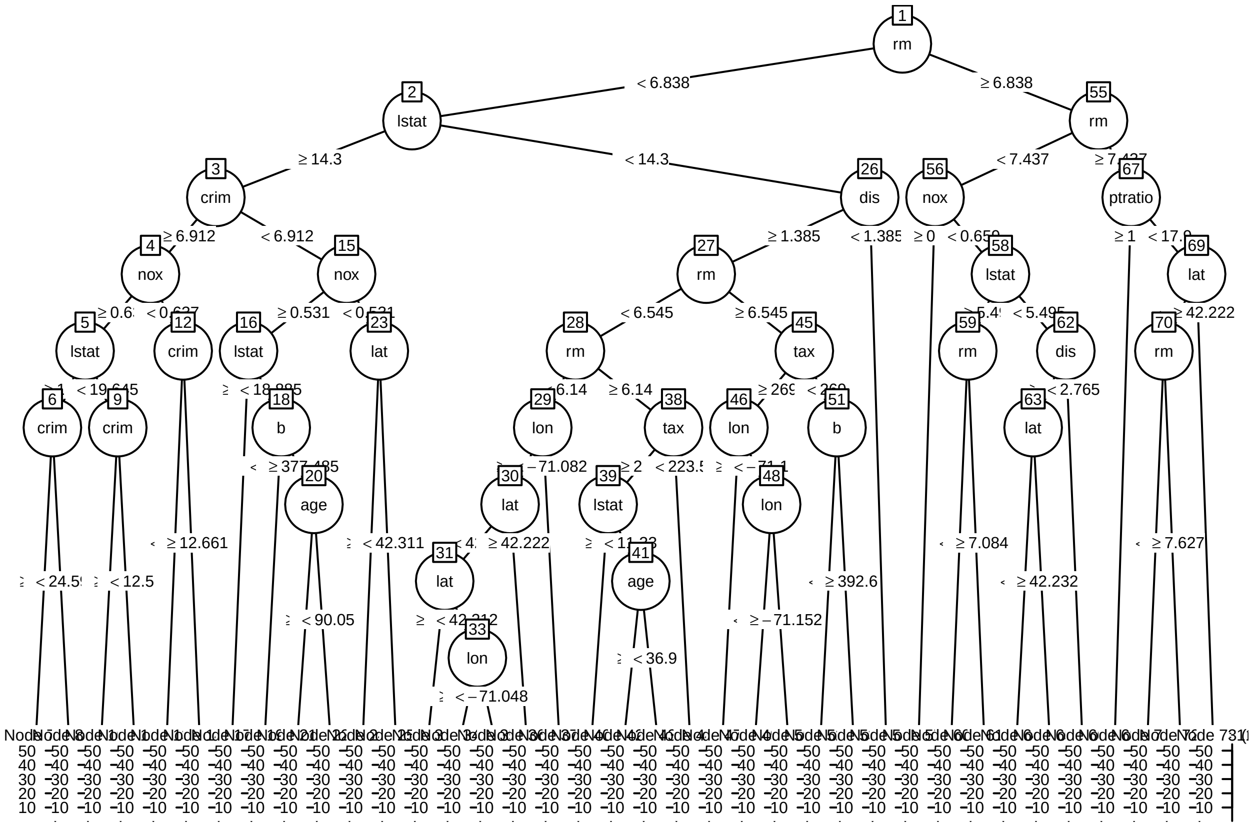
```
set.seed(9384)

train <- sample(1:nrow(BostonHousing2), 0.8*nrow(BostonHousing2))
boston_train <- BostonHousing2[train,]
boston_test <- BostonHousing2[-train,]
```

CART

As before, we are interested in training a model for predicting the median home values (`medv`), using all features available. However, in this notebook we want to use CART as the prediction method. Make sure to grow a large tree, since we want to prune it back later.

```
# build a larger tree
tree2 <- rpart(medv ~ ., data = boston_train, method = "anova",
               control = rpart.control(minsplit = 10, # minimal obs in a node
                                       minbucket = 3, # minimal obs in any terminal node
                                       cp = 0.001, # min improvement through splitting
                                       maxdepth = 30 # maximum tree depth
                                       ))
party_tree2 <- as.party(tree2)
plot(party_tree2, gp = gpar(fontsize = 6))
```



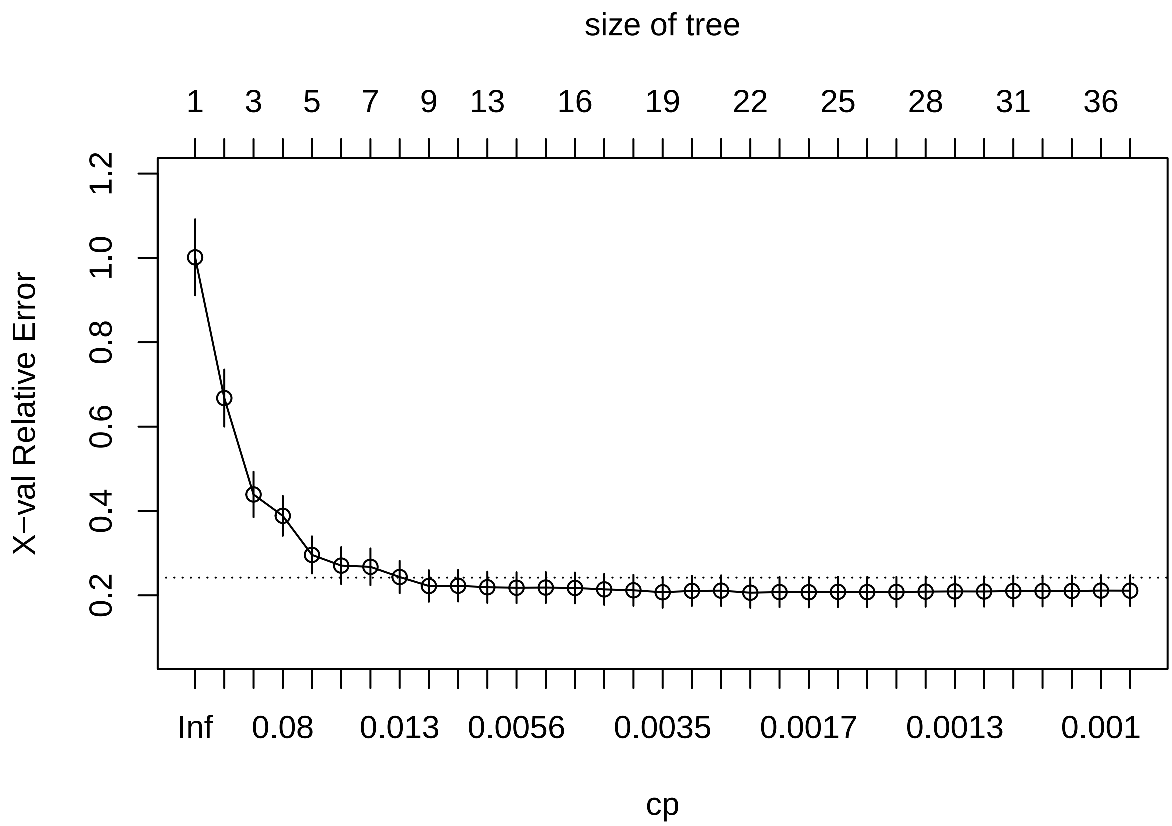
This large tree is likely to overfit and will not generalize well to new data. Therefore, we use `printcp()` and `plotcp()` that help us to determine the best subtree.

```
printcp(tree2)
```

```
##
## Regression tree:
## rpart(formula = medv ~ ., data = boston_train, method = "anova",
##     control = rpart.control(minsplit = 10, minbucket = 3, cp = 0.001,
##         maxdepth = 30))
##
## Variables actually used in tree construction:
##  [1] age     b       crim    dis     lat     lon     lstat   nox     ptratio
## [10] rm      tax
##
## Root node error: 35535/404 = 87.959
##
## n= 404
##
##           CP nsplit rel error  xerror     xstd
## 1  0.4565952      0  1.000000 1.00141 0.090237
## 2  0.1596255      1  0.543405 0.66770 0.067684
## 3  0.1040341      2  0.383779 0.43916 0.054033
## 4  0.0615025      3  0.279745 0.38854 0.047334
## 5  0.0290994      4  0.218243 0.29586 0.044003
## 6  0.0259984      5  0.189143 0.27048 0.043780
```

```
## 7  0.0252640      6  0.163145 0.26759 0.043610
## 8  0.0069852      7  0.137881 0.24340 0.038564
## 9  0.0068438      8  0.130896 0.22216 0.037322
## 10 0.0063681     11  0.110364 0.22275 0.037401
## 11 0.0056780     12  0.103996 0.21917 0.037047
## 12 0.0055366     13  0.098318 0.21803 0.036753
## 13 0.0051020     14  0.092781 0.21845 0.036646
## 14 0.0048107     15  0.087679 0.21748 0.036644
## 15 0.0044513     16  0.082868 0.21411 0.036619
## 16 0.0040860     17  0.078417 0.21196 0.036880
## 17 0.0030686     18  0.074331 0.20729 0.036806
## 18 0.0030198     19  0.071263 0.21055 0.035551
## 19 0.0025376     20  0.068243 0.21114 0.036283
## 20 0.0018645     21  0.065705 0.20622 0.035827
## 21 0.0018179     22  0.063841 0.20773 0.035903
## 22 0.0016243     23  0.062023 0.20725 0.035896
## 23 0.0015804     24  0.060398 0.20828 0.035891
## 24 0.0014355     25  0.058818 0.20760 0.035863
## 25 0.0013252     26  0.057383 0.20801 0.035793
## 26 0.0013069     27  0.056057 0.20879 0.035884
## 27 0.0012211     28  0.054750 0.20926 0.035890
## 28 0.0011483     29  0.053529 0.20908 0.035896
## 29 0.0011469     30  0.052381 0.21017 0.036527
## 30 0.0011407     31  0.051234 0.21017 0.036527
## 31 0.0010317     32  0.050093 0.21037 0.036534
## 32 0.0010166     35  0.046998 0.21123 0.036584
## 33 0.0010000     36  0.045982 0.21108 0.036578
```

```
plotcp(tree2)
```



Briefly summarize the output of these two functions. What is plotted/printed here?

We prune the tree to avoid overfitting of the data. With the complexity parameter of 0.0018645, we get the least cross validated error of 0.24890. This is when 21 splits are applied to the decision tree. The dotted line in the plot points out where 'xerror' equals to 0.3. It shows that trees with more than 7 splits (not including 7) have 'xerrors' lower than 0.3 and the change is minor, but a smaller 'cp' does not gaurantee a lower 'xerror.'

end

On this basis, we are interested in picking the cp value that is associated with the smallest CV error. Store this value in an object.

```
minx <- which.min(tree2$cptable[,"xerror"])
mincp <- tree2$cptable[minx,"CP"]
mincp # 0.001864483
```

```
## [1] 0.001864483
```

In addition, we could also pick the best subtree based on the 1-SE rule. Again, store the corresponding cp value in an object.

```r
minx <- which.min(tree2$cptable[,"xerror"])
minxse <- tree2$cptable[minx,"xerror"] + tree2$cptable[minx,"xstd"]
minse <- which(tree2$cptable[1:minx,"xerror"] < minxse)
mincp2 <- tree2$cptable[minse[1],"CP"]
mincp2 # 0.006843847
```

```
## [1] 0.006843847
```

Now we can get the best subtree with the `prune()` function. First based on the smallest CV error...

```r
p_tree_small_CV <- prune(tree2, cp = mincp)
p_tree_small_CV
```

```
## n= 404
##
## node), split, n, deviance, yval
##       * denotes terminal node
##
##   1) root 404 35535.34000 22.686140
##     2) rm< 6.8375 330 13274.68000 19.685150
##       4) lstat>=14.3 139  2657.36200 14.825180
##         8) crim>=6.91188 59   763.50750 11.649150
##          16) nox>=0.6365 47   440.21320 10.659570
##            32) lstat>=19.645 35   199.06570  9.542857 *
##            33) lstat< 19.645 12    70.19667 13.916670 *
##          17) nox< 0.6365 12    97.00250 15.525000 *
##         9) crim< 6.91188 80   859.79550 17.167500
##          18) nox>=0.531 64   535.68480 16.373440
##            36) lstat>=18.885 21   138.89810 14.123810 *
##            37) lstat< 18.885 43   238.60650 17.472090 *
##          19) nox< 0.531 16   122.33940 20.343750 *
##       5) lstat< 14.3 191  4944.96800 23.221990
##        10) dis>=1.38485 188  2759.45500 22.794680
##          20) rm< 6.5445 151  1339.51900 21.697350
##            40) rm< 6.1405 76   478.05740 20.423680
##              80) lon>=-71.08215 59   325.12030 19.838980 *
##              81) lon< -71.08215 17    62.76235 22.452940 *
##            41) rm>=6.1405 75   613.23920 22.988000
##              82) tax>=223.5 71   380.33320 22.657750
##               164) lstat>=11.33 14    31.11429 20.157140 *
##               165) lstat< 11.33 57   240.17510 23.271930 *
##              83) tax< 223.5 4    87.71000 28.850000 *
##          21) rm>=6.5445 37   496.07300 27.272970
##            42) tax>=269 26   230.71120 25.773080
##              84) lon>=-71.1 15    59.25333 24.033330 *
##              85) lon< -71.1 11    64.14727 28.145450 *
##            43) tax< 269 11    68.61636 30.818180 *
##        11) dis< 1.38485 3     0.00000 50.000000 *
##     3) rm>=6.8375 74  6035.39900 36.068920
##       6) rm< 7.437 50  1947.16100 31.172000
##        12) nox>=0.659 3    27.92000 14.400000 *
##        13) nox< 0.659 47  1021.47500 32.242550
```

```
##          26) lstat>=5.495 26   368.39880 30.265380
##            52) rm< 7.0835 14    72.85429 27.357140 *
##            53) rm>=7.0835 12    38.98917 33.658330 *
##          27) lstat< 5.495 21   425.59810 34.690480
##            54) dis>=2.7648 18   102.14940 33.294440 *
##            55) dis< 2.7648 3    77.88667 43.066670 *
##       7) rm>=7.437 24   391.34960 46.270830
##        14) ptratio>=17.9 4    46.94750 40.125000 *
##        15) ptratio< 17.9 20   163.10000 47.500000 *
```

...and now based on the 1-SE rule.

```
p_tree_one_SE <- prune(tree2, cp = mincp2)
p_tree_one_SE
```

```
## n= 404
##
## node), split, n, deviance, yval
##       * denotes terminal node
##
##  1) root 404 35535.3400 22.68614
##    2) rm< 6.8375 330 13274.6800 19.68515
##      4) lstat>=14.3 139  2657.3620 14.82518
##        8) crim>=6.91188 59   763.5075 11.64915 *
##        9) crim< 6.91188 80   859.7955 17.16750 *
##      5) lstat< 14.3 191  4944.9680 23.22199
##       10) dis>=1.38485 188  2759.4550 22.79468
##         20) rm< 6.5445 151  1339.5190 21.69735
##           40) rm< 6.1405 76   478.0574 20.42368 *
##           41) rm>=6.1405 75   613.2392 22.98800 *
##         21) rm>=6.5445 37   496.0730 27.27297 *
##       11) dis< 1.38485 3     0.0000 50.00000 *
##    3) rm>=6.8375 74  6035.3990 36.06892
##      6) rm< 7.437 50  1947.1610 31.17200
##       12) nox>=0.659 3    27.9200 14.40000 *
##       13) nox< 0.659 47  1021.4750 32.24255 *
##      7) rm>=7.437 24   391.3496 46.27083 *
```

Now, plot the smaller tree.

```
tree1 <- rpart(medv ~ ., data = boston_train)
tree1
```

```
## n= 404
##
## node), split, n, deviance, yval
##       * denotes terminal node
##
##  1) root 404 35535.3400 22.68614
##    2) rm< 6.8375 330 13274.6800 19.68515
##      4) lstat>=14.3 139  2657.3620 14.82518
##        8) crim>=6.91188 59   763.5075 11.64915 *
##        9) crim< 6.91188 80   859.7955 17.16750 *
```

```
##      5) lstat< 14.3 191  4944.9680 23.22199
##       10) rm< 6.5445 153  2920.6570 22.06732
##         20) lstat>=9.66 82   502.1888 20.58049 *
##         21) lstat< 9.66 71  2027.8330 23.78451
##           42) indus< 14.48 64   586.9511 22.88281 *
##           43) indus>=14.48 7   913.0943 32.02857 *
##       11) rm>=6.5445 38   998.9982 27.87105 *
##    3) rm>=6.8375 74  6035.3990 36.06892
##      6) rm< 7.437 50  1947.1610 31.17200
##       12) lstat>=9.65 8   409.2750 23.17500 *
##       13) lstat< 9.65 42   928.8190 32.69524 *
##      7) rm>=7.437 24   391.3496 46.27083 *
```

```
summary(tree1)
```

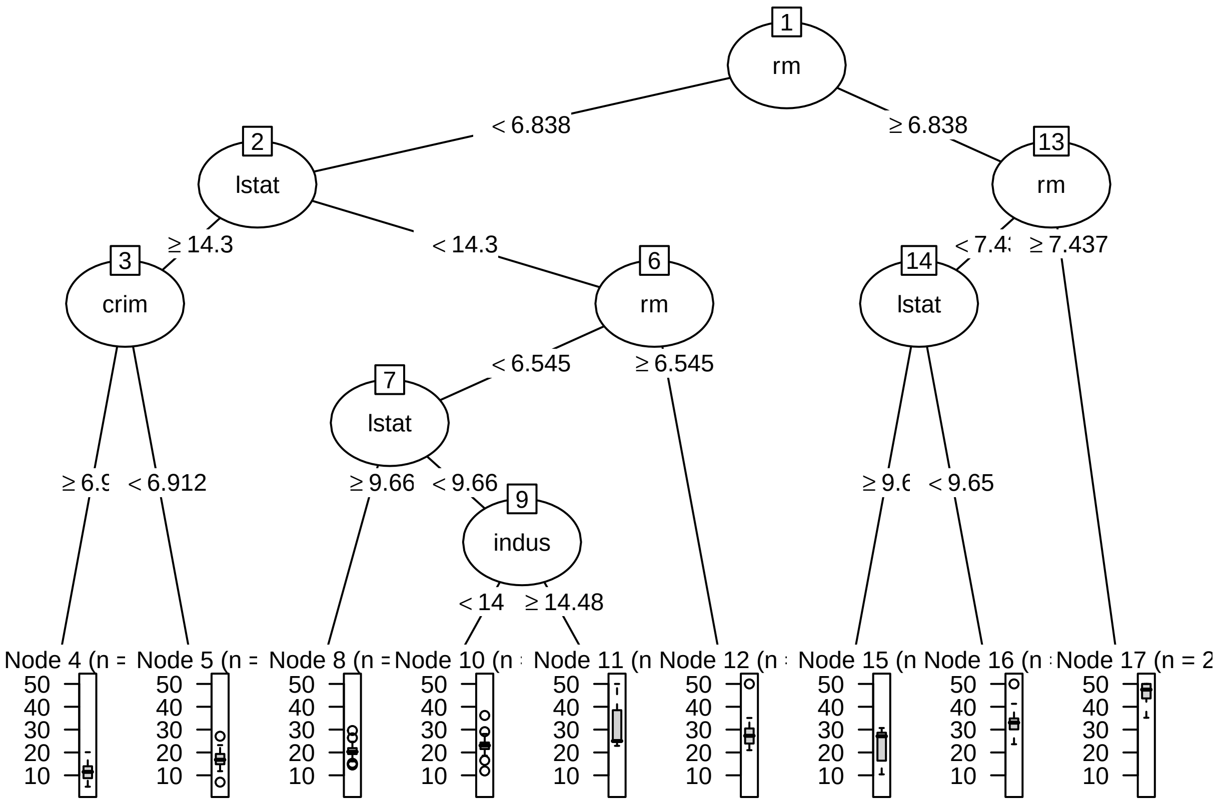
```
## Call:
## rpart(formula = medv ~ ., data = boston_train)
##   n= 404
##
##           CP nsplit rel error    xerror       xstd
## 1 0.45659520      0 1.0000000 1.0058112 0.09072378
## 2 0.15962553      1 0.5434048 0.6355163 0.06498087
## 3 0.10403412      2 0.3837793 0.4084787 0.04991205
## 4 0.02909945      3 0.2797452 0.3166916 0.04422542
## 5 0.02885333      4 0.2506457 0.2927256 0.04457137
## 6 0.01713975      5 0.2217924 0.2735420 0.04712084
## 7 0.01292266      6 0.2046526 0.2565365 0.04033433
## 8 0.01000000      8 0.1788073 0.2490509 0.04046544
##
## Variable importance
##      rm   lstat   indus     nox     dis     age     tax ptratio    crim      zn
##      32      22       9       7       6       5       5       5       3       2
##     rad     lon     lat
##       2       1       1
##
## Node number 1: 404 observations,    complexity param=0.4565952
##   mean=22.68614, MSE=87.95877
##   left son=2 (330 obs) right son=3 (74 obs)
##   Primary splits:
##       rm      < 6.8375    to the left,  improve=0.4565952, (0 missing)
##       lstat   < 9.725     to the right, improve=0.4521823, (0 missing)
##       lon     < -71.0685  to the right, improve=0.2747556, (0 missing)
##       indus   < 7.225     to the right, improve=0.2716115, (0 missing)
##       ptratio < 18.75     to the right, improve=0.2605731, (0 missing)
##   Surrogate splits:
##       lstat   < 4.83      to the right, agree=0.876, adj=0.324, (0 split)
##       ptratio < 14.55     to the right, agree=0.847, adj=0.162, (0 split)
##       indus   < 3.985     to the right, agree=0.842, adj=0.135, (0 split)
##       zn      < 87.5      to the left,  agree=0.829, adj=0.068, (0 split)
##       nox     < 0.4045    to the right, agree=0.822, adj=0.027, (0 split)
##
## Node number 2: 330 observations,    complexity param=0.1596255
##   mean=19.68515, MSE=40.22629
##   left son=4 (139 obs) right son=5 (191 obs)
```

```
##   Primary splits:
##       lstat   < 14.3      to the right, improve=0.4273059, (0 missing)
##       nox     < 0.6695    to the right, improve=0.3007320, (0 missing)
##       crim    < 9.311465  to the right, improve=0.2632838, (0 missing)
##       dis     < 2.37495   to the left,  improve=0.2323209, (0 missing)
##       ptratio < 19.9      to the right, improve=0.2276603, (0 missing)
##   Surrogate splits:
##       nox   < 0.5765    to the right, agree=0.800, adj=0.525, (0 split)
##       age   < 83.6      to the right, agree=0.800, adj=0.525, (0 split)
##       indus < 16.57     to the right, agree=0.788, adj=0.496, (0 split)
##       dis   < 2.37495   to the left,  agree=0.782, adj=0.482, (0 split)
##       tax   < 434.5     to the right, agree=0.773, adj=0.460, (0 split)
##
## Node number 3: 74 observations,    complexity param=0.1040341
##   mean=36.06892, MSE=81.55944
##   left son=6 (50 obs) right son=7 (24 obs)
##   Primary splits:
##       rm      < 7.437     to the left,  improve=0.6125342, (0 missing)
##       lstat   < 4.68      to the right, improve=0.3985944, (0 missing)
##       ptratio < 19.15     to the right, improve=0.2160262, (0 missing)
##       lon     < -71.04625 to the right, improve=0.2078671, (0 missing)
##       rad     < 16        to the right, improve=0.1549302, (0 missing)
##   Surrogate splits:
##       lstat   < 3.99      to the right, agree=0.824, adj=0.458, (0 split)
##       lon     < -71.15755 to the right, agree=0.716, adj=0.125, (0 split)
##       zn      < 81.25     to the left,  agree=0.703, adj=0.083, (0 split)
##       indus   < 1.23      to the right, agree=0.703, adj=0.083, (0 split)
##       ptratio < 14.75     to the right, agree=0.703, adj=0.083, (0 split)
##
## Node number 4: 139 observations,    complexity param=0.02909945
##   mean=14.82518, MSE=19.11771
##   left son=8 (59 obs) right son=9 (80 obs)
##   Primary splits:
##       crim  < 6.91188   to the right, improve=0.3891299, (0 missing)
##       lstat < 19.605    to the right, improve=0.3345170, (0 missing)
##       dis   < 2.0037    to the left,  improve=0.3337001, (0 missing)
##       nox   < 0.657     to the right, improve=0.3321810, (0 missing)
##       tax   < 567.5     to the right, improve=0.2824666, (0 missing)
##   Surrogate splits:
##       rad < 16        to the right, agree=0.871, adj=0.695, (0 split)
##       tax < 567.5     to the right, agree=0.856, adj=0.661, (0 split)
##       lat < 42.212    to the left,  agree=0.755, adj=0.424, (0 split)
##       nox < 0.657     to the right, agree=0.755, adj=0.424, (0 split)
##       dis < 2.202     to the left,  agree=0.734, adj=0.373, (0 split)
##
## Node number 5: 191 observations,    complexity param=0.02885333
##   mean=23.22199, MSE=25.88988
##   left son=10 (153 obs) right son=11 (38 obs)
##   Primary splits:
##       rm    < 6.5445    to the left,  improve=0.20734470, (0 missing)
##       lstat < 9.54      to the right, improve=0.19906760, (0 missing)
##       dis   < 1.6141    to the right, improve=0.10956960, (0 missing)
##       lon   < -71.03785 to the right, improve=0.09574916, (0 missing)
##       chas  splits as  LR, improve=0.08694740, (0 missing)
```

```
##   Surrogate splits:
##       lstat < 5.055     to the right, agree=0.869, adj=0.342, (0 split)
##       crim  < 0.017895  to the right, agree=0.827, adj=0.132, (0 split)
##       zn    < 87.5      to the left,  agree=0.812, adj=0.053, (0 split)
##       lon   < -70.8315  to the left,  agree=0.806, adj=0.026, (0 split)
##       dis   < 10.648    to the left,  agree=0.806, adj=0.026, (0 split)
##
## Node number 6: 50 observations,    complexity param=0.01713975
##   mean=31.172, MSE=38.94322
##   left son=12 (8 obs) right son=13 (42 obs)
##   Primary splits:
##       lstat   < 9.65      to the right, improve=0.3127974, (0 missing)
##       ptratio < 18.95     to the right, improve=0.2101861, (0 missing)
##       rad     < 16        to the right, improve=0.1555746, (0 missing)
##       crim    < 2.935235  to the right, improve=0.1555746, (0 missing)
##       tax     < 534.5     to the right, improve=0.1555746, (0 missing)
##   Surrogate splits:
##       crim  < 7.393425  to the right, agree=0.92, adj=0.500, (0 split)
##       nox   < 0.659     to the right, agree=0.90, adj=0.375, (0 split)
##       rad   < 16        to the right, agree=0.90, adj=0.375, (0 split)
##       tax   < 534.5     to the right, agree=0.90, adj=0.375, (0 split)
##       indus < 15.015    to the right, agree=0.88, adj=0.250, (0 split)
##
## Node number 7: 24 observations
##   mean=46.27083, MSE=16.30623
##
## Node number 8: 59 observations
##   mean=11.64915, MSE=12.9408
##
## Node number 9: 80 observations
##   mean=17.1675, MSE=10.74744
##
## Node number 10: 153 observations,    complexity param=0.01292266
##   mean=22.06732, MSE=19.08926
##   left son=20 (82 obs) right son=21 (71 obs)
##   Primary splits:
##       lstat < 9.66      to the right, improve=0.13374900, (0 missing)
##       crim  < 7.24712   to the left,  improve=0.11433340, (0 missing)
##       lon   < -71.03785 to the right, improve=0.08881250, (0 missing)
##       rm    < 6.1405    to the left,  improve=0.08419838, (0 missing)
##       dis   < 1.68515   to the right, improve=0.08213046, (0 missing)
##   Surrogate splits:
##       nox   < 0.519     to the right, agree=0.771, adj=0.507, (0 split)
##       age   < 48.1      to the right, agree=0.739, adj=0.437, (0 split)
##       dis   < 4.48025   to the left,  agree=0.739, adj=0.437, (0 split)
##       crim  < 0.098325  to the right, agree=0.725, adj=0.408, (0 split)
##       indus < 7.625     to the right, agree=0.719, adj=0.394, (0 split)
##
## Node number 11: 38 observations
##   mean=27.87105, MSE=26.28943
##
## Node number 12: 8 observations
##   mean=23.175, MSE=51.15937
##
```

```
## Node number 13: 42 observations
##   mean=32.69524, MSE=22.11474
##
## Node number 20: 82 observations
##   mean=20.58049, MSE=6.124253
##
## Node number 21: 71 observations,    complexity param=0.01292266
##   mean=23.78451, MSE=28.56103
##   left son=42 (64 obs) right son=43 (7 obs)
##   Primary splits:
##       indus < 14.48     to the left,  improve=0.2602717, (0 missing)
##       age   < 81.8      to the left,  improve=0.2364868, (0 missing)
##       crim  < 0.484795  to the left,  improve=0.2220958, (0 missing)
##       dis   < 2.6221    to the right, improve=0.2080417, (0 missing)
##       rad   < 6.5       to the left,  improve=0.1718898, (0 missing)
##   Surrogate splits:
##       crim < 1.163695  to the left,  agree=0.972, adj=0.714, (0 split)
##       nox  < 0.589     to the left,  agree=0.972, adj=0.714, (0 split)
##       age  < 86.05     to the left,  agree=0.972, adj=0.714, (0 split)
##       dis  < 2.04295   to the right, agree=0.958, adj=0.571, (0 split)
##       rad  < 16        to the left,  agree=0.944, adj=0.429, (0 split)
##
## Node number 42: 64 observations
##   mean=22.88281, MSE=9.171111
##
## Node number 43: 7 observations
##   mean=32.02857, MSE=130.442
```

```r
party_tree1 <- as.party(tree1)
plot(party_tree1, gp = gpar(fontsize = 9))
```



Prediction

Finally, we can use the pruned trees to predict the outcome in the holdout (test) set.

```
y_tree_small_CV <- predict(p_tree_small_CV, newdata = boston_test, type = "vector")
y_tree_one_SE <- predict(p_tree_one_SE, newdata = boston_test, type = "vector")
```

Use at least two performance measures to evaluate the prediction performance of both trees.

```
# first performance measures
MSE_tree_small_CV <- mean((y_tree_small_CV-boston_test$medv)^2)
MSE_tree_small_CV ## 17.0959
```

```
## [1] 17.0959
```

```
MSE_tree_one_SE <- mean((y_tree_one_SE-boston_test$medv)^2)
MSE_tree_one_SE ## 21.35381
```

```
## [1] 21.35381
```

```
# second performance measures
eval_results <- function(true, predicted, df) {
  SSE <- sum((predicted - true)^2)
```

```r
  SST <- sum((true - mean(true))^2)
  R_squared <- 1 - SSE / SST
  RMSE = sqrt(SSE/nrow(df))


# Model performance metrics
data.frame(
  RMSE = RMSE,
  Rsquared = R_squared
)

}

eval_results(boston_test$medv, y_tree_small_CV, boston_test) ## RMSE: 4.134719; R-squared: 0.7555617
```

```
##       RMSE  Rsquared
## 1 4.134719 0.7555617
```

```r
eval_results(boston_test$medv, y_tree_one_SE, boston_test) ## RMSE: 4.621019; R-squared: 0.6946818
```

```
##       RMSE  Rsquared
## 1 4.621019 0.6946818
```

Which tree would you recommend for prediction purposes?

For prediction purposes, I would recommend the tree with the smallest cross-validation value because it has a lower MSE of 17.0959 comparing to 21.35381 from the tree with 1-SE away from the smallest CV. Smaller RMSE indicates that the forecasted values of medv is closer to the actual medv within the testing data. The greater R-sqaured further proves that the tree with the smallest CV is better than the other one.

end