Tree based methods I

Regression tree example

STAT 32950-24620

Spring 2025 (wk9)

Tree-based models

- A popular tool in supervised learning
- Data: a response variable (output),
  several explanatory variables (inputs, predictors, features).
- Method: Find a partition of the space of explanatory variables,
  in each region the response variable is relatively homogeneous.
- Results: Provide a constant prediction in each partition region.

Regression tree & classification tree

- **Regression Tree** for estimating a continuous response variable

  — Take an average as the prediction in each partition region.

- **Classification Tree** for classifying a categorical response var.

  — Take a vote as the prediction in each partition region.

- Partition of the explanatory variable space is obtained by splitting the range of a predictor, one at a time.
- The partition regions are rectangles or hyper-rectangles.

Regression Tree

Regression Tree: the response variable is continuous.

(vs Classification Tree: the response variable is categorical)

```
# libraries for Regression Tree
library(MASS)
library(tree)      # earlier than "rpart" package
library(rpart)     # newer alternative to "tree"
library(rpart.plot)  # nicer tree plots
```

rpart — Recursive Partitioning And Regression Trees

Data Example

Data: Boston housing

$n = 506$ observations, $p = 13$ input variables.

Response variable:

medv: median value of owner-occupied homes in USD 1000's

Example data: Explanatory variables

```
- crim: per capita crime rate by town
- zn:   % residential land zoned for lots over 25k sq.ft
- indus: % of non-retail business acres per town
- chas: Charles River dummy variable
     (= 1 if tract bounds river; 0 otherwise)
- nox:  nitric oxides concentration (parts per 10 million)
- rm:   average number of rooms per dwelling
- age:  % of owner-occupied units built prior to 1940
- dis:  weighted distances to 5 Boston employment centres
- rad:  index of accessibility to radial highways
- tax:  full-value property-tax rate per USD 10,000
- ptratio:  pupil-teacher ratio by town
- black:    1000(Bk - 0.63)^2, Bk = % of blacks by town
- lstat:    percentage of lower status of the population
```


Check data format

```
str(Boston); #summary(Boston)
```

```
## 'data.frame':    506 obs. of  14 variables:
##  $ crim   : num  0.00632 0.02731 0.02729 0.03237 0.0690
##  $ zn     : num  18 0 0 0 0 0 12.5 12.5 12.5 12.5 ...
##  $ indus  : num  2.31 7.07 7.07 2.18 2.18 2.18 7.87 7.8
##  $ chas   : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ nox    : num  0.538 0.469 0.469 0.458 0.458 0.458 0.
##  $ rm     : num  6.58 6.42 7.18 7 7.15 ...
##  $ age    : num  65.2 78.9 61.1 45.8 54.2 58.7 66.6 96.
##  $ dis    : num  4.09 4.97 4.97 6.06 6.06 ...
##  $ rad    : int  1 2 2 3 3 3 5 5 5 5 ...
##  $ tax    : num  296 242 242 222 222 222 311 311 311 31
##  $ ptratio: num  15.3 17.8 17.8 18.7 18.7 18.7 15.2 15.
##  $ black  : num  397 397 393 395 397 ...
##  $ lstat  : num  4.98 9.14 4.03 2.94 5.33 ...
##  $ medv   : num  24 21.6 34.7 33.4 36.2 28.7 22.9 27.1
```


Regression Tree on the training data

We split the dataset into two equal sizes,

creating training and testing data.

```
set.seed(1)
# random sample from row numbers
train = sample(1:nrow(Boston), nrow(Boston)/2)
```

Fit a regression tree on the training data:

```
tree.boston=tree(medv~.,Boston,subset=train)
```

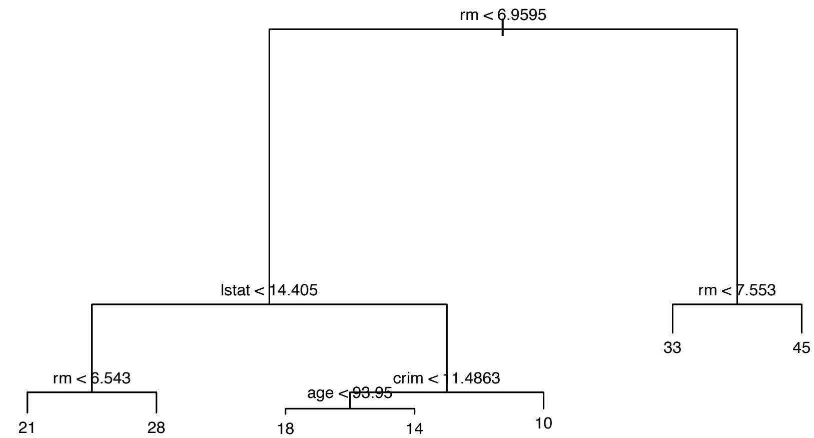
Check the fitted tree

```r
summary(tree.boston)
```

```
##
## Regression tree:
## tree(formula = medv ~ ., data = Boston, subset = train)
## Variables actually used in tree construction:
## [1] "rm"    "lstat" "crim"  "age"
## Number of terminal nodes:  7
## Residual mean deviance:  10.4 = 2550 / 246
## Distribution of residuals:
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
## -10.200  -1.780  -0.177   0.000   1.920  16.600
```

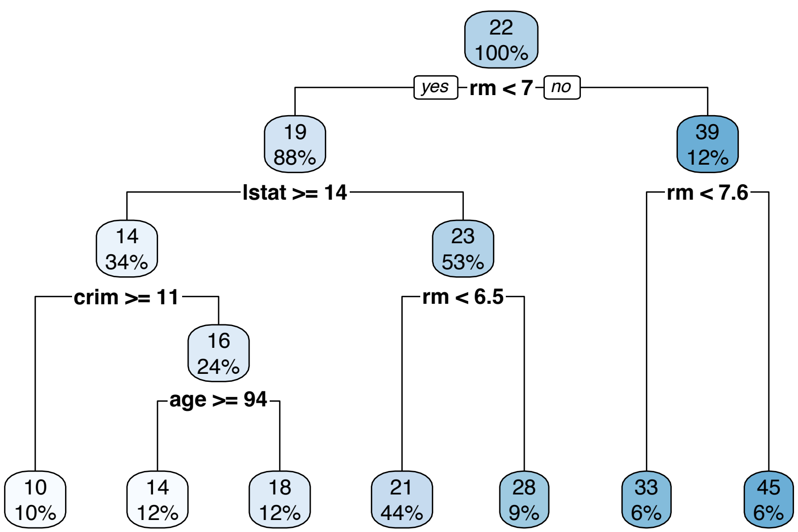

Plot the fitted regression tree (using tree)

```r
plot(tree.boston)
text(tree.boston,cex=0.7,digits=2)
```



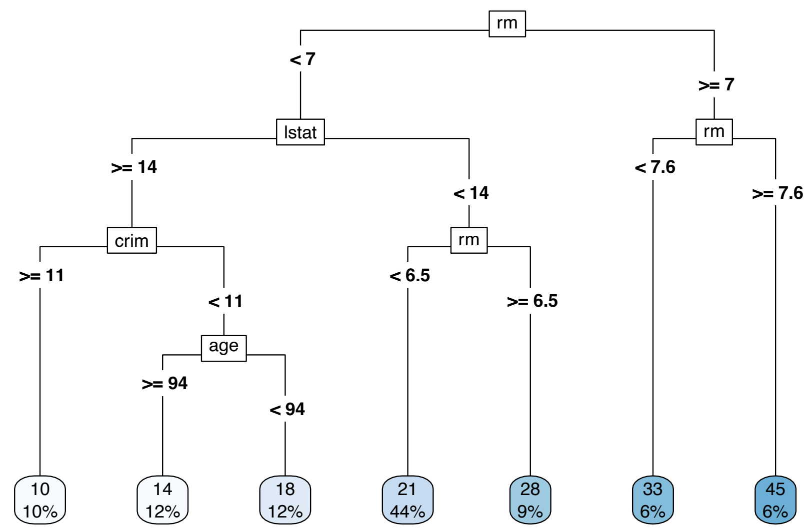
Plot the regression tree (using rpart)

```r
rpart.plot(rpart(medv~.,Boston,subset=train))
```



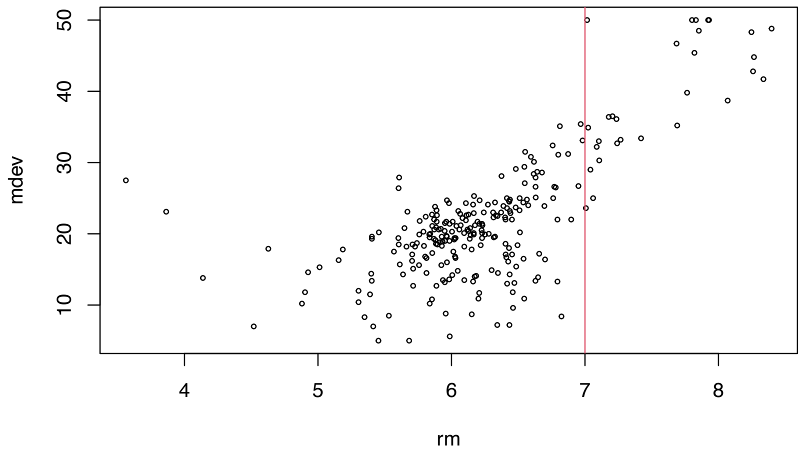
Plot the tree in another style (using rpart)

```r
rpart.plot(rpart(medv~.,Boston,subset=train),type=5)
```



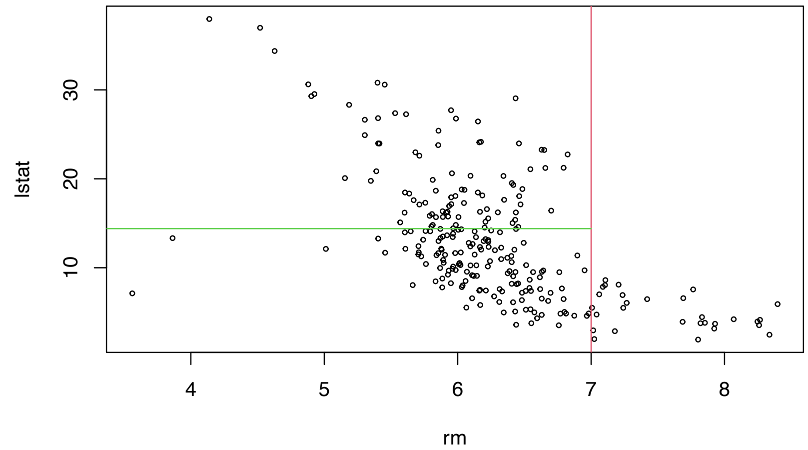
Feature space stratification: Step 1 of the split

```
plot(Boston[train,]$rm,Boston[train,]$medv,cex=.5,
     xlab="rm",ylab="mdev"); abline(v=7, col=2)
```



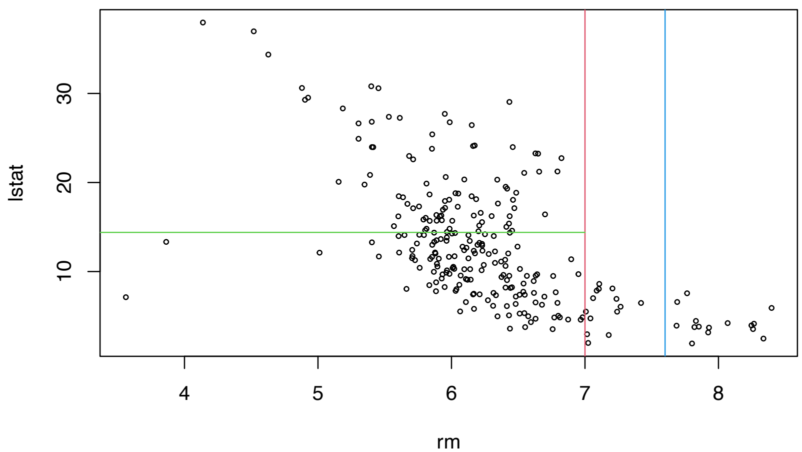
Feature space stratification: Steps 2

```
plot(Boston[train,]$rm,Boston[train,]$lstat,cex=.5,
     xlab="rm",ylab="lstat")
abline(v=7,col=2); segments(3,14.4, 7,14.4,col=3)
```



Feature space stratification: Steps 3

```
plot(Boston[train,]$rm,Boston[train,]$lstat,cex=.5,
     xlab="rm",ylab="lstat"); abline(v=7,col=2);
segments(3,14.4, 7,14.4,col=3); abline(v=7.6,col=4)
```



How to built a regression tree

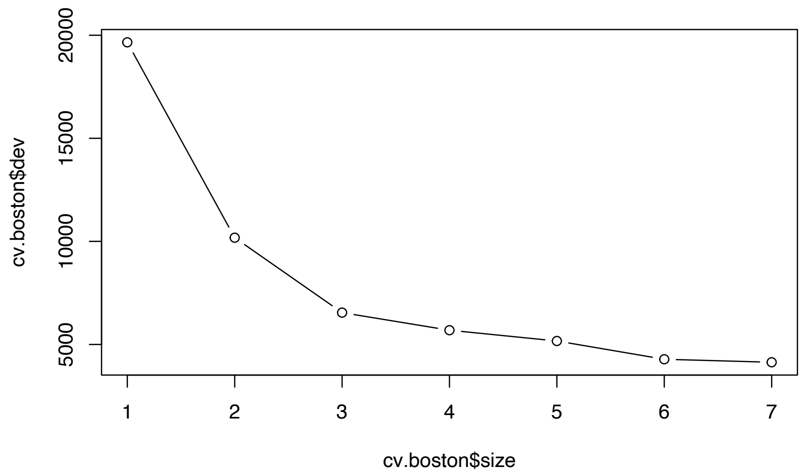
- Grow the tree upside down from the root to leaves.
- Start with a single region R_k, iterate.
  - Select a region R_k, a predictor X_j, a spliting point s, such that splitting R_k with the rule $X_j < s$ optimally reduces the RSS

$$\sum_{m=1}^{|T|} \sum_{x_i \in R_m} (y_i - \bar{y}_{R_m})^2$$

  - Redefine the regions with an additional split.
  - Iterate.
- Terminate when there are very few observations (e.g. ≤ 5) in a region.

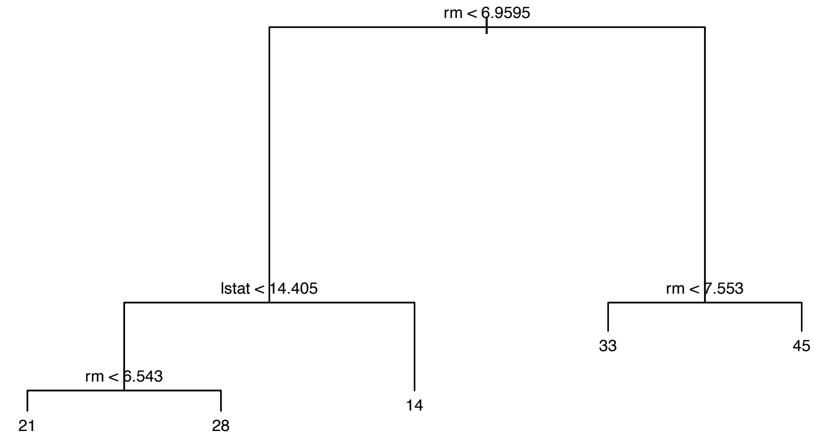
Tree deviance vs tree size

```r
cv.boston=cv.tree(tree.boston) # default 10-folds
plot(cv.boston$size,cv.boston$dev,type='b')
```



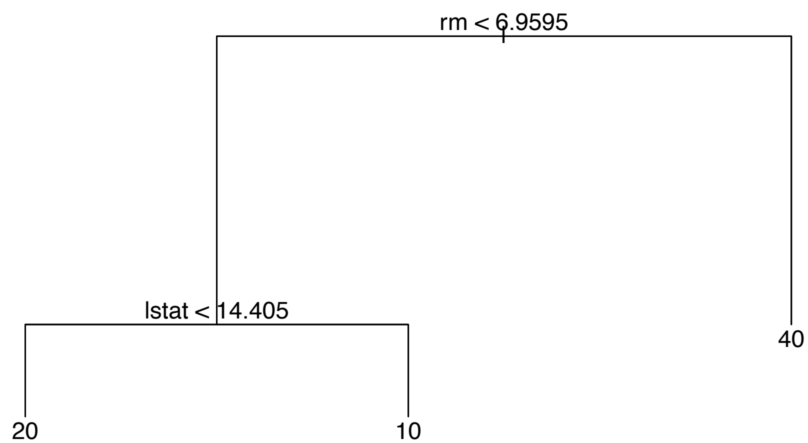
Prune tree using deviance (5 regions)

```r
plot(prune.tree(tree.boston,best=5)) # best=tree-size
text(prune.tree(tree.boston,best=5),cex=0.7,digits =2)
```



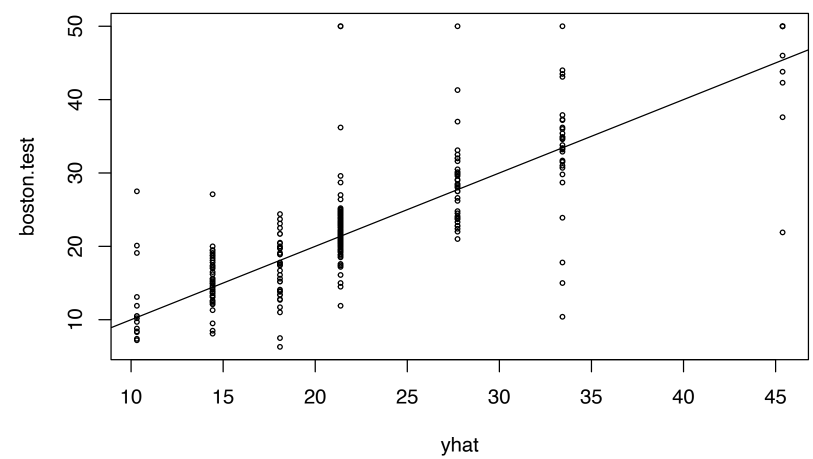
Prune tree using deviance (3 regions)

```r
plot(prune.tree(tree.boston,best =3))
text(prune.tree(tree.boston,best =3))
```



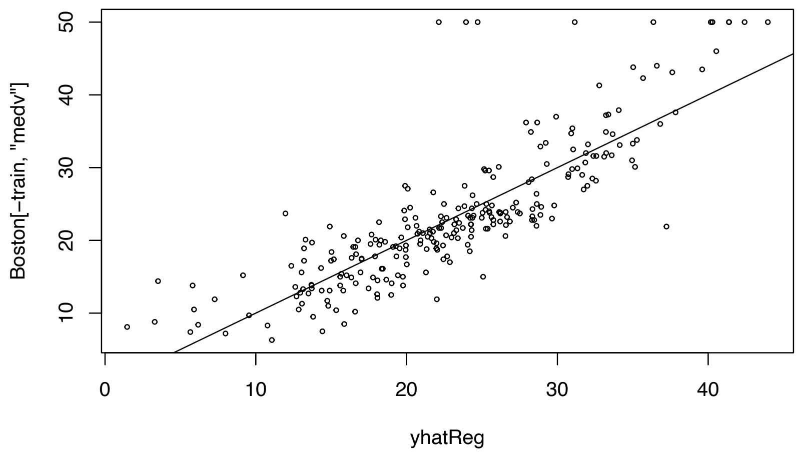
Prediction by the full tree on testing data

```r
yhat=predict(tree.boston,newdata=Boston[-train,])
boston.test=Boston[-train,"medv"]
plot(yhat,boston.test,cex=.5); abline(0,1)
```



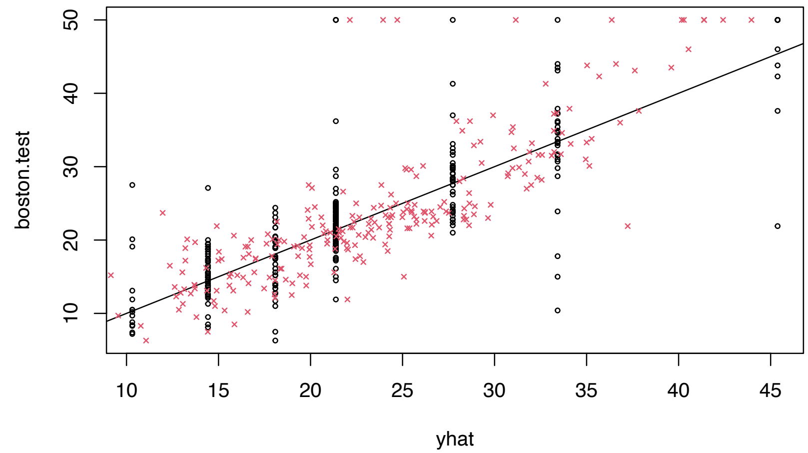
Prediction by the linear reg. model on testing data

```r
yhatReg=predict(lm(medv~.,Boston,subset=train),
    newdata=Boston[-train,]) # result similar to 'train'
plot(yhatReg,Boston[-train,"medv"],cex=.5); abline(0,1)
```



Comparison of predictions by 'tree' vs 'lm'

```r
yhat=predict(tree.boston,newdata=Boston[-train,])
boston.test=Boston[-train,"medv"]
plot(yhat,boston.test,cex=.5); abline(0,1)
points(yhatReg,Boston[-train,"medv"],cex=.5,pch=4,col=2)
```



Comparison of mean SS Residuals of tree vs lm

```r
mean((yhat-boston.test)^2)
```

```
## [1] 35.29
```

```r
mean((yhatReg-boston.test)^2)
```

```
## [1] 26.86
```

Which method is better?

Note: comparison of mean SS residual on training vs testing data.

Cross validation

- Split the training data into 10 folds (default)
- For $k = 1, \cdots, 10$, using every fold except the kth
- Construct trees $T_1, \cdots, T_m$ for a range of α in

$$\min_T \left(\sum_{m=1}^{|T|} \sum_{x_i \in R_m} (y_i - \bar{y}_{R_m})^2 + \alpha |T| \right)$$

Cost complexity pruning

- For each tree T_i, calculate RSS on the test set

  Remove the **Weakest link**, the subtree minimizes

$$\frac{RSS(T_1) - RSS(T_0)}{|T_0| - |T_1|}$$

- Select α that minimizes the average testing error.