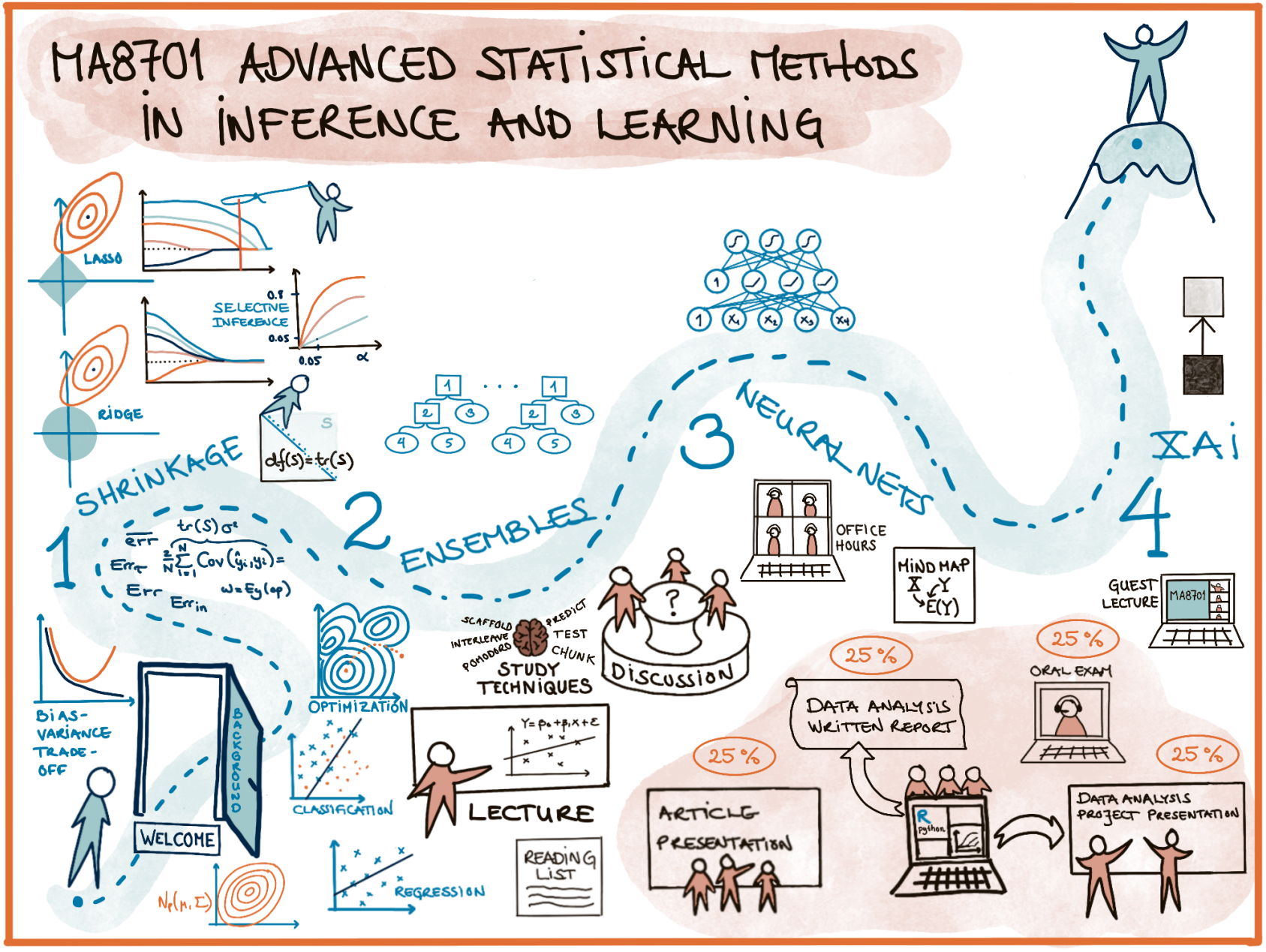
MA8701 Advanced methods in statistical inference and learning

L5: Trees, bagging and random forest

Mette Langaas IMF/NTNU

07 February, 2021

Ensembles - first act



Outline

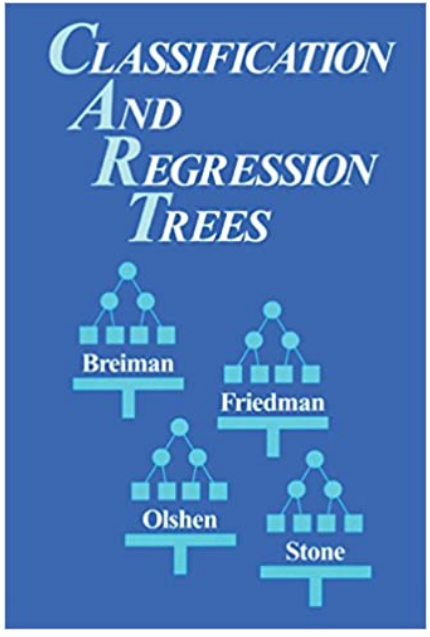
- Trees
- Many trees with bootstrap aggregation
- Many trees into a random forest
- Conclusions

Literature L5

- [ELS] The Elements of Statistical Learning: Data Mining, Inference, and Prediction, Second Edition (Springer Series in Statistics, 2009) by Trevor Hastie, Robert Tibshirani, and Jerome Friedman. Ebook. Chapter 8.7 (bagging), 9.2 (trees), 9.6 (missing data), 15 (random forest, not 15.3.3 and 15.4.3).

Trees

(ELS 9.2)


CLASSIFICATION
AND
REGRESSION
TREES
Breiman
Friedman
Olshen
Stone

Main idea

(for regression or classification)

- Derive a set of decision rules for segmenting the predictor space into a number of regions.
- We classify a new observation into one of these regions by applying the derived decision rules.
- Then we typically use the mean (regression problems) or a majority vote (classification problems) of the training observations in this region as the prediction in the region.
- Key advantage: interpretability.

We will only allow *recursive binary partitions* of the predictor space, using some stopping criterion.

From regions in predictor space to decision tree

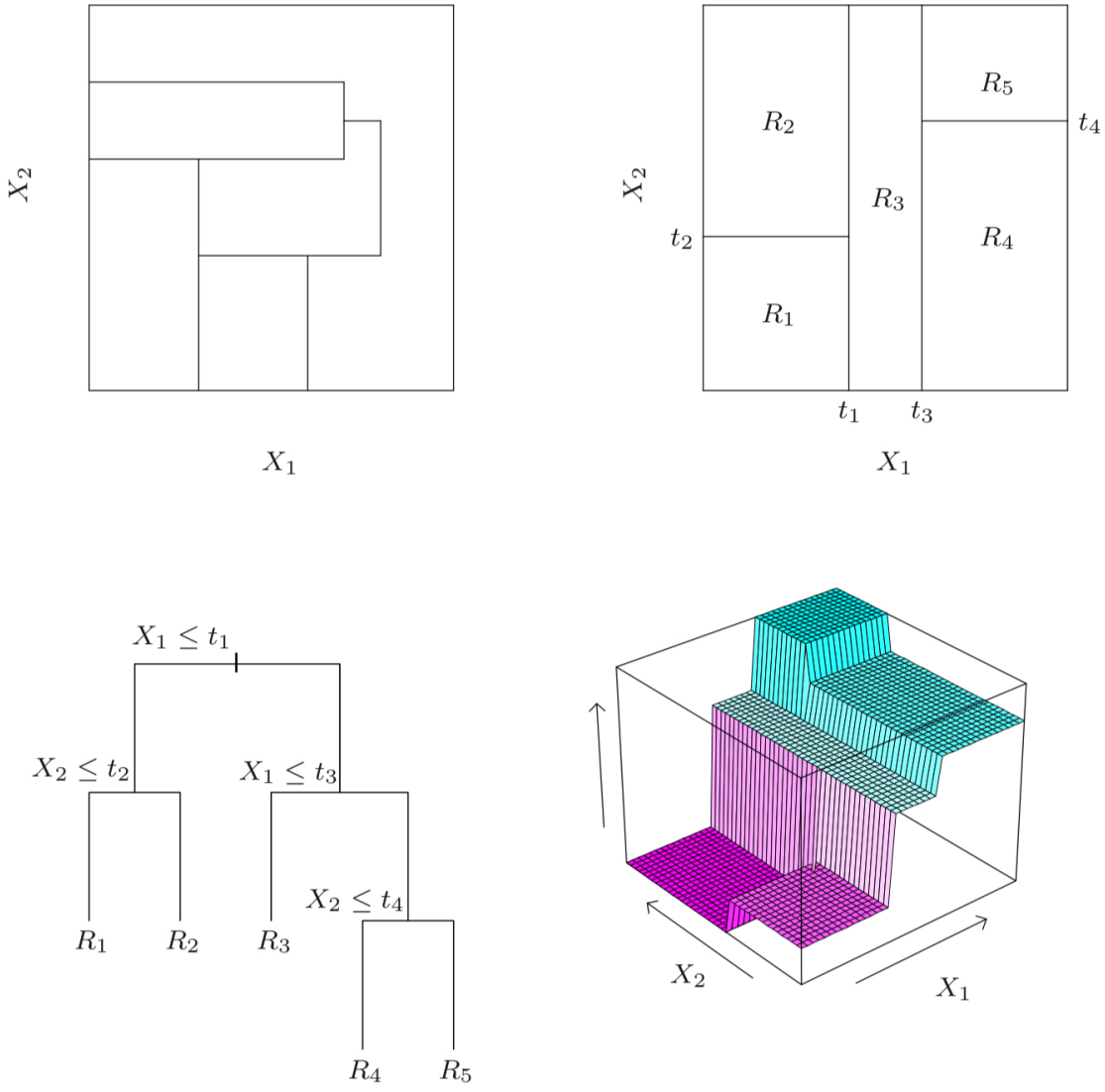


FIGURE 9.2. *Partitions and CART. Top right panel shows a partition of a two-dimensional feature space by recursive binary splitting, as used in CART, applied to some fake data. Top left panel shows a general partition that cannot be obtained from recursive binary splitting. Bottom left panel shows the tree corresponding to the partition in the top right panel, and a perspective plot of the prediction surface appears in the bottom right panel.*

Regression tree

Fitting a regression tree

Assume that we have a dataset consisting of N pairs $(\mathbf{x}_i, Y_i)$, $i = 1, \ldots, N$, and each predictor is $\mathbf{x}_i = (x_{i1}, x_{i2}, ..., x_{ip})$.
We want:

$$\hat{f}(X_i) = \sum_{m=1}^{M} \hat{c}_m I(X_i \in R_m)$$

where $\hat{c}_m$ is the estimate for region R_m.

Two steps:

1. Divide the predictor space into non-overlapping regions $R_1, R_2, \ldots, R_m$.
2. For every observation that falls into region R_j we make the same prediction - which is the mean of the responses for the training observations that fall into R_j.

$$\hat{c}_m = \text{ave}(y_i \mid x_i \in R_m)$$

How to divide the predictor space into non-overlapping regions $R_1, R_2, \ldots, R_m$?

Ripley (1996, p 216): Two types of optimality. a) Optimality of the partitioning of the predictor space : only feasible for small dimensions. b) Given partitioning of predictor space, how to represent this by a tree in the best possible way (=minimal expected number of tests) is a NP-complete problem. (NP=nondeterministic polynomial, a NP problem can be solved in polynomial time.)

A *greedy* approach is taken (aka top-down) - called *recursive binary splitting.*

Recursive binary splitting

Pruning

Imagine that we have a data set with many predictors, and that we fit a large tree. Then, the number of observations from the training set that falls into some of the regions R_j may be small, and we may be concerned that we have overfitted the training data.

Pruning is a technique for solving this problem.

By *pruning* the tree we reduce the size or depth of the decision tree. When we reduce the number of terminal nodes and regions $R_1, ..., R_M$, each region will probably contain more observations.

If we have a large dataset with many explanatory variables and terminal nodes, we can also prune the tree if we want to create a simpler tree and increase the interpretability of the model, or just avoid "unneccesary" splits (for classification that could mean that both branches give the same classification rule).

Cost complexity pruning

We first build a large tree T_0 by recursive binary splitting. Then we try to find a sub-tree $T \subset T_0$ that (for a given value of α) minimizes

$$C_\alpha(T) = Q(T) + \alpha|T|,$$

where $Q(T)$ is our cost function, $|T|$ is the number of terminal nodes in tree T. The parameter α is then a parameter penalizing the number of terminal nodes, ensuring that the tree does not get too many branches.

We proceed by repeating the the process for a larger value of α, and this way we get a sequence of smaller of smaller sub-trees where each tree is the best sub-tree of the previous tree.

For regression trees we choose $Q(T) = \sum_{m=1}^{|T|} \sum_{x_i \in R_m} (y_i - \hat{c}_m)^2$, and or classification trees (to come next) the entropy, Gini or misclassification rate (the last most popular).

Given a value of α we get a pruned tree (but the same pruned tree for "ranges" of α). For $\alpha = 0$ we get T_0 and as α increases we get smaller and smaller trees.

Technically an algorithm called *weakest link pruning* is used, where internal nodes in the tree is collapsed to get the smallest increase pr node for $Q(T)$. This is done successively until the tree is a single node tree.

Please study this note from Bo Lindqvist in MA8701 in 2017 for an example of how we perform cost complexity pruning in detail. Alternatively, this method, with proofs, are given in Ripley (1996), Section 7.2.

The consept of pruning is not central in our course, since we will mostly use trees as building blocks in bagging, random forest and boosting (where pruning is not "needed").

Pruning in practice

The penalty parameter α is a *model hyperparameter* and is chosen by cross-validation.

R: function `tree` in package `tree`

by Brian D. Ripley: Fit a Classification or Regression Tree

Description: A tree is grown by binary recursive partitioning using the response in the specified formula and choosing splits from the terms of the right-hand-side.

tree(formula, data, weights, subset, na.action = na.pass, control = tree.control(nobs, …), method = "recursive.partition", split = c("deviance", "gini"), model = FALSE, x = FALSE, y = TRUE, wts = TRUE, …)

- Details: A tree is grown by binary recursive partitioning using the response in the specified formula and choosing splits from the terms of the right-hand-side. Numeric variables are divided into X < a and X > a; the levels of an unordered factor are divided into two non-empty groups. The split which maximizes the reduction in impurity is chosen, the data set split and the process repeated. Splitting continues until the terminal nodes are too small or too few to be split.
- A numerical response gives regression while a factor response gives classification.
- The default choice for a function to minimize is the deviance, and for normal data (as we may assume for regression), the deviance is proportional to the MSE. For the interested reader, this is the connection between the deviance and the MSE for regression https://www.math.ntnu.no/emner/TMA4315/2018h/2MLR.html#deviance

---

- Tree growth is limited to a depth of 31 by the use of integers to label nodes.
- Factor predictor variables can have up to 32 levels. This limit is imposed for ease of labelling, but since their use in a classification tree with three or more levels in a response involves a search over 2^(k-1) - 1 groupings for k levels, the practical limit is much less.

R: `rpart` package

A competing R function is `rpart`, explained in package vignette. Short on using `rpart` R package for CART.

Regression example

(ISLR book, Section 8.3.4.)

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

- Collected by the U.S Census Service concerning housing in the area of Boston Massachusetts, US.
- Two tasks often performed: predict nitrous oxide level (nox), or predict the median value of a house with in a "town" (medv).

Variables

- CRIM - per capita crime rate by town
- ZN - proportion of residential land zoned for lots over 25,000 sq.ft.
- INDUS - proportion 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 - proportion of owner-occupied units built prior to 1940
- DIS - weighted distances to five Boston employment centres
- RAD - index of accessibility to radial highways

- TAX - full-value property-tax rate per $10,000
- PTRATIO - pupil-teacher ratio by town
- B - #1000(Bk - 0.63)^2# where Bk is the proportion of African Americans by town (black below)
- LSTAT - % lower status of the population
- MEDV - Median value of owner-occupied homes in $1000's (seems to be a truncation)

Data

Boston data used from the `MASS` R package. Data are divided into a training and a test set with 70/30 split.

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

```
##      crim zn indus chas   nox    rm  age    dis rad tax ptratio  black lstat
## 1 0.00632 18  2.31    0 0.538 6.575 65.2 4.0900   1 296    15.3 396.90  4.98
## 2 0.02731  0  7.07    0 0.469 6.421 78.9 4.9671   2 242    17.8 396.90  9.14
## 3 0.02729  0  7.07    0 0.469 7.185 61.1 4.9671   2 242    17.8 392.83  4.03
## 4 0.03237  0  2.18    0 0.458 6.998 45.8 6.0622   3 222    18.7 394.63  2.94
## 5 0.06905  0  2.18    0 0.458 7.147 54.2 6.0622   3 222    18.7 396.90  5.33
## 6 0.02985  0  2.18    0 0.458 6.430 58.7 6.0622   3 222    18.7 394.12  5.21
##   medv
## 1 24.0
## 2 21.6
## 3 34.7
## 4 33.4
## 5 36.2
## 6 28.7
```

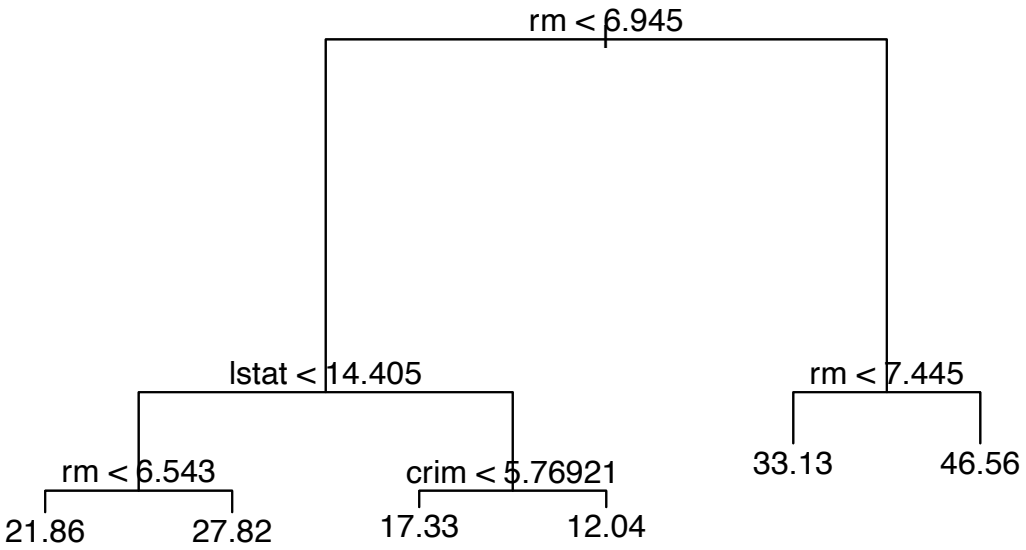
Regression tree

First with `tree` and default parameters.

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

```
##
## Regression tree:
## tree(formula = medv ~ ., data = Boston, subset = train)
## Variables actually used in tree construction:
## [1] "rm"    "lstat" "crim"
## Number of terminal nodes:  6
## Residual mean deviance:  14.86 = 5172 / 348
## Distribution of residuals:
##      Min.   1st Qu.    Median      Mean   3rd Qu.      Max.
## -11.36000  -2.25600  -0.04933   0.00000   2.16700  28.14000
```

```
text(tree.boston,pretty=0)
```

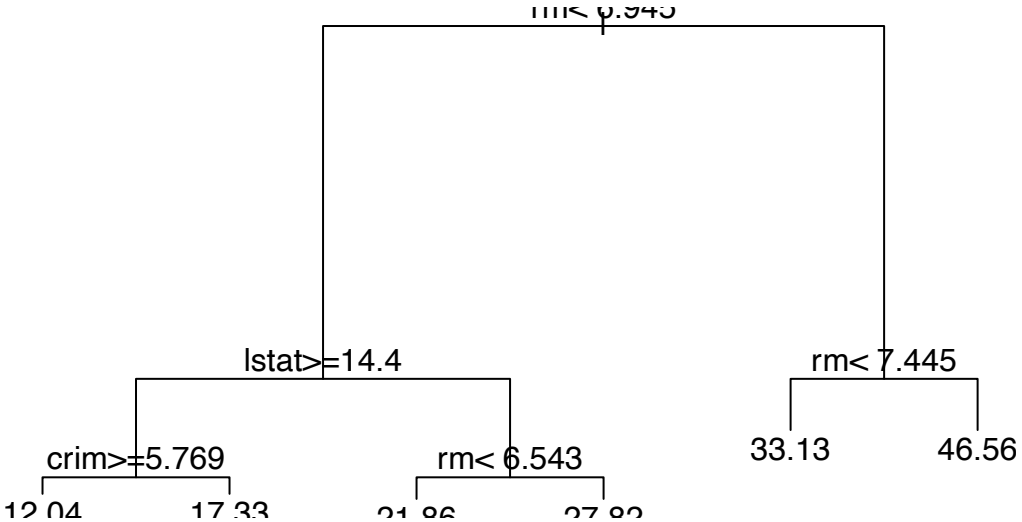



```
tree.boston
```

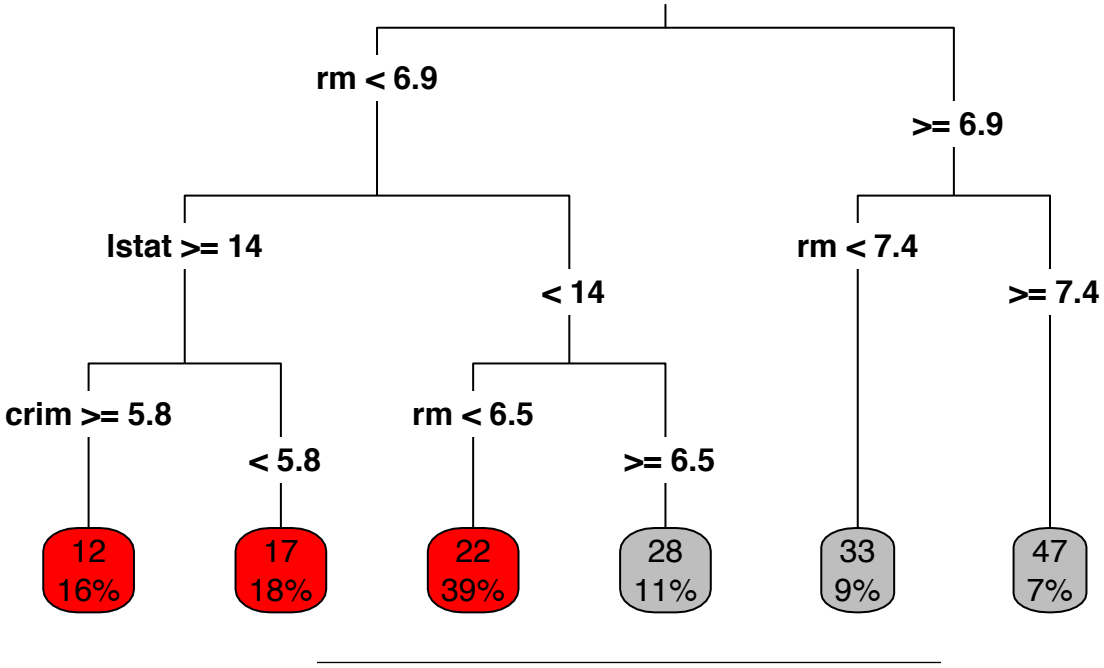
```
## node), split, n, deviance, yval
##       * denotes terminal node
##
##  1) root 354 32270.0 22.95
##    2) rm < 6.945 296 10830.0 19.82
##      4) lstat < 14.405 177  3681.0 23.17
##        8) rm < 6.543 138  1690.0 21.86 *
##        9) rm > 6.543 39   908.2 27.82 *
##      5) lstat > 14.405 119  2215.0 14.84
##       10) crim < 5.76921 63   749.9 17.33 *
##       11) crim > 5.76921 56   636.1 12.04 *
##    3) rm > 6.945 58  3754.0 38.92
##      6) rm < 7.445 33   749.7 33.13 *
##      7) rm > 7.445 25   438.0 46.56 *
```

Then with the `rpart` R package, and default parameters. This gives the same tree as `tree`. (What do you think of the tree from `rpart.plot`?)

```
boston.rpart <- rpart(formula = medv~. , data = Boston,subset=train)
plot(boston.rpart)
text(boston.rpart,pretty=0)
```



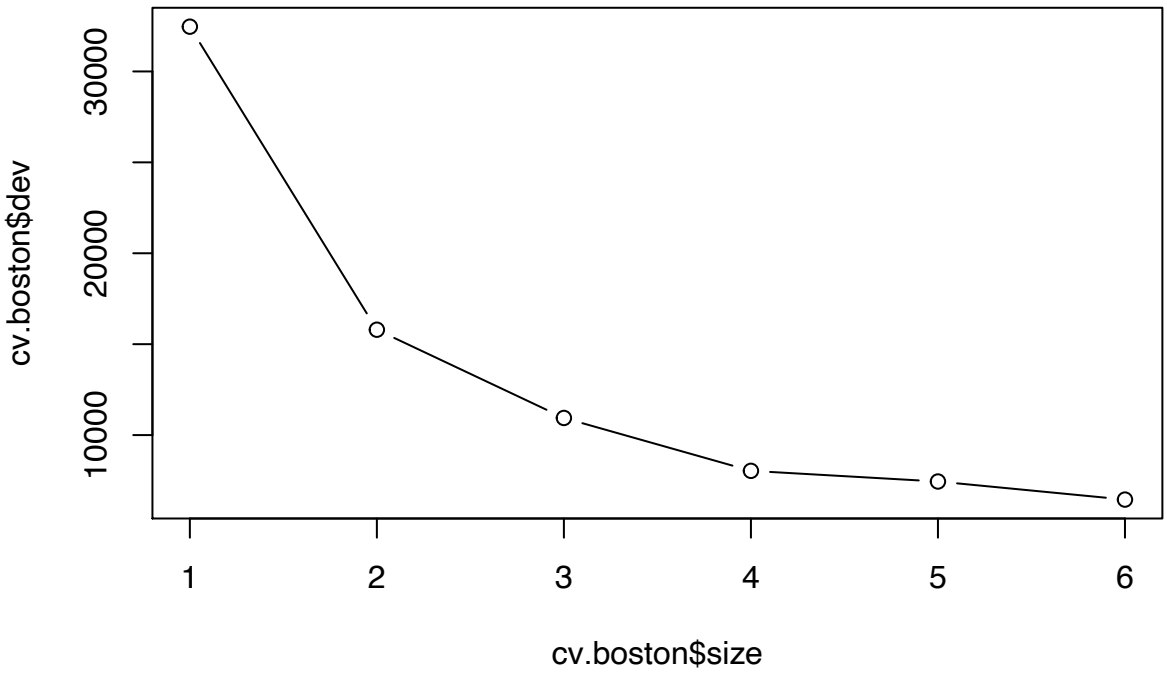

```
rpart.plot(boston.rpart,type = 3, box.palette = c("red", "grey"), fallen.leaves = TRUE)
```



Need to prune?

The default criterion to monitor is the deviance. For normal regression the deviance is proportional to the MSE.

```
cv.boston=cv.tree(tree.boston)
plot(cv.boston$size,cv.boston$dev,type='b')
```

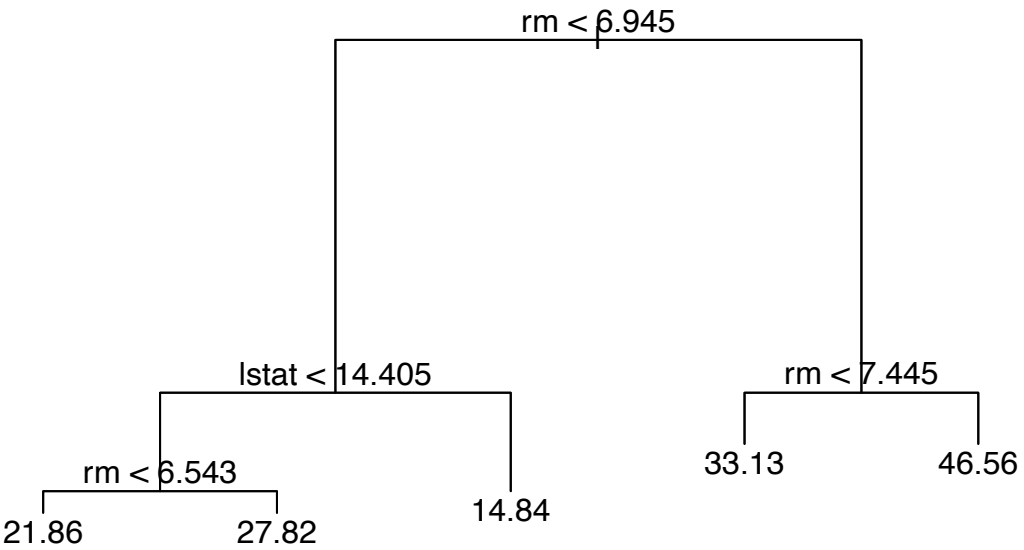


Most complex tree selected.

Pruning

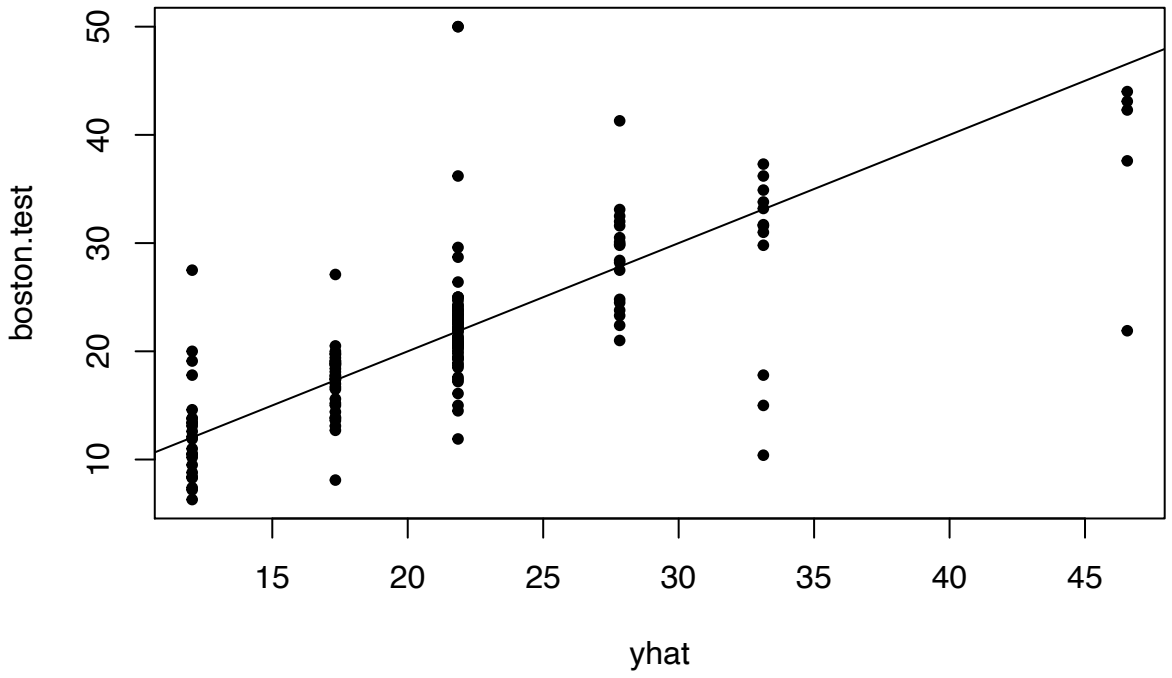
Just to show pruning (even if most complex tree was selected).

```
prune.boston=prune.tree(tree.boston,best=5)
plot(prune.boston)
text(prune.boston,pretty=0)
```



Test error for full tree

```
yhat=predict(tree.boston,newdata=Boston[-train,])
boston.test=Boston[-train,"medv"]
plot(yhat,boston.test, pch=20)
abline(0,1)
```




```
print("MSE on test set for tree")
```

```
## [1] "MSE on test set for tree"
```

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

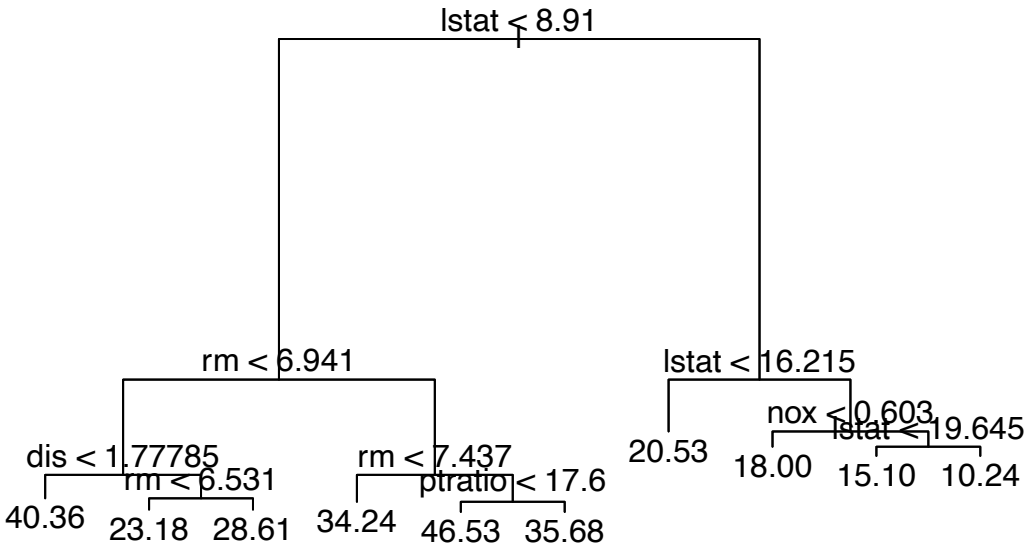
```
## [1] 36.2319
```

Effect of changing training set

```
set.seed(4)
train2 = sample(1:nrow(Boston), 0.7*nrow(Boston))
tree.boston2=tree(medv~.,Boston,subset=train2)
summary(tree.boston2); plot(tree.boston2)
```

```
##
## Regression tree:
## tree(formula = medv ~ ., data = Boston, subset = train2)
## Variables actually used in tree construction:
## [1] "lstat"   "rm"      "dis"     "ptratio" "nox"
## Number of terminal nodes:  10
## Residual mean deviance:  14.31 = 4921 / 344
## Distribution of residuals:
##     Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
## -16.0600  -2.0070  -0.0288   0.0000   1.8850  15.7600
```

```
text(tree.boston2,pretty=0)
```




```
tree.boston2
```

```
## node), split, n, deviance, yval
##       * denotes terminal node
##
##  1) root 354 30810.0 22.58
##    2) lstat < 8.91 128  9746.0 31.26
##      4) rm < 6.941 75  2746.0 26.43
##        8) dis < 1.77785 5   702.1 40.36 *
##        9) dis > 1.77785 70  1004.0 25.43
##         18) rm < 6.531 41   244.3 23.18 *
##         19) rm > 6.531 29   258.8 28.61 *
##      5) rm > 6.941 53  2776.0 38.09
##       10) rm < 7.437 32   657.1 34.24 *
##       11) rm > 7.437 21   925.3 43.95
##         22) ptratio < 17.6 16   209.8 46.53 *
##         23) ptratio > 17.6 5   266.9 35.68 *
##    3) lstat > 8.91 226  5965.0 17.66
##      6) lstat < 16.215 125  1549.0 20.53 *
##      7) lstat > 16.215 101  2117.0 14.11
##       14) nox < 0.603 31   456.5 18.00 *
##       15) nox > 0.603 70   984.4 12.39
##         30) lstat < 19.645 31   204.7 15.10 *
##         31) lstat > 19.645 39   372.5 10.24 *
```

```
xnew=Boston[1,]
predict(tree.boston,newdata=xnew)
```

```
##        1
## 27.82308
```

```
predict(tree.boston2,newdata=xnew)
```

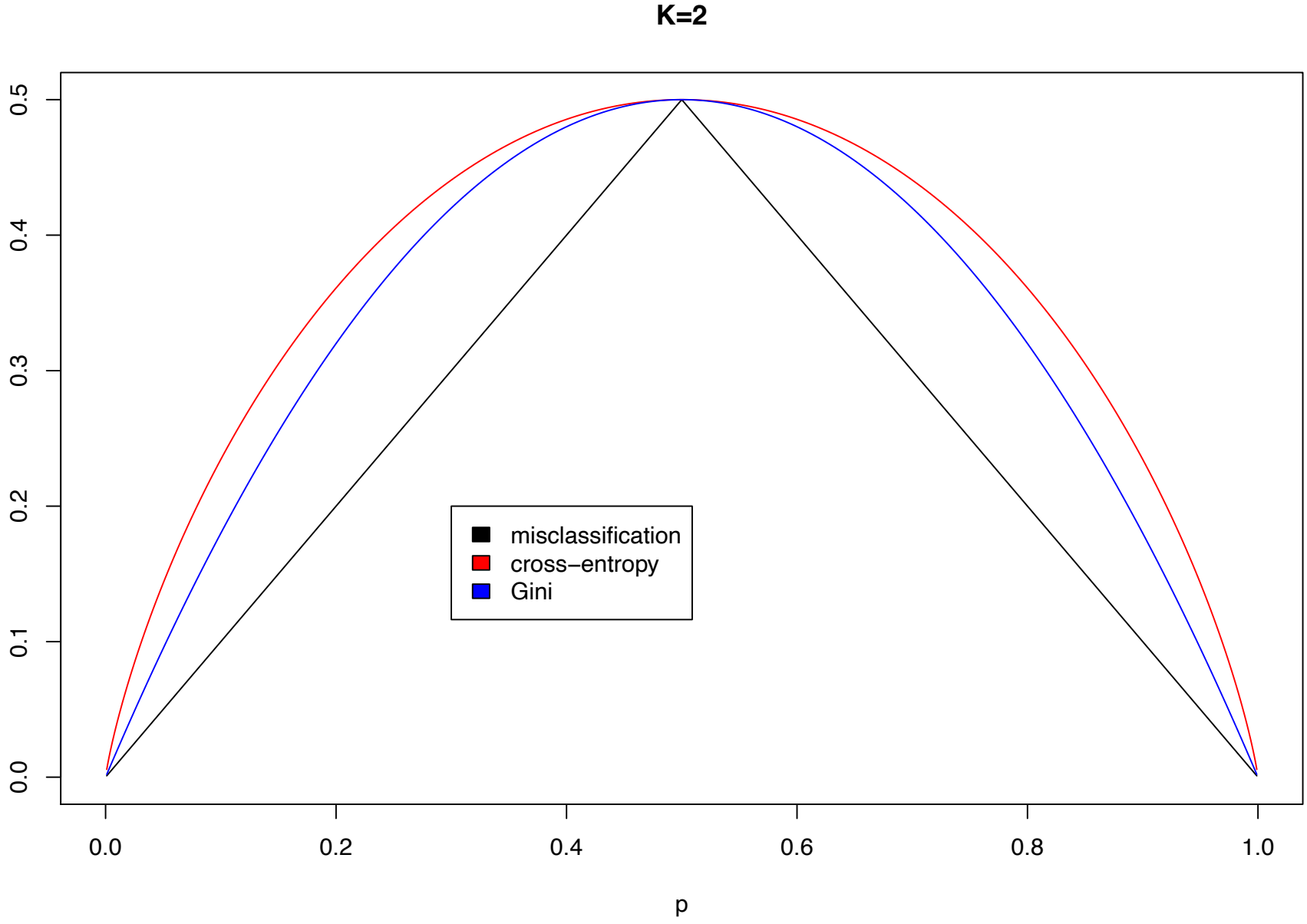
```
##        1
## 28.61034
```

Classification tree

There are only minor differences between working with regression and classification trees.

Fitting a classification tree

Let K be the number of classes for the response.
Building a decision tree in this setting is similar to building a regression tree for a quantitative response, but there are two main differences: *the prediction and the splitting criterion*


K=2
misclassification
cross−entropy
Gini
0.0
0.1
0.2
0.3
0.4
0.5
0.0
0.2
0.4
0.6
0.8
1.0
p

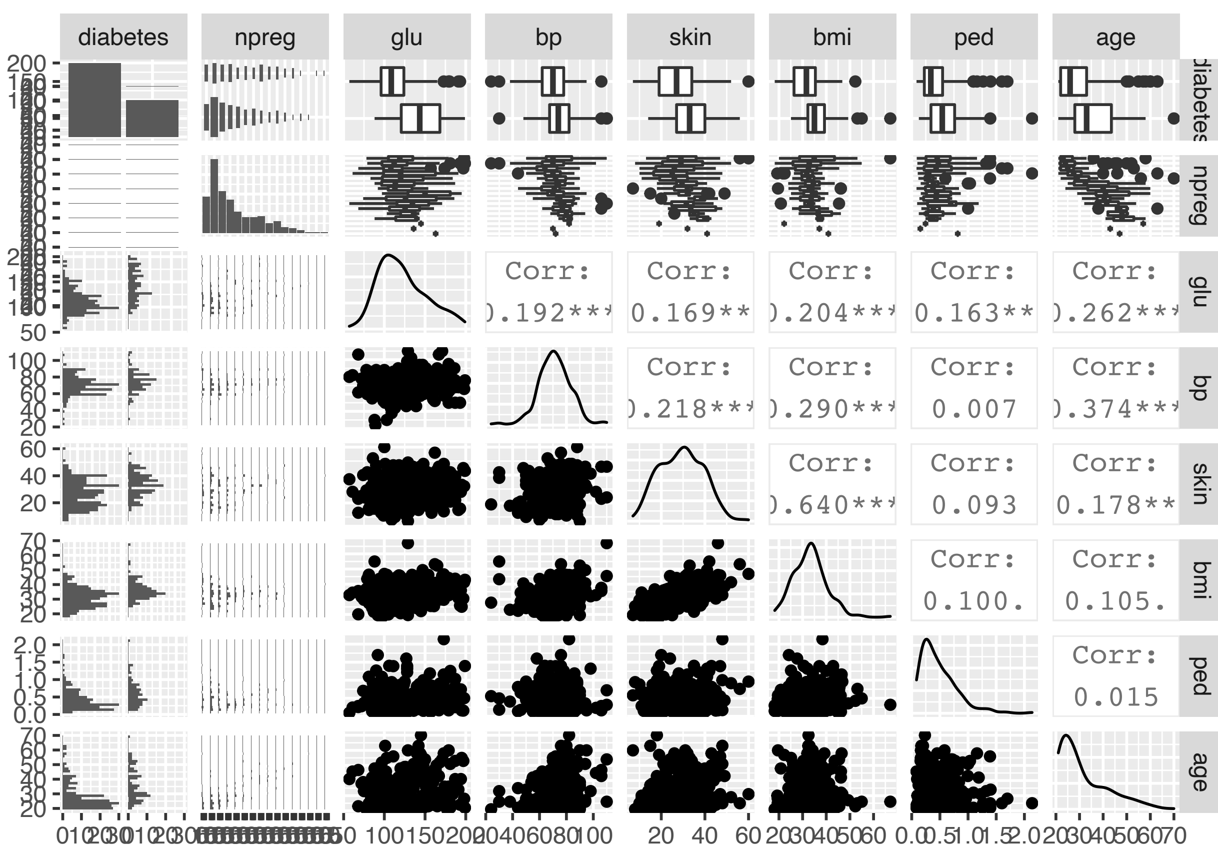
Classification example

We will use the classical data set of *diabetes* from a population of women of Pima Indian heritage in the US, available in the R `MASS` package. The following information is available for each woman:

- diabetes: `0`= not present, `1`= present
- npreg: number of pregnancies
- glu: plasma glucose concentration in an oral glucose tolerance test
- bp: diastolic blood pressure (mmHg)
- skin: triceps skin fold thickness (mm)
- bmi: body mass index (weight in kg/(height in m)^2)
- ped: diabetes pedigree function.
- age: age in years

We will use a training set (called `ctrain`) with 300 observations (200 non-diabetes and 100 diabetes cases) and a test set (called `ctest`) with 232 observations (155 non-diabetes and 77 diabetes cases). Our aim is to make a classification rule for diabetes (or not) based on the available data.

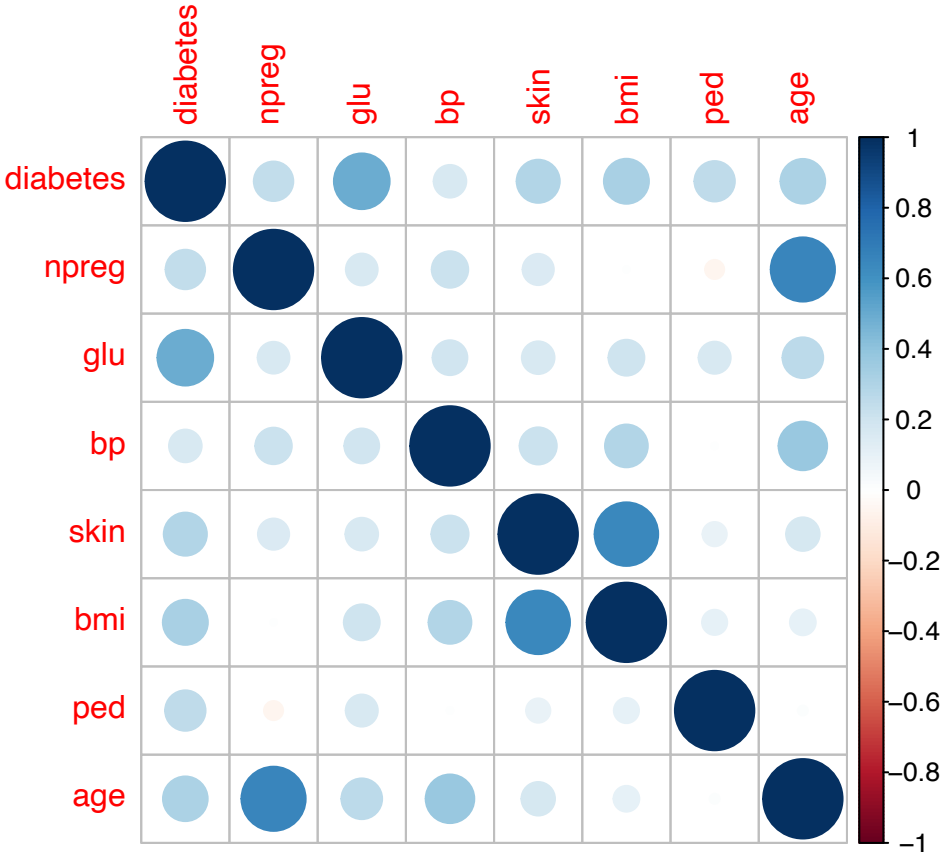
(There is a version of the training data with missing values, `Pima.tr2` in the `MASS` library. Here a slightly different shuffling for training and test data than in `MASS` is used, because these data were used in a project in TMA4268.)




```
##     diabetes          npreg             glu              bp
##  Min.   :0.0000   Min.   : 0.000   Min.   : 57.0   Min.   : 24.00
##  1st Qu.:0.0000   1st Qu.: 1.000   1st Qu.:100.0   1st Qu.: 64.00
##  Median :0.0000   Median : 2.000   Median :117.0   Median : 70.00
##  Mean   :0.3333   Mean   : 3.437   Mean   :122.7   Mean   : 71.23
##  3rd Qu.:1.0000   3rd Qu.: 5.000   3rd Qu.:143.0   3rd Qu.: 78.00
```

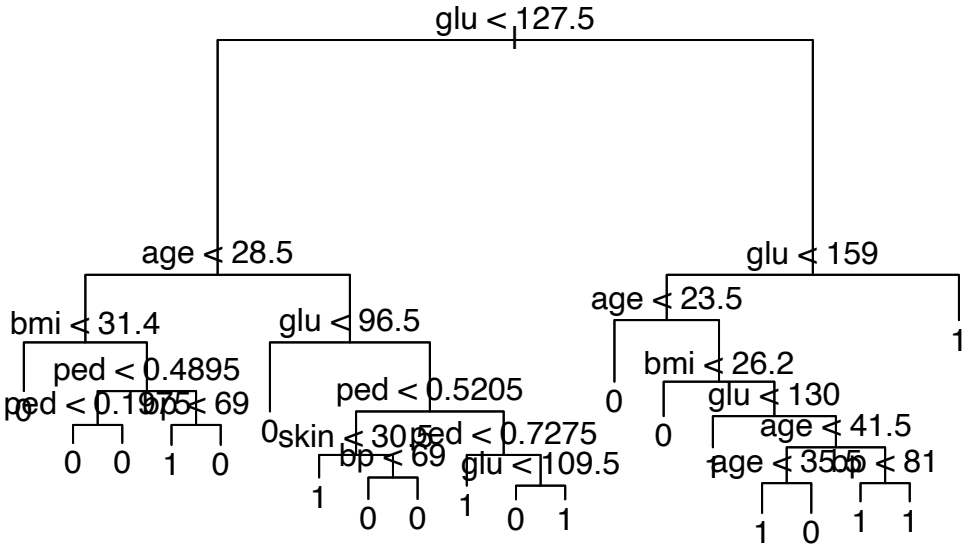
```
##  Max.   :1.0000   Max.   :17.000   Max.   :199.0   Max.   :110.00
##       skin             bmi              ped              age
##  Min.   : 7.00   Min.   :18.20   Min.   :0.0850   Min.   :21.00
##  1st Qu.:21.00   1st Qu.:27.88   1st Qu.:0.2532   1st Qu.:23.00
##  Median :29.00   Median :32.90   Median :0.4075   Median :28.00
##  Mean   :29.22   Mean   :33.09   Mean   :0.4789   Mean   :31.48
##  3rd Qu.:37.00   3rd Qu.:37.12   3rd Qu.:0.6525   3rd Qu.:37.25
##  Max.   :60.00   Max.   :67.10   Max.   :2.1370   Max.   :70.00
```

```
##
##  0  1  2  3  4  5  6  7  8  9 10 11 12 13 15 17
## 37 76 43 34 22 16 16 17 10 12  6  5  3  1  1  1
```



Full tree performance

```
##
## Classification tree:
## tree(formula = factor(diabetes) ~ npreg + glu + bp + skin + bmi +
##     ped + age, data = ctrain)
## Variables actually used in tree construction:
## [1] "glu"  "age"  "bmi"  "ped"  "bp"   "skin"
## Number of terminal nodes:  20
## Residual mean deviance:  0.4805 = 134.5 / 280
## Misclassification error rate: 0.12 = 36 / 300
```




```
## node), split, n, deviance, yval, (yprob)
##       * denotes terminal node
##
##   1) root 300 381.900 0 ( 0.66667 0.33333 )
##     2) glu < 127.5 189 168.700 0 ( 0.83598 0.16402 )
##       4) age < 28.5 121  64.090 0 ( 0.92562 0.07438 )
##         8) bmi < 31.4 62   0.000 0 ( 1.00000 0.00000 ) *
##         9) bmi > 31.4 59  50.400 0 ( 0.84746 0.15254 )
##          18) ped < 0.4895 40  15.880 0 ( 0.95000 0.05000 )
##            36) ped < 0.1975 12  10.810 0 ( 0.83333 0.16667 ) *
##            37) ped > 0.1975 28   0.000 0 ( 1.00000 0.00000 ) *
##          19) ped > 0.4895 19  25.010 0 ( 0.63158 0.36842 )
##            38) bp < 69 14  19.410 1 ( 0.50000 0.50000 ) *
##            39) bp > 69 5   0.000 0 ( 1.00000 0.00000 ) *
##       5) age > 28.5 68  85.610 0 ( 0.67647 0.32353 )
##        10) glu < 96.5 20   0.000 0 ( 1.00000 0.00000 ) *
##        11) glu > 96.5 48  66.210 0 ( 0.54167 0.45833 )
##          22) ped < 0.5205 28  31.490 0 ( 0.75000 0.25000 )
##            44) skin < 30.5 9  12.370 1 ( 0.44444 0.55556 ) *
##            45) skin > 30.5 19  12.790 0 ( 0.89474 0.10526 )
##              90) bp < 69 5   6.730 0 ( 0.60000 0.40000 ) *
##              91) bp > 69 14   0.000 0 ( 1.00000 0.00000 ) *
##          23) ped > 0.5205 20  22.490 1 ( 0.25000 0.75000 )
##            46) ped < 0.7275 10   0.000 1 ( 0.00000 1.00000 ) *
##            47) ped > 0.7275 10  13.860 1 ( 0.50000 0.50000 )
##              94) glu < 109.5 5   5.004 0 ( 0.80000 0.20000 ) *
##              95) glu > 109.5 5   5.004 1 ( 0.20000 0.80000 ) *
##     3) glu > 127.5 111 147.200 1 ( 0.37838 0.62162 )
##       6) glu < 159 70  97.040 1 ( 0.50000 0.50000 )
##        12) age < 23.5 11   0.000 0 ( 1.00000 0.00000 ) *
##        13) age > 23.5 59  79.730 1 ( 0.40678 0.59322 )
##          26) bmi < 26.2 5   0.000 0 ( 1.00000 0.00000 ) *
##          27) bmi > 26.2 54  70.050 1 ( 0.35185 0.64815 )
##            54) glu < 130 9   0.000 1 ( 0.00000 1.00000 ) *
##            55) glu > 130 45  61.290 1 ( 0.42222 0.57778 )
##             110) age < 41.5 26  34.650 0 ( 0.61538 0.38462 )
```

```
##               220) age < 35.5 19  26.290 1 ( 0.47368 0.52632 ) *
##               221) age > 35.5 7   0.000 0 ( 1.00000 0.00000 ) *
##             111) age > 41.5 19  16.570 1 ( 0.15789 0.84211 )
##               222) bp < 81 9  11.460 1 ( 0.33333 0.66667 ) *
##               223) bp > 81 10   0.000 1 ( 0.00000 1.00000 ) *
##       7) glu > 159 41  37.480 1 ( 0.17073 0.82927 ) *
```

```
## [1] "Performance on training set"
```

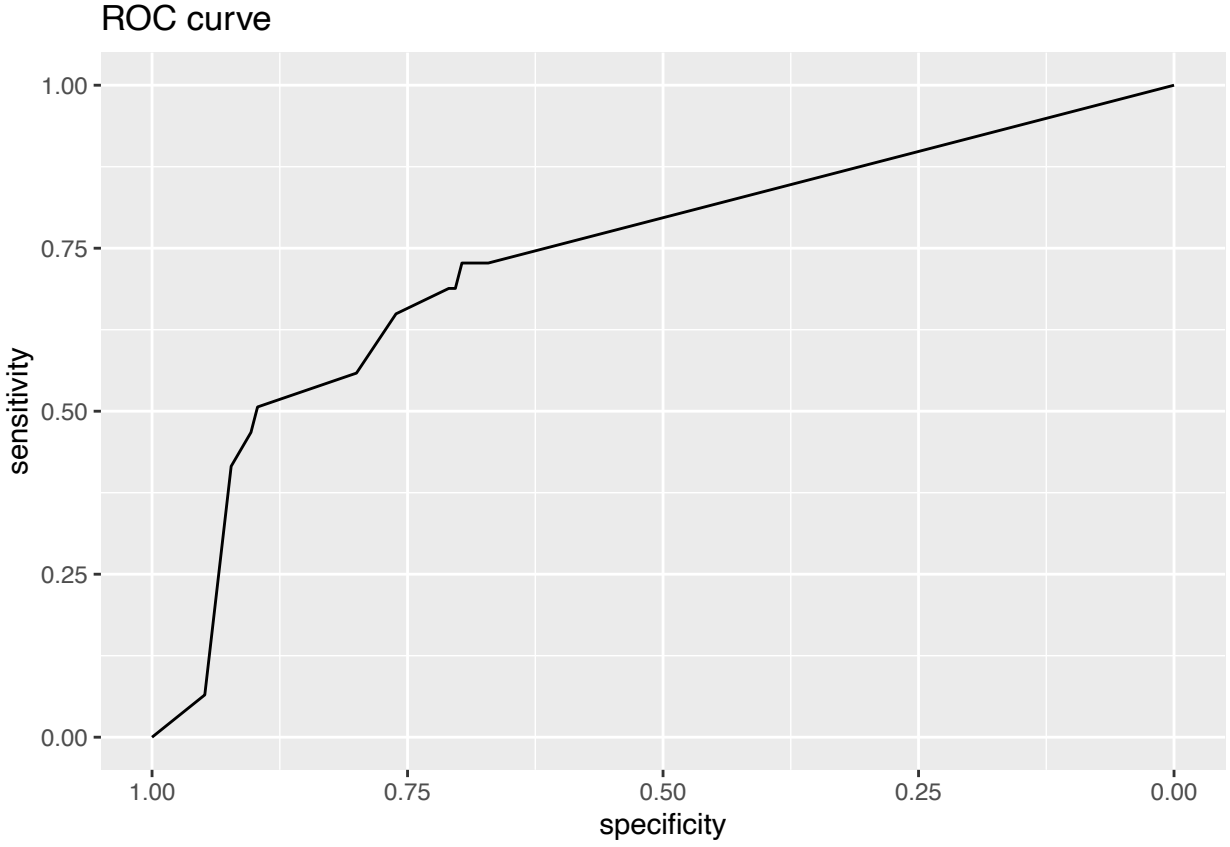
```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 169   5
##          1  31  95
##
##                Accuracy : 0.88
##                  95% CI : (0.8378, 0.9145)
##     No Information Rate : 0.6667
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.7465
##
##  Mcnemar's Test P-Value : 3.091e-05
##
##             Sensitivity : 0.8450
##             Specificity : 0.9500
##          Pos Pred Value : 0.9713
##          Neg Pred Value : 0.7540
##              Prevalence : 0.6667
##          Detection Rate : 0.5633
##    Detection Prevalence : 0.5800
##       Balanced Accuracy : 0.8975
##
##        'Positive' Class : 0
##
```

```
## [1] "Performance on test set"
```

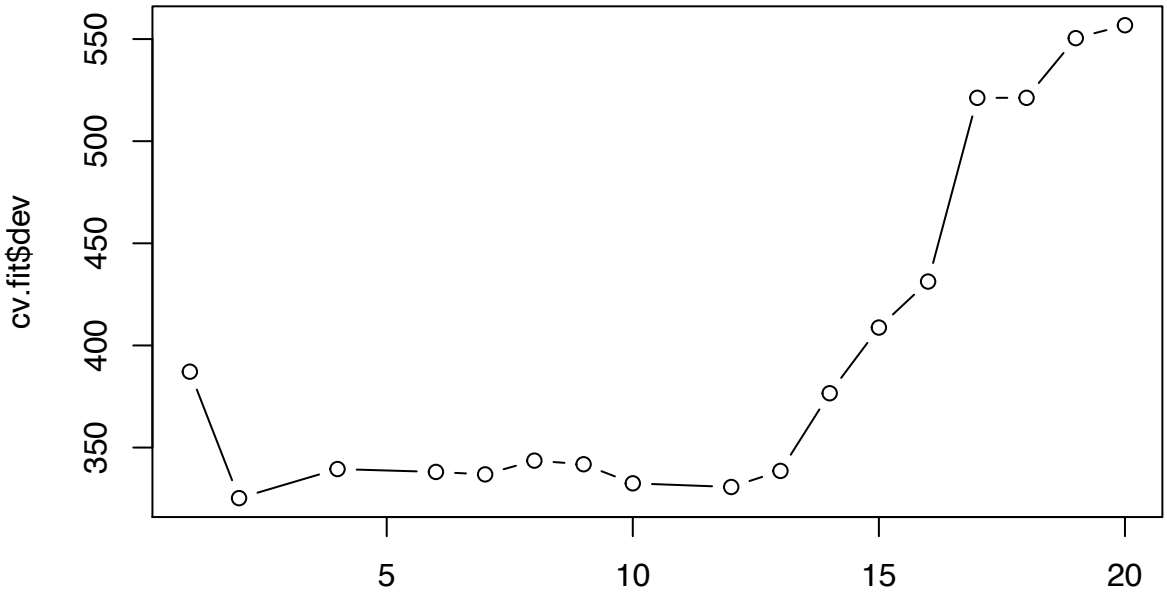
```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 110  24
##          1  45  53
##
##                Accuracy : 0.7026
##                  95% CI : (0.6393, 0.7606)
##     No Information Rate : 0.6681
##     P-Value [Acc > NIR] : 0.14766
##
##                   Kappa : 0.3724
##
##  Mcnemar's Test P-Value : 0.01605
##
##             Sensitivity : 0.7097
```

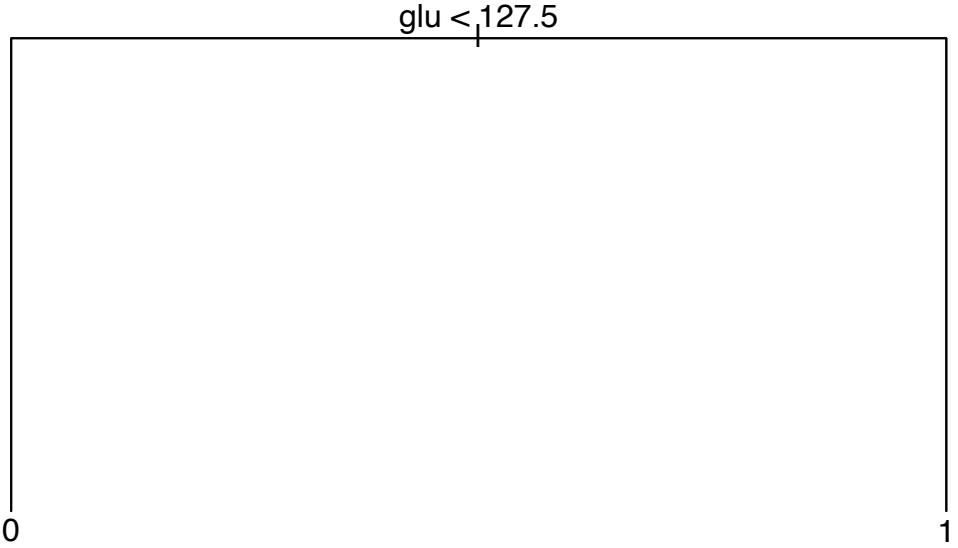
```
##             Specificity : 0.6883
##          Pos Pred Value : 0.8209
##          Neg Pred Value : 0.5408
##              Prevalence : 0.6681
##          Detection Rate : 0.4741
##    Detection Prevalence : 0.5776
##       Balanced Accuracy : 0.6990
##
##        'Positive' Class : 0
##

## Area under the curve: 0.7362
```



Pruned tree performance






```
## node), split, n, deviance, yval, (yprob)
##       * denotes terminal node
##
##   1) root 300 381.900 0 ( 0.66667 0.33333 )
##     2) glu < 127.5 189 168.700 0 ( 0.83598 0.16402 )
##       4) age < 28.5 121  64.090 0 ( 0.92562 0.07438 )
##         8) bmi < 31.4 62   0.000 0 ( 1.00000 0.00000 ) *
##         9) bmi > 31.4 59  50.400 0 ( 0.84746 0.15254 )
##          18) ped < 0.4895 40  15.880 0 ( 0.95000 0.05000 )
##            36) ped < 0.1975 12  10.810 0 ( 0.83333 0.16667 ) *
##            37) ped > 0.1975 28   0.000 0 ( 1.00000 0.00000 ) *
##          19) ped > 0.4895 19  25.010 0 ( 0.63158 0.36842 )
##            38) bp < 69 14  19.410 1 ( 0.50000 0.50000 ) *
##            39) bp > 69 5   0.000 0 ( 1.00000 0.00000 ) *
```

```
##       5) age > 28.5 68  85.610 0 ( 0.67647 0.32353 )
##        10) glu < 96.5 20   0.000 0 ( 1.00000 0.00000 ) *
##        11) glu > 96.5 48  66.210 0 ( 0.54167 0.45833 )
##          22) ped < 0.5205 28  31.490 0 ( 0.75000 0.25000 )
##            44) skin < 30.5 9  12.370 1 ( 0.44444 0.55556 ) *
##            45) skin > 30.5 19  12.790 0 ( 0.89474 0.10526 )
##              90) bp < 69 5   6.730 0 ( 0.60000 0.40000 ) *
##              91) bp > 69 14   0.000 0 ( 1.00000 0.00000 ) *
##          23) ped > 0.5205 20  22.490 1 ( 0.25000 0.75000 )
##            46) ped < 0.7275 10   0.000 1 ( 0.00000 1.00000 ) *
##            47) ped > 0.7275 10  13.860 1 ( 0.50000 0.50000 )
##              94) glu < 109.5 5   5.004 0 ( 0.80000 0.20000 ) *
##              95) glu > 109.5 5   5.004 1 ( 0.20000 0.80000 ) *
##     3) glu > 127.5 111 147.200 1 ( 0.37838 0.62162 )
##       6) glu < 159 70  97.040 1 ( 0.50000 0.50000 )
##        12) age < 23.5 11   0.000 0 ( 1.00000 0.00000 ) *
##        13) age > 23.5 59  79.730 1 ( 0.40678 0.59322 )
##          26) bmi < 26.2 5   0.000 0 ( 1.00000 0.00000 ) *
##          27) bmi > 26.2 54  70.050 1 ( 0.35185 0.64815 )
##            54) glu < 130 9   0.000 1 ( 0.00000 1.00000 ) *
##            55) glu > 130 45  61.290 1 ( 0.42222 0.57778 )
##             110) age < 41.5 26  34.650 0 ( 0.61538 0.38462 )
##               220) age < 35.5 19  26.290 1 ( 0.47368 0.52632 ) *
##               221) age > 35.5 7   0.000 0 ( 1.00000 0.00000 ) *
##             111) age > 41.5 19  16.570 1 ( 0.15789 0.84211 )
##               222) bp < 81 9  11.460 1 ( 0.33333 0.66667 ) *
##               223) bp > 81 10   0.000 1 ( 0.00000 1.00000 ) *
##       7) glu > 159 41  37.480 1 ( 0.17073 0.82927 ) *
```

```
## [1] "Performance on training set"
```

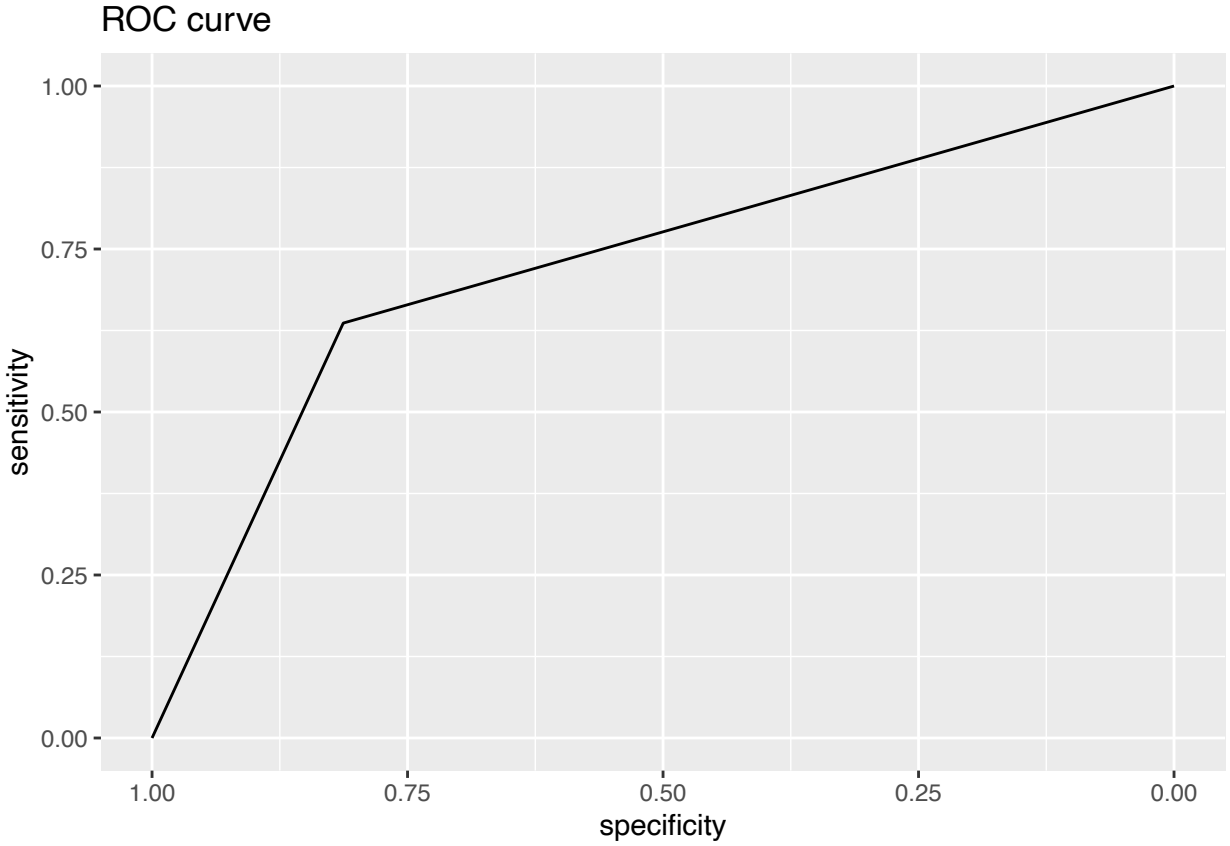
```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 158  31
##          1  42  69
##
##                Accuracy : 0.7567
##                  95% CI : (0.704, 0.8041)
##     No Information Rate : 0.6667
##     P-Value [Acc > NIR] : 0.0004426
##
##                   Kappa : 0.4672
##
##  Mcnemar's Test P-Value : 0.2418354
##
##             Sensitivity : 0.7900
##             Specificity : 0.6900
##          Pos Pred Value : 0.8360
##          Neg Pred Value : 0.6216
##              Prevalence : 0.6667
##          Detection Rate : 0.5267
##    Detection Prevalence : 0.6300
##       Balanced Accuracy : 0.7400
```

```
##
##        'Positive' Class : 0
##
```

```
## [1] "Performance on test set"
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 126  28
##          1  29  49
##
##                Accuracy : 0.7543
##                  95% CI : (0.6937, 0.8083)
##     No Information Rate : 0.6681
##     P-Value [Acc > NIR] : 0.002725
##
##                   Kappa : 0.4478
##
##  Mcnemar's Test P-Value : 1.000000
##
##             Sensitivity : 0.8129
##             Specificity : 0.6364
##          Pos Pred Value : 0.8182
##          Neg Pred Value : 0.6282
##              Prevalence : 0.6681
##          Detection Rate : 0.5431
##    Detection Prevalence : 0.6638
##       Balanced Accuracy : 0.7246
##
##        'Positive' Class : 0
##
```

```
## Area under the curve: 0.7246
```



Tree issues

Building a regression (classification) tree

1. Use recursive binary splitting to grow a large tree on the training data, stopping only when each terminal node has fewer than some minimum number of observations.
2. Apply cost complexity pruning to the large tree in order to obtain a sequence of best sub-trees, as a function of α.
3. Use K-fold cross-validation to choose α. That is, divide the training observations into K folds. For each k $= 1, \ldots, K$:
   - Repeat Steps 1 and 2 on all but the kth fold of the training data.
   - Evaluate the mean squared prediction (misclassification, gini, cross-entropy) error on the data in the left-out kth fold, as a function of α.
   - Average the results for each value of α, and pick α to minimize the average error.
4. Return the sub-tree from Step 2 that corresponds to the chosen value of α.

Missing covariates

(ELS 9.6, and van Buuren: "Flexible imputation of missing data")

When performing data analysis we often encounter data sets where some observations have missing values. It is important to understand the underlying mechanism for the observations to be missing, so that we may treat the missing data appropriately.

Tree issues

Building a regression (classification) tree

1. Use recursive binary splitting to grow a large tree on the training data, stopping only when each terminal node has fewer than some minimum number of observations.
2. Apply cost complexity pruning to the large tree in order to obtain a sequence of best sub-trees, as a function of α.
3. Use K-fold cross-validation to choose α. That is, divide the training observations into K folds. For each k $= 1, \ldots, K$:
   - Repeat Steps 1 and 2 on all but the kth fold of the training data.
   - Evaluate the mean squared prediction (misclassification, gini, cross-entropy) error on the data in the left-out kth fold, as a function of α.
   - Average the results for each value of α, and pick α to minimize the average error.
4. Return the sub-tree from Step 2 that corresponds to the chosen value of α.

Conclusions