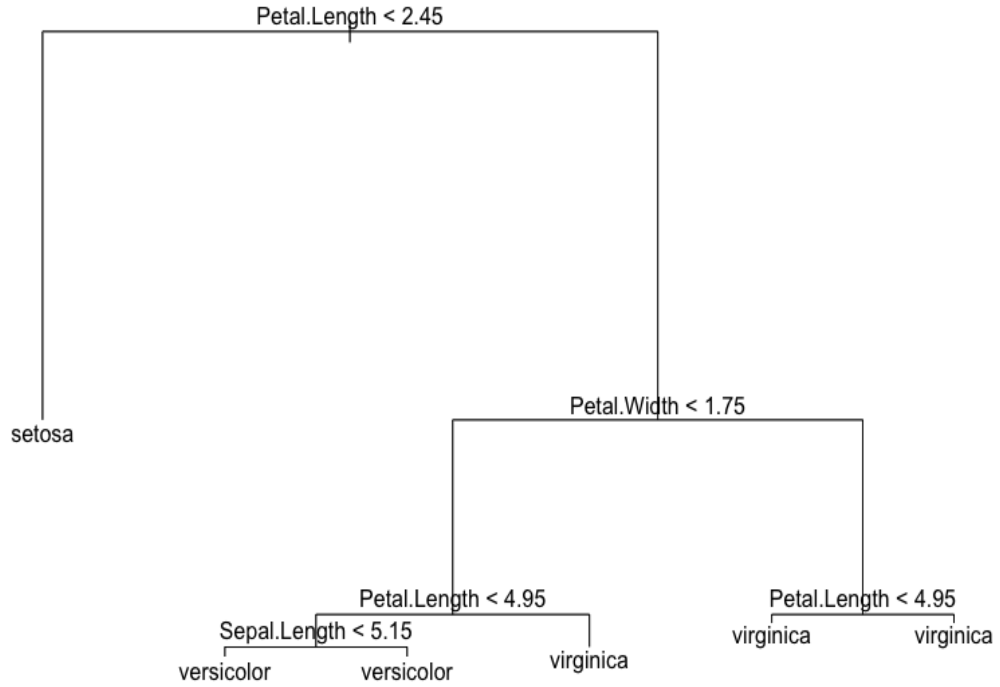
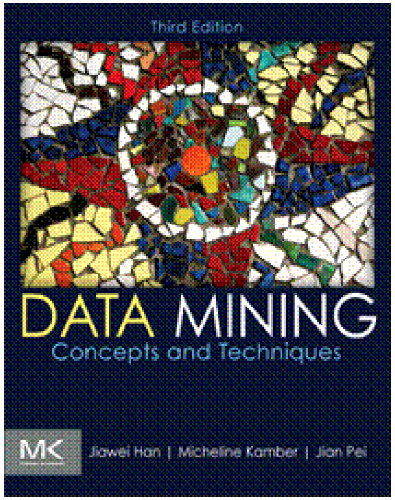
Classification by Decision Tree Induction

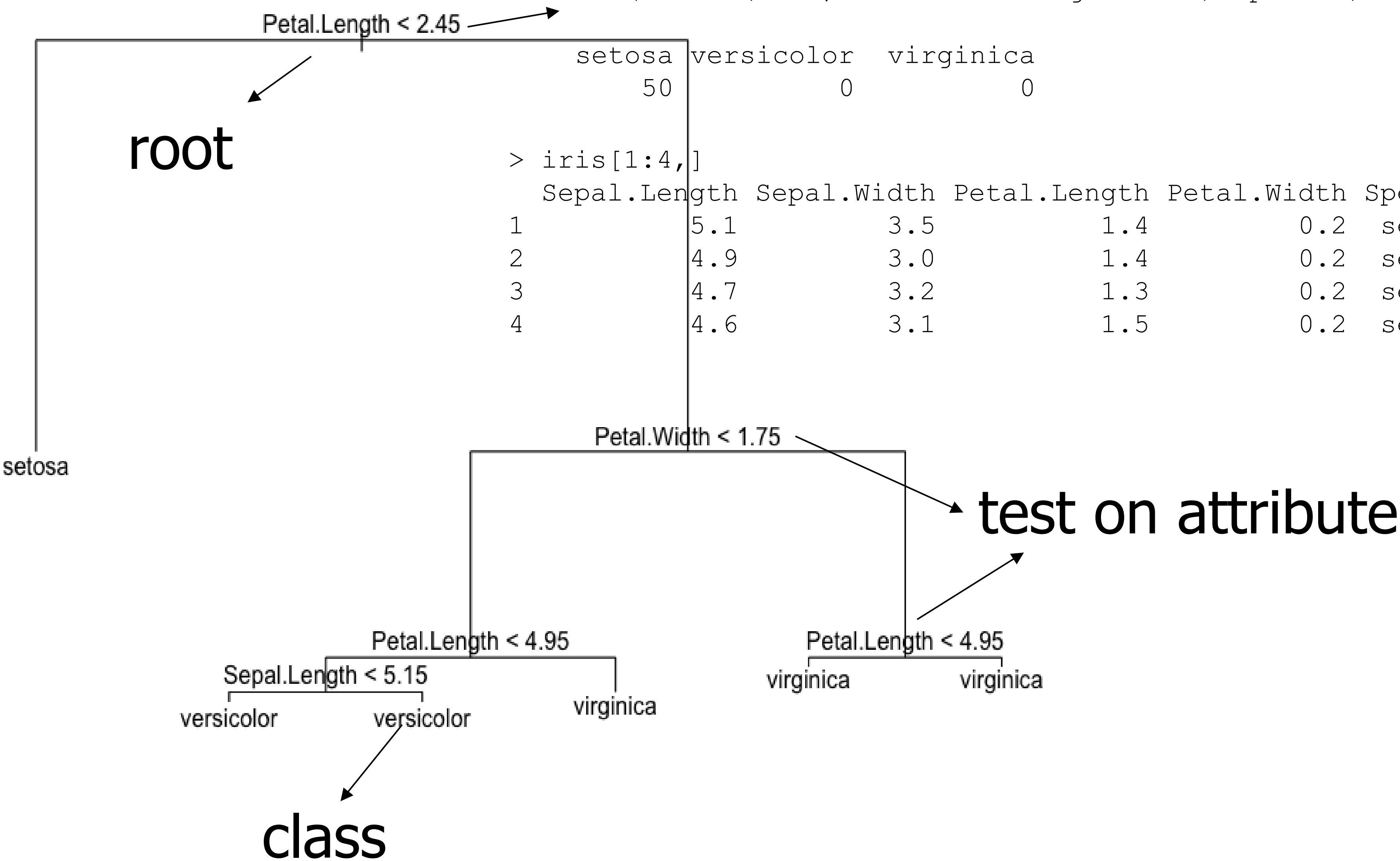
Ronnie Alves (alvesrco@gmail.com)
https://sites.google.com/site/alvesrco/





Decision tree: An Example

```
> table(subset(iris,iris$Petal.Length<2.45)$Species)

    setosa versicolor  virginica
        50          0          0

> iris[1:4,]
  Sepal.Length Sepal.Width Petal.Length Petal.Width Species
1          5.1         3.5          1.4         0.2  setosa
2          4.9         3.0          1.4         0.2  setosa
3          4.7         3.2          1.3         0.2  setosa
4          4.6         3.1          1.5         0.2  setosa
```



Algorithm for Decision Tree Induction

- Basic algorithm (a greedy algorithm)
  - Tree is constructed in a top-down recursive divide-and-conquer manner
  - At start, all the training examples are at the root
  - Attributes are categorical (if continuous-valued, they are discretized in advance)
  - Examples are partitioned recursively based on selected attributes
  - Test attributes are selected on the basis of a heuristic or statistical measure (e.g., information gain)
- Conditions for stopping partitioning
  - All samples for a given node belong to the same class
  - There are no remaining attributes for further partitioning – majority voting is employed for classifying the leaf
  - There are no samples left

Decision Trees: Pioneers

J. Ross Quinlan, ID3 (Iterative Dichotomiser), 1978-1980s

- C4.5

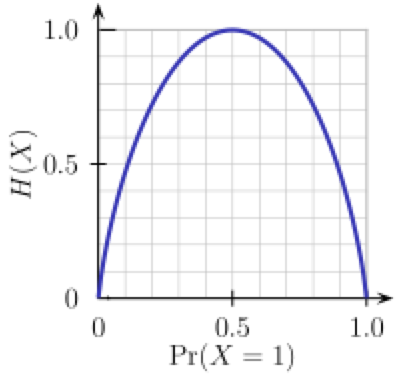
L. Breiman, J. Friedman, R. Olshen, and C. Stone publish the book Classification and Regression Trees (CART) , binary decision trees, 1984.

ID3, C4.5, and CART adopt a **greedy** (i.e. nonbacktracking) approach.

Most algorithms for decision tree induction also follow a **top-down approach**.

Brief Review of Entropy

- Entropy (Information Theory)
  - A measure of uncertainty associated with a random variable
  - Calculation: For a discrete random variable *Y* taking *m* distinct values $\{y_1, \dots, y_m\}$,
    - $H(Y) = -\sum_{i=1}^{m} p_i \log(p_i)$, where $p_i = P(Y = y_i)$
  - Interpretation:
    - Higher entropy => higher uncertainty
    - Lower entropy => lower uncertainty
- Conditional Entropy
  - $H(Y|X) = \sum_x p(x) H(Y|X = x)$



m = 2

Entropy: An Example

Given a set S, containing only positive and negative examples of some target concept (a **2 class problem**), the entropy of set S relative to this simple, binary classification is defined as:

Entropy(S) = - pp log2 pp – pn log2 pn

To illustrate, suppose S is a collection of 25 examples, including 15 positive and 10 negative examples [15+, 10-]. Then the entropy of S relative to this classification is

Entropy(S) = - (15/25) log2 (15/25) - (10/25) log2 (10/25) = 0.970

Attribute Selection: Information Gain

Select the attribute with the **highest information gain**

Let p_i be the probability that an arbitrary tuple in D belongs to class C_i, estimated by $|C_{i, D}|/|D|$

Expected information (entropy) needed to classify a tuple in D:

$$Info(D) = -\sum_{i=1}^{m} p_i \log_2(p_i)$$

Information needed (after using A to split D into v partitions) to classify D:

$$Info_A(D) = \sum_{j=1}^{v} \frac{|D_j|}{|D|} \times Info(D_j)(D_j)$$

Information gained by branching on attribute A

$$Gain(A) = Info(D) - Info_A(D)$$

Attribute Selection: Information Gain

Class P: buys_computer = "yes" (9)
Class N: buys_computer = "no" (5)

(i) $$Info(D) = I(9,5) = -\frac{9}{14}\log_2\left(\frac{9}{14}\right) - \frac{5}{14}\log_2\left(\frac{5}{14}\right) = 0.940$$

age	p_i	n_i	$I(p_i, n_i)$
<=30	2	3	0.971
31…40	4	0	0
>40	3	2	0.971

age	income	student	credit_rating	buys_computer
<=30	high	no	fair	no
<=30	high	no	excellent	no
31…40	high	no	fair	yes
>40	medium	no	fair	yes
>40	low	yes	fair	yes
>40	low	yes	excellent	no
31…40	low	yes	excellent	yes
<=30	medium	no	fair	no
<=30	low	yes	fair	yes
>40	medium	yes	fair	yes
<=30	medium	yes	excellent	yes
31…40	medium	no	excellent	yes
31…40	high	yes	fair	yes
>40	medium	no	excellent	no

(ii) $$Info_{age}(D) = \frac{5}{14}I(2,3) + \frac{4}{14}I(4,0) + \frac{5}{14}I(3,2) = 0.694$$

$\frac{5}{14}I(2,3)$ means "age <=30" has 5 out of 14 samples, with 2 yes'es and 3 no's. Hence

(iii) $$Gain(age) = Info(D) - Info_{age}(D) = 0.246$$

Similarly,

$$Gain(income) = 0.029$$
$$Gain(student) = 0.151$$
$$Gain(credit_{rating}) = 0.048$$

Computing Information-Gain for Continuous-Valued Attributes

Let attribute A be a continuous-valued attribute

Must determine the *best split point* for A

Sort the value A in increasing order

Typically, **the midpoint between each pair of adjacent values** is considered as a possible *split point*

- $(a_i+a_{i+1})/2$ is the midpoint between the values of a_i and a_{i+1}

The point with the *minimum expected information requirement* for A is selected as the split-point for A

Split:

D1 is the set of tuples in D satisfying A ≤ split-point, and D2 is the set of tuples in D satisfying A > split-point

Attribute Selection: Gain Ratio

Information gain measure **is biased towards attributes with a large number of values**

C4.5 (a successor of ID3) uses gain ratio to overcome the problem (normalization to information gain)

$$SplitInfo_A(D) = -\sum_{j=1}^{v} \frac{|D_j|}{|D|} \times \log_2\left(\frac{|D_j|}{|D|}\right)$$

GainRatio(A) = Gain(A)/SplitInfo(A)

Ex.

$$SplitInfo_{income}(D) = -\underbrace{\frac{4}{14} \times \log_2\left(\frac{4}{14}\right)}_{\text{low}} - \overbrace{\frac{6}{14} \times \log_2\left(\frac{6}{14}\right)}^{\text{medium}} - \underbrace{\frac{4}{14} \times \log_2\left(\frac{4}{14}\right)}_{\text{high}} = 1.557$$

gain_ratio(income) = 0.029/1.557 = 0.019

The attribute with the **maximum gain ratio** is selected as the splitting attribute

Attribute Selection: Gini Index

If a data set D contains examples from n classes, gini index, $gini(D)$ is defined as

$$gini(D) = 1 - \sum_{j=1}^{n} p_j^2$$

where p_j is the relative frequency of class j in D

If a data set D is split on A into two subsets D_1 and D_2, the *gini* index $gini(D)$ is defined as

$$gini_A(D) = \frac{|D_1|}{|D|} gini(D_1) + \frac{|D_2|}{|D|} gini(D_2)$$

Reduction in Impurity:

$$\Delta gini(A) = gini(D) - gini_A(D)$$

The attribute provides the smallest $gini_{split}(D)$ (or the largest reduction in impurity) is chosen to split the node (*need to enumerate all the possible splitting points for each attribute*)

Attribute Selection: Gini Index

Ex. D has 9 tuples in buys_computer = "yes" and 5 in "no"

Suppose the attribute income partitions D into 10 in D_1: {low, medium} and 4 in D_2

$$gini(D) = 1 - \left(\frac{9}{14}\right)^2 - \left(\frac{5}{14}\right)^2 = 0.459$$

$$gini_{income \in \{low, medium\}}(D) = \left(\frac{10}{14}\right) Gini(D_1) + \left(\frac{4}{14}\right) Gini(D_2)$$

$$= \frac{10}{14}\left(1 - \left(\frac{7}{10}\right)^2 - \left(\frac{3}{10}\right)^2\right) + \frac{4}{14}\left(1 - \left(\frac{2}{4}\right)^2 - \left(\frac{2}{4}\right)^2\right)$$

$$= 0.443$$

$$= Gini_{income \in \{high\}}(D).$$

$Gini_{\{low,high\}}$ is 0.458; $Gini_{\{medium,high\}}$ is 0.450. Thus, split on the {low,medium} (and {high}) since it has the lowest Gini index

Task-4: What attribute will give the minimum Gini index?

Attribute Selection: Biases

The three measures, in general, return good results but

Information gain:

- biased towards multivalued attributes

Gain ratio:

- tends to prefer unbalanced splits in which one partition is much smaller than the others

Gini index:

- biased to multivalued attributes
- has difficulty when # of classes is large
- tends to favor tests that result in equal-sized partitions and purity in both partitions

Attribute Selection: Other Measures

CHAID: a popular decision tree algorithm, measure based on χ^2 test for independence

C-SEP: performs better than info. gain and gini index in certain cases

G-statistic: has a close approximation to χ^2 distribution

MDL (Minimal Description Length) principle (i.e., the simplest solution is preferred):

The best tree as the one that requires the fewest # of bits to both (1) encode the tree, and (2) encode the exceptions to the tree

Multivariate splits (partition based on multiple variable combinations)

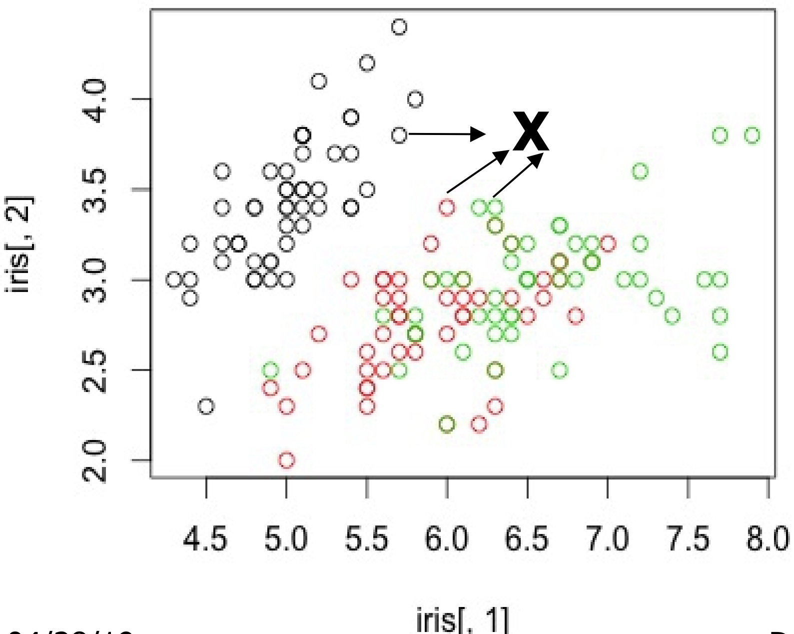
CART: finds multivariate splits based on a linear comb. of attrs.

Others...

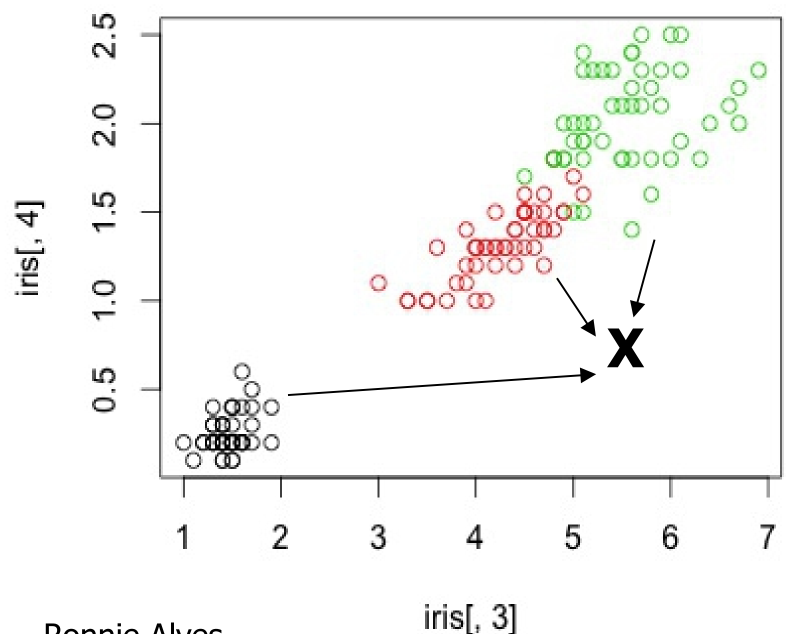
What kind of flower is this?

i) setosa, (ii) versicolor, iii) virginica

Sepals



Petals



Decision Tree: *tree* R Package

```
####### Decision trees for classification

####### By Ronnie Alves

library(tree)

attach(iris)

iris[1:4,]
```

```
> iris[1:4,]
  Sepal.Length Sepal.Width Petal.Length Petal.Width Species
1          5.1         3.5          1.4         0.2  setosa
2          4.9         3.0          1.4         0.2  setosa
3          4.7         3.2          1.3         0.2  setosa
4          4.6         3.1          1.5         0.2  setosa
```

```
xtabs(~Species,data=iris)
```

```
> xtabs(~Species,data=iris)
Species
    setosa versicolor  virginica
        50         50         50
```

Decision Tree: *tree* R Package

```
dt1 = tree(Species ~ ., iris, split="gini")

dt1
```

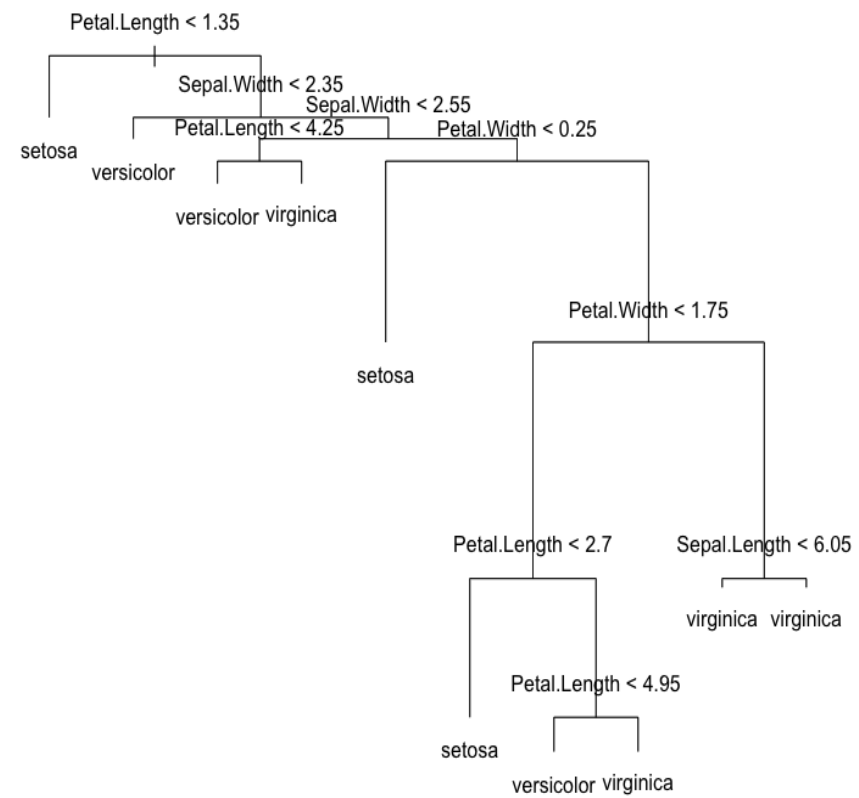
```
node), split, n, deviance, yval, (yprob)
      * denotes terminal node

  1) root 150 329.600 setosa ( 0.33333 0.33333 0.33333 )
    2) Petal.Length < 1.35 11   0.000 setosa ( 1.00000 0.00000 0.00000 ) *
    3) Petal.Length > 1.35 139 303.600 versicolor ( 0.28058 0.35971 0.35971 )
      6) Sepal.Width < 2.35 7   5.742 versicolor ( 0.00000 0.85714 0.14286 ) *
      7) Sepal.Width > 2.35 132 288.900 virginica ( 0.29545 0.33333 0.37121 )
       14) Sepal.Width < 2.55 11  14.420 versicolor ( 0.00000 0.63636 0.36364 )
         28) Petal.Length < 4.25 6   0.000 versicolor ( 0.00000 1.00000 0.00000 ) *
         29) Petal.Length > 4.25 5   5.004 virginica ( 0.00000 0.20000 0.80000 ) *
       15) Sepal.Width > 2.55 121 265.000 virginica ( 0.32231 0.30579 0.37190 )
         30) Petal.Width < 0.25 26   0.000 setosa ( 1.00000 0.00000 0.00000 ) *
         31) Petal.Width > 0.25 95 188.700 virginica ( 0.13684 0.38947 0.47368 )
           62) Petal.Width < 1.75 52  79.640 versicolor ( 0.25000 0.69231 0.05769 )
            124) Petal.Length < 2.7 13   0.000 setosa ( 1.00000 0.00000 0.00000 ) *
            125) Petal.Length > 2.7 39  21.150 versicolor ( 0.00000 0.92308 0.07692 )
              250) Petal.Length < 4.95 34   0.000 versicolor ( 0.00000 1.00000 0.00000 ) *
              251) Petal.Length > 4.95 5   6.730 virginica ( 0.00000 0.40000 0.60000 ) *
           63) Petal.Width > 1.75 43   9.499 virginica ( 0.00000 0.02326 0.97674 )
            126) Sepal.Length < 6.05 7   5.742 virginica ( 0.00000 0.14286 0.85714 ) *
            127) Sepal.Length > 6.05 36   0.000 virginica ( 0.00000 0.00000 1.00000 ) *
```

Decision Tree: *tree* R Package

```
summary(dt1)
```

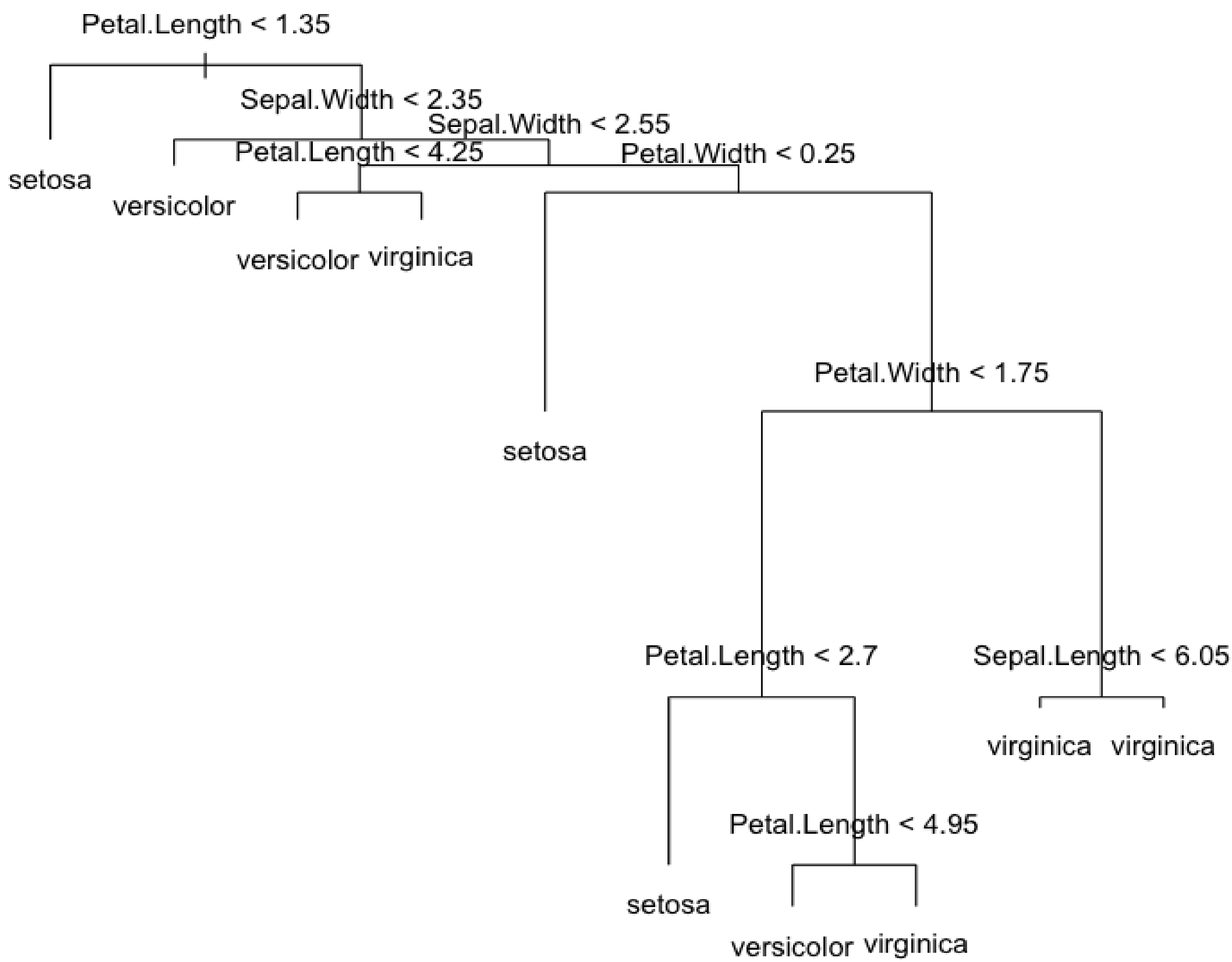
```
> summary(dt1)

Classification tree:
tree(formula = Species ~ ., data = iris, split = "gini")
Number of terminal nodes:  10
Residual mean deviance:  0.1658 = 23.22 / 140
Misclassification error rate: 0.03333 = 5 / 150
```

```
plot(dt1)

text(dt1)
```



Decision Tree: *tree* R Package

```
summary(dt1)
```




```
plot(dt1)

text(dt1)
```

Decision Tree: *tree* R Package

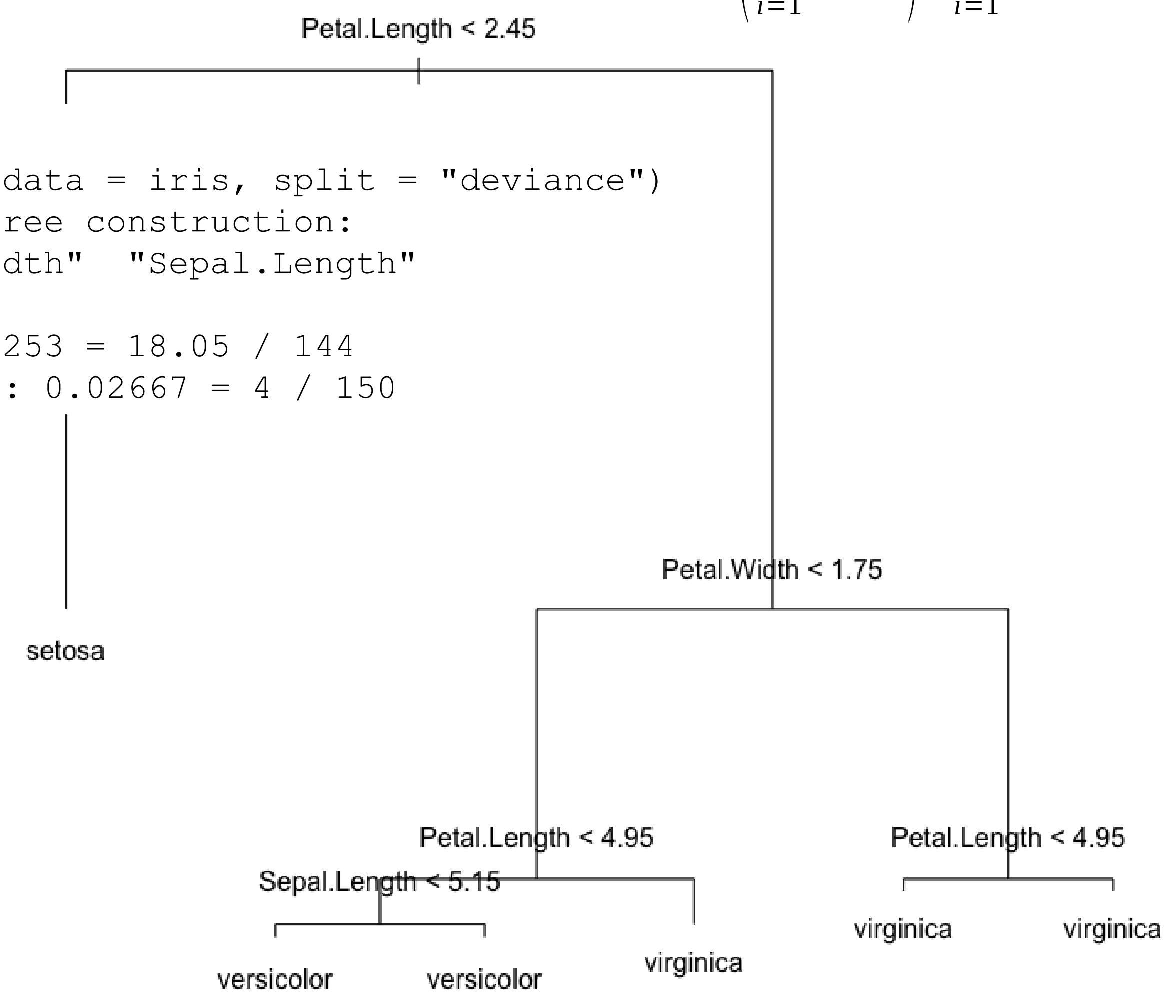
```
dt2 = tree(Species ~ ., iris, split="deviance")
```

$$-2\,LogL = -2\log\left(\prod_{i=1}^{n} f(y_i)\right) = \sum_{i=1}^{n} -2\log f(y_i)$$

```
summary(dt2)
```

```
Classification tree:
tree(formula = Species ~ ., data = iris, split = "deviance")
Variables actually used in tree construction:
[1] "Petal.Length" "Petal.Width"  "Sepal.Length"
Number of terminal nodes:  6
Residual mean deviance:  0.1253 = 18.05 / 144
Misclassification error rate: 0.02667 = 4 / 150
```

```
plot(dt2)

text(dt2)
```



Which attribute selection measure is the best?

Time complexity of decision tree induction generally increases exponentially with **tree height**.

Measures that tend to produce **shallower trees** may be preferred. Though, shallow trees tend to have a large number of leaves and **high error rates**.

Despite several comparative studies, no one attribute selection measure has been found to be significantly superior to others.

Most measures give good results!

Decision tree: Summary

The construction of decision tree classifiers does not require any domain knowledge or parameter setting.

Exploratory data analysis.

Decision tree can **handle high-dimensional data**.

Intuitive knowledge representation.

Decision trees are biased towards **attribute selection measures** (information gain, gain ratio, gini index...).

Decision trees usually **overfitting** the data.

TASK-5: How scalable is decision tree induction? What about other efficient strategies/algorithms?

Tree Pruning

prepruning

postpruning

Tree Pruning

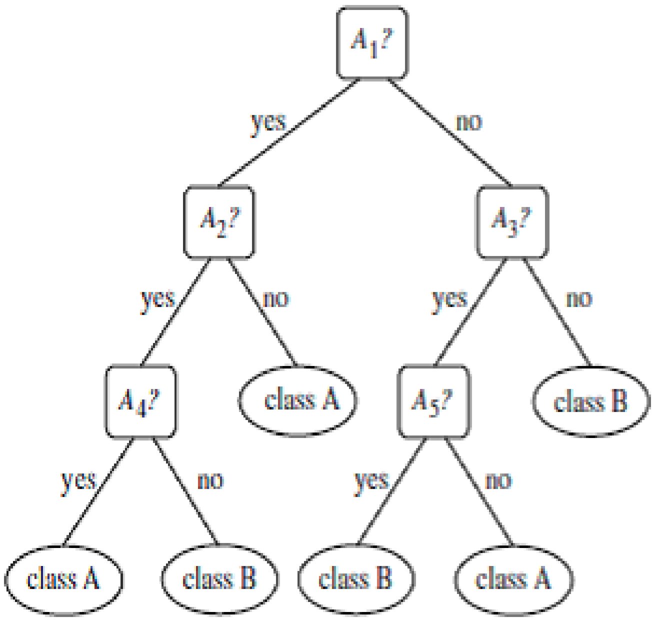
When decision tree is built, many of the branches will reflect **anomalies** in the training data due to **noise** or **outliers**.

Tree pruning methods address this problem of **overfitting** the data.

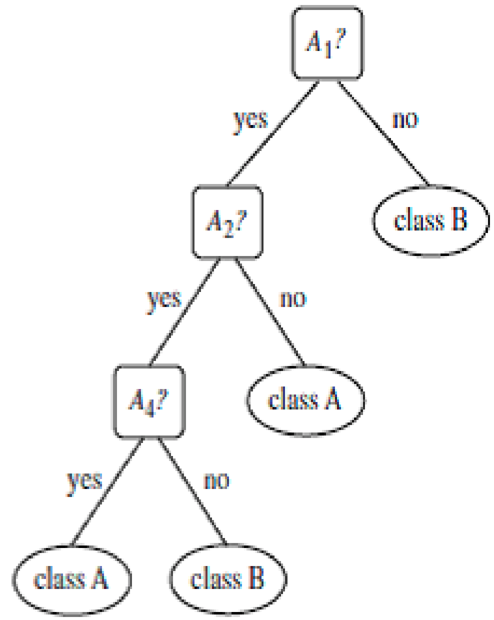
The main goal is to use **statistical measures** to remove the **least reliable branches**.

Pruned trees tend to be smaller and **less complex** and, thus easier to comprehend. Furthermore, they are **faster** and **better** at correctly classifying independent test data.

Tree Pruning



unpruned



pruned

Prepruning

A tree is pruned by **halting** its construction early. Upon halting, the node becomes a leaf.

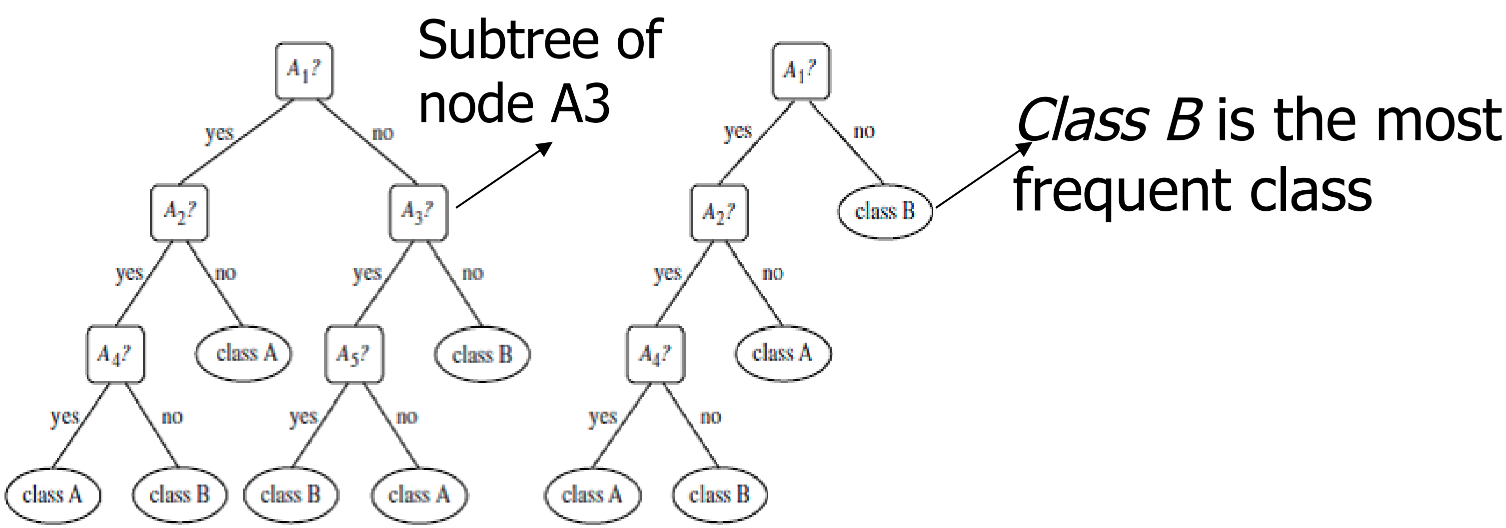
Measures like statistical significance, information gain, Gini index, and so on can be used to asses the **goodness of a split**.

- The split should falls bellow a **threshold**
- **High** threshold could result in **oversimplified** trees, whereas **low** threshold could result in very **little simplification**

Postpruning

Removes subtrees from a fully grown tree. A subtree at a given node is pruned by **removing its branches** and replacing it with a leaf.

The leaf is labelled with the **most frequent class** among the replaced subtree.



Postpruning: CART

The approach considers the **cost complexity of a tree** to be a function of the number of leaves in the tree and the **error rate** of the tree.

Starting from the bottom of the tree, for each internal node *N*, it computes the cost complexity of the subtree at *N*, and the cost complexity of the subtree at *N* if it were to be pruned. If pruning the subtree result in smaller cost complexity, then subtree is pruned.

A **pruning set** of class-labeled tuples is used to estimate cost complexity. The smallest decision tree that **minimizes** the **cost complexity** is preferred.

Cost Complexity

$$CC(T) = Err(T) + \alpha L(T)$$

CC(T) = cost complexity of a tree

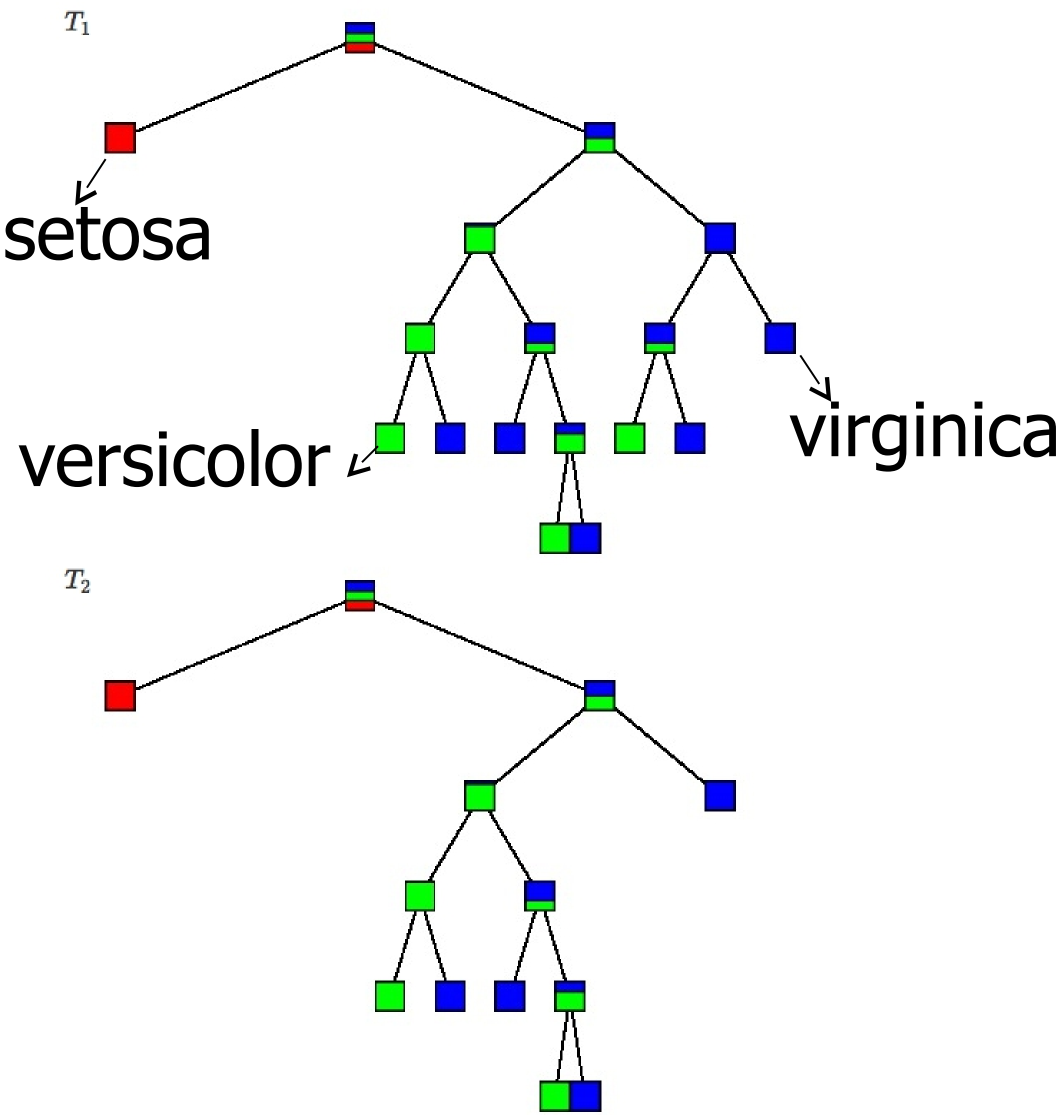
Err(T) = proportion of misclassified records

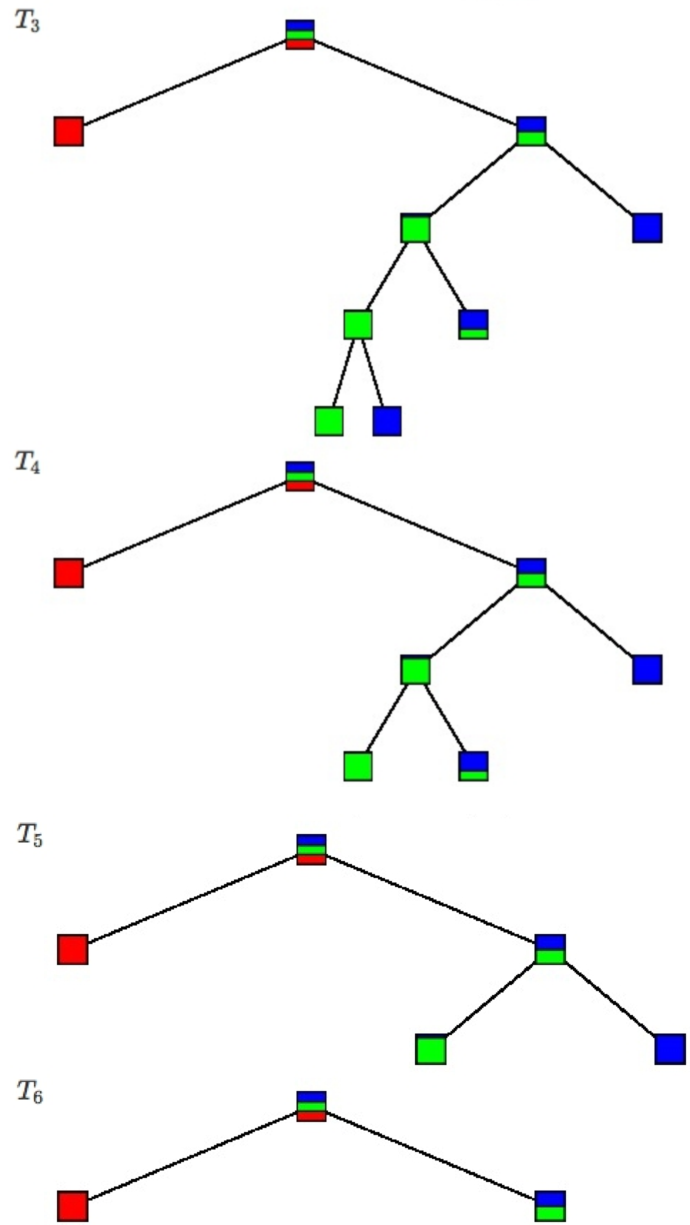
α = penalty factor attached to tree size (set by user)

Among trees of given size, choose the one with lowest CC

Do this for each size of tree

IRIS, CC(T)





$$T_{max} \succeq T_1 \succ T_2 \succ \ldots \succ T_m = \{t_1\}.$$

Using Validation Error to Prune

Pruning process yields a set of trees of different sizes and associated error rates

Two trees of interest:

- Minimum error tree
  - Has lowest error rate on validation data
- Best pruned tree
  - Smallest tree within one std. error of min. error
  - This adds a bonus for simplicity/parsimony

Error rates on pruned trees

# Decision Nodes	% Error Training	% Error Validation	
41	0	2.133333	
40	0.04	2.2	
39	0.08	2.2	
38	0.12	2.2	
37	0.16	2.066667	
36	0.2	2.066667	
35	0.2	2.066667	
14	1.16	1.533333	
13	1.16	1.6	
12	1.2	1.6	
11	1.2	1.466667	<-- Min. Err. Tree
10	1.6	1.666667	
9	2.2	1.666667	
8	2.2	1.866667	
7	2.24	1.866667	
6	2.24	1.6	<-- Best Pruned Tree
5	4.44	1.8	
4	5.08	2.333333	
3	5.24	3.466667	

Postpruning

C4.5 uses a **pessimistic pruning**, similar to the cost complexity but, without a pruning set. It **uses** the **training set** to estimate error rates.

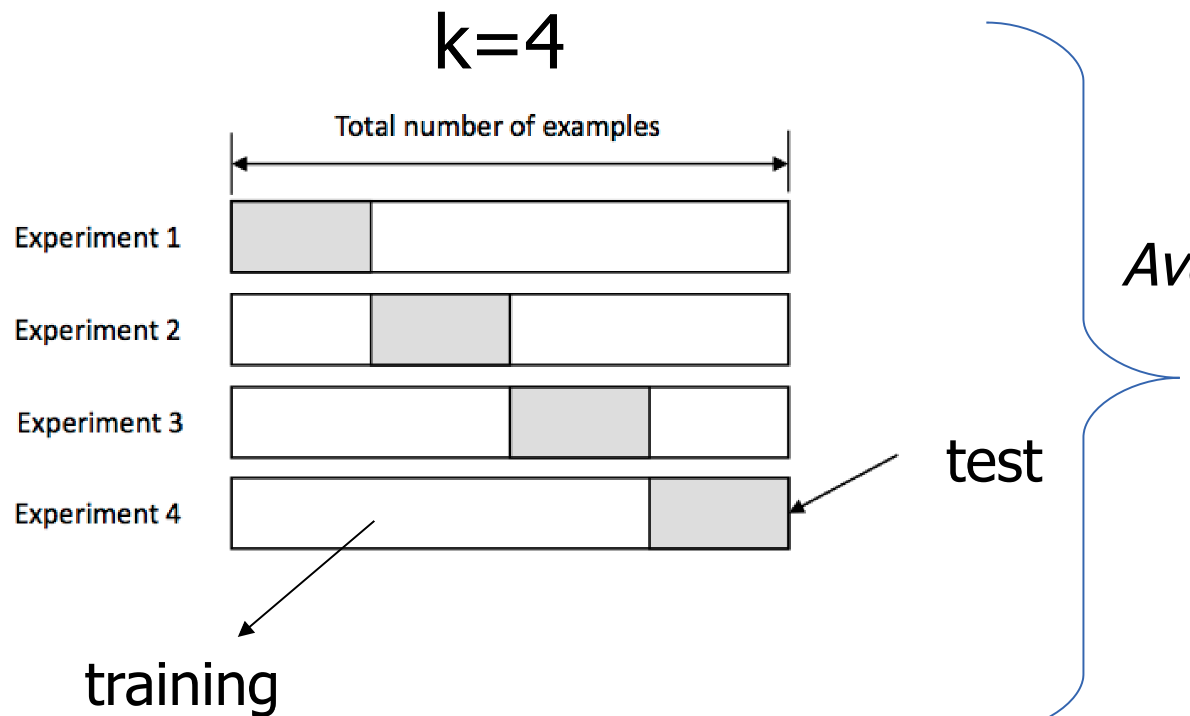
- Estimate accuracy/error based on training data is overly optimistic, thus, **strongly biased**.

Other measures could be used such as the Minimum Description Length (**MDL**). The simple solution is preferred.

Postpruning requires more computation than prepruning, though leads to a more reliable tree.

Postpruning

Avoid overfitting by exploring **k-fold cross validation** and then pruning.



Average Error Rate

$$E = \frac{1}{K}\sum_{i=1}^{K} E_i$$

Decision Tree: *tree* R Package

```
####### Decision trees for classification
####### By Ronnie Alves
library(tree)
attach(iris)
iris[1:4,]
```

```
> iris[1:4,]
  Sepal.Length Sepal.Width Petal.Length Petal.Width Species
1          5.1         3.5          1.4         0.2  setosa
2          4.9         3.0          1.4         0.2  setosa
3          4.7         3.2          1.3         0.2  setosa
4          4.6         3.1          1.5         0.2  setosa
```

```
xtabs(~Species,data=iris)
```

```
> xtabs(~Species,data=iris)
Species
    setosa versicolor  virginica
        50         50         50
```

Decision Tree: *tree* R Package

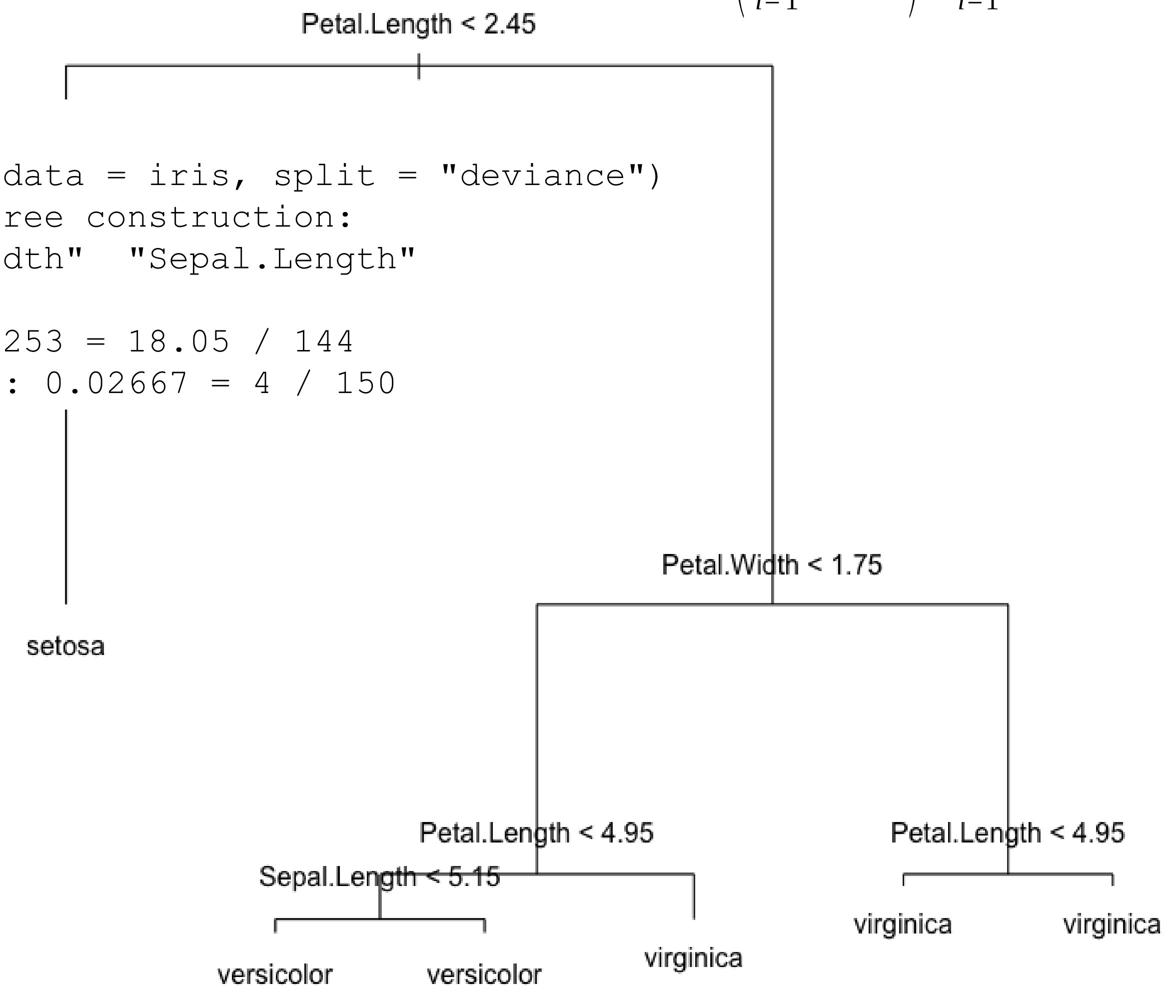
```
dt2 = tree(Species ~ ., iris, split="deviance")
```

$$-2\,LogL = -2\log\left(\prod_{i=1}^{n} f(y_i)\right) = \sum_{i=1}^{n} -2\log f(y_i)$$

```
summary(dt2)
```

```
Classification tree:
tree(formula = Species ~ ., data = iris, split = "deviance")
Variables actually used in tree construction:
[1] "Petal.Length" "Petal.Width"  "Sepal.Length"
Number of terminal nodes:  6
Residual mean deviance:  0.1253 = 18.05 / 144
Misclassification error rate: 0.02667 = 4 / 150
```

```
plot(dt2)

text(dt2)
```

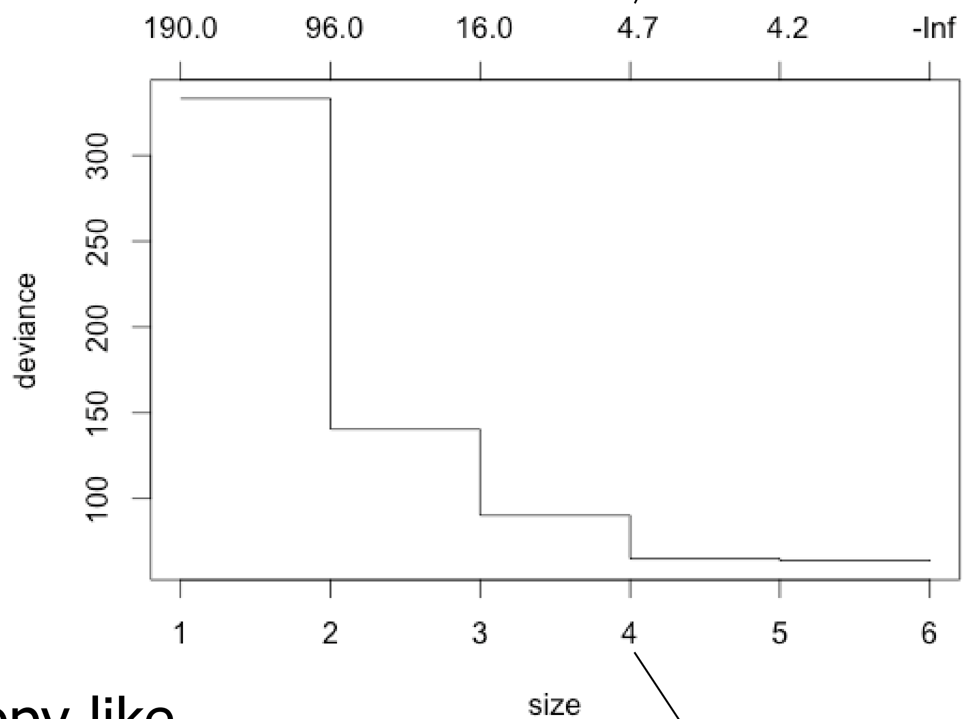


Decision Tree: *tree* R Package

```
ct = cv.tree(dt2, FUN=prune.tree)
```

CC(T)

```
plot(ct)
```



Deviance ~ entropy like

Prune with 4 nodes!

Decision Tree: *tree* R Package

```
pt = prune.tree(dt2, best=4)

pt
```

4 nodes

```
node), split, n, deviance, yval, (yprob)
      * denotes terminal node

 1) root 150 329.600 setosa ( 0.33333 0.33333 0.33333 )
   2) Petal.Length < 2.45 50   0.000 setosa ( 1.00000 0.00000 0.00000 ) *
   3) Petal.Length > 2.45 100 138.600 versicolor ( 0.00000 0.50000 0.50000 )
     6) Petal.Width < 1.75 54  33.320 versicolor ( 0.00000 0.90741 0.09259 )
      12) Petal.Length < 4.95 48   9.721 versicolor ( 0.00000 0.97917 0.02083
      13) Petal.Length > 4.95 6   7.638 virginica ( 0.00000 0.33333 0.66667 )
     7) Petal.Width > 1.75 46   9.635 virginica ( 0.00000 0.02174 0.97826 ) *
```

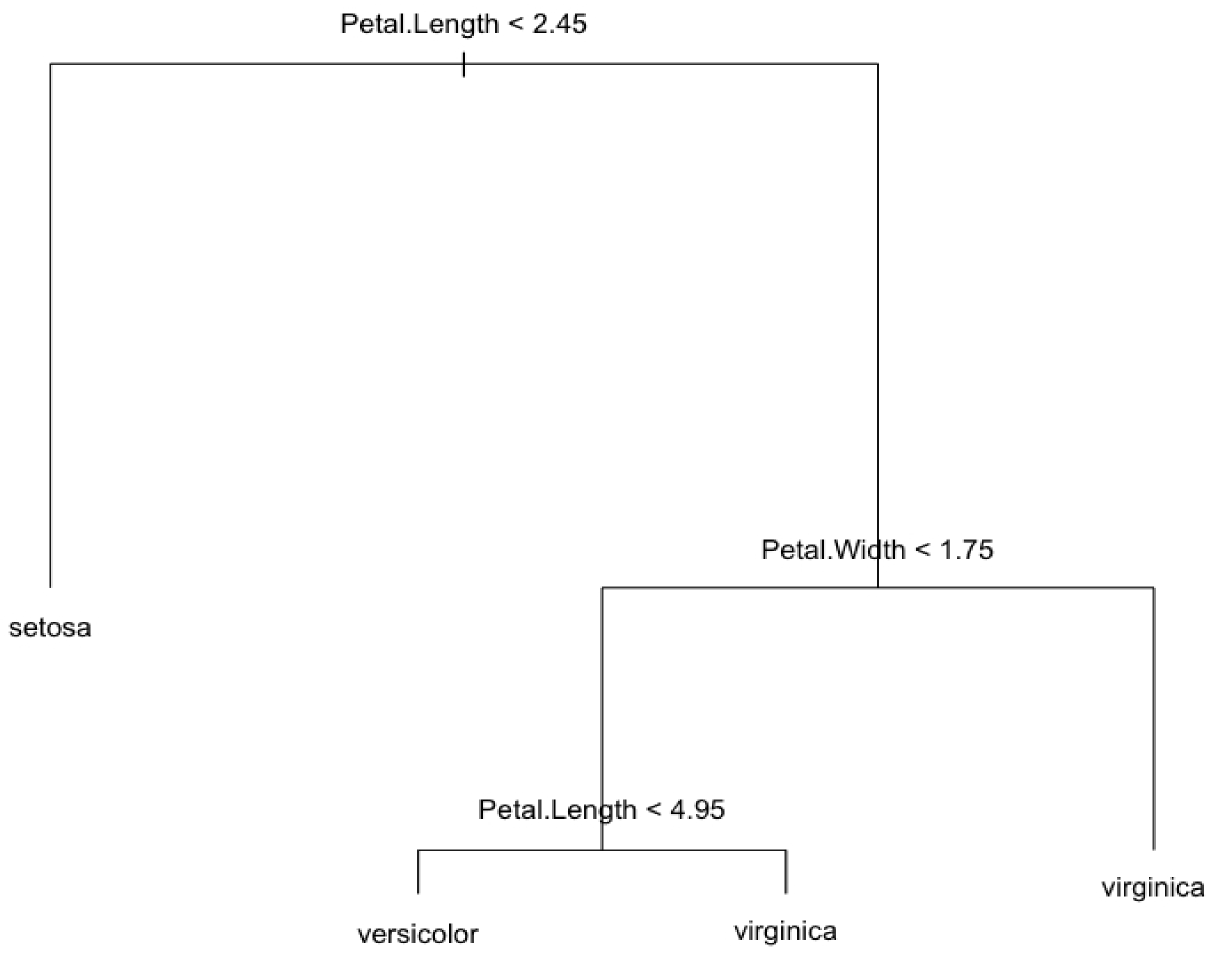
```
dt2
```

6 nodes

```
 1) root 150 329.600 setosa ( 0.33333 0.33333 0.33333 )
   2) Petal.Length < 2.45 50   0.000 setosa ( 1.00000 0.00000 0.00000 ) *
   3) Petal.Length > 2.45 100 138.600 versicolor ( 0.00000 0.50000 0.50000 )
     6) Petal.Width < 1.75 54  33.320 versicolor ( 0.00000 0.90741 0.09259 )
      12) Petal.Length < 4.95 48   9.721 versicolor ( 0.00000 0.97917 0.0208
        24) Sepal.Length < 5.15 5   5.004 versicolor ( 0.00000 0.80000 0.200
        25) Sepal.Length > 5.15 43   0.000 versicolor ( 0.00000 1.00000 0.00
      13) Petal.Length > 4.95 6   7.638 virginica ( 0.00000 0.33333 0.66667
     7) Petal.Width > 1.75 46   9.635 virginica ( 0.00000 0.02174 0.97826 )
      14) Petal.Length < 4.95 6   5.407 virginica ( 0.00000 0.16667 0.83333
      15) Petal.Length > 4.95 40   0.000 virginica ( 0.00000 0.00000 1.0000
```

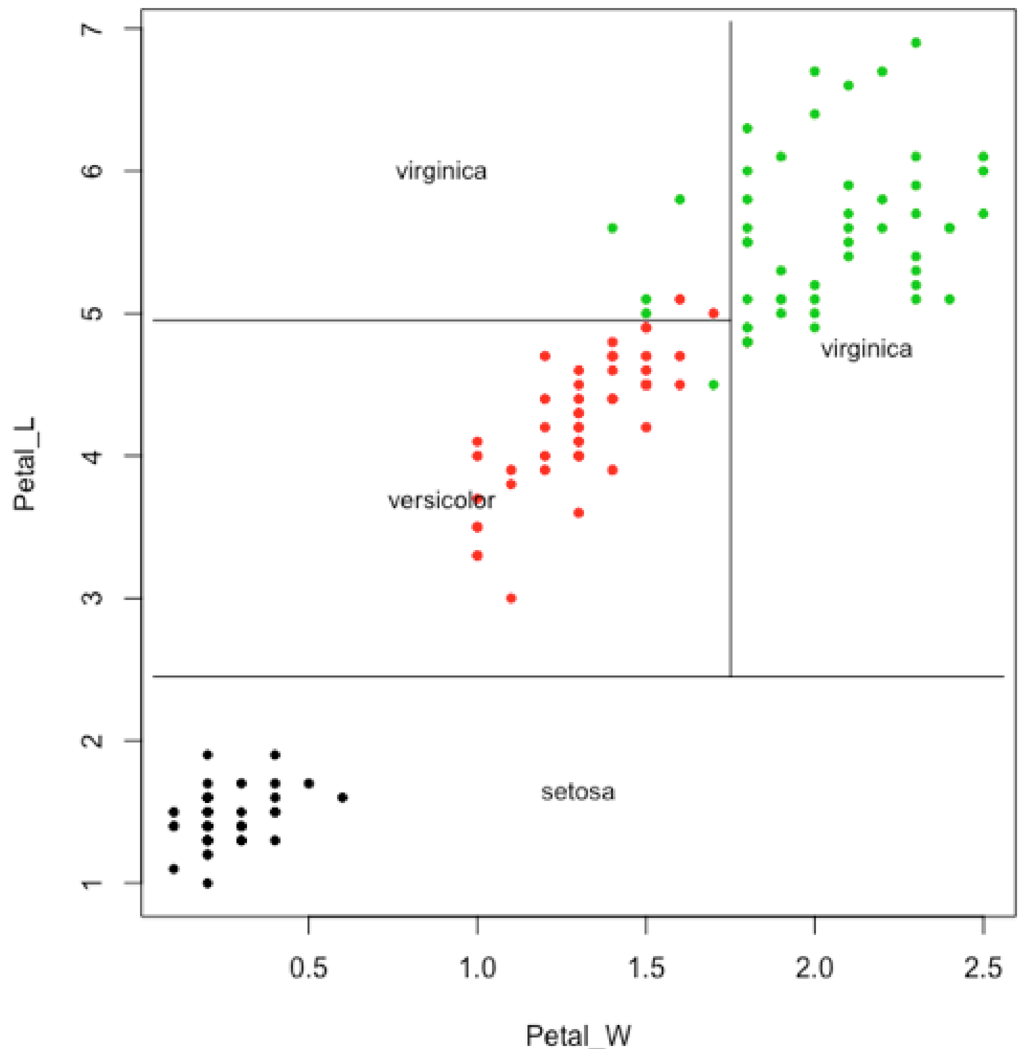
Decision Tree: *tree* R Package

```
pt = prune.tree(dt2, best=4)

plot(pt)

text(pt)
```



Decision Tree: *tree* R Package

```
plot(iris$Petal.Width,
iris$Petal.Length,col=iris$Species,pch=20,
     xlab="Petal_W",ylab="Petal_L")
partition.tree(pt,ordvars =
c("Petal.Width","Petal.Length"),
add = TRUE,cex=0.9)
```



Decision tree: Pruning

Decision trees usually **overfitting** the data.

To avoid overfitting **k-fold cross validation** can be employed to prune not reliable branches.

Postpruning is more effective than prepruning but it requires more computational resources.