東南大學

基于rpart包的决策树应用

王卫杰

2009.12.12

基于rpart包的决策 树应用

- 基本概念
- 几种最常 见的决策树介绍
- 决策 树的建立、修剪及rpart包
- 决策 树的评估
- 决策 树的用途
- 实例1：如何建树、读树
- 实例2：如何修剪树

基本概念

- 决策树是功能强大而且相当受欢迎的分类和预测工具。该方法系supervised learning，以树状图为基础，其输出结果为一系列简单实用的规则，故得名决策树。
- 亦称其为分类树（因变量为分类数据的情况），或者回归树（因变量为连续变量的情况）。

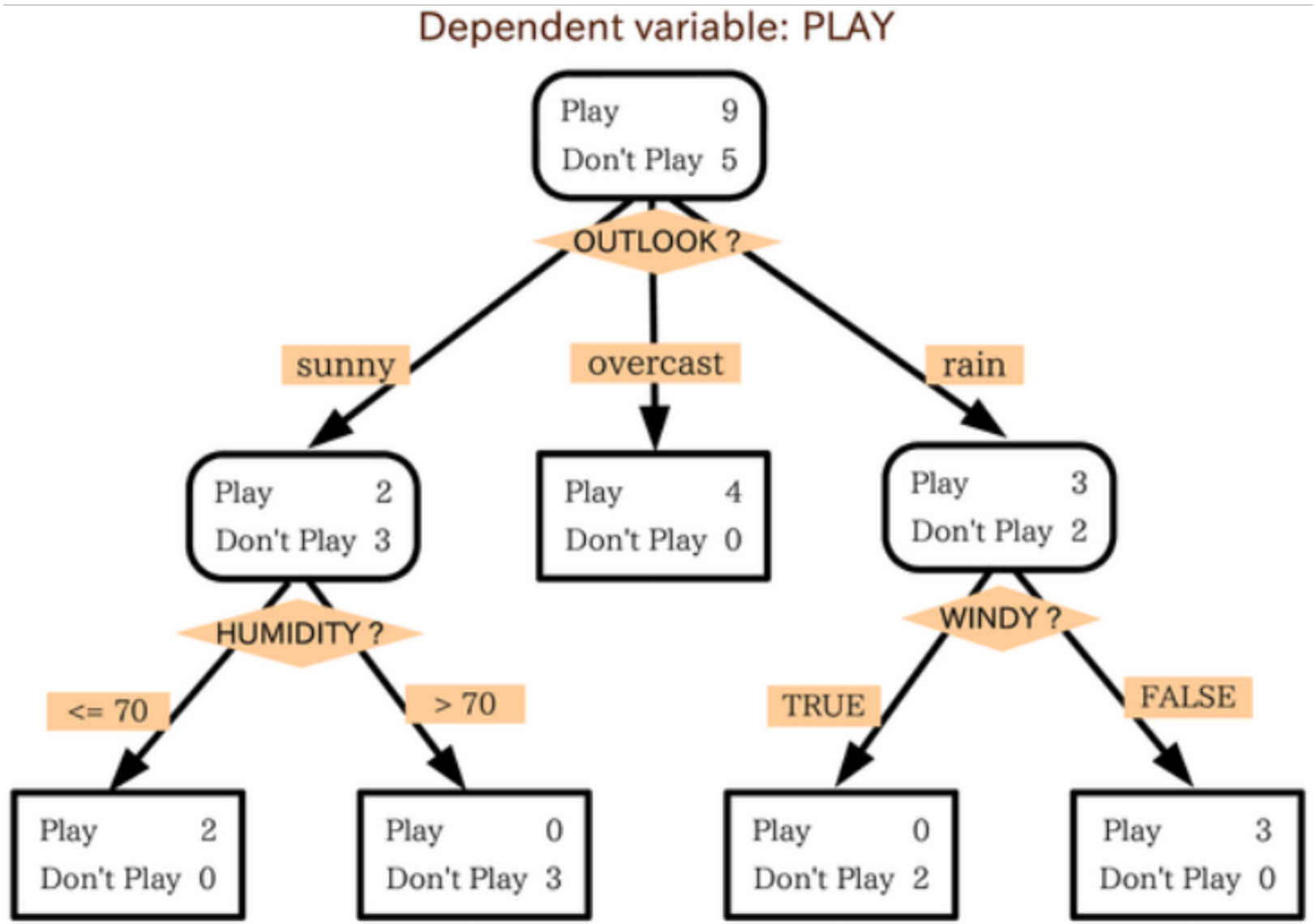
基本概念

- 1984年Breiman等出版的Classification and Regression Tree一书，使得decision tree开始于统计界获得认同
- 1986年Quinlan在Machine Learning Journal发表Induction of decision tree文章，介绍了ID3算法，开启了日后在data mining 领域上的后续研究
- 著名的C4.5, CART, CHAID等算法提出

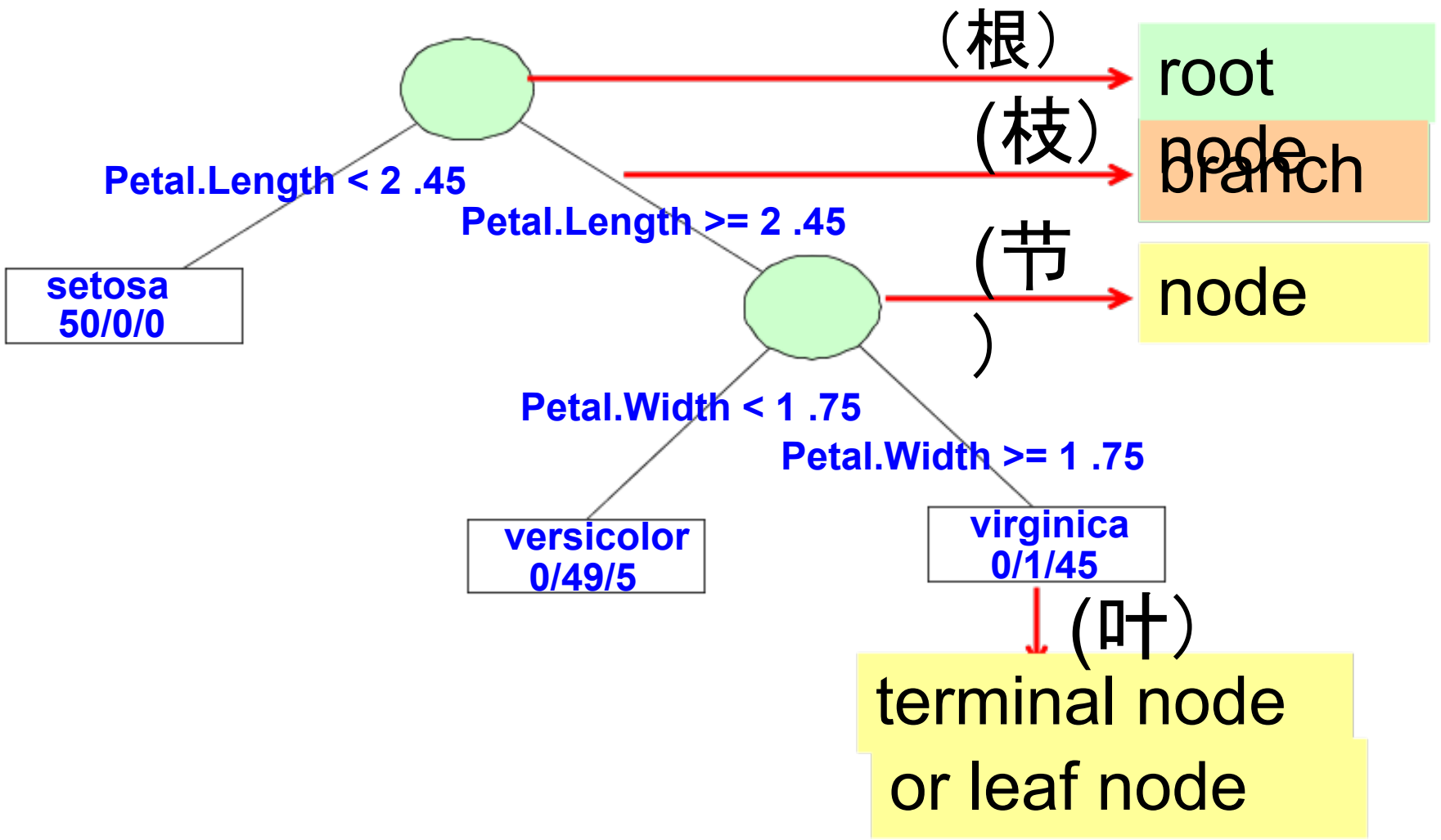
基本概念

Play 9
Don't Play 5
OUTLOOK ?
sunny
overcast
rain
Play 2
Don't Play 3
HUMIDITY ?
Play 4
Don't Play 0
Play 3
Don't Play 2
WINDY ?
<= 70
> 70
Play 2
Don't Play 0
Play 0
Don't Play 3
TRUE
FALSE
Play 0
Don't Play 2
Play 3
Don't Play 0
Dependent variable: PLAY

基本概念


Dependent variable: PLAY
Play 9
Don't Play 5
OUTLOOK ?
sunny
overcast
rain
Play 2
Don't Play 3
Play 4
Don't Play 0
Play 3
Don't Play 2
HUMIDITY ?
WINDY ?
<= 70
> 70
TRUE
FALSE
Play 2
Don't Play 0
Play 0
Don't Play 3
Play 0
Don't Play 2
Play 3
Don't Play 0

基本概念


（根）
root node
(枝)
branch
(节)
node
Petal.Length < 2 .45
Petal.Length >= 2 .45
setosa
50/0/0
Petal.Width < 1 .75
Petal.Width >= 1 .75
versicolor
0/49/5
virginica
0/1/45
(叶)
terminal node
or leaf node

基本概念

以Tree的方式来表 现的predictive model

- 通过把实例（历史数据）从根节点排列到某个叶子节点来分类实例，叶子节点即为实例所属的分类。
- 在每个分支，都有一个分类的问题来将数据做进一步的分类。树上每个节点说明了对实例的某个属性的测试，节点的每个后继分支对应于该属性的一个可能值。
- 从树的顶端即根节点开始分类，愈往下走，子节点内的数据同质性越高。

最常见的决策树

3种最常见的决策树

CART

- Classification and Regression Tree
- **Developed by Breiman, Friedman, Olshen, Stone in 1984.**

C4.5

- **The extension of the basic ID3 algorithm**
- **Developed by Quinlan in 1993**
- **Last research version: C4.8, implemented in Weka as J4.8 (Java)**
- **Commercial successor: C5.0 (available from Rulequest)**

CHAID

- **Chi-squared Automatic Interaction Detection**
- **Developed by Kass in 1980**

最常见的决策树-CART

- Classification and Regression Tree 分类与回归树
- Developed by Breiman, Friedman, Olshen, Stone in 1984.
- 以每个节点的动态临界值作为条件判断式子
- 每个节点采用二分法，即每个节点只能有2个子节点；这也是CART与C4.5的最大区别。C4.5可以在每个节点上产生不同数量的分支。
- 用Gini Ratio作衡量指标。如果分散的指标程度很高，说明数据中有很多类别；相反，如果指标程度低，说明单一类别的成员居多。

最常见的决策树- C4.5

- ID3 algorithm的改良版
- 最新 C4.8, implemented in Weka as J4.8 (Java)
- 商业版 C5.0 (available from Rulequest)
- 先建完整的决策树，再使用自定义的错误率（Predicted Error Rate)对每个内部节点进行修剪
- 不同的节点，特征值离散化结果不同

最常见的决策树-

CHAID

- Chi-squared Automatic Interaction Detection
- Developed by Kass in 1980
- 利用卡方检验预测两个变数是否需要合并。能够产生最大类别差异的预测变数，将成为节点的分割变数。
- 计算节点中类别的p值。根据p值的大小决定决策树是否生长。因而不需要像CART、C4.5那样再进行修剪。

最常见的决策树

3种最常见的决策树

分类	参数类型	分类规则	修剪规则
C4.5	分类参数	信息量	Node error rate
CHAID	分类参数	卡方分配	无
CART	连续参数 分类参数	Gini ratio	Entire error rate

最常见的决策树

Top 10 algorithms in data mining

C4.5、K-Means、SVM、Apriori、EM、PageRank

AdaBoost、kNN、Naïve Bayes、 **CART**

Xindong Wu, Vipin Kumar, et al. (2008). Top 10 algorithms in data mining, Knowledge Information System 14: 1-37

IEEE International Conference on Data Mining in December 2006

决策树的建立

- 建立决策 树，利用训练样本生成决策树。
  - 开始数据都在根节点
  - 递归的进行数据分片
- 修剪决策 树

控制误差和树的规模

- 使用决策 树对未知数据进行分类

  按照决策树上采用的分割属性逐层往下，直到一个叶子节点

决策树的建立

1. rpart package

递归分割与回归树

(**Recursive partitioning** and regression trees)

rpart.pdf

>library(rpart)

>?rpart

决策树的建立

2. 决策树的建立 rpart( )函数

rpart(formula, data, weights, subset, na.action = na.rpart, method, model = FALSE, x = FALSE, y = TRUE, parms, control, cost, ...)

- **formula** 回归方程lm( )形式：y~x1+x2+x3+x4
- **data** 包含前面公式的数据框
- **na.action** 缺失数据的处理办法。默认办法是删除因变量缺失的观测值，保留自变量缺失的观测值。
- **method** 根据树末端因变量的数据类型选择分割方法：

anova(连续型)、poisson（计数型）、class（离散型）、exp（生存型）.

- **parms** 设置3个参数：先验概率、损失矩阵、分类纯度
- **control** 控制每个节点上的最小样本量、交叉验证的次数、复杂性参数

决策树的修剪

3.修剪函数 prune ( )函数

prune(tree, ...)

- tree: 决策树对象。
- cp: complexity parameter，复杂性参数，

用来设定修剪用的参数，一般遵循1-SE规则。

决策树的建立

- 事前修剪(Pre-Pruning)
  - 使用统计阀值加以衡量，如卡方值或信息获取值等，评价是否该继续分割某内部节点或立刻停止
- 事后修剪(Post-Pruning)
  - 建树的过程中允许树枝繁叶茂，当完成建树后，进行修剪

决策树的衡量

- 利用training data建立决策树，将其应用在test data上，观察分类的正确率，以此衡量决策树的有效程度。
- 对决策树节点的衡量：
  - 进入节点的数据数目。
  - 如果是叶部节点，可观察数据分类的方式。
  - 该节点将数据正确分类的比率。

决策树的用途

- Data Exploration
  - 根据decision tree所选的predictors及split的值，探索数据，了解数据特性。
  - 使用者可对其结果进行人工确认，可依个人的专业知识修改decision tree，可将其改成decision rule。
- Data Preprocessing
  - decision tree可处理不同类型(numeric, categorical)的数据，且执行速度快。
  - 可用来判断一些predictors，将predicators或decision rule提供给其他data mining技术(如neural network、nearest-neighbor 、统计方法)使用，以缩短data mining技术的运行时间。

决策树的用途

- 用作clusters。
  - 但由于属于supervised learning，其clustering的目的由某个所要预测的目标引导。
- 用作links。
  - 可用作找出predictors间的links，但其执行过程中不会为了找出这些links来作最佳化。
- 用于找出Outliers
  - Outliers通常会有和其他资料很不同的predictors或落在某些预测范围的leaf node 。

决策树的用途

- 极适合用于产生Rules
  - 将decision trees转成简单使用的rules。
- 可用于sequences
  - Decision trees可用在time series prediction.
- 可用于text
  - Decision trees已被用于text classification与information retrieval来处理当文字的dimension不太大的情况。

决策树的用途

聚类分析、判别分析、决策树的差异

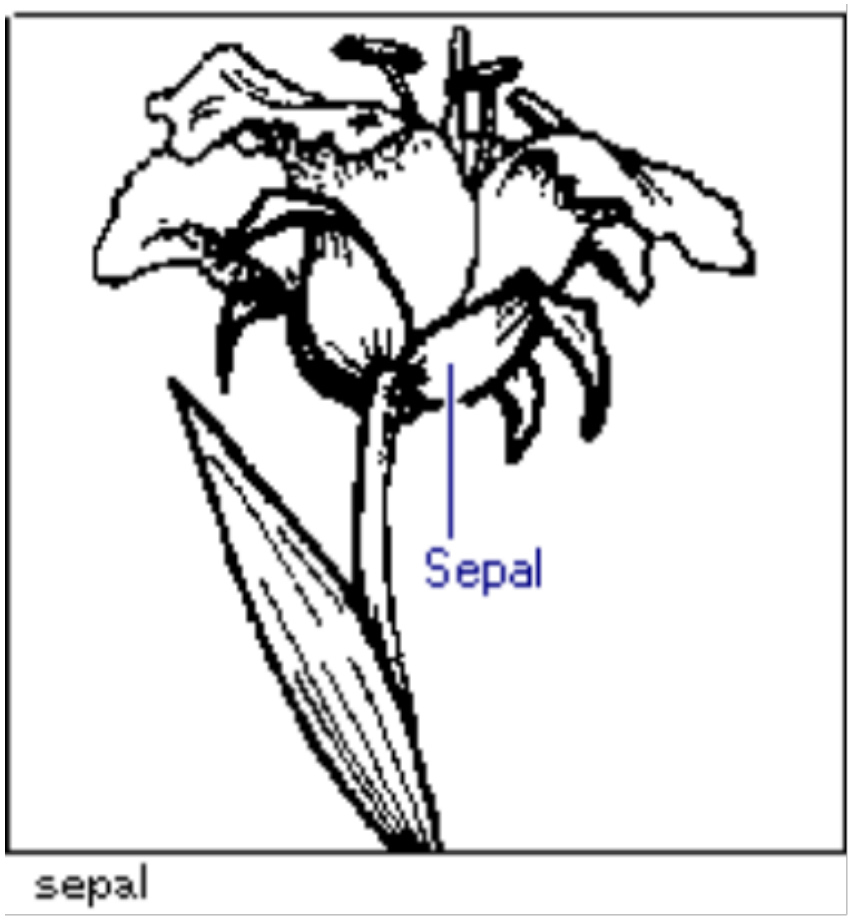
- 聚类分析：不需数据类别，可以将数据分类，但不能提供数据分类的法则；
- 判别分析：需要数据类别，可将数据分类，但不能提供数据分类的法则；
- 分类树：需要数据类别，既可将数据分类，又可输出分类法则（决策规则）。

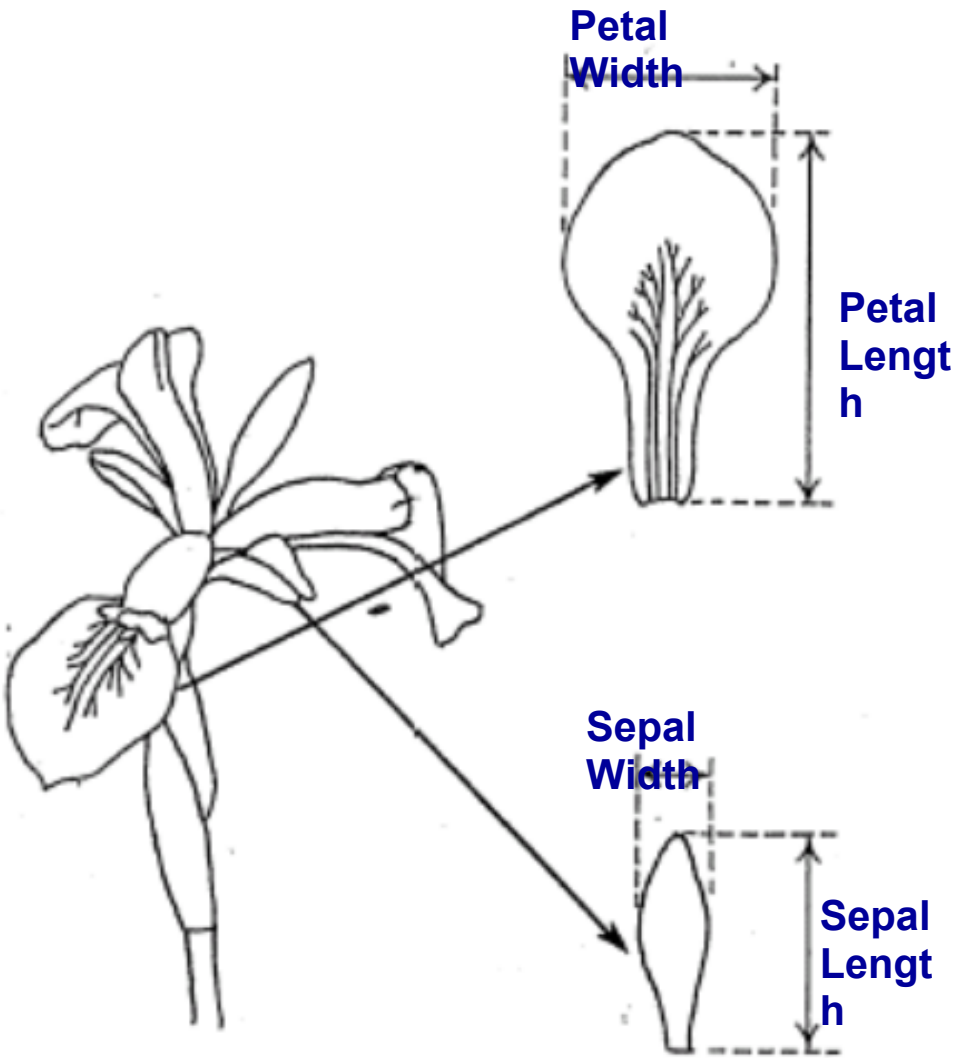
实例 1 - How to construct the tree

The data iris in rpart package

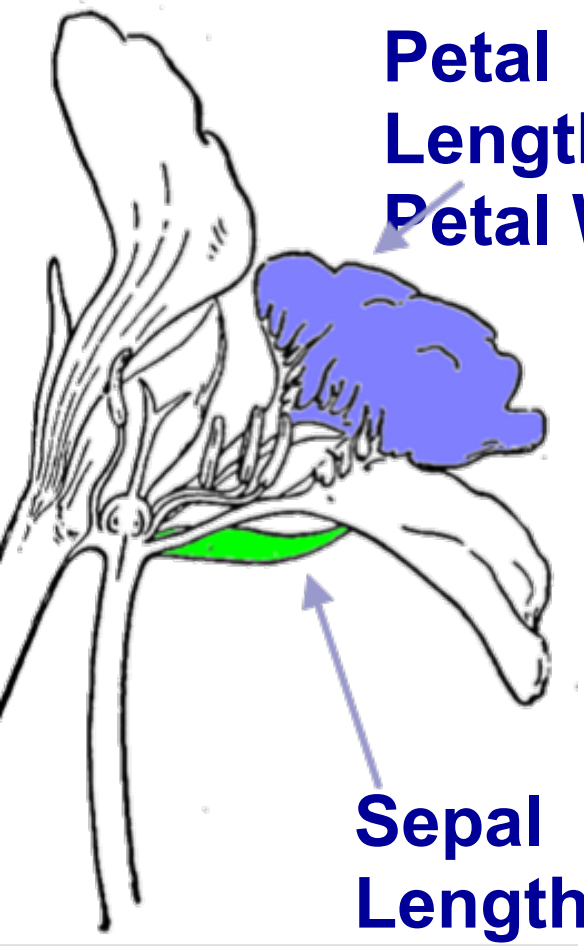
- Iris dataset relates species to **petal and sepal** dimensions reported in centimeters.
- Originally used by R.A. Fisher and E. Anderson for a **discriminant analysis** example.
- Data is pre-packaged in R dataset library.

实例 1


Sepal
sepal


Petal
Width
Petal
Lengt
h
Sepal
Width
Sepal
Lengt
h

实例 1 – What species are these flowers?



Petal Length
Petal Width

Sepal Length
Sepal Width

tree

Setosa

Versicolor

Virginica

实例 1

```
>library(rpart)
>data(iris) # this command is not necessary.
>head(iris)
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
5 5.0 3.6 1.4 0.2 setosa
6 5.4 3.9 1.7 0.4 setosa
```

```
> names(iris)
[1] "Sepal.Length" "Sepal.Width" "Petal.Length" "Petal.Width" "Species"
```

```
> nrow(iris)
[1] 150
```

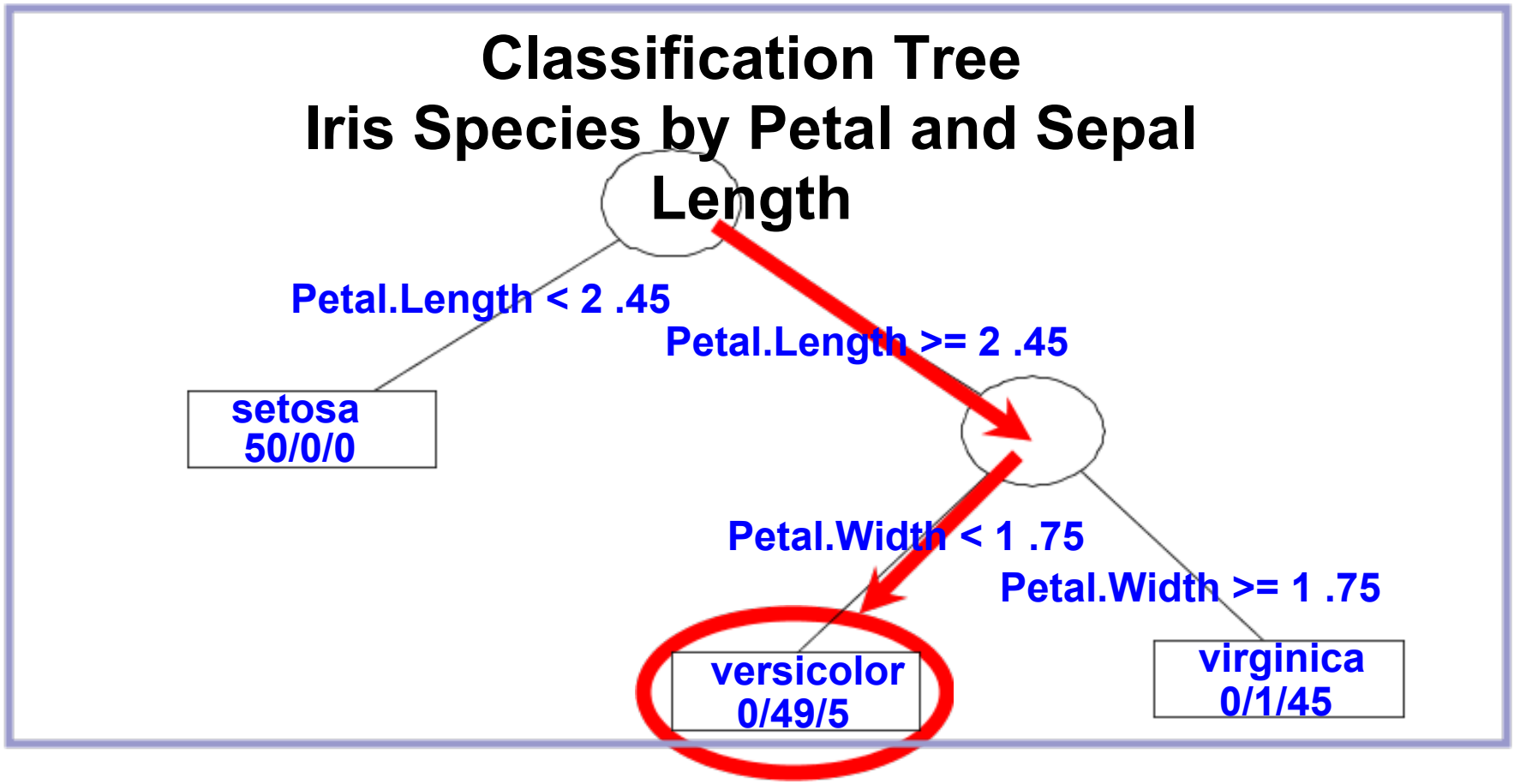
实例 1

```
# Let iris.rp = tree object fitting Species vs. all other
iris.rp = rpart(Species~., iris, method="class")

# Plot tree diagram with uniform spacing,
# diagonal branches, a 10% margin, and a title
plot(iris.rp, uniform=T, branch=0, margin=0.1, main="
Classification Tree\nIris Species by Petal and Sepal
Length")

# Add labels to tree with final counts,
# fancy shapes, and blue text color
text(iris.rp, use.n=T, fancy=T, col="blue")
```

实例 1



Rule 1: if Petal.Length>=2.45&Petal.Width<1.75, then it is versicolor (0/49/5)

Rule2: if Petal.Length>=2.45&Petal.Width>=1.75, then it is virginica (0/1/45)

Rule 3: if Petal.Length<2.45, then it is setosa (50/0/0)

实例 1

```
> table(iris$Species)
setosa versicolor virginica
50 50 50
```

setosa(50/0/0): 50朵setosa被正确辨认
versicolor (0/49/5): 49朵versicolor正确辨认，
5朵virginica被错辨认为versicolor
virginica (0/1/45): 45朵virginica正确辨认，
1朵versicolor被错辨认为virginica

实例 1

Predicting

```
> even.n=2*(1:75)-1 #生成奇数
> iris.train=iris[even.n,] #生成training data（序号为奇数）
> iris.test=iris[-even.n,] #生成test data
> iris.rp2=rpart(Species~., iris.train, method="class")
```

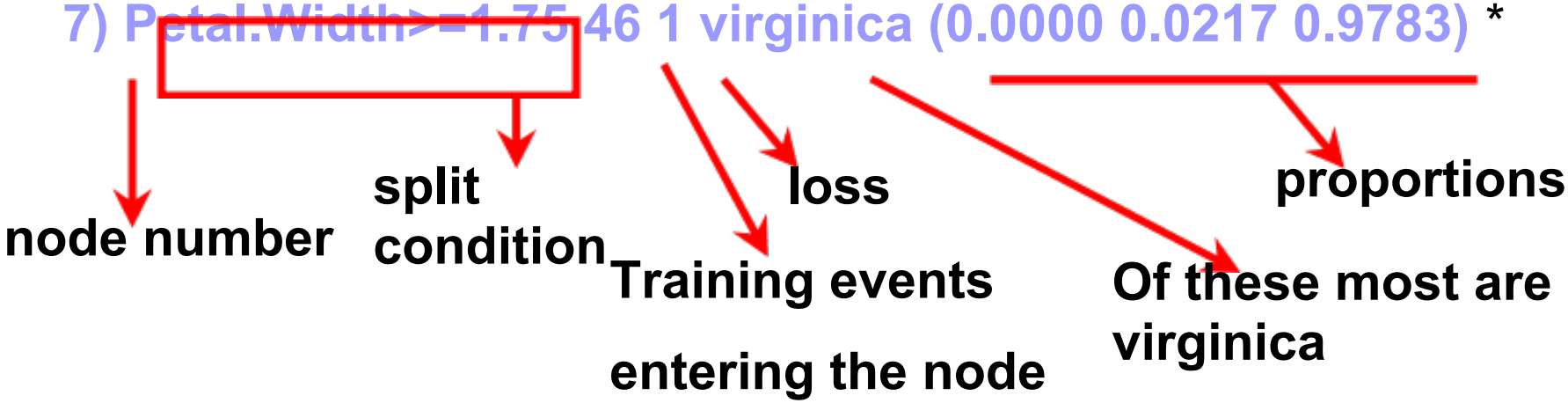
```
> iris.rp3=predict(iris.rp2, iris.test[,-5], type="class")
> table(iris.test[,5],iris.rp3)
```

```
iris.rp3
setosa versicolor virginica
setosa 25 0 0
versicolor 0 24 1
virginica 0 3 22
```

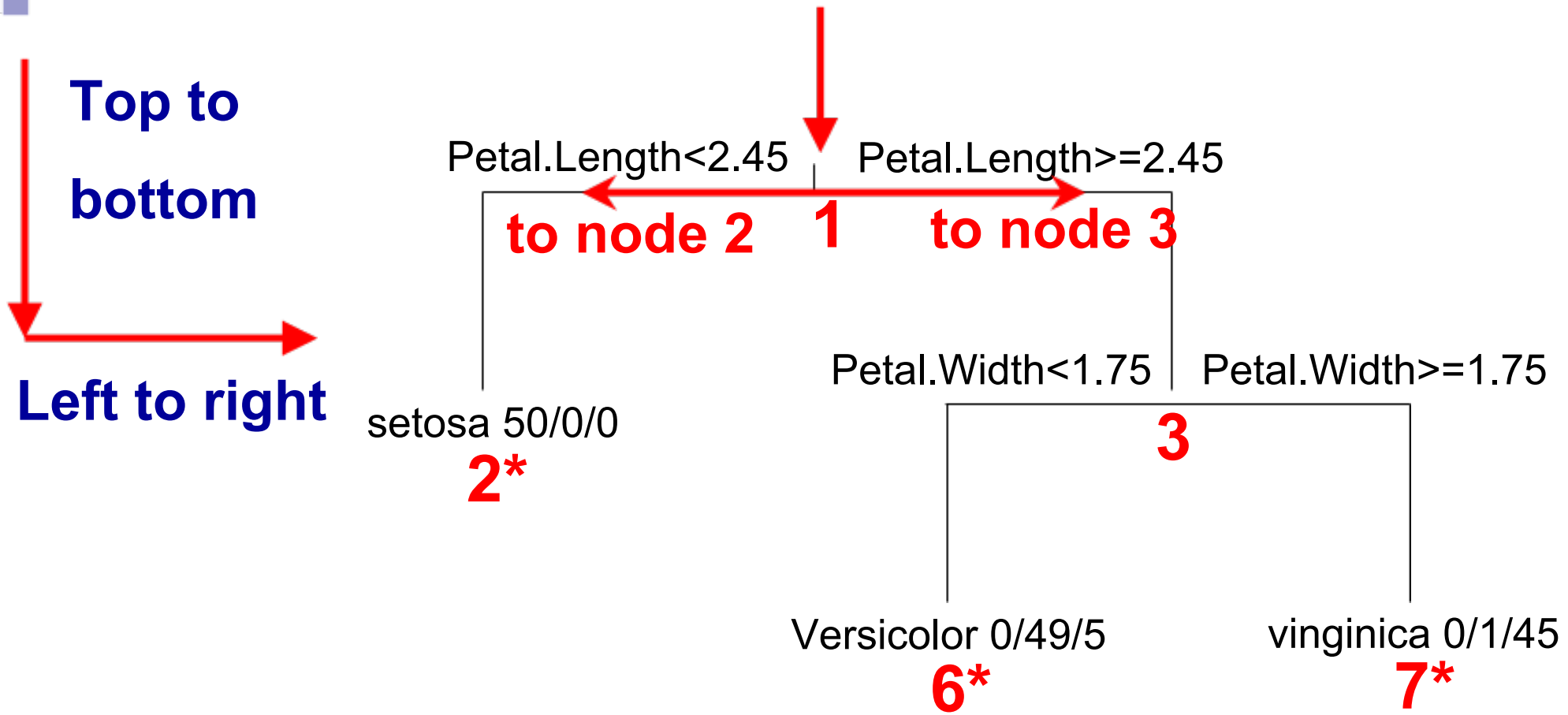
实例 1

```
> print(iris.rp,digit=3)
n= 150

node), split, n, loss, yval, (yprob)
* denotes terminal node
1) root 150 100 setosa (0.3333 0.3333 0.3333)
2) Petal.Length< 2.45 50 0 setosa (1.0000 0.0000 0.0000) *
3) Petal.Length>=2.45 100 50 versicolor (0.0000 0.5000 0.5000)
6) Petal.Width< 1.75 54 5 versicolor (0.0000 0.9074 0.0926) *
7) Petal.Width>=1.75 46 1 virginica (0.0000 0.0217 0.9783) *
```



实例 1


Top to bottom
Left to right
Petal.Length<2.45
Petal.Length>=2.45
to node 2
1
to node 3
setosa 50/0/0
2*
Petal.Width<1.75
Petal.Width>=1.75
3
Versicolor 0/49/5
6*
vinginica 0/1/45
7*

实例 1

rpart object

```
> attributes(iris.rp)
$names
[1] "frame" "where" "call" "terms" "cptable" "splits" "method"
"parms" "control" "functions“ "y" "ordered"

$ylevels
[1] "setosa" "versicolor" "virginica"

$class
[1] "rpart"
```

实例 1

> **print(iris.rp$cptable, digits=2)**

CP nsplit rel error xerror xstd
1 0.50 0 1.00 1.19 0.050
2 0.44 1 0.50 0.64 0.061
3 0.01 2 0.06 0.09 0.029

- **CP**: complexity parameter
- **nsplit**: the number of splits
- **relerror**: the relative error rate for predic
  for the data that generated the
- **xerror**: cross-validated error. xerror is
  an abbreviation for cross-validated
- **xstd**: the standard derivation of cross-validated

> **printcp(iris.rp,digits=2)**

Classification tree:
rpart(formula = Species ~ ., data = iris, method = "class")
Variables actually used in tree construction:
[1] Petal.Length Petal.Width
Root node error: 100/150 = 0.67
n= 150
CP nsplit rel error xerror xstd
1 0.50 0 1.00 1.19 0.050
2 0.44 1 0.50 0.64 0.061
3 0.01 2 0.06 0.09 0.029

实例 1

- **CP**: complexity parameter
- **nsplit**: the number of splits
- **rel error**: the relative error rate for predictions

for the data that generated the tree

- **xerror**: cross-validated error.x in xerror is

an abbreviation for cross-validated

- **xstd**: the standard derivation of cross-validated

实例2 - How to prune?

The data cpus in package MASS

System Speed (mhz) | Memory (kb) | Cache (kb) | Channels | Performance Benchmark

name syct mmin mmax cach chmin chmax perf estperf
1 ADVISOR 32/60 125 256 6000 256 16 128 198 199
2 AMDAHL 470V/7 29 8000 32000 32 8 32 269 253
3 AMDAHL 470/7A 29 8000 32000 32 8 32 220 253
4 AMDAHL 470V/7B 29 8000 32000 32 8 32 172 253
5 AMDAHL 470V/7C 29 8000 16000 32 8 16 132 132
6 AMDAHL 470V/8 26 8000 32000 64 8 32 318 290

实例 2

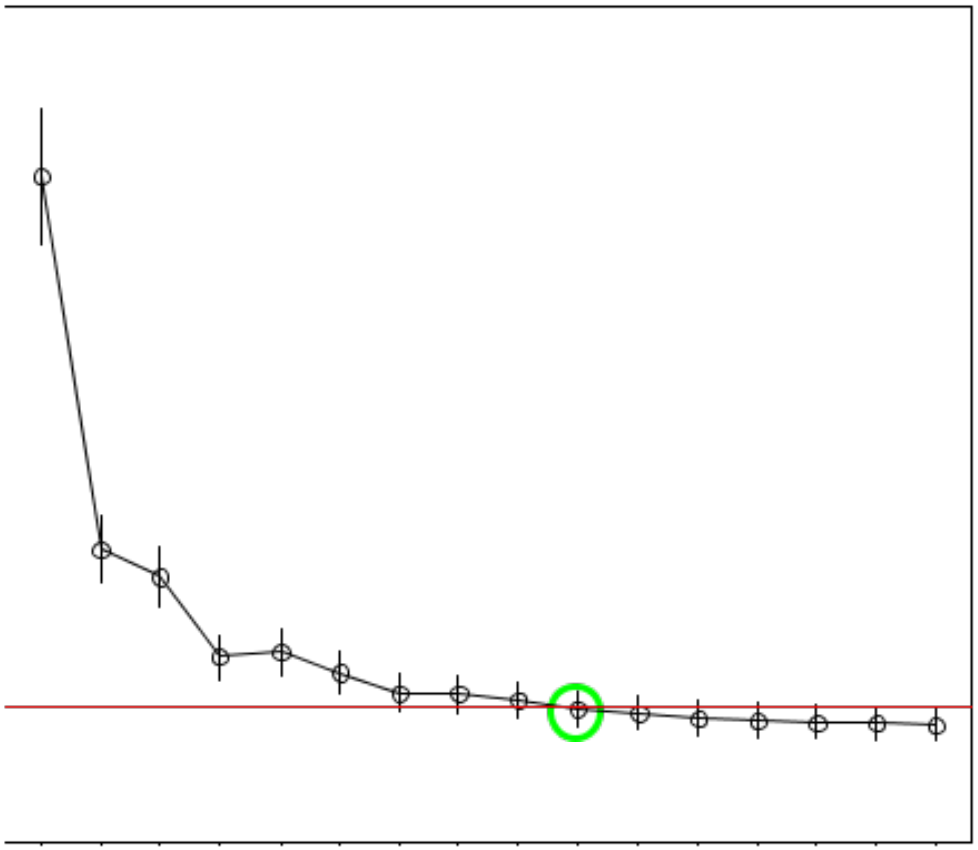
```
> library(MASS)
> library(rpart)
> data(cpus)
> attach(cpus)
# construct the tree, complexity parameter is set as 0.001
> cpus.rp=rpart(log(perf)~.,cpus[,2:8],cp=0.001,method="anova")

# print and plot the complex parameter
> printcp(cpus.rp,digit=2)
> plotcp(cpus.rp,lty=1,col=2)

# plot the tree
> plot(cpus.rp,uniform=T,margin=0.1,main="Test of regression tree for cpus")
> text(cpus.rp,digits=2)
```

实例 2

CP	nsplit	rel error	xerror	xstd
1	0.5493	0	1.00	1.01	0.097
2	0.0893	1	0.45	0.47	0.048
3	0.0876	2	0.36	0.44	0.043
4	0.0328	3	0.27	0.32	0.031
5	0.0269	4	0.24	0.33	0.032
6	0.0186	5	0.21	0.30	0.030
7	0.0168	6	0.20	0.27	0.027
8	0.0158	7	0.18	0.27	0.027
9	0.0095	9	0.15	0.26	0.026
10	**0.0055**	**10**	**0.14**	**0.25**	**0.025**
11	0.0052	11	0.13	0.24	0.025
12	0.0044	12	0.13	0.23	0.025
13	0.0023	13	0.12	0.23	0.025
14	0.0023	14	0.12	0.23	0.025
15	0.0014	15	0.12	0.22	0.025
16	0.0010	16	0.12	0.22	0.025

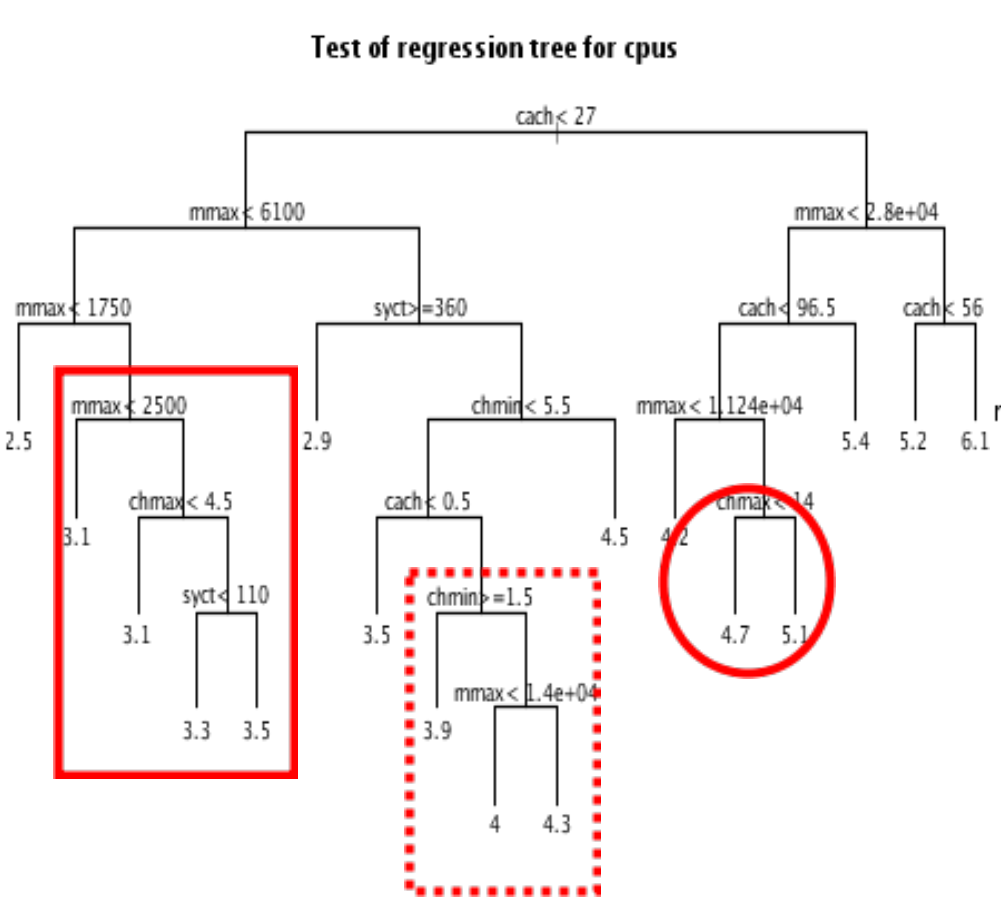


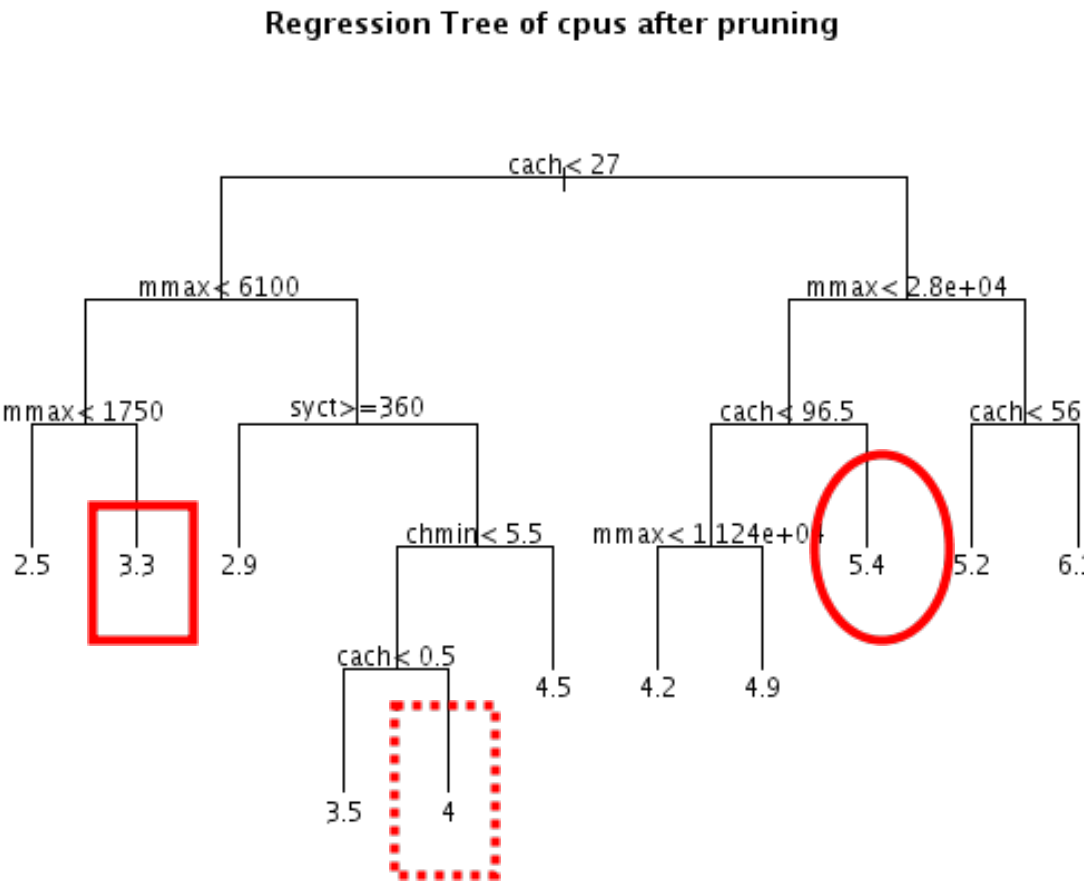
实例 2

```
# prune  the tree
# the complexity parameter is set as 0.0055
> cpus.rp2=prune(cpus.rp, cp=0.0055)

# plot the new tree
> plot(cpus.rp2,uniform=T,margin=0.1,main="
Regression Tree of cpus after pruning")
> text(cpus.rp2,digits=2)
```

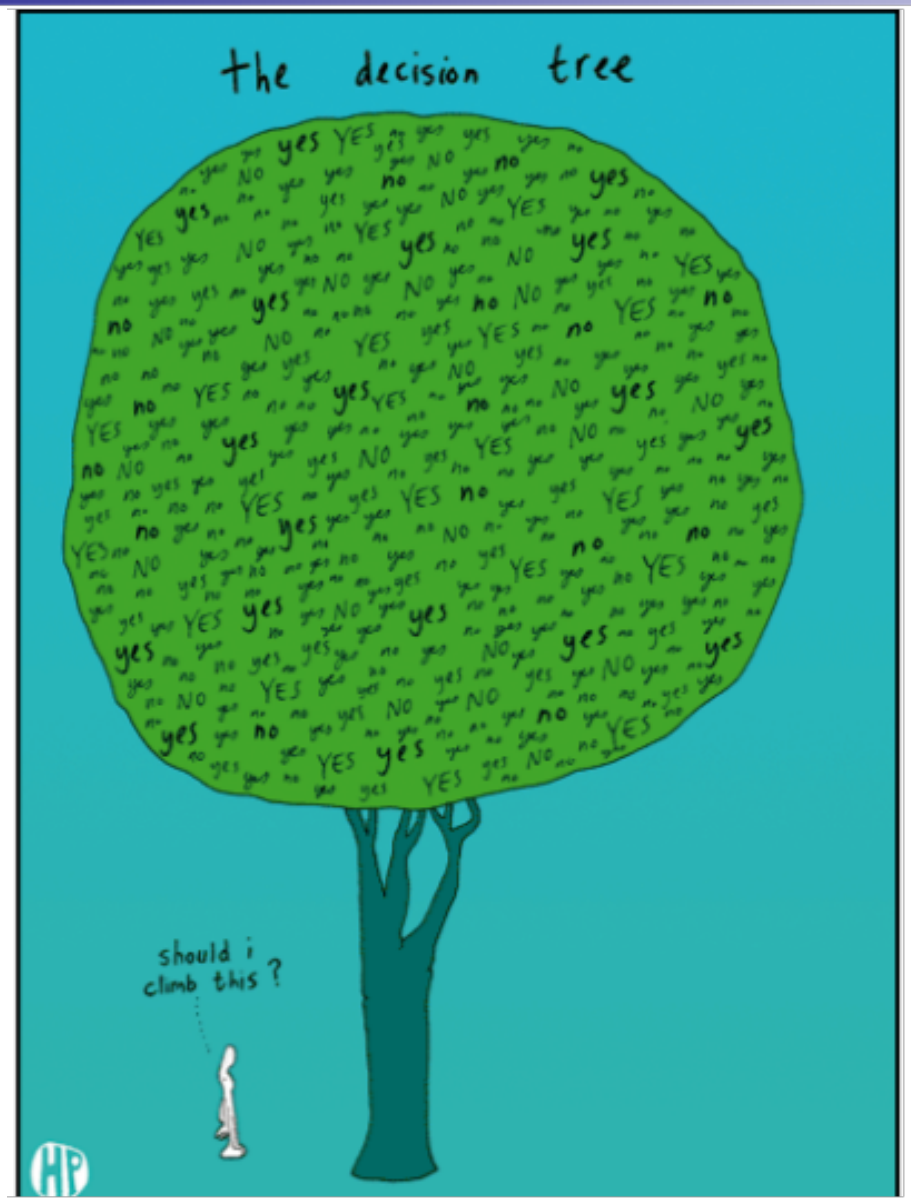
实例 2





Before pruning

After pruning



東南大學