数据挖掘导论
Introduction to Data Mining

第三章：分类问题a

王浩
Email：haowang@szu.edu.cn

第一部分目标：

- 认识分类问题
- 分类模型建立与评价
- 常见分类方法
- 认识分类树
- 分类树构建
- 分类树评价

分类问题——定义：

- 给定一组数据（训练集），如右图
- 每一条记录都可由一组(x, y) 来表示, x 代表属性集，y 代表类别标记

  x: attribute, predictor, independent variable, input

  y: class, response, dependent variable, output
- 任务:学习一个模型，利用每一条记录的属性集x 去预测它对应的类别y。

ID	Home Owner	Marital Status	Annual Income	Defaulted Borrower
1	Yes	Single	125K	No
2	No	Married	100K	No
3	No	Single	70K	No
4	Yes	Married	120K	No
5	No	Divorced	95K	Yes
6	No	Married	60K	No
7	Yes	Divorced	220K	No
8	No	Single	85K	Yes
9	No	Married	75K	No
10	No	Single	90K	Yes

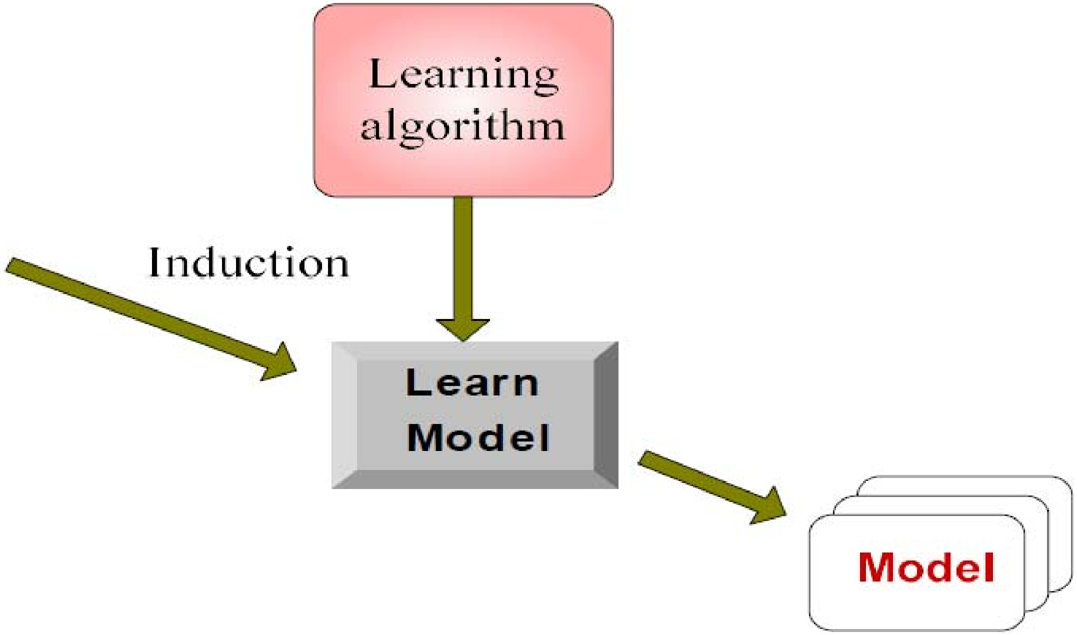
分类问题——举例：

任务	特征集合, x	类别标签, y
评价用户的信用	APP平台采集到的相关数据	信用良好或是信誉不良
顾客流失率预测	平台采集到的顾客画像数据	流失或是不流失
邮件分类	从邮件主题和内容中提取特征	普通邮件、垃圾邮件
肿瘤细胞识别	从x射线或核磁共振扫描数据中提取的特征	恶性或良性细胞

模型建立流程：

Tid	Attrib1	Attrib2	Attrib3	Class
1	Yes	Large	125K	No
2	No	Medium	100K	No
3	No	Small	70K	No
4	Yes	Medium	120K	No
5	No	Large	95K	Yes
6	No	Medium	60K	No
7	Yes	Large	220K	No
8	No	Small	85K	Yes
9	No	Medium	75K	No
10	No	Small	90K	Yes

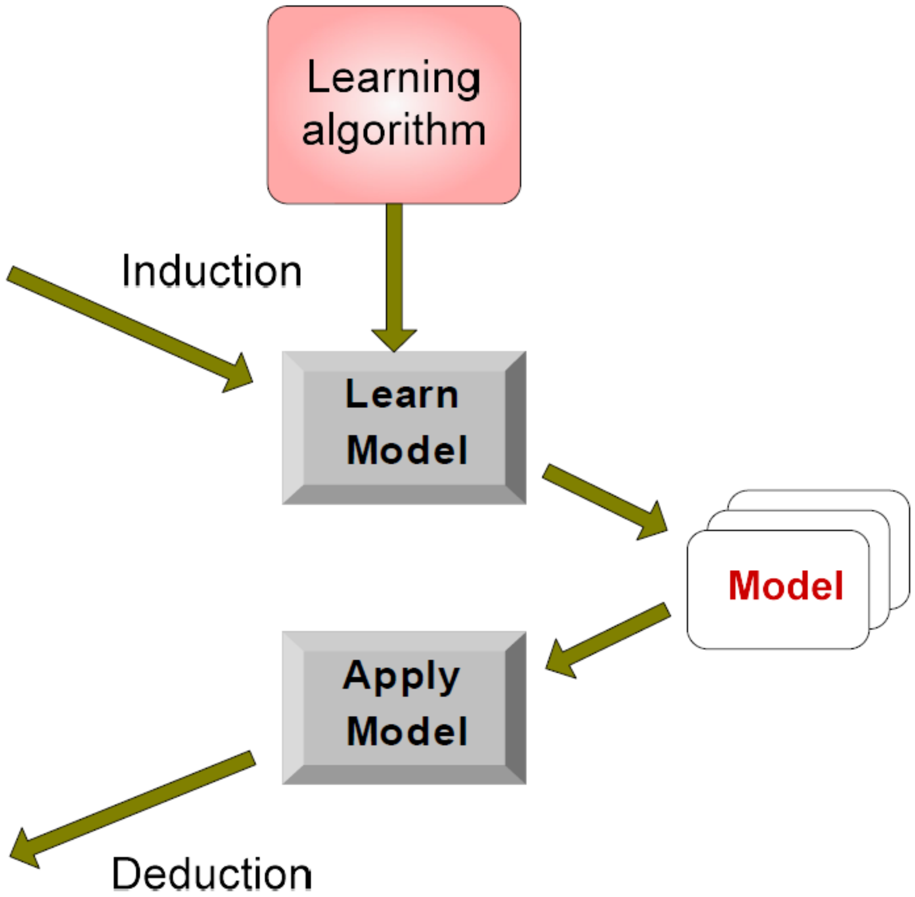
Training Set



模型建立流程：

Tid	Attrib1	Attrib2	Attrib3	Class
1	Yes	Large	125K	No
2	No	Medium	100K	No
3	No	Small	70K	No
4	Yes	Medium	120K	No
5	No	Large	95K	Yes
6	No	Medium	60K	No
7	Yes	Large	220K	No
8	No	Small	85K	Yes
9	No	Medium	75K	No
10	No	Small	90K	Yes

Training Set



Tid	Attrib1	Attrib2	Attrib3	Class
11	No	Small	55K	?
12	Yes	Medium	80K	?
13	Yes	Large	110K	?
14	No	Small	95K	?
15	No	Large	67K	?

Test Set

模型评价：

真正例（TP）：将正类预测为正类

假负类（FN）：将正类预测为负类

假正类（FP）：将负类预测为正类

真负类（TN）：将负类预测为负类

精确率 (Precision） = TP / (TP+FP)　　除以预测的正类

召回率（Recall）　= TP / (TP+FN)　　除以真实的正类

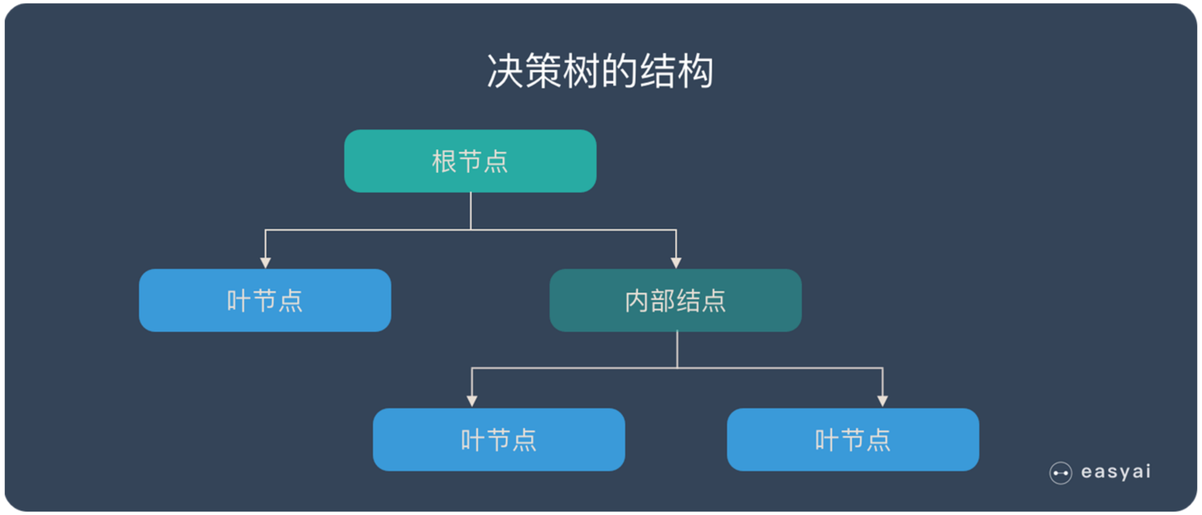
Predicted

Actual	Cat	Dog	Pig
Cat	40	20	10
Dog	35	85	40
Pig	0	10	20

计算如图所示的三分类混淆矩阵，计算各类别预测结果的精确率与召回率。

分类方法:

- 基本分类
  - 决策树
  - 规则方法
  - 最近邻方法
  - 神经网络
  - 贝叶斯方法
  - 支持向量机(SVM)
  - …
- 集成分类
  - Boosting, Bagging, 随机森林…

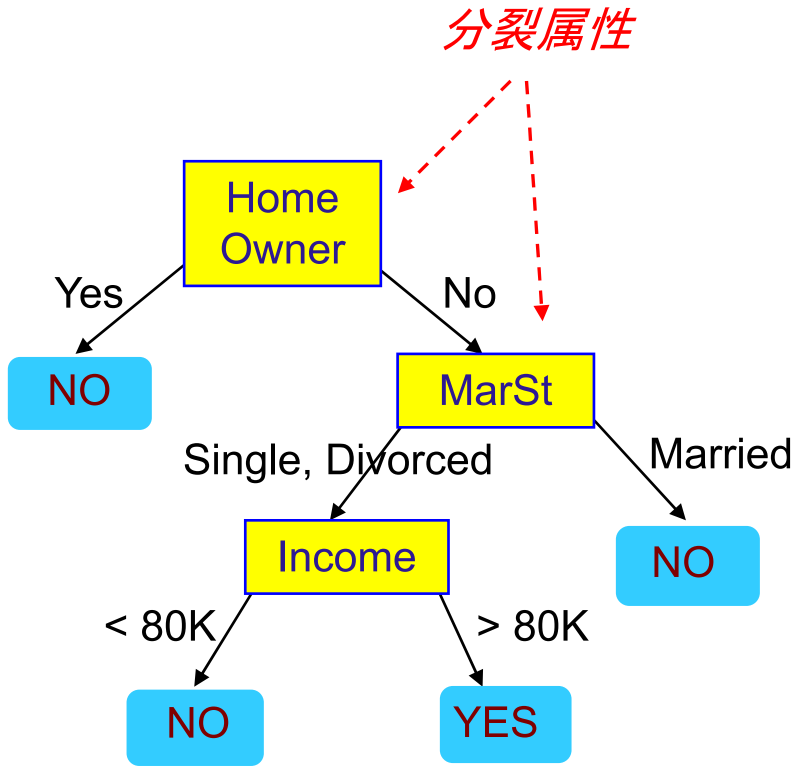


决策树——训练举例:

categorical categorical continuous class

ID	Home Owner	Marital Status	Annual Income	Defaulted Borrower
1	Yes	Single	125K	No
2	No	Married	100K	No
3	No	Single	70K	No
4	Yes	Married	120K	No
5	No	Divorced	95K	Yes
6	No	Married	60K	No
7	Yes	Divorced	220K	No
8	No	Single	85K	Yes
9	No	Married	75K	No
10	No	Single	90K	Yes

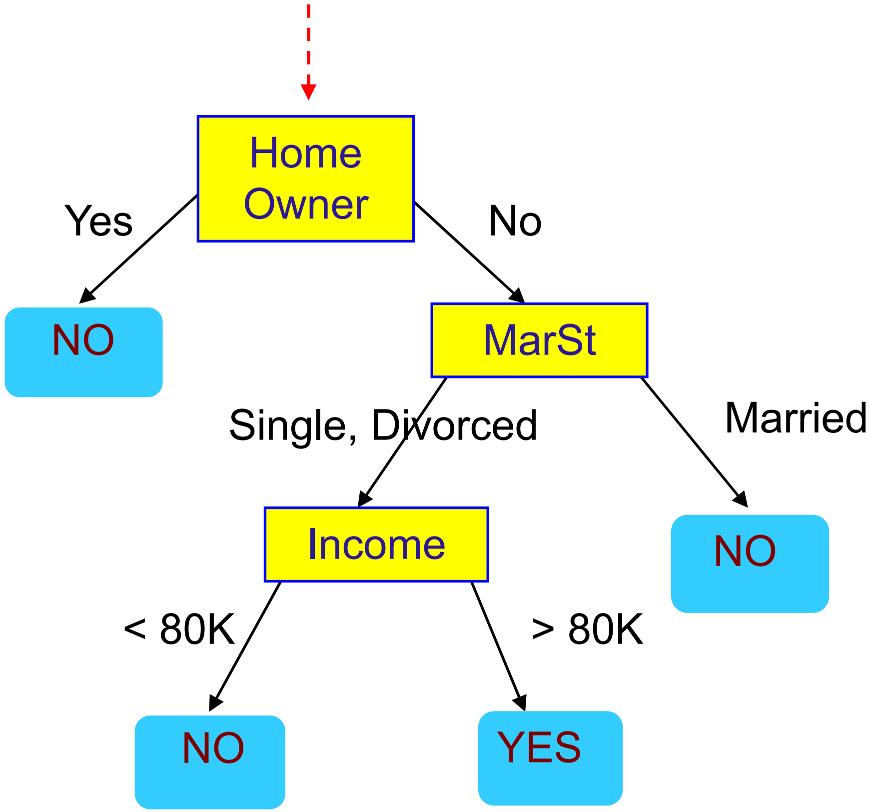
训练数据



模型:决策树

决策树——测试举例：

Start from the root of tree.



Test Data

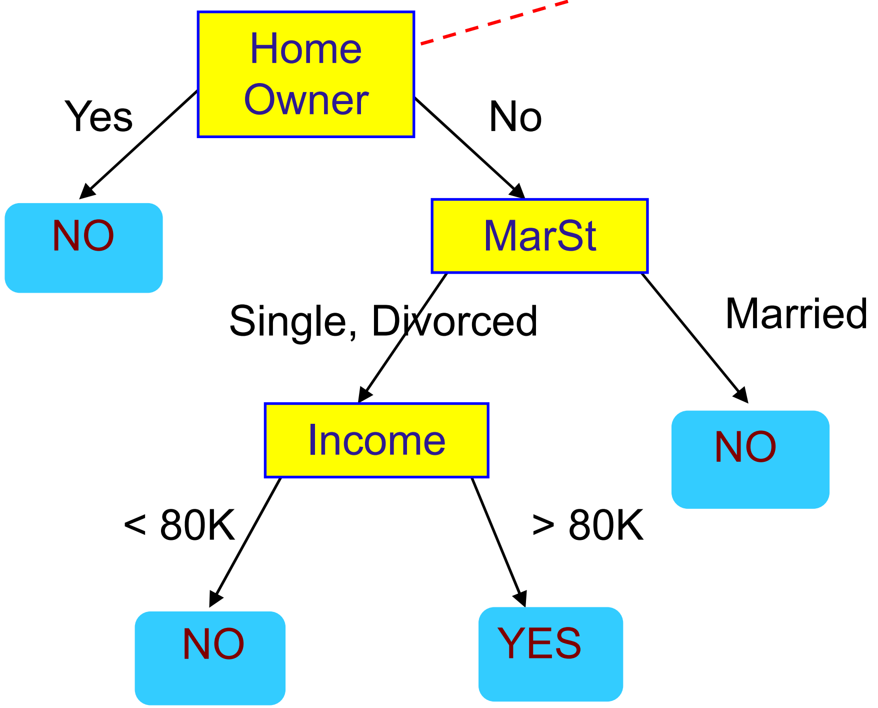
Home Owner	Marital Status	Annual Income	Defaulted Borrower
No	Married	80K	?

10

决策树——测试举例：

Test Data

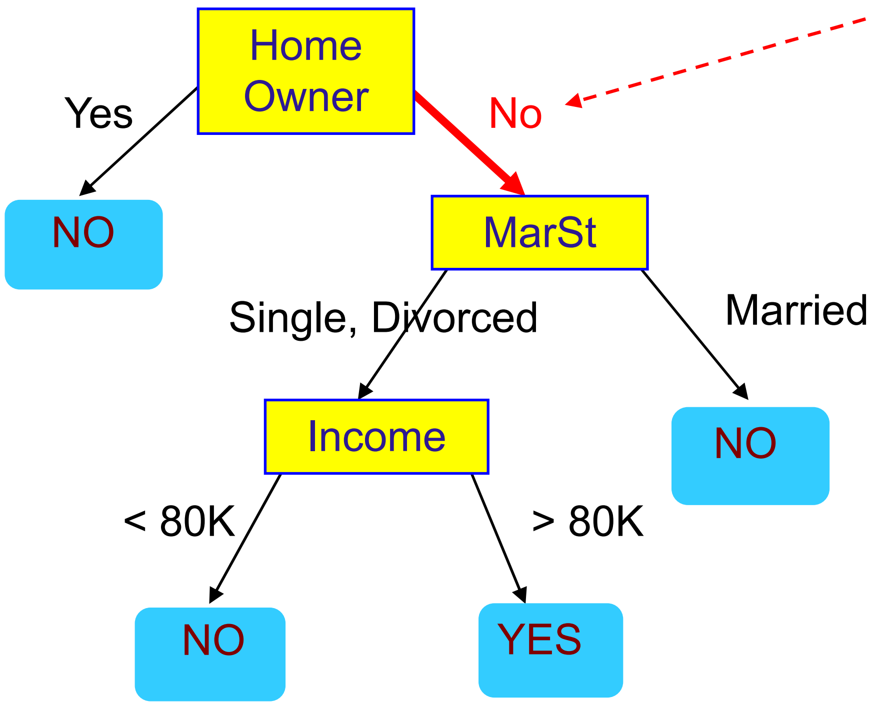
Home Owner	Marital Status	Annual Income	Defaulted Borrower
No	Married	80K	?



决策树——测试举例：

Test Data

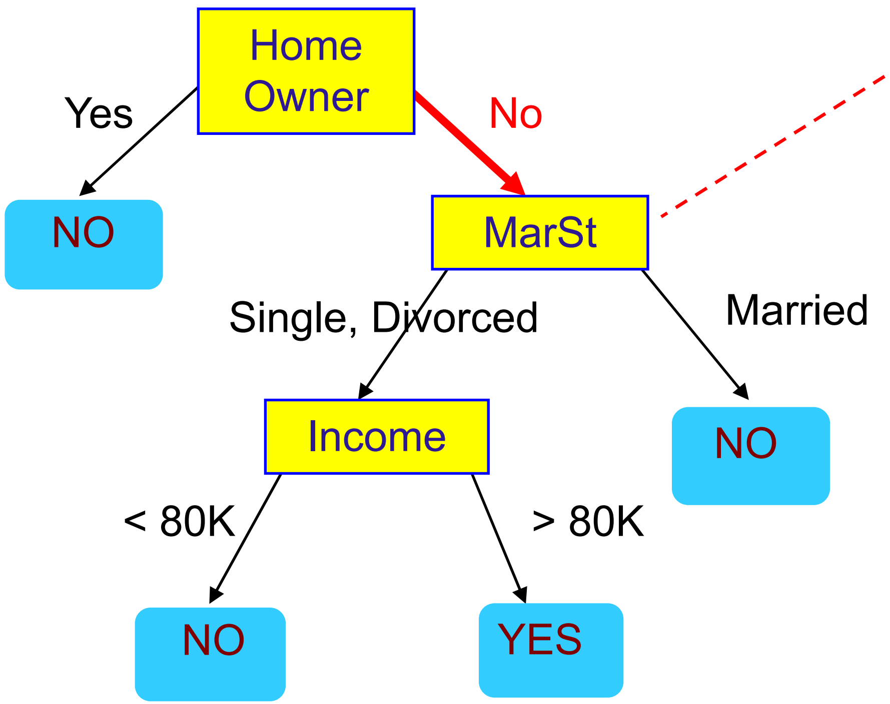
Home Owner	Marital Status	Annual Income	Defaulted Borrower
No	Married	80K	?



决策树——测试举例：

Test Data

Home Owner	Marital Status	Annual Income	Defaulted Borrower
No	Married	80K	?

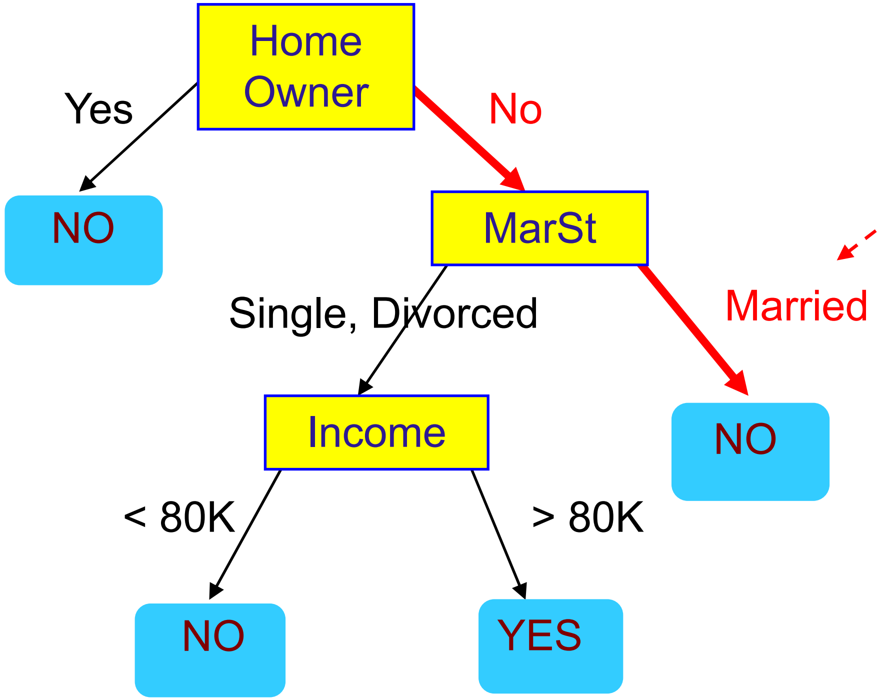


13

决策树——测试举例：

Test Data

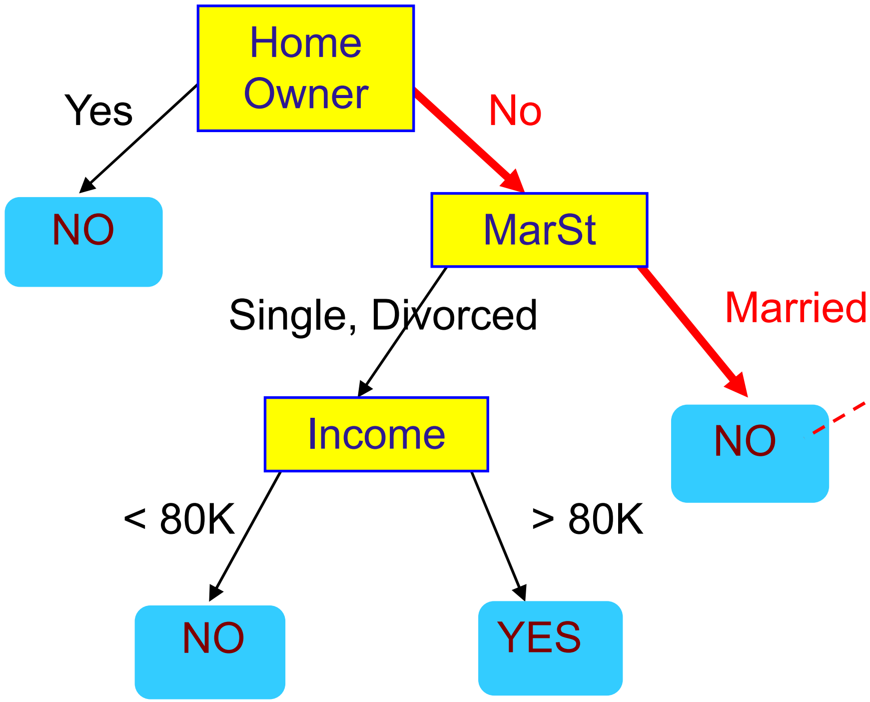
Home Owner	Marital Status	Annual Income	Defaulted Borrower
No	Married	80K	?



决策树——测试举例：

Test Data

Home Owner	Marital Status	Annual Income	Defaulted Borrower
No	Married	80K	?



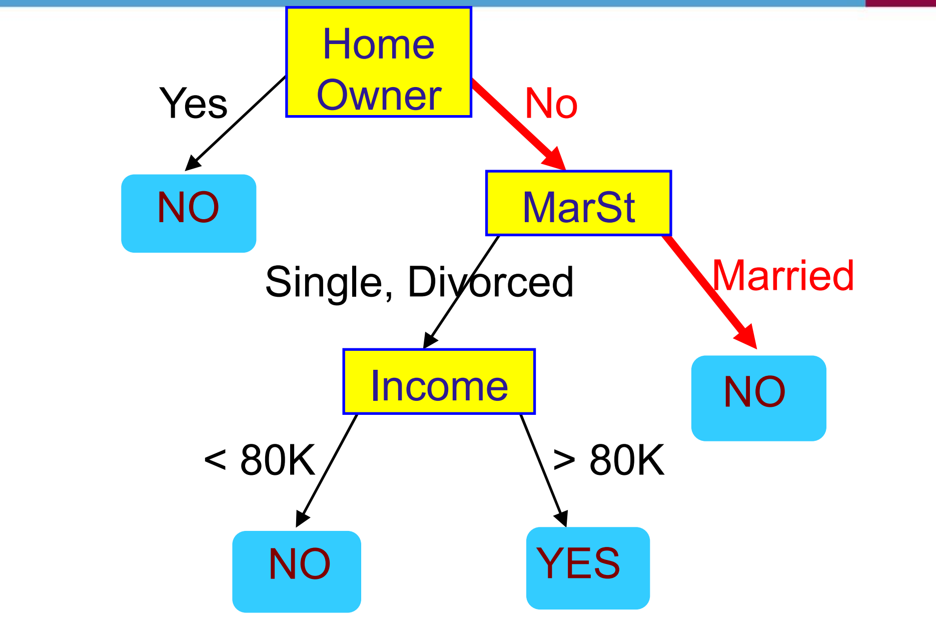
将默认值设置为“No”

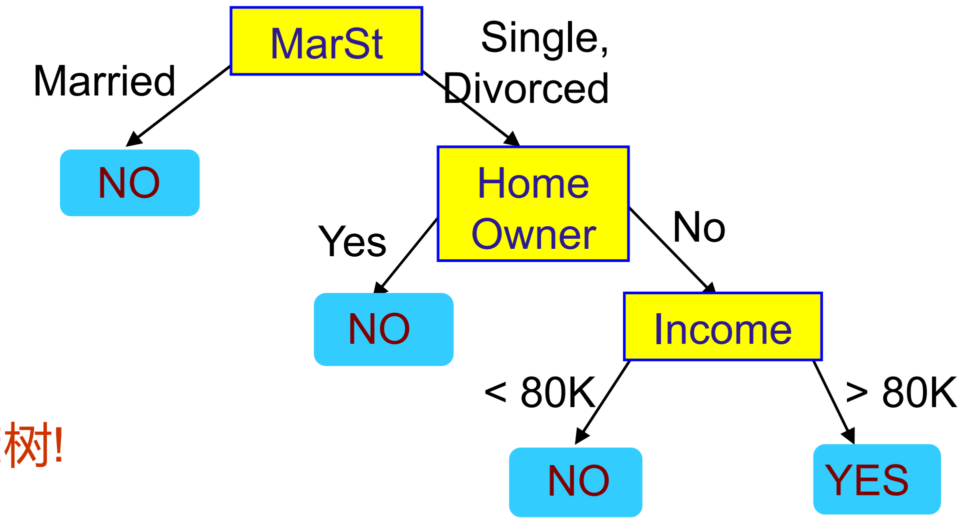
15

决策树——举例:

categorical categorical continuous class

ID	Home Owner	Marital Status	Annual Income	Defaulted Borrower
1	Yes	Single	125K	**No**
2	No	Married	100K	**No**
3	No	Single	70K	**No**
4	Yes	Married	120K	**No**
5	No	Divorced	95K	**Yes**
6	No	Married	60K	**No**
7	Yes	Divorced	220K	**No**
8	No	Single	85K	**Yes**
9	No	Married	75K	**No**
10	No	Single	90K	**Yes**



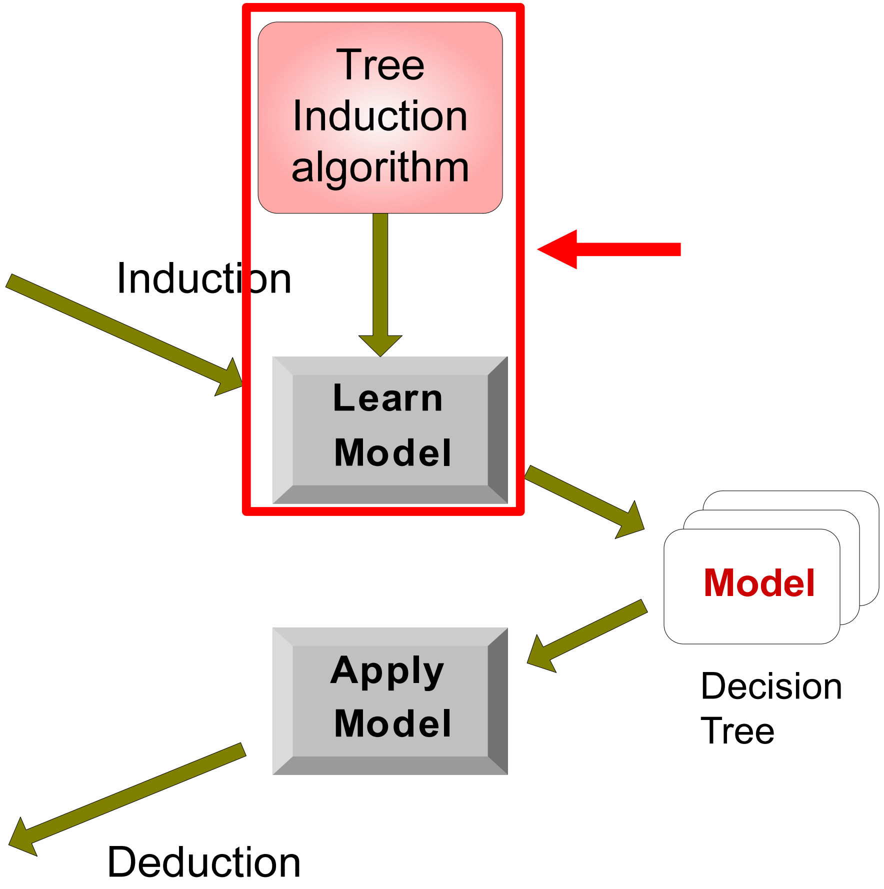


相同的训练数据可以构建多种不同的决策树!

决策树——构建：

Tid	Attrib1	Attrib2	Attrib3	Class
1	Yes	Large	125K	No
2	No	Medium	100K	No
3	No	Small	70K	No
4	Yes	Medium	120K	No
5	No	Large	95K	Yes
6	No	Medium	60K	No
7	Yes	Large	220K	No
8	No	Small	85K	Yes
9	No	Medium	75K	No
10	No	Small	90K	Yes

Training Set



Tid	Attrib1	Attrib2	Attrib3	Class
11	No	Small	55K	?
12	Yes	Medium	80K	?
13	Yes	Large	110K	?
14	No	Small	95K	?
15	No	Large	67K	?

Test Set

决策树——构建方法:

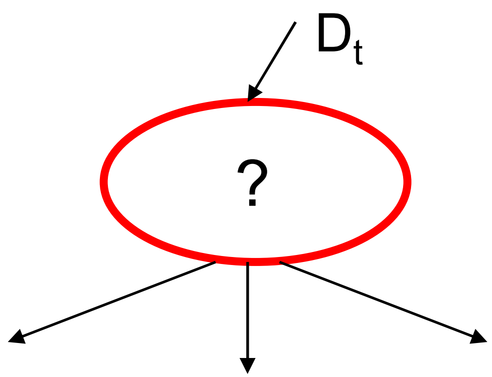
- 决策树构建理论:
  - Hunt’s 算法 (one of the earliest)
  - CART
  - ID3, C4.5
  - SLIQ,SPRINT

决策树构建——Hunt’s 算法

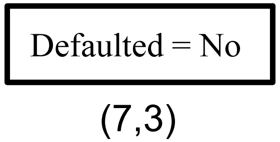
- Hunt算法通过将训练记录相继划分成较纯的子集，以递归方式建立决策树。
- 设 Dt 是与结点 t 相关联的训练记录集，而 y={y1,y2,…,yc}是类标号，Hunt算法的递归定义如下：

1. 如果 Dt 中所有记录都属于同一个类，则 t 是叶结点，用 yt 标记。
2. 如果 Dt 中包含属于多个类的记录，则选择一个属性测试条件（attribute test condition），将记录划分成较小的子集。对于测试条件的每个输出，创建一个子结点，并根据测试结果将 Dt 中的记录分布到子结点中。然后，对于每个子结点，递归地调用该算法。

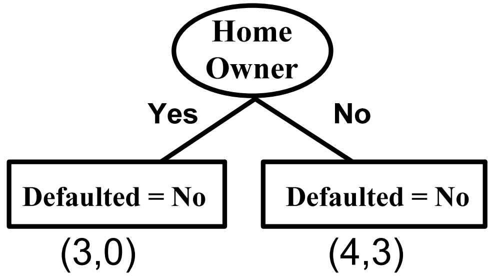
ID	Home Owner	Marital Status	Annual Income	Defaulted Borrower
1	Yes	Single	125K	No
2	No	Married	100K	No
3	No	Single	70K	No
4	Yes	Married	120K	No
5	No	Divorced	95K	Yes
6	No	Married	60K	No
7	Yes	Divorced	220K	No
8	No	Single	85K	Yes
9	No	Married	75K	No
10	No	Single	90K	Yes



决策树构建——Hunt’ s 算法

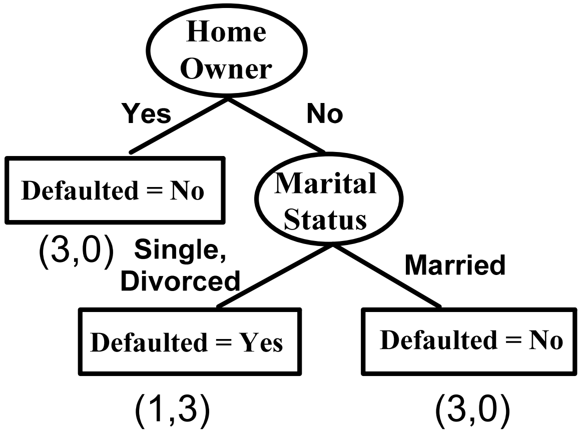


(a)

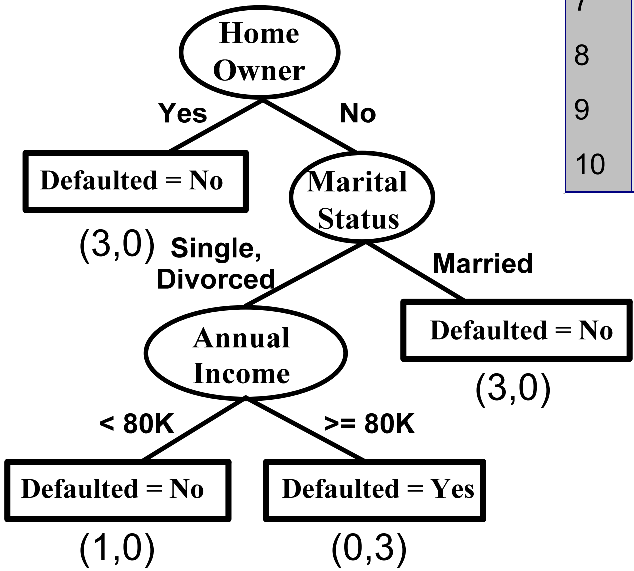


(b)

ID	Home Owner	Marital Status	Annual Income	Defaulted Borrower
1	Yes	Single	125K	No
2	No	Married	100K	No
3	No	Single	70K	No
4	Yes	Married	120K	No
5	No	Divorced	95K	Yes
6	No	Married	60K	No
7	Yes	Divorced	220K	No
8	No	Single	85K	Yes
9	No	Married	75K	No
10	No	Single	90K	Yes



(c)



(d)

决策树如何构建

- 训练记录如何分裂?
  - 选择测试条件的方法
    - 依赖属性类型
  - 测试条件的评价
- 分裂过程何时停止?
  - 停止分类如果所有记录属于同一类或者所有数据有相同的属性值
  - 提前终止

测试条件表达方法

Depends on attribute types

– Binary（二元）
– Nominal（标称）
– Ordinal（有序）
– Continuous（连续）

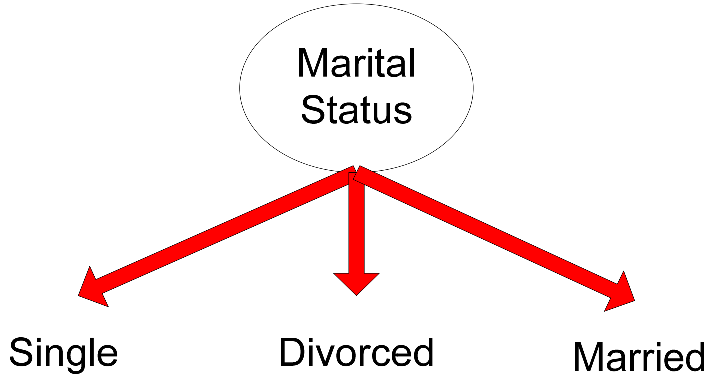
Depends on number of ways to split

– 2-way split（二路分裂）
– Multi-way split（多路分裂）

二元、标称属性的测试条件

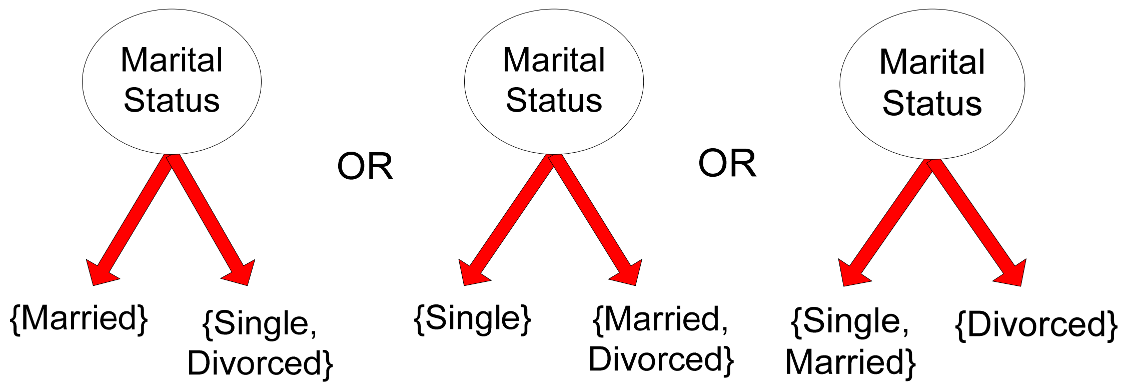
Multi-way split（多路分裂）：

– 使用和属性值一样多的分类



Binary split（二分裂）：

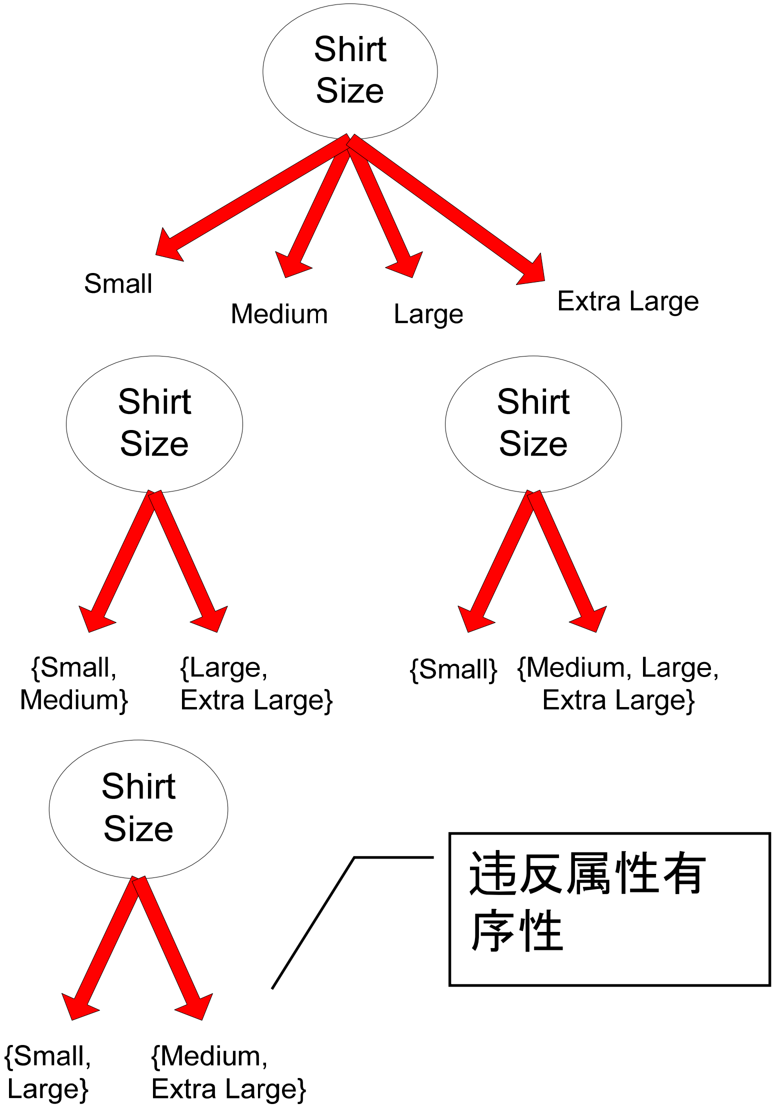
– 将属性值划分为两个子集



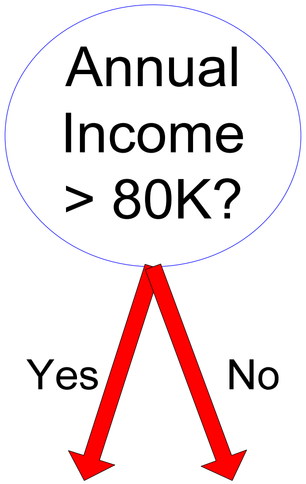
– 某些决策树算法（如CART）只产生二元划分，这些算法可创建k个属性值二元划分的$2^{k-1}-1$种方法。

有序属性测试条件

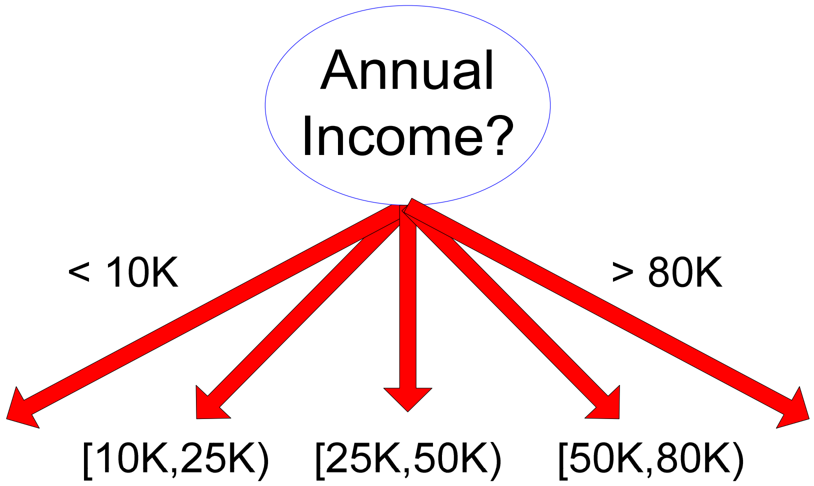
- Multi-way split（多路分裂）：
  – 使用和属性值一样多的分类
- Binary split （二分裂）：
  – 将属性值划分为两个子集
  – 保持属性值的顺序属性



连续属性测试条件



(i) Binary split



(ii) Multi-way split

连续属性的分裂

不同处理方式

– 离散化地处理有序的分类属性：

可以通过等间隔分段、等频率分段或聚类来找到范围

- Static（静态） – 在开始时进行一次离散化
- Dynamic（动态） – 在每个节点反复进行离散化

– 二值划分: $(A < v)$ or $(A \geq v)$

- consider all possible splits and finds the best cut

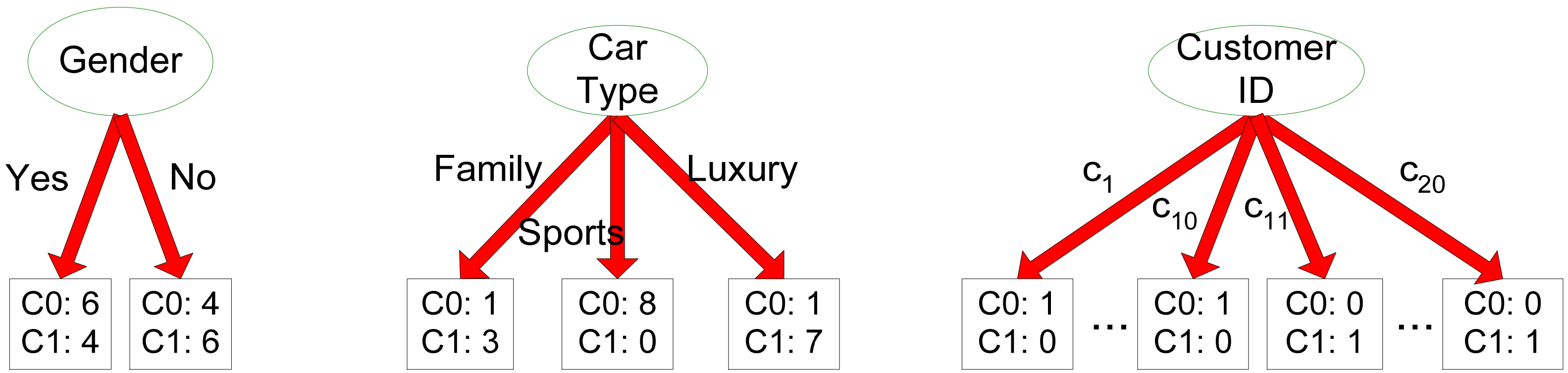
（考虑所有情况，找出最好的划分）

- can be more compute intensive（计算密集）

如何确定最好的分裂

在分裂之前: 10 records of class 0,
10 records of class 1。

Customer Id	Gender	Car Type	Shirt Size	Class
1	M	Family	Small	C0
2	M	Sports	Medium	C0
3	M	Sports	Medium	C0
4	M	Sports	Large	C0
5	M	Sports	Extra Large	C0
6	M	Sports	Extra Large	C0
7	F	Sports	Small	C0
8	F	Sports	Small	C0
9	F	Sports	Medium	C0
10	F	Luxury	Large	C0
11	M	Family	Large	C1
12	M	Family	Extra Large	C1
13	M	Family	Medium	C1
14	M	Luxury	Extra Large	C1
15	F	Luxury	Small	C1
16	F	Luxury	Small	C1
17	F	Luxury	Medium	C1
18	F	Luxury	Medium	C1
19	F	Luxury	Medium	C1
20	F	Luxury	Large	C1



Which test condition is the best?

如何确定最好的分裂

- 贪心（Greedy）策略:
  - 数据类别越纯（purer）优先级越高

- 需要计算点的不纯性（ impurity ）：

C0: 5 C1: 5	C0: 9 C1: 1
High degree of impurity	Low degree of impurity

不纯性(impurity)计算

- 基尼指数 Gini Index

$$Gini\ Index = 1 - \sum_{i=0}^{c-1} p_i(t)^2$$

- 熵 Entropy

$$Entropy = -\sum_{i=0}^{c-1} p_i(t) log_2 p_i(t)$$

- 误分率 Misclassification error

$$Classification\ error = 1 - \max[p_i(t)]$$

$\boldsymbol{p_i(t)}$ 是在节点t第i个类别出现的频率;
$\mathbf{c}$是类别的总数。

找到最好的分裂

1. 分裂前计算不纯性(P)
2. 分裂后计算不纯性(M)
   - 计算每个子节点（child nodes）的不纯性
   - M 是子节点不纯性的加权平均
3. 选择能获得最高增益的属性作为测试条件

$$Gain = P - M;$$

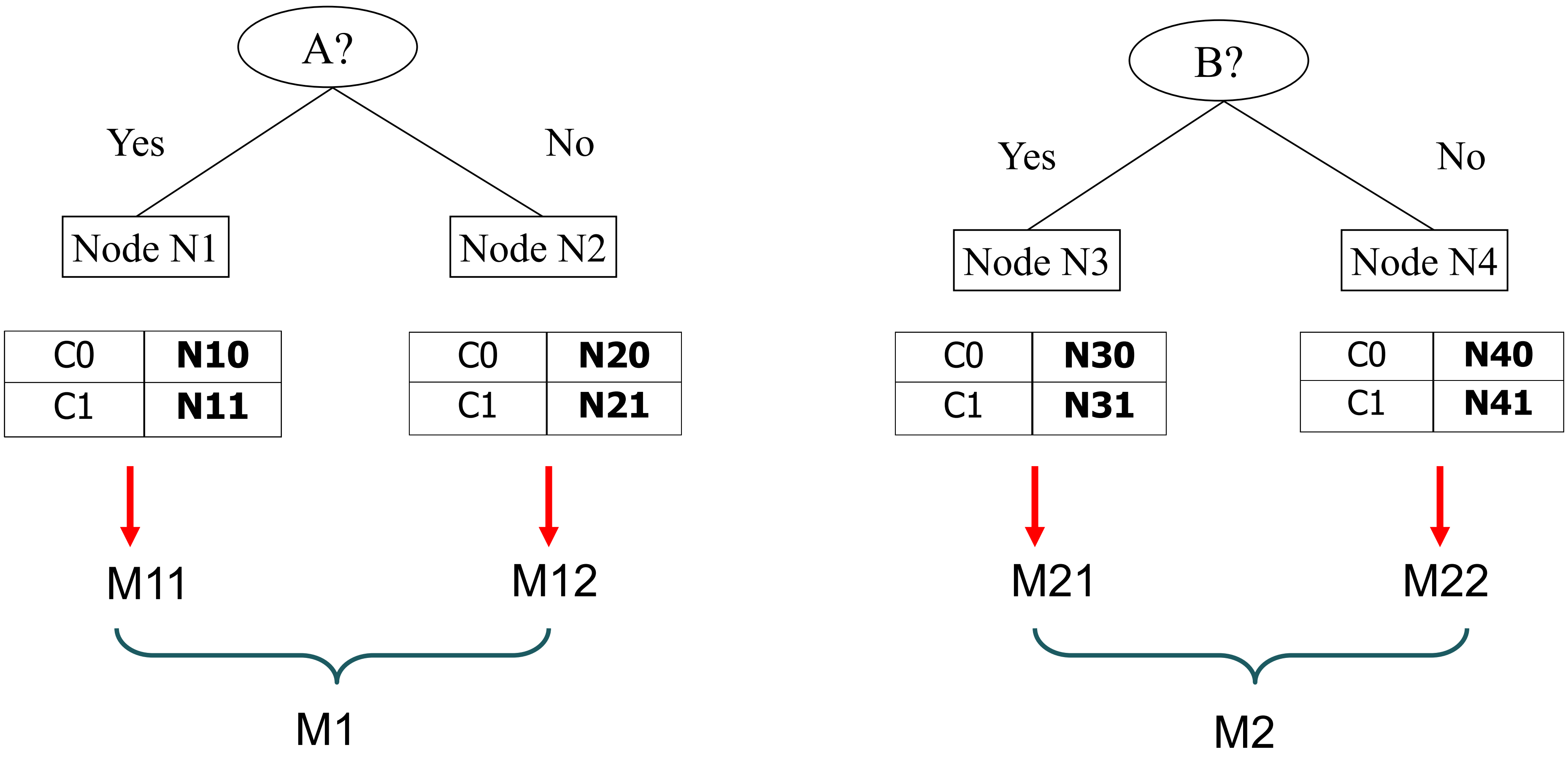
或者是分裂后最小的不纯性(M)作为测试条件。

找到最好的分裂

Before Splitting:

C0	**N00**
C1	**N01**

→ P



Gain = P – M1 vs P – M2

计算不纯性GINI

- 节点t 的Gini指数：

$$Gini\ Index = \sum_{i=0}^{c-1} p_i(t)(1-p_i(t)) = 1 - \sum_{i=0}^{c-1} p_i(t)^2$$

其中$\boldsymbol{p_i}(\boldsymbol{t})$ 是在节点t第i个类别出现的频率; **c**是类别的总数。

  - 最大值：$1-1/c$，当记录在所有类别中平均分布时，意味着对分类最不利的情况
  - 最小值： 0，当所有的记录都属于同一类时，意味着最有利于分类的情况
  - 在CART、SLIQ、SPRINT等决策树算法中均使用了基尼系数

计算不纯性GINI

- 节点t 的Gini指数：

$$Gini\ Index = 1 - \sum_{i=0}^{c-1} p_i(t)^2$$

  – 对二分类问题 (p, 1 – p):
    - GINI = 1 – p^2 – (1 – p)^2 = 2p (1-p)

C1	**0**
C2	**6**
Gini=0.000	

$2 * 0 * 1$

C1	**1**
C2	**5**
Gini=0.278	

$2 * 1/6 * 5/6$

C1	**2**
C2	**4**
Gini=0.444	

$2 * 2/6 * 4/6$

C1	**3**
C2	**3**
Gini=0.500	

$2 * 3/6 * 3/6$

计算单个节点Gini值

$$Gini\ Index = 1 - \sum_{i=0}^{c-1} p_i(t)^2$$

C1	**0**
C2	**6**

P(C1) = 0/6 = 0 P(C2) = 6/6 = 1

Gini = 1 – $P(C1)^2$ – $P(C2)^2$ = 1 – 0 – 1 = 0

C1	**1**
C2	**5**

P(C1) = 1/6 P(C2) = 5/6

Gini = 1 – $(1/6)^2$ – $(5/6)^2$ = 0.278

C1	**2**
C2	**4**

P(C1) = 2/6 P(C2) = 4/6

Gini = 1 – $(2/6)^2$ – $(4/6)^2$ = 0.444

计算集合中点的Gini值

- 当一个节点 p 分裂为k 个部分

$$GINI_{split} = \sum_{i=1}^{k} \frac{n_i}{n} GINI(i)$$

where, n_i = 子节点i中的记录数,

n = 父节点p中的记录数.

- 选择能使节点基尼指数最小化的属性

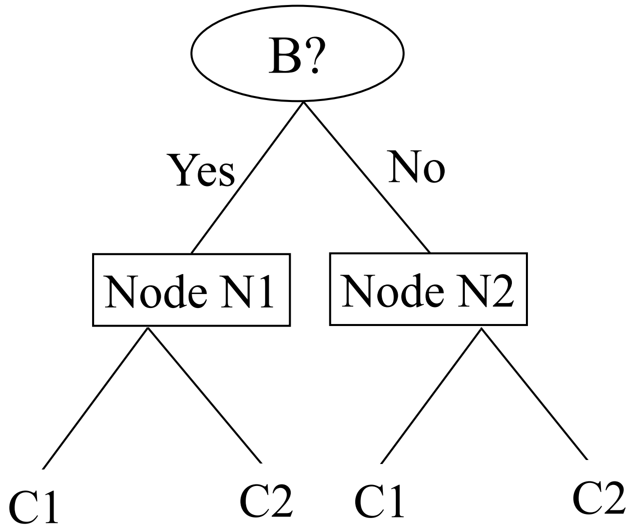
- 在CART、SLIQ、SPRINT等决策树算法中均使用了Gini指数

二元属性: 计算GINI值

- 分裂成两个部分
- 加权划分的效果:
  - 寻求更大更纯的划分

	Parent
C1	7
C2	5
Gini = 0.486	

	N1	N2
C1	5	2
C2	1	4
Gini=0.361		



1. Gini(N1) = $1 – (5/6)^2 – (1/6)^2$
   $= 0.278$

Gain = 0.486 – 0.361 = 0.125

2. Gini(N2) = $1 – (2/6)^2 – (4/6)^2$
   $= 0.444$

3. Weighted Gini of N1 N2 = 6/12 * 0.278 +6/12 * 0.444 = 0.361

标称 (Categorical) 属性: 计算GINI值

- 对数据集中的每个类进行计数
- 用计数矩阵（count matrix）来做决定

Multi-way split

	CarType		
	Family	Sports	Luxury
C1	1	8	1
C2	3	0	7
Gini	0.1625		

Two-way split
(find best partition of values)

	CarType	
	{Sports, Luxury}	{Family}
C1	9	1
C2	7	3
Gini	0.468	

	CarType	
	{Sports}	{Family, Luxury}
C1	8	2
C2	0	10
Gini	0.167	

Which of these is the best?

连续属性: 计算GINI值

- 基于一个属性值进行二元决策（Binary Decisions ）
- 分割值的几种选择
  - Number of possible splitting values = Number of distinct values

  （可能分割值的数目=不同值的数目）
- 每个分割值都有一个与之相关联的计数矩阵
  - 在每个分区内进行类计数, A ≤ v and A > v
- 选择v的简单方法:
  - 对于每个v，扫描数据库收集计数矩阵并计算其基尼指数
  - 缺点计算效率低下! 重复工作。

ID	Home Owner	Marital Status	Annual Income	Defaulted
1	Yes	Single	125K	No
2	No	Married	100K	No
3	No	Single	70K	No
4	Yes	Married	120K	No
5	No	Divorced	95K	Yes
6	No	Married	60K	No
7	Yes	Divorced	220K	No
8	No	Single	85K	Yes
9	No	Married	75K	No
10	No	Single	90K	Yes

Annual Income ?

	≤ 80	> 80
Defaulted Yes	0	3
Defaulted No	3	4

连续属性: 计算GINI值

高效计算: 对每一个属性,
- Sort the attribute on values（排序)
- 线性放缩这些值，每隔一段时间更新计数矩阵和计算基尼值
- 选择分裂位置，得到最小的基尼值

Cheat	No	No	No	Yes	Yes	Yes	No	No	No	No
	Annual Income									
Sorted Values →	60	70	75	85	90	95	100	120	125	220

Split Positions →	55		65		72		80		87		92		97		110		122		172		230	
	<=	>	<=	>	<=	>	<=	>	<=	>	<=	>	<=	>	<=	>	<=	>	<=	>	<=	>
Yes	0	3	0	3	0	3	0	3	1	2	2	1	3	0	3	0	3	0	3	0	3	0
No	0	7	1	6	2	5	3	4	3	4	3	4	3	4	4	3	5	2	6	1	7	0
Gini	0.420		0.400		0.375		0.343		0.417		0.400		***0.300***		0.343		0.375		0.400		0.420	

基于GINI值构建决策树

Day	Outlook	Temp.	Decision
1	Sunny	Hot	No
2	Overcast	Hot	Yes
3	Rain	Mild	Yes
4	Rain	Cool	Yes
5	Rain	Cool	No
6	Overcast	Cool	Yes
7	Sunny	Mild	No
8	Sunny	Cool	Yes
9	Sunny	Mild	Yes
10	Overcast	Mild	Yes

Outlook	Yes	No	Number of instances
Sunny	2	2	4
Overcast	3	0	3
Rain	2	1	3

Temp.	Yes	No	Number of instances
Hot	1	1	2
Mild	3	1	4
Cool	3	1	4

$$Gini(Outlook = Sunny) = 1 - \left(\frac{2}{4}\right)^2 - \left(\frac{2}{4}\right)^2 = 0.5$$

$$Gini(Outlook = Overcast) = 1- \left(\frac{3}{3}\right)^2 - \left(\frac{0}{3}\right)^2 = 0$$

$$Gini(Outlook = Rain) = 1- \left(\frac{2}{3}\right)^2 - \left(\frac{1}{3}\right)^2 = 0.44$$

$$Gini(Outlook) = \left(\frac{4}{10}\right) * 0.5 + \left(\frac{3}{10}\right) * 0 + \left(\frac{3}{10}\right) * 0.44 = 0.332$$

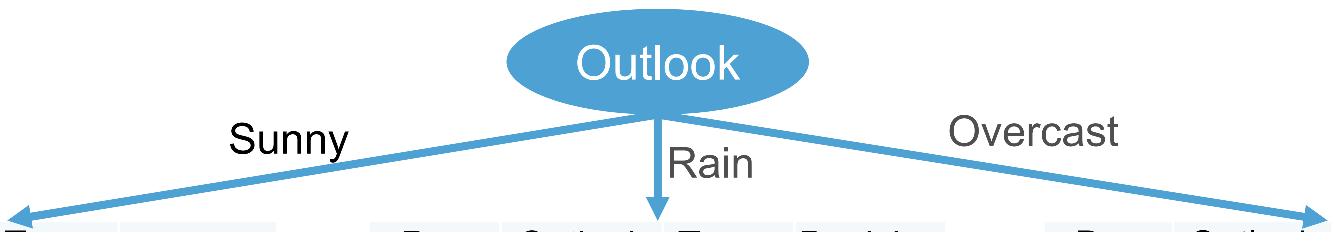
$$Gini(Temp.) = ?$$

连续属性: 计算GINI值

Day	Outlook	Temp.	Decision
1	Sunny	Hot	No
2	Overcast	Hot	Yes
3	Rain	Mild	Yes
4	Rain	Cool	Yes
5	Rain	Cool	No
6	Overcast	Cool	Yes
7	Sunny	Mild	No
8	Sunny	Cool	Yes
9	Sunny	Mild	Yes
10	Overcast	Mild	Yes

$$Gini(Temp.) = 0.4$$

$$Gini(Outlook) = 0.332$$



Day	Outlook	Temp.	Decision
1	Sunny	Hot	No
7	Sunny	Mild	No
8	Sunny	Cool	Yes
9	Sunny	Mild	Yes

Day	Outlook	Temp.	Decision
3	Rain	Mild	Yes
4	Rain	Cool	Yes
5	Rain	Cool	No

Day	Outlook	Temp.	Decision
2	Overcast	Hot	Yes
6	Overcast	Cool	Yes
10	Overcast	Mild	Yes

Yes

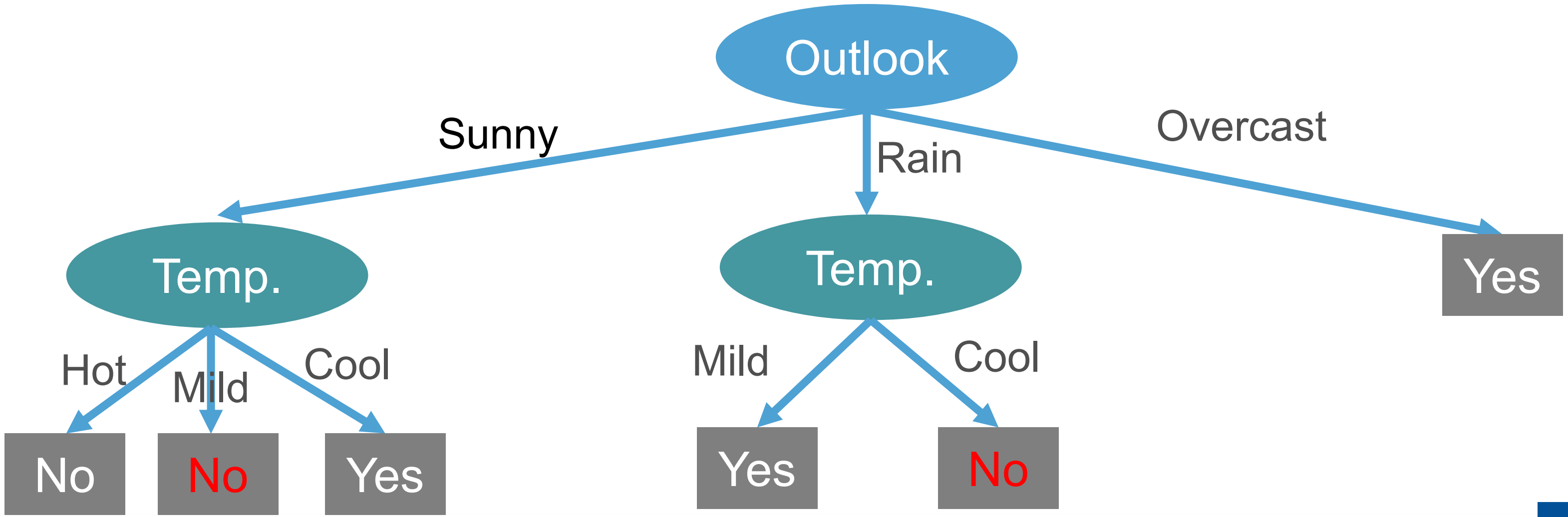
基于GINI值构建决策树

Day	Outlook	Temp.	Decision
1	Sunny	Hot	No
7	Sunny	Mild	No
8	Sunny	Cool	Yes
9	Sunny	Mild	Yes

Day	Outlook	Temp.	Decision
3	Rain	Mild	Yes
4	Rain	Cool	Yes
5	Rain	Cool	No

Temp.	Yes	No	Number of instances
Hot	0	1	1
Mild	1	1	2
Cool	1	0	1

Temp.	Yes	No	Number of instances
Hot	0	0	0
Mild	1	0	1
Cool	1	1	2



基于GINI值构建决策树

日期	天气	温度	湿度	风力	是否施肥
202101	晴天	炎热	高	弱风	否
202102	晴天	炎热	高	强风	否
202103	阴天	炎热	高	弱风	是
202104	雨天	温	高	弱风	是
202105	雨天	冷	中	弱风	是
202106	雨天	冷	中	强风	否
202107	阴天	冷	中	强风	是
202108	晴天	温	高	弱风	否
202109	晴天	冷	中	弱风	是
202110	雨天	温	中	弱风	是
202111	晴天	温	中	强风	是
202112	阴天	温	高	强风	是
202201	阴天	炎热	中	弱风	是
202202	雨天	温	高	强风	否

计算不纯度：Entropy

- 节点 t 的熵（Entropy）：

$$Entropy = -\sum_{i=0}^{c-1} p_i(t) log_2 p_i(t)$$

其中$p_i(t)$是在节点t第i个类别出现的频率; c是类别的总数。

- 最大值：$\log_2 c$，当记录在所有类别中平均分布时，意味着对分类最不利的情况
- 最小值： 0，当所有的记录都属于同一类时，意味着最有利于分类的情况
- 熵的计算和基尼值的计算很相似

计算单个节点的熵

$$Entropy = -\sum_{i=0}^{c-1} p_i(t) log_2 p_i(t)$$

C1	**0**
C2	**6**

P(C1) = 0/6 = 0 P(C2) = 6/6 = 1

Entropy = – 0 log 0 – 1 log 1 = – 0 – 0 = 0

C1	**1**
C2	**5**

P(C1) = 1/6 P(C2) = 5/6

Entropy = – (1/6) $\log_2$ (1/6) – (5/6) $\log_2$ (5/6) = 0.65

C1	**2**
C2	**4**

P(C1) = 2/6 P(C2) = 4/6

Entropy = – (2/6) $\log_2$ (2/6) – (4/6) $\log_2$ (4/6) = 0.92

计算分类后的信息增益

信息增益:

$$Gain_{split} = Entropy(p) - \sum_{i=1}^{k} \frac{n_i}{n} Entropy(i)$$

父节点（Parent Node） p 被分裂为 k 个部分（子节点）

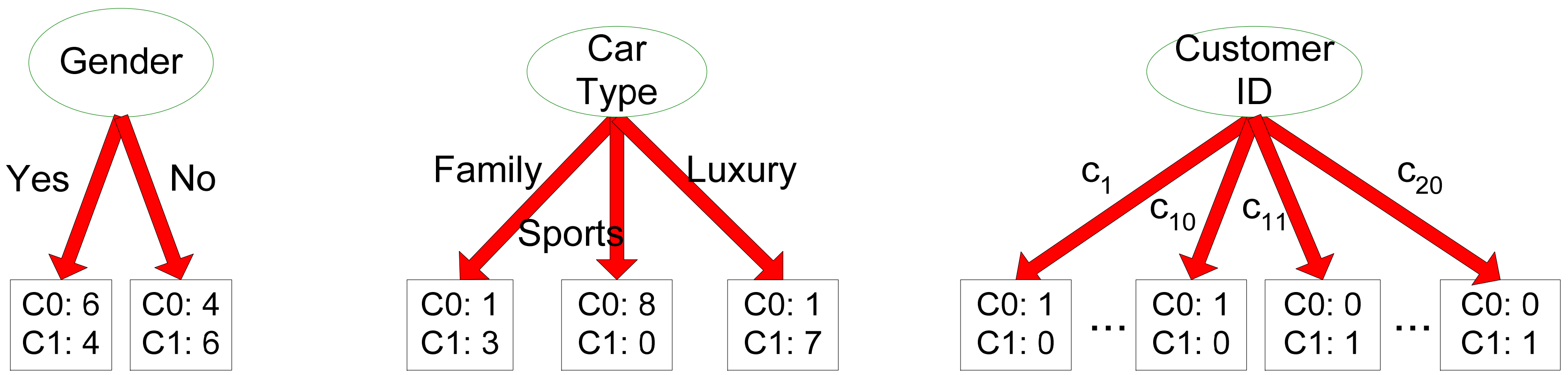
n_i 是子节点i中的记录数

- 选择达到最大增益的分割方式
- 在ID3、 C4.5 等决策树中使用
- Information gain is the mutual information between the class variable and the splitting variable

（信息增益是类变量与分裂变量之间的互信息）

信息增益问题

- 信息增益趋向分裂更多的子集，每一个子集越小越纯



➢ 客户ID具有最高的信息增益，因为所有子节点的熵为零

增益率的计算

- Gain Ratio（增益率）：

$$Gain\ Ratio = \frac{Gain_{split}}{Split\ Info} \qquad Split\ Info = -\sum_{i=1}^{k} \frac{n_i}{n} log_2 \frac{n_i}{n}$$

父节点（Parent Node） p 被分裂为 k 个部分（子节点）

n_i 是子节点i中的记录数; n是父节点的记录数;

  – 通过划分熵调整信息增益($Split\ Info$).
    ◆ 更高的熵的划分(large number of small partitions)会被惩罚!
  – 在C4.5中使用
  – 用于克服信息增益 (Information Gain) 的不足

- 如果某个属性产生了大量的划分，它的划分信息（Spilt Info）将会很大，从而降低增益率，这样基于增益率该属性不会被划分。

增益率的计算

- Gain Ratio:

$$Gain\ Ratio = \frac{Gain_{split}}{Split\ Info} \qquad Split\ Info = -\sum_{i=1}^{k} \frac{n_i}{n} log_2 \frac{n_i}{n}$$

父节点（Parent Node） p 被分裂为 k 个部分（子节点）

n_i 是子节点i中的记录数; n是父节点的记录数;

	CarType		
	Family	Sports	Luxury
C1	1	8	1
C2	3	0	7
Gini			

	CarType	
	{Sports, Luxury}	{Family}
C1	9	1
C2	7	3

	CarType	
	{Sports}	{Family, Luxury}
C1	8	2
C2	0	10

计算不纯性：分类错误

- 节点t 的分类错误（Classification error）

$$\boxed{Error(t) = 1 - \max_i [p_i(t)]}$$

  - 最大值：$1 - 1/c$，当记录在所有类别中平均分布时，意味着对分类最不利的情况
  - 最小值：0，当所有的记录都属于同一类时，意味着最有利于分类的情况

计算单个节点错误

$$Error(t) = 1 - \max_{i}[p_i(t)]$$

C1	**0**
C2	**6**

P(C1) = 0/6 = 0   P(C2) = 6/6 = 1

Error = 1 – max (0, 1) = 1 – 1 = 0

C1	**1**
C2	**5**

P(C1) = 1/6   P(C2) = 5/6

Error = 1 – max (1/6, 5/6) = 1 – 5/6 = 1/6

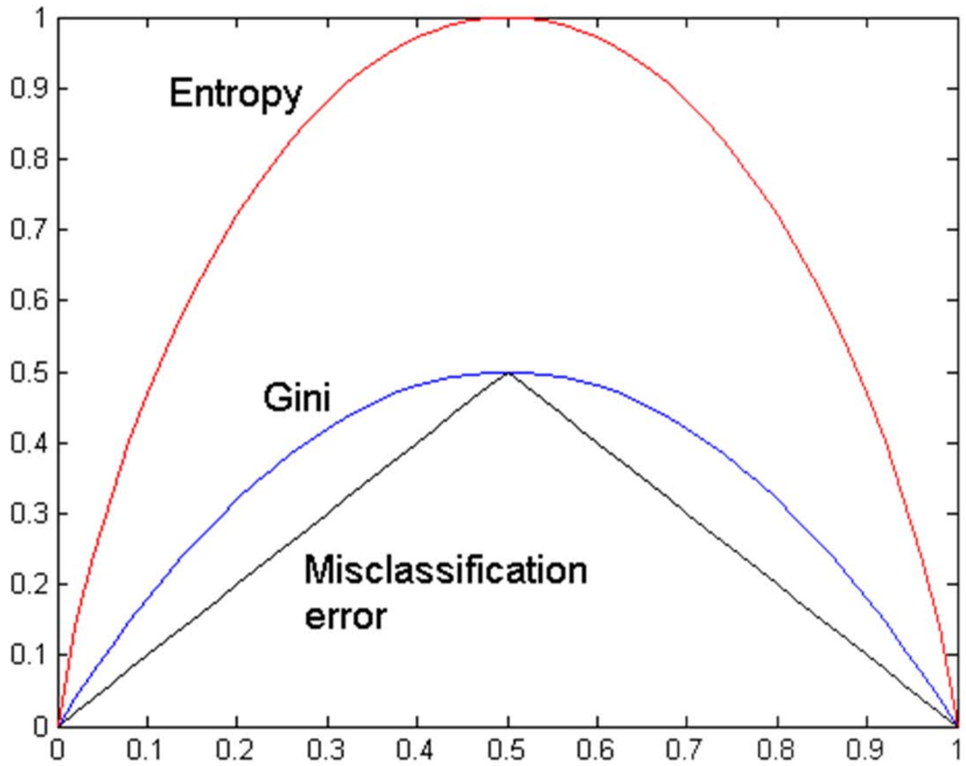
C1	**2**
C2	**4**

P(C1) = 2/6   P(C2) = 4/6

Error = 1 – max (2/6, 4/6) = 1 – 4/6 = 1/3

比较不纯性度量方式

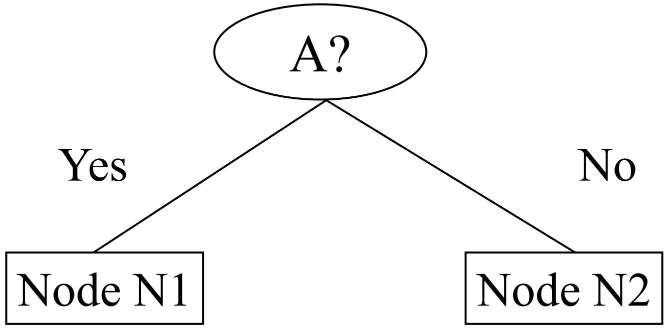
对二分类问题:



p表示属于其中一个类的记录所占的比例

不同的不纯性度量是一致的。但是，作为测试条件的属性选择仍然因不纯性度量的选择而异。

分类错误率 v.s. Gini值



	Parent
C1	**7**
C2	**3**
Gini = 0.42	

基于Gini指数计算:

Gini(N1)
$= 1 – (3/3)^2 – (0/3)^2$
= 0

Gini(N2)
$= 1 – (4/7)^2 – (3/7)^2$
= 0.489

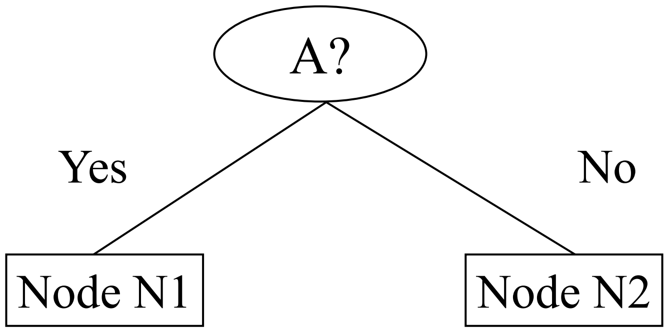
	N1	**N2**
C1	**3**	**4**
C2	**0**	**3**
Gini=0.342		

Gini(Children)
= 3/10 * 0
+ 7/10 * 0.489
= 0.342

Gini improves but error remains the same!!

基于分类错误计算：父亲节点的分类错误是1-0.7=0.3，儿子节点分别是0，3/7，所以，增益是0.3-0*3/10-3/7*7/10=0

分类错误率 v.s. Gini值



	Parent
C1	7
C2	3
Gini = 0.42	

	N1	N2
C1	3	4
C2	0	3
Gini=0.342		

	N1	N2
C1	3	4
C2	1	2
Gini=0.416		

Misclassification error for all three cases = 0.3 !

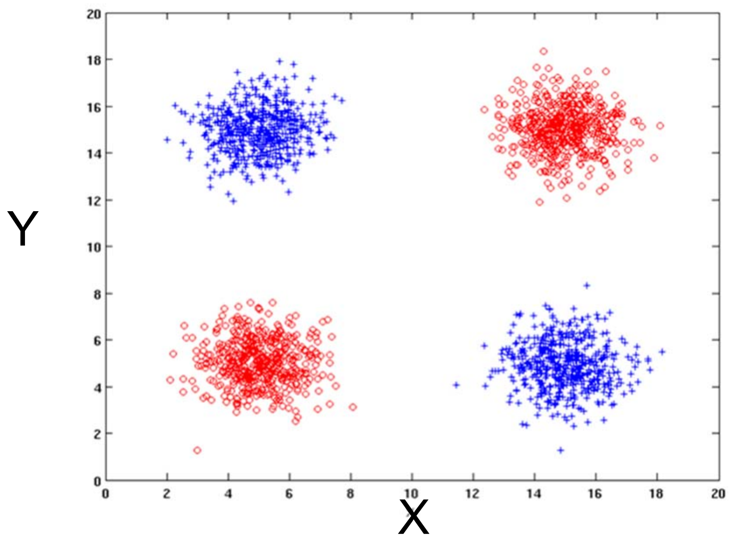
决策树停止分裂的条件

- 停止分裂直到所有节点属于同一类
- 停止分裂当所有记录有相同的属性值
- 提前终止（Early termination） (to be discussed later)

决策树分类的优缺点

- 优点:
  - ✓ 构造成本相对低廉
  - ✓ 对未知记录进行分类的速度非常快
  - ✓ 对于小型树木来说很容易解释
  - ✓ 抗噪声(特别是使用避免过拟合的方法时)
  - ✓ 可以轻松处理冗余属性吗
  - ✓ 可以轻松处理不相关的属性(除非属性相互作用)
- 缺点:
  - ✓ 由于分割标准的贪婪本质，交互属性（ interacting attributes ）(可以在一起区分类，但不能单独区分类)可能会被忽略，而倾向于其他鉴别能力较差的属性。
  - ✓ 每个决策边界只涉及一个属性

对待交互（interactions）

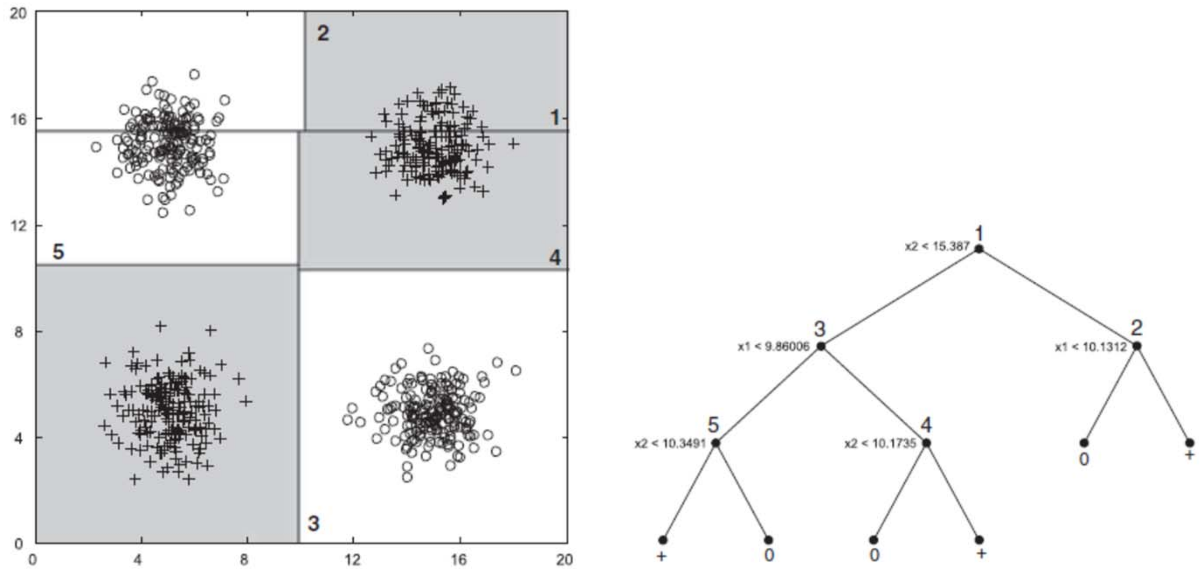


+ : 1000 instances

o : 1000 instances

Entropy (X) : 0.99
Entropy (Y) : 0.99

对待交互（interactions）

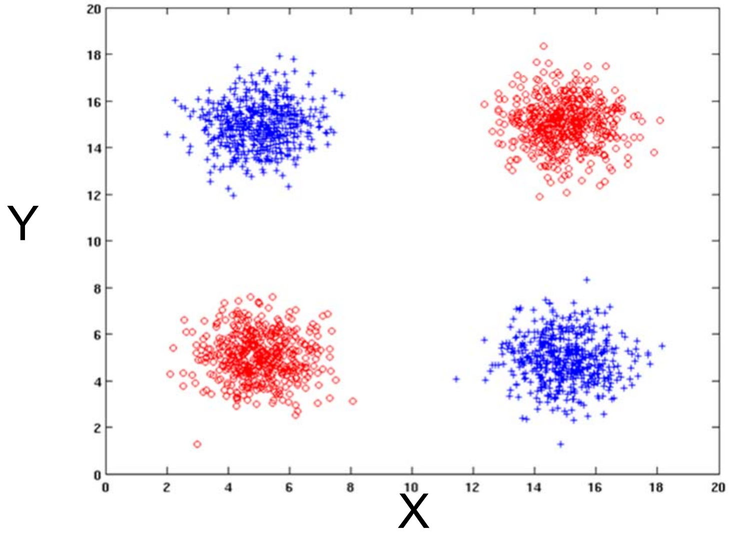


(a) Decision boundary for tree with 6 leaf nodes.

(b) Decision tree with 6 leaf nodes.

Figure 3.28. Decision tree with 6 leaf nodes using X and Y as attributes. Splits have been numbered from 1 to 5 in order of other occurrence in the tree.

处理给定不相关属性的交互



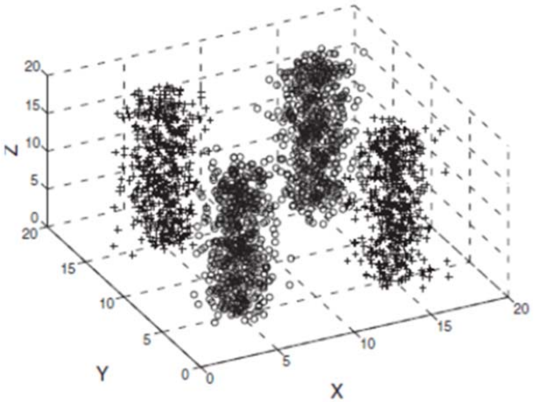
+ : 1000 instances

o : 1000 instances

添加Z作为均匀分布产生的噪声属性

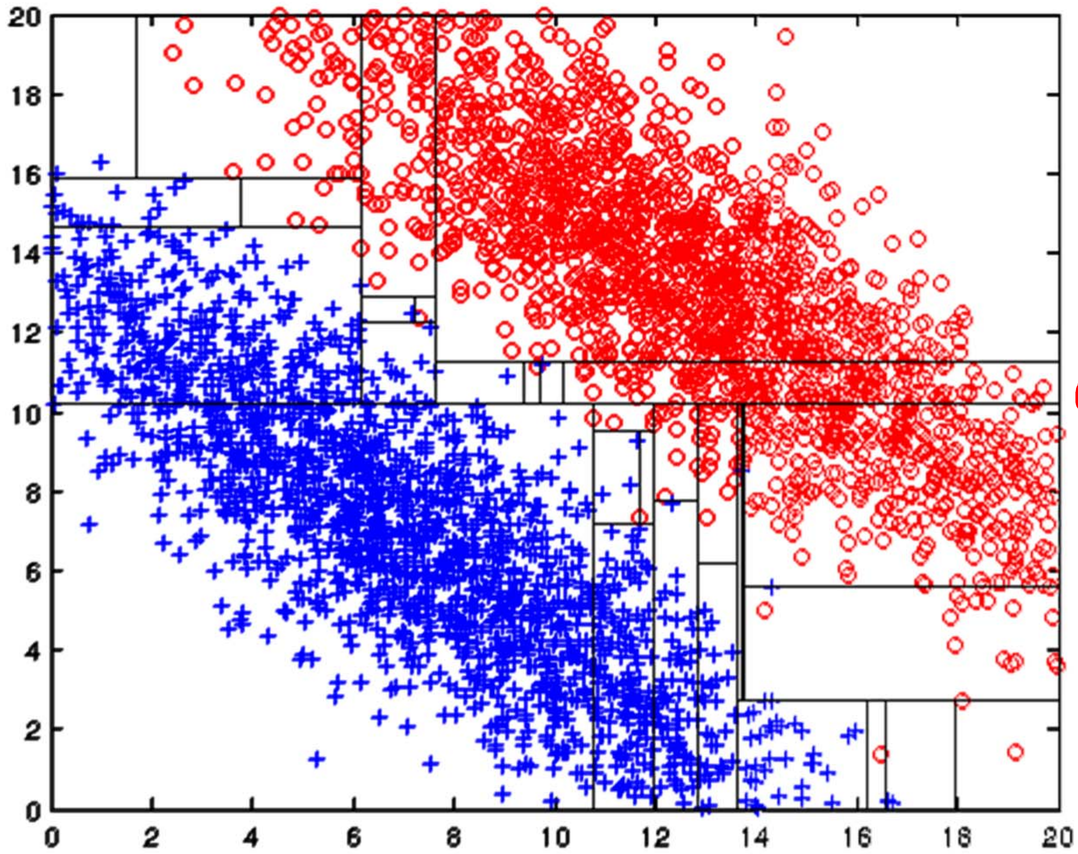
Entropy (X) : 0.99
Entropy (Y) : 0.99
Entropy (Z) : 0.98

属性Z将被用于进行分割!



(a) Three-dimensional data with attributes X, Y, and Z.

基于单一属性的决策边界的局限性



Both positive (+) and negative (o) classes generated from skewed Gaussians with centers at (8,8) and (12,12) respectively.

第二部分目标:

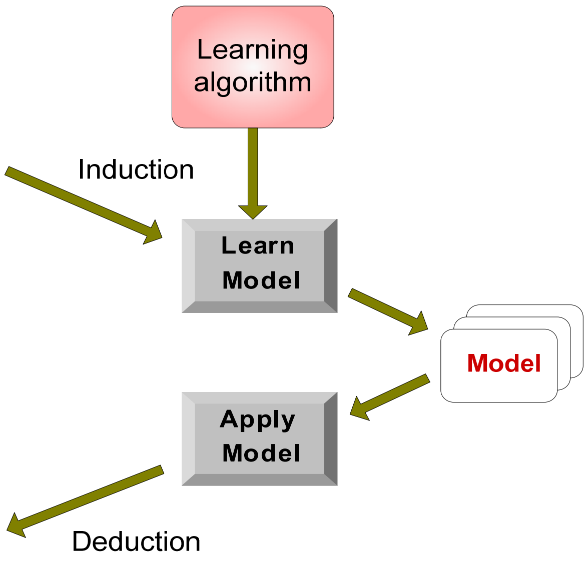
- 过拟合与欠拟合
- 分类模型评估方法
- ROC
- 样本不均衡与模型效果评价

分类误差

- 训练误差 (**Training errors**): 训练集上的误差
- 测试误差 (**Test errors**): 测试集上的误差
- 泛化误差 (**Generalization errors**): 从与训练集具有相同分布的数据集中选择数据进行测试的误差的期望值。(Expected error of a model over random selection of records from same distribution.)

Tid	Attrib1	Attrib2	Attrib3	Class
1	Yes	Large	125K	No
2	No	Medium	100K	No
3	No	Small	70K	No
4	Yes	Medium	120K	No
5	No	Large	95K	Yes
6	No	Medium	60K	No
7	Yes	Large	220K	No
8	No	Small	85K	Yes
9	No	Medium	75K	No
10	No	Small	90K	Yes

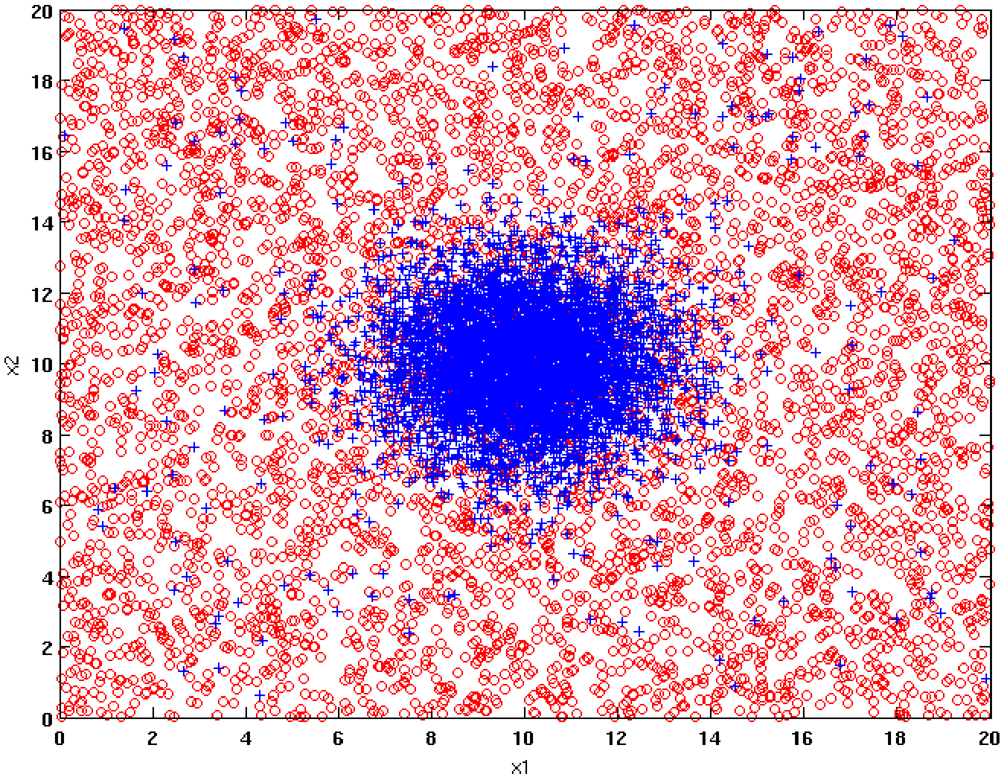
Training Set



Tid	Attrib1	Attrib2	Attrib3	Class
11	No	Small	55K	?
12	Yes	Medium	80K	?
13	Yes	Large	110K	?
14	No	Small	95K	?
15	No	Large	67K	?

Test Set

示例数据集



Two class problem:

+ : 5400 instances

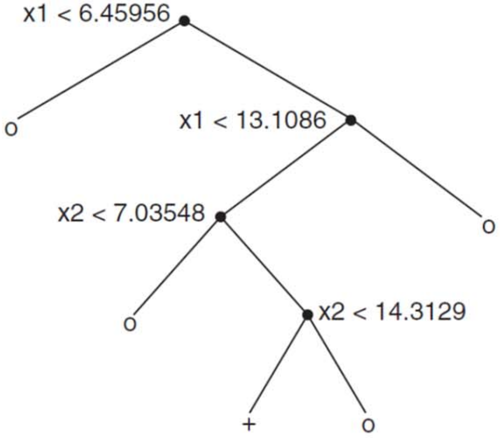
- **5000 instances generated from a Gaussian centered at (10,10)**
- **400 noisy instances added**

o : 5400 instances

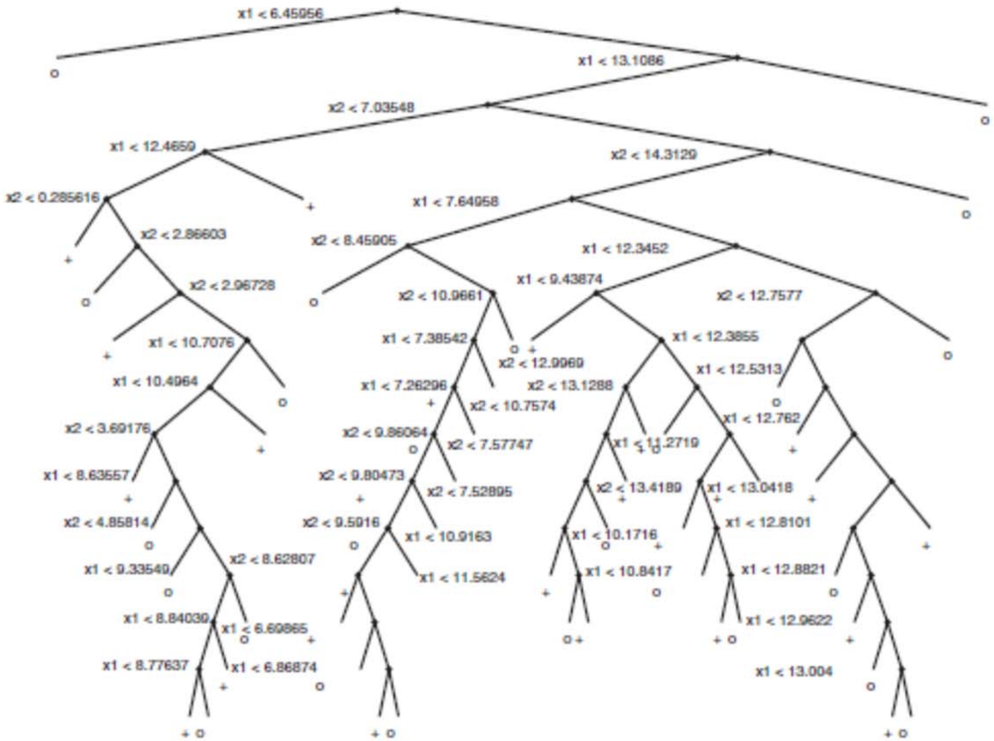
- **Generated from a uniform distribution**

10 % of the data used for training and 90% of the data used for testing

决策树



包含4个节点的决策树

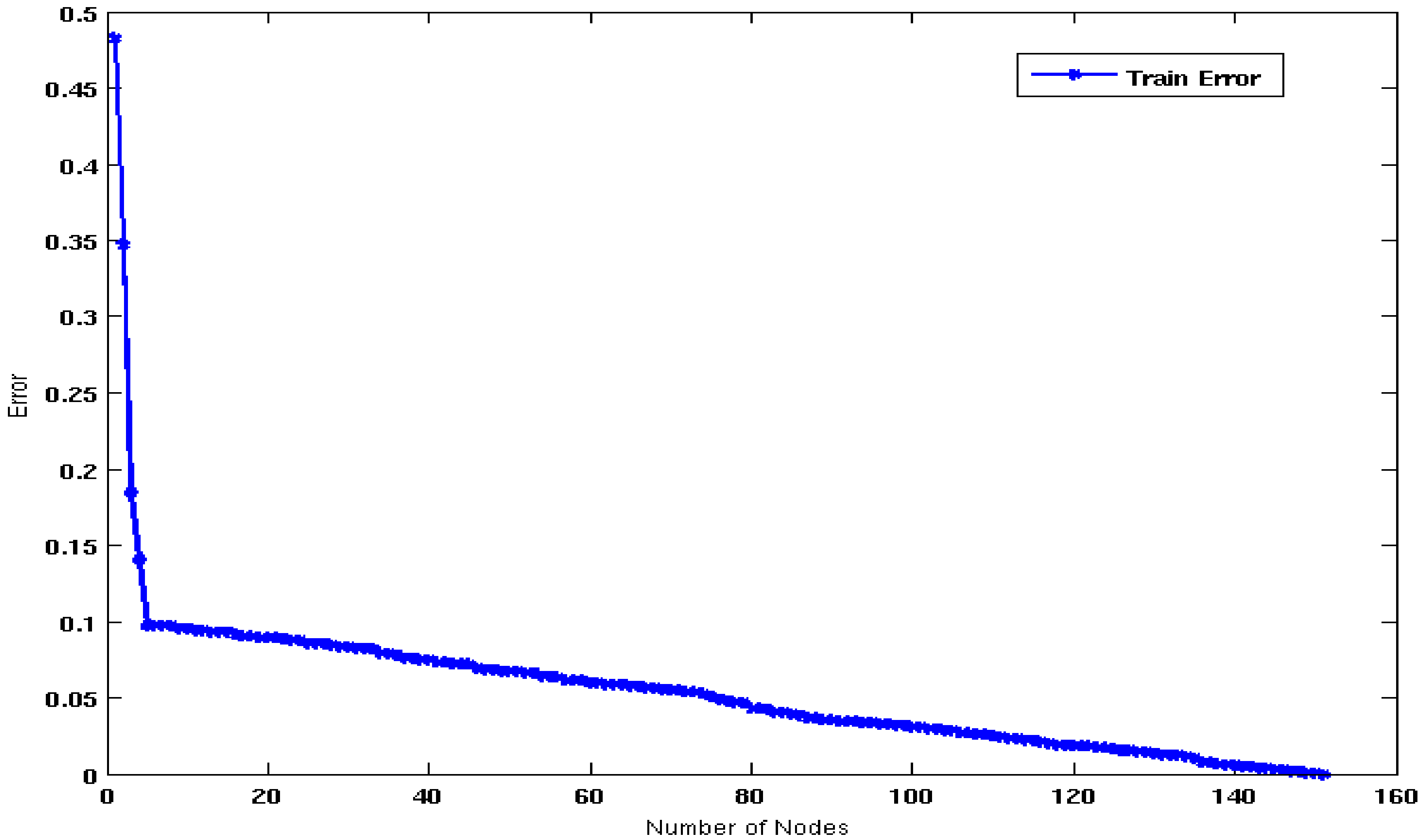


包含50个节点的决策树

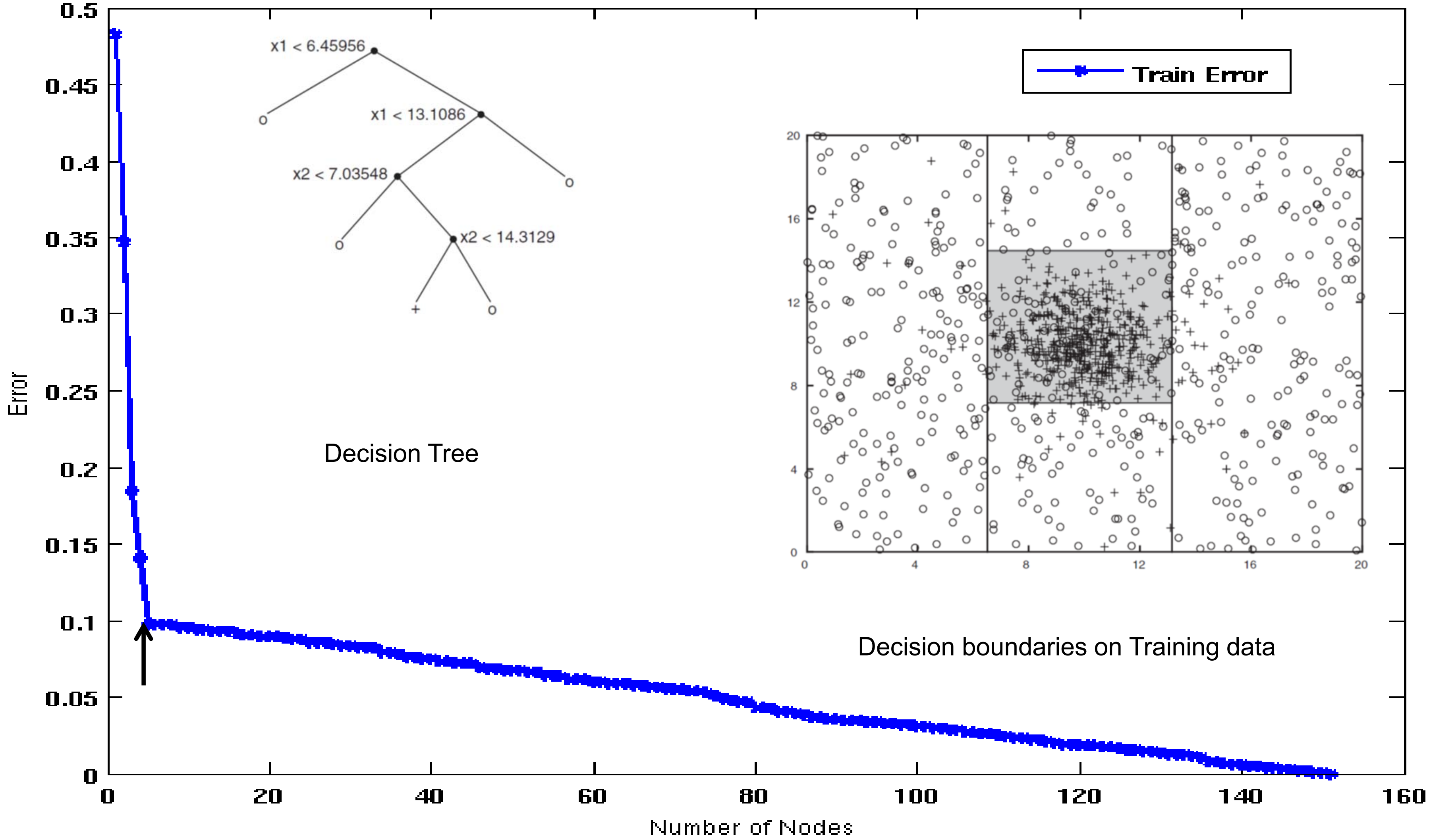
哪个决策树更好?

决策树

Increasing number of nodes in Decision Trees

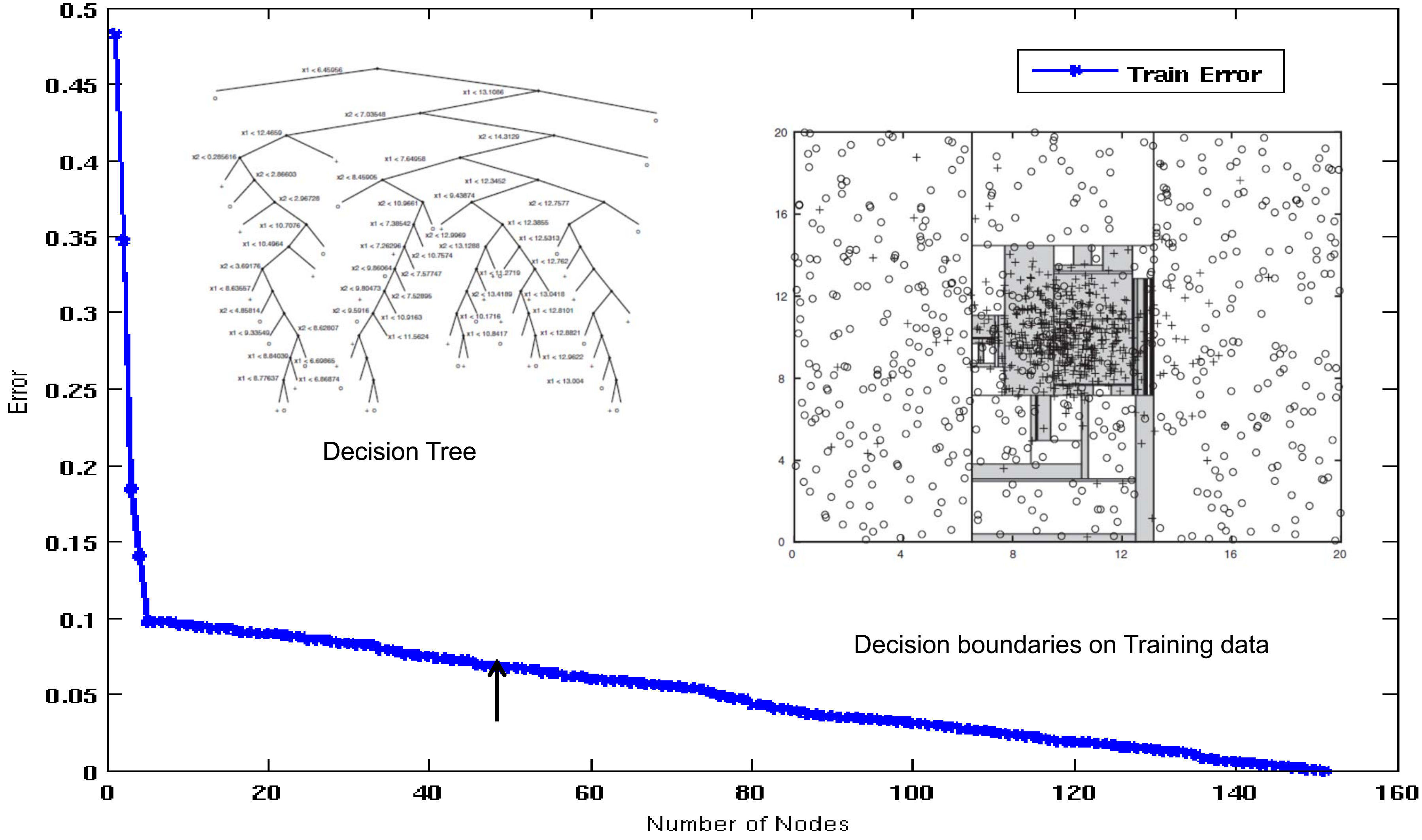


Decision Tree with 4 nodes



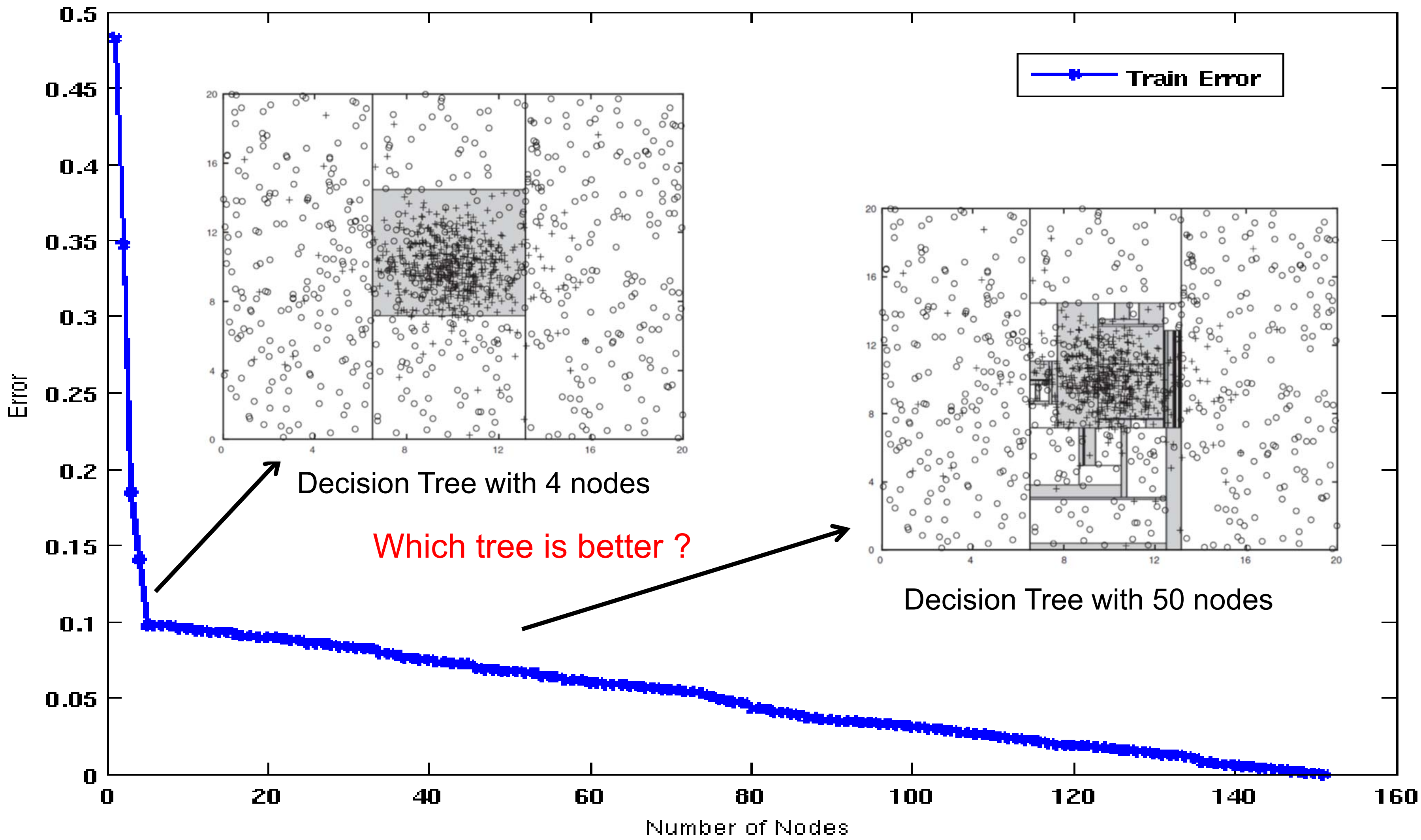
决策树

Decision Tree with 50 nodes

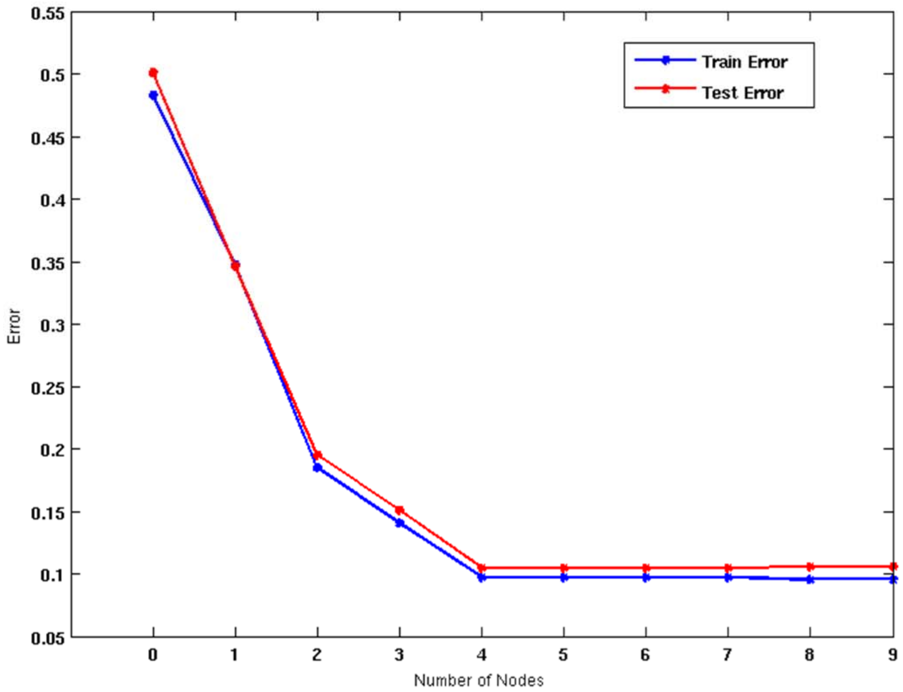


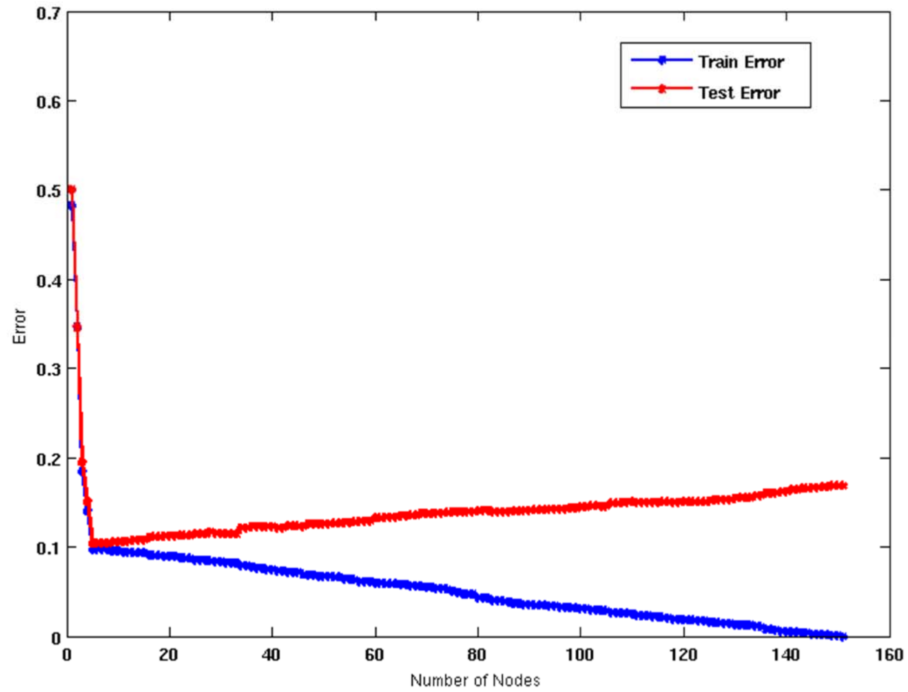
决策树的过拟合

哪个决策树更好?



欠拟合与过拟合



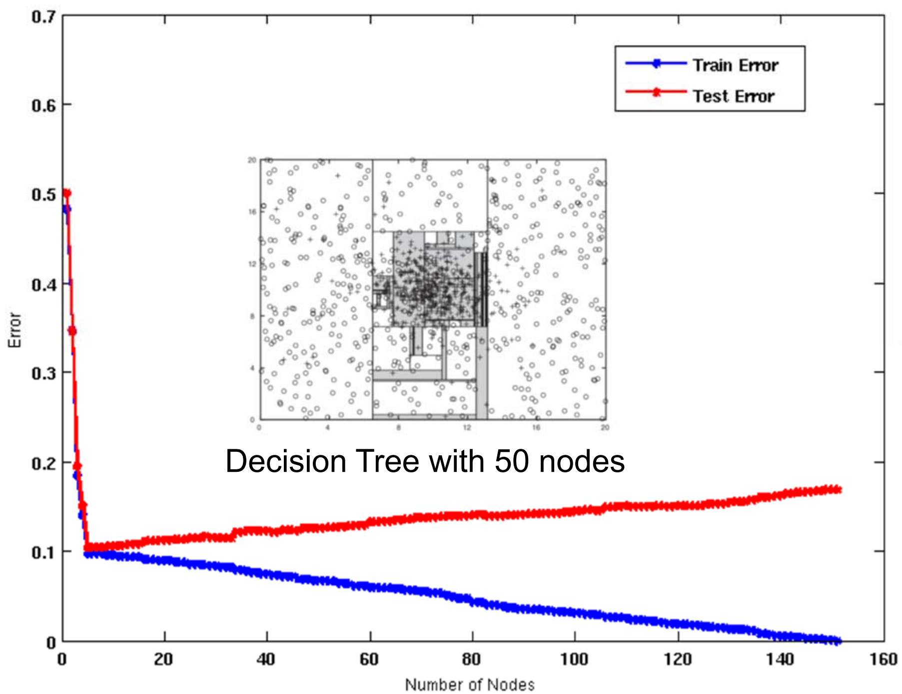


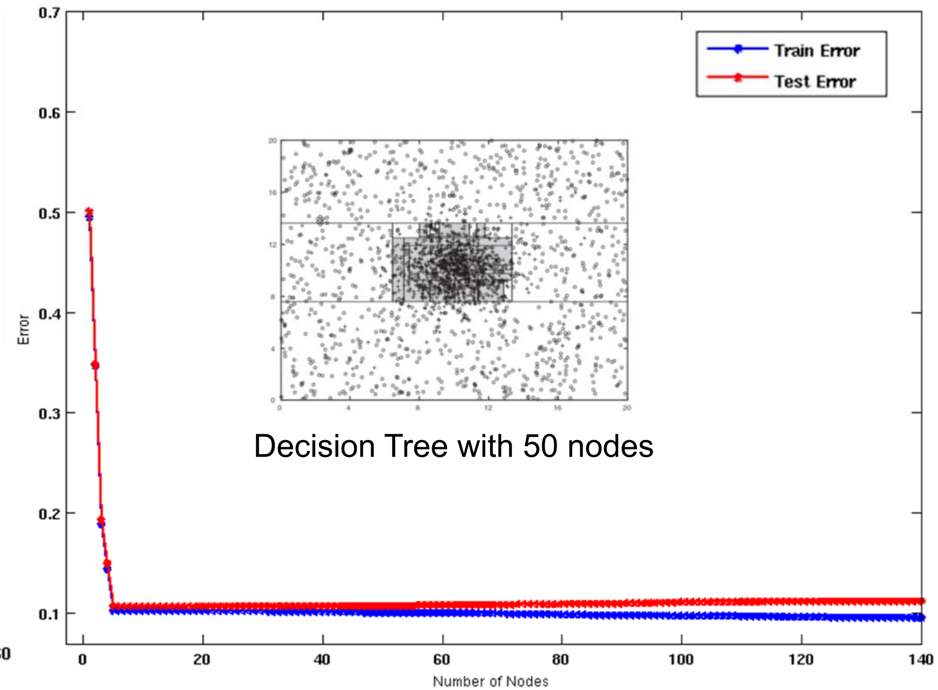
•随着模型变得越来越复杂，测试错误可能会开始增加，即使训练错误可能依然减少

Underfitting: 模型太简单, 训练集和测试集误差太大

Overfitting: 模型太复杂, 训练误差小但测试误差大

过拟合– 训练集大小的影响





Using twice the number of data instances

- 在给定的模型大小下，增加训练数据的大小可以减少训练和测试错误之间的差异

过拟合的原因

- 训练集不足
- 模型过于复杂
  - 多重比较过程 （Multiple Comparison Procedure）

多重比较过程

- 预测证券市场在未来10天的升降情况
- Random guessing:

  $P(correct)$ = 0.5

- 一个人在这10次能随机猜对至少8次的概率:

Day 1	Up
Day 2	Down
Day 3	Down
Day 4	Up
Day 5	Down
Day 6	Down
Day 7	Up
Day 8	Up
Day 9	Up
Day 10	Down

$$P(\#correct \geq 8) = \frac{\binom{10}{8} + \binom{10}{9} + \binom{10}{10}}{2^{10}} = 0.0547$$

多重比较过程

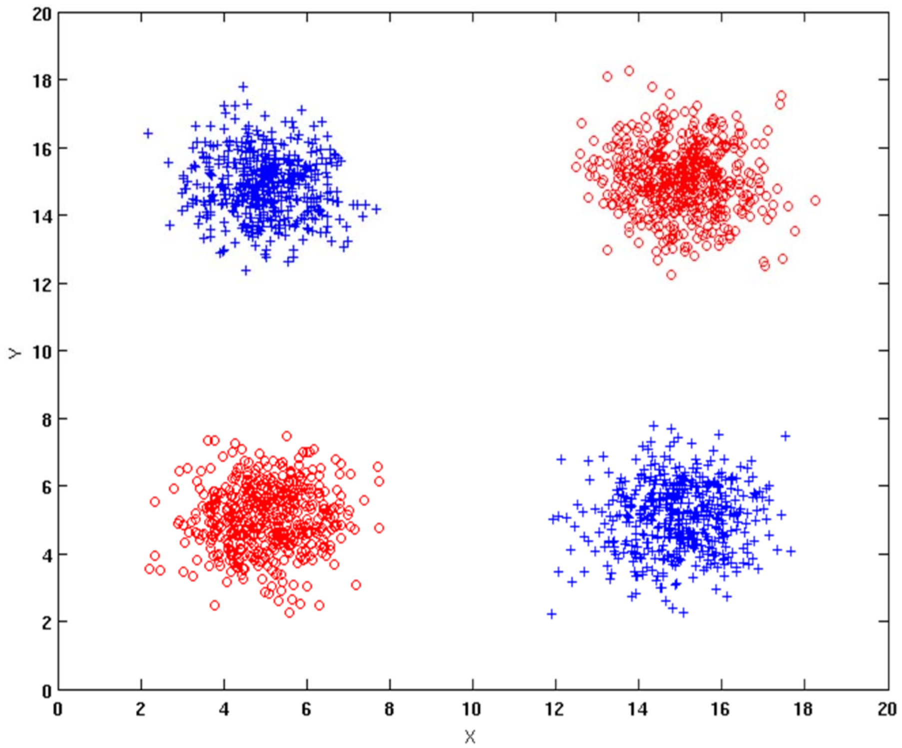
- 方法:
  - 有50位分析师
  - 每位分析师进行10次随机预测
  - 而后选择预测准确率最高的一位
- 至少一位分析师做出至少8次正确预测的概率

$$P(\#correct \geq 8) = 1-(1-0.0547)^{50} = 0.9399$$

多重比较过程的影响

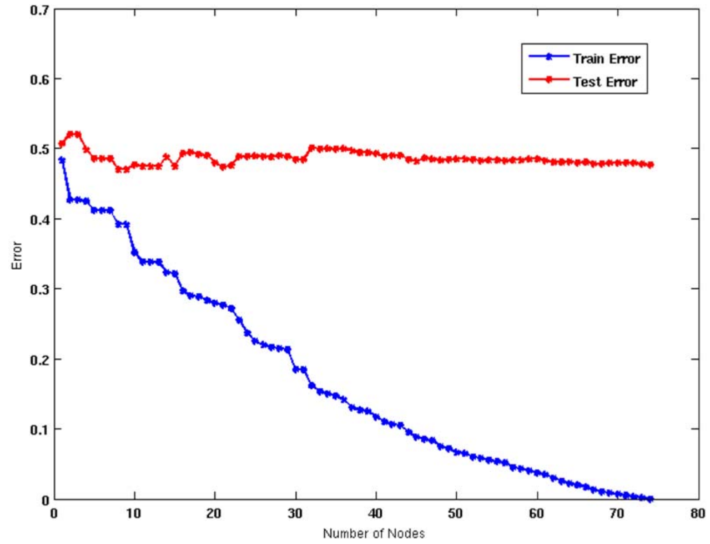
- 很多算法采用如下贪心测量（greedy strategy）：
  1. 初始化模型: M
  2. 替换模型: $M' = M \cup \gamma$， 其中 γ是要添加到模型中的组件(如：决策树中的一个分类条件)
  3. 如模型效果提升则继续更新 M', $\Delta(M,M') > \alpha$
- 通常, γ 是从一组可替换组件集合$\Gamma = \{\gamma_1, \gamma_2, \ldots, \gamma_k\}$中选择的；
- 如果有许多可供选择的方法，算法可能会不经意地向模型中添加不相关的组件，导致模型过拟合。

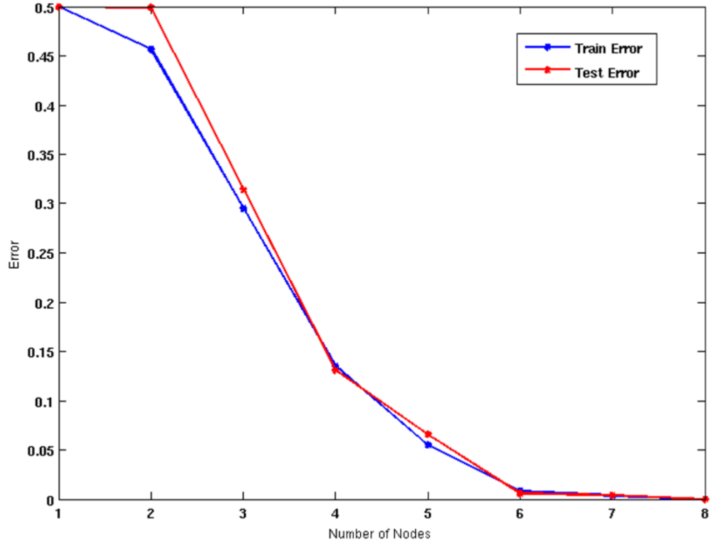
多重比较过程的影响



Use additional 100 noisy variables generated from a uniform distribution along with X and Y as attributes.

Use 30% of the data for training and 70% of the data for testing





Using only X and Y as attributes

过拟合

- 过拟合结果的决策树更加复杂
- 训练错误不能够用来评估模型在已有数据上的的好坏了
- 需要新的评估方法

使用验证集

- 将训练数据分成两部分:
  - 训练集（Training set）：
    - 用于建立模型
  - 验证集（Validation set）：
    - 用于估计泛化误差
    - 注意:验证集与测试集不同
- 缺点:
  - 可用于培训的数据更少

降低模型复杂性

- Rationale: Occam’s Razor（奥卡姆剃刀原理）
  - 给定两个泛化误差相似的模型，简单的模型优于复杂的模型
  - 复杂模型更可能被数据中的错误影响到
  - 因此，评估模型时要考虑其复杂性

Gen. Error(Model) = Train. Error(Model, Train. Data) +
α x Complexity(Model)

决策树的复杂度估计

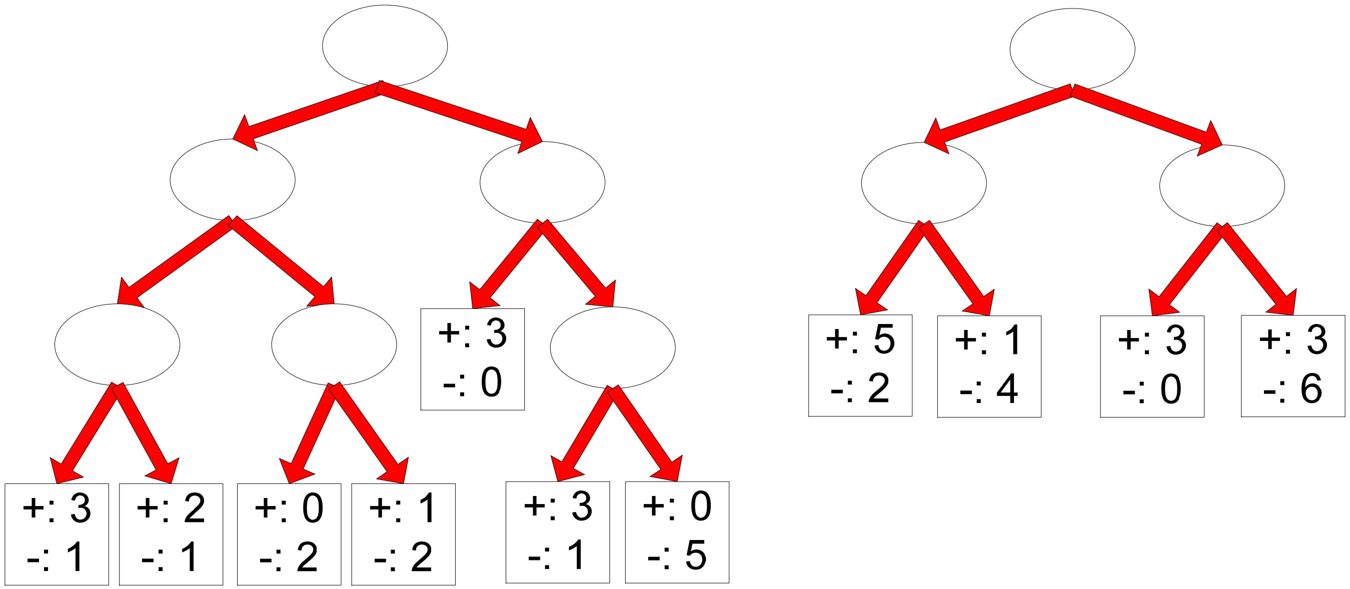
口悲观误差估计（Pessimistic Error Estimate）：

– 对于包含k个叶节点的决策树T的悲观误差估计为：

$$err_{gen}(T) = err(T) + \Omega \times \frac{k}{N_{train}}$$

– err(T): 训练集的误差率
– Ω: trade-off hyper-parameter (similar to α)
  • 添加叶节点的相对成本
– k: 叶节点个数
– N_{train}: 训练集中训练样本数

决策树的复杂度估计



Decision Tree, T_L

Decision Tree, T_R

$e(T_L) = 4/24$

$e(T_R) = 6/24$

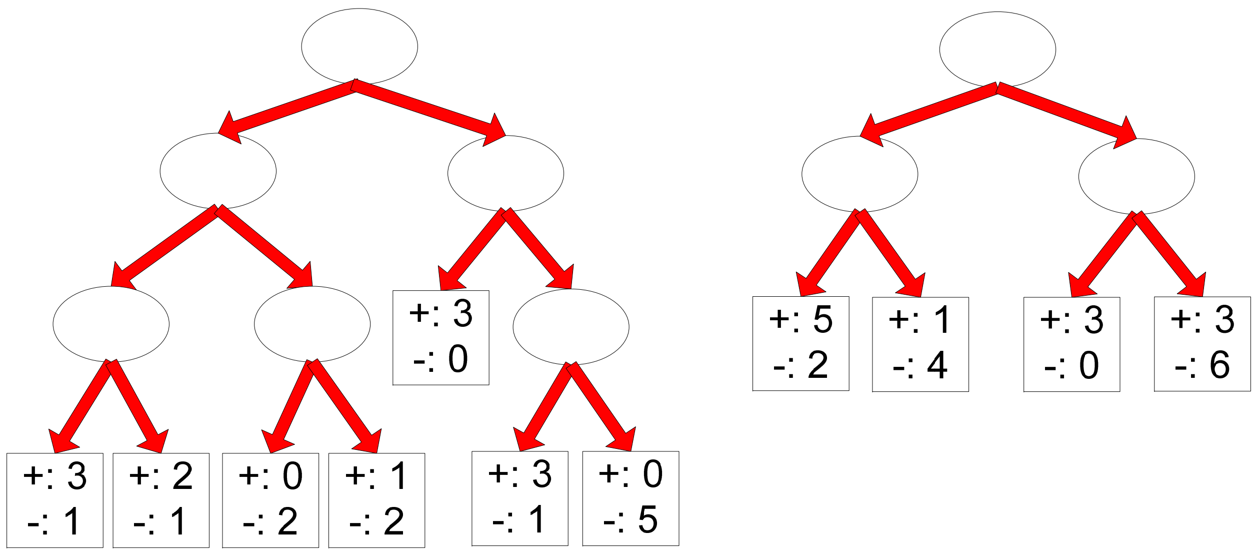
$\Omega = 1$

$e_{gen}(T_L) = 4/24 + 1*7/24 = 11/24 = 0.458$

$e_{gen}(T_R) = 6/24 + 1*4/24 = 10/24 = 0.417$

决策树的复杂度估计

- 再代入误差估计(Resubstitution Estimate):
  - 用训练误差作为泛化误差的乐观估计(optimistic estimate)
  - 又称为乐观误差估计(optimistic error estimate)



Decision Tree, T_L

Decision Tree, T_R

$e(T_L) = 4/24$

$e(T_R) = 6/24$

模型评估

- Metrics for Performance Evaluation
  - How to evaluate the performance of a model?
- Methods for Performance Evaluation
  - How to obtain reliable estimates?
- Methods for Model Comparison
  - How to compare the relative performance among competing models?

样本不均衡问题Class Imbalance Problem

- 许多分类问题，其中类别是倾斜的(来自一个类别的记录比另一个类别的记录多很多)
  - 信用卡欺诈 Credit card fraud
  - 入侵检测 Intrusion detection
  - 生产线缺陷产品检测Defective products in manufacturing assembly line
  - 随机抽取人群进行COVID-19新冠病毒检测结果

- **Key Challenge**:
  - 准确性等评价指标不适用于不平衡类别 （Evaluation measures such as accuracy are not well-suited for imbalanced class）

混淆矩阵

- 混淆矩阵 (Confusion Matrix):

	PREDICTED CLASS		
ACTUAL CLASS		Class=Yes	Class=No
	Class=Yes	a	b
	Class=No	c	d

a: TP (true positive)

b: FN (false negative)

c: FP (false positive)

d: TN (true negative)

准确率 Accuracy

	PREDICTED CLASS	
		Class=Yes	Class=No
ACTUAL CLASS	Class=Yes	a (TP)	b (FN)
	Class=No	c (FP)	d (TN)

- Most widely-used metric:

$$\text{Accuracy} = \frac{a+d}{a+b+c+d} = \frac{TP+TN}{TP+TN+FP+FN}$$

样本不均衡

- 二分类问题
  - 分类为No的样本数= 990；分类为Yes的样本数= 10
- 如果一个模型预测的一切都是NO级的，那么准确性就是990/1000 = 99 %
  - 这是误导，因为这个简单的模型没有检测任何类YES示例
  - 在实际应用中准确的预测少数样本往往更为重要

		PREDICTED CLASS	
		Class=Yes	Class=No
ACTUAL CLASS	Class=Yes	0	10
	Class=No	0	990

样本不均衡

A

	PREDICTED	
ACTUAL	Class=Yes	Class=No
Class=Yes	0	10
Class=No	0	990

Accuracy: 99%

B

	PREDICTED	
ACTUAL	Class=Yes	Class=No
Class=Yes	10	0
Class=No	500	490

Accuracy: 50%

哪个模型更好?

样本不均衡

A

	PREDICTED	
ACTUAL	Class=Yes	Class=No
Class=Yes	5	5
Class=No	0	990

B

	PREDICTED	
ACTUAL	Class=Yes	Class=No
Class=Yes	10	0
Class=No	500	490

哪个模型更好?

其他评价标准

	PREDICTED CLASS		
		Class=Yes	Class=No
ACTUAL CLASS	Class=Yes	a	b
	Class=No	c	d

精确率： $$\text{Precision (p)} = \frac{a}{a+c}$$

召回率： $$\text{Recall (r)} = \frac{a}{a+b}$$

F值 ： $$\text{F - measure (F)} = \frac{2rp}{r+p} = \frac{2a}{2a+b+c}$$

其他评价标准

	PREDICTED CLASS	
ACTUAL CLASS	Class=Yes	Class=No
Class=Yes	10	0
Class=No	10	980

$$\text{Precision (p)} = \frac{10}{10+10} = 0.5$$

$$\text{Recall (r)} = \frac{10}{10+0} = 1$$

$$\text{F-measure (F)} = \frac{2*1*0.5}{1+0.5} = 0.62$$

$$\text{Accuracy} = \frac{990}{1000} = 0.99$$

	PREDICTED CLASS	
ACTUAL CLASS	Class=Yes	Class=No
Class=Yes	1	9
Class=No	0	990

$$\text{Precision (p)} = \frac{1}{1+0} = 1$$

$$\text{Recall (r)} = \frac{1}{1+9} = 0.1$$

$$\text{F-measure (F)} = \frac{2*0.1*1}{1+0.1} = 0.18$$

$$\text{Accuracy} = \frac{991}{1000} = 0.991$$

样本不均衡

A

	PREDICTED CLASS	
ACTUAL CLASS	Class=Yes	Class=No
Class=Yes	40	10
Class=No	10	40

$$\text{Precision (p)} = 0.8$$
$$\text{Recall (r)} = 0.8$$
$$\text{F - measure (F)} = 0.8$$
$$\text{Accuracy} = 0.8$$

B

	PREDICTED CLASS	
ACTUAL CLASS	Class=Yes	Class=No
Class=Yes	40	10
Class=No	1000	4000

$$\text{Precision (p)} =\sim 0.04$$
$$\text{Recall (r)} = 0.8$$
$$\text{F - measure (F)} =\sim 0.08$$
$$\text{Accuracy} =\sim 0.8$$

哪个模型更好?

分类算法性能评估指标总结

	PREDICTED CLASS	
	Yes	No
ACTUAL CLASS Yes	TP	FN
ACTUAL CLASS No	FP	TN

- α is the probability that we reject the null hypothesis when it is true. This is a Type I error or a false positive (FP).
- α 是当零假设为真时，我们接受它的概率。这是第一类错误或假阳性错误。

- β is the probability that we accept the null hypothesis when it is false. This is a Type II error or a false negative (FN).
- β 是当零假设为假时，我们接受它的概率。这是第二类错误或假阴性错误。

$$Accuracy = \frac{TP + TN}{TP + FN + FP + TN}$$

$$ErrorRate = 1 - accuracy$$

$$Precision = Positive\ Predictive\ Value = \frac{TP}{TP + FP}$$

$$Recall = Sensitivity = TP\ Rate = \frac{TP}{TP + FN}$$

$$Specificity = TN\ Rate = \frac{TN}{TN + FP}$$

$$FP\ Rate = \alpha = \frac{FP}{TN + FP} = 1 - specificity$$

$$FN\ Rate = \beta = \frac{FN}{FN + TP} = 1 - sensitivity$$

$$Power = sensitivity = 1 - \beta$$

其他评估指标计算

A	PREDICTED CLASS		
		Class=Yes	Class=No
ACTUAL CLASS	Class=Yes	40	10
	Class=No	10	40

$$\text{Precision (p)} = 0.8$$
$$\text{TPR} = \text{Recall (r)} = 0.8$$
$$\text{FPR} = 0.2$$
$$\text{F-measure (F)} = 0.8$$
$$\text{Accuracy} = 0.8$$

$$\frac{\text{TPR}}{\text{FPR}} = 4$$

B	PREDICTED CLASS		
		Class=Yes	Class=No
ACTUAL CLASS	Class=Yes	40	10
	Class=No	1000	4000

$$\text{Precision (p)} = 0.038$$
$$\text{TPR} = \text{Recall (r)} = 0.8$$
$$\text{FPR} = 0.2$$
$$\text{F-measure (F)} = 0.07$$
$$\text{Accuracy} = 0.8$$

$$\frac{\text{TPR}}{\text{FPR}} = 4$$

模型效果评价

A	PREDICTED CLASS		
ACTUAL CLASS		Class=Yes	Class=No
	Class=Yes	10	40
	Class=No	10	40

$$\text{Precision} \quad (p) = 0.5$$
$$\text{TPR} = \text{Recall} \quad (r) = 0.2$$
$$\text{FPR} = 0.2$$
$$\text{F} - \text{measure} = 0.28$$

B	PREDICTED CLASS		
ACTUAL CLASS		Class=Yes	Class=No
	Class=Yes	25	25
	Class=No	25	25

$$\text{Precision} \quad (p) = 0.5$$
$$\text{TPR} = \text{Recall} \quad (r) = 0.5$$
$$\text{FPR} = 0.5$$
$$\text{F} - \text{measure} = 0.5$$

C	PREDICTED CLASS		
ACTUAL CLASS		Class=Yes	Class=No
	Class=Yes	40	10
	Class=No	40	10

$$\text{Precision} \quad (p) = 0.5$$
$$\text{TPR} = \text{Recall} \quad (r) = 0.8$$
$$\text{FPR} = 0.8$$
$$\text{F} - \text{measure} = 0.61$$

ROC (Receiver Operating Characteristic)

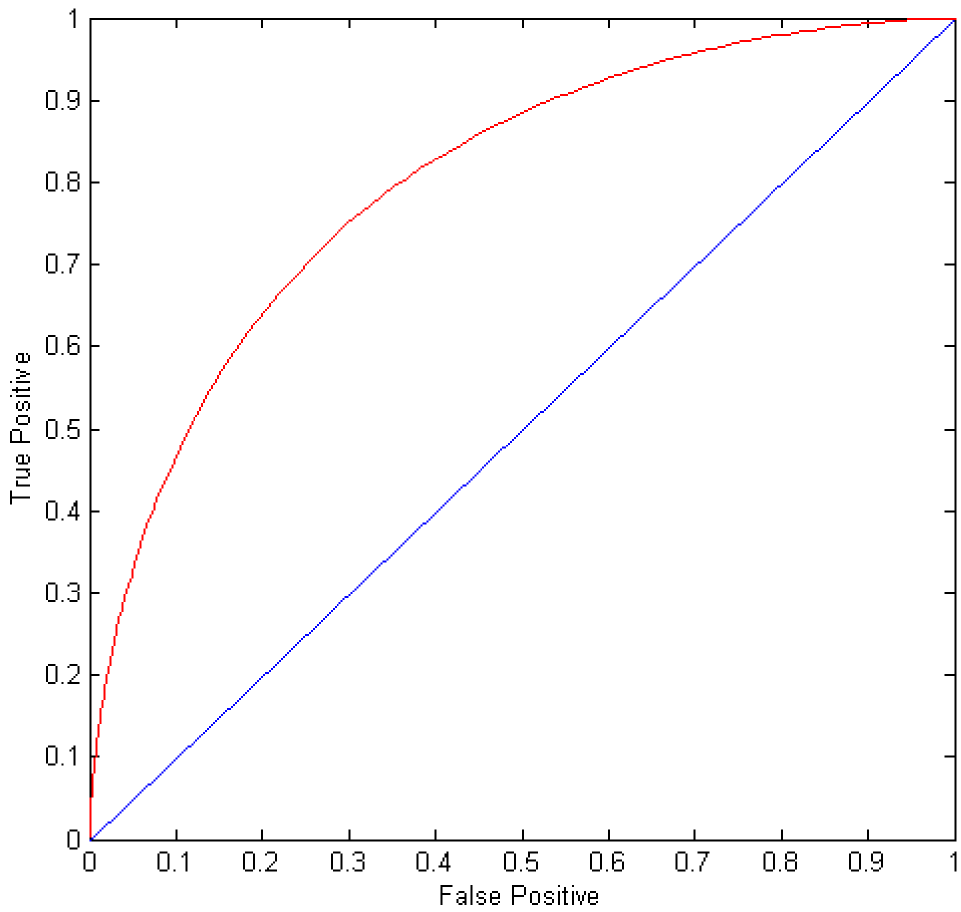
- 一种图形化方法，用于显示检测率(detection rate)和误报率(false alarm rate)之间的权衡
- 发展于20世纪50年代的信号检测理论，用于分析噪声信号
- ROC 曲线x轴是TPR，y轴是FPR.
  - 在ROC曲线的点表示的模型的性能

ROC曲线

(TPR,FPR):

- (0,0):声明一切为负类
- (1,1):声明一切为正类
- (1,0): 理想状态

- 对角线:
  - 随机猜测 Random guessing
  - 对角线下面 Below diagonal line:
    - 预测与真实类别相反



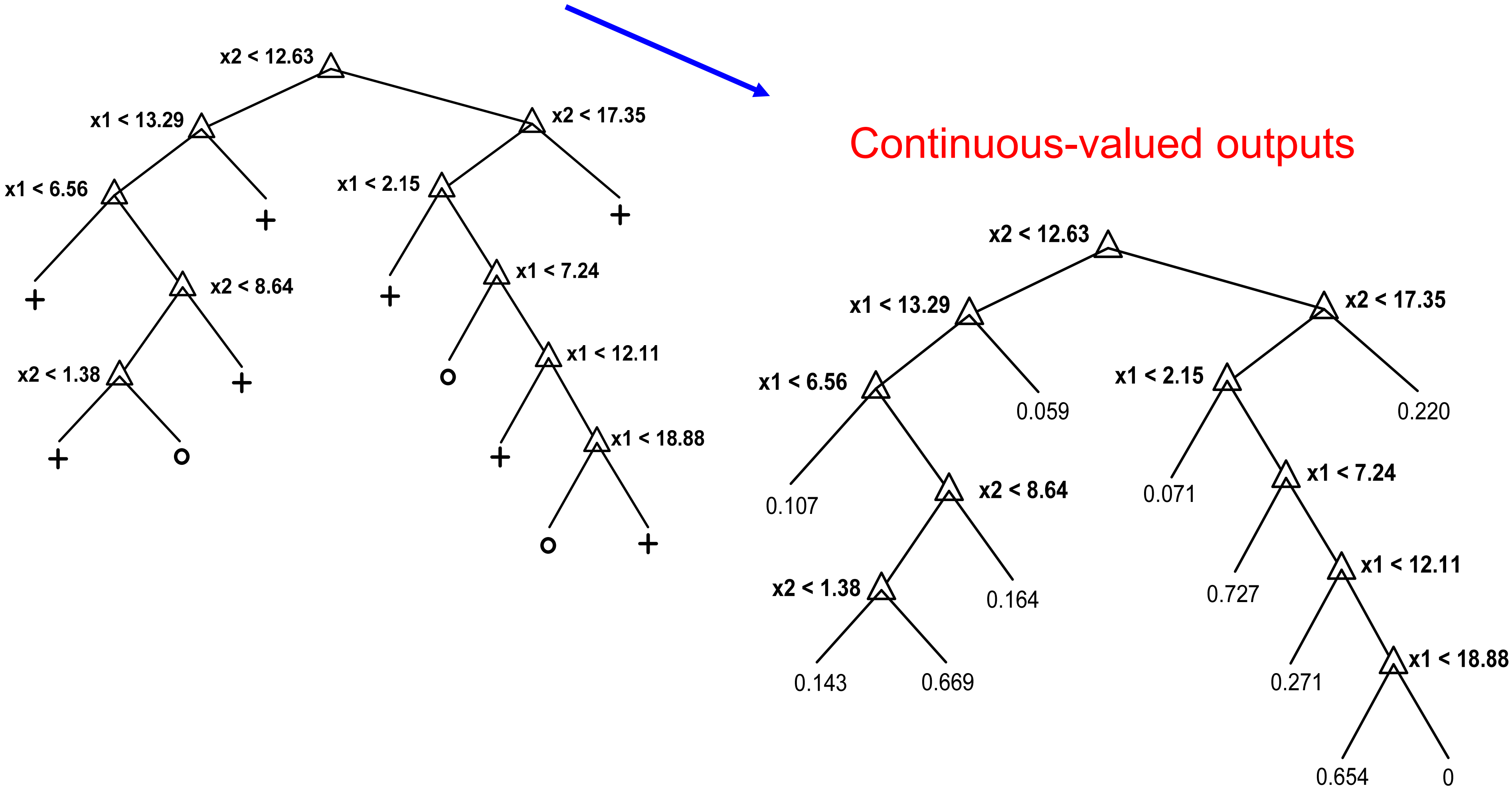
ROC曲线

- 为了绘制ROC曲线，分类器必须产生连续值输出
  - 输出用于对测试记录进行排序，从最有可能的正类记录到最不可能的正类记录
  - 通过对ROC曲线使用不同的阈值，我们可以创建具有TPR/FPR权衡的不同变体分类器
- 许多分类器只产生离散输出(例如，预测类)
  - 如何获得连续值输出?
    - Decision trees, rule-based classifiers, neural networks, Bayesian classifiers, k-nearest neighbors, SVM

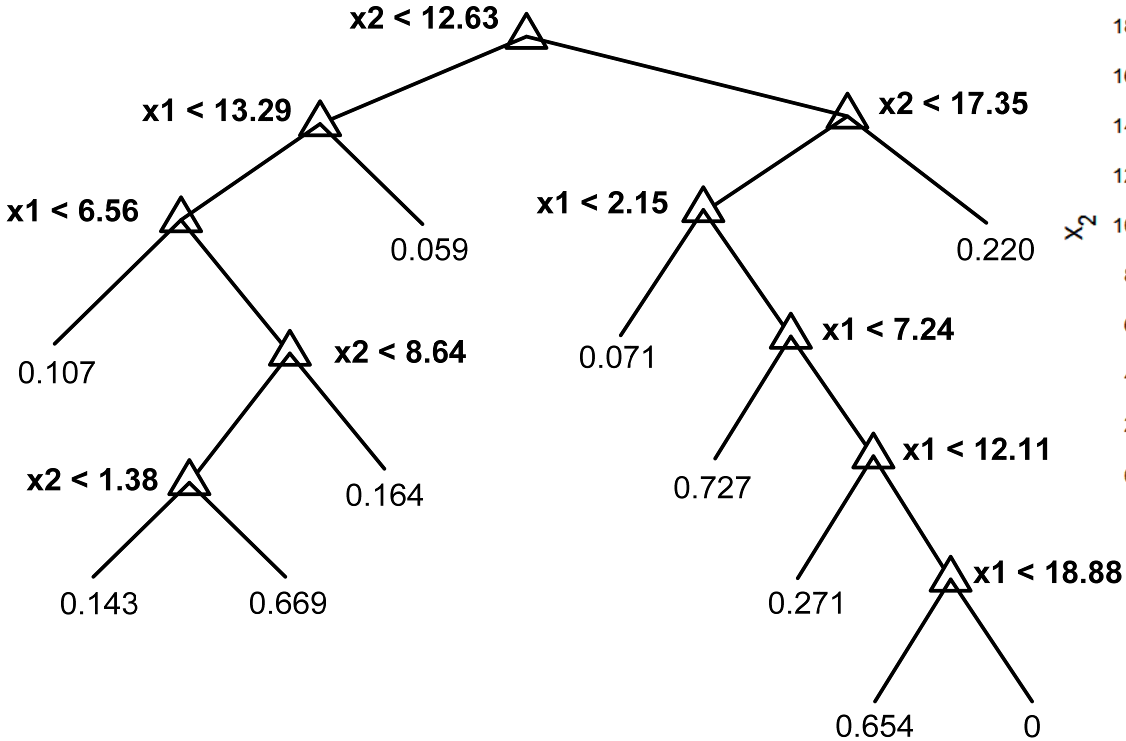
Example: Decision Trees

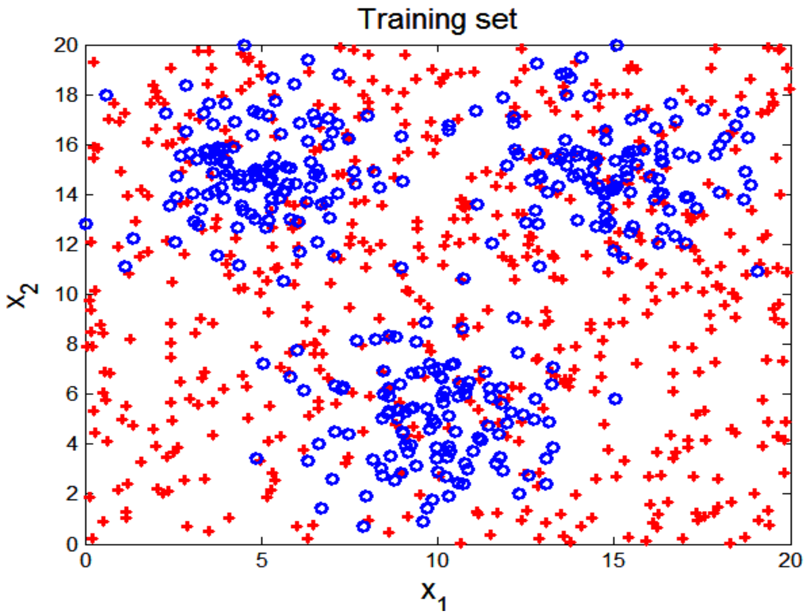
Decision Tree

Continuous-valued outputs



ROC曲线示例





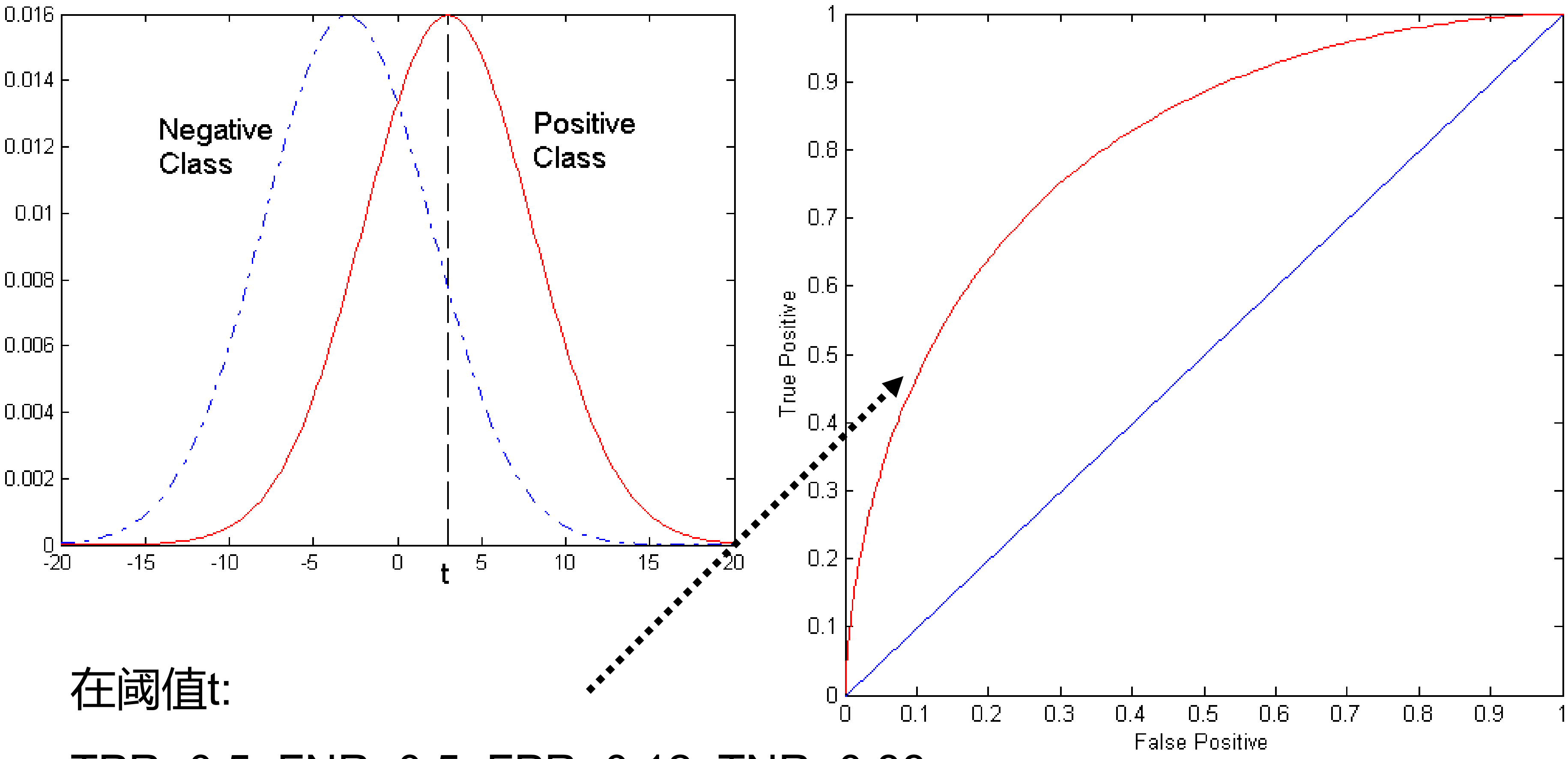
$\alpha = 0.3$		Predicted Class	
		Class o	Class +
Actual Class	Class o	645	209
	Class +	298	948

$\alpha = 0.7$		Predicted Class	
		Class o	Class +
Actual Class	Class o	181	673
	Class +	78	1168

ROC曲线示例

-包含两个类(正和负)的一维数据集

-位于x>t的任何点都被归为正



在阈值t:

TPR=0.5, FNR=0.5, FPR=0.12, TNR=0.88

如何构建ROC曲线

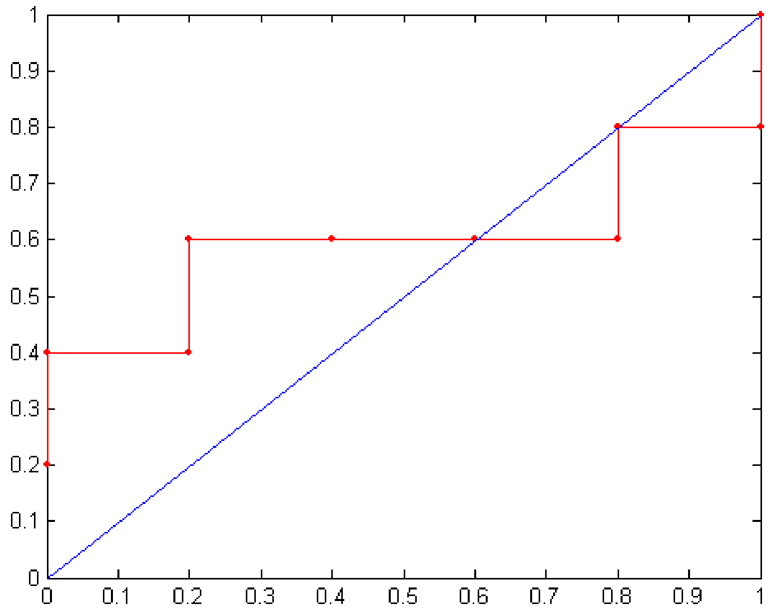
Instance	Score	True Class
1	0.95	+
2	0.93	+
3	0.87	-
4	0.85	-
5	0.85	-
6	0.85	+
7	0.76	-
8	0.53	+
9	0.43	-
10	0.25	+

- 使用分类器为每个实例生成连续值得分
  - 例如，越有可能是属于“+”类的，对应分数就越高
- 根据分数递减排序实例
- 在分数的每个唯一值上添加阈值
- 对于每个阈值进行TP, FP, TN, FN计数
  - TPR = TP/(TP+FN)
  - FPR = FP/(FP + TN)

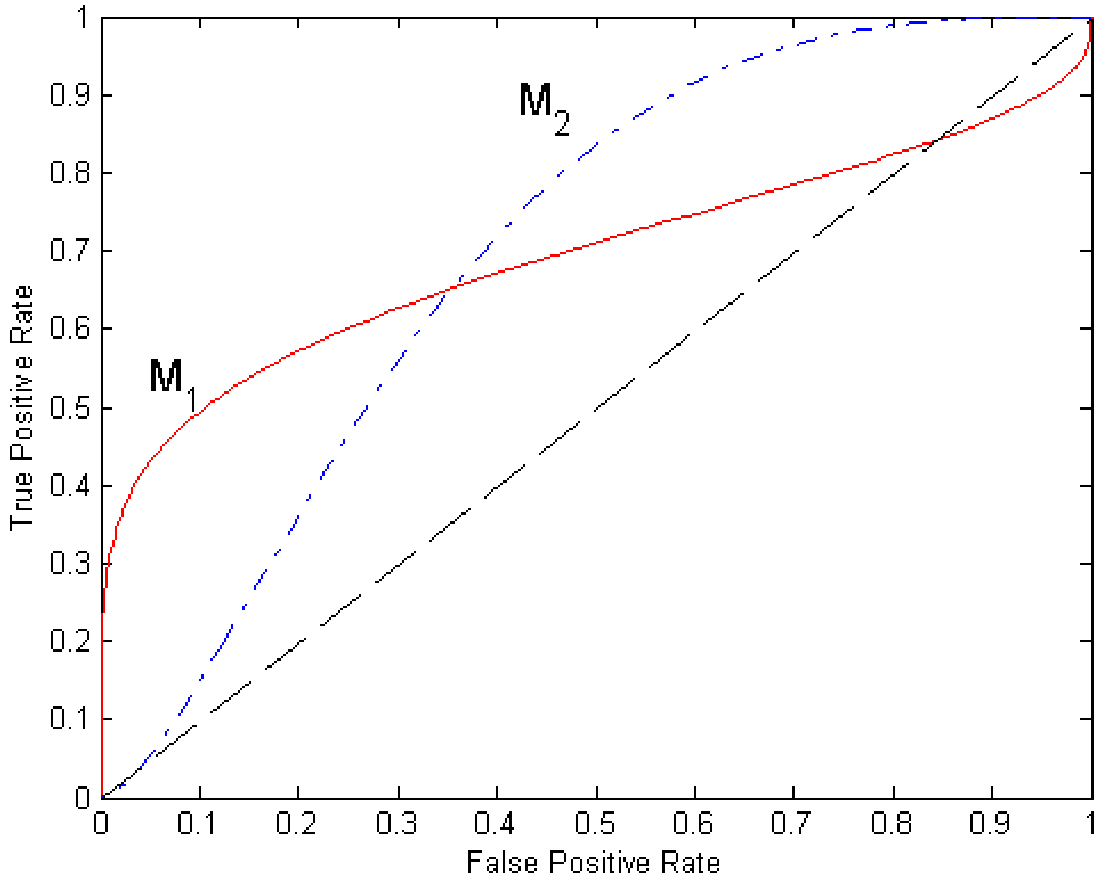
如何构建ROC曲线

Class	+	-	+	-	-	-	+	-	+	+	
Threshold >=	0.25	0.43	0.53	0.76	0.85	0.85	0.85	0.87	0.93	0.95	1.00
TP	5	4	4	3	3	3	3	2	2	1	0
FP	5	5	4	4	3	2	1	1	0	0	0
TN	0	0	1	1	2	3	4	4	5	5	5
FN	0	1	1	2	2	2	2	3	3	4	5
TPR	1	0.8	0.8	0.6	0.6	0.6	0.6	0.4	0.4	0.2	0
FPR	1	1	0.8	0.8	0.6	0.4	0.2	0.2	0	0	0

ROC Curve:



用ROC曲线比较模型



没有一种模式能永远优于其他模式

M_1 is better for small FPR

M_2 is better for large FPR

Area Under ROC curve (AUC)

Ideal:

- Area = 1

Random guess:

- Area = 0.5

处理数据不均衡问题 – 总结

- 有许多措施，但没有一个是在所有情况下都理想的
  - 随机分类器也可以在这些评价方法上取得很高的值
  - TPR/FPR提供了重要的信息，但在许多实际场景下，它本身可能还不够
  - 给定两个分类器，有时你可以看出其中一个严格优于另一个
    - C1严格由于C2：a. C1的TPR和FPR都优于C2; b. C1的TPR与C2相同但FPR优于C2；b. C1的FPR与C2相同但TPR优于C2。
  - 当C1严格优于C2时, C1的F值可以比C2差，当他们各自的评估数据集包含有各自不同的失衡情况时。
  - 分类器C1与C2的优劣，不仅仅取决于分类器本身，更取决于当前的场景(如：类失衡、TP vs FP的重要性、成本/时间权衡等多方面因素)。

模型效果评价

T1	PREDICTED CLASS		
		Class=Yes	Class=No
ACTUAL CLASS	Class=Yes	50	50
	Class=No	1	99

$$Precision\ (p) = 0.98$$
$$TPR = Recall\ (r) = 0.5$$
$$FPR = 0.01$$
$$TPR/FPR = 50$$
$$F-measure = 0.66$$

T2	PREDICTED CLASS		
		Class=Yes	Class=No
ACTUAL CLASS	Class=Yes	99	1
	Class=No	10	90

$$Precision\ (p) = 0.9$$
$$TPR = Recall\ (r) = 0.99$$
$$FPR = 0.1$$
$$TPR/FPR = 9.9$$
$$F-measure = 0.94$$

T3	PREDICTED CLASS		
		Class=Yes	Class=No
ACTUAL CLASS	Class=Yes	99	1
	Class=No	1	99

$$Precision\ (p) = 0.99$$
$$TPR = Recall\ (r) = 0.99$$
$$FPR = 0.01$$
$$TPR/FPR = 99$$
$$F-measure = 0.99$$

模型效果评价-Medium Skew case

T1	PREDICTED CLASS	
ACTUAL CLASS	Class=Yes	Class=No
Class=Yes	50	50
Class=No	10	990

$$\text{Precision} \quad (p) = 0.83$$
$$\text{TPR} = \text{Recall} \quad (r) = 0.5$$
$$\text{FPR} = 0.01$$
$$\text{TPR/FPR} = 50$$
$$\text{F}-\text{measure} = 0.62$$

T2	PREDICTED CLASS	
ACTUAL CLASS	Class=Yes	Class=No
Class=Yes	99	1
Class=No	100	900

$$\text{Precision} \quad (p) = 0.5$$
$$\text{TPR} = \text{Recall} \quad (r) = 0.99$$
$$\text{FPR} = 0.1$$
$$\text{TPR/FPR} = 9.9$$
$$\text{F}-\text{measure} = 0.66$$

T3	PREDICTED CLASS	
ACTUAL CLASS	Class=Yes	Class=No
Class=Yes	99	1
Class=No	10	990

$$\text{Precision} \quad (p) = 0.9$$
$$\text{TPR} = \text{Recall} \quad (r) = 0.99$$
$$\text{FPR} = 0.01$$
$$\text{TPR/FPR} = 99$$
$$\text{F}-\text{measure} = 0.94$$

模型效果评价-High Skew case

T1	PREDICTED CLASS		
		Class=Yes	Class=No
ACTUAL CLASS	Class=Yes	50	50
	Class=No	100	9900

$$\text{Precision } (p) = 0.3$$
$$\text{TPR} = \text{Recall } (r) = 0.5$$
$$\text{FPR} = 0.01$$
$$\text{TPR/FPR} = 50$$
$$\text{F} - \text{measure} = 0.375$$

T2	PREDICTED CLASS		
		Class=Yes	Class=No
ACTUAL CLASS	Class=Yes	99	1
	Class=No	1000	9000

$$\text{Precision } (p) = 0.09$$
$$\text{TPR} = \text{Recall } (r) = 0.99$$
$$\text{FPR} = 0.1$$
$$\text{TPR/FPR} = 9.9$$
$$\text{F} - \text{measure} = 0.165$$

T3	PREDICTED CLASS		
		Class=Yes	Class=No
ACTUAL CLASS	Class=Yes	99	1
	Class=No	100	9900

$$\text{Precision } (p) = 0.5$$
$$\text{TPR} = \text{Recall } (r) = 0.99$$
$$\text{FPR} = 0.01$$
$$\text{TPR/FPR} = 99$$
$$\text{F} - \text{measure} = 0.66$$

基于不均衡训练集构建模型

- 修改训练数据的分布，使罕见类在训练集中得到很好的表示
  - 对大多数人进行抽样调查 Undersample the majority class
  - 过度购买稀有类
  - Oversample the rare class