

# 决策树

---

-----BY YANGZHONGXIU 2020.02.07

# 内容

---

- 决策树方法概览
- 几个相关概念
- 剪枝处理
- 连续值处理
- 缺失值处理
- 应用实例

# 决策树方法概览

---

## 场景1：妈妈给女儿介绍对象

女儿：长得帅不帅？

妈妈：挺帅的

女儿：有没有房子？

妈妈：在老家有一个

女儿：收入高不高？

妈妈：还不错，年薪百万

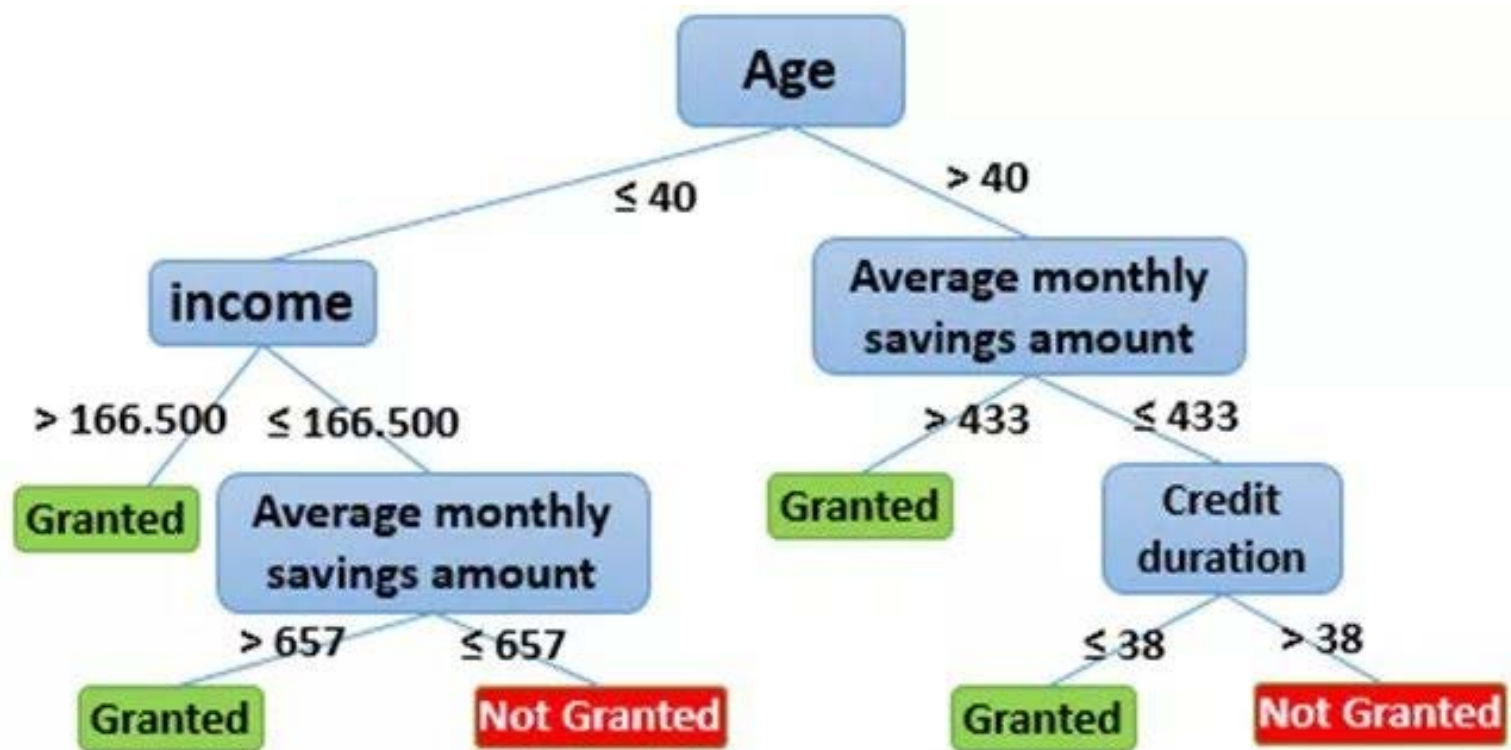
女儿：做什么工作的？

妈妈：IT男，互联网公司做数据挖掘

女儿：好，那我见见

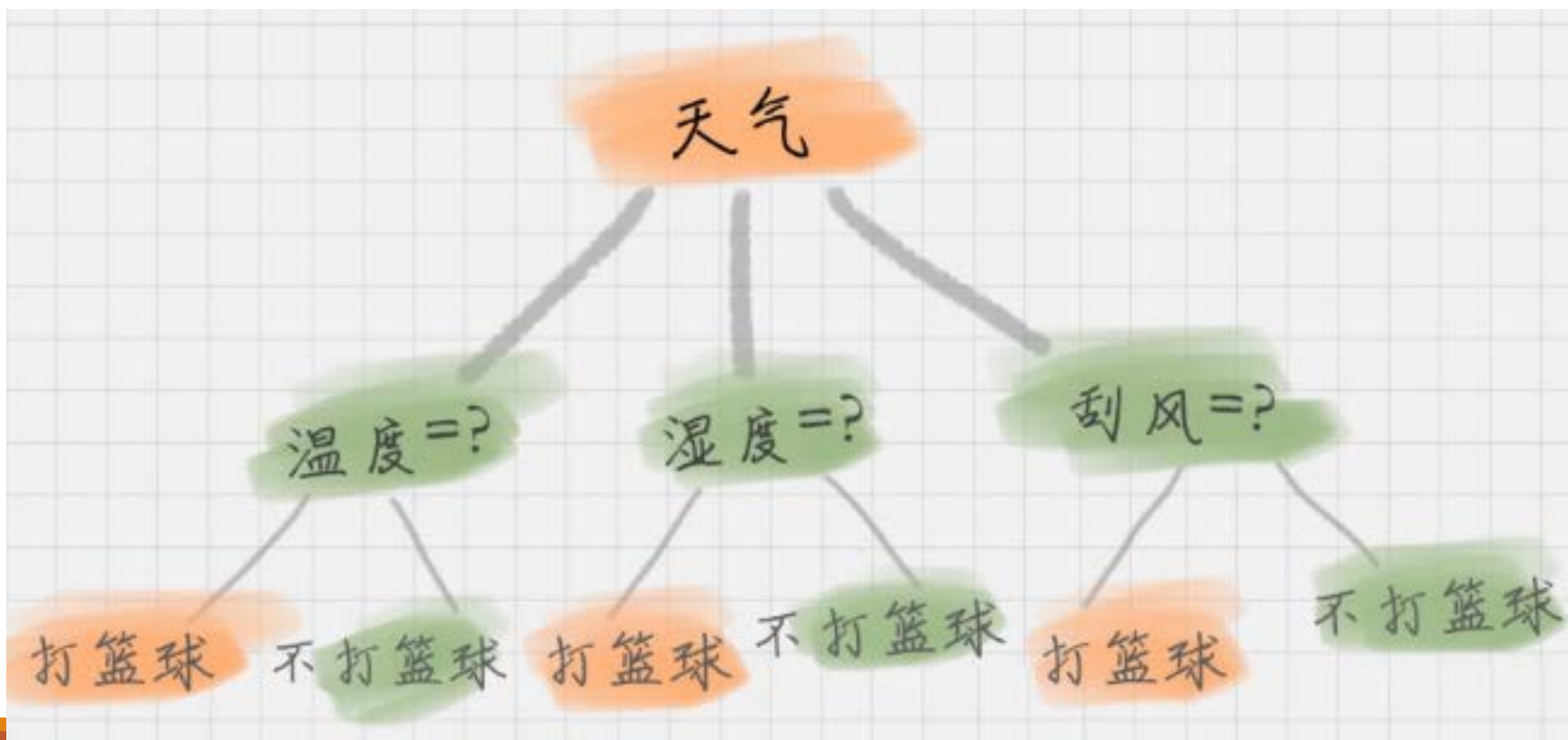
# 决策树方法概览

## 场景2：银行贷款



# 决策树方法概览

## 应用场景3：去不去打篮球



# 决策树方法概览

---

决策树方法在分类、预测、规则提取等领域有着广泛的应用。20世纪70年代后期和80年代初期，机器学习研究者**J.Ross Quinlan**提出了**ID3**算法以后，决策树在机器学习、数据挖掘领域取得极大的发展。Quinlan后来又提出了**C4.5**，成为新的监督学习算法。1984年，几位统计学家提出了**CART**分类算法。ID3和CART算法几乎同时被提出，但都是采用类似的方法从训练样本中学习决策树。

# 决策树方法概览

---

决策树是一种树状结构，它的每一个叶结点对应着一个分类，非叶结点对应着某个属性上的划分，根据样本在该属性上的不同取值将其划分为若干个子集。对于非纯的叶结点，多数类的标号给出到达这个结点的样本所属的类。

构造决策树的核心问题是每一步如何选择适当的属性对样本做拆分。对一个分类问题，从已知类标记的训练样本中学习并构造出决策书是一个自上而下，分而治之的过程

# 决策树方法概览

---

决策树方法分类：

**ID3算法：**其核心是在决策树的各级结点上，使用**信息增益**作为属性的选择标准，来帮助确定生成每个结点所采用的合适属性

**C4.5算法：**C4.5决策树算法相对于ID3算法的重要改进是使用**信息增益率**来选择结点属性。C4.5算法可以克服ID3算法存在的不足。ID3算法只适用于离散的描述属性，而C4.5算法及既能够处理离散的属性，也能够处理连续的属性

**CART算法：**CART决策树是一种十分有效的非参数分类和回归的方法，通过构建数、修剪树、评估树来构建一个二叉树。当终结点是连续变量时，该树为回归树；当终结点是离散变量，该树为分类树。



# 几个相关概念

---

➤ 构造

➤ 剪枝

决策树创建过程

➤ 过拟合

➤ 欠拟合

效果评估

➤ 纯度和信息熵

➤ 信息增益

➤ 信息增益率

➤ 基尼 (GINI) 系数

数据集的纯度

# 几个相关概念---构造

---

构造就是生成一棵完整的决策树。

构造的过程就是选择什么属性作为结点的过程。

构造过程中存在**三种结点**：

**根结点**：树最顶端的结点。

**内部结点**：树中间的结点，其实就是属性。

**叶结点**：树最底部的结点，决策结果

# 几个相关概念---构造

---

构造过程中，需要解决三个问题：

- 1.选择哪个属性作为根结点
- 2.选择哪些属性作为子结点
- 3.什么时候停止并得到目标状态，即叶结点。

# 几个相关概念---剪枝

---

**剪枝**就是给决策树瘦身，这一步实现的目标就是不需要太多的判断，同样可以得到不错的结果。

剪枝的目的是为了防止“过拟合”。

剪枝的方法包括：预剪枝（**Pre-Pruning**）和“后剪枝”（**Post-Pruning**）

# 几个相关概念---剪枝

---

**预剪枝**是在决策树构造时就进行剪枝。方法是在构造过程中对结点进行评估，如果对某个结点进行划分，在验证集中不能带来准确性的提升，那么对这个结点进行划分就没有意义，这时就会把当前结点作为叶结点，不对其进行划分。

**后剪枝**就是在生成决策树之后进行剪枝，通常会从决策树的叶结点开始，逐层向上面对每个结点进行评估。如果剪掉这个结点子树，与保留该结点子树在分类准确性上差别不大，或者剪掉该结点子树，能在验证集中带来准确性的提升，那么就把该结点子树进行剪枝。方法是：用这个结点子树的叶子结点来代替该结点，类标记为这个结点子树中最频繁的那个类。

# 几个相关概念---过拟合与欠拟合

---

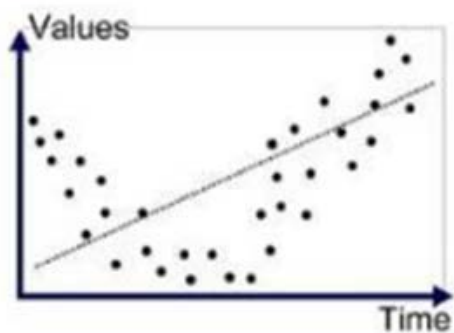
**过拟合**，模型训练结果“太好了”，以至于在实际应用过程中，会存在“死板”的情况，导致分类错误。特征学习太过，把个体特征当整体特征。

**欠拟合**是指模型拟合程度不高，数据距离拟合曲线较远，或指模型没有很好地捕捉到数据特征，不能够很好地拟合数据。特征学习不够充分。

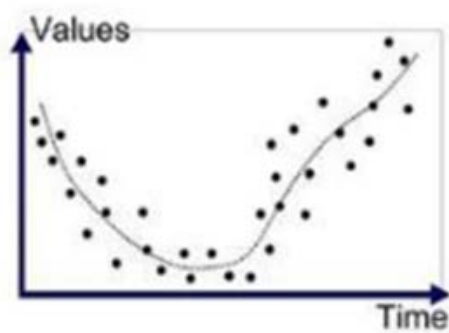
造成过拟合的原因之一就是因为训练集中样本量较小。如果决策树选择的属性过多，构造出来的决策树一定能够“完美”地把训练集中样本分类。但这样就会把训练集中一些数据的特点当成所有数据的特点，但这个忒但不一定是全部数据的特点，这就使得这个决策树在真实的数据分类中出现错误，也就是模型的“泛化能力”差。

**过拟合无法彻底避免，只能做到“缓解”。**

# 几个相关概念---过拟合与欠拟合



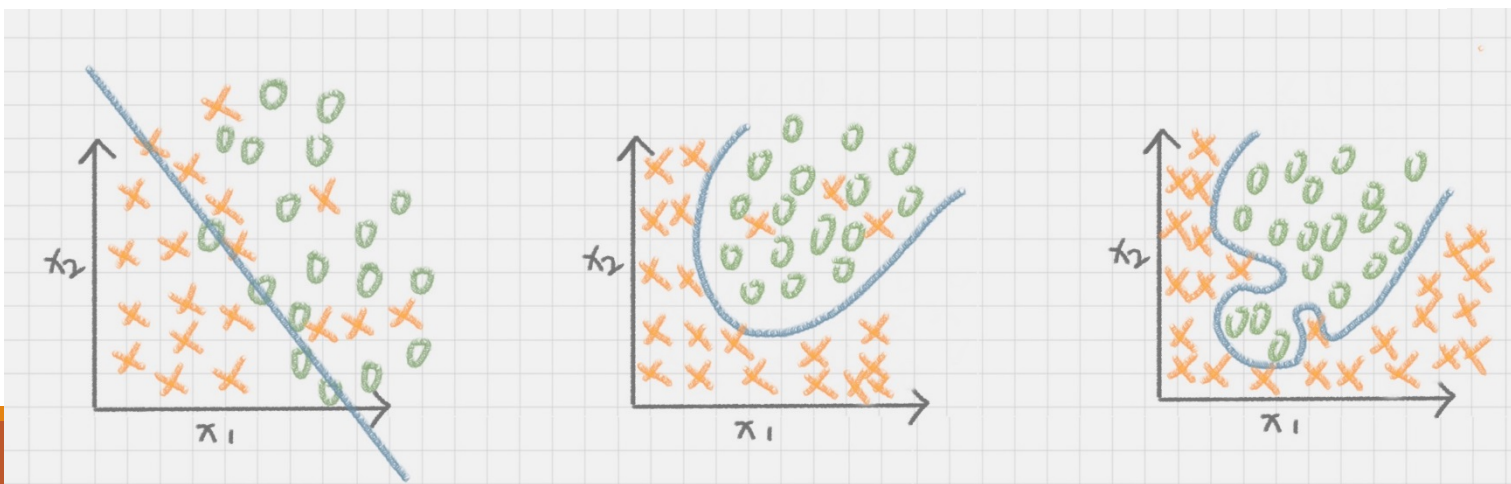
Underfitted



Good Fit/Robust



Overfitted



树叶训练样本



新样本



过拟合模型分类结果:  
→ 不是树叶  
(误以为树叶必须有锯齿)

欠拟合模型分类结果:  
→ 是树叶  
(误以为绿色的都是树叶)



# 几个相关概念---泛化能力

---

**泛化能力**（generalization ability）是指机器学习算法对新样本的适应能力。学习的目的是学到隐含在数据背后的规律，对具有同一规律的学习集以外的数据，经过训练的网络也能给出合适的输出，该能力称为泛化能力。通常期望经训练样本训练的网络具有较强的泛化能力，也就是对新输入给出合理响应的能力。应当指出并非训练的次数越多越能得到正确的输入输出映射关系。网络的性能主要用它的泛化能力来衡量。

# 几个相关概念---纯度和信息熵

---

决策树划分过程，希望决策树的分支结点所包含的样本尽可能属于同一类别，即结点的“纯度（**purity**）”越来越高。

**信息熵（information entropy）**是度量样本集合纯度最常用的一种指标。假设当前样本集合D中第k类样本所占的比例为 $p_k(k=1,2,3,...,|y|)$ ，则D的信息熵定义为：

$$\text{Ent}(D) = -\sum_{k=1}^{|y|} p_k \log_2 p_k$$

**Ent(D)**的值越小，则D的纯度越高。

# 几个相关概念---信息增益

假定离散属性 $a$ 有 $V$ 个可能的取值 $\{a^1, a^2, \dots, a^V\}$ , 若使用 $a$ 来对样本集 $D$ 进行划分, 则会产生 $V$ 各分支点, 其中第 $v$ 个分支结点包含了 $D$ 中所有在属性 $a$ 上取值为 $a^v$ 的样本, 记为 $D^v$ 。可根据公式计算出 $D^v$ 的信息熵, 考虑到不同分支结点所含样本数的不同, 给分支结点赋予权重 $|D^v|/|D|$ , 即样本数越多的分支结点的影响越大, 可计算出用属性 $a$ 对样本集 $D$ 进行划分所得的“**信息增益 (information gain)**”。

$$\text{Gain}(D, a) = \text{Ent}(D) - \sum_{v=1}^V \frac{|D^v|}{|D|} \text{Ent}(D^v)$$

一般而言, **信息增益越大, 则意味着使用属性 $a$ 来进行划分所获得的“纯度提升”越大。**

因此, 通常使用信息增益来作为决策树的划分属性选择。ID3 便是使用信息增益作为决策树划分属性选择。

# 几个相关概念---信息增益

编号	色泽	根蒂	敲声	纹理	脐部	触感	好瓜
1	青绿	蜷缩	浊响	清晰	凹陷	硬滑	是
2	乌黑	蜷缩	沉闷	清晰	凹陷	硬滑	是
3	乌黑	蜷缩	浊响	清晰	凹陷	硬滑	是
4	青绿	蜷缩	沉闷	清晰	凹陷	硬滑	是
5	浅白	蜷缩	浊响	清晰	凹陷	硬滑	是
6	青绿	稍蜷	浊响	清晰	稍凹	软粘	是
7	乌黑	稍蜷	浊响	稍糊	稍凹	软粘	是
8	乌黑	稍蜷	浊响	清晰	稍凹	硬滑	是
9	乌黑	稍蜷	沉闷	稍糊	稍凹	硬滑	否
10	青绿	硬挺	清脆	清晰	平坦	软粘	否
11	浅白	硬挺	清脆	模糊	平坦	硬滑	否
12	浅白	蜷缩	浊响	模糊	平坦	软粘	否
13	青绿	稍蜷	浊响	稍糊	凹陷	硬滑	否
14	浅白	稍蜷	沉闷	稍糊	凹陷	硬滑	否
15	乌黑	稍蜷	浊响	清晰	稍凹	软粘	否
16	浅白	蜷缩	浊响	模糊	平坦	硬滑	否
17	青绿	蜷缩	沉闷	稍糊	稍凹	硬滑	否

# 几个相关概念---信息增益

---

西瓜集数据集，该数据集包含17个样本，用来学习一棵能预测没有剖开的瓜是不是好瓜的决策树。 $|y|=2$ 。D包含所有样例。其中 $p_1=8/17$ ， $p_2=9/17$ 。

根结点的信息熵

$$\text{Ent}(D) = -\sum_{k=1}^2 p_k \log_2 p_k = -\left(\frac{8}{17} \log_2 \frac{8}{17} + \frac{9}{17} \log_2 \frac{9}{17}\right) = 0.998$$

然后，计算出当前属性集合{色泽，根蒂，敲声，纹理，脐部，触感}中每个属性的增益。

以属性“色泽”为例：

“色泽”有三个可能的取值{青绿，乌黑，浅白}，则可得三个子集，分别记为：D1(色泽=青绿)，D2(色泽=乌黑)，D3(色泽=浅白)

# 几个相关概念---信息增益

---

以属性“色泽”为例：

“色泽”有三个可能的取值{青绿，乌黑，浅白}，则可得三个子集，分别记为：D1(色泽=青绿)，D2(色泽=乌黑)，D3(色泽=浅白)

D1包含编号为{1,4,6,10,13,17}的6个样例，其中正比例为 $p1=3/6$ ，反比例为 $p2=3/6$ 。

D2包含编号为{2,3,7,8,9,15}的6个样例，其中正比例为 $p1=4/6$ ，反比例为 $p2=2/6$ 。

D3包含编号为{5,11,12,14,16}的5个样例，其中正比例为 $p1=1/5$ ，反比例为 $p2=4/5$ 。

# 几个相关概念---信息增益

---

可计算出D1,D2,D3三个分支的信息熵为:

$$\text{Ent}(D1) = - \left( \frac{3}{6} \log_2 \frac{3}{6} + \frac{3}{6} \log_2 \frac{3}{6} \right) = 1.000$$

$$\text{Ent}(D2) = - \left( \frac{4}{6} \log_2 \frac{4}{6} + \frac{2}{6} \log_2 \frac{2}{6} \right) = 0.918$$

$$\text{Ent}(D3) = - \left( \frac{1}{5} \log_2 \frac{1}{5} + \frac{4}{5} \log_2 \frac{4}{5} \right) = 0.722$$

根据信息增益公式, 可计算出“色泽”信息增益为:

$$\begin{aligned} \text{Gain}(D, \text{色泽}) &= \text{Ent}(D) - \sum_{v=1}^V \frac{|D_v|}{|D|} \text{Ent}(D_v) \\ &= 0.998 - \left( \frac{6}{17} \times 1.000 + \frac{6}{17} \times 0.918 + \frac{5}{17} \times 0.722 \right) \\ &= 0.109 \end{aligned}$$

# 几个相关概念---信息增益

---

类似，可计算出其它属性信息增益：

$$\text{Gain}(D, \text{根蒂}) = 0.143$$

$$\text{Gain}(D, \text{敲声}) = 0.141$$

$$\text{Gain}(D, \text{纹理}) = 0.381$$

$$\text{Gain}(D, \text{脐部}) = 0.289$$

$$\text{Gain}(D, \text{触感}) = 0.006$$

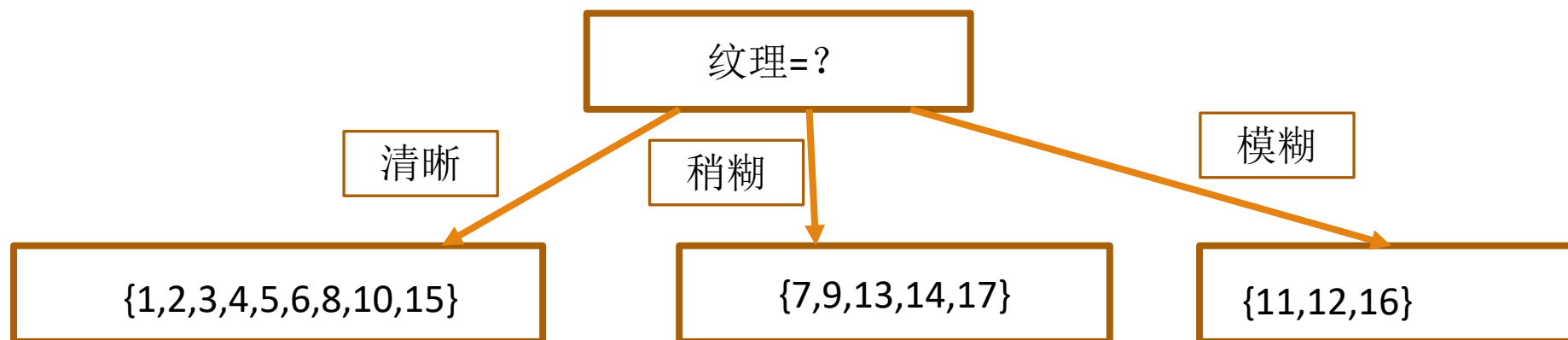
$$\text{Gain}(D, \text{色泽}) = 0.109$$

显然，属性“纹理”信息增益最大，于是它被选为划分属性。



# 几个相关概念---信息增益

---



然后，决策树学习算法将对每个分支结点作进一步划分，以上图第一分支结点(纹理=“清晰”)为例，该结点包含的样例集合D1中编号为{1,2,3,4,5,6,8,10,15}的9个样例，可用属性集合为{色泽，根蒂，敲声，脐部，触感}，基于D1的信息增益为：

# 几个相关概念---信息增益

---

类似，可计算出其它属性信息增益：

$$\text{Gain}(D1, \text{色泽}) = 0.043$$

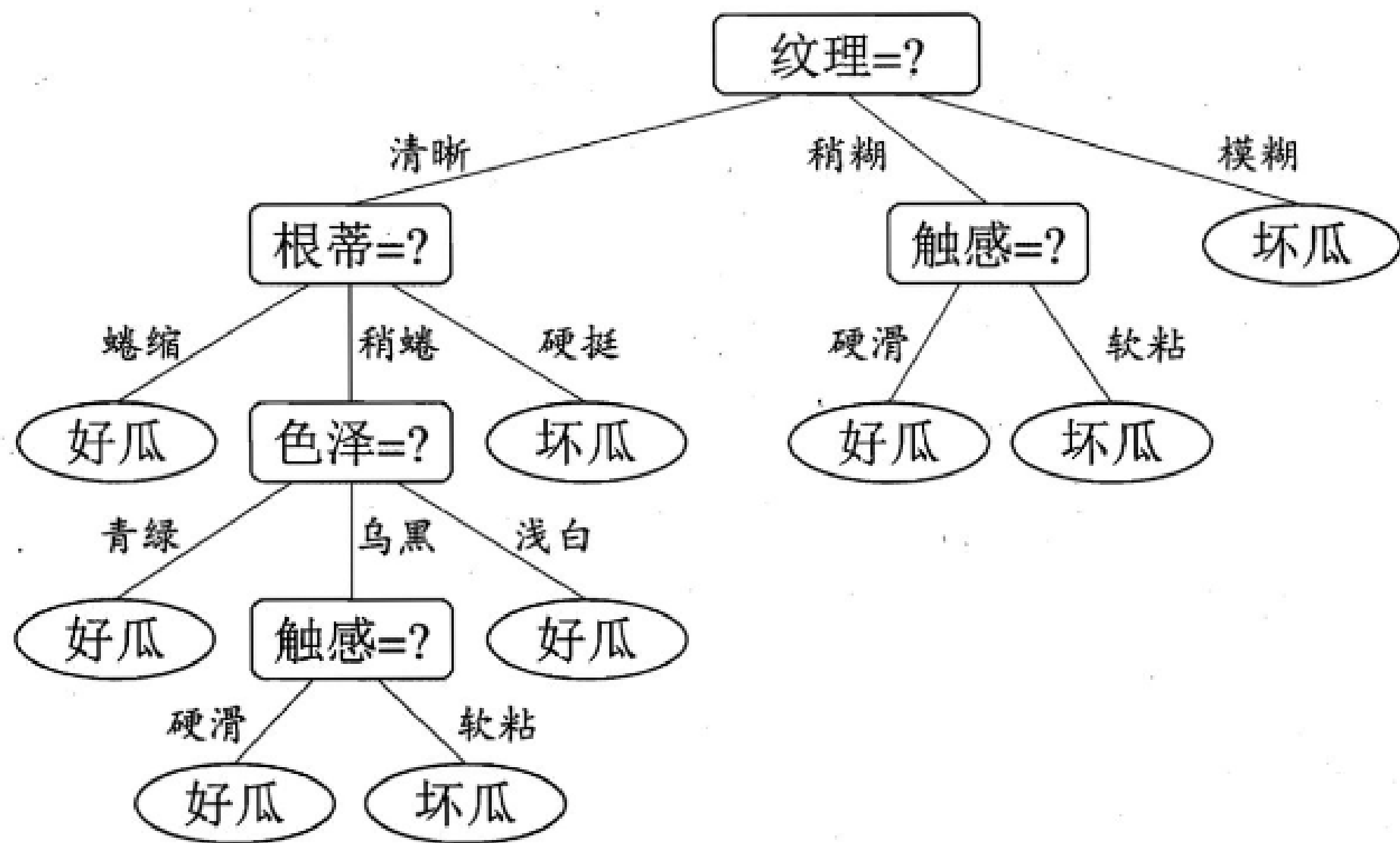
$$\text{Gain}(D1, \text{根蒂}) = 0.458$$

$$\text{Gain}(D1, \text{敲声}) = 0.331$$

$$\text{Gain}(D1, \text{脐部}) = 0.458$$

$$\text{Gain}(D1, \text{触感}) = 0.458$$

“根蒂”，“脐部”，“触感”三个属性均取得最大的信息增益，可选择其中之一作为划分属性。类似的，对每个分支结点进行上述操作，最终得到决策树如下：



# 几个相关概念---信息增益率

---

信息增益缺陷：

ID3 在计算的时候，**倾向于选择取值多的属性**。比如上面，如果将“编号”也作为一个候选划分属性，则可计算出它的信息增益为0.998，远大于其它属性。原因：“编号”将产生17个分支，每个分支结点仅包含一个样本，这些分支的样本纯度已经最大，然而，这样的决策树显然不具备“泛化能力”，无法对新样本进行有效预测。

另外，信息增益仅对离散数据进行处理，**不支持对连续数据的处理**。

# 几个相关概念---信息增益率

为了减少上述缺陷，**J.Ross Quinlan** 1993年提出C4.5决策树算法。该算法不直接使用信息增益，而采用信息增益率（gain ratio）来选择最优划分属性。

增益率公式为：

$$\text{Gain\_ratio}(D,a) = \frac{\text{Gain}(D,a)}{IV(a)}$$

其中

$$IV(a) = -\sum_{v=1}^{|Dv|} \frac{|Dv|}{|D|} \log_2 \frac{|Dv|}{|D|}$$

称为属性a的“固有价值(intrinsic value)”。属性a的可能取值数目越多（即V越大），则IV(a)的值通常会越大。

如前面 $IV(\text{触感}) = -(\frac{12}{17} \log_2 \frac{12}{17} + \frac{5}{17} \log_2 \frac{5}{17}) = 0.874$

# 几个相关概念---信息增益率

---

如前面

$$IV(\text{触感}) (v=2) = -(\frac{12}{17} \log_2 \frac{12}{17} + \frac{5}{17} \log_2 \frac{5}{17}) = 0.874$$

$$IV(\text{色泽}) (v=3) = 1.580$$

$$IV(\text{编号}) (v=17) = 4.008$$

需要注意：**增益率对可取值数目较少的属性有偏好**，因此，C4.5算法并不是这届选择增益率最大的候选划分属性，而使用了一个启发式：**先从候选划分属性中找出信息增益高于平均水平的属性，再从中选择增益率最高的。**

# 几个相关概念---基尼指数

---

CART决策树使用“基尼指数(Gini index)”来选择划分属性。数据集D的纯度可用基尼值来度量：

$$\text{Gini}(D) = \sum_{k=1}^{|y|} \sum_{k' \neq k} p_k p_{k'} = 1 - \sum_{k=1}^{|y|} p_k^2$$

直观说来，Gini(D)反映了从数据集D中随机抽取两个样本，其类别标记不一致的概率，因此，Gini(D)越小，则数据集D的纯度越高。

属性 a 的基尼指数定义为：

$$\text{Gini}(D, a) = \sum_{v=1}^{|D^v|} \frac{|D^v|}{|D|} \text{Gini}(D^v)$$

在候选属性集合 A 中，选择那个使得划分后基尼指数最小的属性作为最优划分属性。

# 几个相关概念---剪枝处理

**剪枝(pruning)**是决策树学习算法对付“**过拟合**”的主要手段，在决策树学习中，为了尽可能正确分类训练样本，结点划分过程将不断重复，有时会造成决策树分支过多，这是就可能因训练样本学的“太好”以至于把训练集自身的一些特点当作所有数据都具有的一般性质而导致过拟合。因此，可以通过主动去掉一些分支来降低过拟合风险。

决策树剪枝的基本策略包括：**预剪枝 (pre-pruning)**和**后剪枝(post-pruning)**。**预剪枝是在决策树生成过程中**，对每个结点在划分前先进行估计，若当前结点的划分不能带来决策树泛化性能提升，则停止划分，并将当前结点标记为叶结点；**后剪枝则是先从训练集生成一棵完整的决策树，然后自底向上对非叶子节点进行考察**，若将该节点对应的子树替换为叶结点带来决策树泛化能力提升，则将该子树替换为叶结点。



# 几个相关概念---剪枝处理

---

如何判断泛化能力的提升？

可以将样本按照比例分成两部分：训练集和验证集。通过训练集来建立决策树，通过验证集来进行评估。

表 4.2 西瓜数据集 2.0 划分出的训练集(双线上部)与验证集(双线下部)

编号	色泽	根蒂	敲声	纹理	脐部	触感	好瓜
1	青绿	蜷缩	浊响	清晰	凹陷	硬滑	是
2	乌黑	蜷缩	沉闷	清晰	凹陷	硬滑	是
3	乌黑	蜷缩	浊响	清晰	凹陷	硬滑	是
6	青绿	稍蜷	浊响	清晰	稍凹	软粘	是
7	乌黑	稍蜷	浊响	稍糊	稍凹	软粘	是
10	青绿	硬挺	清脆	清晰	平坦	软粘	否
14	浅白	稍蜷	沉闷	稍糊	凹陷	硬滑	否
15	乌黑	稍蜷	浊响	清晰	稍凹	软粘	否
16	浅白	蜷缩	浊响	模糊	平坦	硬滑	否
17	青绿	蜷缩	沉闷	稍糊	稍凹	硬滑	否
编号	色泽	根蒂	敲声	纹理	脐部	触感	好瓜
4	青绿	蜷缩	沉闷	清晰	凹陷	硬滑	是
5	浅白	蜷缩	浊响	清晰	凹陷	硬滑	是
8	乌黑	稍蜷	浊响	清晰	稍凹	硬滑	是
9	乌黑	稍蜷	沉闷	稍糊	稍凹	硬滑	否
11	浅白	硬挺	清脆	模糊	平坦	硬滑	否
12	浅白	蜷缩	浊响	模糊	平坦	软粘	否
13	青绿	稍蜷	浊响	稍糊	凹陷	硬滑	否

# 几个相关概念---预剪枝处理





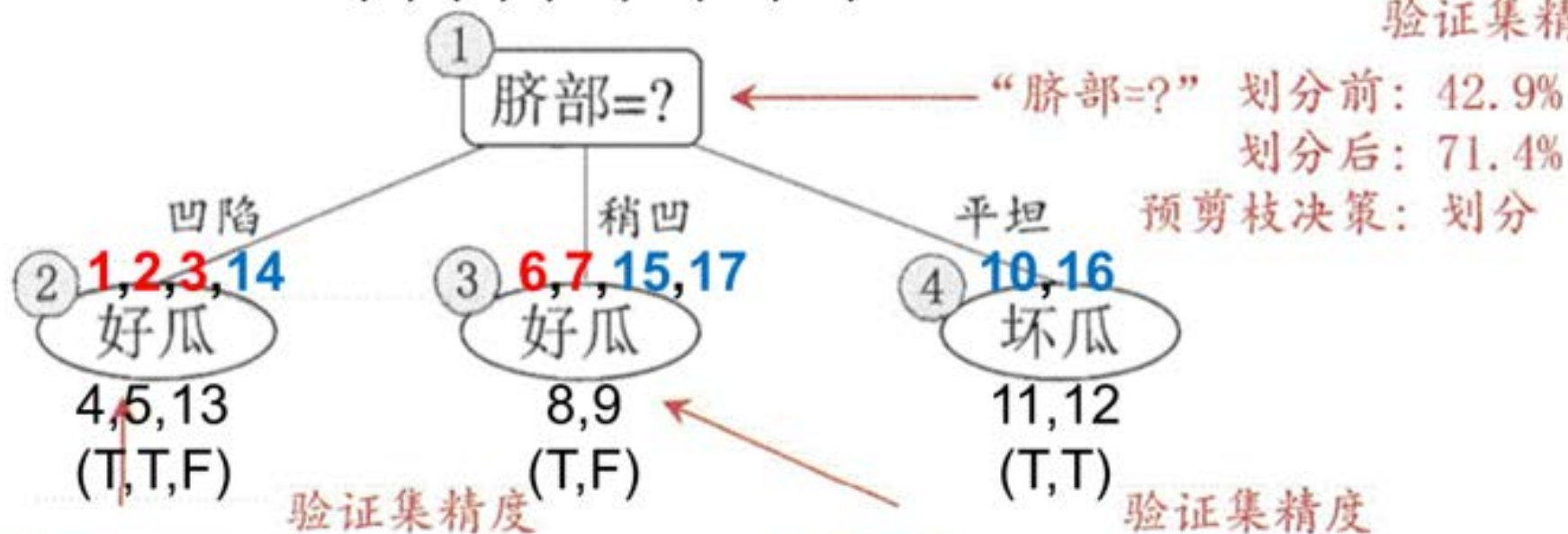
# 预剪枝

精度：正确分类的样本占所有样本的比例

验证集：4,5,8,9,11,12,13

训练集：好瓜 坏瓜  
1,2,3,6,7,10,14,15,16,17

验证集精度



验证集精度

验证集精度

“色泽=?” 划分前: 71.4%

划分后: 57.1%

预剪枝决策: 禁止划分

“根蒂=?” 划分前: 71.4%

划分后: 71.4%

预剪枝决策: 禁止划分

# 几个相关概念---预剪枝

---

预剪枝使得决策树的很多分支都没有“展开”

优点：

- 降低过拟合的风险
- 减少了训练时间开销和测试时间开销

不足：

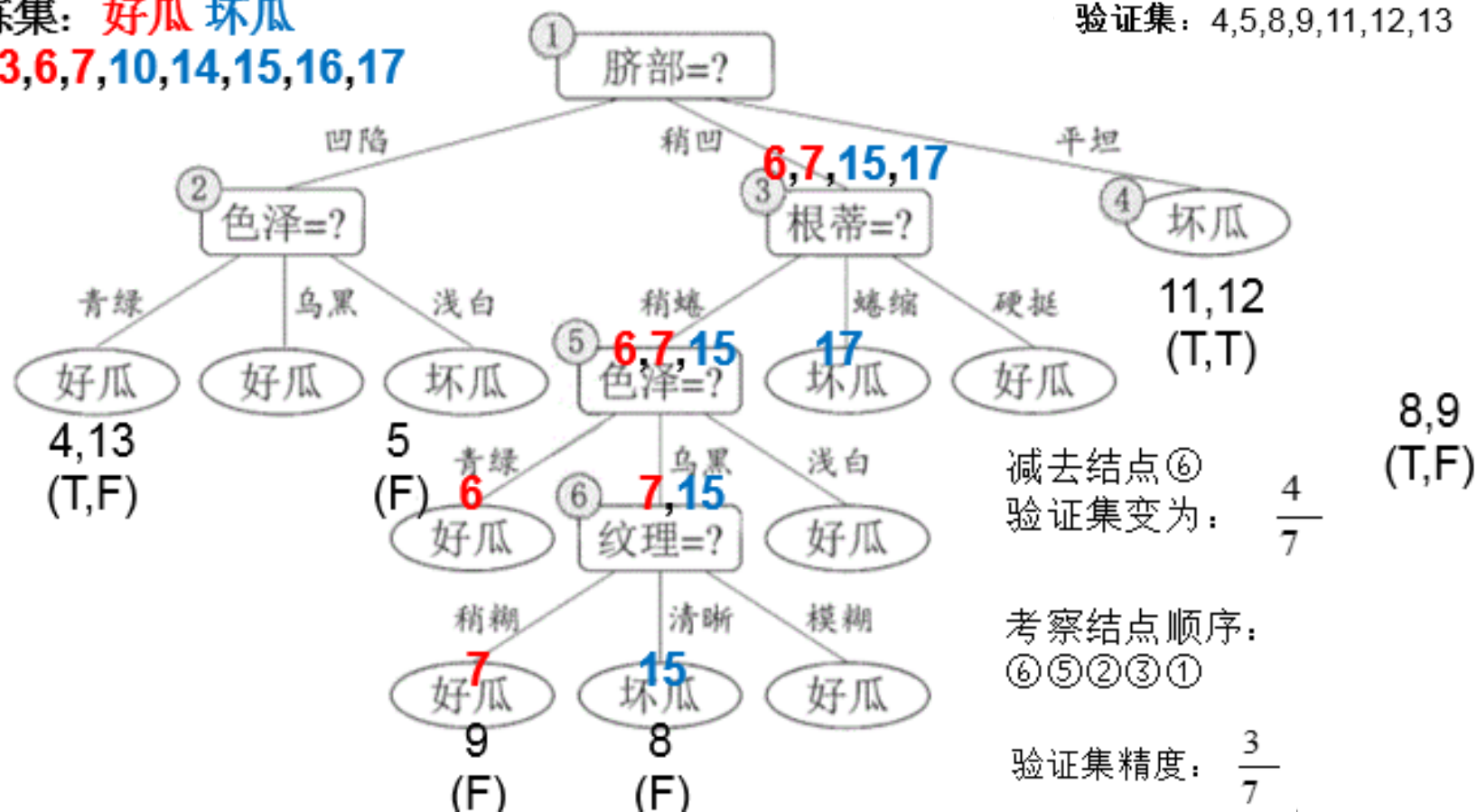
- 基于“贪心”本质禁止某些分支展开，带来了欠拟合的风险

# 几个相关概念---后剪枝处理

训练集: 好瓜 坏瓜

1,2,3,6,7,10,14,15,16,17

验证集: 4,5,8,9,11,12,13





原分支“色泽=?” 验证集精度

剪枝前: 57.1%

剪枝后: 71.4%

后剪枝决策: 剪枝

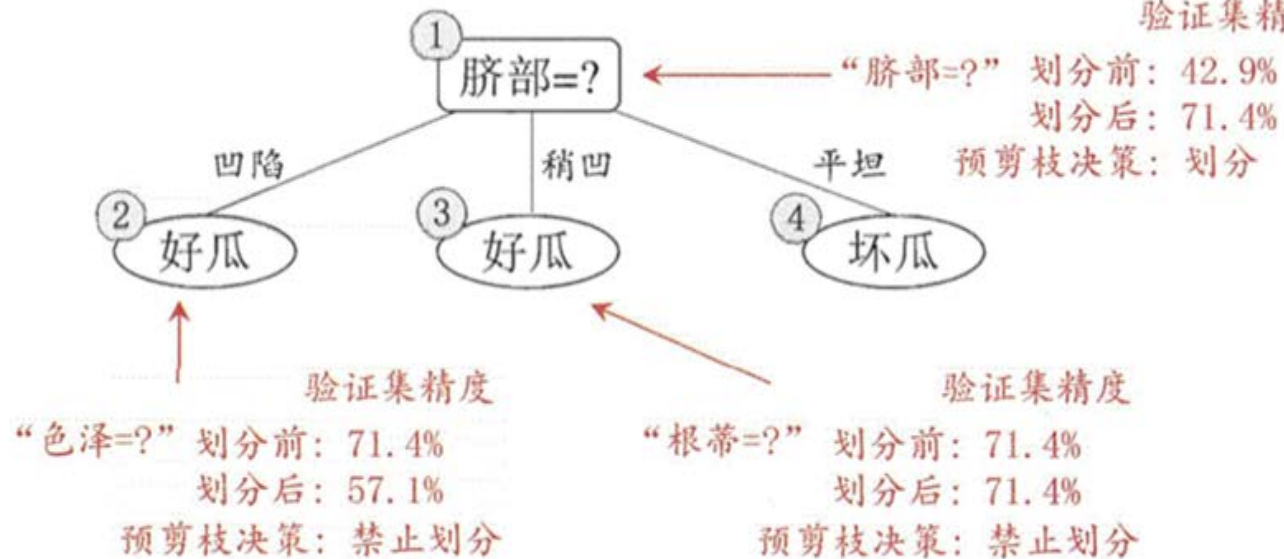
原分支“纹理=?” 验证集精度

剪枝前: 42.9%

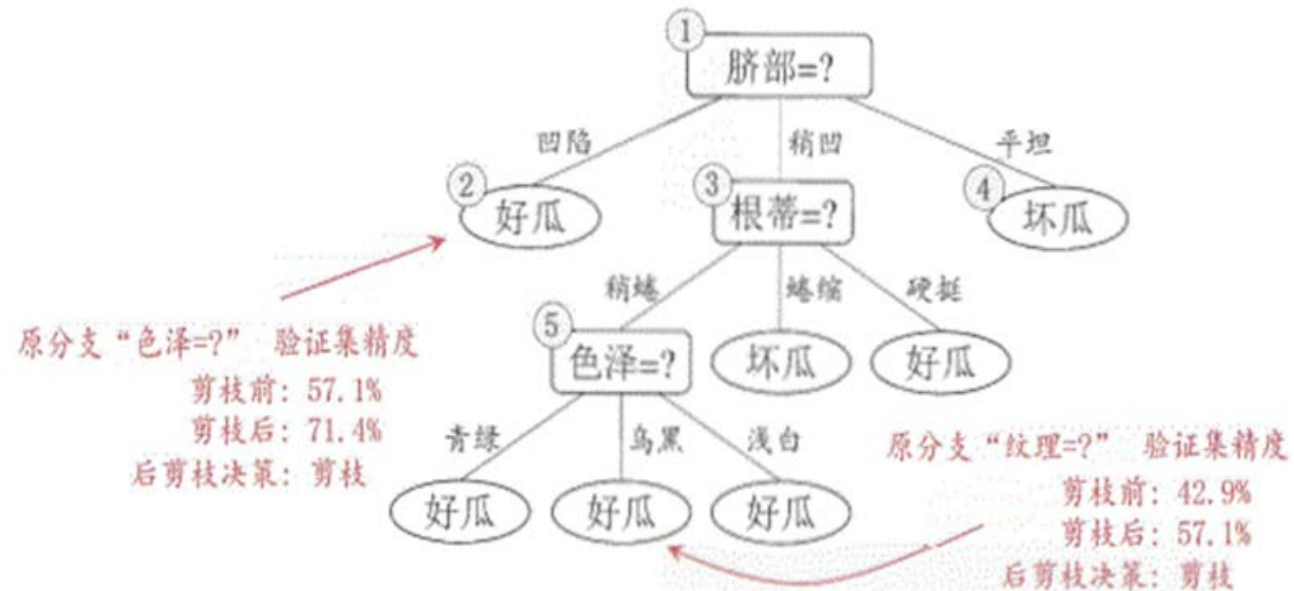
剪枝后: 57.1%

后剪枝决策: 剪枝

验证集精度



预剪枝



后剪枝



# 几个相关概念---后剪枝处理

---

后剪枝使得决策树通常比预剪枝决策树保留了更多的分支，一般情况下，后剪枝决策树欠拟合风险小，泛化能力优于预剪枝决策树。

优点：

- 欠拟合风险小
- 泛化能力优于预剪枝

不足：

- 训练时间开销较大

# C4.5相比ID3改进之处

---

## 1.采用信息增益率作为属性选择

因为 ID3 在计算的时候，倾向于选择取值多的属性。为了避免这个问题，C4.5 采用信息增益率的方式来选择属性。

信息增益率 = 信息增益 / 属性熵。

当属性有很多值的时候，相当于被划分成了许多份，虽然信息增益变大了，但是对于 C4.5 来说，属性熵也会变大，所以整体的信息增益率并不大。

## 2. 采用悲观剪枝

ID3 构造决策树的时候，容易产生过拟合的情况。在 C4.5 中，会在决策树构造之后采用悲观剪枝（PEP），这样可以提升决策树的泛化能力。悲观剪枝是后剪枝技术中的一种，通过递归估算每个内部节点的分枝错误率，比较剪枝前后这个节点的分枝错误率来决定是否对其进行剪枝。这种剪枝方法不再需要一个单独的测试数据集。

# C4.5相比ID3改进之处

---

## 3. 离散化处理连续属性

C4.5 可以处理连续属性的情况，对连续的属性进行离散化的处理。比如打篮球存在的“湿度”属性，不按照“高、中”划分，而是按照湿度值进行计算，那么湿度取什么值都有可能。该怎么选择这个阈值呢，C4.5 选择具有最高信息增益的划分所对应的阈值。

## 4. 处理缺失值

针对数据集不完整的情况，C4.5 也可以进行处理。

# 几个相关概念---连续值处理

---

前面讨论的都是离散属性来生成决策树，如果遇到属性是连续值的情况如何处理？

方法：将连续值采用“二分法”进行离散化处理。C4.5决策树算法采用该机制。

# 几个相关概念---连续值处理

---

样本集  $D$

连续属性  $a$ ，有  $n$  个不同的取值，将  $n$  个取值从小到大排序：

$$\{a^1, a^2, \dots, a^n\}$$

划分点  $t$ （数值）将  $D$  划分为两个子集  $D_t^-$  和  $D_t^+$

$$\begin{array}{ccc} \{a^1, a^2, \dots, a^i\} & | & \{a^{i+1}, \dots, a^n\} \\ \hline D_t^- & t & D_t^+ \end{array}$$

显然，对相邻的属性取值  $a^i$   $a^{i+1}$  来说， $t$  在区间  $[a^i, a^{i+1})$  中取任意值所产生的划分结果都相同

可考察包含  $n - 1$  个元素的候选划分点集合

$$T_a = \left\{ \frac{a^i + a^{i+1}}{2} \mid 1 \leq i \leq n - 1 \right\},$$

即把区间  $[a^i, a^{i+1})$  的中位点  $\frac{a^i + a^{i+1}}{2}$  作为候选划分点. 然后, 我们就可像离散属性值一样来考察这些划分点, 选取最优的划分点进行样本集合的划分.

$$\begin{aligned} \text{Gain}(D, a) &= \max_{t \in T_a} \text{Gain}(D, a, t) \\ &= \max_{t \in T_a} \text{Ent}(D) - \sum_{\lambda \in \{-, +\}} \frac{|D_t^\lambda|}{|D|} \text{Ent}(D_t^\lambda), \end{aligned}$$

其中  $\text{Gain}(D, a, t)$  是样本集  $D$  基于划分点  $t$  二分后的信息增益. 于是, 我们就可选择使  $\text{Gain}(D, a, t)$  最大化的划分点.



表 4.3 西瓜数据集 3.0

编号	色泽	根蒂	敲声	纹理	脐部	触感	密度	含糖率	好瓜
1	青绿	蜷缩	浊响	清晰	凹陷	硬滑	0.697	0.460	是
2	乌黑	蜷缩	沉闷	清晰	凹陷	硬滑	0.774	0.376	是
3	乌黑	蜷缩	浊响	清晰	凹陷	硬滑	0.634	0.264	是
4	青绿	蜷缩	沉闷	清晰	凹陷	硬滑	0.608	0.318	是
5	浅白	蜷缩	浊响	清晰	凹陷	硬滑	0.556	0.215	是
6	青绿	稍蜷	浊响	清晰	稍凹	软粘	0.403	0.237	是
7	乌黑	稍蜷	浊响	稍糊	稍凹	软粘	0.481	0.149	是
8	乌黑	稍蜷	浊响	清晰	稍凹	硬滑	0.437	0.211	是
9	乌黑	稍蜷	沉闷	稍糊	稍凹	硬滑	0.666	0.091	否
10	青绿	硬挺	清脆	清晰	平坦	软粘	0.243	0.267	否
11	浅白	硬挺	清脆	模糊	平坦	硬滑	0.245	0.057	否
12	浅白	蜷缩	浊响	模糊	平坦	软粘	0.343	0.099	否
13	青绿	稍蜷	浊响	稍糊	凹陷	硬滑	0.639	0.161	否
14	浅白	稍蜷	沉闷	稍糊	凹陷	硬滑	0.657	0.198	否
15	乌黑	稍蜷	浊响	清晰	稍凹	软粘	0.360	0.370	否
16	浅白	蜷缩	浊响	模糊	平坦	硬滑	0.593	0.042	否
17	青绿	蜷缩	沉闷	稍糊	稍凹	硬滑	0.719	0.103	否

离散属性：脐部 根蒂 色泽...

连续属性：密度 含糖率...

密度	好瓜
0.243	否
0.245	否
0.343	否
0.360	否
0.403	是1
0.437	是2
0.481	是3
0.556	是4
0.593	否
0.680	是5
0.634	是6
0.639	否
0.657	否
0.666	否
0.697	是7
0.719	否
0.774	是8

$D_t^-$

如何  
算出?

$t = 0.381$

$D_t^+$

根结点包含17个训练样本，密度有17个不同取值  
候选划分点集合包含16个候选值  
每一个划分点能得到一个对应的信息增益

$$\text{Gain}(D, a) = \max_{t \in T_a} \text{Gain}(D, a, t)$$

$$= \max_{t \in T_a} \text{Ent}(D) - \sum_{\lambda \in \{-, +\}} \frac{|D_t^\lambda|}{|D|} \text{Ent}(D_t^\lambda),$$



根结点的信息熵仍为:  $\text{Ent}(D) = 0.998$

$$\text{Ent}(D_t^-) = -\left(\frac{0}{4} \log_2 \frac{0}{4} + \frac{4}{4} \log_2 \frac{4}{4}\right) = 0$$

$$\text{Ent}(D_t^+) = -\left(\frac{8}{13} \log_2 \frac{8}{13} + \frac{5}{13} \log_2 \frac{5}{13}\right) = 0.961$$

$\text{Gain}(D, \text{密度}, 0.381)$

$$= \text{Ent}(D) - \left[\frac{4}{17} \times \text{Ent}(D_t^-) + \frac{13}{17} \times \text{Ent}(D_t^+)\right]$$

$$= 0.263$$



$$\therefore \text{Gain}(D, a, t) = \text{Gain}(D, \text{密度}, 0.381) = 0.998 - \left( \frac{1}{17} * 0 + \frac{16}{17} * 0.961 \right) = 0.263$$

当  $t = 0.420$  时:

$$D_t^- = \{0.243, 0.245, 0.343, 0.360, 0.403\}, \quad D_t^+ = \{0.437, \dots, 0.657, 0.666, 0.697, 0.719, 0.774\}$$

$$\text{Ent}(D_t^-) = -\left( \frac{1}{5} \cdot \log_2 \frac{1}{5} + \frac{4}{5} \cdot \log_2 \frac{4}{5} \right) = 0.722,$$

$$\text{Ent}(D_t^+) = -\left( \frac{7}{12} \cdot \log_2 \frac{7}{12} + \frac{5}{12} \cdot \log_2 \frac{5}{12} \right) = 0.980,$$

$$\therefore \text{Gain}(D, a, t) = \text{Gain}(D, \text{密度}, 0.420) = 0.998 - \left( \frac{5}{17} * 0.722 + \frac{12}{17} * 0.980 \right) = 0.094$$

当  $t = 0.459$  时:

$$D_t^- = \{0.243, 0.245, 0.343, 0.360, 0.403, 0.437\}, \quad D_t^+ = \{0.481, \dots, 0.666, 0.697, 0.719, 0.774\}$$

$$\text{Ent}(D_t^-) = -\left( \frac{2}{6} \cdot \log_2 \frac{2}{6} + \frac{4}{6} \cdot \log_2 \frac{4}{6} \right) = 0.918,$$

$$\text{Ent}(D_t^+) = -\left( \frac{6}{11} \cdot \log_2 \frac{6}{11} + \frac{5}{11} \cdot \log_2 \frac{5}{11} \right) = 0.994,$$

$$\therefore \text{Gain}(D, a, t) = \text{Gain}(D, \text{密度}, 0.459) = 0.998 - \left( \frac{6}{17} * 0.918 + \frac{11}{17} * 0.994 \right) = 0.03$$

当  $t = 0.518$  时:

$$D_t^- = \{0.243, 0.245, 0.343, 0.360, 0.403, 0.437, 0.481\}, \quad D_t^+ = \{0.556, \dots, 0.697, 0.719, 0.774\}$$

# 几个相关概念---缺失值处理

---

出现缺失值数据如何处理？

放弃（简单，数据量充足并且缺失值数据不多可以考虑）

样本量本来不多，**不可放弃**，如何处理？？

需解决两个问题：

- 1 ) 如何在缺失值情况下进行划分属性选择？ （如何计算信息增益）
- 2 ) 给定划分属性，如样本数据在该属性上缺失，如何对样本进行划分？ （对于缺失属性值的样本如何将它从父结点划分到子结点中）

表 4.4 西瓜数据集 2.0 $\alpha$ 

编号	色泽	根蒂	敲声	纹理	脐部	触感	好瓜
1	—	蜷缩	浊响	清晰	凹陷	硬滑	是
2	乌黑	蜷缩	沉闷	清晰	凹陷	—	是
3	乌黑	蜷缩	—	清晰	凹陷	硬滑	是
4	青绿	蜷缩	沉闷	清晰	凹陷	硬滑	是
5	—	蜷缩	浊响	清晰	凹陷	硬滑	是
6	青绿	稍蜷	浊响	清晰	—	软粘	是
7	乌黑	稍蜷	浊响	稍糊	稍凹	软粘	是
8	乌黑	稍蜷	浊响	—	稍凹	硬滑	是
9	乌黑	—	沉闷	稍糊	稍凹	硬滑	否
10	青绿	硬挺	清脆	—	平坦	软粘	否
11	浅白	硬挺	清脆	模糊	平坦	—	否
12	浅白	蜷缩	—	模糊	平坦	软粘	否
13	—	稍蜷	浊响	稍糊	凹陷	硬滑	否
14	浅白	稍蜷	沉闷	稍糊	凹陷	硬滑	否
15	乌黑	稍蜷	浊响	清晰	—	软粘	否
16	浅白	蜷缩	浊响	模糊	平坦	硬滑	否
17	青绿	—	沉闷	稍糊	稍凹	硬滑	否

现实任务中，尤其在属性数目较多时，存在大量样本出现缺失值。  
出于成本和隐私的考虑

$D$  : 训练集

$\tilde{D}$  : 训练集中属性 $\mathbf{a}$ 上没有缺失值的样本子集

$\tilde{D}^v$  :  $\tilde{D}$  被属性 $\mathbf{a}$ 划分后的样本子集

$\tilde{D}_k$  :  $\tilde{D}$  中属于第 $k$ 类的样本子集

假定我们为每个样本  $\mathbf{x}$  赋予一个权重  $w_{\mathbf{x}}$ , 并定义

$$\rho = \frac{\sum_{\mathbf{x} \in \tilde{D}} w_{\mathbf{x}}}{\sum_{\mathbf{x} \in D} w_{\mathbf{x}}}, \quad \rho : \text{无缺失值样本所占比例} \quad (4.9)$$

$$\tilde{p}_k = \frac{\sum_{\mathbf{x} \in \tilde{D}_k} w_{\mathbf{x}}}{\sum_{\mathbf{x} \in \tilde{D}} w_{\mathbf{x}}} \quad (1 \leq k \leq |\mathcal{Y}|), \quad \tilde{p}_k : \text{无缺失值样本中第} k \text{类所占比例} \quad (4.10)$$


$$\tilde{r}_v = \frac{\sum_{\mathbf{x} \in \tilde{D}^v} w_{\mathbf{x}}}{\sum_{\mathbf{x} \in \tilde{D}} w_{\mathbf{x}}} \quad (1 \leq v \leq V). \quad (4.11)$$

$\tilde{r}_v$  : 无缺失值样本中在属性  $a$  上取值  $a^v$  的样本所占比例

直观地看, 对属性  $a$ ,  $\rho$  表示无缺失值样本所占的比例,  $\tilde{p}_k$  表示无缺失值样本中第  $k$  类所占的比例,  $\tilde{r}_v$  则表示无缺失值样本中在属性  $a$  上取值  $a^v$  的样本所占的比例. 显然,  $\sum_{k=1}^{|\mathcal{Y}|} \tilde{p}_k = 1$ ,  $\sum_{v=1}^V \tilde{r}_v = 1$ .

基于上述定义, 我们可将信息增益的计算式(4.2)推广为

$$\begin{aligned} \text{Gain}(D, a) &= \rho \times \text{Gain}(\tilde{D}, a) \\ &= \rho \times \left( \text{Ent}(\tilde{D}) - \sum_{v=1}^V \tilde{r}_v \text{Ent}(\tilde{D}^v) \right), \end{aligned} \quad (4.12)$$


 无缺失值的样本子集  $\tilde{D}$  上的信息增益

其中由式(4.1), 有

$$\text{Ent}(\tilde{D}) = - \sum_{k=1}^{|\mathcal{Y}|} \tilde{p}_k \log_2 \tilde{p}_k.$$

对于问题2：对于有缺失值的样本如何将它从父结点划分到子结点中

- 若样本  $x$  在划分属性  $a$  上的取值已知，则将  $x$  划入与其取值对应的子结点，且样本权值在子结点中保持为  $w_x$

- 若样本  $x$  在划分属性  $a$  上的取值未知，则将  $x$  同时划入所有子结点，且样本权值在子结点中调整为  $\tilde{r}_v \cdot w_x$ ，就是让同一个样本以不同的概率划入不同的子结点中。

其中，  $w_x$  是为每个样本  $x$  赋予的一个权重

$\tilde{r}_v$  : 无缺失值样本中在属性  $a$  上取值  $a^v$  的样本所占比例

运用：

问题1 属性值缺失时，如何进行划分属性选择？  
=属性值缺失时，如何计算缺失属性的信息增益？



根结点包含样本集  $D$  中全部17个样本

属性“色泽”无缺失值的样例子集  $\tilde{D} = \{2, 3, 4, 6, 7, 8, 9, 10, 11, 12, 14, 15, 16, 17\}$   
包含14个样例：

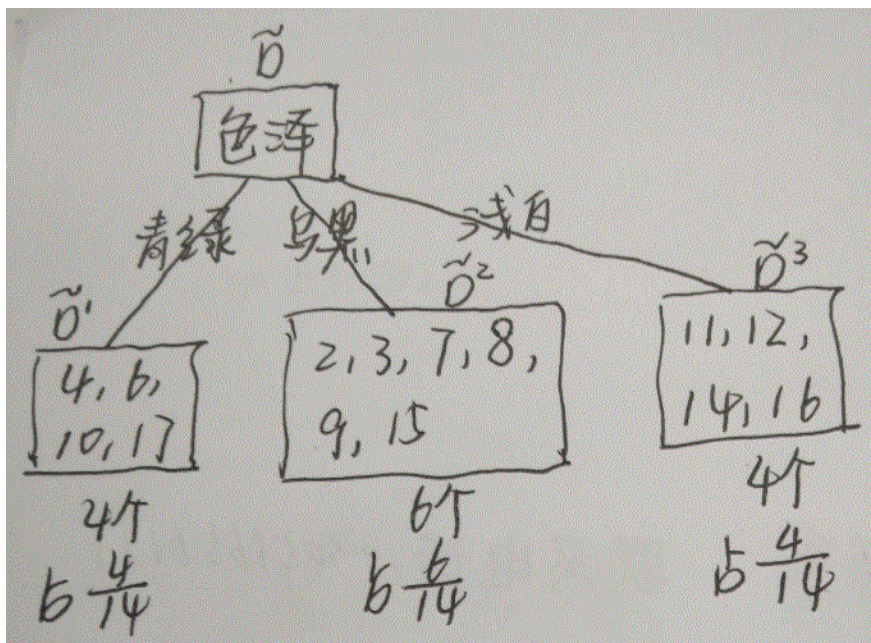
$$\rho = \frac{14}{17}$$

无缺失值样本所占比例

好瓜(6个)

坏瓜(8个)

$\tilde{p}_k$  : 无缺失值样本中第k类所占比例



$$\begin{aligned} \text{Ent}(\tilde{D}) &= - \sum_{k=1}^2 \tilde{p}_k \log_2 \tilde{p}_k \\ &= - \left( \frac{6}{14} \log_2 \frac{6}{14} + \frac{8}{14} \log_2 \frac{8}{14} \right) = 0.985. \end{aligned}$$

$$\text{Ent}(\tilde{D}^1) = - \left( \frac{2}{4} \log_2 \frac{2}{4} + \frac{2}{4} \log_2 \frac{2}{4} \right) = 1.000,$$

$$\text{Ent}(\tilde{D}^2) = - \left( \frac{4}{6} \log_2 \frac{4}{6} + \frac{2}{6} \log_2 \frac{2}{6} \right) = 0.918,$$

$$\text{Ent}(\tilde{D}^3) = - \left( \frac{0}{4} \log_2 \frac{0}{4} + \frac{4}{4} \log_2 \frac{4}{4} \right) = 0.000,$$

$\tilde{r}_v$  : 无缺失值样本中在属性  $a$  上取值  $a^v$  的样本所占比例

因此, 样本子集  $\tilde{D}$  上属性“色泽”的信息增益为

$$\begin{aligned}\text{Gain}(\tilde{D}, \text{色泽}) &= \text{Ent}(\tilde{D}) - \sum_{v=1}^3 \tilde{r}_v \text{Ent}(\tilde{D}^v) \\ &= 0.985 - \left( \frac{4}{14} \times 1.000 + \frac{6}{14} \times 0.918 + \frac{4}{14} \times 0.000 \right) \\ &= 0.306 .\end{aligned}$$

于是, 样本集  $D$  上属性“色泽”的信息增益为

$$\text{Gain}(D, \text{色泽}) = \rho \times \text{Gain}(\tilde{D}, \text{色泽}) = \frac{14}{17} \times 0.306 = 0.252 .$$

$\rho$  : 无缺失值样本所占比例

类似地可计算出所有属性在  $D$  上的信息增益:

$$\text{Gain}(D, \text{色泽}) = 0.252; \quad \text{Gain}(D, \text{根蒂}) = 0.171;$$

$$\text{Gain}(D, \text{敲声}) = 0.145; \quad \text{Gain}(D, \text{纹理}) = 0.424;$$

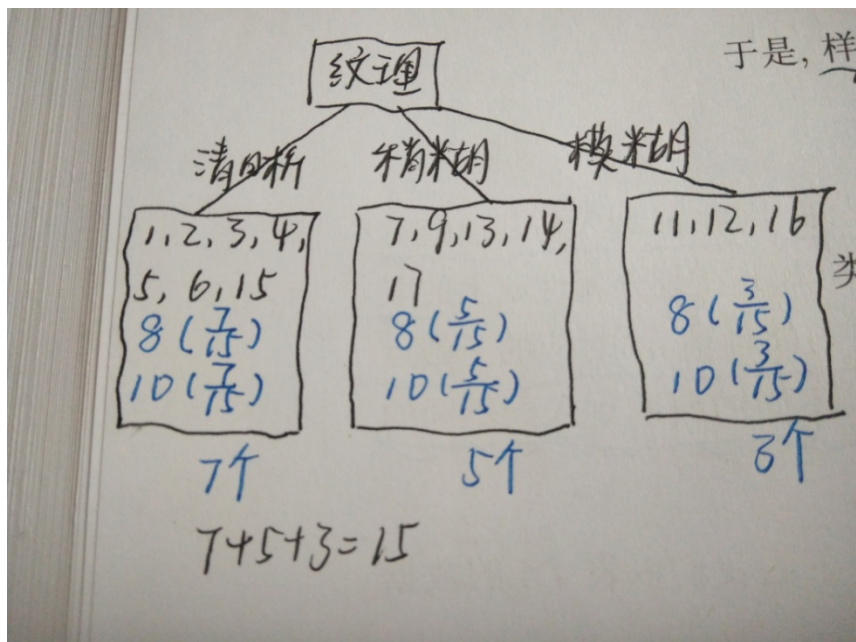
$$\text{Gain}(D, \text{脐部}) = 0.289; \quad \text{Gain}(D, \text{触感}) = 0.006.$$

“纹理”被用于对根结点进行划分

问题2 给定划分属性, 若样本在该属性上的值缺失, 如何对样本进行划分?



$\tilde{r}_v$  : 无缺失值样本中在属性  $a$  上取值  $a^v$  的样本所占比例



样本划分原则:

- 属性值已知, 划入与其取值对应的子结点, 样本权值不变, 仍为  $w_x$
- 属性值未知, 划入所有子结点, 样本权值调整为  $\tilde{r}_v \cdot w_x$ , 让同一个样本以不同的概率划入不同的子结点中

“纹理”属性值缺失的样本编号为: 8, 10

$D \setminus \{8, 10\}$  权值为:  $w_x = 1$

{8}和{10}同时进入三个分支中, 权值分别为:  $\frac{7}{15}, \frac{5}{15}, \frac{3}{15}$

# 应用实例分析——鸢尾花分类

---

iris 数据集构造一棵分类决策树。鸢尾花可以分成 Setosa、Versicolour 和 Virginica 三个品种，在这个数据集中，针对每一个品种，都有 50 个数据，每个数据中包括了 4 个属性，分别是花萼长度、花萼宽度、花瓣长度和花瓣宽度。通过这些数据，需要你来预测鸢尾花卉属于三个品种中的哪一种。

# 应用实例分析——鸢尾花分类

---

```
# encoding=utf-8
```

```
from sklearn.model_selection import train_test_split
```

```
from sklearn.metrics import accuracy_score
```

```
from sklearn.tree import DecisionTreeClassifier
```

```
from sklearn.datasets import load_iris
```

```
# 准备数据集
```

```
iris=load_iris()
```

# 应用实例分析——鸢尾花分类

---

# 获取特征集和分类标识

```
features = iris.data
```

```
labels = iris.target
```

# 随机抽取 33% 的数据作为测试集，其余为训练集

```
train_features, test_features, train_labels, test_labels = train_test_split(features, labels, test_size=0.33, random_state=0)
```

# 创建 CART 分类树

```
clf = DecisionTreeClassifier(criterion='gini')
```

# 拟合构造 CART 分类树

```
clf = clf.fit(train_features, train_labels)
```

# 应用实例分析——鸢尾花分类

---

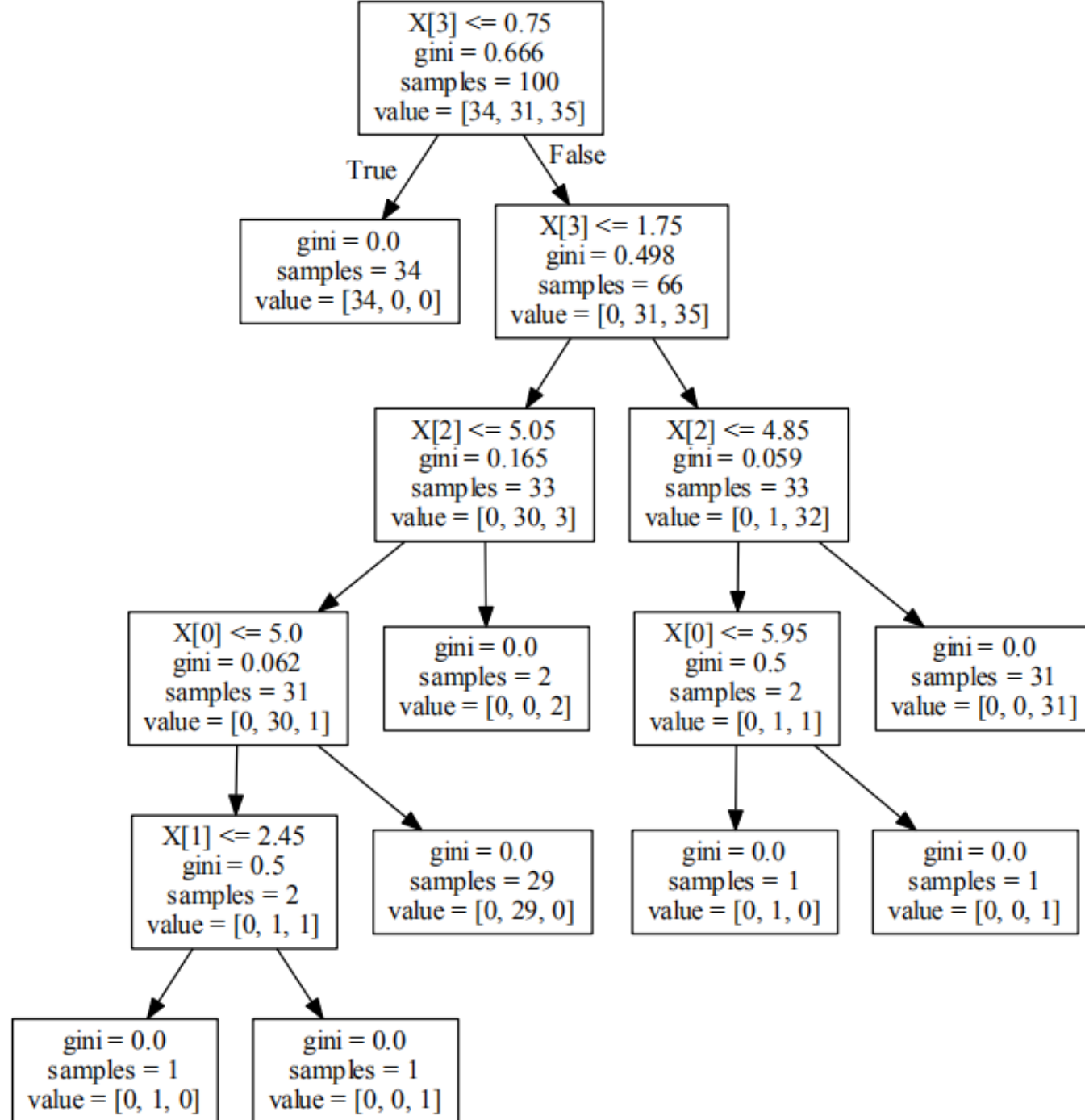
# 用 CART 分类树做预测

```
test_predict = clf.predict(test_features)
```

# 预测结果与测试集结果作比对

```
score = accuracy_score(test_labels, test_predict)
```

```
print("CART 分类树准确率 %.4lf" % score)
```



# 应用实例分析——波士顿房价预测

---

使用到 `sklearn` 自带的波士顿房价数据集，该数据集给出了影响房价的一些指标，比如犯罪率，房产税等，最后给出了房价。使用 **CART** 回归树对波士顿房价进行预测

# 应用实例分析——波士顿房价预测

---

```
# encoding=utf-8
from sklearn.metrics import mean_squared_error
from sklearn.model_selection import train_test_split
from sklearn.datasets import load_boston
from sklearn.metrics import
    r2_score,mean_absolute_error,mean_squared_error
from sklearn.tree import DecisionTreeRegressor
# 准备数据集
boston=load_boston()
```



# 应用实例分析——波士顿房价预测

---

# 探索数据

```
print(boston.feature_names)
```

# 获取特征集和房价

```
features = boston.data
```

```
prices = boston.target
```

# 随机抽取 33% 的数据作为测试集，其余为训练集

```
train_features, test_features, train_price, test_price =  
train_test_split(features, prices, test_size=0.33)
```

# 应用实例分析——波士顿房价预测

---

# 创建 CART 回归树

```
dtr=DecisionTreeRegressor()
```

# 拟合构造 CART 回归树

```
dtr.fit(train_features, train_price)
```

# 预测测试集中的房价

```
predict_price = dtr.predict(test_features)
```

# 测试集的结果评价

```
print('回归树二乘偏差均值:', mean_squared_error(test_price, predict_price))
```

```
print('回归树绝对值偏差均值:', mean_absolute_error(test_price, predict_price))
```

# 应用实例分析——泰坦尼克乘客生存预测

---

sklearn 中自带的决策树分类器 `DecisionTreeClassifier`，方法如下：

```
clf = DecisionTreeClassifier(criterion='entropy')
```

sklearn 中只实现了 ID3 与 CART 决策树

在构造 `DecisionTreeClassifier` 类时，其中有一个参数是 `criterion`，意为标准。它决定了构造的分类树是采用 ID3 分类树，还是 CART 分类树，对应的取值分别是 `entropy` 或者 `gini`：

**entropy**: 基于信息熵，也就是 ID3 算法，实际结果与 C4.5 相差不大；

**gini**: 默认参数，基于基尼系数。CART 算法是基于基尼系数做属性划分的，所以 `criterion=gini` 时，实际上执行的是 CART 算法。

# 应用实例分析——泰坦尼克乘客生存预测

---

通过设置 `criterion='entropy'` 可以创建一个ID3 决策树分类器，然后打印下 `clf`。

这里我们看到了很多参数，除了设置 `criterion` 采用不同的决策树算法外，一般建议使用默认的参数，默认参数不会限制决策树的最大深度，不限制叶子节点数，认为所有分类的权重都相等。当然你也可以调整这些参数，来创建不同的决策树模型。

参数表	作用
criterion	在基于特征划分数据集合时，选择特征的标准。默认是gini, 也可以是entropy。
splitter	在构造树时，选择属性特征的原则，可以是best或者random。默认是best，best代表在所有的特征中选择最好的，random代表在部分特征中选择最好的。
max_depth	决策树的最大深度，我们可以控制决策树的深度来防止决策树过拟合
max_features	在划分数据集时考虑的最多的特征值数量。为int或float类型。其中int值是每次split时最大特征数；float值是百分数，即特征数=max_features * n_features。
min_samples_split	当节点的样本数少于min_samples_split时，不再继续分裂。默认值为2

max_leaf_nodes	最大叶子节点数。int类型，默认为None。 默认情况下是不设置最大叶子节点数，特征不多时，不用设置。特征多时，可以通过设置最大叶子节点数，防止过拟合。
min_impurity_decrease	节点划分最小不纯度。float类型，默认值为0。 节点的不纯度必须大于这个阈值，否则该节点不再生成子节点。通过设置，可以限制决策树的增长。
min_impurity_split	信息增益的阈值。信息增益必须大于这个阈值，否则不分裂。
class_weight	类别权重。默认为None，也可以是dict或balanced。 dict类型：指定样本各类别的权重，权重大的类别在决策树构造的时候会进行偏倚。 balanced：算法自己计算权重，样本量少的类别所对应的样本权重会更高。
presort	bool类型，默认是false，表示在拟合前，是否对数据进行排序来加快树的构建。当数据集较小时，使用presort=true会加快分类器构造速度。当数据集庞大时，presort=true会导致整个分类非常缓慢。

# 应用实例分析——泰坦尼克乘客生存预测

---

在构造决策树分类器后，我们可以使用 `fit` 方法让分类器进行拟合，使用 `predict` 方法对新数据进行预测，得到预测的分类结果，也可以使用 `score` 方法得到分类器的准确率。

下面这个表格是 `fit` 方法、`predict` 方法和 `score` 方法的作用。

方法表	作用
<code>fit(features, labels)</code>	通过特征矩阵，分类标识， 让分类器进行拟合
<code>predict(features)</code>	返回预测结果
<code>score(features, labels)</code>	返回准确率



# 应用实例分析——泰坦尼克乘客生存预测

---

## Titanic 乘客生存预测

泰坦尼克海难是著名的十大灾难之一，究竟多少人遇难，各方统计的结果不一。

其中数据集格式为 `csv`，一共有两个文件：

`train.csv` 是训练数据集，包含特征信息和存活与否的标签；

`test.csv`: 测试数据集，只包含特征信息。

在训练集中，包括了以下字段，它们具体为：

字段	描述
PassengerId	乘客编号
Survived	是否幸存
Pclass	船票等级
Name	乘客姓名
Sex	乘客性别
SibSp	亲戚数量（兄妹、配偶数）
Parch	亲戚数量（父母、子女数）
Ticket	船票号码
Fare	船票价格
Cabin	船舱
Embarked	登陆港口

# 应用实例分析——泰坦尼克乘客生存预测

---

## 生存预测的关键流程

- 准备阶段：我们首先需要对训练集、测试集的数据进行探索，分析数据质量，并对数据进行清洗，然后通过特征选择对数据进行降维，方便后续分类运算；
- 分类阶段：首先通过训练集的特征矩阵、分类结果得到决策树分类器，然后将分类器应用于测试集。然后我们对决策树分类器的准确性进行分析，并对决策树模型进行可视化。

数据获取

数据探索

清洗数据

特征选择

准备阶段

决策树模型

模型评估  
& 预测

决策树可视化

分类阶段

# 应用实例分析——泰坦尼克乘客生存预测

---

## 模块 1：数据探索

- ✓ 使用 `info()` 了解数据表的基本情况：行数、列数、每列的数据类型、数据完整度；
- ✓ 使用 `describe()` 了解数据表的统计情况：总数、平均值、标准差、最小值、最大值等；
- ✓ 使用 `describe(include=['O'])` 查看字符串类型（非数字）的整体情况；
- ✓ 使用 `head` 查看前几行数据（默认是前 5 行）；
- ✓ 使用 `tail` 查看后几行数据（默认是最后 5 行）。

# 应用实例分析——泰坦尼克乘客生存预测

---

```
import pandas as pd
```

```
# 数据加载
```

```
train_data = pd.read_csv('./Titanic_Data/train.csv')
```

```
test_data = pd.read_csv('./Titanic_Data/test.csv')
```

```
# 数据探索
```

```
print(train_data.info())
```

```
print('-'*30)
```

# 应用实例分析——泰坦尼克乘客生存预测

---

```
print('-'*30)
```

```
print(train_data.describe(include=['O']))
```

```
print('-'*30)
```

```
print(train_data.head())
```

```
print('-'*30)
```

```
print(train_data.tail())
```

# 应用实例分析——泰坦尼克乘客生存预测

---

## 模块 2：数据清洗

过数据探索，发现 Age、Fare 和 Cabin 这三个字段的数据有所缺失。其中 Age 为年龄字段，是数值型，可以通过平均值进行补齐；Fare 为船票价格，是数值型，也可以通过其他人购买船票的平均值进行补齐。



# 应用实例分析——泰坦尼克乘客生存预测

---

# 使用平均年龄来填充年龄中的 nan 值

```
train_data['Age'].fillna(train_data['Age'].mean(), inplace=True)
```

```
test_data['Age'].fillna(test_data['Age'].mean(),inplace=True)
```

# 使用票价的均值填充票价中的 nan 值

```
train_data['Fare'].fillna(train_data['Fare'].mean(), inplace=True)
```

```
test_data['Fare'].fillna(test_data['Fare'].mean(),inplace=True)
```

# 应用实例分析——泰坦尼克乘客生存预测

---

Cabin 为船舱，有大量的缺失值。在训练集和测试集中的缺失率分别为 77% 和 78%，无法补齐；Embarked 为为登陆港口，有少量的缺失值，可以把缺失值补齐。

首先观察下 Embarked 字段的取值，方法如下：

```
print(train_data['Embarked'].value_counts())
```

发现一共就 3 个登陆港口，其中 S 港口人数最多，占到了 72%，因此我们将其余缺失的 Embarked 数值均设置为 S：

```
# 使用登录最多的港口来填充登录港口的 nan 值
```

```
train_data['Embarked'].fillna('S', inplace=True)
```

```
test_data['Embarked'].fillna('S',inplace=True)
```

# 应用实例分析——泰坦尼克乘客生存预测

---

## 模块 3：特征选择

特征选择是分类器的关键。特征选择不同，得到的分类器也不同。

通过数据探索我们发现，`PassengerId` 为乘客编号，对分类没有作用，可以放弃；`Name` 为乘客姓名，对分类没有作用，可以放弃；`Cabin` 字段缺失值太多，可以放弃；`Ticket` 字段为船票号码，杂乱无章且无规律，可以放弃。其余的字段包括：`Pclass`、`Sex`、`Age`、`SibSp`、`Parch` 和 `Fare`，这些属性分别表示了乘客的船票等级、性别、年龄、亲戚数量以及船票价格，可能会和乘客的生存预测分类有关系。先将 `Pclass`、`Sex`、`Age` 等这些其余的字段作特征，放到特征向量 `features` 里。

# 应用实例分析——泰坦尼克乘客生存预测

---

# 特征选择

```
features = ['Pclass', 'Sex', 'Age', 'SibSp', 'Parch', 'Fare', 'Embarked']
```

```
train_features = train_data[features]
```

```
train_labels = train_data['Survived']
```

```
test_features = test_data[features]
```

# 应用实例分析——泰坦尼克乘客生存预测

---

特征值里有一些是字符串，这样不方便后续的运算，需要转成数值类，比如 Sex 字段，有 male 和 female 两种取可以把它变成 Sex=male 和 Sex=female 两个字段，数值用 0 或 1 来表示。

同理 Embarked 有 S、C、Q 三种可能，也可以改成 Embarked=S、Embarked=C 和 Embarked=Q 三个字段，数值用 0 或 1 来表示。

可以使用 sklearn 特征选择中的 DictVectorizer 类，用它将可以处理符号化的对象，将符号转成数字 0/1 进行表示。具体方法如下：

```
from sklearn.feature_extraction import DictVectorizer
```

```
dvec=DictVectorizer(sparse=False)
```

```
train_features=dvec.fit_transform(train_features.to_dict(orient='record'))
```

# 应用实例分析——泰坦尼克乘客生存预测

---

代码中使用了 `fit_transform` 这个函数，它可以特征向量转化为特征值矩阵。然后我

们看下 `dvec` 在转化后的特征属性是怎样的，即查看 `dvec` 的 `feature_names_` 属性值，方法如下：

```
print(dvec.feature_names_)
```

# 应用实例分析——泰坦尼克乘客生存预测

---

## 模块 4：决策树模型

使用 ID3 算法，即在创建 `DecisionTreeClassifier` 时，设置 `criterion='entropy'`，然后使用 `fit` 进行训练，将特征值矩阵和分类标识结果作为参数传入，得到决策树分类器。

```
from sklearn.tree import DecisionTreeClassifier
```

```
# 构造 ID3 决策树
```

```
clf = DecisionTreeClassifier(criterion='entropy')
```

```
# 决策树训练
```

```
clf.fit(train_features, train_labels)
```

# 应用实例分析——泰坦尼克乘客生存预测

---

## 模块 5：模型预测 & 评估

在预测中，我们首先需要得到测试集的特征值矩阵，然后使用训练好的决策树 `clf` 进行预测，得到预测结果 `pred_labels`:

```
test_features=dvec.transform(test_features.to_dict(orient='record'))
```

```
# 决策树预测
```

```
pred_labels = clf.predict(test_features)
```



# 应用实例分析——泰坦尼克乘客生存预测

---

在模型评估中，决策树提供了 `score` 函数可以直接得到准确率，但是我们并不知道真实的预测结果，所以无法用预测值和真实的预测结果做比较。只能使用训练集中的数据进行模型评估，可以使用决策树自带的 `score` 函数计算下得到的结果：

# 得到决策树准确率

```
acc_decision_tree = round(clf.score(train_features, train_labels), 6)
```

```
print(u'score 准确率为 %.4lf' % acc_decision_tree)
```

# 应用实例分析——泰坦尼克乘客生存预测

---

用训练集做训练，再用训练集自身做准确率评估自然会很高

如何解决这个问题？？？

# 应用实例分析——泰坦尼克乘客生存预测

---

## K 折交叉验证的方式

原理是拿出大部分样本进行训练，少量的用于分类器的验证。K 折交叉验证，就是做 K 次交叉验证，每次选取 K 分之一的数据作为验证，其余作为训练。轮流 K 次，取平均值。

K 折交叉验证的原理是这样的：

- ✓ 1 . 将数据集平均分割成 K 个等份；
- ✓ 2 . 使用 1 份数据作为测试数据，其余作为训练数据；
- ✓ 3 . 计算测试准确率；
- ✓ 4 . 使用不同的测试集，重复 2、3 步骤。

# 应用实例分析——泰坦尼克乘客生存预测

---

在 sklearn 的 model\_selection 模型选择中提供了 cross\_val\_score 函数。cross\_val\_score 函数中的参数 cv 代表对原始数据划分成多少份，也就是我们的 K 值，一般建议 K 值取 10，因此可以设置 CV=10，我们可以对比下 score 和 cross\_val\_score 两种函数的正确率的评估结果：

```
import numpy as np
```

```
from sklearn.model_selection import cross_val_score
```

```
# 使用 K 折交叉验证 统计决策树准确率
```

```
print(u'cross_val_score 准确率为 %.4lf' % np.mean(cross_val_score(clf,  
train_features, train_labels, cv=10)))
```

# 应用实例分析——泰坦尼克乘客生存预测

---

## 模块 6：决策树可视化

sklearn 的决策树模型对我们来说，还是比较抽象的。可以使用 Graphviz 可视化工具帮我们把决策树呈现出来。

安装 Graphviz 库需要下面的几步：

1. 安装 graphviz 工具，这里是它的下载地址：  
<http://www.graphviz.org/download/>

2. 将 Graphviz 添加到环境变量 PATH 中；

3. 需要 Graphviz 库，如果没有可以使用 `pip install graphviz` 进行安装。

(Anaconda3 直接在命令提示页面下输入:`conda install graphviz`进行安装)

# 应用实例分析——泰坦尼克乘客生存预测

---

## 模块 6：决策树可视化

sklearn 的决策树模型对我们来说，还是比较抽象的。可以使用 Graphviz 可视化工具帮我们把决策树呈现出来。

安装 Graphviz 库需要下面的几步：

- 1. 安装 graphviz 工具，这里是它的下载地址：  
<http://www.graphviz.org/download/>
- 2. 将 Graphviz 添加到环境变量 PATH 中；
- 3. 需要 Graphviz 库，如果没有可以使用 `pip install graphviz` 进行安装。

使用 Graphviz 对决策树模型进行呈现，最后得到一个决策树可视化的 PDF 文件。

# 应用实例分析——泰坦尼克乘客生存预测

---

## 总结：

- 特征选择是分类模型好坏的关键。
- 模型准确率需要考虑是否有测试集的实际结果可以做对比，当测试集没有真实结果可以对比时，需要使用 K 折交叉验证  
`cross_val_score`;
- Graphviz 可视化工具可以很方便地将决策模型呈现出来。





# CART算法

## 概念

分类树：预测数据集中的类别，结果是离散值

回归树：预测数值型的目标值，比如房价，结果是连续值

## 分类树

GINI系数：反应了样本的不确定度

sklearn工具：DecisionTreeClassifier类

案例：给iris数据集进行分类

## 回归树

节点划分标准：最小绝对偏差（LAD），或者使用最小二乘偏差（LSD）

sklearn工具：DecisionTreeRegressor类

案例：对boston房价进行回归预测

## CCP剪枝

后剪枝方法，cost-complexity prune，中文叫做代价复杂度

STEP1、基于表面误差率增益值，来判断剪枝前后的误差，从而生成子树序列

STEP2、通过验证集，在第一步生成的子树序列中找到最优的决策树

# 决策树实战

## sklearn中的DecisionTreeClassifier类

### 参数表

- criterion参数决定决策树模型
  - CART算法: criterion='gini'
  - ID3算法: criterion='entropy'
- splitter, max\_depth, max\_features, min\_samples\_split, min\_samples\_leaf, max\_leaf\_nodes, min\_impurity\_decrease, min\_impurity\_split, class\_weight, presort

### 方法表

- fit: 拟合
- predict: 返回预测结果
- score: 返回准确率

## Titanic乘客生存预测

### 准备阶段

- 数据探索: 很重要, 方便我们对数据清洗、特征选择做决策
- 数据清洗: 重点考虑数据补齐, 格式类型一致
- 特征选择: 对于字符串类型的特征, 可以使用DictVectorizer类进行转化

### 分类阶段

- 决策树模型: DecisionTreeClassifier类
- 模型预测&评估: 针对没有实际测试集结果的, 需要用K折交叉验证
- 决策树可视化: Graphviz工具