决策树与随机森林

主讲老师：袁 方

OUTLINES

1. 预备知识
2. 树模型与线性模型
3. 决策树（Decision Tree）
4. 集成学习介绍（Ensemble learning）
5. Bagging - Random Forest
6. Boosting - GBDT

预备知识

- Entropy（热力学）：热力学中表征**物质状态**的参量之一，其物理意义是**体系混乱程度**的度量。
- 熵（概率论）：度量随机变量的**不确定性**。
- Information Entropy（信息论）：信息是用来消除随机不确定性的东西，度量样本集合**纯度**的指标

假设随机变量X的可能取值有$x_1, x_2, \cdots, x_n$ 。对于每一个可能的取值x_i，其概率$p(X = x_i) = p_i$

假定当前样本集合D中第k类样本所占的比例为$p_k\ (k = 1,2, \ldots, |n|)$，则D的信息熵为：

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

Ent(D)的值越小，则样本集合D的纯度越高。

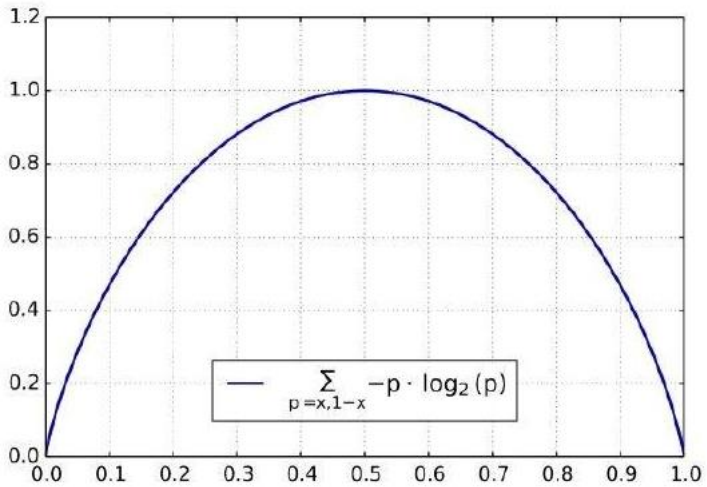
预备知识

假设随机变量X的可能取值有$x_1, x_2, \cdots, x_n$ 。对于每一个可能的取值x_i，其概率$p(X = x_i) = p_i$

假定当前样本集合D中第k类样本所占的比例为p_k $(k = 1,2, \ldots, |n|)$，则D的信息熵为：

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

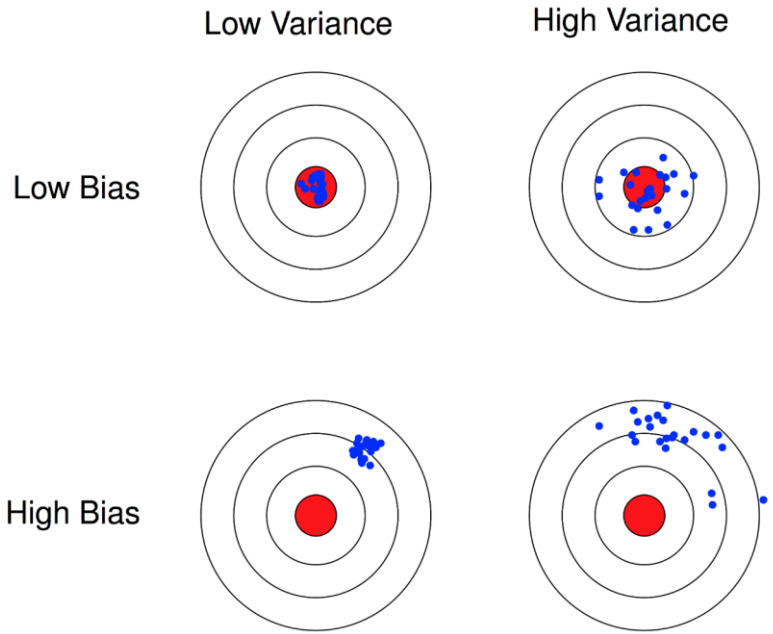
$Ent(D)$的值越小，则样本集合D的纯度越高。



Love or not?!
Entropy? Devil?

预备知识

- Variance（方差）：反映模型每一次**输出结果**与**模型输出期望**之间的误差，即模型的稳定性。
- Bias（偏差）：反映模型在样本上的**输出**与**真实值**之间的误差，即模型本身的精准度



Error = Bias + Variance

OUTLINES

1. 预备知识
2. 树模型与线性模型
3. 决策树（Decision Tree）
4. 集成学习介绍（Ensemble learning）
5. Bagging - Random Forest
6. Boosting - GBDT

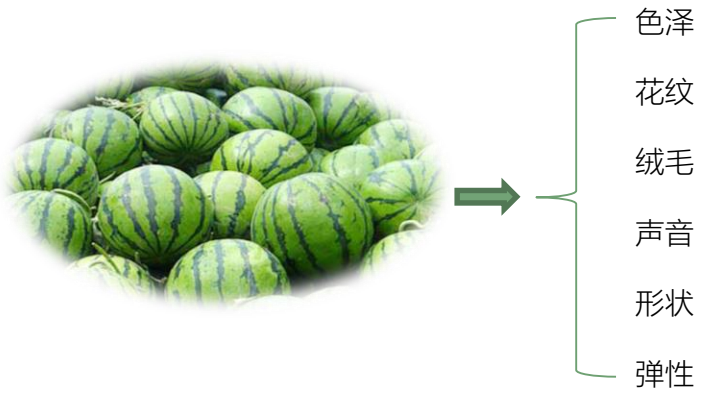
树模型与线性模型

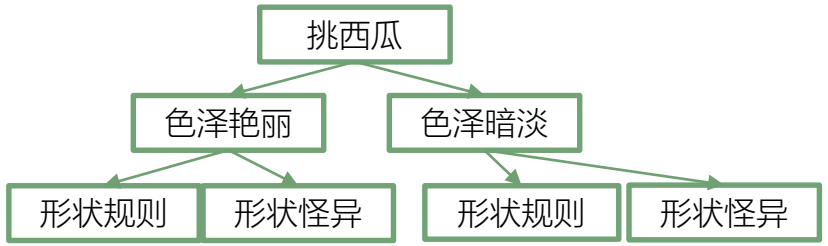
如何做一个决策？买个瓜？

- 色泽
- 花纹
- 绒毛
- 声音
- 形状
- 弹性

树模型与线性模型

如何做一个决策？买个瓜？



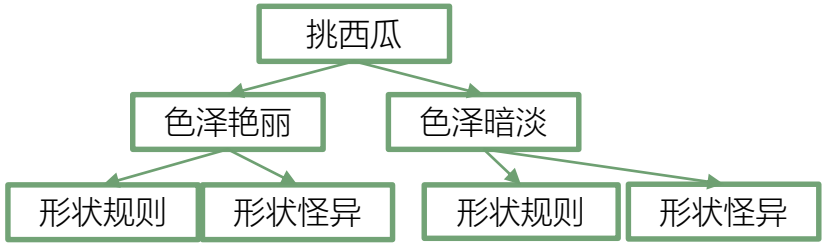


树模型

$$score = f(w_{color}x_{color} + w_{shape}x_{shape} + \cdots)$$

线性模型

树模型与线性模型



树模型

$$score = f(w_{color}x_{color} + w_{shape}x_{shape} + \cdots)$$

线性模型

线性模型

- 所有特征变换为概率
- 所有特征给予权重相加
- 只能找到线性分割

树模型

- 一个一个特征进行处理
- 对每一个特征做一个划分
- 可以找到非线性分割

OUTLINES

1. 预备知识
2. 树模型与线性模型
3. 决策树（Decision Tree）
4. 集成学习介绍（Ensemble learning）
5. Bagging - Random Forest
6. Boosting - GBDT

决策树（Decision Tree）

假定当前样本集合D中第k类样本所占的比例为$p_k\ (k = 1,2,\dots,|n|)$，

则D的信息熵为：$Ent(D) = -\sum_{k=1}^{|n|} p_k \log_2 p_k$

$Ent(D)$的值越小，则样本集合D的纯度越高。

目标：分类后**类内据**最小，**纯度最高**

提升纯度
- 信息增益（ID3）
- 信息增益率（C4.5）
- 基尼系数（CART）

决策树（Decision Tree）-ID3

假定**离散属性**a有V个可能的取值$\{a^1, a^2, \ldots, a^V\}$

以属性a对样本集合D进行划分，则会产生V个分支节点。其中第v个分支节点包含了样本集合D中所有在属性a上取值为a^v的样本，记作D^v

不同分支节点包含的样本数不同，给分支节点赋予权重$|D^v|/|D|$

样本数越多，分支节点的影响越大

决策树（Decision Tree）-ID3

离散属性a对样本集合D进行划分的“**信息增益**”(information gain)

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

信息增益越大，意味着使用属性a对样本集合D进行划分所获得的

“**纯度提升**”越大

$$a_*=\underset{a\in A}{\mathrm{argmax}}\,\mathrm{Gain}(D,a)$$

决策树（Decision Tree）-ID3

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

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

$$= -p_1 \log_2 p_1 \; -p_2 \log_2 p_2$$

$$= -\frac{8}{17}\log_2 \frac{8}{17} - \frac{9}{17}\log_2 \frac{9}{17}$$

$$= 0.998$$

决策树（Decision Tree）-ID3

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

$A = \{a_1 = \text{色泽}, a_2 = \text{根蒂}, a_3 = \text{敲声},$
$a_4 = \text{纹理}, a_5 = \text{脐部}, a_6 = \text{触感}\}$

以属性$\{a_1 = \text{色泽}\}$为例，有3个可能的取值：

$\{a_1^1 = \text{青绿}, a_1^2 = \text{乌黑}, a_1^3 = \text{浅白}\}$

对样本集合D进行划分，得到3个子集：

$D^1(\text{色泽} = \text{青绿})$

$D^2(\text{色泽} = \text{乌黑})$

$D^3(\text{色泽} = \text{浅白})$

决策树（Decision Tree）-ID3

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

对样本集合D进行划分，得到3个子集：

$$D^1(\text{色泽} = \text{青绿})$$

$$D^2(\text{色泽} = \text{乌黑})$$

$$D^3(\text{色泽} = \text{浅白})$$

针对每个子集，分别计算$Ent(D^i)$：

$$Ent(D^1) = -\frac{3}{6}\log_2\frac{3}{6} \ -\frac{3}{6}\log_2\frac{3}{6} = 1.000$$

$$Ent(D^2) = -\frac{4}{6}\log_2\frac{4}{6} \ -\frac{2}{6}\log_2\frac{2}{6} = 0.918$$

$$Ent(D^3) = -\frac{1}{5}\log_2\frac{1}{5} \ -\frac{4}{5}\log_2\frac{4}{5} = 0.722$$

决策树（Decision Tree）-ID3

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

计算出属性$\{a_1=$ 色泽$\}$的信息增益为：

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

$$= 0.998 - \left(\frac{6}{17} \times 1.000 + \frac{6}{17} \times 0.918 + \frac{5}{17} \times 0.722\right)$$

$$= 0.100$$

决策树（Decision Tree）-ID3

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

$$Gain(D, \text{色泽}) = 0.100$$

同理，可以分别计算出其他属性的信息增益：

$$Gain(D, \text{根蒂}) = 0.143$$

$$Gain(D, \text{敲声}) = 0.141$$

$$Gain(D, \text{纹理}) = 0.381$$

$$Gain(D, \text{脐部}) = 0.289$$

$$Gain(D, \text{触感}) = 0.006$$

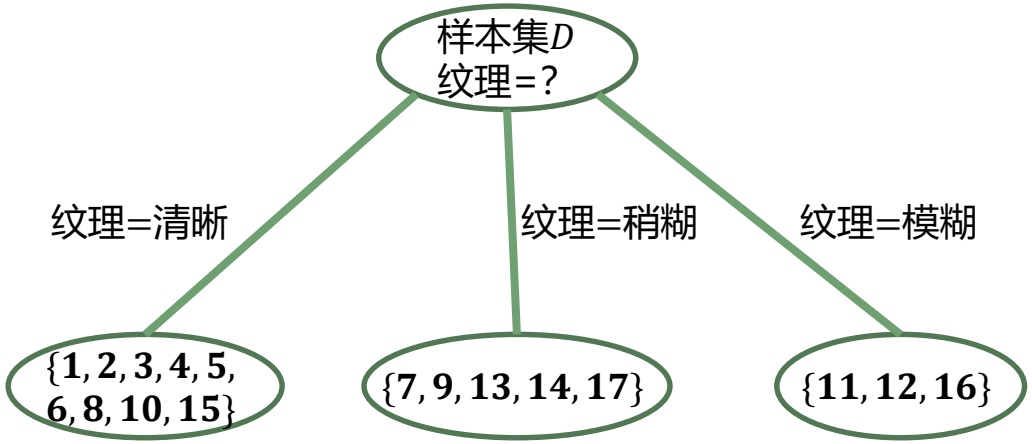
由此可知，$Gain(D, \text{纹理})$最大，

即属性“纹理”的信息增益最大，

因此，选择“纹理”作为划分属性。

决策树（Decision Tree）-ID3

编号	纹理
1	清晰
2	清晰
3	清晰
4	清晰
5	清晰
6	清晰
7	稍糊
8	清晰
9	稍糊
10	清晰
11	模糊
12	模糊
13	稍糊
14	稍糊
15	清晰
16	模糊
17	稍糊



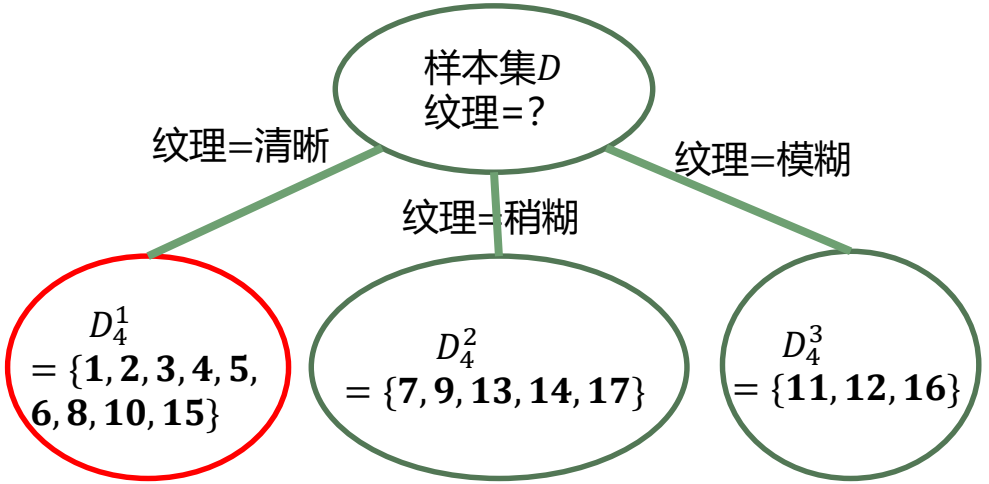
决策树（Decision Tree）-ID3

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

根据此数据集，采用ID3算法，计算并训练出完整的决策树。

决策树（Decision Tree）-ID3

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



$A = \{a_1 = \text{色泽}, a_2 = \text{根蒂}, a_3 = \text{敲声}, a_5 = \text{脐部}, a_6 = \text{触感}\}$

$\{D_1^1 = \text{青绿}, D_1^2 = \text{乌黑}, D_1^3 = \text{浅白}\}$

$$Ent(D_4^1) = -\frac{7}{9}\log_2\frac{7}{9} \ -\frac{2}{9}\log_2\frac{2}{9} = 0.764$$

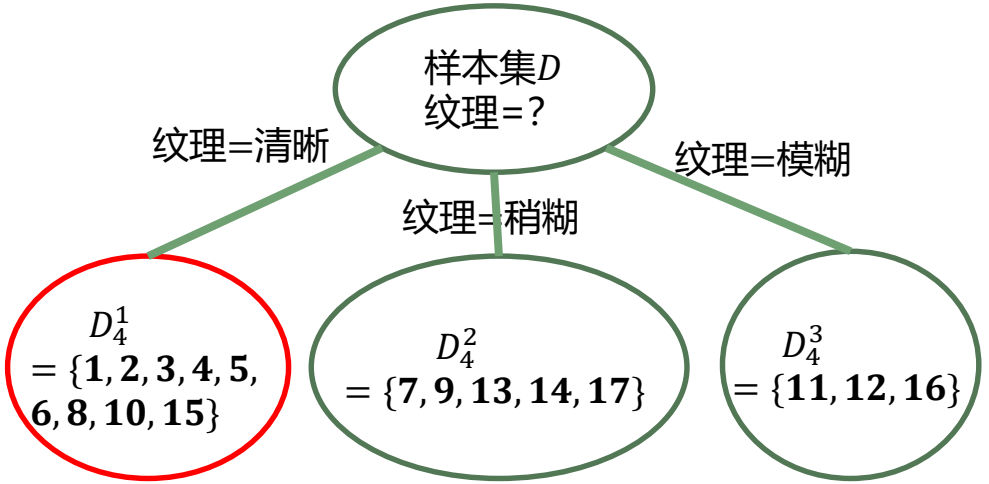
$$Ent(D_{41}^1) = -\frac{3}{4}\log_2\frac{3}{4} \ -\frac{1}{4}\log_2\frac{1}{4} = 0.811$$

$$Ent(D_{41}^2) = -\frac{3}{4}\log_2\frac{3}{4} \ -\frac{1}{4}\log_2\frac{1}{4} = 0.811$$

$$Ent(D_{41}^3) = -\frac{1}{1}\log_2\frac{1}{1} \ -\frac{0}{1}\log_2\frac{0}{1} = 0$$

决策树（Decision Tree）-ID3

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



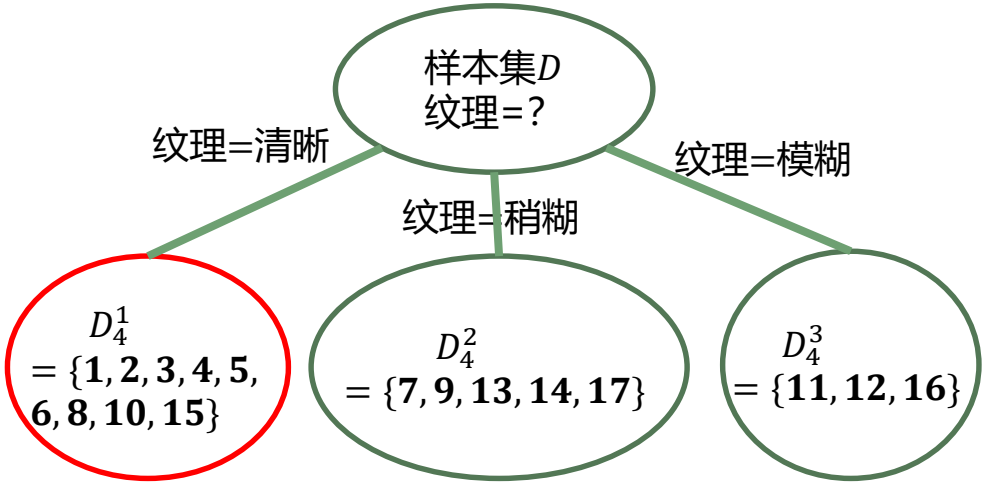
$$Gain(D_4^1, \text{色泽}) = Ent(D_4^1) - \sum_{v=1}^{V} \frac{|D_{41}^v|}{|D_{41}^1|} Ent(D_{41}^v)$$

$$= 0.764 - \left(\frac{4}{9} \times 0.811 + \frac{4}{9} \times 0.811 + \frac{1}{9} \times 0\right)$$

$$= 0.043$$

决策树（Decision Tree）-ID3

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



$$Gain(D_4^1, \text{色泽}) = 0.043$$

$$Gain(D_4^1, \text{根蒂}) = 0.458$$

$$Gain(D_4^1, \text{敲声}) = 0.331$$

$$Gain(D_4^1, \text{脐部}) = 0.458$$

$$Gain(D_4^1, \text{触感}) = 0.458$$

决策树（Decision Tree）-ID3

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

Probs：

- 以“编号”作为属性，对数据集进行划分，得到信息增益为0.998，远大于其他属性
- 将产出17个分支节点，且每个分支节点中只包含1个样本，纯度最大
- 不具备泛化能力
- ID3，即采用信息增益来选择划分属性，对可取值数目较多的属性有偏好，容易构造庞大的宽且浅的决策树

决策树（Decision Tree）

假定当前样本集合D中第k类样本所占的比例为$p_k\ (k = 1,2,\dots,|n|)$，

则D的信息熵为：$Ent(D) = -\sum_{k=1}^{|n|} p_k \log_2 p_k$

$Ent(D)$的值越小，则样本集合D的纯度越高。

目标：分类后类内据最小，纯度最高

- 提升纯度
  - 信息增益（ID3）
  - 信息增益率（C4.5）
  - 基尼系数（CART）

决策树（Decision Tree）-C4.5

增益率 (gain ratio)，减少信息增益偏好带来的不利影响，定义如下：

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

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

IV(a)是**属性a的固有值**(intrinsic value)，属性a可能的取值**数目越多**，则**IV(a)越大**

决策树（Decision Tree）-C4.5

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

$$A = \{a_1 = \text{色泽}, a_2 = \text{根蒂}, a_3 = \text{敲声},$$
$$a_4 = \text{纹理}, a_5 = \text{脐部}, a_6 = \text{触感}\}$$

$$IV(a_1 = \text{色泽}) = -\sum_{v=1}^{V} \frac{|D^v|}{|D|} \log_2 \frac{|D^v|}{|D|}$$

$$= -\frac{6}{17}\log_2\frac{6}{17} - \frac{6}{17}\log_2\frac{6}{17} - \frac{5}{17}\log_2\frac{5}{17}$$

$$= 1.580$$

决策树（Decision Tree）-C4.5

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

$$A = \{a_1 = \text{色泽}, a_2 = \text{根蒂}, a_3 = \text{敲声},$$
$$a_4 = \text{纹理}, a_5 = \text{脐部}, a_6 = \text{触感}\}$$

$$IV(a_0 = \text{编号}) = 4.088$$
$$IV(a_1 = \text{色泽}) = 1.580$$
$$IV(a_2 = \text{根蒂}) = 1.580$$
$$IV(a_3 = \text{敲声}) = 1.580$$
$$IV(a_4 = \text{纹理}) = 1.580$$
$$IV(a_5 = \text{脐部}) = 1.580$$
$$IV(a_6 = \text{触感}) = 0.874$$

决策树（Decision Tree）-C4.5

$Gain(D, 色泽) = 0.100$	$IV(a_1 = 色泽) = 1.580$	$Gain_{ratio(D,a_0)} = Gain(D, a_0)/IV(a_0) = 0.063$
$Gain(D, 根蒂) = 0.143$	$IV(a_2 = 根蒂) = 1.580$	$Gain_{ratio(D,a_0)} = Gain(D, a_0)/IV(a_0) = 0.091$
$Gain(D, 敲声) = 0.141$	$IV(a_3 = 敲声) = 1.580$	$Gain_{ratio(D,a_0)} = Gain(D, a_0)/IV(a_0) = 0.089$
$Gain(D, 纹理) = 0.381$	$IV(a_4 = 纹理) = 1.580$	$Gain_{ratio(D,a_0)} = Gain(D, a_0)/IV(a_0) = 0.241$
$Gain(D, 脐部) = 0.289$	$IV(a_5 = 脐部) = 1.580$	$Gain_{ratio(D,a_0)} = Gain(D, a_0)/IV(a_0) = 0.183$
$Gain(D, 触感) = 0.006$	$IV(a_6 = 触感) = 0.874$	$Gain_{ratio(D,a_0)} = Gain(D, a_0)/IV(a_0) = 0.007$

决策树（Decision Tree）-C4.5

Probs:

增益率对取值数目较少的属性有所偏好

C4.5算法不直接选择增益率最大的候选划分属性

启发式方法:

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

决策树（Decision Tree）

假定当前样本集合D中第k类样本所占的比例为$p_k\ (k = 1,2,\dots,|n|)$，

则D的信息熵为：$Ent(D) = -\sum_{k=1}^{|n|} p_k \log_2 p_k$

$Ent(D)$的值越小，则样本集合D的纯度越高。

目标：分类后类内据最小，纯度最高

- 提升纯度
  - 信息增益（ID3）
  - 信息增益率（C4.5）
  - 基尼系数（CART）

决策树（Decision Tree）-CART

CART，classification and regression tree，一种著名的决策树算法，

可用于分类和回归

采用"**基尼指标**"进行划分属性的选择。

基尼指数 (Gini index)：是度量样本集合纯度最常用的一种指标

反映了从样本集合D中随机抽取两个样本，其类别**标记不一致**的概率。

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

Gini(D)的值越小，则样本集合D的纯度越高。

决策树（Decision Tree）-CART

针对属性a的“基尼指数” (Gini index)定义为：

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

$\mathrm{Gini_index}(D, a)$的值越小，以属性a进行划分越优，即：

$$a_* = \underset{a \in A}{\operatorname{argmin}}\, \mathrm{Gini\_index}(D, a)$$

决策树（Decision Tree）-CART

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

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

$$= 1 - \left(p_1^2 + p_2^2\right)$$

$$= 1 - \left(\left(\frac{8}{17}\right)^2 + \left(\frac{9}{17}\right)^2\right)$$

$$= 0.498$$

决策树（Decision Tree）-CART

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

以 $\{a_1$=色泽$\}$对样本集合D进行划分，

针对每个子集，分别计算$Gini(D^i)$：

$$Gini(D^1) = 1 - \left(\left(\frac{3}{6}\right)^2 + \left(\frac{3}{6}\right)^2\right) = 0.500$$

$$Gini(D^2) = 1 - \left(\left(\frac{4}{6}\right)^2 + \left(\frac{2}{6}\right)^2\right) = 0.444$$

$$Gini(D^3) = 1 - \left(\left(\frac{1}{5}\right)^2 + \left(\frac{4}{5}\right)^2\right) = 0.320$$

决策树（Decision Tree）-CART

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

$$A = \{a_1 = \text{色泽}, a_2 = \text{根蒂}, a_3 = \text{敲声},$$
$$a_4 = \text{纹理}, a_5 = \text{脐部}, a_6 = \text{触感}\}$$

$$Gini\_index(D, a_1) = \sum_{v=1}^{V} \frac{|D^v|}{|D|} Gini(D^v)$$

$$= \frac{6}{17} \times 0.500 + \frac{6}{17} \times 0.444 + \frac{5}{17} \times 0.320$$

$$= 0.427$$

决策树（Decision Tree）-CART

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

根据此数据集，采用CART算法，

计算并训练出完整的决策树。

决策树（Decision Tree）

假定当前样本集合D中第k类样本所占的比例为$p_k\ (k = 1,2,\dots,|n|)$，

则D的信息熵为：$Ent(D) = -\sum_{k=1}^{|n|} p_k \log_2 p_k$

$Ent(D)$的值越小，则样本集合D的纯度越高。

目标：分类后类内据最小，纯度最高

- 提升纯度
  - 信息增益（ID3）
  - 信息增益率（C4.5）
  - 基尼系数（CART）

决策树（Decision Tree)

Probs:

- 缺失值
- 连续值

决策树（Decision Tree）

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		蜷缩	沉闷	清晰	凹陷	硬滑	是
3	乌黑		浊响	清晰		硬滑	是
4	青绿	蜷缩	沉闷		凹陷		是
5	浅白	蜷缩		清晰	凹陷	硬滑	是
6		稍蜷	浊响		稍凹	软粘	是
7	乌黑	稍蜷	浊响	稍糊		软粘	是
8	乌黑		浊响	清晰	稍凹	硬滑	是
9		稍蜷	沉闷	稍糊	稍凹	硬滑	否
10	青绿	硬挺		清晰	平坦	软粘	否
11	浅白	硬挺	清脆	模糊		硬滑	否
12	浅白		浊响	模糊	平坦		否
13	青绿	稍蜷	浊响		凹陷	硬滑	否
14	浅白	稍蜷	沉闷	稍糊	凹陷	硬滑	否
15	乌黑	稍蜷		清晰	稍凹	软粘	否
16	浅白	蜷缩	浊响	模糊		硬滑	否
17	青绿	蜷缩	沉闷	稍糊	稍凹	硬滑	否

ρ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**

$\tilde{r}_v$表示**无缺失值样本**中属性a上取值a^v**的样本**所占的**比例**

$$Ent(\tilde{D}) = -\sum_{k=1}^{2} \tilde{p}_k \log_2 \tilde{p}_k$$

$$= -\left(\frac{6}{14}\log_2\frac{6}{14} + \frac{8}{14}\log_2\frac{8}{14}\right)$$

$$= 0.985$$

决策树（Decision Tree）

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		蜷缩	沉闷	清晰	凹陷	硬滑	是
3	乌黑		浊响	清晰		硬滑	是
4	青绿	蜷缩	沉闷		凹陷		是
5	浅白	蜷缩		清晰	凹陷	硬滑	是
6		稍蜷	浊响		稍凹	软粘	是
7	乌黑	稍蜷	浊响	稍糊		软粘	是
8	乌黑		浊响	清晰	稍凹	硬滑	是
9		稍蜷	沉闷	稍糊	稍凹	硬滑	否
10	青绿	硬挺		清晰	平坦	软粘	否
11	浅白	硬挺	清脆	模糊		硬滑	否
12	浅白		浊响	模糊	平坦		否
13	青绿	稍蜷	浊响		凹陷	硬滑	否
14	浅白	稍蜷	沉闷	稍糊	凹陷	硬滑	否
15	乌黑	稍蜷		清晰	稍凹	软粘	否
16	浅白	蜷缩	浊响	模糊		硬滑	否
17	青绿	蜷缩	沉闷	稍糊	稍凹	硬滑	否

ρ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

$\tilde{r}_v$表示无缺失值样本中属性a上取值a^v的样本所占的比例

青绿 $Ent(\tilde{D}^1) = -\left(\frac{2}{5}\log_2\frac{2}{5} + \frac{3}{5}\log_2\frac{3}{5}\right) = 0.971$

乌黑 $Ent(\tilde{D}^2) = -\left(\frac{3}{4}\log_2\frac{3}{4} + \frac{1}{4}\log_2\frac{1}{4}\right) = 0.811$

浅白 $Ent(\tilde{D}^3) = -\left(\frac{1}{5}\log_2\frac{1}{5} + \frac{4}{5}\log_2\frac{4}{5}\right) = 0.722$

决策树（Decision Tree）

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		蜷缩	沉闷	清晰	凹陷	硬滑	是
3	乌黑		浊响	清晰		硬滑	是
4	青绿	蜷缩	沉闷		凹陷		是
5	浅白	蜷缩		清晰	凹陷	硬滑	是
6		稍蜷	浊响		稍凹	软粘	是
7	乌黑	稍蜷	浊响	稍糊		软粘	是
8	乌黑		浊响	清晰	稍凹	硬滑	是
9		稍蜷	沉闷	稍糊	稍凹	硬滑	否
10	青绿	硬挺		清晰	平坦	软粘	否
11	浅白	硬挺	清脆	模糊		硬滑	否
12	浅白		浊响	模糊	平坦		否
13	青绿	稍蜷	浊响		凹陷	硬滑	否
14	浅白	稍蜷	沉闷	稍糊	凹陷	硬滑	否
15	乌黑	稍蜷		清晰	稍凹	软粘	否
16	浅白	蜷缩	浊响	模糊		硬滑	否
17	青绿	蜷缩	沉闷	稍糊	稍凹	硬滑	否

ρ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

$\tilde{r}_v$表示无缺失值样本中属性a上取值a^v的样本所占的比例

$$Gain(\tilde{D}, \text{色泽}) = Ent(\tilde{D}) - \sum_{v=1}^{3} \tilde{r}_v Ent(\tilde{D}^v)$$

$$= 0.985 - \left(\frac{5}{14} \times 0.971 + \frac{4}{14} \times 0.811 + \frac{5}{14} \times 0.722 \right)$$

$$= 0.149$$

决策树（Decision Tree）

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		蜷缩	沉闷	清晰	凹陷	硬滑	是
3	乌黑		浊响	清晰		硬滑	是
4	青绿	蜷缩	沉闷		凹陷		是
5	浅白	蜷缩		清晰	凹陷	硬滑	是
6		稍蜷	浊响		稍凹	软粘	是
7	乌黑	稍蜷	浊响	稍糊		软粘	是
8	乌黑		浊响	清晰	稍凹	硬滑	是
9		稍蜷	沉闷	稍糊	稍凹	硬滑	否
10	青绿	硬挺		清晰	平坦	软粘	否
11	浅白	硬挺	清脆	模糊		硬滑	否
12	浅白		浊响	模糊	平坦		否
13	青绿	稍蜷	浊响		凹陷	硬滑	否
14	浅白	稍蜷	沉闷	稍糊	凹陷	硬滑	否
15	乌黑	稍蜷		清晰	稍凹	软粘	否
16	浅白	蜷缩	浊响	模糊		硬滑	否
17	青绿	蜷缩	沉闷	稍糊	稍凹	硬滑	否

ρ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

$\tilde{r}_v$表示无缺失值样本中属性a上取值a^v的样本所占的比例

$$Gain(D, \text{色泽}) = \rho \times Gain(\tilde{D}, \text{色泽})$$
$$= \frac{14}{17} \times 0.149$$
$$= 0.123$$

决策树（Decision Tree)

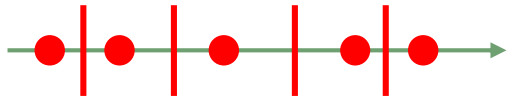
Probs:

- 缺失值
- 连续值

决策树（Decision Tree）

编号	色泽	根蒂	脐部	触感	密度	含糖率	好瓜
1	青绿	蜷缩	凹陷	硬滑	0.697	0.46	是
2	乌黑	蜷缩	凹陷	硬滑	0.774	0.376	是
3	乌黑	蜷缩	凹陷	硬滑	0.634	0.264	是
4	青绿	蜷缩	凹陷	硬滑	0.608	0.318	是
5	浅白	蜷缩	凹陷	硬滑	0.556	0.215	是
6	青绿	稍蜷	稍凹	软粘	0.403	0.237	是
7	乌黑	稍蜷	稍凹	软粘	0.481	0.149	是
8	乌黑	稍蜷	稍凹	硬滑	0.437	0.211	是
9	乌黑	稍蜷	稍凹	硬滑	0.666	0.091	否
10	青绿	硬挺	平坦	软粘	0.243	0.267	否
11	浅白	硬挺	平坦	硬滑	0.245	0.057	否
12	浅白	蜷缩	平坦	软粘	0.343	0.099	否
13	青绿	稍蜷	凹陷	硬滑	0.639	0.161	否
14	浅白	稍蜷	凹陷	硬滑	0.657	0.198	否
15	乌黑	稍蜷	稍凹	软粘	0.36	0.37	否
16	浅白	蜷缩	平坦	硬滑	0.593	0.042	否
17	青绿	蜷缩	稍凹	硬滑	0.719	0.103	否

二分法



属性a在D上有n个不同的取值，从小到大排列：

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

划分点t将D分为D_t^-和D_t^+，

每次t选择相邻两个取值的中位数：

$$T_a = \left\{ \frac{a^i + a^{i+1}}{2} | 1 \le i \le n-1 \right\}$$

决策树（Decision Tree）

编号	色泽	根蒂	脐部	触感	密度	含糖率	好瓜
1	青绿	蜷缩	凹陷	硬滑	0.697	0.46	是
2	乌黑	蜷缩	凹陷	硬滑	0.774	0.376	是
3	乌黑	蜷缩	凹陷	硬滑	0.634	0.264	是
4	青绿	蜷缩	凹陷	硬滑	0.608	0.318	是
5	浅白	蜷缩	凹陷	硬滑	0.556	0.215	是
6	青绿	稍蜷	稍凹	软粘	0.403	0.237	是
7	乌黑	稍蜷	稍凹	软粘	0.481	0.149	是
8	乌黑	稍蜷	稍凹	硬滑	0.437	0.211	是
9	乌黑	稍蜷	稍凹	硬滑	0.666	0.091	否
10	青绿	硬挺	平坦	软粘	0.243	0.267	否
11	浅白	硬挺	平坦	硬滑	0.245	0.057	否
12	浅白	蜷缩	平坦	软粘	0.343	0.099	否
13	青绿	稍蜷	凹陷	硬滑	0.639	0.161	否
14	浅白	稍蜷	凹陷	硬滑	0.657	0.198	否
15	乌黑	稍蜷	稍凹	软粘	0.36	0.37	否
16	浅白	蜷缩	平坦	硬滑	0.593	0.042	否
17	青绿	蜷缩	稍凹	硬滑	0.719	0.103	否

像处理离散属性值一样，选择最优划分点

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

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

决策树（Decision Tree）

编号	密度	好瓜
10	0.243	否
11	0.245	否
12	0.343	否
15	0.36	否
6	0.403	是
8	0.437	是
7	0.481	是
5	0.556	是
16	0.593	否
4	0.608	是
3	0.634	是
13	0.639	否
14	0.657	否
9	0.666	否
1	0.697	是
17	0.719	否
2	0.774	是

$$T_{\text{密度}} = \{0.244, 0.294, 0.351, \mathbf{0.381} \ldots\}$$

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

$$Ent(D) = -\left(\frac{8}{17}\log_2\frac{8}{17} + \frac{9}{17}\log_2\frac{9}{17}\right) = 0.998$$

$$\sum_{\lambda \in \{-,+\}} \frac{|D_{t=0.381}^\lambda|}{|D|} Ent\left(D_{t=0.381}^\lambda\right)$$

$$= \frac{4}{17} \times \left(-\frac{0}{4}\log_2\frac{0}{4} - \frac{4}{4}\log_2\frac{4}{4}\right) + \frac{13}{17} \times \left(-\frac{8}{13}\log_2\frac{8}{13} - \frac{5}{13}\log_2\frac{5}{13}\right) = 0.735$$

$$Gain(D,a) = 0.998 - 0.735 = 0.263$$

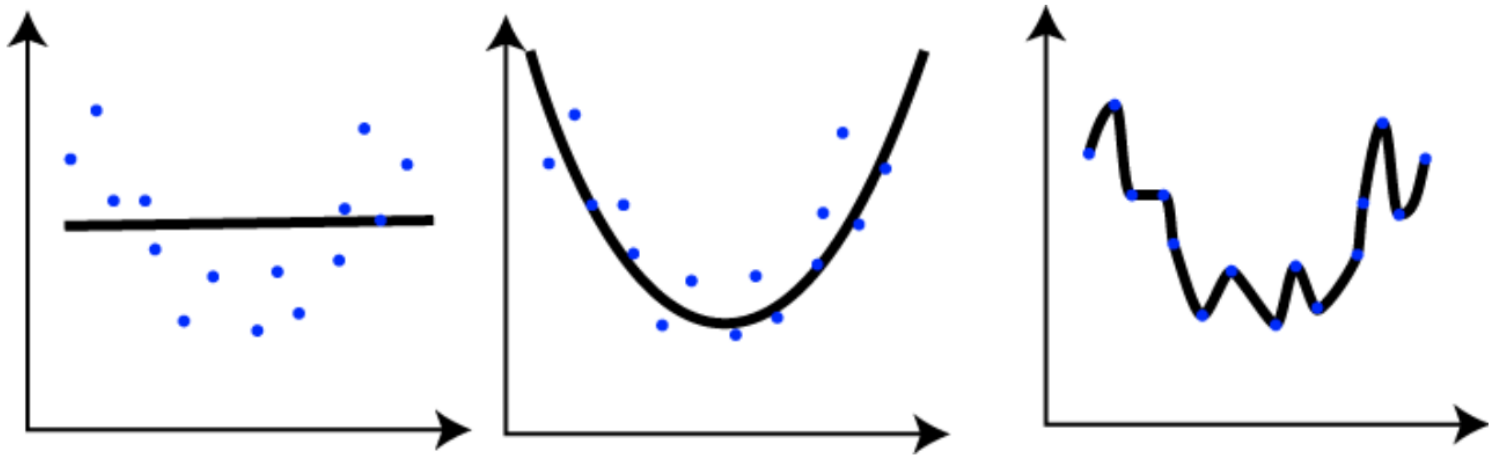
决策树（Decision Tree）

Probs：

- Overfitting

决策树（Decision Tree）

Overfitting



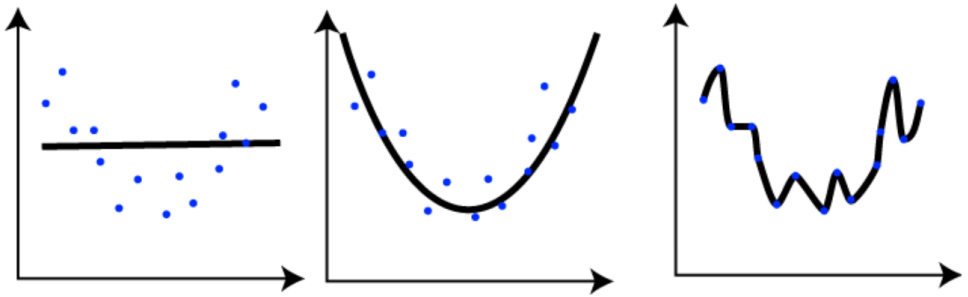
分支过多，训练得**"太好"**

把训练集的**特性**误当成**一般**性质

模型**缺乏泛化**性

决策树（Decision Tree）

Overfitting



把训练集的特点误当成一般性质

模型缺乏泛化性

分支过多，训练得"太好"

预剪枝

划分前进行泛化性评估

停止划分并标记为叶节点

后剪枝

自底向上对非叶节点评估泛化性

子树替换成叶节点

决策树（Decision Tree）

泛化性评估

采用留出法，将样本集分成“**训练集**”和“**验证集**”

用“训练集”训练结点的类别，

用“验证集”计算剪枝前后的精度变化。

决策树（Decision Tree）

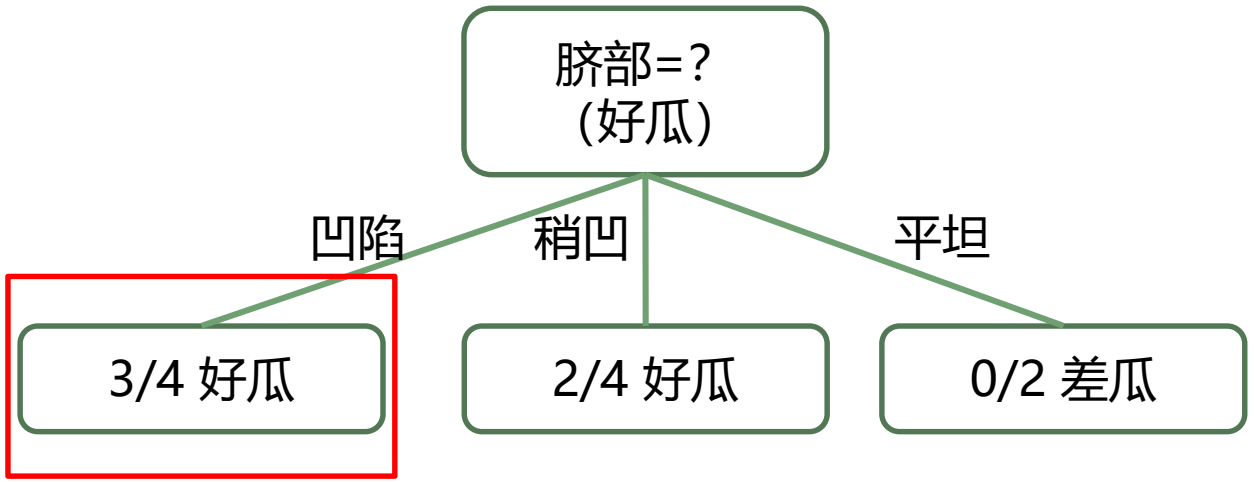
	编号	色泽	根蒂	敲声	纹理	脐部	触感	好瓜
训练集	1	青绿	蜷缩	浊响	清晰	凹陷	硬滑	是
	2	乌黑	蜷缩	沉闷	清晰	凹陷	硬滑	是
	3	乌黑	蜷缩	浊响	清晰	凹陷	硬滑	是
	6	青绿	稍蜷	浊响	清晰	稍凹	软粘	是
	7	乌黑	稍蜷	浊响	稍糊	稍凹	软粘	是
	10	青绿	硬挺	清脆	清晰	平坦	软粘	否
	14	浅白	稍蜷	沉闷	稍糊	凹陷	硬滑	否
	15	乌黑	稍蜷	浊响	清晰	稍凹	软粘	否
	16	浅白	蜷缩	浊响	模糊	平坦	硬滑	否
	17	青绿	蜷缩	沉闷	稍糊	稍凹	硬滑	否
验证集	4	青绿	蜷缩	沉闷	清晰	凹陷	硬滑	是
	5	浅白	蜷缩	浊响	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

决策树（Decision Tree）

预剪枝： 所有决策树的构建方法，都是在无法进一步**降低熵**的情况下才会停止创建分支的过程，为了避免过拟合，可以设定一个**阈值**，熵减小的数量小于这个阈值，即使还可以继续降低熵，也停止继续创建分支。也可以按照其它常用的metric是否下降，比如**准确率**

决策树（Decision Tree）

	脐部	好瓜
训练集	凹陷	是
	凹陷	是
	凹陷	是
	稍凹	是
	稍凹	是
	平坦	否
	凹陷	否
	稍凹	否
	平坦	否
	稍凹	否
验证集	凹陷	是
	凹陷	是
	稍凹	是
	稍凹	否
	平坦	否
	平坦	否
	凹陷	否

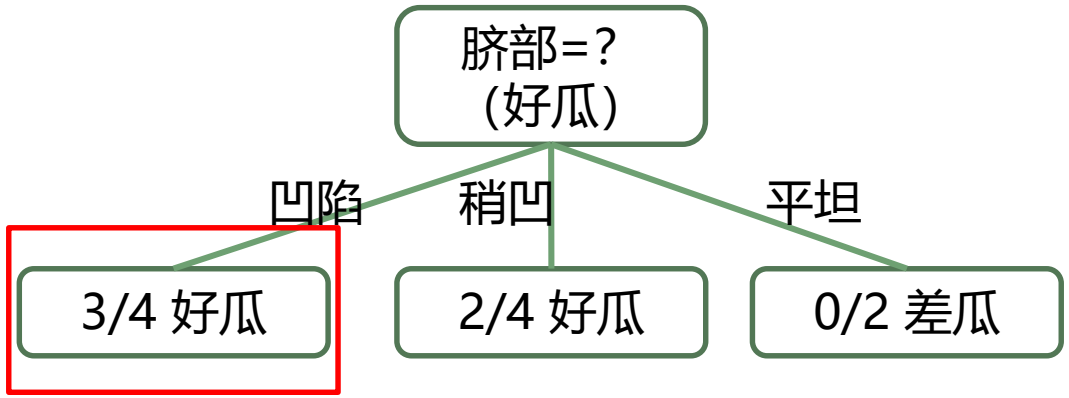


验证集中的精准率为 (2+1+2)/7=71.4%

针对红框标出的样本子集，计算“色泽”属性进行划分是否需要预剪枝?

决策树（Decision Tree）

	脐部	色泽	好瓜
训练集	凹陷	青绿	是
	凹陷	乌黑	是
	凹陷	乌黑	是
	凹陷	浅白	否
验证集	凹陷	青绿	是
	凹陷	浅白	是
	稍凹	乌黑	是
	稍凹	乌黑	否
	平坦	浅白	否
	平坦	浅白	否
	凹陷	青绿	否



验证集中的准确率为 (2+1+2)/7=71.4%

色泽属性进行划分后：

验证集中的准确率为 (1+1+2)/7=57.1%

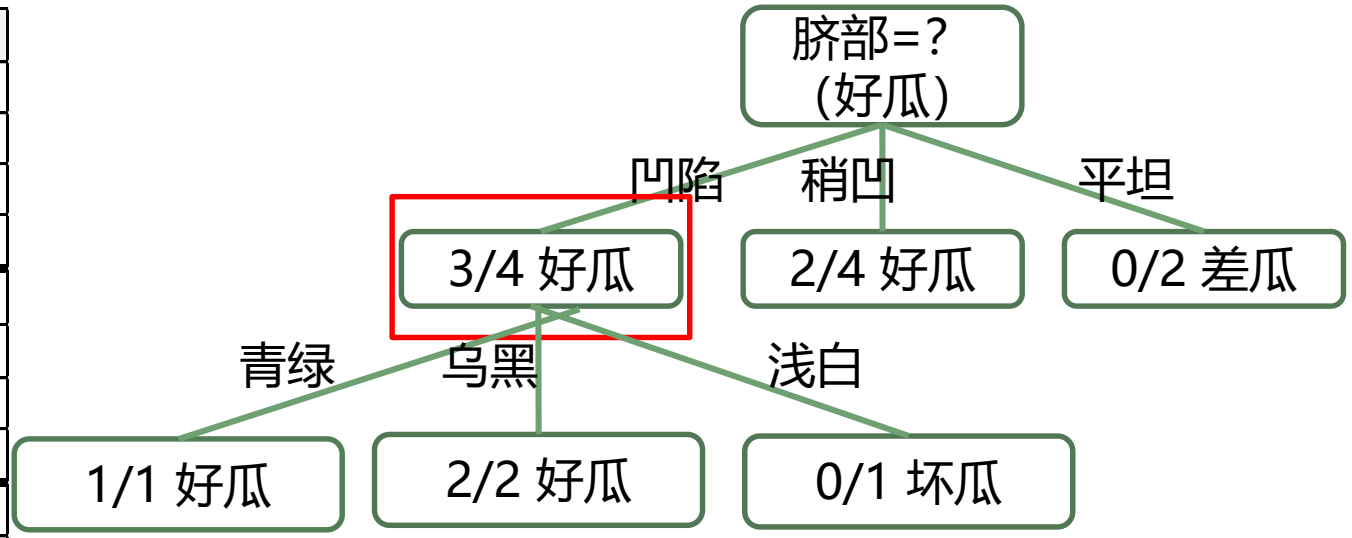
准确率下降，预剪枝（此结果不分枝）

决策树（Decision Tree）

后剪枝： **决策树构造完成后**进行剪枝。剪枝的过程是对拥有同样父节点的一组节点进行检查，判断如果将其合并，Metric(熵/Accuracy)的增加量是否小于某一阈值。如果确实小，则这一组节点可以合并一个节点，其中包含了所有可能的结果。后剪枝是目前最普遍的做法。

决策树（Decision Tree）

	脐部	色泽	好瓜
训练集	凹陷	青绿	是
	凹陷	乌黑	是
	凹陷	乌黑	是
	凹陷	浅白	否
验证集	凹陷	青绿	是
	凹陷	浅白	是
	稍凹	乌黑	是
	稍凹	乌黑	否
	平坦	浅白	否
	平坦	浅白	否
	凹陷	青绿	否



Bottom to Top
从深度最深的非叶子节点结点开始计算

色泽属性进行划分后：
验证集中的准确率为 (1+1+2)/7=57.1%

剪枝：Reduced-Error Pruning (REP,错误率降低剪枝)
验证集中的准确率为 (2+1+2)/7=71.4%

决策树（Decision Tree）- code

scikit-learn -V 0.20.2

sklearn.tree: Decision Trees

The `sklearn.tree` module includes decision tree-based models for classification and regression.

User guide: See the Decision Trees section for further details.

`tree.DecisionTreeClassifier` ([criterion, ...])	A decision tree classifier.
`tree.DecisionTreeRegressor` ([criterion, ...])	A decision tree regressor.
`tree.ExtraTreeClassifier` ([criterion, ...])	An extremely randomized tree classifier.
`tree.ExtraTreeRegressor` ([criterion, ...])	An extremely randomized tree regressor.
`tree.export_graphviz` (decision_tree[, ...])	Export a decision tree in DOT format.

sklearn.tree.DecisionTreeClassifier

class sklearn.tree. **DecisionTreeClassifier** (*criterion='gini', splitter='best', max_depth=None, min_samples_split=2, min_samples_leaf=1, min_weight_fraction_leaf=0.0, max_features=None, random_state=None, max_leaf_nodes=None, min_impurity_decrease=0.0, min_impurity_split=None, class_weight=None, presort=False*) [source]

决策树（Decision Tree）- code

scikit-learn -V 0.20.2

criterion : string, optional (default="gini")

The function to measure the quality of a split. Supported criteria are "gini" for the Gini impurity and "entropy" for the information gain.

max_depth : int or None, optional (default=None) ≤ 8

The maximum depth of the tree. If None, then nodes are expanded until all leaves are pure or until all leaves contain less than min_samples_split samples.

min_samples_split : int, float, optional (default=2)

The minimum number of samples required to split an internal node:

- If int, then consider min_samples_split as the minimum number.
- If float, then min_samples_split is a percentage and ceil(min_samples_split * n_samples) are the minimum number of samples for each split.

min_samples_leaf : int, float, optional (default=1) 经验上必须大于100

The minimum number of samples required to be at a leaf node:

- If int, then consider min_samples_leaf as the minimum number.
- If float, then min_samples_leaf is a percentage and ceil(min_samples_leaf * n_samples) are the minimum number of samples for each node.

决策树（Decision Tree）- code

scikit-learn -V 0.20.2

min_impurity_decrease : float, optional (default=0.)

A node will be split if this split induces a decrease of the impurity greater than or equal to this value.

The weighted impurity decrease equation is the following:

```
N_t / N * (impurity - N_t_R / N_t * right_impurity
                    - N_t_L / N_t * left_impurity)
```

where `N` is the total number of samples, `N_t` is the number of samples at the current node, `N_t_L` is the number of samples in the left child, and `N_t_R` is the number of samples in the right child.

`N`, `N_t`, `N_t_R` and `N_t_L` all refer to the weighted sum, if `sample_weight` is passed.

min_impurity_split : float,

Threshold for early stopping in tree growth. A node will split if its impurity is above the threshold, otherwise it is a leaf.

决策树（Decision Tree）- code

鸢尾花Iris数据集：

常用的分类实验数据集

数据集包含150个数据样本

样本分为3类，每类50个数据，每个数据包含4个属性

属性：花萼长度，花萼宽度，花瓣长度，花瓣宽度

类别：Setosa，Versicolour，Virginica

dt.iris.ipynb

决策树（Decision Tree）- code

```
In [18]: iris.feature_names
```

```
Out[18]: ['sepal length (cm)',
          'sepal width (cm)',
          'petal length (cm)',
          'petal width (cm)']
```

```
In [20]: iris.data
```

```
Out[20]: array([[ 5.1,  3.5,  1.4,  0.2],
                [ 4.9,  3. ,  1.4,  0.2],
                [ 4.7,  3.2,  1.3,  0.2],
                [ 4.6,  3.1,  1.5,  0.2],
                [ 5. ,  3.6,  1.4,  0.2],
                [ 5.4,  3.9,  1.7,  0.4],
                [ 4.6,  3.4,  1.4,  0.3],
                [ 5. ,  3.4,  1.5,  0.2],
                [ 4.4,  2.9,  1.4,  0.2],
                [ 4.9,  3.1,  1.5,  0.1],
                [ 5.4,  3.7,  1.5,  0.2]
```

决策树（Decision Tree）- code

```
In [19]: iris.target_names
```

```
Out[19]: array(['setosa', 'versicolor', 'virginica'],
               dtype='|S10')
```

```
In [21]: iris.target
```

```
Out[21]: array([0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
                0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
                0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
                1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
                1, 1, 1, 1, 1, 1, 1, 1, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
                2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
                2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2])
```

决策树（Decision Tree）- code

In [20]:
```
from sklearn.datasets import load_iris
from sklearn import tree
iris = load_iris()
```

In [21]:
```
iris.data[:3]
```

Out[21]:
```
array([[ 5.1,  3.5,  1.4,  0.2],
       [ 4.9,  3. ,  1.4,  0.2],
       [ 4.7,  3.2,  1.3,  0.2]])
```

In [22]:
```
iris.target[:3]
```

Out[22]:
```
array([0, 0, 0])
```

In [23]:
```
clf = tree.DecisionTreeClassifier()
clf = clf.fit(iris.data, iris.target)
clf
```

Out[23]:
```
DecisionTreeClassifier(class_weight=None, criterion='gini', max_depth=None,
            max_features=None, max_leaf_nodes=None,
            min_impurity_split=1e-07, min_samples_leaf=1,
            min_samples_split=2, min_weight_fraction_leaf=0.0,
            presort=False, random_state=None, splitter='best')
```

决策树（Decision Tree）- code

sklearn.tree.export_graphviz

sklearn.tree.**export_graphviz**(*decision_tree*, *out_file="tree.dot"*, *max_depth=None*, *feature_names=None*, *class_names=None*, *label='all'*, *filled=False*, *leaves_parallel=False*, *impurity=True*, *node_ids=False*, *proportion=False*, *rotate=False*, *rounded=False*, *special_characters=False*, *precision=3*) [source]

Export a decision tree in DOT format.

决策树（Decision Tree）- code

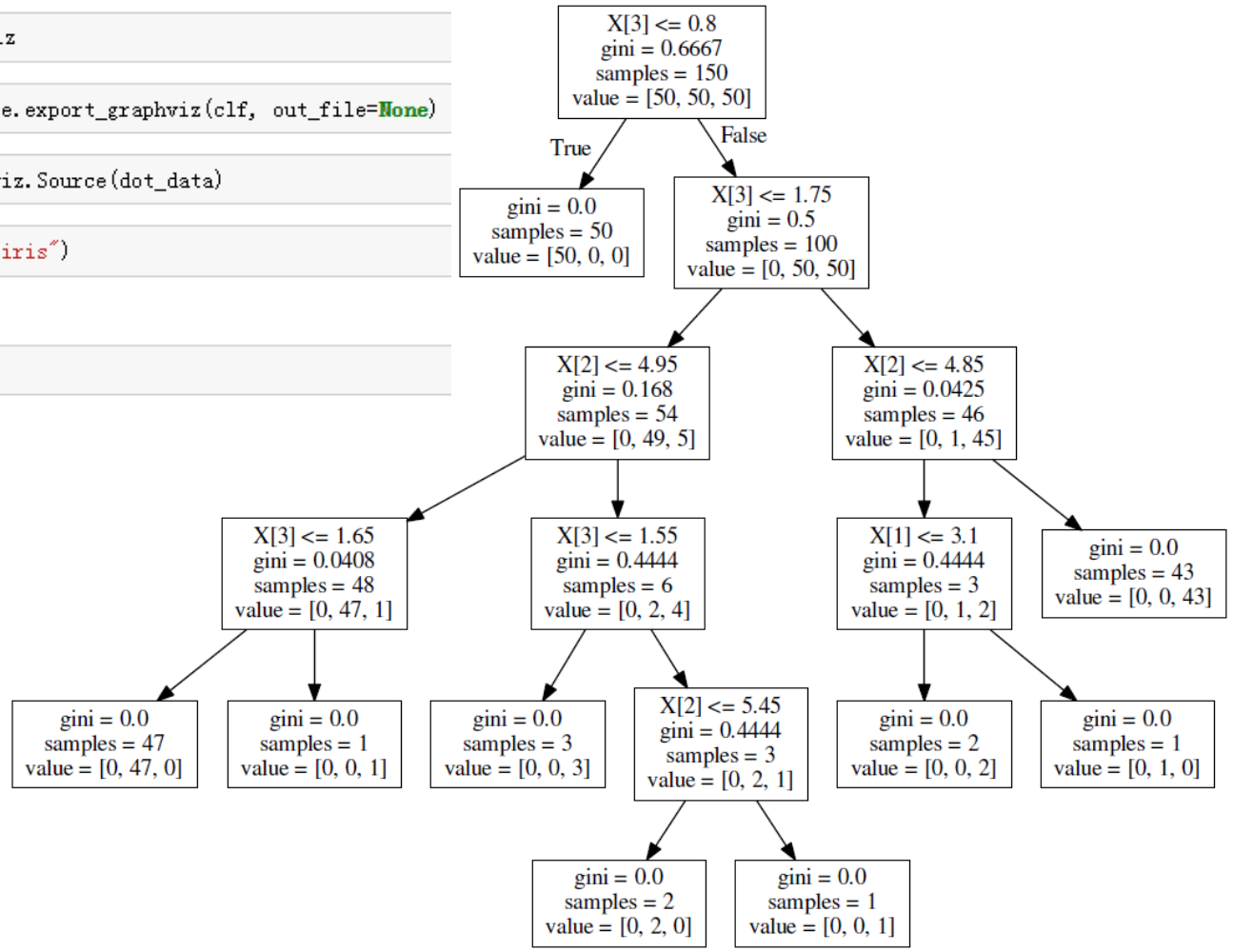
```
In [50]: import graphviz

In [51]: dot_data = tree.export_graphviz(clf, out_file=None)

In [52]: graph = graphviz.Source(dot_data)

In [53]: graph.render("iris")
Out[53]: 'iris.pdf'

In [55]: graph
```



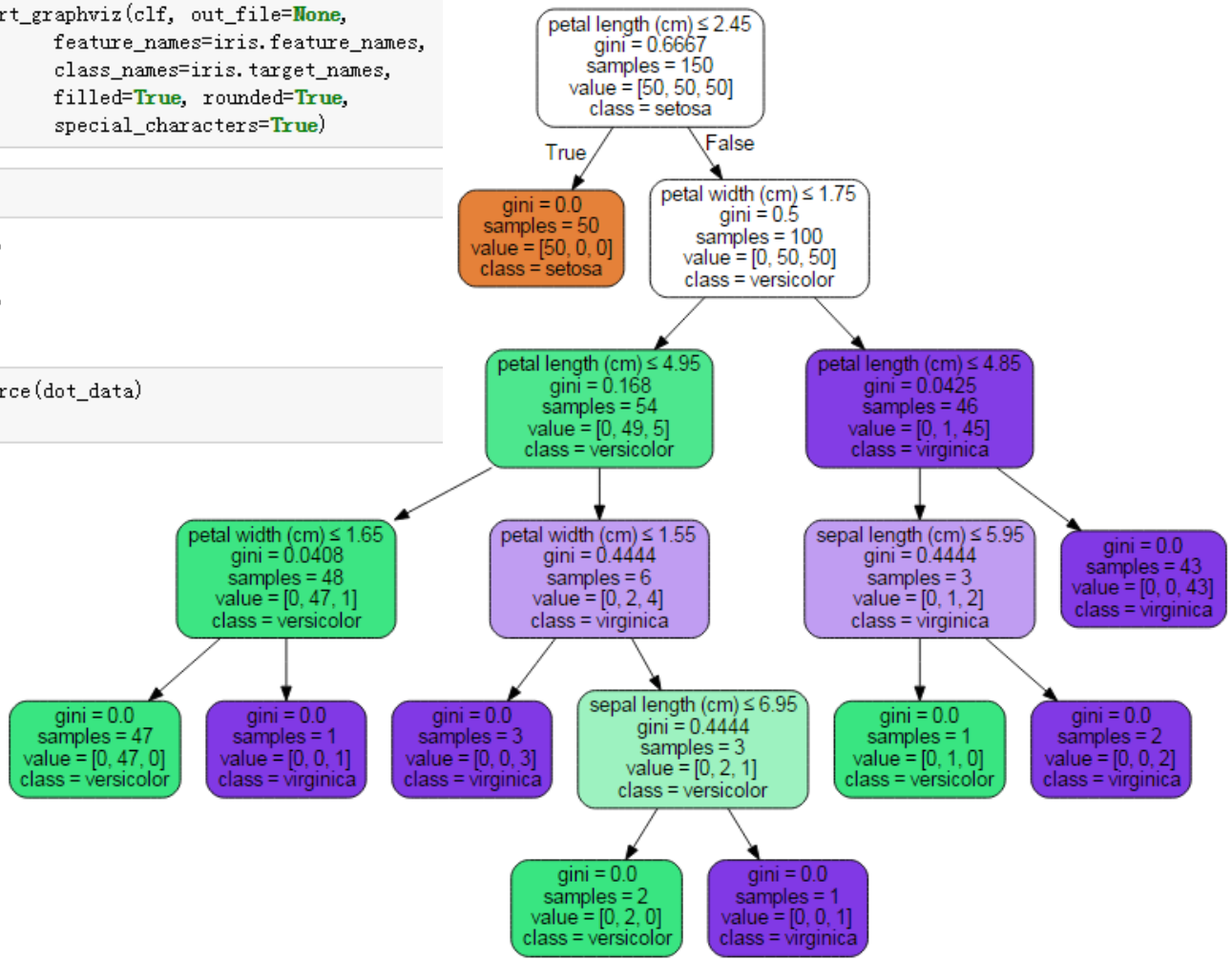
决策树（Decision Tree）- code

```
In [56]: dot_data = tree.export_graphviz(clf, out_file=None,
                                         feature_names=iris.feature_names,
                                         class_names=iris.target_names,
                                         filled=True, rounded=True,
                                         special_characters=True)

In [57]: iris.feature_names

Out[57]: ['sepal length (cm)',
          'sepal width (cm)',
          'petal length (cm)',
          'petal width (cm)']

In [58]: graph = graphviz.Source(dot_data)
         graph
```



决策树（Decision Tree）- code

```
In [45]: from sklearn.externals import joblib

In [46]: joblib.dump(clf, "train_model.m")
Out[46]: ['train_model.m']

In [47]: clf2 = joblib.load("train_model.m")

In [48]: clf2
Out[48]: DecisionTreeClassifier(class_weight=None, criterion='gini', max_depth=None,
                                max_features=None, max_leaf_nodes=None,
                                min_impurity_split=1e-07, min_samples_leaf=1,
                                min_samples_split=2, min_weight_fraction_leaf=0.0,
                                presort=False, random_state=None, splitter='best')

In [49]: clf2.predict(iris.data[:1, :])
Out[49]: array([0])
```

决策树（Decision Tree）- code

```
In [7]: iris.data[:1, :]
Out[7]: array([[ 5.1,  3.5,  1.4,  0.2]])

In [11]: iris.target[:1]
Out[11]: array([0])

In [12]: clf.predict(iris.data[:1, :])
Out[12]: array([0])

In [13]: clf.predict_proba(iris.data[:1, :])
Out[13]: array([[ 1.,  0.,  0.]])
```

决策树（Decision Tree）-Tips

1. 如果**特征数量过多**的话，决策树会更**倾向于过拟合**。找到样本和特征之间合适的**比例**很重要，因为如果拥有一个高维空间的树只有少量的样本，那么它会更倾向于过拟合。

2. 考虑提前用一些**降维**的方法处理一下（比如PCA），可以使树更容易发现那些有区别性的特征

决策树（Decision Tree）-Tips

3. 在训练树的时候可以用export**函数**使树可视化，在初始化树的深度时，设置max_depth = 3，可以感受得到树是如何去适应你的数据的，之后再增加树的深度

4. **树每增加一层**，需要的**样本数是之前的两倍。**使用max_depth来控制数的大小，以防止过拟合

决策树（Decision Tree）-Tips

5. 使用min_samples_split 或者 min_samples_leaf 来控制每个叶子节点样本的数量。虽然一个很小的数量通常意味着树会过拟合，但是一个很大的量会阻碍树从数据中的学习。

尝试 min_samples_leaf = 5 作为初始值。如果样本规模变化幅度很激烈的话，可以在这两个参数中设置一个浮点数作为百分比。

min_samples_leaf 确定了叶子中最小的样本数量

min_samples_split 可以增加一个随意的小叶子

决策树（Decision Tree）-Tips

6. 在训练之前把树调整**平衡**，可以预防树倾向于那些有优势的类别。

可以通过在每个类别中抽取相同数量的样本、或者通过**标准化采样**权重的总和（使得每个类的总和都是一个值），使树更加的平衡。

而且要注意到基于权重的预剪枝准则（比如min_weight_fraction_leaf），相比于那些对采样权重无所谓的准则（比如 min_samples_leaf），可以使树更加不偏向那些有优势的类别

决策树（Decision Tree）-Tips

7. 如果有采样权重，那么通过基于权重的预剪枝准则（比如min_weight_fraction_leaf）可以使优化树结构的过程变的更轻松，刚提到的min_weight_fraction_leaf 可以保证叶子节点至少包含采样权重总和的一小部分

决策树（Decision Tree）-Tips

8. 所有的决策树内部都使用 np.float32 数组，如果训练数据不是这个格式，会生成一个数据集的副本

9. 如果输入矩阵X非常**稀疏**，建议在训练和预测前转换成稀疏的csc_matrix。如果特征再很多样本中都存在零值，相比一个稠密的矩阵来说，稀疏的矩阵可以使训练**时间快**几个数量级

OUTLINES

1. 预备知识
2. 树模型与线性模型
3. 决策树（Decision Tree）
4. 集成学习介绍（Ensemble learning）
5. Bagging - Random Forest
6. Boosting - GBDT

集成学习（Ensemble Learning）

集成学习通过构建并结合**多个学习器**来完成学习任务.只包含**同种类型**的个体学习器，这样的集成是“**同质**”的；包含**不同类型**的个体学习器，这样的集成是“**异质**”的. 集成学习通过将多个学习器进行结合，常可获得比单一学习器显著**优越的泛化性能**.

根据个体学习器的生成方式，目前的集成学习方法大致可分为两大类，即个体学习器间存在**强依赖关系**、必须**串行**生成的**序列化**方法，以及个体学习器间**不存在强依赖关系**、可同时生成的**并行化**方法；前者的代表是Boosting(GBDT)，后者的代表是Bagging (Random Forest)

OUTLINES

1. 预备知识
2. 树模型与线性模型
3. 决策树（Decision Tree）
4. 集成学习介绍（Ensemble learning）
5. Bagging - Random Forest
6. Boosting - GBDT

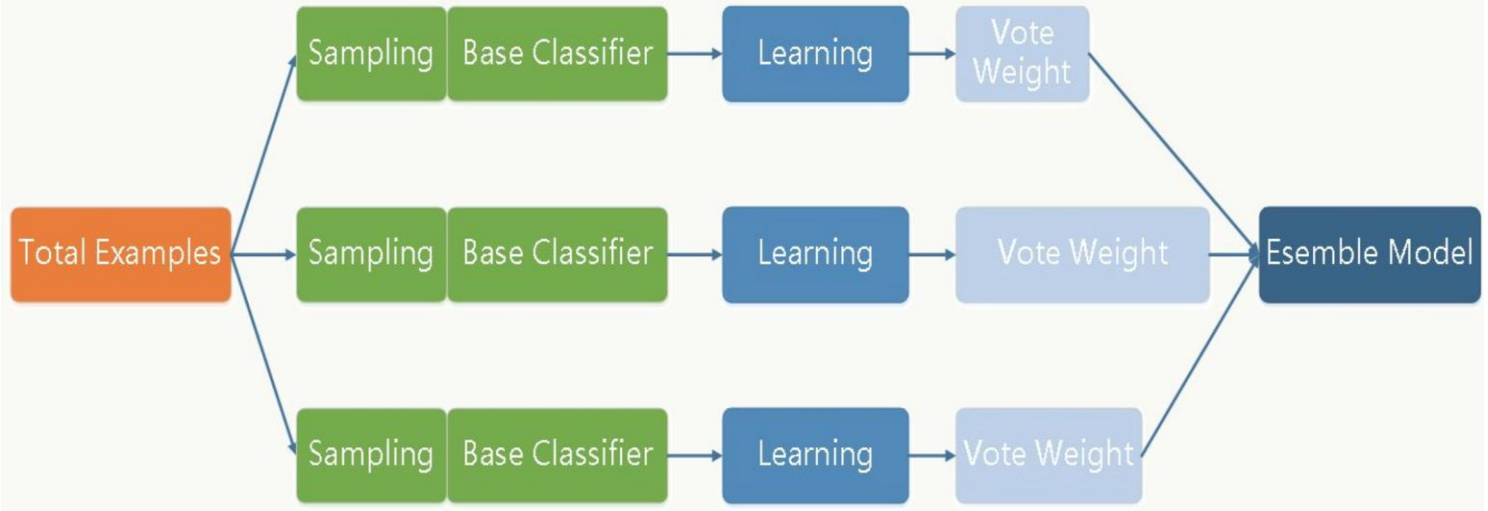
集成学习（Ensemble Learning）-Bagging

Bagging, bootstrap aggregating, **自助法**

采用的是**随机有放回**的选择训练数据然后构造分类器，最后组合

1. 给定一个**弱学习算法**, 和一个**训练集**
2. **单个**弱学习算法**准确率不高**
3. 将该学习算法**使用多次**, 得出**预测函数序列**, 进行**投票**或者**平均**
4. 最后结果准确率将得到提高

集成学习（Ensemble Learning）-Bagging



几个特点：

- 样本选择，训练集是在原始集中**有放回选取**的，从原始集中选出的各轮训练集之间是独立的
- 样例权重，使用**均匀取样**，每个样例的权重相等
- 预测函数，所有**预测函数的权重相等**
- **并行计算**，各个预测函数可以**并行**生成

集成学习（Ensemble Learning）-Bagging-Random Forest

森林会议：这货是松鼠 or 老鼠?

集成学习（Ensemble Learning）-Bagging-Random Forest

随机森林(Random Forest, 以下简称RF)

构造**很多棵决策树**，形成一个森林，然后用这些决策树共同决策

输出类别

有两个随机:

1. 输入数据是随机的从整体的训练数据中选取一部分作为一棵决策

树的构建，而且是有放回的选取；

2. 每棵决策树的构建所需的特征是从整体的特征集随机的选取的，

这两个随机过程使得随机森林很大程度上避免了过拟合现象的出

现。

集成学习（Ensemble Learning）-Bagging-Random Forest

RF构造过程：

Step1-**选样本**: 从训练数据中选取n个数据作为训练数据输入，一般情况下**n是远小于整体的训练数据**N的，这样就会造成有一部分数据是无法被取到，这部分数据称为**袋外数据**，可以使用袋外数据做误差估计。

Step2-**选特征**: 选取了输入的训练数据的之后，需要构建决策树，具体方法是每一个分裂结点从整体的特征集M中选取m个特征构建，一般情况下**m远小于**M。

集成学习（Ensemble Learning）-Bagging-Random Forest

RF构造过程：

Step3-**构建树**: 在构造每棵决策树的过程中，按照选取最小的基尼指数进行分裂节点的选取进行决策树的构建（CART决策树）。决策树的其他结点都采取相同的分裂规则进行构建，直到该节点的所有训练样例都属于同一类或者达到树的最大深度。

Step4-**建森林**: 重复第1、2、3步多次，每一次输入数据对应一颗决策树，这样就得到了随机森林，可以用来对预测数据进行决策。

集成学习（Ensemble Learning）-Bagging-Random Forest

RF构造过程：

Step5-**做决策**: 输入的训练数据选择好了，多棵决策树也构建好了，

对待预测数据进行预测，比如说输入一个待预测数据，然后多棵

决策树同时进行决策，最后采用投票的方式进行类别的决策。

集成学习（Ensemble Learning）-Bagging-Random Forest

RF优点：

- 在数据集上表现良好，两个随机性的引入，使得随机森林**不容易陷入过拟合**，训练出的**模型的方差小**，**泛化能力强**。
- 在当前的很多数据集上，相对其他算法有着很大的优势，两个随机性的引入，使得随机森林对部分**特征缺失不敏感**，具有**很好的抗噪声能力**。
- 它能够处理很**高维度（feature很多）的数据**，并且**不用做特征选择**，对数据集的适应能力强：既能处理离散型数据，也能处理连续型数据，数据集无需规范化。

集成学习（Ensemble Learning）-Bagging-Random Forest

RF优点：

- 在创建随机森林的时候，对generalization error使用的是**无偏估计**。
- 在训练过程中，能够检测到feature间的互相影响。
- 训练**速度快**，可以得到变量重要性排序。
- 容易做成**并行化**方法，对于大数据时代的大样本训练速度有优势。
- 实现比较**简单**。

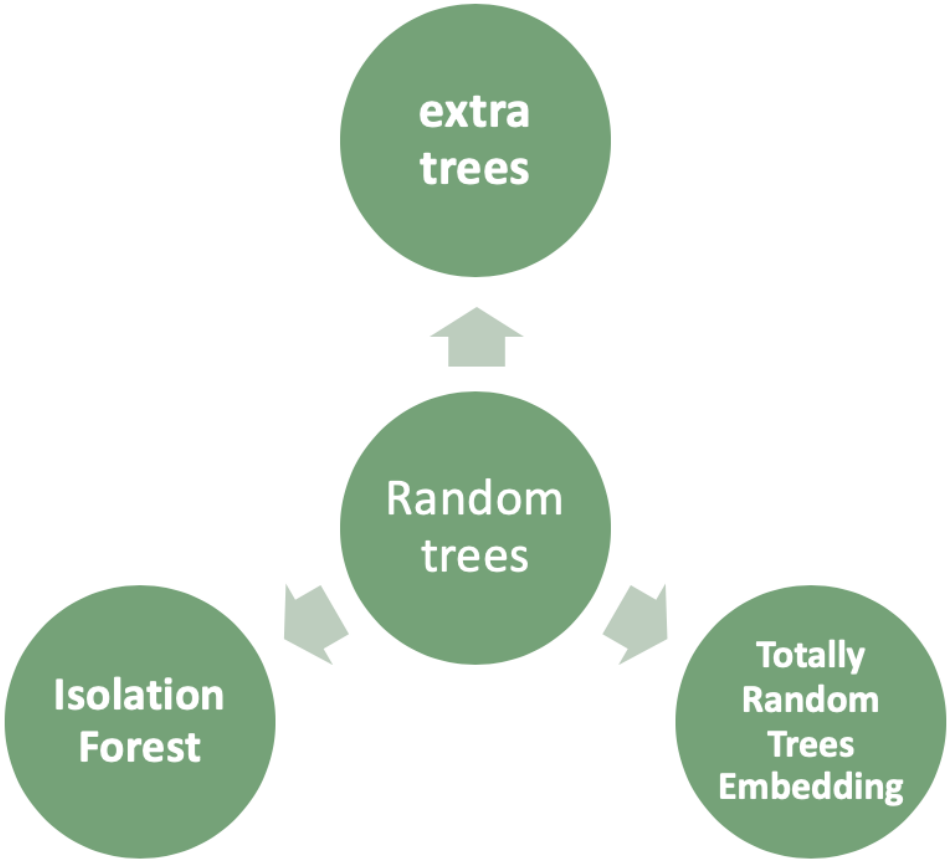
集成学习（Ensemble Learning）-Bagging-Random Forest

RF缺点：

- 在某些噪音比较大的样本集上，RF模型容易陷入过拟合。
- 取值划分比较多的特征容易对RF的决策产生更大的影响，

集成学习（Ensemble Learning）-Bagging-Random Forest

RF变种：



集成学习（Ensemble Learning）-Bagging-Random Forest

RF变种-extra trees：是RF的一个变种，原理几乎和RF一模一样，

仅有区别有：

1. 对于每个决策树的训练集，RF采用的是随机采样bootstrap来选择采样集作为每个决策树的训练集，而extra trees一般**不采用随机采样**，即每个决策树采用原始训练集。

集成学习（Ensemble Learning）-Bagging-Random Forest

RF变种-extra trees：是RF的一个变种，原理几乎和RF一模一样，仅有区别有：

2. 在选定了划分特征后，RF的决策树会基于信息增益，基尼系数，均方差之类的原则，选择一个最优的特征值划分点。但是extra trees比较的激进，他会**随机的选择一个特征值**来划分决策树。（extra trees的泛化能力比RF更好）

集成学习（Ensemble Learning）-Bagging-Random Forest

RF变种-Totally Random Trees Embedding (TRTE)

：一种**非监督学习的数据转化方法**，将**低维的数据集映射到高维**

（类似支持向量机中的核方法）

1. TRTE在数据转化的过程也使用了类似于RF的方法，建立T个决策树来拟合数据。当决策树建立完毕以后，数据集里的每个数据在T个决策树中叶子节点的位置也定下来了，从而可以组合生成高维特征

集成学习（Ensemble Learning）-Bagging-Random Forest

RF变种-Totally Random Trees Embedding (TRTE)

：一种非监督学习的数据转化方法，将低维的数据集映射到高维

（类似支持向量机中的核方法）

Eg: 有3颗决策树，每个决策树有5个叶子节点，

某个数据特征x划分到第一个决策树的第2个叶子节点，第二个决策树的第3个叶子节点，第三个决策树的第5个叶子节点。

则x映射后的特征编码为(0,1,0,0,0, 0,0,1,0,0, 0,0,0,0,1), 有15维的高维特征。

集成学习（Ensemble Learning）-Bagging-Random Forest

3.2.4.3.1. sklearn.ensemble.RandomForestClassifier

class sklearn.ensemble. **RandomForestClassifier** (*n_estimators=10, criterion='gini', max_depth=None, min_samples_split=2, min_samples_leaf=1, min_weight_fraction_leaf=0.0, max_features='auto', max_leaf_nodes=None, min_impurity_decrease=0.0, min_impurity_split=None, bootstrap=True, oob_score=False, n_jobs=1, random_state=None, verbose=0, warm_start=False, class_weight=None*) ¶ [source]

Generate a random n-class classification problem.

```
>>> from sklearn.ensemble import RandomForestClassifier
>>> from sklearn.datasets import make_classification
>>>
>>> X, y = make_classification(n_samples=1000, n_features=4,
...                            n_informative=2, n_redundant=0,
...                            random_state=0, shuffle=False)
>>> clf = RandomForestClassifier(max_depth=2, random_state=0)
>>> clf.fit(X, y)
RandomForestClassifier(bootstrap=True, class_weight=None, criterion='gini',
            max_depth=2, max_features='auto', max_leaf_nodes=None,
            min_impurity_decrease=0.0, min_impurity_split=None,
            min_samples_leaf=1, min_samples_split=2,
            min_weight_fraction_leaf=0.0, n_estimators=10, n_jobs=1,
            oob_score=False, random_state=0, verbose=0, warm_start=False)
>>> print(clf.feature_importances_)
[ 0.17287856  0.80608704  0.01884792  0.00218648]
>>> print(clf.predict([[0, 0, 0, 0]]))
[1]
```

集成学习（Ensemble Learning）-Bagging-Random Forest

```
# random forest
from sklearn.ensemble import RandomForestClassifier
```

```
rf = RandomForestClassifier()
```

```
rf.fit(iris.data, iris.target)
```

```
/Users/meizu/Work/py_env/my_scikit_learn/lib/python2.7/site-packages/sklearn/ensemble/forest.py:246: FutureWarning: T
he default value of n_estimators will change from 10 in version 0.20 to 100 in 0.22.
  "10 in version 0.20 to 100 in 0.22.", FutureWarning)
```

```
RandomForestClassifier(bootstrap=True, class_weight=None, criterion='gini',
            max_depth=None, max_features='auto', max_leaf_nodes=None,
            min_impurity_decrease=0.0, min_impurity_split=None,
            min_samples_leaf=1, min_samples_split=2,
            min_weight_fraction_leaf=0.0, n_estimators=10, n_jobs=None,
            oob_score=False, random_state=None, verbose=0,
            warm_start=False)
```

```
rf.predict(iris.data[100:109,:])
```

```
array([2, 2, 2, 2, 2, 2, 1, 2, 2])
```

```
iris.target[100:109]
```

```
array([2, 2, 2, 2, 2, 2, 2, 2, 2])
```

集成学习（Ensemble Learning）-Bagging-Random Forest

3.2.4.3.1. sklearn.ensemble.RandomForestClassifier

class sklearn.ensemble. **RandomForestClassifier** (*n_estimators=10, criterion='gini', max_depth=None, min_samples_split=2, min_samples_leaf=1, min_weight_fraction_leaf=0.0, max_features='auto', max_leaf_nodes=None, min_impurity_decrease=0.0, min_impurity_split=None, bootstrap=True, oob_score=False, n_jobs=1, random_state=None, verbose=0, warm_start=False, class_weight=None*) ¶ [source]

sklearn.ensemble.RandomTreesEmbedding

class sklearn.ensemble. **RandomTreesEmbedding** (*n_estimators=10, max_depth=5, min_samples_split=2, min_samples_leaf=1, min_weight_fraction_leaf=0.0, max_leaf_nodes=None, min_impurity_decrease=0.0, min_impurity_split=None, sparse_output=True, n_jobs=1, random_state=None, verbose=0, warm_start=False*) [source]

sklearn.ensemble.IsolationForest

class sklearn.ensemble. **IsolationForest** (*n_estimators=100, max_samples='auto', contamination=0.1, max_features=1.0, bootstrap=False, n_jobs=1, random_state=None, verbose=0*) [source]

决策树（Decision Tree）- code

sklearn.tree.ExtraTreeClassifier

class sklearn.tree. **ExtraTreeClassifier** (*criterion='gini', splitter='random', max_depth=None, min_samples_split=2, min_samples_leaf=1, min_weight_fraction_leaf=0.0, max_features='auto', random_state=None, max_leaf_nodes=None, min_impurity_decrease=0.0, min_impurity_split=None, class_weight=None*) [source]

Warning: Extra-trees should only be used within ensemble methods.

max_features : int, float, string or None, optional (default=None)

The number of features to consider when looking for the best split:

- If int, then consider max_features features at each split.
- If float, then max_features is a percentage and int(max_features * n_features) features are considered at each split.
- If “auto”, then max_features=sqrt(n_features).
- If “sqrt”, then max_features=sqrt(n_features).
- If “log2”, then max_features=log2(n_features).
- If None, then max_features=n_features.

当max_features =1时，extra Tree和random tree一样

随机森林算法 —— 代码实现

In [1]:
```
#random forest test
from sklearn.model_selection import cross_val_score
from sklearn.datasets import make_blobs
from sklearn.ensemble import RandomForestClassifier
from sklearn.ensemble import ExtraTreesClassifier
from sklearn.tree import DecisionTreeClassifier
```

In [6]:
```
X, y = make_blobs(n_samples=10000, n_features=10, centers=100,random_state=0)
X[:1],y[:1]
```

Out[6]:
```
(array([[ 6.46907649,  4.25070317, -8.63694437,  4.04478517,  9.01725363,
          4.53587229, -4.67027643, -0.48172814, -6.44996141, -2.65984972]]),
 array([85]))
```

In [7]:
```
clf = DecisionTreeClassifier(max_depth=None, min_samples_split=2,random_state=0)
scores = cross_val_score(clf, X, y)
print(scores.mean())
```

```
0.979408793821
```

In [8]:
```
clf = RandomForestClassifier(n_estimators=10, max_depth=None,min_samples_split=2, random_state=0)
scores = cross_val_score(clf, X, y)
print(scores.mean())
```

```
0.999607843137
```

In [9]:
```
clf = ExtraTreesClassifier(n_estimators=10, max_depth=None,min_samples_split=2, random_state=0)
scores = cross_val_score(clf, X, y)
print(scores.mean())
```

```
0.999898989899
```

OUTLINES

1. 预备知识
2. 树模型与线性模型
3. 决策树（Decision Tree）
4. 集成学习介绍（Ensemble learning）
5. Bagging - Random Forest
6. Boosting - GBDT

集成学习（Ensemble Learning）

集成学习通过构建并结合**多个学习器**来完成学习任务.只包含**同种类型**的个体学习器，这样的集成是"**同质**"的；包含**不同类型**的个体学习器，这样的集成是"**异质**"的. 集成学习通过将多个学习器进行结合，常可获得比单一学习器显著**优越的泛化性能**.

根据个体学习器的生成方式，目前的集成学习方法大致可分为两大类，即个体学习器间存在**强依赖关系**、必须**串行**生成的**序列化**方法，以及个体学习器间**不存在强依赖关系**、可同时生成的**并行化**方法；前者的代表是Boosting(GBDT)，后者的代表是Bagging (Random Forest)

集成学习（Ensemble Learning）-Boosting-GBDT

GBDT，Gradient Boosting Decision Tree，**梯度提升**决策树

别名很多：

- GBT（Gradient Boosting Tree）
- GTB（Gradient Tree Boosting ）
- GBRT（Gradient Boosting Regression Tree）
- MART(Multiple Additive Regression Tree)

集成学习（Ensemble Learning）-Boosting-GBDT

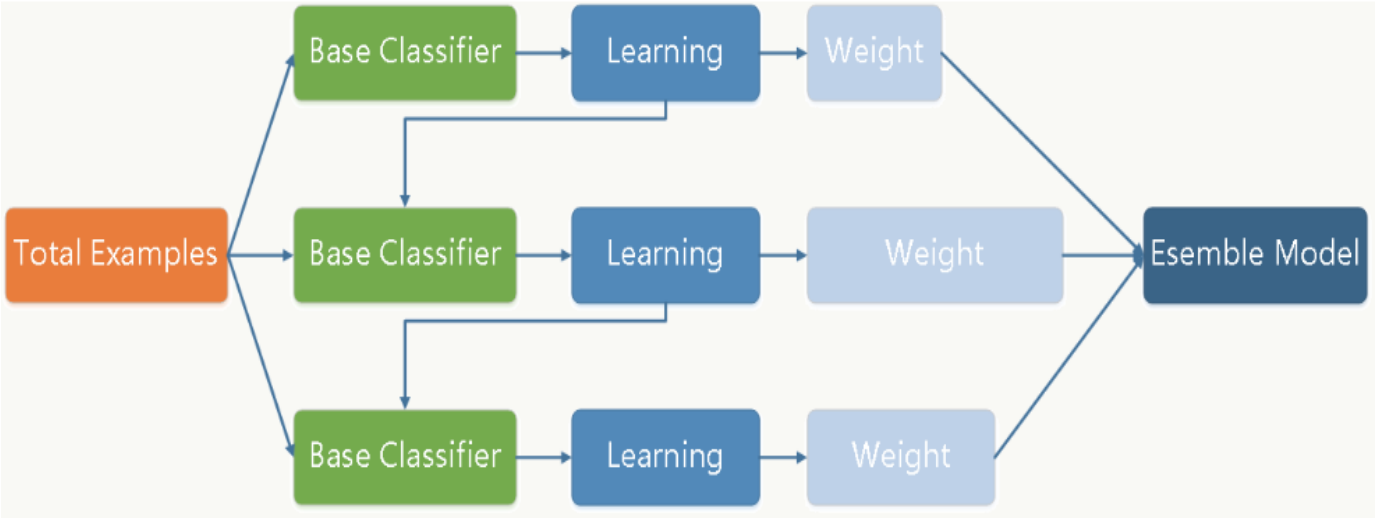
Boosting，集成学习方法之一

一种用来提高弱分类算法准确度的方法，属于迭代算法

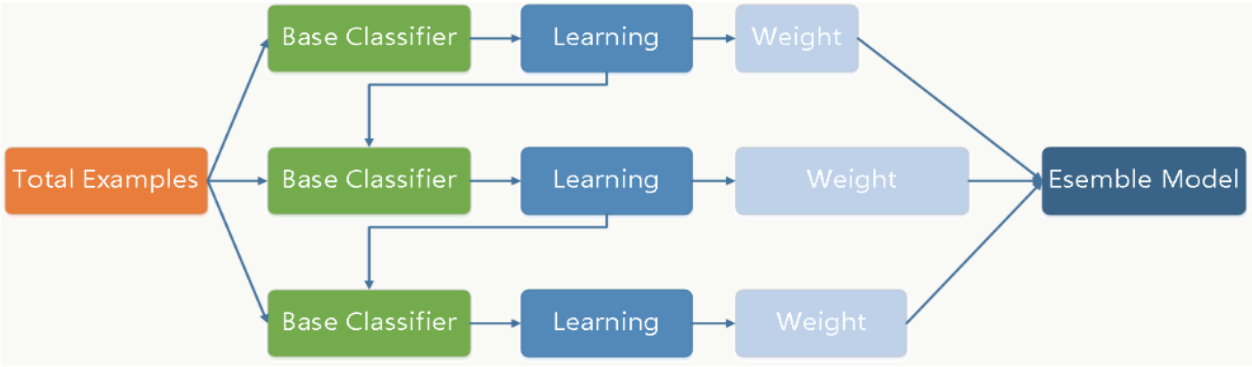
通过不断地使用一个弱学习器**弥补**前一个弱学习器的“**不足**”的过程，

来**串行**地构造一个较强的学习器，能够使目标函数值足够小

集成学习（Ensemble Learning）-Boosting-GBDT



集成学习（Ensemble Learning）-Boosting-GBDT



Step1: 先用一个初始值来学习一棵决策树

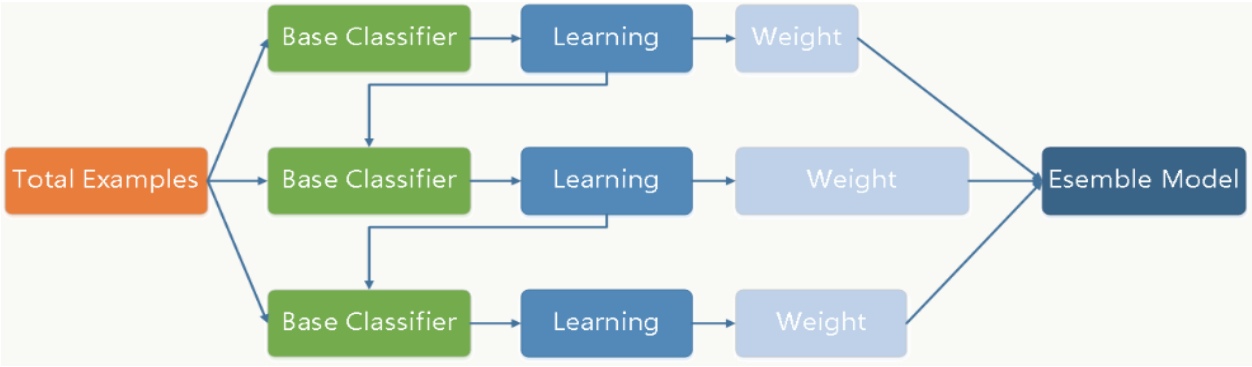
Step2: 叶子处可以得到预测的值，以及预测之后的**残差**

Step3: 然后后面的决策树就要基于前面决策树的残差来学习

Step4: 直到**预测值**和**真实值**的**残差为零**

Step5: 最后对于测试样本的预测值，就是前面许多棵决策树**预测值的累加**

集成学习（Ensemble Learning）-Boosting-GBDT



预测年龄问题

数据集：

用户ID	消费金额	百度提问/回答	年龄
A	800	提问	14
B	500	回答	16
C	1200	提问	24
D	3000	回答	26
E	3000	提问	?

集成学习（Ensemble Learning）-Boosting-GBDT

用户ID	消费金额	百度提问/回答	年龄
A	800	提问	14
B	500	回答	16
C	1200	提问	24
D	3000	回答	26
E	3000	提问	?

训练数据的均值：20岁

（这个很重要，因为一开始需要设置预测的**均值**，这样后面才会有残差）

每个样本的特征有两个：

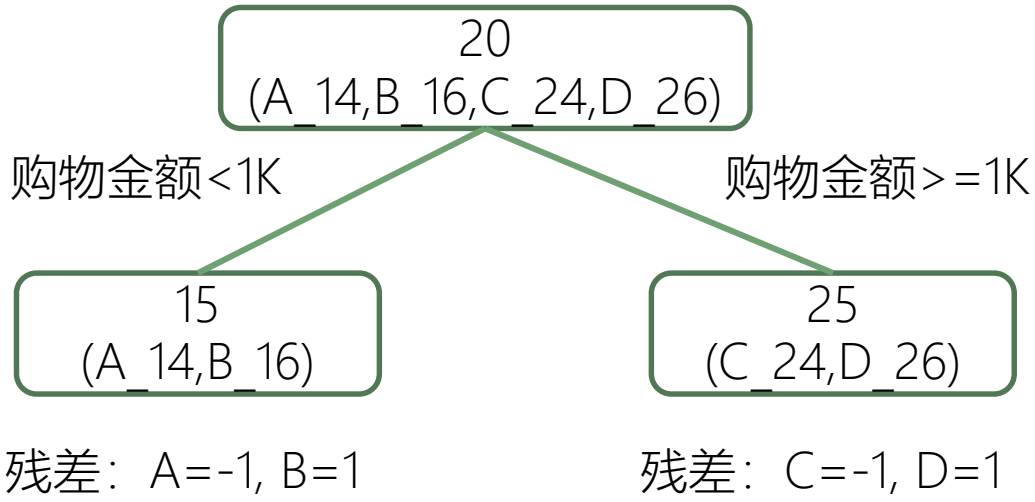
- 购物金额是否小于1K
- 经常去百度提问还是回答

集成学习（Ensemble Learning）-Boosting-GBDT

预测年龄问题

训练环节

首先，输入初值20岁，根据第一个特征（具体选择哪些特征可以根据信息增益来计算选择，假设是“购物金额”），可以把4个样本分成两类



集成学习（Ensemble Learning）-Boosting-GBDT

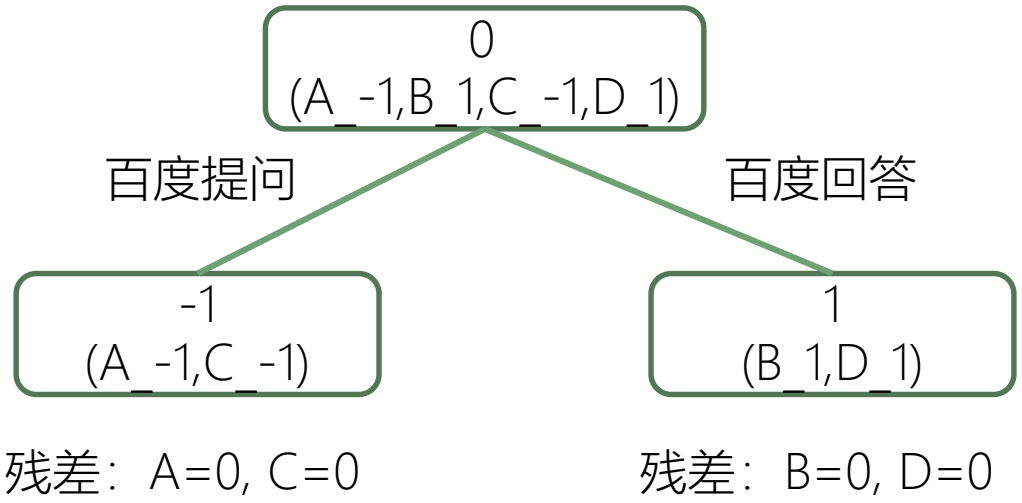
预测年龄问题

训练环节

学习第二棵决策树，我们把第一棵的残差样本（A, -1岁）、（B，1岁）、

（C, -1岁）、（D, 1岁）输入，并将残差均值作为根节点

残差总和为0，停止学习



集成学习（Ensemble Learning）-Boosting-GBDT

预测年龄问题

测试环节

测试样本：请预测一个**购物金额为3k，经常去百度问**淘宝相关问题的女

生的年龄

我们提取2个特征：购物金额3k，经常去百度上面问问题

第一棵树 —> 购物金额大于1k —> 右叶子，初步说明这个女生25岁

第二棵树 —> 经常去百度提问 —> 左叶子，说明这个女生的残差为-1

叠加前面每棵树得到的结果：25-1=24岁，**最终预测结果为24岁**

集成学习（Ensemble Learning）-Boosting-GBDT

GBDT for classification?

总结

1. 预备知识：Entropy、 Variance 、Bias
2. **树**模型与**线性**模型
3. 决策树（Decision Tree）：ID3、C4.5、CART
4. 集成学习介绍（Ensemble learning)：Bagging、Boosting
5. Bagging - Random Forest
6. Boosting - GBDT

Thanks!