分类和回归
Classification and Regression

分类和回归

- 3.1 分类和回归的定义
- 3.2 分类算法
  - 决策树
  - 集成学习方法
  - KNN
  - SVM
  - 朴素贝叶斯
  - 神经网络
- 3.3 回归算法
  - 线性回归
  - 非线性回归

分类和回归

- 3.1 分类和回归的定义
- 3.2 分类算法
  - 决策树
  - 集成学习方法
  - KNN
  - SVM
  - 朴素贝叶斯
  - 神经网络
- 3.3 回归算法
  - 线性回归
  - 非线性回归

3.1 分类和回归的定义

- 分类（Classification）
  - 给定一个数据集$D=\{t_1, t_2, \ldots, t_n\}$和一个类别集合$C=\{C_1, C_2, \ldots, C_m\}$，**数据分类**就是通过定义一个映射$f: D \rightarrow C$，为数据集$D$中的每条数据$t_i$分配$C$中的一个类$C_j$。
- 回归（Regression）
  - 可以视为一种特殊的分类，当分类的类别是一个连续值时（可看成无限多类），就是**回归**。

3.1 分类和回归的定义——示例

- 分类
  - 银行贷款员需要分析数据，来弄清哪些贷款申请者是安全的，哪些是有风险的。
  - 构造一个映射（**模型**）将申请者分为两类：
    - 安全
    - 有风险
- 回归
  - 银行贷款员需要分析数据，来预测贷给某个顾客多少钱是安全的。
  - 构造一个映射（**模型**）来预测一个连续值。

如何建立具体的映射（模型）？

3.1 分类和回归的定义

- 数据分类和预测的步骤如下：
  - 第一步——**建立模型**
  - 第二步——**使用模型**
- 下面以分类为例，详细介绍这两个步骤。

3.1 分类和回归的定义

- 第一步——**建立模型**
  - 训练数据集：由若干数据（通常用n维属性向量表示）和它们相对应的类标号组成。
    - 训练样本：训练数据集中的单个数据及其类标号。
  - 从训练数据集“学习”相关知识来构造分类模型。
  - 分类模型可能会以分类规则、决策树或数学公式等形式呈现出来。
- 第二步——**使用模型**
  - 对未知类别的数据进行分类（分配类别标号）。

第一步——建立模型

训练数据集

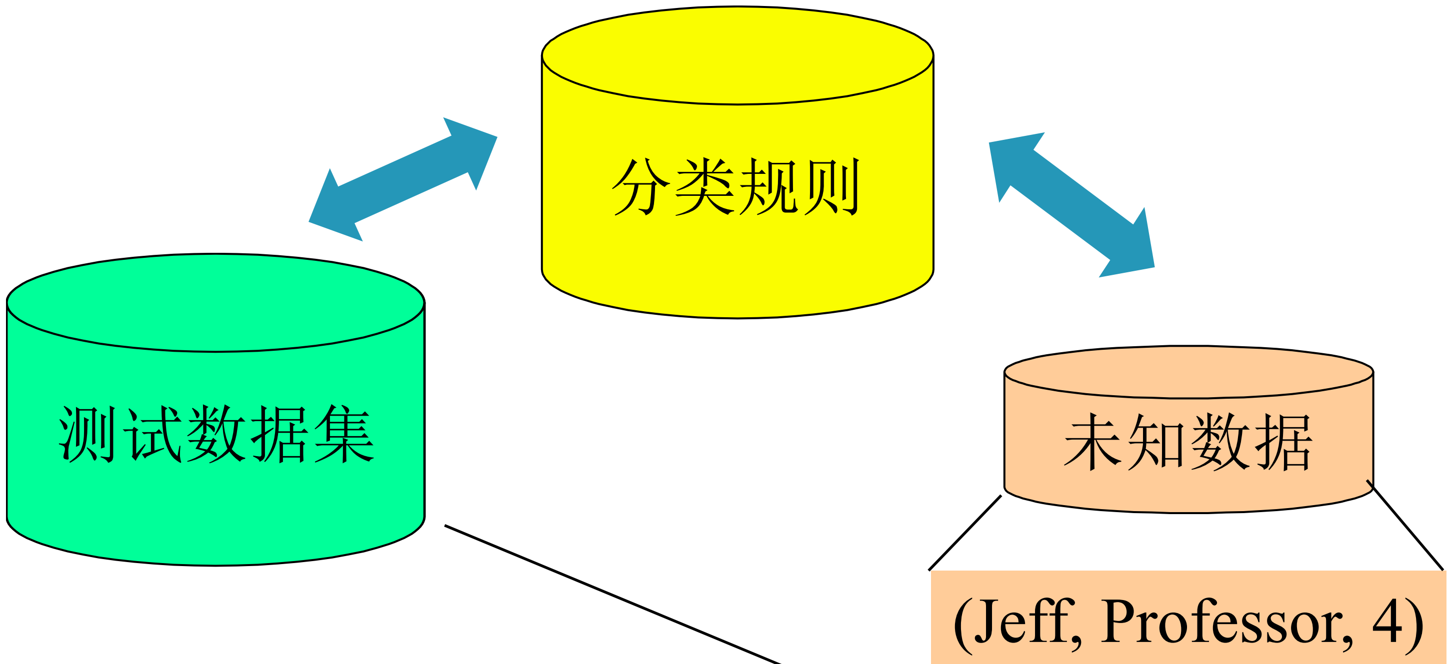
分类模型

分类规则

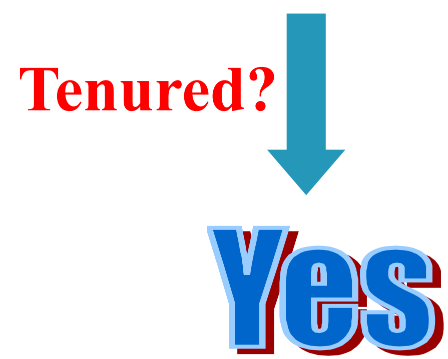
NAME	RANK	YEARS	TENURED
Mike	Assistant Prof	3	no
Mary	Assistant Prof	7	yes
Bill	Professor	2	yes
Jim	Associate Prof	7	yes
Dave	Assistant Prof	6	no
Anne	Associate Prof	3	no

IF rank = ‘professor’
OR years > 6
THEN tenured = ‘yes’

第二步——使用模型



NAME	RANK	YEARS	TENURED
Tom	Assistant Prof	2	no
Merlisa	Associate Prof	7	no
George	Professor	5	yes
Joseph	Assistant Prof	7	yes

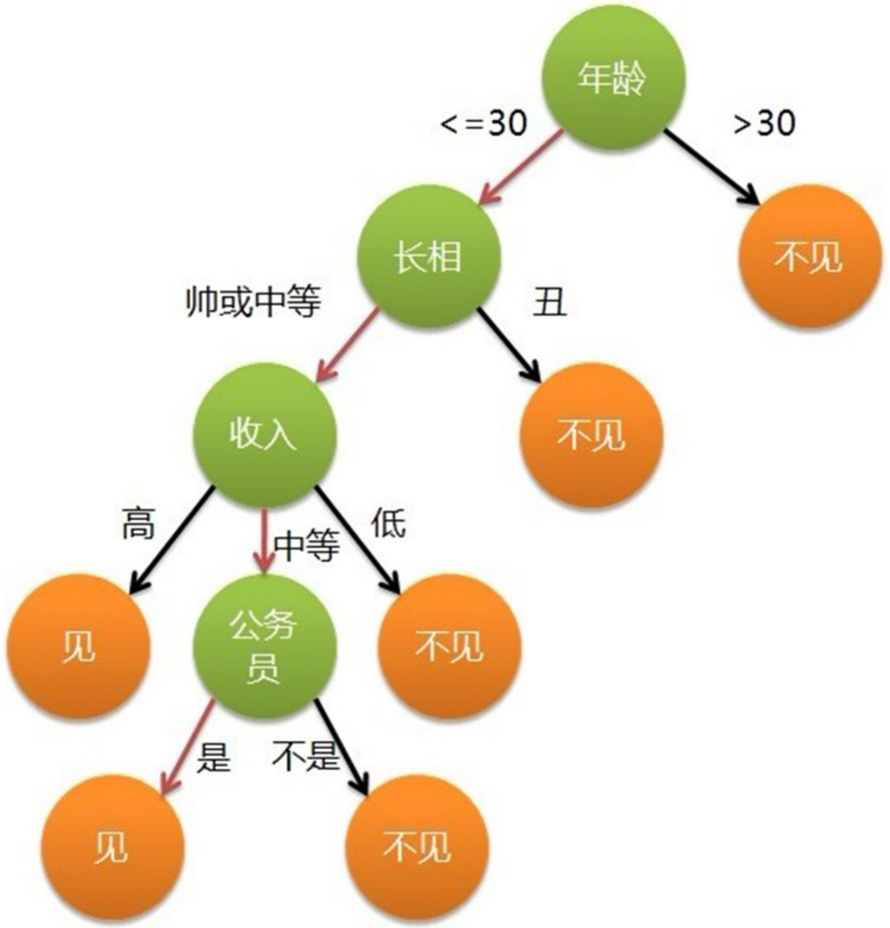


分类和回归

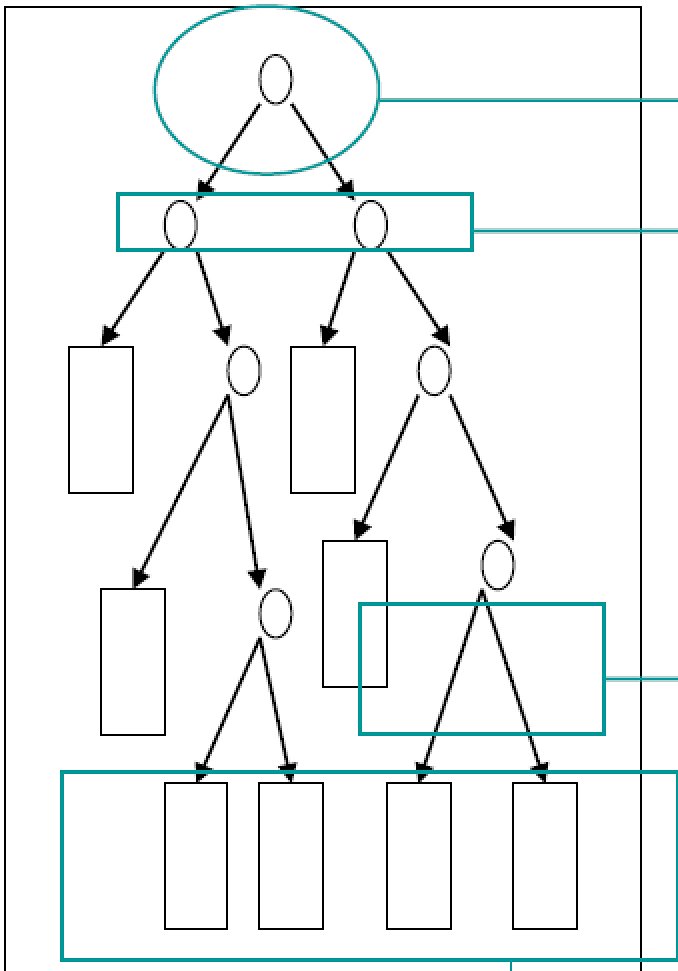
- 3.1 分类和回归的定义
- 3.2 分类算法
  - 决策树
  - 集成学习方法
  - KNN
  - SVM
  - 朴素贝叶斯
  - 神经网络
- 3.3 回归算法
  - 线性回归
  - 非线性回归

决策树

- **什么是决策树？**
  - 由数据的不同属性逐次划分数据集，直至得到的数据子集只包含同一类数据（或达到某种限制条件）为止，这样形成的一种树型结构，称为决策树。
    - 结构上类似于一个流程图
    - 每个内部结点（绿色）表示在一个属性上的测试；
    - 每个分枝（黑线箭头）代表一个测试结果的输出；
    - 每个叶结点存放一个类标号（黄色）。



决策树



根节点 (root node)

非叶子节点 (non-leaf node)

(代表测试条件，对数据属性的测试)

分支 (branches) (代表测试结果)

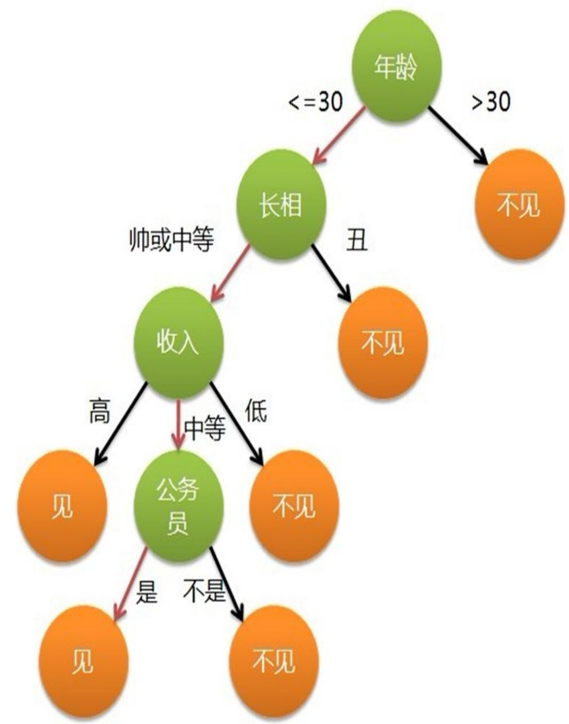
叶节点 (leaf node)

(代表分类后所获得的分类标记)

决策树

什么是决策树？

- 由树的根结点到某个叶结点的属性的**合取**可形成一条分类规则；所有规则的**析取**可形成一整套分类规则。
  - **IF 年龄 = “>30” THEN 见面 = “no”**
  - **IF 年龄 = “<=30” AND 长相 = “丑” THEN 见面= “no”**
  - **IF 年龄 = “<=30” AND 长相= “不丑” AND 收入 = “低” THEN 见面 = “no”**
  - **IF 年龄 = “<=30” AND 长相= “不丑” AND 收入 = “高” THEN 见面 = “yes”**
- 本质上，决策树是**一组if-then规则的集合**，其算法运行过程就是通过一系列规则对数据进行分类的过程。



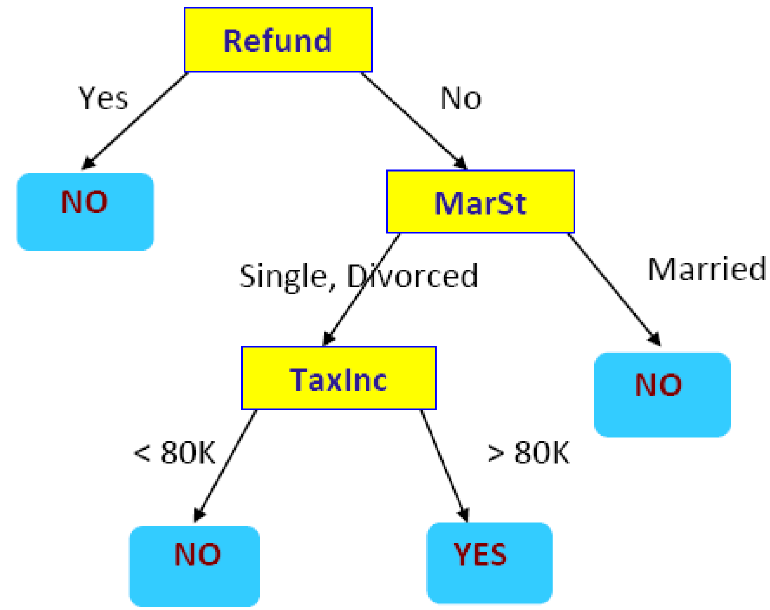
决策树

categorical categorical continuous class

Tid	Refund	Marital Status	Taxable Income	Cheat
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

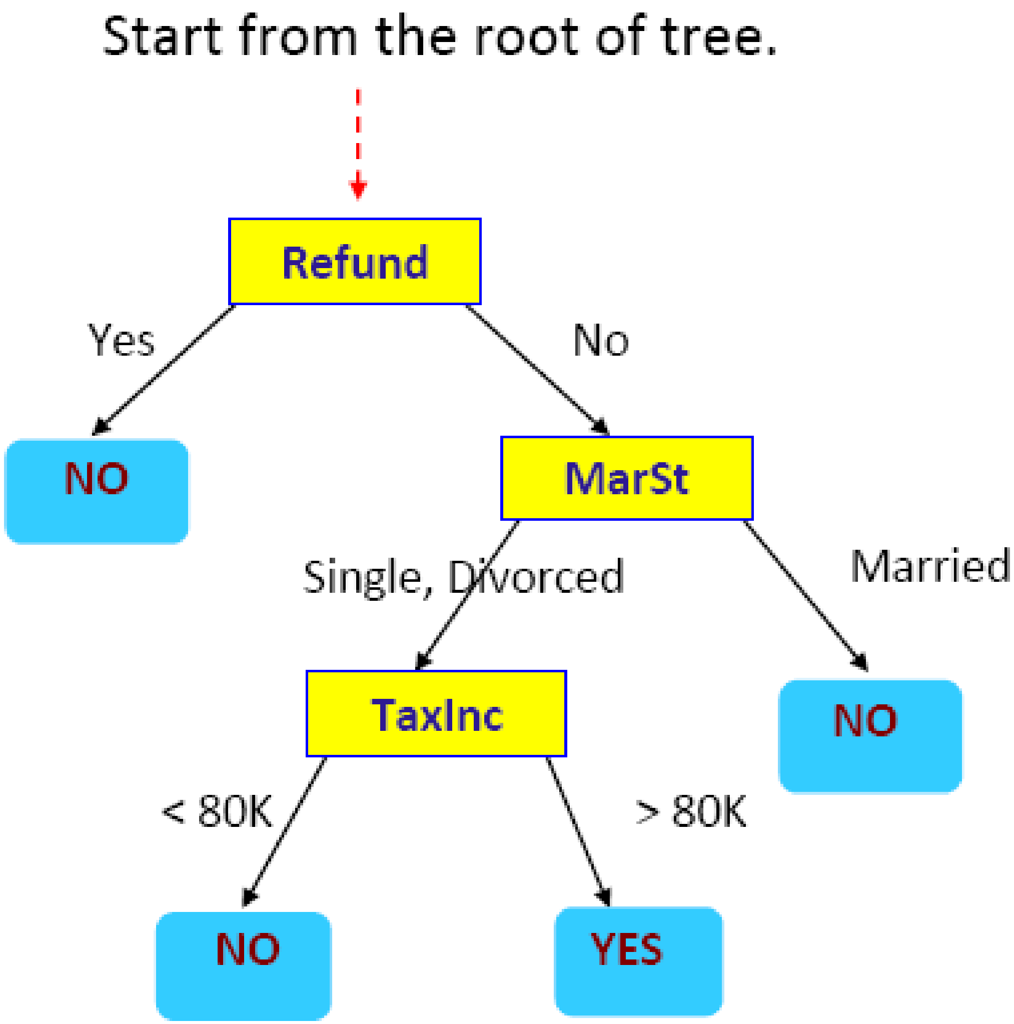
生成

如何从训练数据集生成相应决策树，是本节所关注的内容。



目标：根据客户的如下属性，是否还款、婚姻状况、收入水平，来判断客户是否存在“金融欺骗”行为。

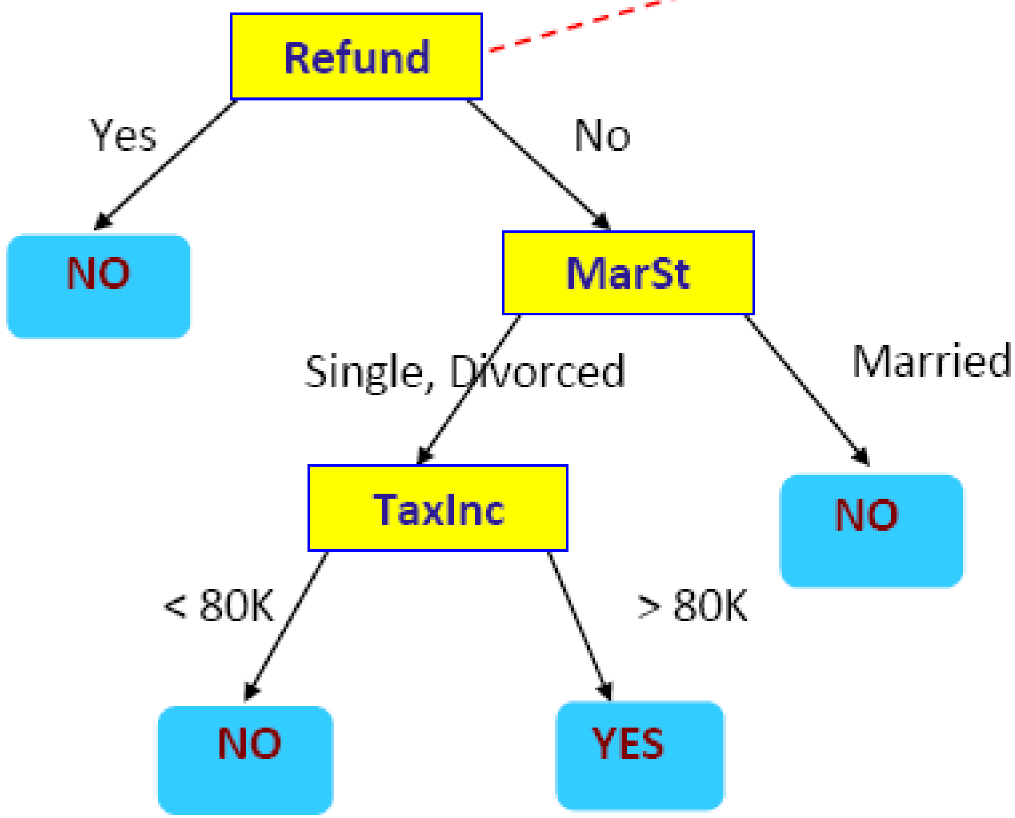
决策树——分类过程



Test Data

Refund	Marital Status	Taxable Income	Cheat
No	Married	80K	?

决策树——分类过程



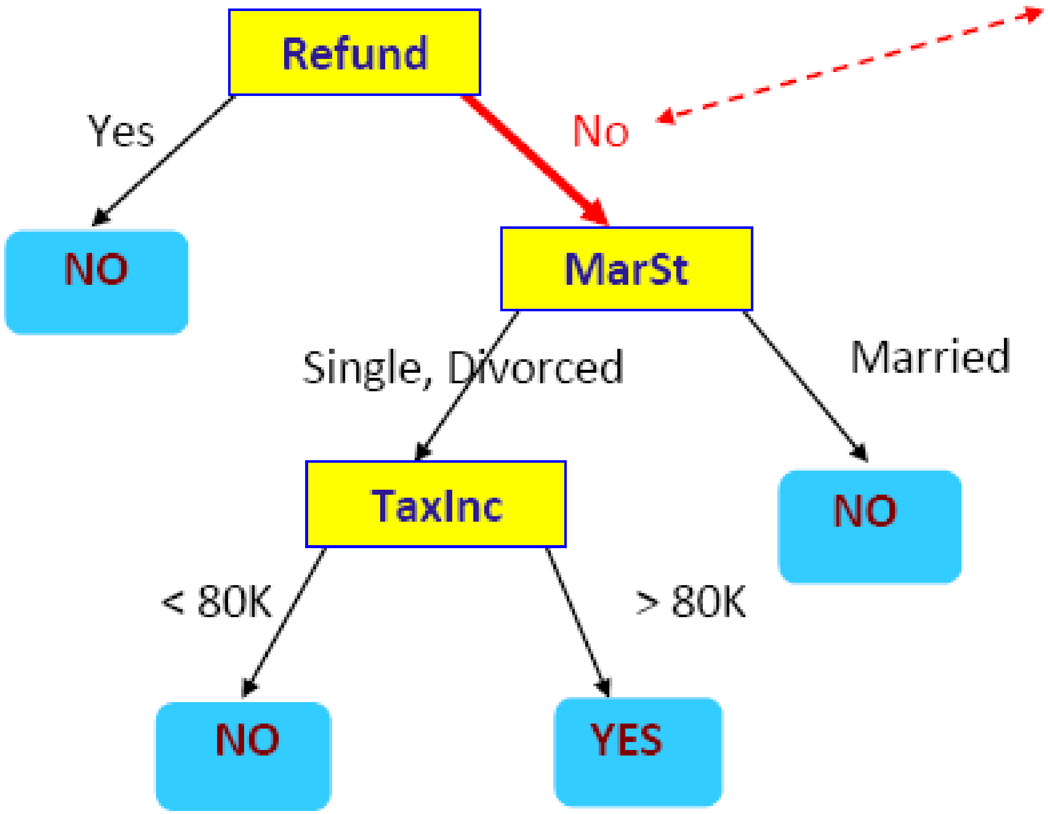
Test Data

Refund	Marital Status	Taxable Income	Cheat
No	Married	80K	?

决策树——分类过程

Test Data

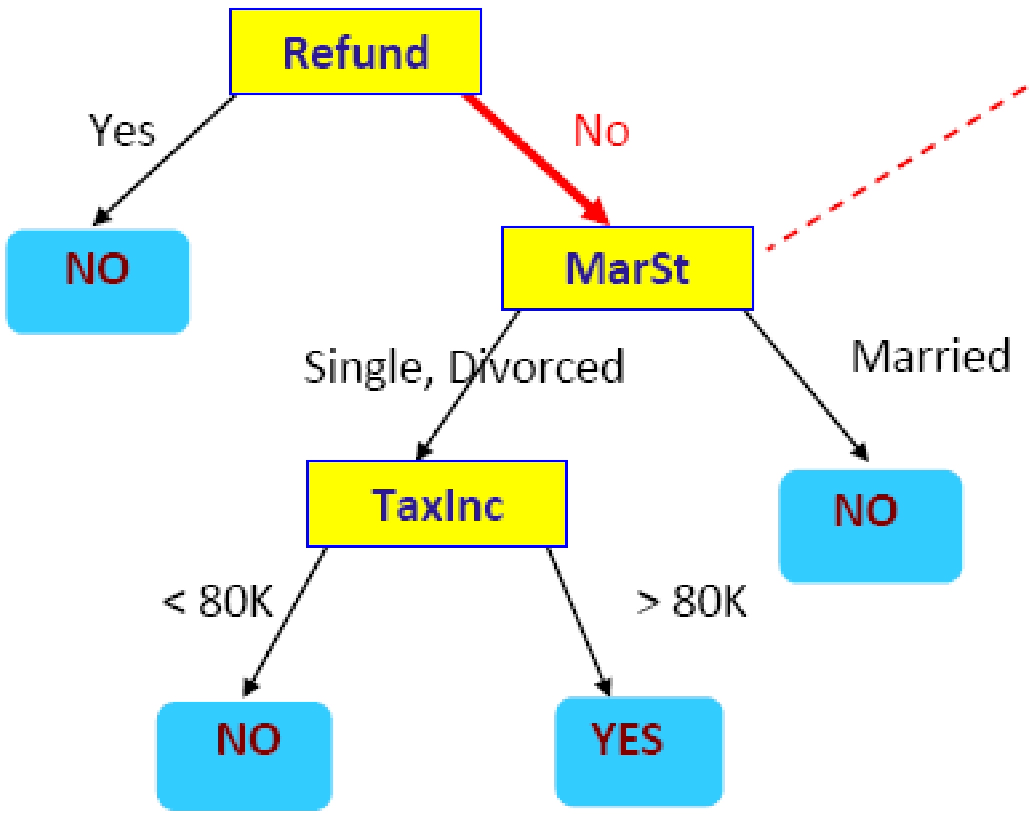
Refund	Marital Status	Taxable Income	Cheat
No	Married	80K	?



决策树——分类过程

Test Data

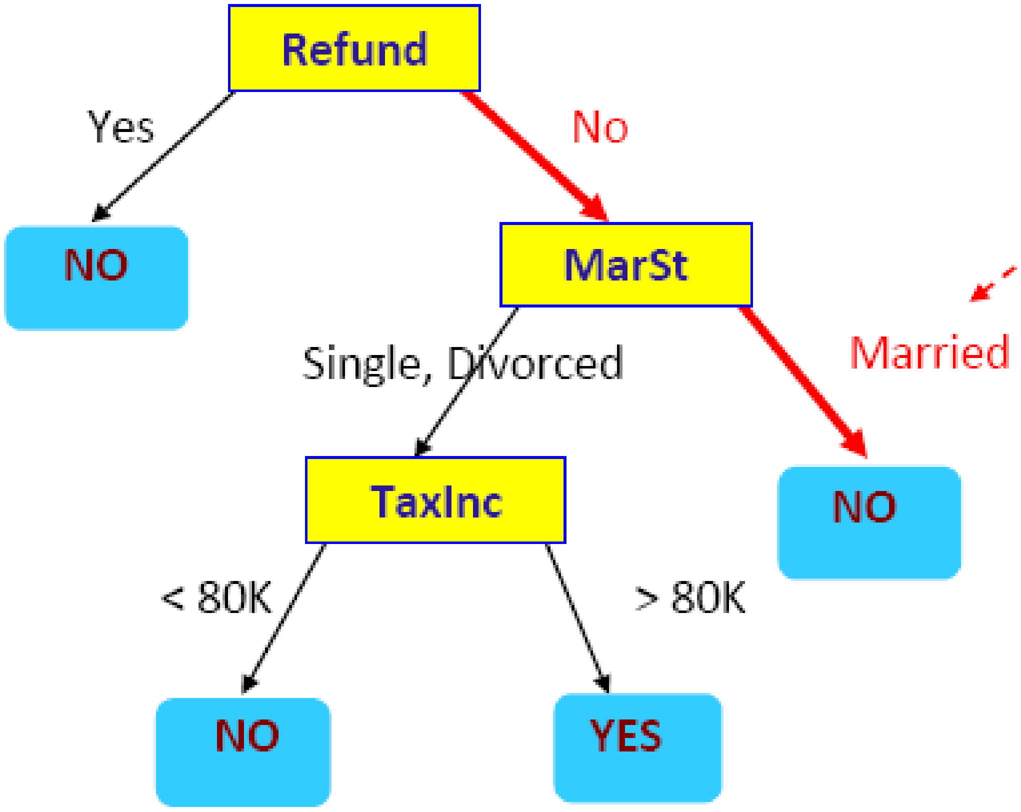
Refund	Marital Status	Taxable Income	Cheat
No	Married	80K	?



决策树——分类过程

Test Data

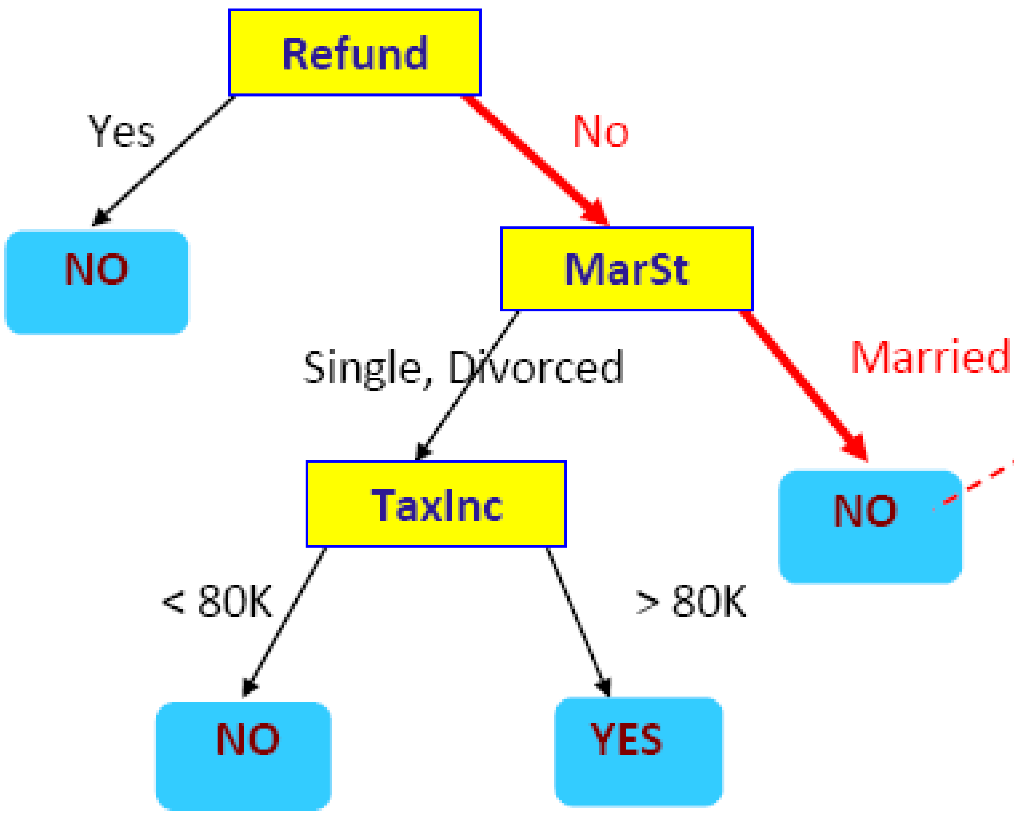
Refund	Marital Status	Taxable Income	Cheat
No	Married	80K	?



决策树——分类过程

Test Data

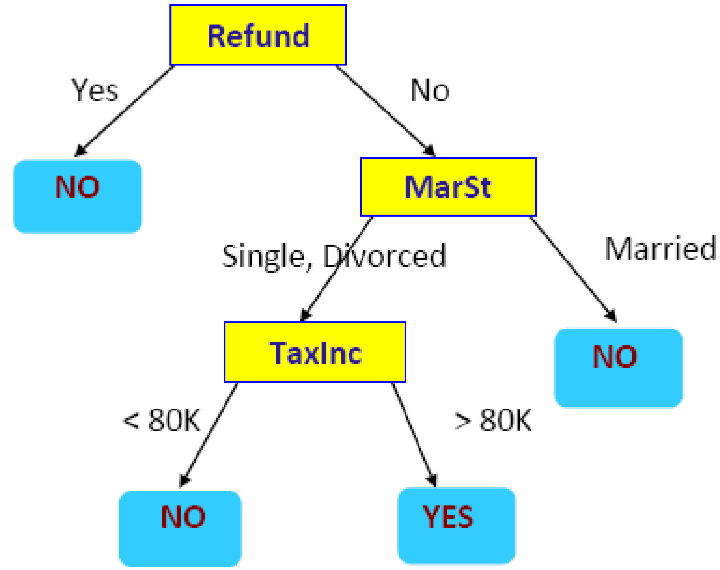
Refund	Marital Status	Taxable Income	Cheat
No	Married	80K	?



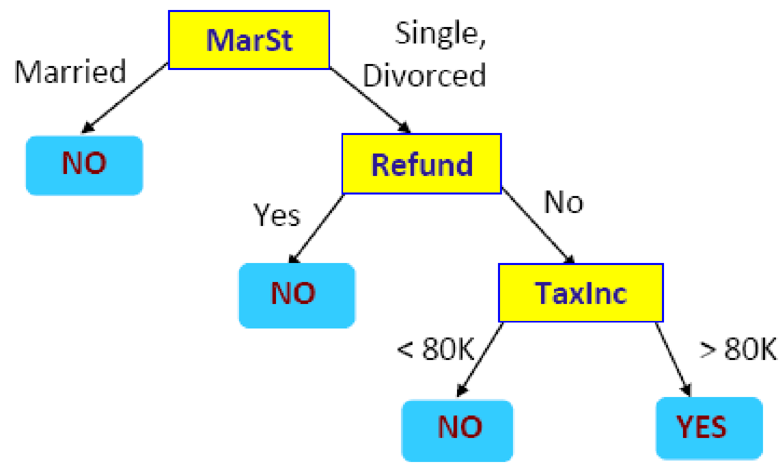
Assign Cheat to “No”

决策树——属性选择的次序问题

Tid	Refund	Marital Status	Taxable Income	Cheat
	categorical	categorical	continuous	class
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



哪棵树更好？ or 哪种次序更好？



决策树

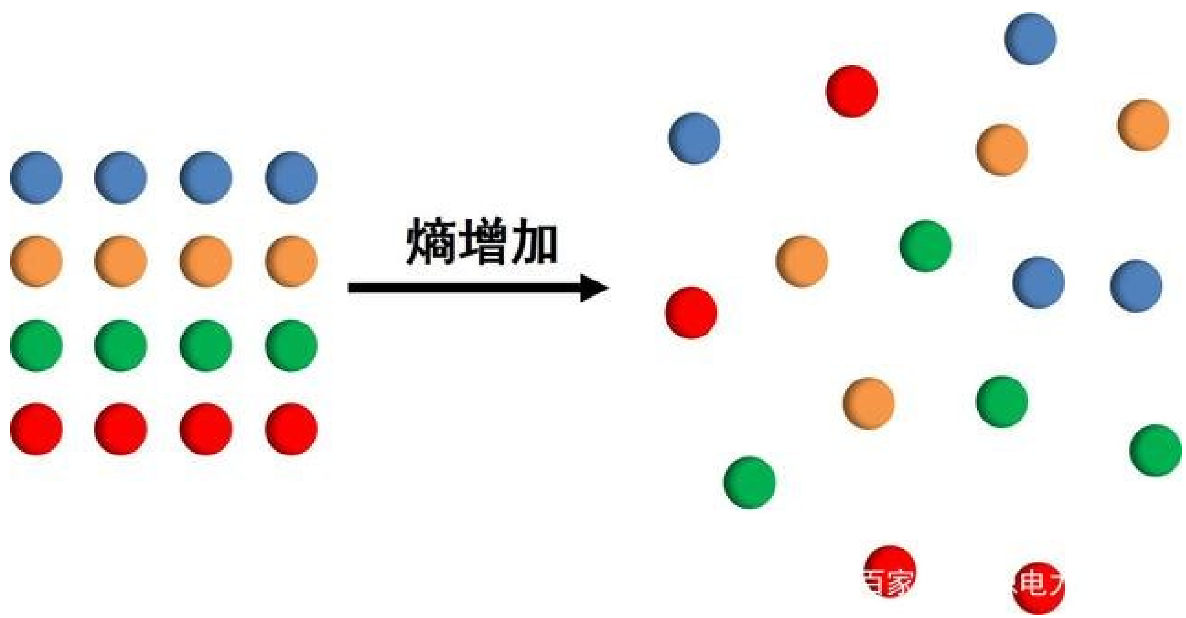
- 决策树算法：
  - 决策树算法是一种归纳分类算法，它通过对训练集的学习，挖掘出有用的规则，用于对新数据进行预测。
  - 决策树算法属于监督学习方法。
  - 决策树归纳的基本算法是贪心算法，自顶向下来构建决策树。
  - 贪心算法：在每一步选择中都采取在当前状态下最好优的选择。
  - 在决策树的生成过程中，分割方法即属性选择的度量是关键。

决策树

- 决策树算法关注的主要问题：
  - 特征选择（选择哪个属性作为分类依据）
    - 信息增益
    - 信息增益比
    - 基尼指数/平方误差
  - 决策树的生成算法
    - ID3算法
    - C4.5算法
    - CART算法
  - 决策树的剪枝策略：决策树的贪心特性容易生成过多的分枝造成过拟合，需要对其进行控制，以提高对未知数据分类的准确性。
    - 预剪枝方法
    - 后剪枝方法

信息熵

- 熵(entropy)是物理学中的一个重要的概念，它最初作为一个热力学函数被提出，用于描述系统的热力学状态。熵的概念可以从不同的角度进行解释和定义，但最基本的是它表示系统无序或混乱的程度，熵越大表示系统的混乱程度越高。



信息熵

- Shannon 1948年提出的信息论理论：
- 信息熵(entropy)：信息量大小的度量，即表示随机变量不确定性的度量。


同样两组苹果，哪一组包含的信息量大（信息熵高）？

信息熵

- 设X是一个取有限个值的离散随机变量，其概率分布为：

$$P(X=x_i)=p_i\ ,\quad i=1,2,\cdots,n$$

- 则随机变量X整体的信息熵定义为：

$$\text{信息熵}\ H=-\sum_{i=1}^{n}p(x_i)\log_2 p(x_i)$$

- 设有随机变量(X,Y),其联合概率分布为：

$$P(X=x_i,Y=y_j)=p_{ij}\ ,\quad i=1,2,\cdots,n\ ;\quad j=1,2,\cdots,m$$

- 则在已知随机变量X的条件下随机变量Y的不确定性的度量，即条件熵H(Y|X)定义为：

$$H(Y|X)=\sum_{i=1}^{n}p_iH(Y|X=x_i)$$

信息增益

- 信息增益(Information gain):特征A对训练数据集D的信息增益，$g(D,A)$，定义为集合D的熵$H(D)$与特征A给定条件下D的条件熵$H(D|A)$之差，即：

$$g(D,A) = H(D) - H(D|A)$$

- 信息增益表示得知特征A的信息而使得类Y的信息的不确定性减少的程度
- 数据集D原来的不确定程度为$H(D)$，根据特征A进行划分后，数据集D的不确定程度变成了$H(D|A)$，二者之差就表示特征A对D的信息增益（也即衡量特征A能够多大程度上区分数据集D中的类别）
- 根据信息增益，就可以进行特征选择

信息熵

- 设训练数据集为D
- |D|表示其样本容量，即样本个数
- 设有K个类Ck, k = 1,2，…K，
- |Ck |为属于类Ck的样本个数
- 数据集D的信息熵H(D):

$$H(D) = -\sum_{k=1}^{K} \frac{|C_k|}{|D|} log_2 \frac{|C_k|}{|D|}$$

- 右边数据中

数量	是	否
15	9	6

	年龄	有工作	有房子	信用	类别
0	青年	否	否	一般	否
1	青年	否	否	好	否
2	青年	是	否	好	是
3	青年	是	是	一般	是
4	青年	否	否	一般	否
5	中年	否	否	一般	否
6	中年	否	否	好	否
7	中年	是	是	好	是
8	中年	否	是	非常好	是
9	中年	否	是	非常好	是
10	老年	否	是	非常好	是
11	老年	否	是	好	是
12	老年	是	否	好	是
13	老年	是	否	非常好	是
14	老年	否	否	一般	否

$$H(D) = -\sum_{k=1}^{K} \frac{|C_k|}{|D|} log_2 \frac{|C_k|}{|D|} = -\frac{9}{15} log_2 \frac{9}{15} - \frac{6}{15} log_2 \frac{6}{15}$$

$$= 0.971$$

条件信息熵

- 特征A有n个不同的 取值{a1,a2…an}根据特征A的取值将D划分为n个子集D1,….,Dn
- |Di|为 Di的样本个数
- 记子集Di中属于类Ck的样本集合为Dik
- |Dik|为Dik的样本个数
- 则特征A对数据集D的条件熵H(D|A):

$$H(D|A)=\sum_{i=1}^{n}\frac{|D_i|}{|D|}H(D_i)$$

$$=-\sum_{i=1}^{n}\frac{|D_i|}{|D|}\sum_{k=1}^{K}\frac{|D_{ik}|}{|D_i|}\log_2\frac{|D_{ik}|}{|D_i|}$$

	年龄	有工作	有房子	信用	类别
0	青年	否	否	一般	否
1	青年	否	否	好	否
2	青年	是	否	好	是
3	青年	是	是	一般	是
4	青年	否	否	一般	否
5	中年	否	否	一般	否
6	中年	否	否	好	否
7	中年	是	是	好	是
8	中年	否	是	非常好	是
9	中年	否	是	非常好	是
10	老年	否	是	非常好	是
11	老年	否	是	好	是
12	老年	是	否	好	是
13	老年	是	否	非常好	是
14	老年	否	否	一般	否

条件信息熵

	年龄	有工作	有房子	信用	类别
0	青年	否	否	一般	否
1	青年	否	否	好	否
2	青年	是	否	好	是
3	青年	是	是	一般	是
4	青年	否	否	一般	否
5	中年	否	否	一般	否
6	中年	否	否	好	否
7	中年	是	是	好	是
8	中年	否	是	非常好	是
9	中年	否	是	非常好	是
10	老年	否	是	非常好	是
11	老年	否	是	好	是
12	老年	是	否	好	是
13	老年	是	否	非常好	是
14	老年	否	否	一般	否

- 按年龄划分：

年龄	数量	是	否
青年	5	2	3
中年	5	3	2
老年	5	4	1

- 计算分组的信息熵 $H(D_i)$：

$$H(D|A_1=\text{青年}) = -\frac{2}{5}log_2\frac{2}{5} - \frac{3}{5}log_2\frac{3}{5} = 0.971$$

$$H(D|A_1=\text{中年}) = -\frac{3}{5}log_2\frac{3}{5} - \frac{2}{5}log_2\frac{2}{5} = 0.971$$

$$H(D|A_1=\text{老年}) = -\frac{4}{5}log_2\frac{4}{5} - \frac{1}{5}log_2\frac{1}{5} = 0.7219$$

- 计算年龄对数据集D的条件熵熵H(D|年龄)

$$H(D|\text{年龄}) = \sum_{i=1}^{n}\frac{|D_i|}{|D|}H(D_i)$$

$$= \frac{5}{15}\times 0.971 + \frac{5}{15}\times 0.971 + \frac{5}{15}\times 0.7219$$
$$= 0.8880$$

信息增益

- 特征A对数据集D的信息增益g(D,A)：

$$g(D,A) = H(D) - H(D|A)$$

- 计算年龄对数据集D的信息增益：

g(D,年龄)=H(D)-H(D|年龄)

=0.971-0.888

=0.083

- 按此方法依次计算工作、房子、信用对D的信息增益，可以比较选择出信息增益最大的特征。

	年龄	有工作	有房子	信用	类别
0	青年	否	否	一般	否
1	青年	否	否	好	否
2	青年	是	否	好	是
3	青年	是	是	一般	是
4	青年	否	否	一般	否
5	中年	否	否	一般	否
6	中年	否	否	好	否
7	中年	是	是	好	是
8	中年	否	是	非常好	是
9	中年	否	是	非常好	是
10	老年	否	是	非常好	是
11	老年	否	是	好	是
12	老年	是	否	好	是
13	老年	是	否	非常好	是
14	老年	否	否	一般	否

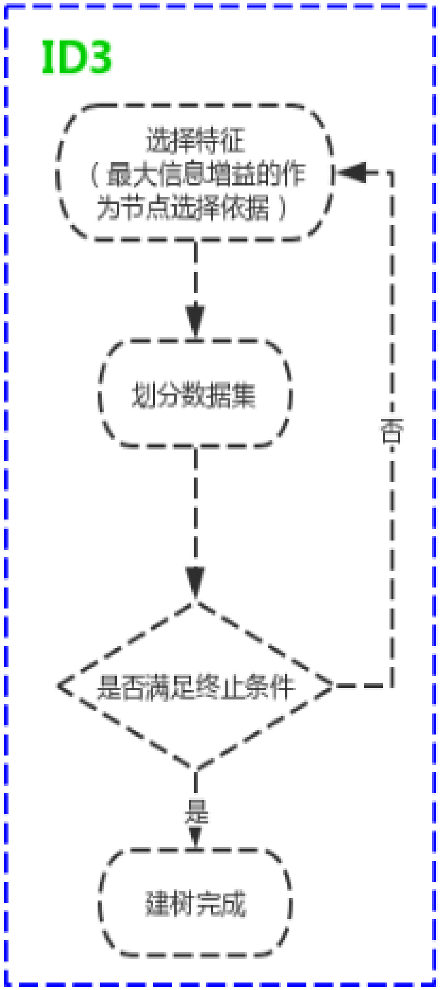
ID3算法（Quinlan 于1979年提出）

- 思想：在选择根结点和各个内部结点的分枝特征时，采用**信息增益**作为度量标准，特征的信息增益值越大，表示它的区分度就越高，使用该特征进行分类的效果就越好，因此每次都会选择具有**最高信息增益**的特征作为分枝特征。
- 算法步骤为：

1.计算数据集中所有特征的信息增益，选择信息增益最大的特征作为当前的决策节点对数据集进行划分

2.根据该特征的不同取值形成该结点的不同分枝，得到划分后的多个子集（每个特征只用一次，用完抛弃）

3.重复1,2步，对各分枝中的样本子集进行递归划分，直到子集中所有样本同属一类，或者没有剩余的特征用于进一步划分为止。



ID3算法——示例（buy_computer）

age	income	student	credit_rating	buy_computer
youth	high	no	fair	no
youth	high	no	excellent	no
middle_aged	high	no	fair	yes
senior	medium	no	fair	yes
senior	low	yes	fair	yes
senior	low	yes	excellent	no
middle_aged	low	yes	excellent	yes
youth	medium	no	fair	no
youth	low	yes	fair	yes
senior	medium	yes	fair	yes
youth	medium	yes	excellent	yes
middle_aged	medium	no	excellent	yes
middle_aged	high	yes	fair	yes
senior	medium	no	excellent	no

ID3算法——示例（buy_computer）

- 首先，计算数据集分类所需的信息熵：
  - 在数据集X中，给定的样本数量为14，类标号为*Yes* (表示购买电脑)的样本数量为n_1=9，类标号为*No* (表示不购买电脑)的样本数量为n_2=5，因此数据集中两个类别的先验概率分别为：

$$p(Yes)=n_1/total=9/14$$

$$p(No)=n_2/total=5/14$$

  - 对数据集分类所需的信息熵为：

age	income	student	credit_rating	buy_computer
youth	high	no	fair	no
youth	high	no	excellent	no
middle_aged	high	no	fair	yes
senior	medium	no	fair	yes
senior	low	yes	fair	yes
senior	low	yes	excellent	no
middle_aged	low	yes	excellent	yes
youth	medium	no	fair	no
youth	low	yes	fair	yes
senior	medium	yes	fair	yes
youth	medium	yes	excellent	yes
middle_aged	medium	no	excellent	yes
middle_aged	high	yes	fair	yes
senior	medium	no	excellent	no

$$H(X)= -p(Yes)*log(p(Yes))-p(No)*log(p(No)) = -9/14*log(9/14)-5/14*log(5/14)\approx 0.94$$

ID3算法——示例（buy_computer）

- 其次，计算各属性划分数据集时的信息增益：
  - 先计算特征*age*的条件熵。由于属性*age*有三个不同取值（*youth*，*middle_aged*，*senior*），因此可将数据集划分成三个子集：X_1, X_2和X_3。
  - 对于子集X_1(age=youth)，它的样本数量为n_1=5，其中类标号为*Yes*的数量n_{11}=2，类标号为*No*的数量n_{12}=3，则这两类样本在子集X_1中所占的比例分别为：

    $p_{11}=n_{11}/n_1=2/5=0.4$

    $p_{12}=n_{12}/n_1=3/5=0.6$
  - 这样，子集X_1的信息熵为：

age	income	student	credit_rating	buy_computer
youth	high	no	fair	no
youth	high	no	excellent	no
middle_aged	high	no	fair	yes
senior	medium	no	fair	yes
senior	low	yes	fair	yes
senior	low	yes	excellent	no
middle_aged	low	yes	excellent	yes
youth	medium	no	fair	no
youth	low	yes	fair	yes
senior	medium	yes	fair	yes
youth	medium	yes	excellent	yes
middle_aged	medium	no	excellent	yes
middle_aged	high	yes	fair	yes
senior	medium	no	excellent	no

$$H(X_1)= -p_{11}*log(p_{11})-p_{12}*log(p_{12}) =-0.4*log(0.4)-0.6*log(0.6)$$

ID3算法——示例（buy_computer）

- 其次，计算各属性划分数据集时的信息增益：
  - 先计算属性*age*的条件熵。由于属性*age*有三个不同取值（*youth*，*middle_aged*，*senior*），因此可将数据集划分成三个子集：X_1, X_2和X_3。
  - 对于子集X_2(*age=middle_aged*)，它的样本数量为n_2=4，其中类标号为*Yes*的数量n_{12}=4，类标号为*No*的数量n_{22}=0，则这两类样本在子集X_2中所占的比例分别为：

$$p_{21}=n_{12}/n_2=4/4=1$$

$$p_{22}=n_{22}/n_2=0/4=0$$

  - 这样，子集X_2的信息熵为：

$$H(X_2)= -p_{12}*log(p_{12})-p_{22}*log(p_{22}) =0$$

age	income	student	credit_rating	buy_computer
youth	high	no	fair	no
youth	high	no	excellent	no
middle_aged	high	no	fair	yes
senior	medium	no	fair	yes
senior	low	yes	fair	yes
senior	low	yes	excellent	no
middle_aged	low	yes	excellent	yes
youth	medium	no	fair	no
youth	low	yes	fair	yes
senior	medium	yes	fair	yes
youth	medium	yes	excellent	yes
middle_aged	medium	no	excellent	yes
middle_aged	high	yes	fair	yes
senior	medium	no	excellent	no

ID3算法——示例（buy_computer）

- 其次，计算各属性划分数据集时的信息增益：
  - 先计算属性*age*的条件熵。由于属性*age*有三个不同取值（*youth*，*middle_aged*，*senior*），因此可将数据集划分成三个子集：X_1, X_2和X_3。
  - 对于子集X_3(*age=senior*)，它的样本数量为n_3=5，其中类标号为*Yes*的数量n_{13}=3，类标号为*No*的数量n_{23}=2，则这两类样本在子集X_3中所占的比例分别为：

$$p_{13}=n_{13}/n_3=3/5=0.6$$

$$p_{23}=n_{23}/n_3=2/5=0.4$$

  - 这样，子集X_3的信息熵为：

$$H(X_3)= -p_{13}*log(p_{13})-p_{23}*log(p_{23}) =-0.6*log(0.6)-0.4*log(0.4)$$

age	income	student	credit_rating	buy_computer
youth	high	no	fair	no
youth	high	no	excellent	no
middle_aged	high	no	fair	yes
senior	medium	no	fair	yes
senior	low	yes	fair	yes
senior	low	yes	excellent	no
middle_aged	low	yes	excellent	yes
youth	medium	no	fair	no
youth	low	yes	fair	yes
senior	medium	yes	fair	yes
youth	medium	yes	excellent	yes
middle_aged	medium	no	excellent	yes
middle_aged	high	yes	fair	yes
senior	medium	no	excellent	no

ID3算法——示例（buy_computer）

- 其次，计算各特征划分数据集时的信息增益：
  - 先计算属性*age*的条件熵。由于属性*age*有三个不同取值（*youth*，*middle_aged*，*senior*），因此可将数据集划分成三个子集：X_1, X_2和X_3。
  - 由于子集X_1, X_2和X_3各自的信息熵分别为$\mathrm{H}(X_1)$，$\mathrm{H}(X_2)$和$\mathrm{H}(X_3)$，因此，属性*age*划分数据集的条件熵为：

$$H(X/age) = 5/14*H(X_1) + 4/14*H(X_2) + 5/14*H(X_3) \approx 0.694$$

  - 计算属性*age*的信息增益为：

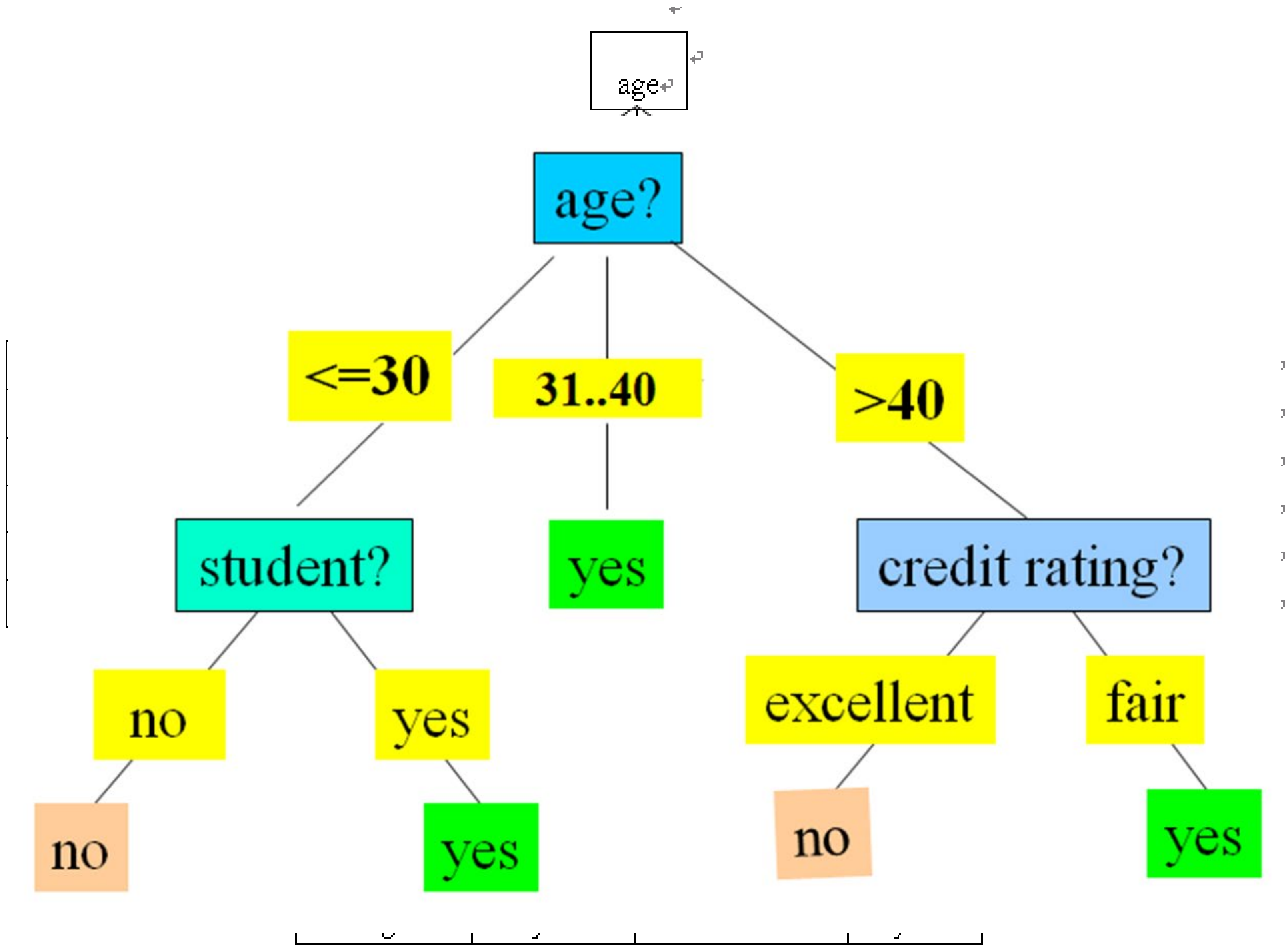
$$Gain(age) = H(X) - \mathrm{H}(X/age) = 0.94 - 0.694 = 0.246$$

ID3算法——示例（buy_computer）

- 最后，计算各特征划分数据集时的信息增益：
  - 按上述方式，可依次计算其他特征的信息增益分别为：
    - $Gain(income) = 0.029$
    - $Gain(student) = 0.151$
    - $Gain(credit\_rating) = 0.048$
- 在4个属性中，*age*的信息增益最大(0.246)，因此先以该属性来划分数据集。

age	income	student	credit_rating	buy_computer
youth	high	no	fair	no
youth	high	no	excellent	no
middle_aged	high	no	fair	yes
senior	medium	no	fair	yes
senior	low	yes	fair	yes
senior	low	yes	excellent	no
middle_aged	low	yes	excellent	yes
youth	medium	no	fair	no
youth	low	yes	fair	yes
senior	medium	yes	fair	yes
youth	medium	yes	excellent	yes
middle_aged	medium	no	excellent	yes
middle_aged	high	yes	fair	yes
senior	medium	no	excellent	no

ID3算法——示例（buy_computer）



ID3算法

- ID3算法的**优点**：
  - ID3算法通常只需要测试一部分属性就可完成对训练数据集的分类。
  - 从ID3算法构建的决策树中，很容易获得相应的决策规则。

ID3算法

- ID3算法的**缺点**：
  - ID3算法在选择节点的特征时，使用的信息增益更倾向于选择取值种类较多的特征进行划分，而不一定是**最优特征**进行划分。
  - ID3算法只能对特征值为**离散型**的数据集进行划分（构建决策树），不能处理特征值为连续型的数据集。
  - 贪心特性，会分到不能分为止，没有剪枝策略，**容易过拟合**

C4.5算法（Quinlan 于1993年提出）

- C4.5算法使用**信息增益比**来确定分枝特征，能够克服ID3算法使用信息增益时偏向于取值类型较多特征的不足，是对ID3算法的一种改进
  - 属性A的**信息增益比**的定义为：

$$g_R(D,A)=\frac{g(D,A)}{H_A(D)}$$

    - 其中数据集D关于特征A的值的熵：

$$H_A(D)=-\sum_{i=1}^{n}\frac{|D_i|}{|D|}log_2\frac{|D_i|}{|D|}$$

n是特征A取值的个数，D_i为特征A的第i个取值对应的样本集合，当n的值较大时，就会降低信息增益比。

信息增益比

- 计算年龄对数据集D的信息增益：

  $g(D,年龄)=H(D)-H(D|年龄)$

  $=0.971-0.888$

  $=0.083$

- 计算年龄的特征值的熵：

$H_{年龄}(D)= -5/15\log(5/15) - 5/15\log(5/15) -5/15\log(5/15)$

- 计算年龄对数据集D的信息增益比：

  $g_r(D,年龄)= g(D,年龄)/ H_{年龄}(D)= 0.075$

	年龄	有工作	有房子	信用	类别
0	青年	否	否	一般	否
1	青年	否	否	好	否
2	青年	是	否	好	是
3	青年	是	是	一般	是
4	青年	否	否	一般	否
5	中年	否	否	一般	否
6	中年	否	否	好	否
7	中年	是	是	好	是
8	中年	否	是	非常好	是
9	中年	否	是	非常好	是
10	老年	否	是	非常好	是
11	老年	否	是	好	是
12	老年	是	否	好	是
13	老年	是	否	非常好	是
14	老年	否	否	一般	否

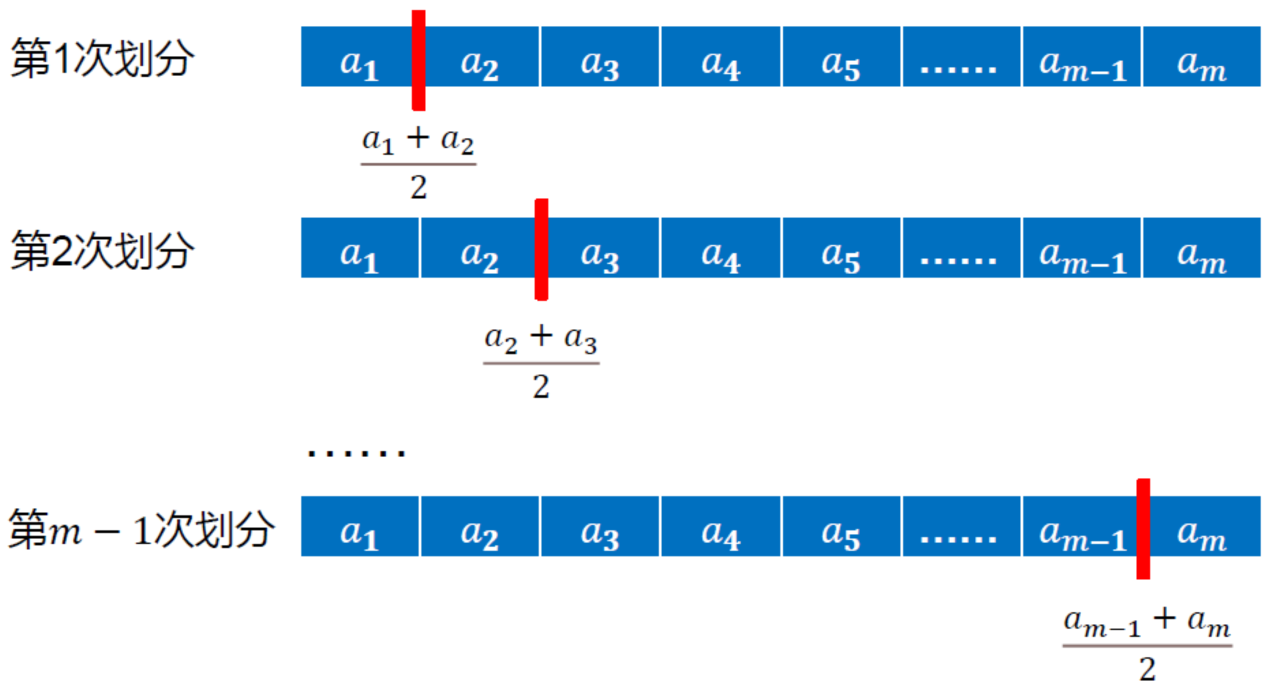
C4.5算法

- C4.5算法既可以处理离散型描述属性，也可以处理连续型描述属性。
  - 当处理离散型属性时，C4.5算法与ID3算法相同；
  - 当处理连续型属性时，C4.5算法需要先将连续型属性转换成离散型属性。
  - 对于连续值属性A，假设在某个结点上的样本数量为m，则C4.5算法将进行如下操作：
    - (1) 将该结点上的所有样本按照属性的取值由小到大排序，得到排序结果$\{a_1, a_2, ..., a_m\}$；

C4.5算法

- C4.5算法既可以处理离散型描述属性，也可以处理连续型描述属性。
  - (2) 在$\{a_1, a_2, ..., a_m\}$中生成m-1个分割点

    其中：第 i 个 ($1 \le i \le m$-1) 分割点的取值设置为 $v_i=(a_i+a_{(i+1)})/2$。



C4.5算法

- C4.5算法既可以处理离散型描述属性，也可以处理连续型描述属性。
  - (3) 分别计算以该点作为二元分类点时的信息增益。从m-1个分割点中选择选择**信息增益比**最大的点作为该连续特征的最佳二元离散分类点。
  - 比如取到的基尼指数最小的点为a_t，则小于a_t的值为类别1，大于a_t的值为类别2，这样就做到了连续特征的离散化
  - 这种离散化算法较为笨重，计算消耗大（需要反复排序），不适合大数据集
  - 要注意的是，与离散属性不同的是，如果当前节点为连续属性，则该属性后面还可以参与子节点的产生选择过程

C4.5算法——离散化示例

- 将"*buy_computer*"中的属性*age*的取值由{*youth*, *middle_aged*, *senior*}改为具体年龄{32, 25, 46, 56, 60, 52, 42, 36, 23, 51, 38, 43, 41, 65}，C4.5算法离散化的具体过程。
  - (1) 对年龄序列由小到大排序，新的序列为{23, 25, 32, 36, 38, 41, 42, 43, 46, 51, 52, 56, 60, 65}；
  - (2) 对新的年龄序列生成分割点：由于样本数量为14，因此可生成13个分割点。
    - 例如：第一个分割点为(23+25)/2=24，它可将数据集划分为年龄在区间[23, 24]的样本和在区间(24, 65]的样本。

age	income	student	credit_rating	buy_computer
youth	high	no	fair	no
youth	high	no	excellent	no
middle_aged	high	no	fair	yes
senior	medium	no	fair	yes
senior	low	yes	fair	yes
senior	low	yes	excellent	no
middle_aged	low	yes	excellent	yes
youth	medium	no	fair	no
youth	low	yes	fair	yes
senior	medium	yes	fair	yes
youth	medium	yes	excellent	yes
middle_aged	medium	no	excellent	yes
middle_aged	high	yes	fair	yes
senior	medium	no	excellent	no

C4.5算法——离散化示例

- (3) 选择最佳分割点。
  - 例如：对于第一个分割点，可以计算得到年龄在区间[23, 24]和(24, 65]的样本数量以及每个区间的样本属于各个类别的数量，从而计算第一个分割点的**信息增益比**。
  - 依此方式，计算其他分割点的**信息增益比**，并从中选出具有**最大信息增益比**的分割点。
- (4) 根据最佳分割点，离散化属性的连续值。
  - 例如：当最佳分割点为37时，数据集中的样本可以根据*age*取值分成两类，一类是≤37，另一类是＞37。
- 说明：在有些情况下，可能需要出现多个最佳分割点，可以任选一个

C4.5算法

- C4.5虽然改进或者改善了ID3算法的几个主要的问题，仍然有优化的空间。
  - 由于决策树算法非常容易过拟合，因此对于生成的决策树必须要进行剪枝。C4.5提供了剪枝方法，但存在剪枝过度或剪枝失败的情况。
  - C4.5只能用于分类
  - 连续离散化方法较为笨重，计算消耗大（需要大量对数运算和反复排序）
  - C4.5和ID3一样生成的是多叉树，即一个父节点可以有多个子节点。很多时候，在计算机中二叉树模型会比多叉树运算效率高。如果采用二叉树，可以提高效率。

CART算法(Breiman等人，1984)

- 无论是ID3还是C4.5,都是基于信息熵来选择特征，会涉及大量的对数运算，比较耗时。CART分类树是一种二叉树，既可以用于创建分类树，也可以用于创建回归树，并使用**基尼系数（分类）**或**平方误差（回归）**来选择特征，基尼系数（均方差）越小，则特征的建模能力越好，这种方式和信息增益(比)是相反的。
- 基尼系数（Gini）：在分类问题中，假设有K个类，样本点属于第k类的概率为p_k，则基尼系数的表达式为

$$Gini(p) = \sum_{k=1}^{K} p_k(1 - p_k) = 1 - \sum_{k=1}^{K} p_k^2$$

- 对于一个给定的数据集D，C_k是D中属于第k类的样本子集，则D的基尼系数：

$$Gini(D) = 1 - \sum_{k=1}^{K} \left(\frac{|C_k|}{|D|}\right)^2$$

CART算法(Breiman等人，1984)

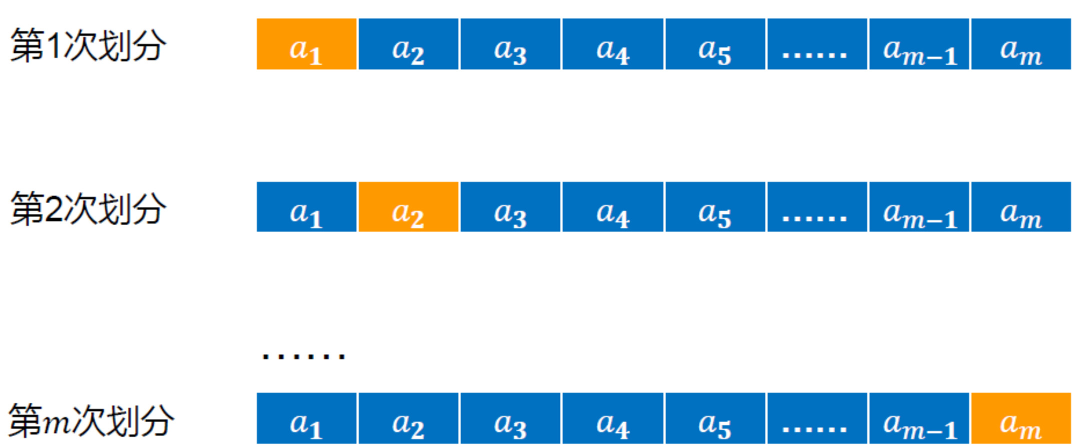
- 如果样本集合D根据特征A是否为a被分割成D_1和D_2，那么基尼指数Gini(D,A)表示经过A=a划分后集合D的不确定性（与ID3算法中的条件熵同理）：

$$Gini(D,A)=\frac{|D_1|}{|D|}Gini(D_1)+\frac{|D_2|}{|D|}Gini(D_2)$$

- 由于CART树是二叉树，因此需要对特征的取值进行二分类划分。对于连续的特征值，划分思路同前边介绍的C4.5对连续值的划分方法（P46），不同之处在于，基于Gini系数最小来寻找最佳划分点；对于离散特征值，则处理方法同理，每次取某个特征值为一类，剩下的所有特征值算作另一类，然后计算这种划分方式的Gini系数，把所有特征值划分的Gini系数算一遍，Gini系数最小的便是最佳划分。

CART算法(Breiman等人，1984)

- 假设特征A有m个离散值。分类标准是：每一次将其中一个特征分为一类，其他非该特征分为另一类。依照这个标准遍历所有分类情况，计算每个分类下的基尼系数，最后选择最小的作为最终的特征划分。
- 比如第1次取$\{a_1\}$为类别1，那么剩下的特征$\{a_2, a_3,\ldots\ldots, a_{m-1}, a_m\}$为类别2，由此遍历，第$m$次取$\{a_m\}$为类别1，那么剩下的特征$\{a_1,a_2, a_3,\ldots\ldots,a_{m-1}\}$为类别2。



CART算法(Breiman等人，1984)

- 如果样本集合*D*根据特征A是否为*a*被分割成D_1和D_2，那么基尼指数Gini(*D*,*A*)表示经过A=*a*划分后集合*D*的不确定性（与ID3算法中的条件熵同理）：

$$Gini(D,A)=\frac{|D_1|}{|D|}Gini(D_1)+\frac{|D_2|}{|D|}Gini(D_2)$$

假设要求Gini(*D*,年龄=青年)的值，其中|*D*|表示整个数据集中样本个数，从编号知值为15，$|D_1|$表示年龄是青年的样本个数，值为5，$|D_2|$表示年龄不是青年的样本个数，值为10。Gini(D_1)表示青年这个类别的基尼指数，对应的类别有两个“是”，三个“否”，代入公式可得：

$$Gini(D1)=\frac{2}{5}*\left(1-\frac{2}{5}\right)+\frac{3}{5}*\left(1-\frac{3}{5}\right)=2*\frac{2}{5}*\left(1-\frac{2}{5}\right)$$

同理可得Gini(D_2)，故

$$\text{Gini}(D,\text{年龄=青年})=\frac{5}{15}*\left[2*\frac{2}{5}*\left(1-\frac{2}{5}\right)\right]+\frac{10}{15}*\left[2*\frac{7}{10}*\left(1-\frac{7}{10}\right)\right]=0.44$$

	年龄	有工作	有房子	信用	类别
0	青年	否	否	一般	否
1	青年	否	否	好	否
2	青年	是	否	好	是
3	青年	是	是	一般	是
4	青年	否	否	一般	否
5	中年	否	否	一般	否
6	中年	否	否	好	否
7	中年	是	是	好	是
8	中年	否	是	非常好	是
9	中年	否	是	非常好	是
10	老年	否	是	非常好	是
11	老年	否	是	好	是
12	老年	是	否	好	是
13	老年	是	否	非常好	是
14	老年	否	否	一般	否

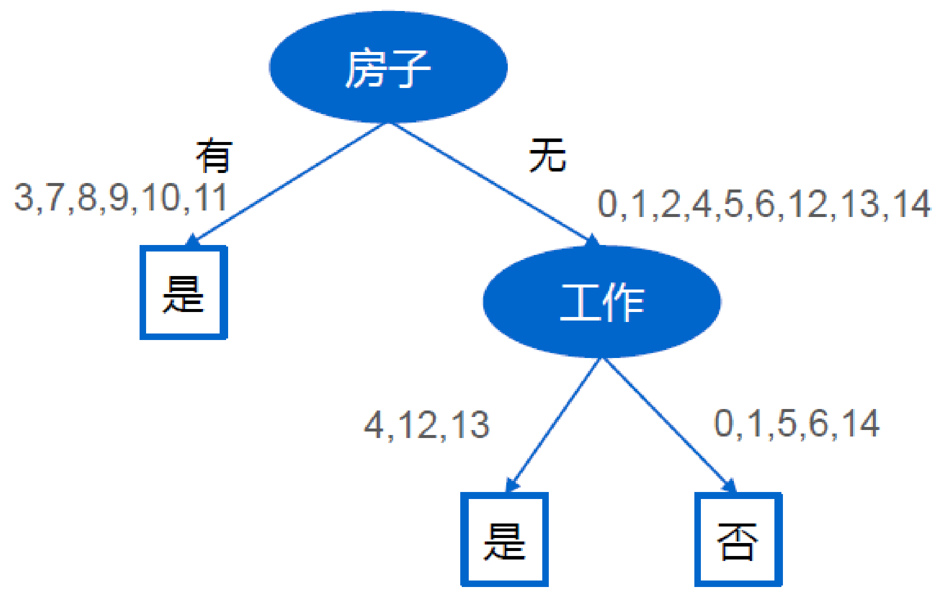
CART算法(Breiman等人，1984)

- 同理可得:

$$\mathrm{Gini}(D,\text{年龄}=\text{中年})=\frac{5}{15}*\left[2*\frac{3}{5}*\left(1-\frac{3}{5}\right)\right]+\frac{10}{15}*\left[2*\frac{6}{10}*\left(1-\frac{6}{10}\right)\right]=0.48$$

$$\mathrm{Gini}(D,\text{年龄}=\text{老年})=\frac{5}{15}*\left[2*\frac{4}{5}*\left(1-\frac{4}{5}\right)\right]+\frac{10}{15}*\left[2*\frac{5}{10}*\left(1-\frac{5}{10}\right)\right]=0.44$$

Gini(D,有工作=是)=0.32
Gini(D,有房子=是)=0.27
Gini(D,信用=非常好)=0.36
Gini(D,信用=好)=0.47
Gini(D,信用=一般)=0.32



CART算法(Breiman等人，1984)

- **平方误差：** 假设y_i表示训练集$D=\{(x_1,y_1),(x_2,y_2),\ldots,(x_n,y_n)\}$的输出变量，是连续变量。$f(x_i)$是预测值，则预测误差可以表示为

$$\sum_{x_i \in D} [y_i - f(x_i)]^2$$

- 对于回归问题，输出的预测值不是类别而是连续变量，假设CART树将输入的样本空间划分成M份（对应M个叶子结点R_m），则CART树通过**平方误差最小化来选择特征和特征的划分点，**并使用叶子节点R_m中所有样本的y_i的均值c_m^*作为最佳的预测值的输出。

$$c_m^* = avg(y_i | x_i \in R_m)$$

CART算法(Breiman等人，1984)

- 问题：如何对输入的样本空间进行划分？
- 启发式：选择第j个变量$x^{(j)}$和它取的某个值s，作为切分变量和切分点，定义两个区域：

$$R_1=\{x|x\leq s\}, R_2=\{x|x>s\}$$

- 然后通过求解平方误差最小化寻找最优切分变量和切分点：

$$\min_{j,s}\left\{\min_{c_1}\sum_{x_i\in R_1}(y_i-c_1)^2+\min_{c_2}\sum_{x_i\in R_2}(y_i-c_2)^2\right\}$$

其中，

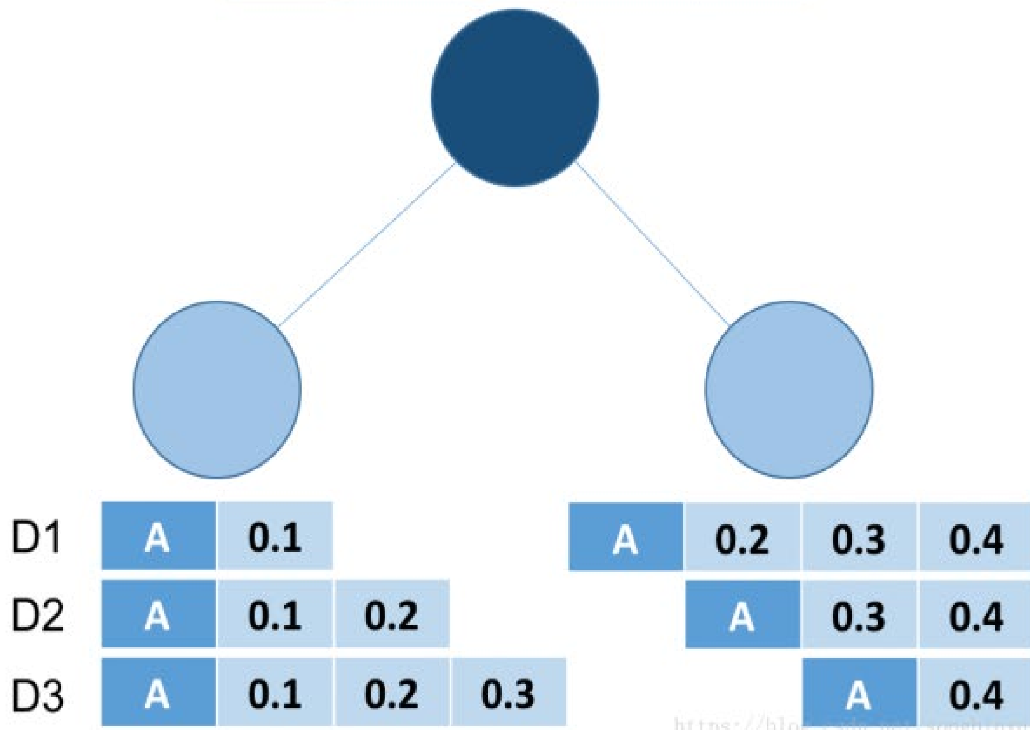
$$c_1=avg(y_i|x_i\in R_1)$$
$$c_2=avg(y_i|x_i\in R_2)$$

- 再对两个区域重复上述划分，直到满足停止条件。

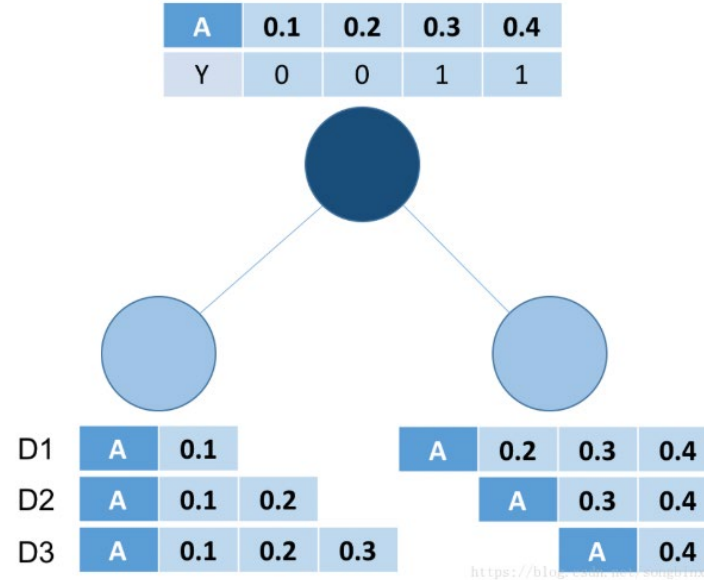
CART算法(Breiman等人，1984)

假设有四个样本，一维特征是 A，它的取值和对应的label如下。每两个取值之间，都能选为一个分界面（例如0.1和0.2之间选0.15为分界面），列举出三种划分情况，分别计算他们的平方误差。

A	0.1	0.2	0.3	0.4
Y	0	0	1	1



CART算法(Breiman



对于 D_1，$c_1=0$，$c_1=(0+1+1)/3=0.67$

$$loss_1=\sum_{i=1}^{1}(y_i-c_1)^2+\sum_{i=1}^{3}(y_i-c_2)^2$$
$$=(0-0)^2+(0-0.67)^2+(1-0.67)^2+(1-0.67)^2=0.67$$

同理对于 D_2、D_3计算得到：

$$loss_2=0$$
$$loss_3=0.67$$

显然D_2这个划分（$R_1=\{0.1,0.2\}$，$R_2=\{0.3,0.4\}$，最佳分裂点 0.25）是特征 A 的最佳分裂。

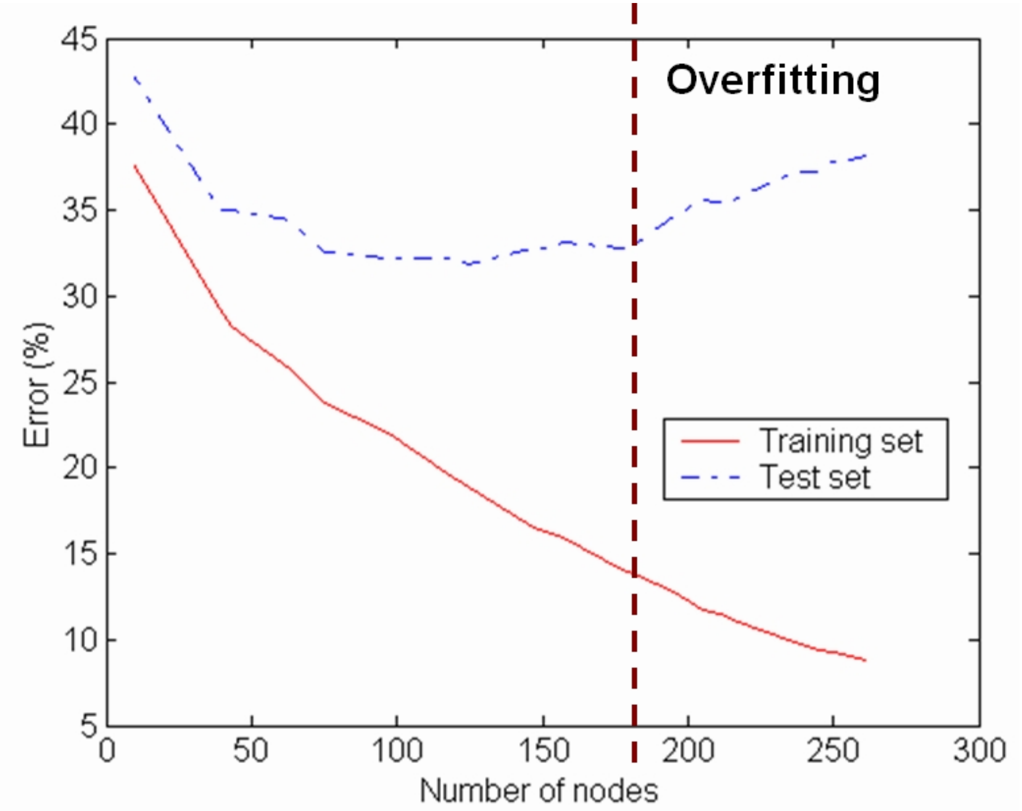
决策树中的剪枝策略

- 决策树会出现的问题
  - 由于数据中的噪声和孤立点，许多分枝反映的是训练数据中的异常。
  - 容易出现过拟合
  - 理想的决策树有三种：

(1)叶子结点数最少；

(2)叶子结点深度最小；

(3)叶子结点数最少且叶子结点深度最小。

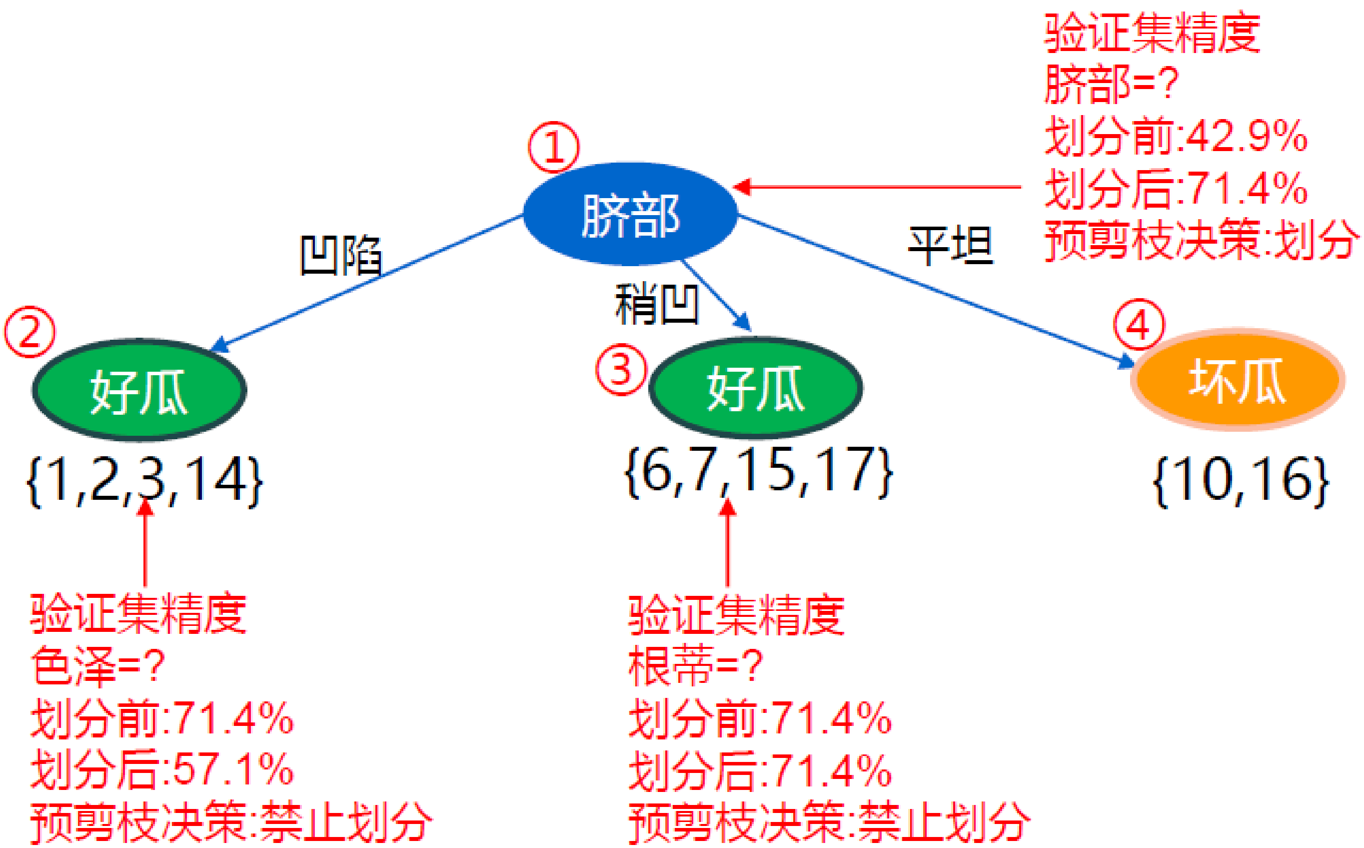


决策树中的剪枝策略

- 两种剪枝策略：
  - 预剪枝：提前停止树的构造，在节点划分前来确定是否继续增长，及早停止增长
  - 主要方法有：
    - •节点内数据样本低于某一阈值；
      - 但是，选择一个合适的阈值通常是很困难的。
    - •所有节点特征都已分裂；
    - •节点划分前准确率比划分后准确率高(一般是在验证集上)

决策树中的剪枝策略

预剪枝



决策树中的剪枝策略

- 两种剪枝策略：
  - 后剪枝：由“完全生长”的树剪去分枝——在已经生成的决策树上进行剪枝，从而得到简化版的剪枝决策树
  - 后剪枝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，后剪枝的欠拟合风险更小，泛化性能往往优于预剪枝决策树。常见剪枝策略如下：

    （1）错误率降低剪枝 REP(Reduced-Error Pruning)

    （2）悲观错误剪枝PEP(Pesimistic-Error Pruning)

    （3）代价复杂度剪枝 CCP(Cost-Complexity Pruning)

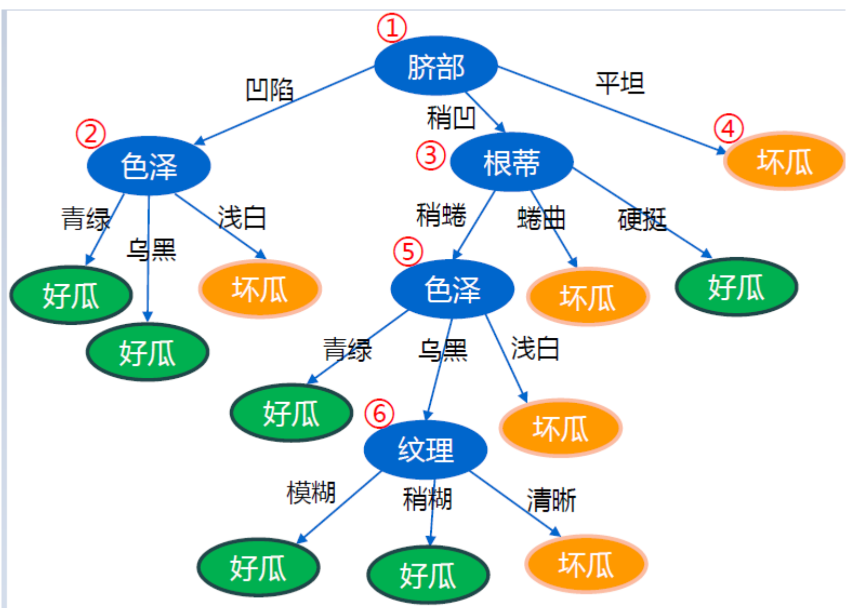
    （4）基于错误的剪枝EBP(Error-Based Pruning)

    （5）最小误差剪枝MEP(Minimum Error Pruning)

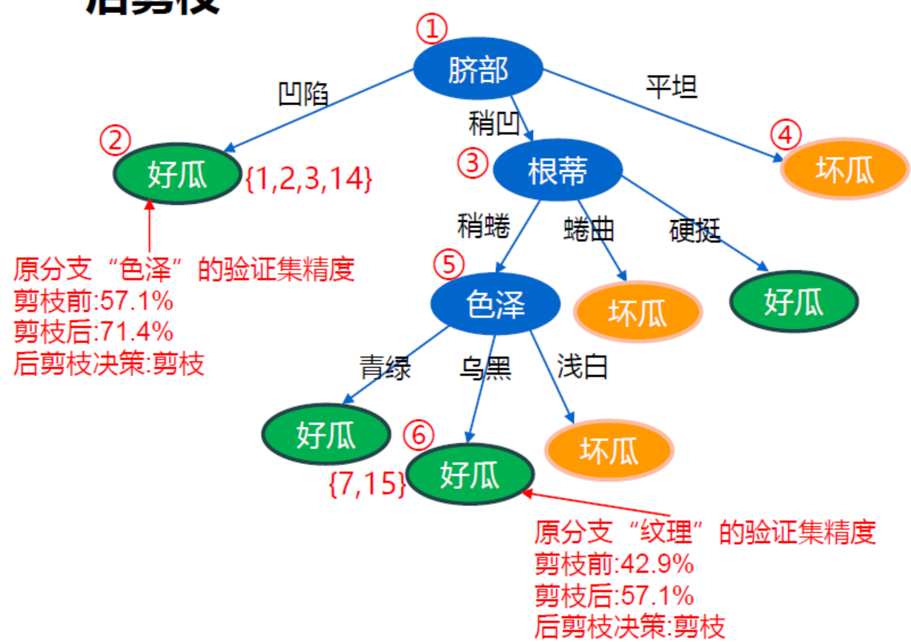
决策树中的剪枝策略

- 错误率降低剪枝 REP(Reduced-Error Pruning)：

REP方法是通过一个新的验证集来纠正树的过拟合问题。对于决策树中的每一个非叶子节点的子树，我们将它替换成一个叶子节点，该叶子节点的类别用大多数原则来确定，这样就产生了一个新的相对简化决策树，然后比较这两个决策树在验证集中的表现。如果新的决策树在验证集中的正确率较高，那么该子树就可以替换成叶子节点，从而达到决策树剪枝的目的。



后剪枝



决策树中的剪枝策略

- 悲观错误剪枝PEP(Pesimistic-Error Pruning)：

PEP方法是在C4.5算法中提出的，也是根据剪枝前后的错误率（或误判数）来决定是否剪枝，是对REP方法的改进：PEP不需要新的验证集，而是在原有的训练集上剪枝，并且PEP是自上而下剪枝的。由于我们还是用生成决策树时相同的训练样本，那么对于每个节点剪枝后的错分率一定是会上升的，因此在计算错分率时需要加一个惩罚因子0.5（二项概率分布逼近正态分布的连续性修正因子）。

对于一叶节点，它覆盖了N个样本，其中有E个错误，那么该叶子节点的错误率为（E+0.5）/N。这个0.5就是惩罚因子，那么一颗子树，它有L个叶子节点，那么该子树的错误率为：

$$p = \frac{\sum_{i=1}^{L} E_i + 0.5L}{\sum_{i=1}^{L} Ni}$$

那么，整棵子树的误判数E=$N*p$，误判数的方差std=$\sqrt{Np(1-p)}$随即引出剪枝的标准：

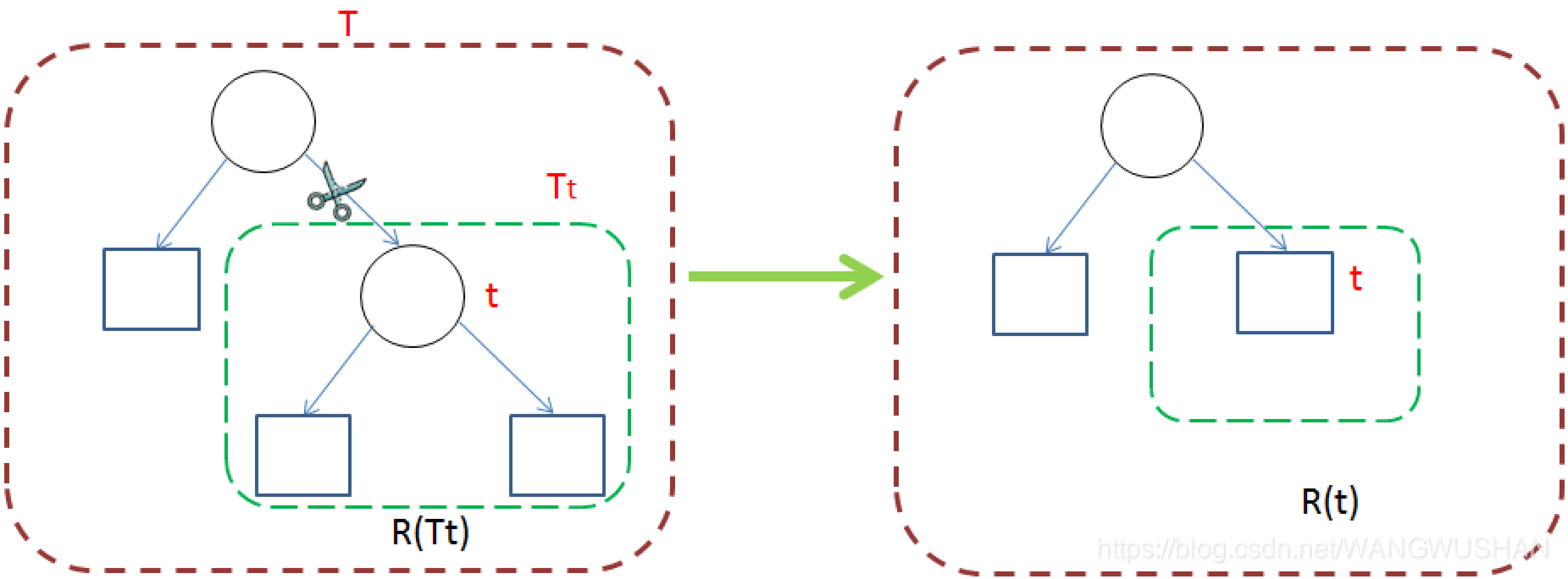
$$E(\text{剪枝后误判数}) - std(\text{剪枝前误判数}) < E(\text{剪枝前误判数})$$

即**剪枝后的误判数<剪枝前的误判数+标准差**（最悲观的误差）

决策树中的剪枝策略

- C4.5的后剪枝：

确认了剪枝判断条件，接下来自顶向下递归遍历每个非叶子节点来判断是否需要剪枝，剪枝后节点类别标签由节点中的多数类别表决决定。



决策树中的剪枝策略

- 代价复杂度剪枝 CCP(Cost-Complexity Pruning)：

CCP方法通过平衡决策树的预测误差（代价）和模型的复杂度来实现剪枝，核心思路是由完全树T_0开始，剪枝部分结点得到T_1，再次剪枝部分结点得到T_2,…直到直到剩下树根的树T_k；然后在验证数据集上对这k个树分别评价，选择损失函数最小的树。

- 代价指的是在剪枝过程中因子树Tt被叶节点替代而增加的错分样本;
- 复杂度表示剪枝后子树Tt减少的叶结点数;
- α则表示剪枝后树的复杂度降低程度与代价间的关系，定义为:

$$\alpha = \frac{R(t) - R(T_t)}{|N| - 1}$$

其中，R(t)表示节点t的错误代价，｜N｜表示子树Tt中的叶节点数， α越大表示剪掉该子树造成的代价越高（误差变大的越多）

CCP算法可以分为两个步骤，

Step 1：按照上述公式从下到上计算每一个非叶节点的α值，然后每一次都剪掉具有最小α值的子树。从而得到一个集合$\{T_0, T_1, T_2, \ldots, T_M\}$，其中$\{T_0\}$表示完整的决策树，$\{T_M\}$表示根节点。

Step 2：根据真实的错误率在集合 $\{T_0, T_1, T_2, \ldots, T_M\}$选出一个最好的决策树。

总结

- 三种决策树算法的比较

算法	连续值处理	缺失值处理	特征选择	树结构	支持类型	剪枝
ID3	不支持	不支持	信息增益	多叉树	分类	不支持
C4.5	支持	支持	信息增益比	多叉树	分类	支持
CART	支持	支持	基尼指数(分类树) 误差平方(回归树)	二叉树	分类、回归	支持

总结

- 决策树算法的优点：
  - 可解释性强，生成的决策树简单直观。
  - 基本不需要预处理，不需要提前归一化，处理缺失值。
  - 既可以处理离散值也可以处理连续值。
  - 可以处理多维度输出的分类问题。
  - 对于异常点的容错能力好，健壮性高。
- 决策树算法的缺点：
  - 容易过拟合，导致泛化能力不强。
  - 决策树会因为样本发生一点点的改动，就会导致树结构的剧烈改变。
  - 寻找最优的决策树是一个NP难的问题，我们一般是通过启发式方法，容易陷入局部最优。
  - 有些比较复杂的关系，决策树很难学习，比如异或。