决策树原理

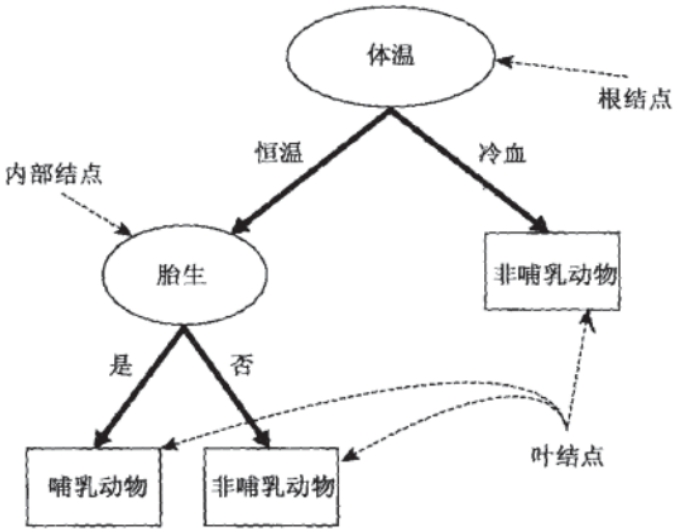
1.1 决策树是如何工作的

决策树（Decision Tree）是一种非参数的有监督学习方法，它能够从一系列有特征和标签的数据中总结出决策规则，并用树状图的结构来呈现这些规则，以解决分类和回归问题。决策树算法容易理解，适用各种数据，在解决各种问题时都有良好表现，尤其是以树模型为核心的各种集成算法，在各个行业和领域都有广泛的应用。

我们来简单了解一下决策树是如何工作的。决策树算法的本质是一种图结构，我们只需要问一系列问题就可以对数据进行分类了。比如说，来看看下面这组数据集，这是一系列已知物种以及所属类别的数据：

名字	体温	表皮覆盖	胎生	水生动物	飞行动物	有腿	冬眠	类标号
人类	恒温	毛发	是	否	否	是	否	哺乳类
鲑鱼	冷血	鳞片	否	是	否	否	否	鱼类
鲸	恒温	毛发	是	是	否	否	否	哺乳类
青蛙	冷血	无	否	半	否	是	是	两栖类
巨蜥	冷血	鳞片	否	否	否	是	否	爬行类
蝙蝠	恒温	毛发	是	否	是	是	是	哺乳类
鸽子	恒温	羽毛	否	否	是	是	否	鸟类
猫	恒温	软毛	是	否	否	是	否	哺乳类
豹纹鲨	冷血	鳞片	是	是	否	否	否	鱼类
海龟	冷血	鳞片	否	半	否	是	否	爬行类
企鹅	恒温	羽毛	否	半	否	是	否	鸟类
豪猪	恒温	刚毛	是	否	否	是	是	哺乳类
鳗	冷血	鳞片	否	是	否	否	否	鱼类
蟒蛇	冷血	无	否	半	否	是	是	两栖类

我们现在的目标是，将动物们分为哺乳类和非哺乳类。那根据已经收集到的数据，决策树算法为我们算出了下面的这棵决策树：



假如我们现在发现了一种新物种Python，它是冷血动物，体表带鳞片，并且不是胎生，我们就可以通过这棵决策树来判断它的所属类别。

可以看出，在这个决策过程中，我们一直在对记录的特征进行提问。最初的问题所在的地方叫做**根节点**，在得到结论前的每一个问题都是**中间节点**，而得到的每一个结论（动物的类别）都叫做**叶子节点**。

关键概念：节点
根节点：没有进边，有出边。包含最初的，针对特征的提问。 中间节点：既有进边也有出边，进边只有一条，出边可以有很多条。都是针对特征的提问。 叶子节点：有进边，没有出边，**每个叶子节点都是一个类别标签。** *子节点和父节点：在两个相连的节点中，更接近根节点的是父节点，另一个是子节点。

决策树算法的核心是要解决两个问题：

1）如何从数据表中找出最佳节点和最佳分枝？

2）如何让决策树停止生长，防止过拟合？

几乎所有决策树有关的模型调整方法，都围绕这两个问题展开。接下来，我们就来了解一下决策树背后的原理。

1.2 构建决策树

接下来讨论如何根据已有的数据集来建立有效的决策树。原则上讲，任意一个数据集上的所有特征都可以被拿来分枝，特征上的任意节点又可以自由组合，所以一个数据集上可以发展出非常非常多棵决策树，其数量可达指数级。在这些树中，总有那么一棵树比其他的树分类效力都好，那样的树叫做"全局最优树"。

关键概念：全局最优，局部最优
全局最优：经过组合形成的，整体来说分类效果最好的模型 局部最优：每一次分枝的时候都向着更好的分类效果分枝，但无法确认如此生成的树在全局上是否是最优的

要在这么多棵决策树中去一次性找到分类效果最佳的那一棵是不可能的，如果通过排列组合来进行筛选，计算量过于大而且低效，因此我们不会这样做。相对的，机器学习研究者们开发了一些有效的算法，能够在合理的时间内构造出具有一定准确率的**次最优**决策树。这些算法基本都执行"**贪心策略**"，即通过局部的最优来达到我们相信是最接近全局最优的结果。

> **关键概念：贪心算法**
>
> 通过实现局部最优来达到接近全局最优结果的算法，所有的树模型都是这样的算法。

最典型的决策树算法是Hunt算法，该算法是由Hunt等人提出的最早的决策树算法。现代，Hunt算法是许多决策树算法的基础，包括ID3、C4.5和CART等。Hunt算法诞生时间较早，且基础理论并非特别完善，此处以应用较广、理论基础较为完善的ID3算法的基本原理开始，讨论如何利用局部最优化方法来创建决策模型。

1.2.1 ID3算法构建决策树

ID3算法原型见于J.R Quinlan的博士论文，是基础理论较为完善，使用较为广泛的决策树模型，在此基础上J.R Quinlan进行优化后，陆续推出了C4.5和C5.0决策树算法，后二者现已称为当前最流行的决策树算法，我们先从ID3开始讲起，再讨论如何从ID3逐渐优化至C4.5。

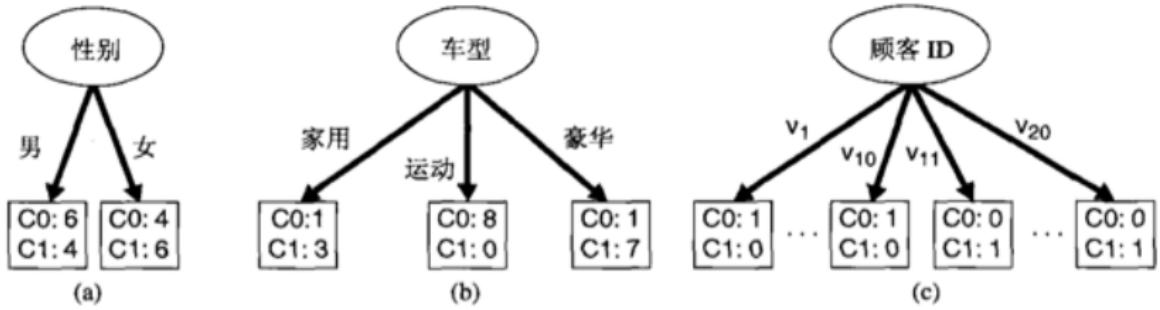
为了要将表格转化为一棵树，决策树需要找出最佳节点和最佳的分枝方法，而衡量这个"最佳"的指标叫做"不纯度"。不纯度基于叶子节点来计算的，所以树中的每个节点都会有一个不纯度，并且子节点的不纯度一定是低于父节点的，也就是说，在同一棵决策树上，叶子节点的不纯度一定是最低的。

> **重要概念：不纯度**
>
> 决策树的每个叶子节点中都会包含一组数据，在这组数据中，如果有某一类标签占有较大的比例，我们就说叶子节点"纯"，分枝分得好。某一类标签占的比例越大，叶子就越纯，不纯度就越低，分枝就越好。
> 如果没有哪一类标签的比例很大，各类标签都相对平均，则说叶子节点"不纯"，分枝不好，不纯度高。

这个其实非常容易理解。**分类型决策树在叶子节点上的决策规则是少数服从多数**，在一个叶子节点上，如果某一类标签所占的比例较大，那所有进入这个叶子节点的样本都回被认为是这一类别。距离来说，如果90%根据规则进入叶子节点的样本都是类别0（叶子比较纯），那新进入叶子节点的测试样本的类别也很有可能是0。但是，如果51%的样本是0，49%的样本是1（极端情况），叶子节点还是会被认为是0类叶子节点，但此时此刻进入这个叶子的测试样本点几乎有一半的可能性应该是类别1。从数学上来说，类分布为（0,100%）的结点具有零不纯性，而均衡分布（50%,50%）的结点具有最高的不纯性。如果叶子本身不纯，那测试样本就很有可能被判断错误，相对的叶子越纯，那样本被判断错误的可能性就越小。

通常来说，不纯度越低，决策树对训练集的拟合越好。现在使用的决策树算法在分枝方法上的核心大多是围绕在对某个不纯度相关指标的最优化上。若我们定义t代表决策树的某节点，D_t是t节点所对应的数据集，设$p(i|t)$表示给定结点t中属于类别i的样本所占的比例，这个比例越高，则代表叶子越纯。

【练习】从下面的图来看，哪个不纯度最高？哪个不纯度最低？



- 怎样计算不纯度？

对于节点不纯度的计算和表示方法因决策树模型而异，但不管不纯度的度量方法如何，都是由**误差率**衍生而来，其计算公式如下：

$$Classification\ error(t) = 1 - \max_{i=1}\ [p(i|t)]$$

误差率越低，则纯度越高。由此还衍生出了其他两个常用指标，一个是ID3中Information gain（信息增益）的计算方法可用Entropy推导，即最为人熟知的**信息熵**，又叫做香农熵，其计算公式如下：

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

其中c表示叶子节点上标签类别的个数，$c-1$表示标签的索引。注意在这里，是从第0类标签开始计算，所以最后的标签类别应该是总共c个标签，c-1为最后一个标签的索引。在计算Entropy时设定$log_2 0 = 0$。

另一个指标则是**Gini（基尼）指数**，主要用于CART决策树的纯度判定中，其计算公式如下：

$$Gini = 1 - \sum_{i=0}^{c-1} [p(i|t)]^2$$

假设在二分类问题中各节点呈现如下分布，则可进一步计算上述三指数的结果

结点 N_1	计数
类 =0	0
类 =1	6

$\text{Gini} = 1 - (0/6)^2 - (6/6)^2 = 0$

$\text{Entropy} = -(0/6)\log_2(0/6) - (6/6)\log_2(6/6) = 0$

$\text{Error} = 1 - \max[0/6, 6/6] = 0$

结点 N_2	计数
类 =0	1
类 =1	5

$\text{Gini} = 1 - (1/6)^2 - (5/6)^2 = 0.278$

$\text{Entropy} = -(1/6)\log_2(1/6) - (5/6)\log_2(5/6) = 0.650$

$\text{Error} = 1 - \max[1/6, 5/6] = 0.167$

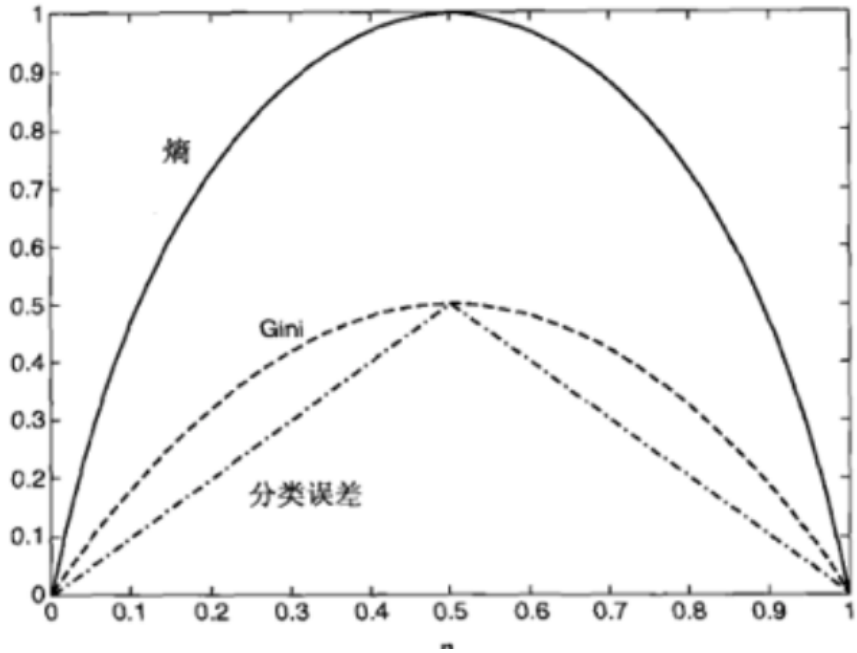
结点 N_3	计数
类 =0	3
类 =1	3

$\text{Gini} = 1 - (3/6)^2 - (3/6)^2 = 0.5$

$\text{Entropy} = -(3/6)\log_2(3/6) - (3/6)\log_2(3/6) = 1$

$\text{Error} = 1 - \max[3/6, 3/6] = 0.5$

能够看出，三种方法本质上都相同，在类分布均衡时（即当p=0.5时）达到最大值，而当所有记录都属于同一个类时（p等于1或0）达到最小值。换而言之，在纯度较高时三个指数均较低，而当纯度较低时，三个指数都比较大，且可以计算得出，熵在0-1区间内分布，而Gini指数和分类误差均在0-0.5区间内分布，三个指数随某变量占比增加而变化的曲线如下所示：



决策树最终的优化目标是使得叶节点的总不纯度最低，即对应衡量不纯度的指标最低。

ID3采用信息熵来衡量不纯度，此处就先以信息熵为例进行讨论。ID3最优条件是叶节点的总信息熵最小，因此**ID3决策树在决定是否对某节点进行切分的时候，会尽可能选取使得该节点对应的子节点信息熵最小的特征进行切分。**换而言之，就是要求父节点信息熵和子节点总信息熵之差要最大。对于ID3而言，二者之差就是信息增益，即Information gain。

但这里需要注意，一个父节点下可能有多个子节点，而每个子节点又有自己的信息熵，所以父节点信息熵和子节点信息熵之差，应该是父节点的信息熵 - 所有子节点信息熵的加权平均。其中，权重是使用单个叶子节点上所占的样本量比上父节点上的总样本量来确定的一个权重。

$$I(child) = \sum_{j=1}^{k} \frac{N(v_j)}{N} I(v_j)$$

而父节点和子节点的不纯度下降数可由下述公式进行计算：

$$\triangle = I(parent) - I(child)$$

I(.)是给定结点的不纯性度量（即是基尼系数或者信息上），*N*是父结点上的样本数，*k*是这一层上子节点的个数，*N(vj)*是与子结点*vj*相关联的样本个数。决策树算法通常选择最大化增益Δ的测试条件，因为对任何分枝过程来说，*I*(parent)都是一个不变的值（因为此时父节点已经存在并且不可修改），所以最大化增益等价于最小化子结点的不纯性衡量的加权平均。最后，当选择熵（entropy）作为公式的不纯性度量时，熵的差就是所谓信息增益（Information gain）Δinfo。

接下来对此进行举例说明。

1.2.2 简单实例

假设现在有如下数据集，是一个消费者个人属性和信用评分数据，标签是"是否会发生购买电脑行为"，仍然是个而分类问题，在此数据集之上我们使用ID3构建决策树模型，并提取有效的分类规则。

	A	B	C	D	E	F
1	ID	age	income	student	credit_rating	Class: buys_computer
2	1	<=30	high	no	fair	no
3	2	<=30	high	no	excellent	no
4	3	31…40	high	no	fair	yes
5	4	>40	medium	no	fair	yes
6	5	>40	low	yes	fair	yes
7	6	>40	low	yes	excellent	no
8	7	31…40	low	yes	excellent	yes
9	8	<=30	medium	no	fair	no
10	9	<=30	low	yes	fair	yes
11	10	>40	medium	yes	fair	yes
12	11	<=30	medium	yes	excellent	yes
13	12	31…40	medium	no	excellent	yes
14	13	31…40	high	yes	fair	yes
15	14	>40	medium	no	excellent	no

首先计算原始数据集的信息熵，我们可设置树模型中每个节点信息熵的表示方法，首先对于根节点而言，信息熵可用$I(s_1, s_2)$来表示，其中s下标1和2代表两个分类水平，s_1和s_2则代表分类水平下的对应样本个数，其计算公式如下所示：

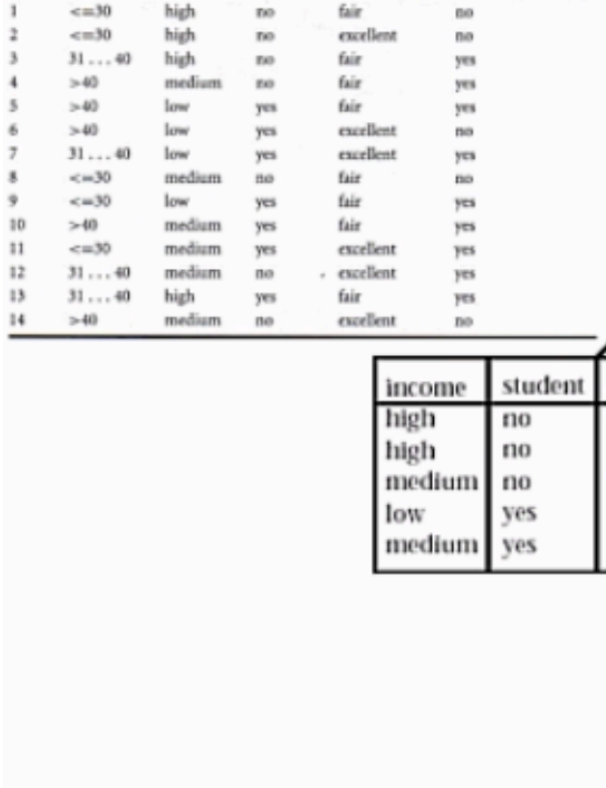
$$Entropy(t) = -\sum_{i=0}^{c-1} p(i|t) log_2 p(i|t)$$

$$I(s_1, s_2) = I(9,5) = -\frac{9}{14} log_2 \frac{9}{14} - \frac{5}{14} log_2 \frac{5}{14} = 0.940$$

即在不进行任何切分前，总信息熵为0.940。然后我们依次选取各特征来尝试进行切分，并计算切分完成后的子节点信息熵是多少。首先选取age列进行切分，age是三分类的离散变量，因此若用age对根节点进行切分，将有三个分支，每个分支分别对应一个age的取值，分支所指向的子节点为对应的切分后的数据集：

Training data tuples from the *AllElectronics* customer database.

RID	age	income	student	credit_rating	Class: buys_computer
1	<=30	high	no	fair	no
2	<=30	high	no	excellent	no
3	31 . . . 40	high	no	fair	yes
4	>40	medium	no	fair	yes
5	>40	low	yes	fair	yes
6	>40	low	yes	excellent	no
7	31 . . . 40	low	yes	excellent	yes
8	<=30	medium	no	fair	no
9	<=30	low	yes	fair	yes
10	>40	medium	yes	fair	yes
11	<=30	medium	yes	excellent	yes
12	31 . . . 40	medium	no	excellent	yes
13	31 . . . 40	high	yes	fair	yes
14	>40	medium	no	excellent	no



income	student	credit_rating	class
high	no	fair	no
high	no	excellent	no
medium	no	fair	no
low	yes	fair	yes
medium	yes	excellent	yes

income	student	credit_rating	class
medium	no	fair	yes
low	yes	fair	yes
low	yes	excellent	no
medium	yes	fair	yes
medium	no	excellent	no

income	student	credit_rating	class
high	no	fair	yes
low	yes	excellent	yes
medium	no	excellent	yes
high	yes	fair	yes

yes

然后我们计算子节点的信息熵：

For *age* = "<=30":

$s_{11} = 2 \quad s_{21} = 3 \quad I(s_{11}, s_{21}) = 0.971$

For *age* = "31 . . . 40":

$s_{12} = 4 \quad s_{22} = 0 \quad I(s_{12}, s_{22}) = 0$

For age = ">40":

$s_{13} = 3 \quad s_{23} = 2 \quad I(s_{13}, s_{23}) = 0.971$

the expected information needed to classify a given sample

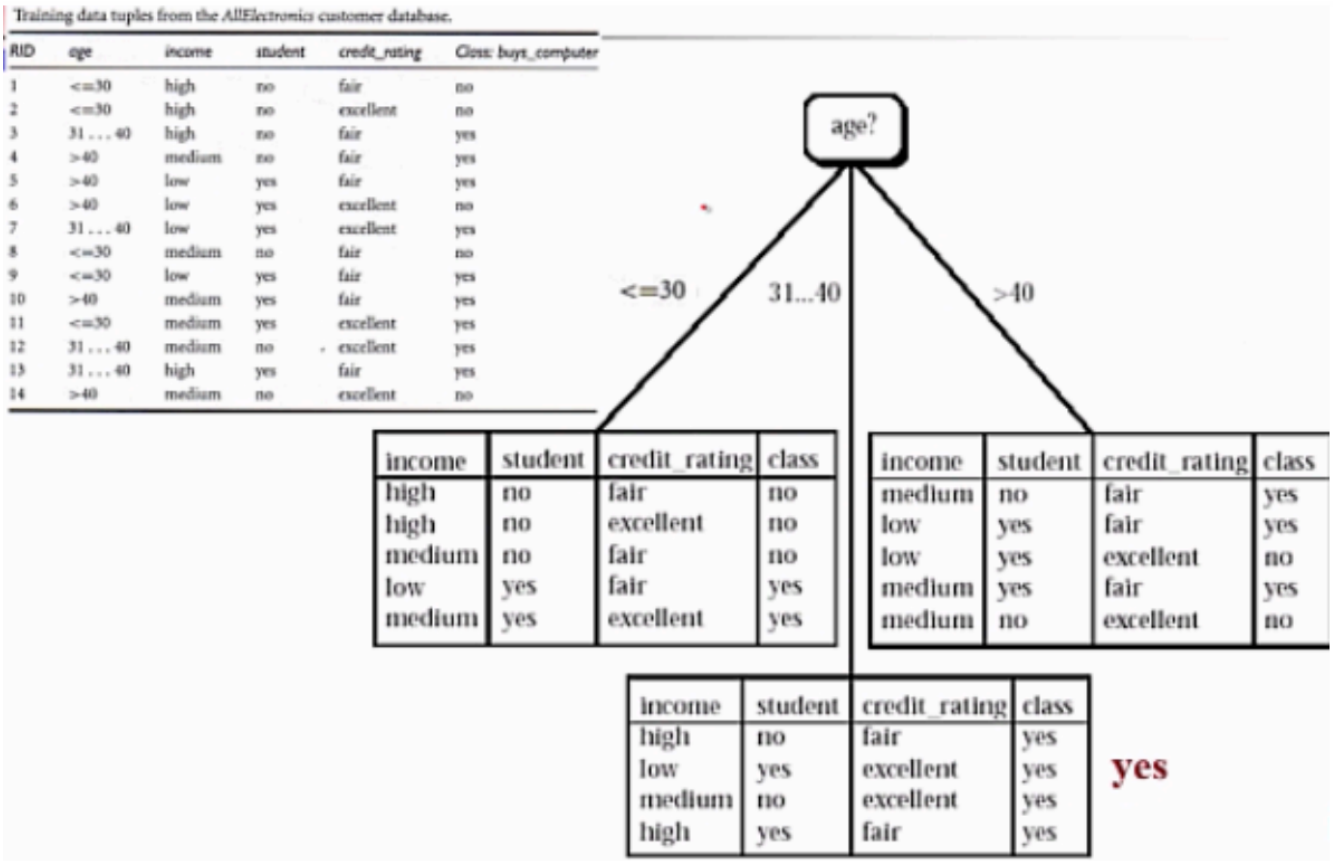
$$E(age) = \frac{5}{14} I(s_{11}, s_{21}) + \frac{4}{14} I(s_{12}, s_{22}) + \frac{5}{14} I(s_{13}, s_{23}) = 0.694$$

即对于age属性而言，在根节点上切分一次之后子节点的信息熵为0.694，由此我们可计算age的信息增益，即局部最优化判别指标：

$$Gain(age) = I(s_1, s_2) - E(age) = 0.246$$

以此类推，我们还能计算其他几个特征的信息增益，最终计算结果如下

Gain(income) = 0.029，Gain(student) = 0.15，Gain(credit_rating) = 0.048，很明显，第一次切分过程将采用age字段进行切分：



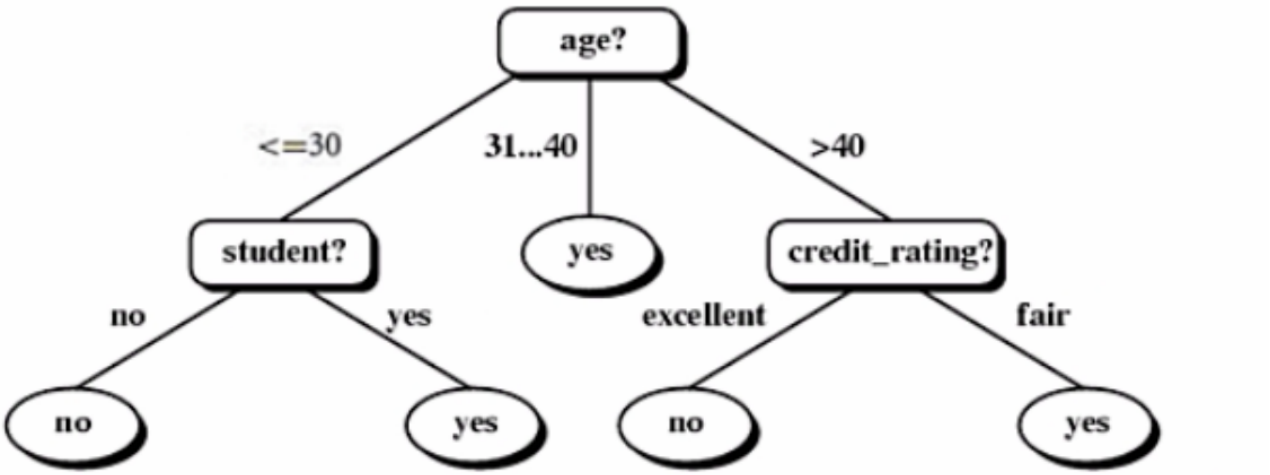
age="<=30"数据集

ID	age	income	student	credit_rating	Class: buys_computer
1	<=30	high	no	fair	no
2	<=30	high	no	excellent	no
8	<=30	medium	no	fair	no
9	<=30	low	yes	fair	yes
11	<=30	medium	yes	excellent	yes

age=">40"数据集

ID	age	income	student	credit_rating	Class: buys_computer
1	<=30	high	no	fair	no
2	<=30	high	no	excellent	no
8	<=30	medium	no	fair	no
9	<=30	low	yes	fair	yes
11	<=30	medium	yes	excellent	yes

第一次切分完成后，分成了三个数据集，其中age=“31...40"分类指标所指向的数据集纯度为1，因此不用再进行切分，而其他两个数据集则需要进一步进行切分，对于age="<=30"的数据集而言使用student特征进行切分后子节点纯度就将为1，而age=">40"的数据集则可采用credit_rating字段进行切分。最终ID3决策树模型如下所示：



总的来说，决策树模型是一个典型的贪心模型，总目标是一个全局最优解，即一整套合理的分类规则使得最终叶节点的纯度最高，但全局最优解在随特征增加而呈现指数级增加的搜索空间内很难高效获取，因此我们退而求其次，考虑采用局部最优来一步步推导结果——只要保证信息增益最大，我们就能得到次最优的模型。当然，局部最优不一定等于全局最优，接下来我们就ID3可能存在的一些问题及改进方向进行一些讨论。

1.2.3 ID3的局限性

ID3局限主要源于局部最优化条件，即信息增益的计算方法，其局限性主要有以下几点：

- 分支度越高（分类水平越多）的离散变量往往子节点的总信息熵会更小，ID3是按照某一列进行切分，有一些列的分类可能不会对我需要的结果有足够好的指示。极限情况下取ID作为切分字段，每个分类的纯度都是100%，因此这样的分类方式是没有效益的
- 不能直接处理连续型变量，若要使用ID3处理连续型变量，则首先需要对连续变量进行离散化
- 对缺失值较为敏感，使用ID3之前需要提前对缺失值进行处理
- 没有剪枝的设置，容易导致过拟合，即在训练集上表现很好，测试集上表现很差

机器学习的关键概念：过拟合与欠拟合

过拟合：模型在训练集上表现很好，在测试集上表现很糟糕，学习能力很强但是学得太过精细了
欠拟合：模型在训练集和测试集上都表现糟糕，学习能力不足

关于剪枝和过拟合相关问题，我们会在最后进行详细讨论，此处先讨论其他局限性如何解决。对于ID3的诸多优化措施，最终也构成了C4.5算法的核心内容。

1.3 C4.5算法 & CART算法

1.3.1 修改局部最优化条件

在C4.5中，首先通过引入分支度（IV：Information Value）（在《数据挖掘导论》一书中被称为划分信息度）的概念，来对信息增益的计算方法进行修正，简而言之，就是在信息增益计算方法的子节点总信息熵的计算方法中添加了随着分类变量水平的惩罚项。而分支度的计算公式仍然是基于熵的算法，只是将信息熵计算公式中的$p(i|t)$(**即某类别样例占总样例数**)改成了$P(v_i)$，即**某子节点的总样本数占父节点总样本数的比例**，这其实就是我们加权求和时的"权重"。这样的一个分支度指标，让我们在切分的时候，自动避免那些分类水平太多，信息熵减小过快的特征影响模型，减少过拟合情况。IV计算公式如下：

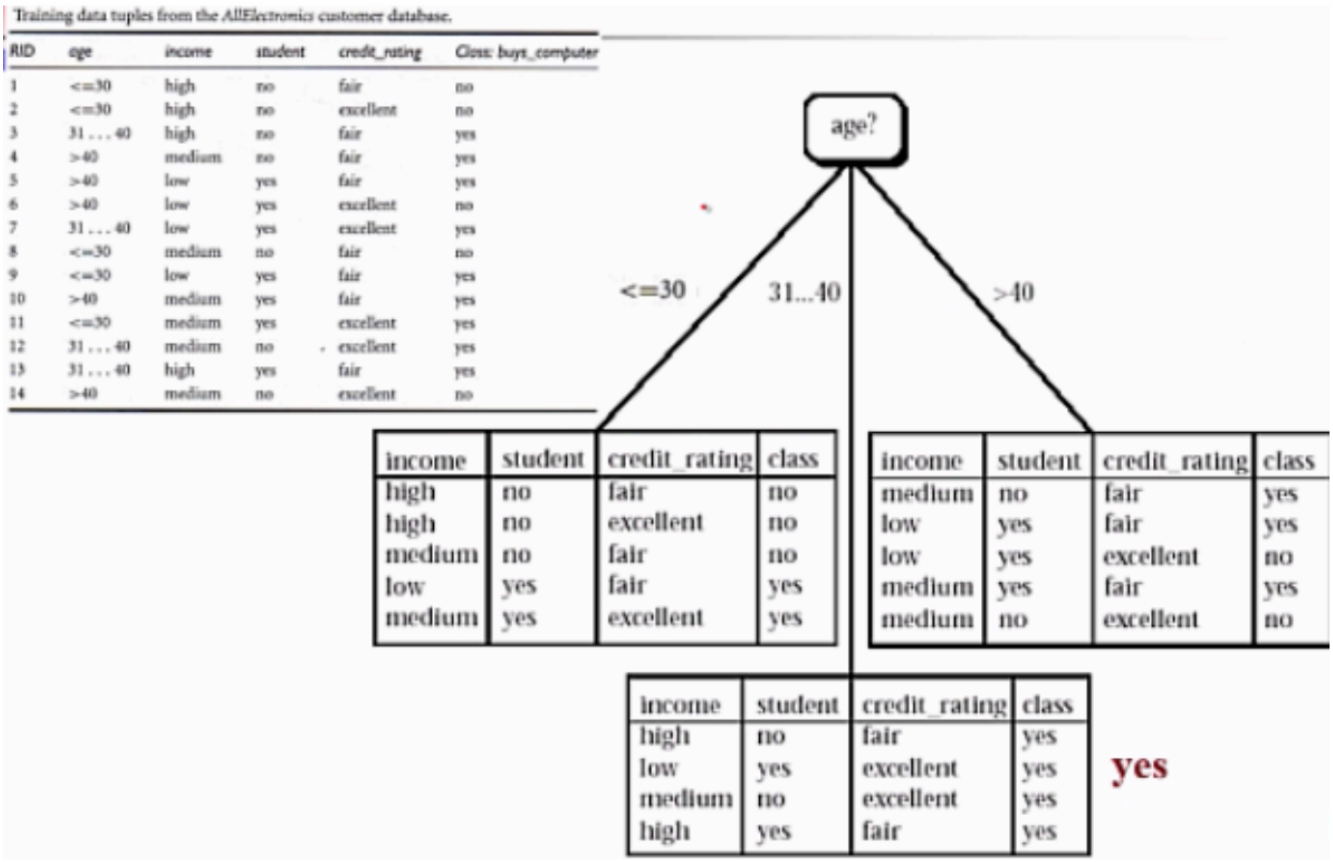
$$Information\ Value = -\sum_{i=1}^{k} P(v_i) log_2 P(v_i)$$

其中，i表示父节点的第i个子节点，v_i表示第i个子节点样例数，$P(v_i)$表示第i个子节点拥有样例数占父节点总样例数的比例。很明显，IV可作为惩罚项带入子节点的信息熵计算中。可以简单计算得出，当取ID字段作为切分字段时，IV值为$log_2 k$。所以IV值会随着叶子节点上样本量的变小而逐渐变大，这就是说一个特征中如果标签分类太多，每个叶子上的IV值就会非常大。

最终，在C4.5中，使用之前的信息增益除以分支度作为选取切分字段的参考指标，该指标被称作Gain Ratio（获利比例，或增益率），计算公式如下：

$$Gain\ Ratio = \frac{Information\ Gain}{Information\ Value}$$

增益比例是我们决定对哪一列进行分枝的标准，我们分枝的是数字最大的那一列，本质是信息增益最大，分支度又较小的列（也就是纯度提升很快，但又不是靠着把类别分特别细来提升的那些特征）。IV越大，即某一列的分类水平越多，Gain ratio实现的惩罚比例越大。当然，我们还是希望GR越大越好。



然后我们可利用GR代替Information Gain重新计算1.2.3的实例，例如计算age字段的GR，由于根据age字段切分后，3个分支分别有5个、4个和5个样例数据，因此age的IV指标计算过程如下：

$$IV(age) = -\frac{5}{14}log_2\frac{5}{14} - \frac{4}{14}log_2\frac{4}{14} - \frac{5}{14}log_2\frac{5}{14} = 1.577$$

进而可计算age列的GR：

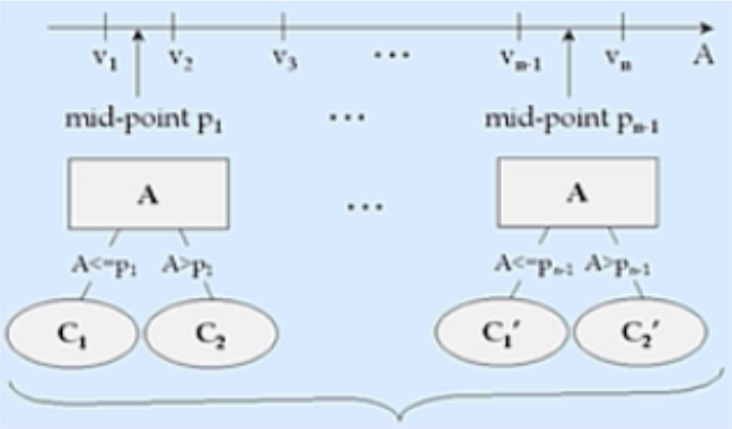
$$GR(age) = \frac{Gain(age)}{IV(age)} = \frac{0.246}{1.577} = 0.156$$

然后可进一步计算其他各字段的GR值，并选取GR值最大的字段进行切分。

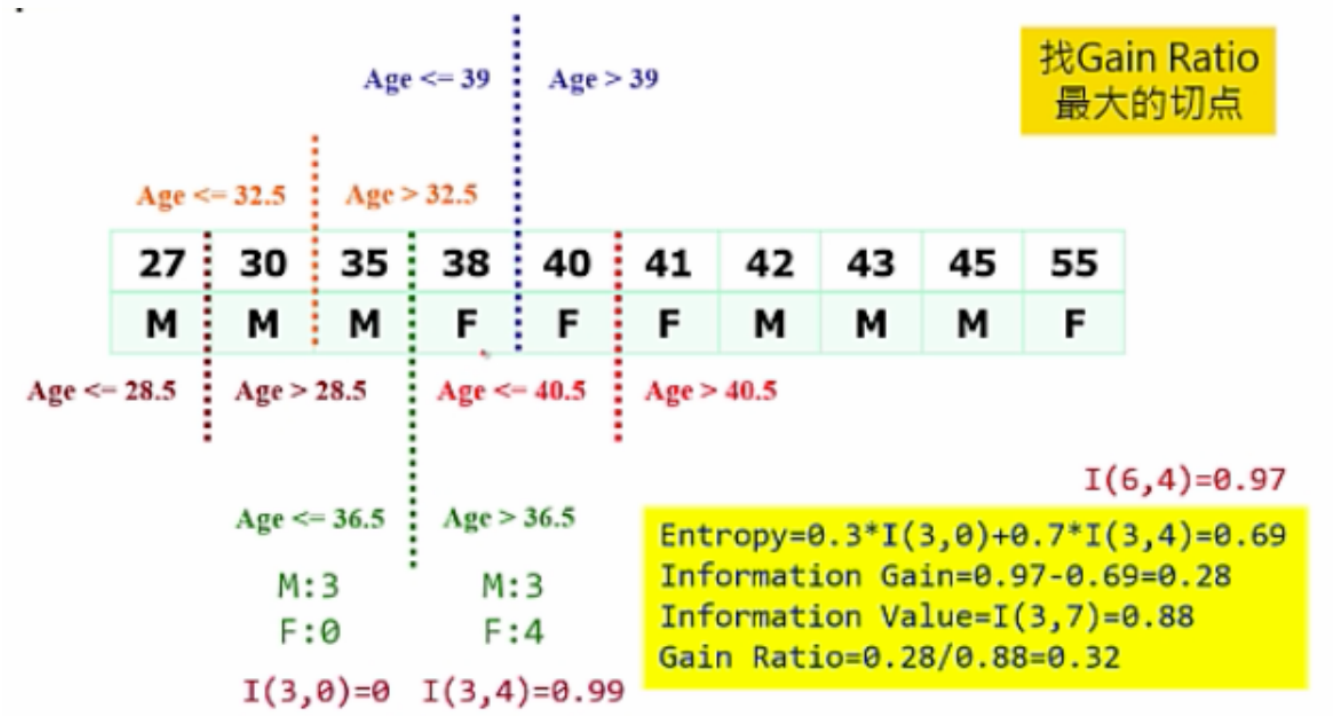
1.3.2 连续变量处理手段

在C4.5中，同样还增加了针对连续变量的处理手段。如果输入特征字段是连续型变量，则有下列步骤：

1. 算法首先会对这一列数进行从小到大的排序
2. 选取相邻的两个数的中间数作为切分数据集的备选点，若一个连续变量有N个值，则在C4.5的处理过程中将产生N-1个备选切分点，并且每个切分点都代表着一种二叉树的切分方案，例如：



这里需要注意的是，**此时针对连续变量的处理并非是将其转化为一个拥有N-1个分类水平的分类变量，而是将其转化成了N-1个二分方案**，而在进行下一次的切分过程中，这N-1个方案都要单独带入考虑，其中每一个切分方案和一个离散变量的地位均相同（一个离散变量就是一个单独的多路切分方案）。例如有如下数据集，数据集中只有两个字段，第一行代表年龄，是特征变量，第二行代表性别，是目标字段，则对年龄这一连续变量的切分方案如图所示：



从上述论述能够看出，在对于包含连续变量的数据集进行树模型构建的过程中要消耗更多的运算资源。但于此同时，我们也会发现，当连续变量的某中间点参与到决策树的二分过程中，往往代表该点对于最终分类结果有较大影响，这也为我们连续变量的分箱压缩提供了指导性意见，例如上述案例，若要对age列进行压缩，则可考虑使用36.5对其进行分箱，则分箱结果对于性别这一目标字段仍然具有较好的分类效果，这也是决策树的最常见用途之一，也是最重要的模型指导分箱的方法。

现在被大量使用的是C4.5的改进CART树，CART树本质其实和C4.5区别不大，只不过CART树所有的层都是二叉树，也就是每层只有两个分枝。让我们用CART树来走过一遍C4.5的流程。假设年龄是特征，性别是标签：

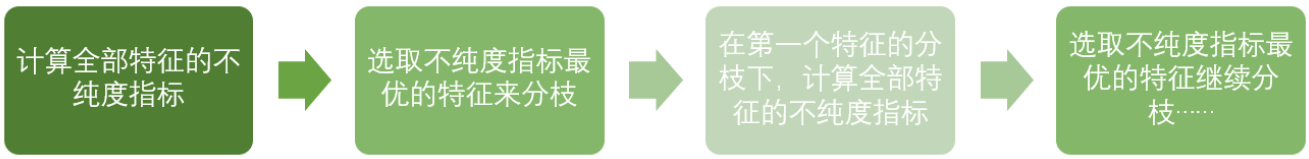
1. 首先将年龄从小到大依此进行排列
2. 然后计算俩俩相邻的年龄的均值
3. 按均值所在的点，对连续性变量进行二分（变成数字形式的，第一类是>均值，第二类是<均值）， 二分得到的点叫做决策树的“ 树桩”。这是说，对于一个含有n个记录的连续性变量来说，就会存在n-1个潜在切分点，这n-1个潜在切分点的获益比例都都会被计算

4. 找每种二分切分方案的获益比例，获益比例最大的切分点，就是切点
5. 切完之后，计算加权信息熵，计算信息增益，引入分支度，可以计算出增益比例了

需要注意的是，这种切分方法，每次切分之后，并没有消费掉一列，只消费掉了一个备选点。在我们原本的切分方法中，ID3每次切分就会消费掉一整个列，但实际上，我们可以只关注每次分类的时候，对信息熵的减少贡献最多的那个分类。按我们的例子来说，我们在分类age的时候，最关注的是31~40岁的那一个分类，我们完全可以实现，31~40一类，其他算一类，然后我们再对“其他”这个类别进行相似的二分类。这就是CART的核心原理，大量地减少了计算的量。

对于KNN算法，我们有一个假设：就是每一个特征对于我们的推断的重要性是一样的。这也是KNN最大的缺陷。而决策树天生就认为每个特征对于推断的重要性不是一样的，而CART则是进一步认为，每个特征下的每个分类对于推断的重要性也不是一样的。

到这里，决策树的基本流程其实可以简单概括如下：



直到没有更多的特征可用，或整体的不纯度指标已经最优，决策树就会停止生长。