决策树

四个部分：属性选择，处理数字属性，处理缺失值，修剪以避免过拟合

基本概念

决策树由节点，边和叶组成

- 节点：测试某个属性的值，树中，最上面的节点是根节点
- 边：对应测试的结果，连接到下一个节点或者叶子
- 叶子：预测结果的终端节点（类标签）

决策树：在层次结构中组织一些类确定示例类标签的测试

注意：改变特征测试的顺序，可以得到一个完全不同的树

什么是好的决策树?

·通常目标是通过最小化泛化误差来找到最优决策树。

·还可以定义其他目标函数，例如，最小化节点数量或最小化平均深度。

寻找与训练数据一致的最小决策树是一个np困难问题，因此我们使用**启发式贪婪方法迭代**地生长具有训练样本的决策树。

决策树的特点：
优点：

计算复杂度不高，输出结果易于理解，对中间值的缺失不敏感，可以处理不相关特征数据。

1.因为是非参数模型，不需要对样本进行预先假设，可以处理复杂样本。

2.计算速度快，结果可解释性强。

3.可以同时处理分类和预测问题，对缺失值不敏感。

缺点：可能会产生过度匹配的问题（需要剪枝）。 适用数据类型：数值型和标称型

1.容易过拟合
2.特征之间存在相互关联时，数据结果表现较差。

决策树的构建过程：

步骤1：将所有的数据看成是一个节点，进入步骤2；

步骤2：从所有的数据特征中挑选一个最优数据特征对节点进行分割，使得分割后的子集有一个在当前条件下最好的分类，进入步骤3；

步骤3：生成若干孩子节点，对每一个孩子节点进行判断，如果满足停止分裂的条件，进入步骤4；否则，进入步骤2；

步骤4：设置该节点是子节点，其输出的结果为该节点数量占比最大的类别。

决策树是一个由根到叶的**递归过程**，在每一个中间结点寻找划分属性，递归重要的是设置**停止条件：**

- （1）当前结点包含的样本属于同一类别，无需划分；
- （2）当前属性集为空，或是所有样本在所有属性上取值相同无法划分，简单理解就是当分到这一节点时，所有的属性特征都用完了，没有特征可用了，就根据label数量多的给这一节点打标签使其

变成叶节点（其实是在用样本出现的后验概率做先验概率）；

- （3）当前结点包含的样本集合为空，不能划分。这种情况出现是因为该样本数据缺少这个属性取值，根据父结点的label情况为该结点打标记（其实是在用父结点出现的后验概率做该结点的先验概率）。

属性(测试)选择标准

在这里属性（特征）的类型分为二种，一种是离散型的，另一种是连续的。比如西瓜的大小，分为大和小二种，类似这样的特征就是离散型的特征。再比如西瓜的重量，2kg、2.2kg...，这种数值型的特征。、

决策树如何进行分类（如何判断根据哪个属性进行划分）

这里主要有几个衡量指标，**信息熵、条件熵、信息增益、信息增益率、基尼系数、误分类率**，

熵（Entropy -1984年香农提出）的概念：一条信息的信息量大小与它的不确定性有直接关系，而熵就是用来度量这件事不确定性的大小。为什么需要了解“熵”这个概念？因为决策树模型的构建和评估都与信息熵密切相关。而 **信息熵（Information Entropy）**是用来度量样本纯度的指标。

ID3决策树需要“最大化信息增益”来对节点进行划分，以下是信息增益计算步骤:

> 输入：训练集D和属性a (这里每个属性a有V个可能的取值{ $a^1, a^2, \ldots, a^v$ })
> 输出：属性a对训练数据集D的信息增益 $Gain(D, a)$
>
> 1. 假定样本集合D中第i类样本所占的比例为 p_i ($(i = 1, 2, 3, \ldots, |n|)$,需要先求信息熵
> 信息熵公式:
> $Ent(D) = -\sum_{i=1}^{|n|} p_i log_2 p_i$ （PS: 这里 $Ent(D)$ 的值越小，样本D的纯度越高。）
> 2. 求离散特征a对数据集D的条件信息熵
> 条件信息熵公式:
> $Ent(D|a) = \sum_{v=1}^{V} \frac{|D^v|}{|D|} Ent(D^v)$
> 3. 计算信息增益(Information Gain)
> 信息增益公式:
> $Gain(D, a) = Ent(D) - Ent(D|a)$

注意：这里在手算的过程中一般会计算出多个信息增益，**选最大的!**

三、C4.5决策树

C4.5是为了解决ID3的一个缺点而产生的。缺点是啥？**如果某个属性的分类很多，也就是分叉超多，那么该属性下的样本就很少，此时的信息增益就非常高，ID3这个愣头青就会认为这个属性适合用作划分。**是，它确实是能划分，但取值较多的属性用作划分依据时，它的泛化能力弱，没法对新样本有效预测，所以**C4.5不依靠信息增益划分样本，而是依靠“信息增益率”。**

这里有个新名词，属性a的“固有值（Intrinsic Value）”，当属性a的可取值数量越大（也就是V越大），那IV(a)的值就越大。以下是C4.5的计算步骤：

> 1. 根据ID3的信息熵和条件信息熵 求信息增益
> 信息增益公式：
> $Gain(D,a) = Ent(D) - Ent(D|a)$
> 2. 计算属性a的固有值IV(a)
> 固有值计算公式：
> $IV(a) = -\sum_{v=1}^{V} \frac{|D^v|}{|D|} log_2 \frac{|D^v|}{|D|}$
> 3. 根据信息增益和固有值求信息增益率
> 信息增益率计算公式：
> $GainRatio(D,a) = \frac{Gain(D,a)}{IV(a)}$

注意：这里不是无脑选择最高的信息增益率，而是启发式地选择：**先从划分出的属性中找到信息增益高于平均的那些属性，然后再从这些属性中选信息增益率最高的。**

直观讲：**基尼指数Gini(D)** 反映的是数据集中随机抽取两个样本，而他们类别标志不一致的概率。*（e.g.从100封邮件中随机抽两个，而这两个邮件“是垃圾邮件”和“不是垃圾邮件”的概率）***基尼指数越小，代表数据集D的纯度越高。**

步骤：

> 1. 计算基尼指数
> $Gini(D) = -\sum_{i=1}^{|n|} \sum_{i' \neq i} p_i p_{i'} = 1 - \sum_{i=1}^{|n|} p_i^2$
> 2. 对于属性a计算条件基尼指数
> $GiniIndex(D,a) = -\sum_{v=1}^{V} \frac{|D^v|}{|D|} Gini(D^v)$

注意：**在属性集合A中，一般选基尼指数最小的属性**，即 $a_* = argMinGiniIndex(D,a)$ 。

数值属性——连续值处理

Numeric (Continuous) Attributes

- Standard method: **binary partition**
  - E.g. temp < 45 and temp ≥ 45
- Solution is straightforward:
  - Sort the attribute values $\{a^1, a^2, \dots, a^n\}$
  - Evaluate info gain (or other measure) for every possible split point of attribute
  - Split points can be placed between values, i.e., $(a^i + a^{i+1})/2$, or directly at values a^i
  - Choose the "best" split point
  - Info gain for best split point is info gain for attribute

$$Gain(D,a) = \max_{t \in T_a} Gain(D,a,t) \qquad T_a = \{\frac{a^i + a^{i+1}}{2} | 1 \le i \le n-1\}$$

Split candidates

- Computationally more demanding

给定训练集D和连续属性a，假定a在D上出现了n个不同的取值，先把这些值从小到大排序，记为$\{a^1, a^2, \dots, a^n\}$. 基于划分点t可将D分为子集D_t^-和D_t^+，其中D_t^-是包含那些在属性a上取值不大于t的样本，D_t^+则是包含那些在属性a上取值大于t的样本。显然，对相邻的属性取值a^i与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相同。因此，对连续属性a，我们可考察包含$n-1$个元素的候选划分点集合

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

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，我们就可以像前面处理离散属性值那样来考虑这些划分点，选择最优的划分点进行样本集合的划分，使用的公式如下：

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

其中$Gain(D,a,t)$是样本集D基于划分点t二分后的信息增益。划分的时候，选择使$Gain(D,a,t)$最大的划分点。

缺失值——缺失值处理

Missing Values

- Assume that attribute a has V possible values $\{a_1, a_2, \ldots, a_V\}$.
- $\tilde{S}$: subset of instances in S whose values of a are not missing.
- $\tilde{S}_i$: subset of instances in $\tilde{S}$ whose values of attribute a is a_i.
- $\tilde{S}^k$: subset of instances in $\tilde{S}$ belonging to the k-th class (k=1,..., $|\mathcal{Y}|$).
- Originally,

$$Gain(S,a) = E(S) - I(S,a) = E(S) - \sum_i \frac{|S_i|}{|S|} E(S_i)$$

- Now we modify the information gain as follows,

$$Gain(S,a) = \rho \times Gain(\tilde{S},a) = \rho \times \left(E(\tilde{S}) - \sum_i \tilde{r}_i E(\tilde{S}_i) \right)$$

proportionally $\rho = \frac{|\tilde{S}|}{|S|}$ $\tilde{r}_i = \frac{|\tilde{S}_i|}{|\tilde{S}|}$ Ignore all instances whose values of attribute a are unknown

$$E(\tilde{S}) = -\sum_{k=1}^{|\mathcal{Y}|} \tilde{p}_k \log_2 \tilde{p}_k \qquad \tilde{p}_k = \frac{|\tilde{S}^k|}{|\tilde{S}|}$$

Missing Values

- In the case of gain ratio, the denominator should be calculated as if the missing values represent an additional value in the attribute domain.

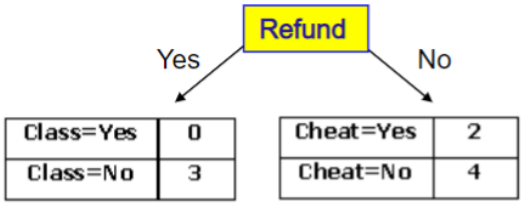
$$GR(S,a) = \frac{Gain(S,a)}{IntI(S,a)} \qquad IntI(S,a) = -\sum_i \frac{|S_i|}{|S|} \log\left(\frac{|S_i|}{|S|}\right)$$

$$GR(S,a) = \frac{\rho \times Gain(\tilde{S},a)}{-\frac{|S\setminus\tilde{S}|}{|S|}\log\left(\frac{|S\setminus\tilde{S}|}{|S|}\right) - \sum_i \frac{|\tilde{S}_i|}{|S|}\log\left(\frac{|\tilde{S}_i|}{|S|}\right)}$$

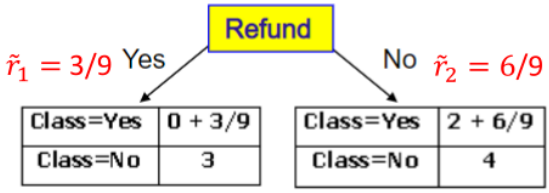
如何给定一个缺少值的实例:

Tid	Refund	Marital Status	Taxable Income	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

Tid	Refund	Marital Status	Taxable Income	Class
10	?	Single	90K	Yes



$$\tilde{r}_i = \frac{|\tilde{S}_i|}{|\tilde{S}|}$$



Probability that Refund=Yes is 3/9

Probability that Refund=No is 6/9

Assign record to the left child with weight = 3/9 and to the right child with weight = 6/9

缺失值以三种方式影响决策树的构建:

- 影响杂质测量的计算方式

$$Gain(S,a) = \rho \times Gain(\tilde{S},a) = \rho \times \left(E(\tilde{S}) - \sum_i \tilde{r}_i E(\tilde{S}_i)\right)$$

$$GR(S,a) = \frac{\rho \times Gain(\tilde{S},a)}{-\frac{|S\setminus\tilde{S}|}{|S|}\log\left(\frac{|S\setminus\tilde{S}|}{|S|}\right) - \sum_i \frac{|\tilde{S}_i|}{|S|}\log\left(\frac{|\tilde{S}_i|}{|S|}\right)}$$

- 影响如何将缺少值的实例分发到子节点

Affects how to distribute instance with missing value to child nodes

- Fractional instances (pieces) $\tilde{r}_i = \frac{|\tilde{S}_i|}{|\tilde{S}|}$

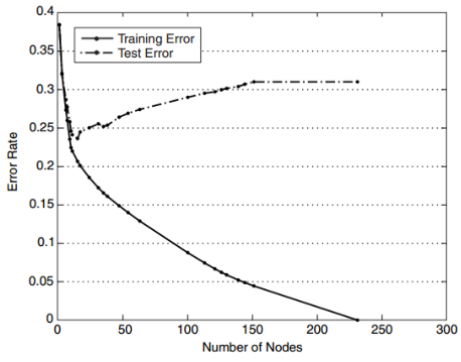
- 影响如何对缺失值的测试实例进行分类

问:只有纯叶子的树总是最好的分类器吗?答:不是。

这棵树在训练集上是最好的分类器，但在新的和未见过的数据上可能不是。

由于过度拟合，树可能不能很好地泛化。

过度拟合



- 欠拟合是因为模型还没有学习数据的真实结构。
- 过拟合的两个原因：噪声点，有代表性的样本点少
- 当树变得太大时，就会出现过拟合。
- 尽管训练误差持续减少，但测试误差开始增加。
- 树的叶节点可以扩展，直到它完全适合训练数据。
- 注意，对于决策树，总是可以找到训练数据的“完美”拟合!(除非数据相互矛盾)
- 虽然复杂树的训练误差可以为零，但它的测试误差可能很大，因为树可能包含碰巧适合某些噪声点的节点。此类节点导致泛化性能较差。
- 与更复杂的树相比，包含较少节点数的树具有更高的训练错误率，但测试错误率较低。
- 缺乏代表性样本导致的过拟合
- 基于少量训练记录做出分类决策的模型也容易出现过拟合。
- 避免过拟合的两种策略：

  Pre-Pruning:•在生成一棵完全适合所有训练数据的成熟树之前停止树生长算法

  Post-Pruning:•生长一棵正确分类所有训练数据的决策树•稍后通过用叶子替换一些节点来简化它

  实践中首选后修剪——预修剪可以“早停”

剪枝——决策树调优方法

在上面我们提到了信息增益、增益率、基尼系数三个不同的划分属性，有人会疑惑，不同的划分原则是否会对结果产生影响，研究表明划分选择的各种准则虽然对决策树的尺寸有较大的影响，但对泛化性能的影响有限。相比而言，剪枝方法和程度对决策树泛化性能的影响更为显著。决策树从上到下划分实际上完成的是从全部到局部的划分，分到局部时可能会受到噪音的影响，容易产生不必要的分枝而过拟合，剪枝是决策树对付“过拟合”的主要手段。

预修剪(提前停止)

停止的三个条件

节点的典型停止条件为:

- 所有实例属于同一类时停止
- 所有属性值相同时停止

更多限制性条件:

- 如果实例数量少于某个阈值，则停止
- 如果实例的类分布独立于可用属性(即，节点上的任何属性和类之间没有统计上显著的关联)，则停止。
- 如果扩大当前节点，则杂质测量中观察到的增益低于某一阈值。

后剪枝：

基本思想：

首先生长一棵完整的树以捕获所有可能的属性交互

然后以自下而上的方式修剪完全生长的树

如果修剪后泛化误差有所改善，则用叶节点替换子树。

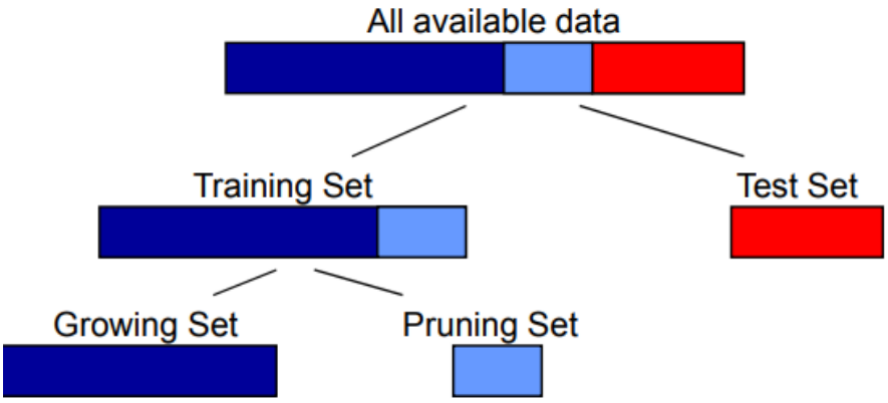
叶节点的类标签由子树中大多数实例的类确定。

用一个新的叶节点替换一个子树，该叶节点的类标签由该子树的大多数记录类决定。

用树归纳法划分数据

在新数据上估计树的精度:“测试集”

•一些后剪枝方法需要一个独立的数据集:“剪枝集”



使用“修剪集”的问题:“生长集”的数据较少

剪枝的算法:

- 减少错误修剪(REP) Reduced Error Pruning

  当树生成后，将子树替换成其叶节点，类别按叶节点中样本最多的类，然后判断剪和不剪之间的性能差异，然后再决定剪不剪枝。

- **代价复杂度剪枝** (CCP) Cost Complexity Pruming

  代价复杂度剪枝分为两步：

  第一步：**在所有非叶子节点中寻找代价复杂度参数最小的节点**，其中代价复杂度参数为：

$$\alpha = \frac{R(T) - R(T_t)}{L(T) - 1}$$

$$R(T) = e(T) \times p(T)$$

$$R(T_t) = \sum_{t \in T} e(t) \times p(t)$$

其中e(T)为错分率，p(T)为该节点所覆盖的样本量占总样本量的比例，L(T)为T节点下叶子节点的个数。

第二步：循环对代价复杂度参数最小的节点进行剪枝（有多个节点同时取到最小值时取叶子节点最多的节点），直到只剩下根节点，可得到一系列的剪枝数{T0, T1, T2, ..., Tm}，其中T0为原始的决策树，Tm为根节点，Ti+1为Ti剪枝后的结果。在这一系列的剪枝树中，根据实际的误差估计决定最优的决策树。

代价复杂度参数α可以理解为代价和复杂度之间的关系，剪枝后叶子节点的个数（复杂度）减少，但是错分样本个数（代价）增多了。CCP是在一系列子树中选择最优树，因此结果也较为准确。

交叉验证

将数据分成k大小相等的分区。

•每次运行时，选择一个分区进行测试，其余分区用于训练。

此过程重复k次，以便每个分区只用于测试一次。

总误差是通过对所有甑运行的误差进行平均得到的。

决策树的特征

建立分类模型的非参数方法。

参数模型与非参数模型

非参数模型（non-parametric model）和参数模型（parametric model）作为数理统计学中的概念，现在也常用于机器学习领域中。在统计学中，参数模型通常假设总体服从某个分布，这个分布可以由一些参数确定，如正态分布由均值和标准差确定，在此基础上构建的模型称为参数模型；非参数模型对于总体的分布不做任何假设或者说是数据分布假设自由，只知道其分布是存在的，所以就无法得到其分布的相关参数，只能通过非参数统计的方法进行推断。

所以说，参数模型和非参数模型中的"参数"并不是模型中的参数，而是数据分布的参数。问题中有没有参数，并不是参数模型和非参数模型的区别。其区别主要在于总体的分布形式是否已知。而为何强调"参数"与"非参数"，主要原因在于参数模型的分布可以有参数直接确定。

参数模型、非参数模型（以及半参数模型）的概念应该源自于统计学中。统计专业中有一门课程叫做《非参数统计》，研究的对象就是秩检验、核密度估计等。

对几篇博客的分析进行了总结和浓缩：

一、非参数模型并不是说模型中没有参数！而是参数很多或者说参数不确定。

这里的non-parametric类似单词priceless，并不是没有价值，而是价值非常高，无价，也就是参数是非常非常非常多的！（注意：所谓"多"的标准，就是参数数目大体和样本规模差不多）

而：可以通过有限个参数来确定一个模型，这样的方式就是"有参数模型"，也就是这里说的参数模型，如线性回归、Logistic回归（假定样本维度为N，则假定N个参数theta1,theta2...thetaN）。

二、参数模型 ： 对学到的函数方程有特定的形式，也就是明确指定了目标函数的形式 -- 比如线性回归模型，就是一次方程的形式，然后通过训练数据学习到具体的参数。

假设可以极大地简化学习过程，但是同样可以限制学习的内容。简化目标函数为已知形式的算法就称为参数机器学习算法。

通过固定大小的参数集(与训练样本数独立)概况数据的学习模型称为参数模型。不管你给与一个参数模型多少数据，对于其需要的参数数量都没有影响。— Artificial Intelligence: A Modern Approach，737页

所以参数机器学习模型包括两个部分：
1、选择合适的目标函数的形式。
2、通过训练数据学习目标函数的参数。

通常来说，目标函数的形式假设是：对于输入变量的线性联合，于是参数机器学习算法通常被称为"线性机器学习算法"。那么问题是，实际的未知的目标函数可能不是线性函数。它可能接近于直线而需要一些微小的调节。或者目标函数也可能完全和直线没有关联，那么我们做的假设是错误的，我们所做的近似就会导致差劲的预测结果。（容易欠拟合）

三、非参数机器学习算法：对于目标函数形式不作过多的假设的算法称为非参数机器学习算法。通过不做假设，算法可以自由的从训练数据中学习任意形式的函数。

当你拥有许多数据而先验知识很少时，非参数学习通常很有用，此时你不需要关注于参数的选取。— Artificial Intelligence: A Modern Approach，757页

非参数理论寻求在构造目标函数的过程中对训练数据作最好的拟合，同时维持一些泛化到未知数据的能力。同样的，它们可以拟合各种形式的函数。（所以说，非参数有目标函数，但是我们不知道目标函数的具体形式也不做出过多假设？）

对于理解非参数模型的一个好例子是k近邻算法，其目标是基于k个最相近的模式对新的数据做预测。这种理论对于目标函数的形式，除了相似模式的数目以外不作任何假设。

四、最后：

常见的参数机器学习模型有：
1、逻辑回归（logistic regression）
2、线性成分分析（linear regression）
3、感知机（perceptron）（假设分类超平面是wx+b=0）

参数机器学习算法有如下优点:
1、简洁：理论容易理解和解释结果。
2、快速：参数模型学习和训练的速度都很快。
3、数据更少：通常不需要大量的数据，在对数据的拟合不很好时表现也不错。

参数机器学习算法的局限性：
1、拘束：以指定的函数形式来指定学习方式。
2、有限的复杂度：通常只能应对简单的问题。
3、拟合度小：实际中通常无法和潜在的目标函数完全吻合，也就是容易出现欠拟合。

常见的非参数机器学习模型有：
1、决策树
2、朴素贝叶斯
3、支持向量机（SVM的例子中，SVM的参数α数目和样本数目相同，从定义看来，因为参数数目和样本规模相当，所以属于无参数模型。当然，SVM通过得到支撑向量的方式，只有若干样本的参数α不为0，从这个角度，SVM还属于“稀疏模型”，这又属于另外一码事了。）
4、神经网络

非参数机器学习算法的优势有：
1、可变性：可以拟合许多不同的函数形式。
2、模型强大：对于目标函数不做假设或者作出很小的假设。
3、表现良好：对于训练样本数据具有良好的拟合性。

非参数机器学习算法的局限性：
1、需要更多数据：对于拟合目标函数需要更多的训练数据。
2、速度慢：因为需要训练很多的参数，所以训练过程通常比较慢。
3、过拟合：有较高的风险发生过拟合，对于预测的效果解释性不高。

	非参	有参
区分	非参数模型的参数取决于数据	模型参数独立于数据
优点	灵活，强大	简单，易理解，训练快，不需要大量数据进行训练
缺点	训练慢，容易过拟合，需要足够数据去训练	限定了模型的结构和复杂度，可能欠拟合
分类方法	KNN，决策树，支持向量机	Logistic回归，简单的神经网络

决策树的特征

1. 建立分类模型的非参数方法。不要对映射函数的形式做任何假设。

2. 基于启发式的方法来指导在巨大的假设空间中的搜索。
3. 计算成本低，适合大型数据集。一旦建立了决策树，对测试记录进行分类的速度非常快，最坏情况下的复杂度为O(w)，w是树的最大深度。
4. 容易解释，尤其是小的。
5. 为学习离散值函数提供一个表达性的表示。
6. 对存在的噪声具有相当的鲁棒性(带有修剪)。
7. 对冗余属性具有鲁棒性。

   如果一个属性与数据中的另一个属性强相关，那么这个属性就是冗余的。一旦选择了另一个，两个冗余属性中的一个将不会用于分裂。
8. 对不相关属性(即对分类任务无用的属性)的鲁棒性不强。

   在树木生长过程中，可能会不小心选择不相关的属性，从而导致过拟合。特征选择技术可以通过在预处理过程中消除无关属性来提供帮助。
9. 数据碎片化(记住高分支属性)•递归分区方法→当我们沿着树向下遍历时，记录的数量变得更小。

   叶节点的记录数量可能太小，无法对类做出统计上显著的决定。

   解决方案:当记录数量低于某个阈值时，不允许进一步分割。
10. 属性选择的不同标准很少会产生很大的差异。
11. 不同的剪枝方法主要改变剪枝树的大小。
12. 子树复制

    在一个决策树中，一个子树可以被复制多次。

    由于决策树在每个内部节点上响应单个属性测试条件。

    相同的测试条件可以应用于属性空间的不同部分(由于分治策略)。
13. 分类边界

    决策树将属性空间划分为不相交的区域，直到每个区域包含同一类的记录。

    两个相邻的不同类别区域之间的边界称为决策边界。

    单变量分裂的决策边界是直线的;即平行于“坐标轴”。

    单变量分割

    限制了决策树对连续属性之间复杂关系建模的表达能力。

    多元分割(多属性分割)

    多个属性可参与单节点分割。

    寻找最佳多变量标准比单变量分割更复杂。

    虽然多元分割可以显著提高树的性能，但不如单一分割受欢迎。

    主要基于输入属性的线性组合。

    C4.5可构建单变量决策树，CART可构建多变量决策树