决策树是一种基本的分类与回归方法，在分类问题中，表示基于特征对实例进行分类的过程。它可以认为是if-then规则的集合，也可以认为是定义在特征空间与类空间上的条件概率分布。相比朴素贝叶斯分类，决策树的优势在于构造过程不需要任何领域知识或参数设置，因此在实际应用中，对于探测式的知识发现，决策树更加适用。

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。

缺点：可能会产生过度匹配问题

适用数据类型：数值型和标称型数据

ps:（标称型：一般在有限的数据中取，而且只存在‘是’和‘否’两种不同的结果（一般用于分类）

数值型：可以在无限的数据中取，而且数值比较具体化，例如4.02,6.23这种值（一般用于回归分析））

决策树关键：选择最优划分属性

1. ID3（基于信息增益选择特征）

计算：

“信息熵”(information entropy)是度量样本集合纯度最常用的一种指标. 假定当前样本集合 D 中第 k 类样本所占的比例为 p_k $(k=1,2,\ldots,|\mathcal{Y}|)$, 则 D 的信息熵定义为

$$\mathrm{Ent}(D) = -\sum_{k=1}^{|\mathcal{Y}|} p_k \log_2 p_k . \tag{4.1}$$

$\mathrm{Ent}(D)$ 的值越小, 则 D 的纯度越高.

假定离散属性 a 有 V 个可能的取值 $\{a^1,a^2,\ldots,a^V\}$, 若使用 a 来对样本集 D 进行划分, 则会产生 V 个分支结点, 其中第 v 个分支结点包含了 D 中所有在属性 a 上取值为 a^v 的样本, 记为 D^v. 我们可根据式(4.1) 计算出 D^v 的信息熵, 再考虑到不同的分支结点所包含的样本数不同, 给分支结点赋予权重 $|D^v|/|D|$, 即样本数越多的分支结点的影响越大, 于是可计算出用属性 a 对样本集 D 进行划分所获得的“信息增益”(information gain)

$$\mathrm{Gain}(D,a) = \mathrm{Ent}(D) - \sum_{v=1}^{V} \frac{|D^v|}{|D|} \mathrm{Ent}(D^v) . \tag{4.2}$$

特点：ID3算法可用于划分标称型数据集，没有剪枝的过程，为了去除过度数据匹配的问题，可通过裁剪合并相邻的无法产生大量信息增益的叶子节点（例

如设置信息增益阀值）。使用信息增益的话其实是有一个**缺点**，那就是它偏向于具有大量值的属性。就是说在训练集中，某个属性所取的不同值的个数越多，那么越有可能拿它来作为分裂属性，而这样做有时候是没有意义的，另外ID3不能处理连续分布的数据特征，于是就有了C4.5算法。

2.C4.5算法（基于增益率选择特征）

$$\mathrm{Gain\_ratio}(D,a)=\frac{\mathrm{Gain}(D,a)}{\mathrm{IV}(a)}\ , \tag{4.3}$$

其中

$$\mathrm{IV}(a)=-\sum_{v=1}^{V}\frac{|D^v|}{|D|}\log_2\frac{|D^v|}{|D|} \tag{4.4}$$

特点：

以下几方面对ID3算法进行了改进：

- 克服了用信息增益选择属性时偏向选择取值多的属性的不足;
- 在树构造过程中进行剪枝;
- 能够完成对连续属性的离散化处理;
- 能够对不完整数据进行处理。

C4.5算法产生的分类规则易于理解、准确率较高；但效率低，因树构造过程中，需要对数据集进行多次的顺序扫描和排序。也是因为必须多次数据集扫描，C4.5只适合于能够驻留于内存的数据集。在实现过程中，C4.5算法在结构与递归上与ID3完全相同，区别只在于选取决决策特征时的决策依据不同，二者都有贪心性质：即通过局部最优构造全局最优。

3.CART（基于基尼指数）

CART二分每个特征（包括标签特征以及连续特征），经过最优二分特征及其最优二分特征值的选择、切分，二叉树生成，剪枝来实现CART算法。对于回归CART树选择误差平方和准则、对于分类CART树选择基尼系数准则进行特征选择，并递归调用构建二叉树过程生成CART树。

4.剪枝

通过降低决策树的复杂度来避免过拟合的过程称为剪枝。

预剪枝：在决策树生成过程中，对每个结点在划分前先进行估计，若当前结点的划分不能带来决策树泛化性能提升，则停止划分并将当前结点标记为叶节点。

后剪枝：先从训练集生成一棵完整的决策树，然后自底向上地对非叶节点进行考察，若将该结点对应的子树替换为叶节点能提升泛化性能，将该子树替换为叶

节点。

预剪枝优点：降低过拟合风险，显著减少了训练时间和测试时间

缺点：基于“贪心”本质禁止这些分支展开，带来欠拟合的风险

后剪枝优点：欠拟合风险小，泛化性能优于预剪枝决策树

缺点：训练时间开销大

本节介绍一种简单的决策树学习的剪枝算法.

决策树的剪枝往往通过极小化决策树整体的损失函数（loss function）或代价函数（cost function）来实现. 设树 T 的叶结点个数为 $|T|$，t 是树 T 的叶结点，该

叶结点有 N_t 个样本点，其中 k 类的样本点有 N_{tk} 个，$k=1,2,\cdots,K$，$H_t(T)$ 为叶结点 t 上的经验熵，$\alpha \geqslant 0$ 为参数，则决策树学习的损失函数可以定义为

$$C_\alpha(T)=\sum_{t=1}^{|T|}N_t H_t(T)+\alpha|T| \tag{5.11}$$

其中经验熵为

$$H_t(T)=-\sum_k \frac{N_{tk}}{N_t}\log\frac{N_{tk}}{N_t} \tag{5.12}$$

在损失函数中，将式 (5.11) 右端的第 1 项记作

$$C(T)=\sum_{t=1}^{|T|}N_t H_t(T)=-\sum_{t=1}^{|T|}\sum_{k=1}^{K}N_{tk}\log\frac{N_{tk}}{N_t} \tag{5.13}$$

这时有

$$C_\alpha(T)=C(T)+\alpha|T| \tag{5.14}$$

式(5.14)中，$C(T)$ 表示模型对训练数据的预测误差，即模型与训练数据的拟合程度，$|T|$ 表示模型复杂度，参数 $\alpha \geqslant 0$ 控制两者之间的影响. 较大的 α 促使选择较简单的模型（树），较小的 α 促使选择较复杂的模型（树）. $\alpha=0$ 意味着只考虑模型与训练数据的拟合程度，不考虑模型的复杂度.

剪枝，就是当 α 确定时，选择损失函数最小的模型，即损失函数最小的子树. 当 α 值确定时，子树越大，往往与训练数据的拟合越好，但是模型的复杂度就越高；相反，子树越小，模型的复杂度就越低，但是往往与训练数据的拟合不好. 损失函数正好表示了对两者的平衡.

相关实现代码参考：https://blog.csdn.net/LY_ysys629/article/details/72809129

https://blog.csdn.net/HerosOfEarth/article/details/52347820