ML实验报告：决策树

霍超凡

2019 年 11 月 17 日

1 实验内容

1. 实现ID3决策树，在给定的数据集上进行5折交叉验证。
2. 观测所得到的决策树在训练集和测试集的准确率，从而判断该决策树是否存在过拟合。
3. 实现预剪枝和后剪枝，并比较预剪枝和后剪枝在训练集和测试集上的准确率。

2 实验要求

完成上述实验的编码，并把实验流程，算法思想和在给定数据集上得到的指标记录到实验报告里。向助教老师演示所实现代码，并解释核心代码的思想。

3 实验环境

Jupyter Notebook + Python3

4 实验原理

决策树作为经典的机器学习算法，有很长历史，自Quinlan,J.R.提出ID3决策树后，决策树算法被后人不断修正，如今已经形成一套完整的理论框架。本实验从初始的决策树算法出发，对比了基于不同准则构造树的算法，重现决策树过拟合现象，并使用剪枝策略缓解过拟合，最后通过图形化展示决策树的构建过程阐述决策树背后的工作机理。

决策树适用于分类问题，给定样本集$D = \{\mathbf{x_1}, \mathbf{x_2}, \cdots, \mathbf{x}_N\}$，对任意$\mathbf{x_i} \in X$，$\mathbf{x_i}$包含$M$个属性，所有的属性构成属性集合$A = \{a_1, a_2, \cdots, a_M\}$，这些属性将成为我们分类样本的依据。每一个样本对应一个类别c_k，类别的所有可能取值构成集合$C = \{c_1, c_2, \cdots, c_K\}$。决策树算法构建一个分类树对样本分类，这棵树存在以下特点：

- 树的内部节点包含分类依据，一个内部节点只能对一个属性进行约束。
- 树的叶子节点包含类别标签，相同的类别标签可以位于不同叶子节点。

构造树的关键在于如何选择属性，根据不同的准则形成了ID3算法、C4.5算法和CART算法。

ID3算法-基于最小信息熵 信息熵出自于信息论，用来衡量信息的不确定性，将其应用到决策树中，用来衡量样本的纯度，样本集D的信息熵为

$$H(D) = -\sum_{k=1}^{K} p_k \log p_k \tag{1}$$

式中，p_k为样本中类别为c_k的概率，在实际计算中使用频率估计概率，记C_k为样本集中类别为c_k样本所构成的集合，则类别c_i在总样本的占比为$p_i = \frac{|C_k|}{|D|}$，将其代入式(1)，得到

$$H(D) = -\sum_{k=1}^{K} \frac{|C_k|}{|D|} \log \frac{|C_k|}{|D|} \tag{2}$$

函数$H(D)$在$[0,1]$上是开口向下的近似抛物曲线，在$p = 0.5$处取到最大值，p从0.5 向0或1变化时，$H(D)$逐渐减小，$H(D)$在$p = 0$或$p = 1$减小到0。当某一类别的样本在样本集的占比较大，信息熵会很小，但是需要注意的是，当样本集类别较多，即使每一个类别占比都不太大，信息熵也会很小。ID3算法期望当选取属性a作为分支属性后，样本集的纯度会提高，即被划分后的样本集的信息熵会减少，使用属性a被划分后的数据集的信息熵为

$$H(D|a) = \sum_{i} \frac{|D_i|}{|D|} H(D_i) \tag{3}$$

式中D_i为D经过属性a划分之后得到的子集。

C4.5算法-基于最大信息增益率 本实验所使用的数据集中样本属性的取值为连续变量，构建的树是典型的二叉树，不用考虑优先选取分枝多的属性所带来的问题。另外，我对C4.5存在一些疑惑，所以这个算法并没有在本实验实现。

CART算法-基于最小Gini系数 CART算法使用Gini系数来衡量数据集的纯度。样本集D上的Gini系数通过下式计算，

$$Gini(D) = 1 - \sum_{k=1}^{K} p_k^2 = 1 - \sum_{k=1}^{K} \left(\frac{|C_k|}{|D|} \right)^2 \tag{4}$$

这个函数和ID3算法所使用的信息熵的函数性质类似，当样本中出现具有较大占比的类别或类别较多且各个类别的占比普遍小时，Gini系数都会很小。

基于最小错误率 另外一种算法是基于最小错误率，定义样本集D上的错误率为将样本D均视为同一种类别的错误率，

$$Error(D) = 1 - \max \left\{ \frac{|C_k|}{D} \right\},\ k = 1, 2, \cdots, K \tag{5}$$

当样本具有较大占比的类别时，错误率较低，与信息熵和Gini系数不同的是，当样本集中的类别较多并且每个类别的占比均比较小时，错误率会很高。这意味着当使用最小错误率作为我们构建树的准则时，会优先将较大占比的类别划分出去。

为了确定最优属性，需要尝试使用不同属性去划分样本集，计算不同划分带来的增益，选取能够带来最大增益的属性作为下一步的分支属性。将样本集D使用数据a进行划分后得到的增益为

$$Gain(D|a) = Criterion(D) - Criterion(D|a) \tag{6}$$

式中$Criterion$是我们选择的衡量集合纯度的函数，它可以使用信息熵函数H、Gini系数$Gini$或错误率函数$Error$。最优属性

$$a^* = \arg\max_{a \in A} Gain(D|a) \tag{7}$$

最后给出决策树的算法，

1. 确定样本集D具有最优划分的属性a。
2. 使用属性a将样本划分为子集$D_1, D_2, \cdots, D_k$。
3. 对划分后的子集重复上述过程，直到满足终止条件。

终止条件：集合内所有的样本均具有相同的类别标签。

5 编程细节

连续变量的处理 本次实验的数据内样本的属性均为连续变量，需要计算出划分的阈值，然后将样本的属性取值大于阈值的样本划分为一个集合，小于或大于阈值的样本划分到另外一个集合。关于阈值的确定，枚举法是一种简单而又容易理解的方法，计算所有可能的阈值对划分的影响，然后挑选出最优阈值，本实验所使用的数据集中属性取值区间很不大，可以很快确定出最优阈值，但是对于大一点的数据集，这种方法不一定是很好的方法。我使用优化搜索的方法快速确定最优阈值，这种方法认为阈值太大、阈值设定太小将样本大部分都分隔到同一个分支效果会不好，最优阈值应该介于属性取值的最大值和最小值之间，所以使用黄金分割搜索方法可以快速确定出最优阈值。这个方法要求寻找最优阈值的被优化的函数必须是凸函数，关于具体的、详细的证明没有继续进行下去，直观感觉上它并不是凸函数，除非选取的准则函数很特殊。

决策树构建时所使用的数据结构 Python中并没有提供指针机制，而是将指针封装在列表和字典中，使用字典最为构建树所使用的数据结构。字典的键是分支依据，字典的值是分支得到子树。

算法细节 由于树具有递归的结构特性，构建树的算法也具有递归函数的形式。构建树和预测时搜索树均使用了递归式编程结构，其具体算法如下：

- 第一步：观察所给数据集内的样本纯度，若个样本均为同一个类别标签，则返回一个带有这个类别标签的叶子节点；否则继续。
- 第二步：计算出每一个属性的分割阈值和对应最优阈值的增益值，从而确定最优划分属性。
- 第三步：根据最优属性的最优阈值，样本集中大于阈值和小于或等于阈值的样本划分成两个互不相交的子集，转第一步。
- 第四步：将返回得到节点按照分支属性和阈值构建内部节点形成一棵树，返回这棵树。

剪枝 关于预剪枝算法，在构建树的算法基础上加上一个判断条件，即在确定划分最优属性和最优划分后，判断是否需要对划分后的子集继续运行上述算法。判断条件按照之前已经构造的树在验证集上的精度来确定。关于后剪枝算法，需要编写额外的函数，当树构建完成后，对树运行后剪枝算法，我在使用的过程中使用后序遍历算法进行剪枝，具体算法如下：

- 第一步：将验证集按照树的根节点的分支条件，划分成两个互不相交的子集和子分支树。
- 第二步：判断子分支树是否为叶子节点，如果不是叶子节点，对各个子分支树在经过划分的验证集上重复运行该算法（注意：子分支树计算完成后需要返回并进行第三步）。
- 第三步：对比当前树在没有分支和经过分支后在验证集的性能，如果不分枝更好则返回该验证集具有最大占比的类别标签，否则继续。
- 第四步：将子分支树出现组装成树并返回这棵树。

6 实验结果

搜索最优阈值 在确定最优分隔阈值时，我发现被优化的函数在不同的分隔阈值上并不严格是凸函数，存在一些局部最小值。我设定精度为0.01采用枚举的方法测得不同阈值下的被划分后得到的样本集的信息熵，得到的结果如图1所示。我们发现除了一些个别例子，我们的算法具有较大的概率能够找到最优分割点，这得益于我们假设和数据的分布一致。

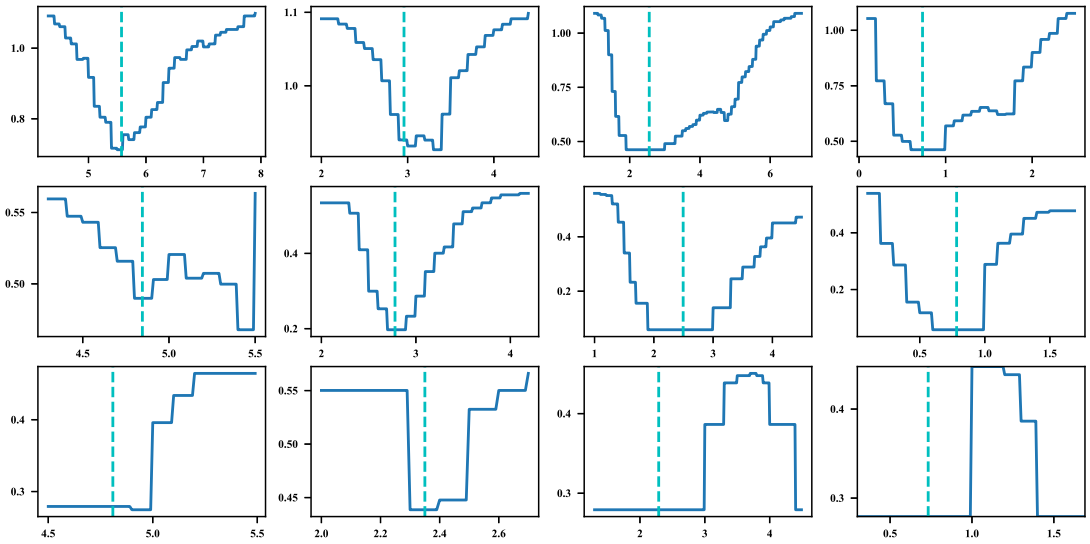

图 1: 使用不同分隔阈值观察不同阈值下样本集内的信息熵，横坐标是阈值，纵坐标为信息熵，每一列对应着不同属性，第一列为第一个属性的阈值和信息熵的关系，第二列为第二个属性，余类推。每一行对应经过不同划分之后得到的样本集，第一行为原始数据集下的不同属性阈值和信息熵的关系，第二行对应经过一次划分之后得到的样本集，余类推。注意到越往下，被划分的数据集内样本个数减少，所呈现的曲线的平滑度越差。竖直天蓝色曲线表示使用黄金分隔算法所求的的最优阈值。

过拟合 为了观察到过拟合现象，我构建不同深度的决策树，并测试它在测试集个训练集上的精度，得到图2，这种过拟合现象并不是频繁发生，一般情况下，如果随机采样，验证集和测试集分布一致，不容易出现过拟合现象，这张图我尝试了一些随机采样才得到。

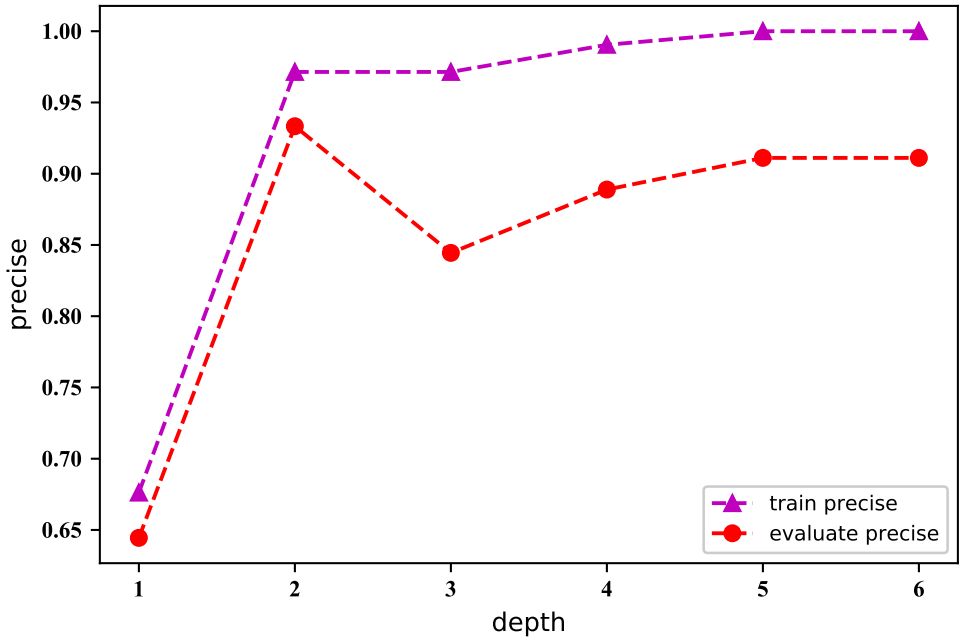

图 2: 该图测试集和验证集的精度随不同深度变化情况，将数据集按照7：3分成训练集和验证集，当深度增加时，决策树在训练集上的精度不断增加，而在验证集的精度减少，出现过拟合现象。

剪枝策略 使用4折交叉验证，测试经过剪枝和不剪枝的决策树的性能，得到的结果如表1第二行所示，没有经过剪枝的决策树的性能略好。表2中的Precise 2没有使用交叉验证，使用相同的训练集和相同的测试集对这三棵决策树进行测试，由于随机采样的缘故，得到的结果存在随机性，结果并不能说明一些问题。

表 1: 剪枝与不剪枝的决策树的性能对比

Pruning	Pre-pruning	Post-pruning	No-pruning
Precise 1	0.956	0.945	**0.972**
Precise 2	0.911	**0.977**	0.911

不同算法 分别使用最小信息熵、最小错误率、最小Gini系数策略构建决策树，并进行5折交叉验证，得到图2结果。仍然将数据集按照7：3划分成训练集和验证集，测试这三种决策树的性能，通过控制决策树的深度，得到不同算法所构建的决策树性能随深度的变化曲线，如图3所示，使用最小错误率和最小Gini 系数所构建的的决策树具有完全相同的性能，而且这两者的过拟合现象没有使用最小Entropy 所构建决策树那么严重，使用最小信息熵所构建的决策树的性能要优于另外两者。

表 2: 不同算法构建的决策树性能对比

Criterion	Entropy	Error	Gini
Precise	**0.96**	0.92	0.92

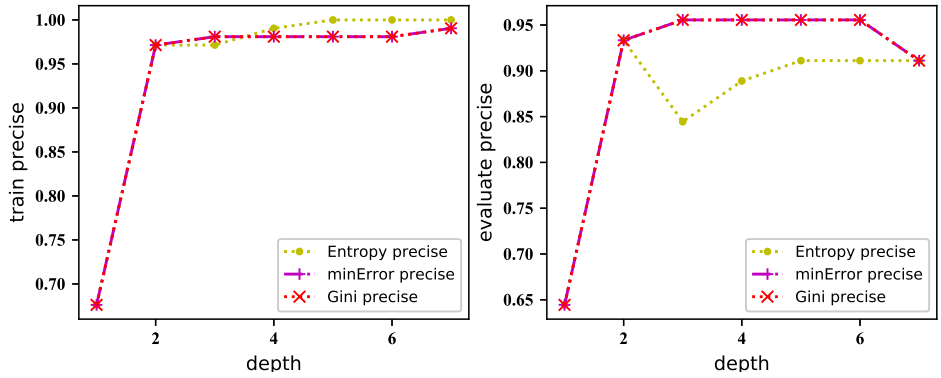

图 3: 不同策略构建的决策树性能对比，左图训练精度对比，右图验证精度对比。

决策树构建过程 我将决策树构建树的过程中样本被划分的情况绘制成散点图 4，图4前两行是未进行划分的原始数据的样本分布情况，4中属性两两组合后得到6种情况，在原始数据运行决策树算法找到最优属性和最优划分，图中黄线显示了最优阈值。将数据集使用刚刚得到的最优阈值划分成两个数据集，在其中的一个子集中重复上述步骤得到图4的中间两行。最后两行的数据是经过两次划分所得到的，该图具体展示了决策树的构建过程。

7 实验结论

- 由于数据的随机划分，训练集和测试集的数据分布存在差异，如果不加控制地构建决策树会出现过拟合现象，而使用剪枝策略可以缓解这种情况。
- 使用不同准则选取最优划分属性所构建的决策树性能相差甚微，也许是数据集的缘故，基于最小错误率和最小Gini系数所构建的决策树具有完全相同的性能，这也许是数据集的缘故。
- 对于连续变量最优阈值的确定，如果变量的分布符合我们的假设，使用黄金分割优化算法可以加快我们的搜索速度。
- 从决策树构建的过程可以看出，决策树适合于处理不同类别属性值具有较大区分度的样本。

8 实验中遇到的问题与解决方案

在编程中遇到一些问题。

第一，黄金优化算法未必能够找到最优分隔。在实际运行过程中出现函数循环递归调用始终不返回结果的情况，例如图5所示，最优属性应该选取Attr.3，但是实际上算法运

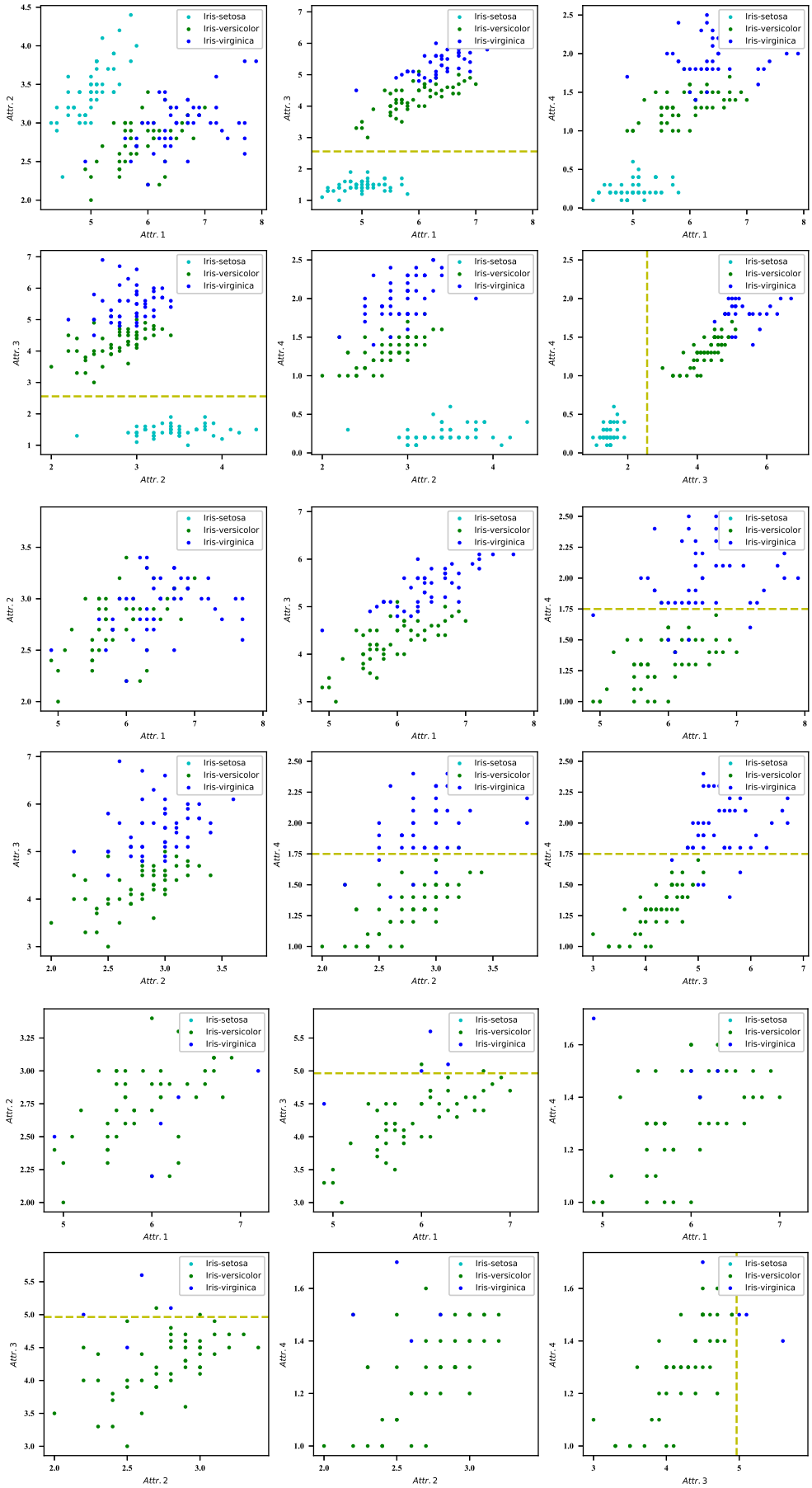

图 4: 该图展示了决策树构建过程中样本被划分的结果。

	Attr.1	Attr.2	Attr.3	Attr.4	label
72	6.3	2.5	4.9	1.5	Iris-versicolor
133	6.3	2.8	5.1	1.5	Iris-virginica
56	6.3	3.3	4.7	1.6	Iris-versicolor
87	6.3	2.3	4.4	1.3	Iris-versicolor

图 5: 一个错误的实例

行的结果为Attr.1，结果导致这4行数据被划分到同一个子分支中，在相同的数据上继续递归调用算法，最终陷入死循环，至于为什么会选取Attr.1作为最优属性而不选取Attr.3 作为最优属性，这是由于黄金分割搜索算法导致的。该算法在属性Attr.1上找到的最优分割点为6.3，计算的错误率为0.25，而在属性Attr.3所找到的最优分隔阈值为4.75，错误率同样为0.25，但是算法先遍历属性Attr.1，所以最终将它作为最优属性。实际上的在属性Attr.4上最优分割点应该在5.0左右，但是实际计算的分割点为4.75，黄金分割算法的初始点λ和μ在区间中间取值，而且由于样本过少，只有4个，在4.7 和4.9 之间任何取值得到的错误率均相同，算法认为自己达到最优分割点，结果导致出现错误。解决方案是，初始情况下样本数比较多可以使用黄金分割加快搜索，在决策树的构建后期使用枚举法，我在实际的编程中没有使用这种复杂的解决方案，而在构建决策树时加上一个限制条件，如果某一个属性内的取同一个值（比如图5中的Attr.1 属性）则停止继续分割。

第二，在进行剪枝时，由于验证集内的数据较少，能够进入决策树的深层的数据可能非常少，甚至没有，遇到这种情况，我进行了如下处理，我希望尽可能保证拟合训练集的精度，尽可能不进行剪枝，遇到验证集为空的情况后，剪枝后的决策树在验证集的精度设置为0。