ref:

1. 机器学习西瓜书 & pumpkin
2. Introduction to Machine Learning with Python

[决策树](#header-n8)  
 [信息熵](#header-n10)  
 [决策数算法一览](#header-n456)  
 [ID3决策树算法：(Quinlan,1986)](#header-n24)  
 [C4.5算法(Quinlan,1993)](#header-n33)  
 [CART决策树(Breiman 1984) —— Gini Index](#header-n89)  
 [基尼系数：反映了从一个集合里面，随机抽取两个样本，类别不一致的概率](#header-n119)  
 [剪枝处理 pruning](#header-n43)  
 [CART 剪枝](#header-n185)  
 [1. 剪枝, 形成一个子树序列](#header-n196)  
 [2. 在剪枝得到的子树序列 中通过交叉验证选取最优子树](#header-n220)   
 [连续值处理](#header-n51)  
 [多变量决策树](#header-n59)  
 [python](#header-n63)  
 [调用sklearn](#header-n64)  
 [sklearn 剪枝](#header-n69)  
 [预剪枝](#header-n132)  
 [min*samples*leaf](#header-n136)  
 [min*samples*split](#header-n138)  
 [max\_ features](#header-n149)  
 [min*impurity*decrease](#header-n151)  
 [举例：用学习曲线来确定超参 max\_depth](#header-n162)  
 [后剪枝](#header-n130)  
 [[1.10.8. Minimal Cost-Complexity Pruning](https://scikit-learn.org/stable/modules/tree.html#id2)](#header-n483)  
 [[post pruning decision trees with cost complexity pruning](https://scikit-learn.org/stable/auto_examples/tree/plot_cost_complexity_pruning.html) 基于CCP(cost complexity proning) 的后剪枝](#header-n486)  
 [Total impurity of leaves vs effective alphas of pruned tree](#header-n542)  
 [训练集和测试集，在alpha不同的时候的准确率](#header-n557)  
 [画图](#header-n70)

# 决策树

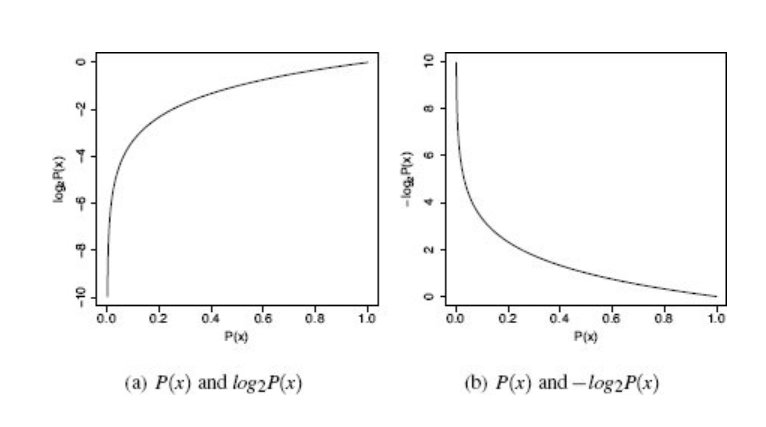
样本纯度 purity：一个集合中的元素尽可能属于同一类别。信息熵 information entropy 是衡量样本集合纯度最常用的一种指标。

## 信息熵

是第 类样本所占的比例。

当个体在 均匀分布，则信息熵越大，为

当所有个体为同一组，则信息熵最小，为0



Shannon 的信息熵模型就是 的一个加权回归

假设离散属性 有 个可能的取值 ，对应 个分支，每个分支 的集合为 。那么使用属性 进行分支的信息增益：

表示用 对集合 进行划分所获得的“纯度提升”（不确定性减少的程度）。

## 决策数算法一览

ref: sklean [1.10.6. 决策树算法: ID3, C4.5, C5.0 和 CART](https://www.sklearncn.cn/11/)

所有种类的决策树算法有哪些以及它们之间的区别？scikit-learn 中实现何种算法呢？

[**ID3**](https://en.wikipedia.org/wiki/ID3_algorithm)**（Iterative Dichotomiser 3）**由 Ross Quinlan 在1986年提出。该算法创建一个多路树，找到每个节点（即以贪心的方式）分类特征，这将产生分类目标的最大信息增益。决策树发展到其最大尺寸，然后通常利用剪枝来提高树对未知数据的泛华能力。

**C4.5 是 ID3 的后继者**，并且通过动态定义将连续属性值分割成一组离散间隔的离散属性（基于数字变量），消除了特征必须被明确分类的限制。C4.5 将训练的树（即，ID3算法的输出）转换成 if-then 规则的集合。然后评估每个规则的这些准确性，以确定应用它们的顺序。如果规则的准确性没有改变，则需要决策树的树枝来解决。

**C5.0** 是 Quinlan 根据专有许可证发布的最新版本。它使用更少的内存，并建立比 C4.5 更小的规则集，同时更准确。

[**CART**](https://en.wikipedia.org/wiki/Predictive_analytics#Classification_and_regression_trees_.28CART.29)**（Classification and Regression Trees （分类和回归树）**）与 C4.5 非常相似，但它不同之处在于它支持数值目标变量（回归），并且不计算规则集。CART 使用在每个节点产生最大信息增益的特征和阈值来构造二叉树。

scikit-learn 使用 CART 算法的优化版本。

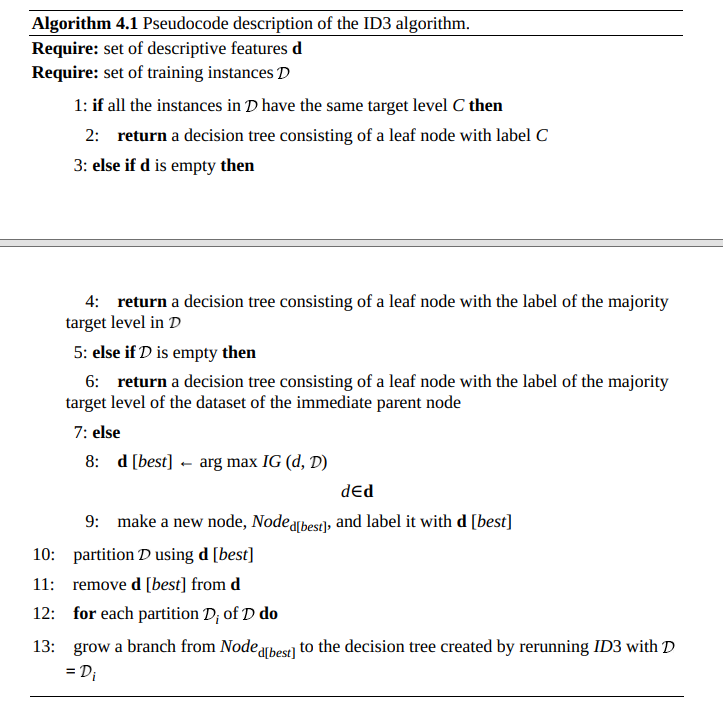
### ID3决策树算法：(Quinlan,1986)

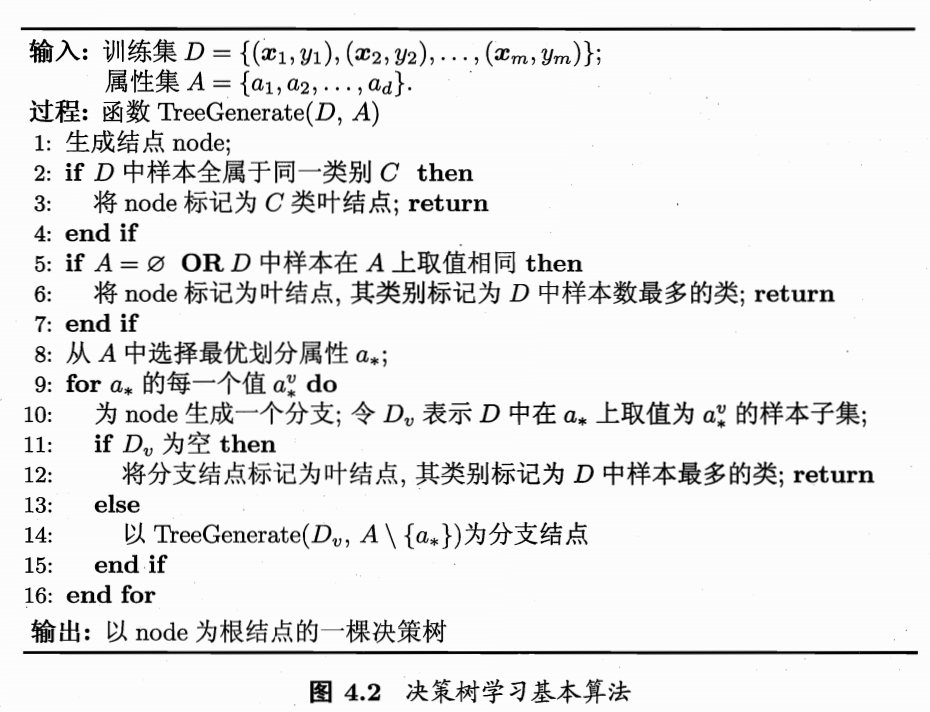
**选择信息增益最大的属性进行分支。**

**该算法的弊端：**对于可取值数目较多的属性有偏好。

# 该方法是基于递归、深度搜索树

注：level 指的是离散变量的不同取值，target level就是目标 y 的不同取值





### C4.5算法(Quinlan,1993)

**不直接使用信息增益，而是使用“增益率” gain ratio：**

IV：固有值，intristic value。属性 可能的取值数目越多，则 IV 越大。

这个函数中， 越小，函数值越大。

当 可能的取值数目越多, 越小，IV越大。IV大了之后，Gain\_ratio就变小。

**该算法的弊端：**对于可取值数目**较少**的属性有偏好。因此，要先选 Gain 大于平均水平的，再从中选择 Gain\_ratio 大的。

### CART决策树(Breiman 1984) —— Gini Index

CART: classification and regression tree

既可以用来分类，也可以用来回归

CART 假设决策树是二叉树，左边为是，右边为否。

**步骤：**

1. 生成决策树
2. 决策树剪枝

#### 基尼系数：反映了从一个集合里面，随机抽取两个样本，类别不一致的概率

那么属性 的基尼系数为：

## 剪枝处理 pruning

* **pre**pruning 预剪枝: 在生成的过程中，估计划分，然后标记为叶节点；
* 预剪枝基于贪心本质展开，有欠拟合的风险
* **post**pruning: 生成之后，再把子树替换为叶节点
* 后剪枝是用验证集来剪枝的

通过泛化性能来判断是否需要剪枝。要把训练集分为 **训练集training set** 和 **验证集 validation set**

### CART 剪枝

ref: 统计学习方法 —— 李航

Highlight:

* **当 大 的时候, 最优子树 偏小; 当 小的时候, 最优子树 偏大**
* **剪枝后得到 逐渐增大的嵌套子树； 和子树一一对应**

CART 剪枝算法从 “完全生长” 的决策树的底端剪去一些子树, 使决策树变小 (模 型变简单), 从而能够对末知数据有更准确的预测。CART 剪枝算法由两步组成: 首先 从生成算法产生的决策树 底端开始不断剪枝, 直到 的根结点, 形成一个子树序 列 ; 然后通过交叉验证法在独立的验证数据集上对子树序列进行测 试, 从中选择最优子树。

#### 1. 剪枝, 形成一个子树序列

在剪枝过程中, 计算**子树的损失函数:**

其中, 为任意子树, 为对训练数据的预测误差 (如基尼指数), 为子树的叶 结点个数, 为参数, 为参数是 时的子树 的整体损失。参数 权衡训 练数据的拟合程度与模型的复杂度。  
对固定的 , 一定存在使损失函数 最小的子树, 将其表示为 在损 失函数 最小的意义下是最优的。容易验证这样的最优子树是唯一的。**当 大 的时候, 最优子树 偏小; 当 小的时候, 最优子树 偏大**。极端情况, 当 时, 整体树是最优的。当 时, 根结点组成的单结点树是最优的。

Breiman 等人证明: 可以用递归的方法对树进行剪枝。将 从小增大, , 产生一系列的区间 ; 剪枝得到的子树序列对应着区间 的最优子树序列 , 序 列中的子树是嵌套的。  
具体地, 从整体树 开始前枝。对 的任意内部结点 , 以 为单结点（叶节点）树的损失 函数是

以 为根结点的子树 的损失函数是

当 及 充分小时, 有不等式（5.29）

当 增大时, 在某一 有

当 再增大时, 不等式 (5.29) 反向。只要 与 有相同的 损失函数值, 而 的结点少, 因此 比 更可取, 对 进行剪枝。  
为此, 对 中每一内部结点 , 计算

**它表示剪枝后整体损失函数减少的程度。**在 中前去 最小的 , 将得到的子树 作为 , 同时将最小的 设为 为区间 的最优子树。  
如此前枝下去, 直至得到根结点。在这一过程中, 不断地增加 的值, 产生新的 区间。

#### 2. 在剪枝得到的子树序列 中通过交叉验证选取最优子树

具体地, 利用独立的验证数据集, 测试子树序列 中各棵子树的平 方误差或基尼指数。平方误差或基尼指数最小的决策树被认为是最优的决策树。在子 树序列中, 每棵子树 都对应于一个参数 。所以, **当最优 子树 确定时, 对应的 也确定了, 即得到最优决策树 。**  
现在写出 CART 剪枝算法。

算法 (CART 剪枝算法)  
输入: CART 算法生成的决策树 ;  
输出: 最优决策树 。  
（1）设 。  
（2）设 。  
（3）自下而上地对各内部结点 计算 以及

这里, 表示以 为根结点的子树, 是对训练数据的预测误差, 是 的叶 结点个数。  
（4）对 的内部结点 进行前枝, 并对叶结点 以多数表决法决定其类, 得到树 。  
（5）设 。  
（6）如果 不是由根结点及两个叶结点构成的树, 则回到步骤 (2); 否则令 。  
（7）采用交叉验证法在子树序列 中选取最优子树 。

## 连续值处理

假设连续值 出现了 n 种可能性，从小到大排序为 ，如果取 的中点作为划分点，则总共可能有 n-1 个划分点：

基于某一个划分点二分之后的信息增益为：

可以选择使得 最大的划分点。

连续属性可以在后续继续作为划分属性。

## 多变量决策树

使用决策变量的线性组合 进行分类

和 可以学习得到。

## python

### 调用sklearn

ref: https://scikit-learn.org/stable/modules/tree.html

### sklearn 剪枝

#### 预剪枝

##### min*samples*leaf

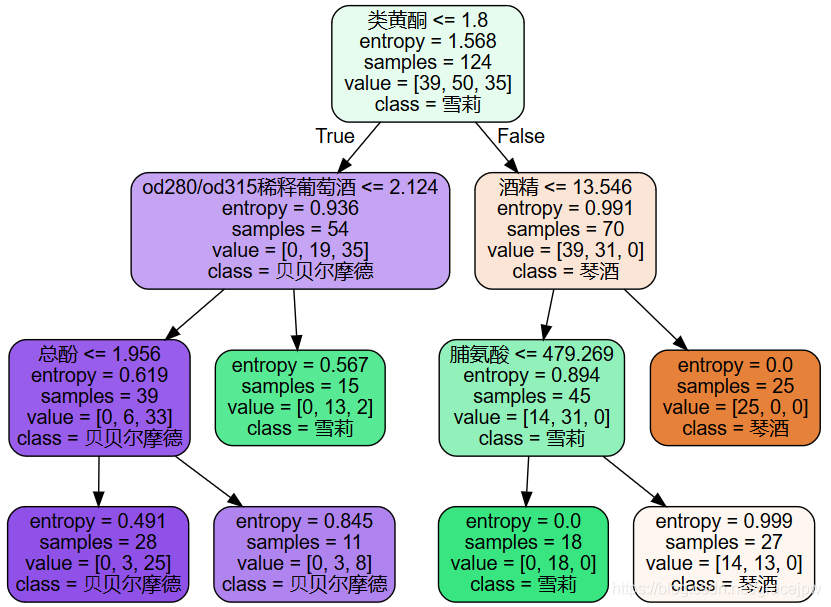
限定一个结点在分支后的每个子结点都必须包含至少min*samples* leaf个训练样本，否则分支就不会发生，或者，分支会朝着满足每个子结点都包含min\\_samples\\_ leaf个样本的方向去发生。   
一般搭配max\_depth使用，**在回归树中有神奇的效果**，可以让模型变得更加平滑。这个参数的数量设置得太小会引起过拟合，设置得太大就会阻止模型学习数据⼀一般来说，建议从=5开始使用。如果叶结点中含有的样本量变化很 大，建议输入浮点数作为样本量的百分比来使用。同时，这个参数可以保证每个叶子的最小尺寸，可以在回归问题中避免低方差，过拟合的叶子结点出现。**对于类别不多的分类问题，=1通常就是最佳选择。**

##### min*samples*split

限定一个结点必须要包含至少min*samples*split个训练样本，这个结点才允许被分支，否则分支就不会发生。

clf = tree.DecisionTreeClassifier(criterion="entropy",
  
 random\_state=30,splitter="random"
  
 ,max\_depth=3
  
 ,min\_samples\_leaf=10
  
 ,min\_samples\_split=10
  
 )
  
clf = clf.fit(Xtrain, Ytrain)
  
dot\_data = tree.export\_graphviz(clf
  
 ,feature\_names= feature\_name
  
 ,class\_names=["琴酒","雪莉","⻉贝尔摩德"]
  
 ,filled=True
  
 ,rounded=True
  
 )
  
graph = graphviz.Source(dot\_data)
  
graph

运行结果



clf.score(Xtrain,Ytrain)

运行结果

0.8306451612903226

clf.score(Xtest,Ytest)

运行结果

0.8518518518518519

##### max\_ features

一般max*depth用作树的”精修“。*   
*max* features限制分支时考虑的特征个数，超过限制个数的特征都会被舍弃。和max*depth异曲同工， max* features是用来限制高维度数据的过拟合的剪枝参数，但其方法比较暴力，是直接限制可以   
使用的特征数量而强行使决策树停下的参数，**在不知道决策树中的各个特征的重要性的情况下，强行设定这个参数可能会导致模型学习不足。如果希望通过降维的方式防止过拟合，建议使用PCA，ICA或者特征选择模块中的降维算法。**

##### min*impurity*decrease

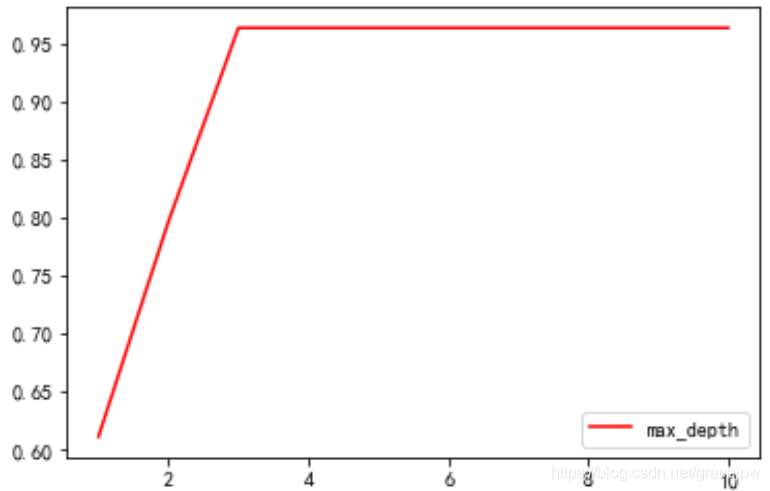
**min*impurity* decrease限制信息增益的大小，信息增益小于设定数值的分支不会发生**。这是在0.19版本中更新的功能，在0.19版本之前时使用min*impurity* split。

##### 举例：用学习曲线来确定超参 max\_depth

那么，具体如何来确定每个参数填写什么值呢？这时候，我们就要使用确定超参数的曲线来进行判断了，继续使用我们已经训练好的决策树模型clf。超参数的学习曲线，是一条以超参数的取值为横坐标，模型的度量指标为纵坐标的曲线， 是用来衡量不同超参数取值下模型的表现的线。在我们建好的决策树里，我们的模型度量指标就是score。

test = []  
for i in range(10):  
 clf = tree.DecisionTreeClassifier(max\_depth=i+1  
 ,criterion="entropy"  
 ,random\_state=30  
 ,splitter="random"  
 )  
 clf = clf.fit(Xtrain, Ytrain)  
 score = clf.score(Xtest, Ytest)  
 test.append(score)  
plt.plot(range(1,11),test,color="red",label="max\_depth")  
plt.legend()  
plt.show()

运行结果



可见，max\_depth=3是最优选择。

总之，无论如何，剪枝参数的默认值会让树无尽生长，这些树在某些数据集上可能非常大，对内存的消耗也非常大。 所以如果数据集非常大，已经预测到无论如何都是要剪枝的，那么，提前设定这些参数来控制树的复杂性和大小会比较好。

#### 后剪枝

references:

[1.10.8. Minimal Cost-Complexity Pruning](https://scikit-learn.org/stable/modules/tree.html#id2)  
[post pruning decision trees with cost complexity pruning](https://scikit-learn.org/stable/auto_examples/tree/plot_cost_complexity_pruning.html)

##### [1.10.8. Minimal Cost-Complexity Pruning](https://scikit-learn.org/stable/modules/tree.html#id2)

reference: Chapter 3 of Breiman 1984

* ：**the complexity parameter**
* **cost-complexity measure**（也叫**子树的损失函数**）, of a given tree T:

是树 T 中叶节点的个数。

原本为 misclassification rate of the terminal nodes，在sklearn中使用 weighted impurity of the terminal nodes。

sklearn说inpurity有多种衡量方法，可能对于classification和regression各有不同，如MSE等

* **Minimal cost-complexity pruning** finds the subtree of T that minimizes .
* 一个单一 node （该node作为叶子节点）的cost complexity measure 是 。 该节点下的子树记为 ，一般而言，。在 取特定值的时候，，使之相等的 记为 ，。 最小的子树会被剪枝。
* **当** > 参数 ccp\_alpha 时，停止。

##### [post pruning decision trees with cost complexity pruning](https://scikit-learn.org/stable/auto_examples/tree/plot_cost_complexity_pruning.html) 基于CCP(cost complexity proning) 的后剪枝

cost complexity parameter ： ccp\_alpha 。该参数越大，被剪掉的树越大。

import matplotlib.pyplot as plt
  
from sklearn.model\_selection import train\_test\_split
  
from sklearn.datasets import load\_breast\_cancer
  
from sklearn.tree import DecisionTreeClassifier

###### Total impurity of leaves vs effective alphas of pruned tree

[DecisionTreeClassifier.cost\_complexity\_pruning\_path](https://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeClassifier.html#sklearn.tree.DecisionTreeClassifier.cost_complexity_pruning_path)

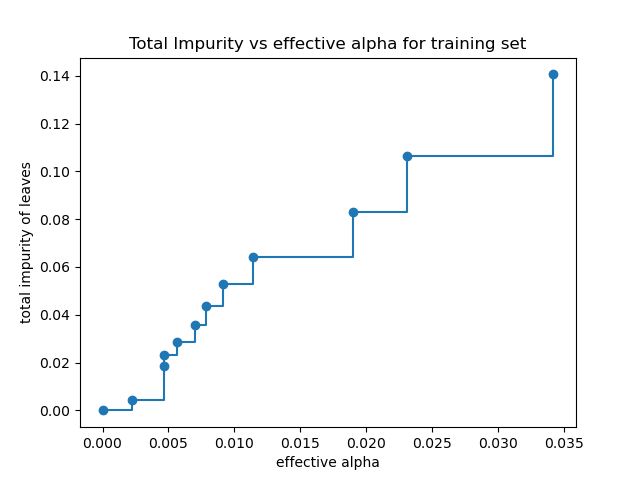
**cost*complexity*pruning\_path**(*X*, *y*, *sample\_weight=None*)[[source]](https://github.com/scikit-learn/scikit-learn/blob/baf0ea25d/sklearn/tree/_classes.py#L607)

该函数返回所有的effective ，以及所有 leaf （叶子节点）的impurity 之和

X, y = load\_breast\_cancer(return\_X\_y=True)  
X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, random\_state=0)  
  
clf = DecisionTreeClassifier(random\_state=0)  
path = clf.cost\_complexity\_pruning\_path(X\_train, y\_train)  
ccp\_alphas, impurities = path.ccp\_alphas, path.impurities

注：下图中去掉了最大的effective ，即只有一个根节点的树（ccp\_alphas[:-1], impurities[:-1]）

fig, ax = plt.subplots()
  
ax.plot(ccp\_alphas[:-1], impurities[:-1], marker="o", drawstyle="steps-post")
  
ax.set\_xlabel("effective alpha")
  
ax.set\_ylabel("total impurity of leaves")
  
ax.set\_title("Total Impurity vs effective alpha for training set")



Out:

Text(0.5, 1.0, 'Total Impurity vs effective alpha for training set')

如果在 DecisionTreeClassifier 训练的时候，加上 ccp\_alpha 参数，那么可以训练出来一组决策树

clfs[-1] ：对应一个node

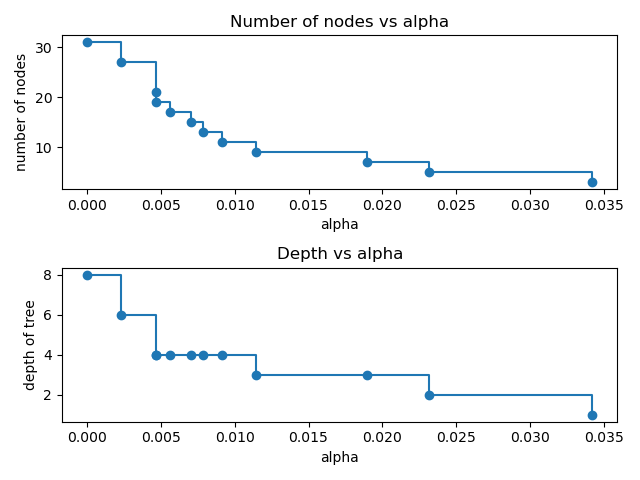
clfs = []
  
for ccp\_alpha in ccp\_alphas:
  
 clf = DecisionTreeClassifier(random\_state=0, ccp\_alpha=ccp\_alpha)
  
 clf.fit(X\_train, y\_train)
  
 clfs.append(clf)
  
print(
  
 "Number of nodes in the last tree is: {} with ccp\_alpha: {}".format(
  
 clfs[-1].tree\_.node\_count, ccp\_alphas[-1]
  
 )
  
)

Out:

Number of nodes in the last tree is: 1 with ccp\_alpha: 0.3272984419327777

剩余的决策树：当 ccp\_alpha 增加，node数减少，树的深度也减少

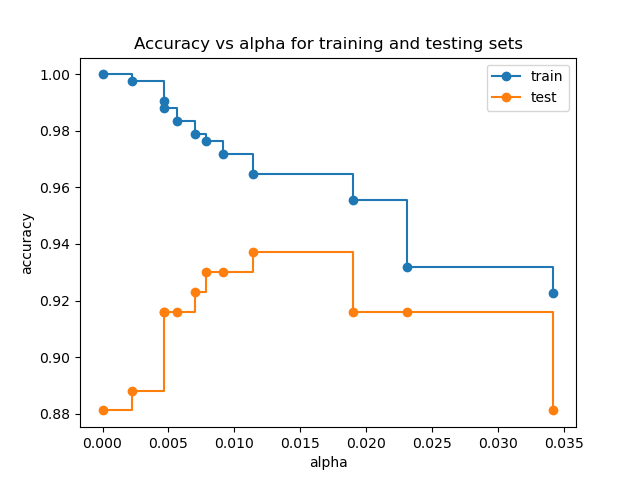
clfs = clfs[:-1]  
ccp\_alphas = ccp\_alphas[:-1]  
  
node\_counts = [clf.tree\_.node\_count for clf in clfs]  
depth = [clf.tree\_.max\_depth for clf in clfs]  
fig, ax = plt.subplots(2, 1)  
ax[0].plot(ccp\_alphas, node\_counts, marker="o", drawstyle="steps-post")  
ax[0].set\_xlabel("alpha")  
ax[0].set\_ylabel("number of nodes")  
ax[0].set\_title("Number of nodes vs alpha")  
ax[1].plot(ccp\_alphas, depth, marker="o", drawstyle="steps-post")  
ax[1].set\_xlabel("alpha")  
ax[1].set\_ylabel("depth of tree")  
ax[1].set\_title("Depth vs alpha")  
fig.tight\_layout()



###### 训练集和测试集，在alpha不同的时候的准确率

alpha不断增大，展示了由过拟合到欠拟合的过程。当ccp\_alpha=0.015 时，测试集准确率最高。

train\_scores = [clf.score(X\_train, y\_train) for clf in clfs]  
test\_scores = [clf.score(X\_test, y\_test) for clf in clfs]  
  
fig, ax = plt.subplots()  
ax.set\_xlabel("alpha")  
ax.set\_ylabel("accuracy")  
ax.set\_title("Accuracy vs alpha for training and testing sets")  
ax.plot(ccp\_alphas, train\_scores, marker="o", label="train", drawstyle="steps-post")  
ax.plot(ccp\_alphas, test\_scores, marker="o", label="test", drawstyle="steps-post")  
ax.legend()  
plt.show()



### 画图

import graphviz   
dot\_data = tree.export\_graphviz(clf, out\_file=None)   
graph = graphviz.Source(dot\_data)   
graph.render("pdfname")

graphviz需要

1. pip install graphviz
2. 去官网下载.exe安装，并且把dot.exe加入path