AI学习笔记--sklearn--DecisionTreeClassifier

决策树(Decision Tree）是在已知各种情况发生概率的基础上，通过构成决策树来求取净现值的期望值大于等于零的概率，评价项目风险，判断其可行性的决策分析方法，是直观运用概率分析的一种图解法。由于这种决策分支画成图形很像一棵树的枝干，故称决策树。在机器学习中，决策树是一个预测模型，他代表的是对象属性与对象值之间的一种映射关系。Entropy = 系统的凌乱程度，使用算法ID3, C4.5和C5.0生成树算法使用熵。这一度量是基于信息学理论中熵的概念。

决策树是一种树形结构，其中每个内部节点表示一个属性上的测试，每个分支代表一个测试输出，每个叶节点代表一种类别。

分类树（决策树）是一种十分常用的分类方法。他是一种监管学习，所谓监管学习就是给定一堆样本，每个样本都有一组属性和一个类别，这些类别是事先确定的，那么通过学习得到一个分类器，这个分类器能够对新出现的对象给出正确的分类。这样的机器学习就被称之为监督学习。

• 决策树组成

□——决策点，是对几种可能方案的选择，即最后选择的最佳方案。如果决策属于多级决策，则决策树的中间可以有多个决策点，以决策树根部的决策点为最终决策方案。[1]

○——状态节点，代表备选方案的经济效果（期望值），通过各状态节点的经济效果的对比，按照一定的决策标准就可以选出最佳方案。由状态节点引出的分支称为概率枝，概率枝的数目表示可能出现的自然状态数目每个分枝上要注明该状态出现的概率。[1]

△——结果节点，将每个方案在各种自然状态下取得的损益值标注于结果节点的右端。[1]

• 决策树算法

机器学习中，决策树是一个预测模型；他代表的是对象属性与对象值之间的一种映射关系。树中每个节点表示某个对象，而每个分叉路径则代表的某个可能的属性值，而每个叶结点则对应从根节点到该叶节点所经历的路径所表示的对象的值。决策树仅有单一输出，若欲有复数输出，可以建立独立的决策树以处理不同输出。数据挖掘中决策树是一种经常要用到的技术，可以用于分析数据，同样也可以用来作预测。

从数据产生决策树的机器学习技术叫做决策树学习, 通俗说就是决策树。

一个决策树包含三种类型的节点：

1. 决策节点：通常用矩形框来表示
2. 机会节点：通常用圆圈来表示
3. 终结点：通常用三角形来表示

决策树学习也是资料探勘中一个普通的方法。在这里，每个决策树都表述了一种树型结构，它由它的分支来对该类型的对象依靠属性进行分类。每个决策树可以依靠对源数据库的分割进行数据测试。这个过程可以递归式的对树进行修剪。 当不能再进行分割或一个单独的类可以被应用于某一分支时，递归过程就完成了。另外，随机森林分类器将许多决策树结合起来以提升分类的正确率。

决策树同时也可以依靠计算条件概率来构造。

决策树如果依靠数学的计算方法可以取得更加理想的效果。 数据库已如下所示：

(x, y) = (x1, x2, x3…, xk, y)

相关的变量 Y 表示我们尝试去理解，分类或者更一般化的结果。 其他的变量x1, x2, x3 等则是帮助我们达到目的的变量。

• 决策树例子

从一个分类例子说起：银行希望能够通过一个人的信息（包括职业、年龄、收入、学历）去判断他是否有贷款的意向，从而更有针对性地完成工作。下表是银行现在能够掌握的信息，我们的目标是通过对下面的数据进行分析建立一个预测用户贷款一下的模型。

表2.1 银行用户信息表

职业	年龄	收入	学历	是否贷款
自由职业	28	5000	高中	是
工人	36	5500	高中	否
工人	42	2800	初中	是
白领	45	3300	小学	是

白领	25	10000	本科	是
白领	32	8000	硕士	否
白领	28	13000	博士	是
自由职业	21	4000	本科	否
自由职业	22	3200	小学	否
工人	33	3000	高中	否
工人	48	4200	小学	否

（注：上表中的数据都由本人捏造，不具有任何实际的意义）

上边中有4个客户的属性，如何综合利用这些属性去判断用户的贷款意向？决策树的做法是每次选择一个属性进行判断，如果不能得出结论，继续选择其他属性进行判断，直到能够"肯定地"判断出用户的类型或者是上述属性都已经使用完毕。比如说我们要判断一个客户的贷款意向，我们可以先根据客户的职业进行判断，如果不能得出结论，再根据年龄作判断，这样以此类推，直到可以得出结论为止。

决策树用树结构实现上述的判断流程，如图2.1所示：

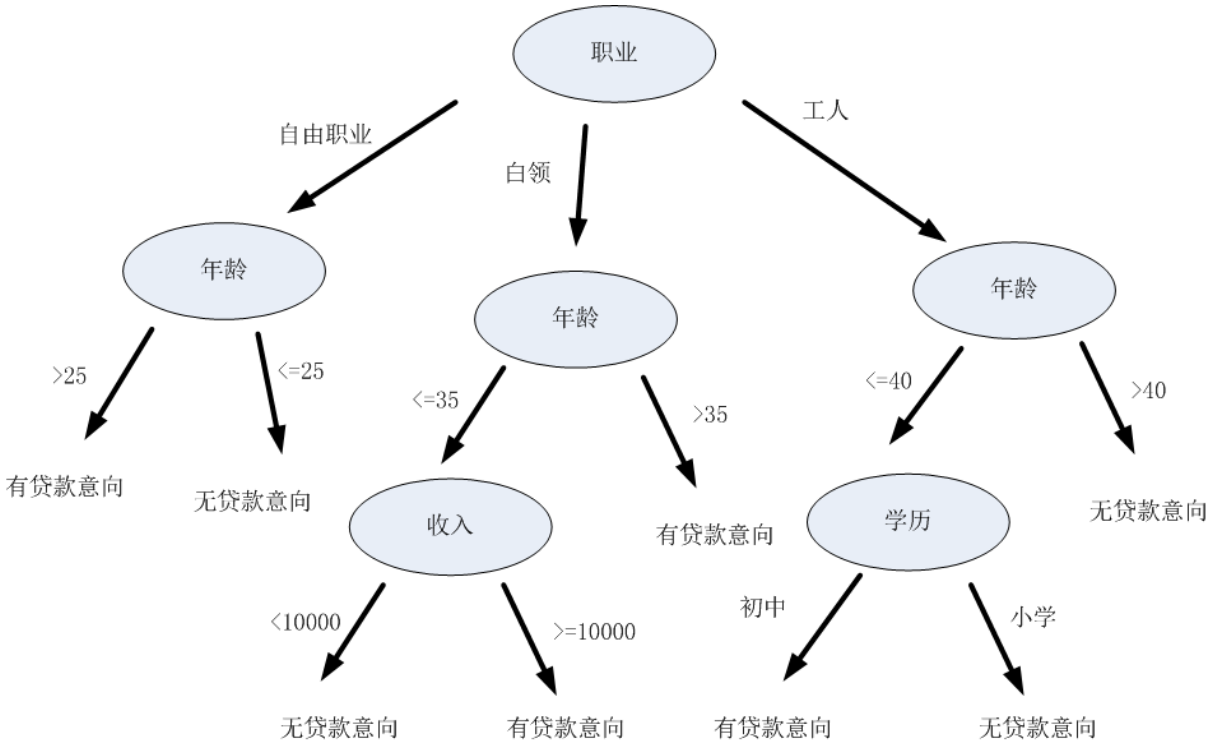
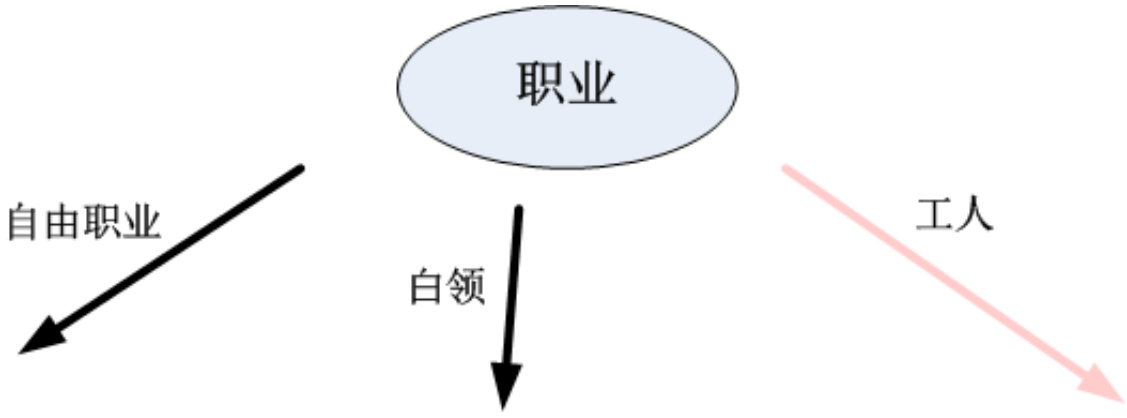


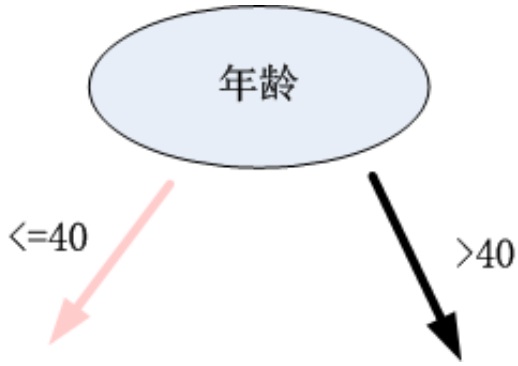
图2.1 银行贷款意向分析决策树示意图

通过图2.1的训练数据，我们可以建议图2.1所示的决策树，其输入是用户的信息，输出是用户的贷款意向。如果要判断某一客户是否有贷款的意向，直接根据用户的职业、收入、年龄以及学历就可以分析得出用户的类型。如某客户的信息为：{职业、年龄，收入，学历}={工人、39， 1800，小学}，将信息输入上述决策树，可以得到下列的分析步骤和结论。

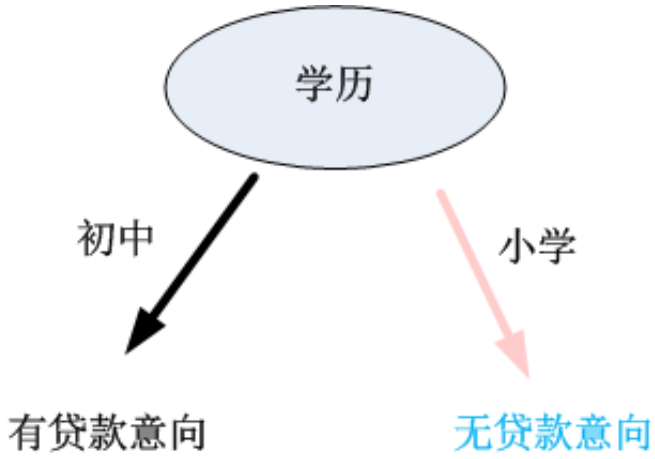
第一步：根据该客户的职业进行判断，选择“工人”分支；



第二步：根据客户的年龄进行选择，选择年龄”<=40”这一分支；



第三步：根据客户的学历进行选择，选择”小学”这一分支，得出该客户无贷款意向的结论。



- 决策树构建

如何构建如图2.1所示一棵决策树呢？决策树的构建是数据逐步分裂的过程，构建的步骤如下：

- 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数据特征对节点进行分割，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。

从上面几点可以看出，决策生成有几个重要的考虑点：

1. 数据如何分割
2. 如何选择分裂的属性
3. 什么时候停止分裂

假如我们已经选择了一个分裂的属性，那怎样对数据进行分裂呢?

分裂属性的数据类型分为离散型和连续性两种情况，对于离散型的数据，按照属性值进行分裂，每个属性值对应一个分裂节点；对于连续性属性，一般性的做法是对数据按照该属性进行排序，再将数据分成若干区间，如[0,10]、[10,20]、[20,30]…，一个区间对应一个节点，若数据的属性值落入某一区间则该数据就属于其对应的节点。

例:

表3.1 分类信息表

职业	年龄	是否贷款
白领	30	否
工人	40	否
工人	20	否
学生	15	否
学生	18	是
白领	42	是

（1）属性1"职业"是离散型变量，有三个取值，分别为白领、工人和学生，根据三个取值对原始的数据进行分割，如下表所示:

表3.2 属性1数据分割表

取值	贷款	不贷款
白领	1	1
工人	0	2
学生	1	1

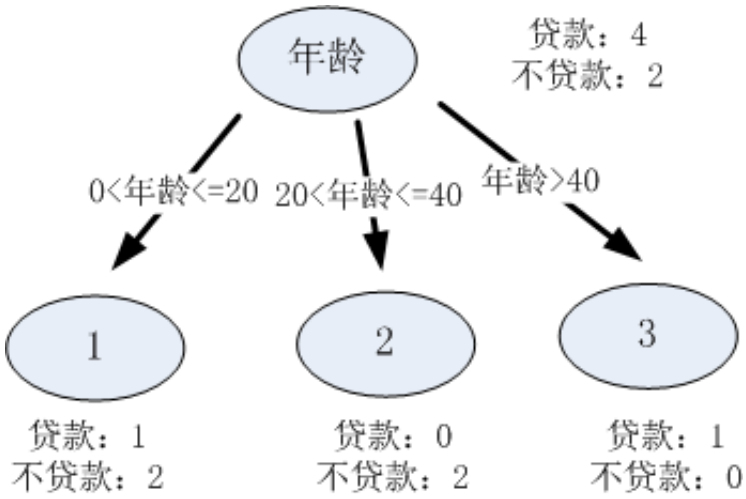
表3.2可以表示成如下的决策树结构：

（2）属性2是连续性变量，这里将数据分成三个区间，分别是[10，20]、[20,30]、[30,40]，则每一个区间的分裂结果如下：

表3.3 属性2数据分割表

区间	贷款	不贷款
[0,20]	1	2
(20,40]	0	2
(40,—]	1	0

表3.3可以表示成如下的决策树结构：



3.2 分裂属性的选择

我们知道了分裂属性是如何对数据进行分割的，那么我们怎样选择分裂的属性呢?

决策树采用贪婪思想进行分裂，即选择可以得到最优分裂结果的属性进行分裂。那么怎样才算是最优的分裂结果？最理想的情况当然是能找到一个属性刚好能够将不同类别分开，但是大多数情况下分裂很难一步到位，我们希望每一次分裂之后孩子节点的数据尽量”纯”，以下图为例：

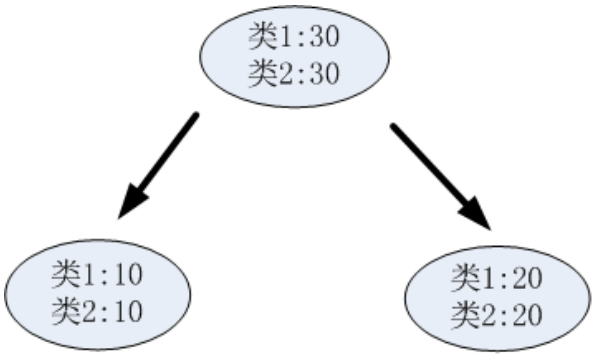


图3.1 按属性1进行分裂

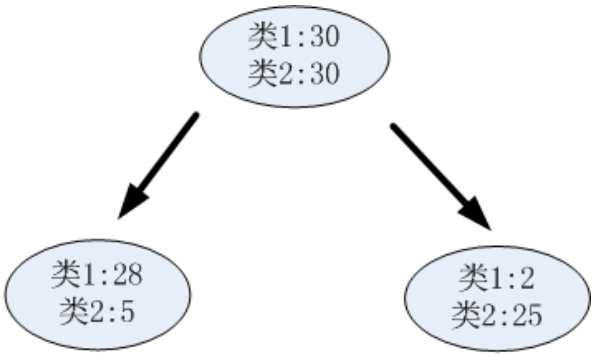


图3.2 按属性2进行分裂

从图3.1和图3.2可以明显看出，属性2分裂后的孩子节点比属性1分裂后的孩子节点更纯：属性1分裂后每个节点的两类的数量还是相同，跟根节点的分类结果相比完全没有提高；按照属性2分裂后每个节点各类的数量相差比较大，可以很大概率认为第一个孩子节点的输出结果为类1，第2个孩子节点的输出结果为2。

选择分裂属性是要找出能够使所有孩子节点数据最纯的属性，决策树使用信息增益或者信息增益率作为选择属性的依据。

（1）信息增益

用信息增益表示分裂前后跟的数据复杂度和分裂节点数据复杂度的变化值，计算公式表示为：

$$Info\_Gain = Gain - \sum_{i=1}^{n} Gain_i$$

其中Gain表示节点的复杂度，Gain越高，说明复杂度越高。信息增益说白了就是分裂前的数据复杂度减去孩子节点的数据复杂度的和，信息增益越大，分裂后的复杂度减小得越多，分类的效果越明显。

节点的复杂度可以用以下两种不同的计算方式：

a）熵

熵描述了数据的混乱程度，熵越大，混乱程度越高，也就是纯度越低；反之，熵越小，混乱程度越低，纯度越高。 熵的计算公式如下所示：

$$Entropy = -\sum_{i=1}^{n} p_i \cdot \log(p_i)$$

其中Pi表示类i的数量占比。以二分类问题为例，如果两类的数量相同，此时分类节点的纯度最低，熵等于1；如果节点的数据属于同一类时，此时节点的纯度最高，熵 等于0。

b）基尼值

基尼值计算公式如下：

$$Gini = 1 - \sum_{i=1}^{n} p_i^2$$

其中Pi表示类i的数量占比。其同样以上述熵的二分类例子为例，当两类数量相等时，基尼值等于0.5 ；当节点数据属于同一类时，基尼值等于0 。基尼值越大，数据越不纯。

例：

以熵作为节点复杂度的统计量，分别求出下面例子的信息增益，图3.1表示节点选择属性1进行分裂的结果，图3.2表示节点选择属性2进行分裂的结果，通过计算两个属性分裂后的信息增益，选择最优的分裂属性。

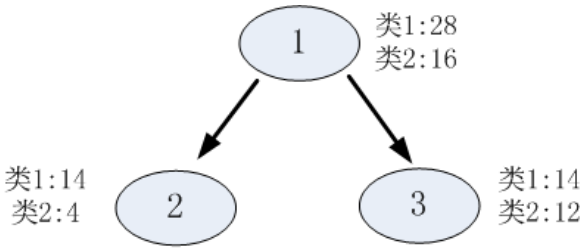


图3.3 根据属性1分裂

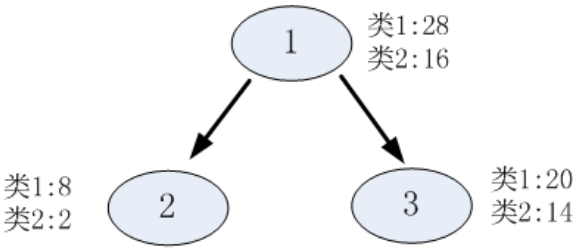


图3.4 根据属性2分裂

属性1：

$$Info_1 = Entropy - \sum_{i=1}^{n} Entropy_i =$$

$$\frac{16}{28+16} \cdot \log\left(\frac{16}{28+16}\right) + \frac{28}{28+16} \cdot \log\left(\frac{28}{28+16}\right) \Rightarrow Entropy$$

$$-\left(\frac{14}{14+4} \cdot \log\left(\frac{14}{14+4}\right) + \frac{4}{14+4} \cdot \log\left(\frac{4}{14+4}\right)\right) \Rightarrow Entropy1$$

$$-\left(\frac{14}{14+12} \cdot \log\left(\frac{14}{14+12}\right) + \frac{12}{14+12} \cdot \log\left(\frac{12}{14+12}\right)\right) \Rightarrow Entropy2$$

$$= 0.81$$

属性2：

$$Info_2 = Entropy - \sum_{i=1}^{n} Entropy_i =$$

$$\frac{16}{28+16} \cdot \log\left(\frac{16}{28+16}\right) + \frac{28}{28+16} \cdot \log\left(\frac{28}{28+16}\right) \Rightarrow Entropy$$

$$-\left(\frac{8}{8+2} \cdot \log\left(\frac{8}{8+2}\right) + \frac{2}{8+2} \cdot \log\left(\frac{2}{8+2}\right)\right) \Rightarrow Entropy1$$

$$-\left(\frac{20}{20+14} \cdot \log\left(\frac{20}{20+14}\right) + \frac{14}{20+14} \cdot \log\left(\frac{14}{20+14}\right)\right) \Rightarrow Entropy2$$

$$= 0.75$$

由于 ，所以属性1与属性2相比是更优的分裂属性，故选择属性1作为分裂的属性。

（2）信息增益率

使用信息增益作为选择分裂的条件有一个不可避免的缺点：倾向选择分支比较多的属性进行分裂。为了解决这个问题，引入了信息增益率这个概念。信息增益率是在信息增益的基础上除以分裂节点数据量的信息增益（听起来很拗口），其计算公式如下：

$$Info\_Ratio = \frac{Info\_Gain}{InstrinsicInfo}$$

其中 $Info\_Gain$ 表示信息增益，$InstrinsicInfo$ 表示分裂子节点数据量的信息增益，其计算公式为：

$$InstrinsicInfo = -\sum_{i=1}^{m} \frac{n_i}{N} \cdot \log(\frac{n_i}{N})$$

其中m表示子节点的数量，n_i 表示第i个子节点的数据量，N表示父节点数据量，说白了， 其实是分裂节点的熵，如果节点的数据链越接近，越大，如果子节点越大，越大，而 $Info\_Ratio$ 就会越小，能够降低节点分裂时选择子节点多的分裂属性的倾向性。信息增益率越高，说明分裂的效果越好。

还是信息增益中提及的例子为例：

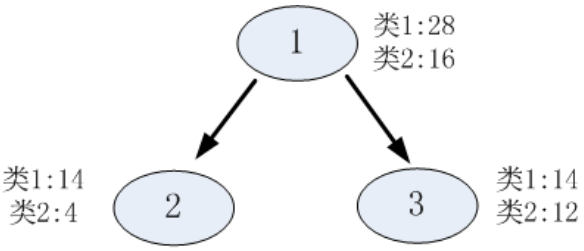


图3.3 根据属性1分裂

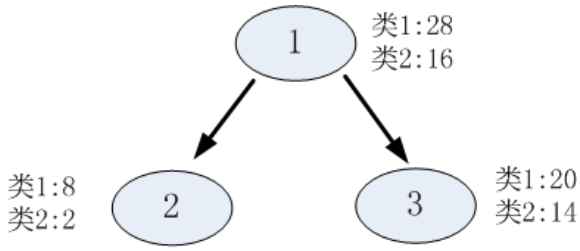


图3.4 根据属性2分裂

属性1的信息增益率：

$$Info\_Gain_1 = 0.81$$

$$IntrinsicInfo_1 = -(\frac{18}{18+26} \cdot \log(\frac{18}{18+26}) + \frac{26}{18+26} \cdot \log(\frac{26}{18+26}))$$
$$= 0.97$$
$$Info\_Ratio_1 = \frac{Info\_Gain_1}{IntrinsicInfo_1} = 0.84$$

属性2的信息增益率：

$$Info\_Gain_2 = 0.75$$

$$IntrinsicInfo_2 = -(\frac{10}{10+34} \cdot \log(\frac{10}{10+34}) + \frac{34}{10+34} \cdot \log(\frac{34}{10+34}))$$
$$= 0.77$$
$$Info\_Ratio_2 = \frac{Info\_Gain_2}{IntrinsicInfo_2} = 0.97$$

由于 $Info\_Ratio_2 > Info\_Ratio_1$ ，故选择属性2作为分裂的属性。

3.3 停止分裂的条件

决策树不可能不限制地生长，总有停止分裂的时候，最极端的情况是当节点分裂到只剩下一个数据点时自动结束分裂，但这种情况下树过于复杂，而且预测的经度不高。一般情况下为了降低决策树复杂度和提高预测的经度，会适当提前终止节点的分裂。

以下是决策树节点停止分裂的一般性条件：

（1）最小节点数

当节点的数据量小于一个指定的数量时，不继续分裂。两个原因：一是数据量较少时，再做分裂容易强化噪声数据的作用；二是降低树生长的复杂性。提前结束分裂一定程度上有利于降低过拟合的影响。

（2）熵或者基尼值小于阀值。

由上述可知，熵和基尼值的大小表示数据的复杂程度，当熵或者基尼值过小时，表示数据的纯度比较大，如果熵或者基尼值小于一定程度数，节点停止分裂。

（3）决策树的深度达到指定的条件

节点的深度可以理解为节点与决策树跟节点的距离，如根节点的子节点的深度为1，因为这些节点与跟节点的距离为1，子节点的深度要比父节点的深度大1。决策树的深度是所有叶子节点的最大深度，当深度到达指定的上限大小时，停止分裂。

（4）所有特征已经使用完毕，不能继续进行分裂。

被动式停止分裂的条件，当已经没有可分的属性时，直接将当前节点设置为叶子节点。

3.4 决策树的构建方法

根据决策树的输出结果，决策树可以分为分类树和回归树，分类树输出的结果为具体的类别，而回归树输出的结果为一个确定的数值。

决策树的构建算法主要有ID3、C4.5、CART三种，其中ID3和C4.5是分类树，CART是分类回归树，将在本系列的***ID3***、***C4.5***和***CART***中分别讲述。

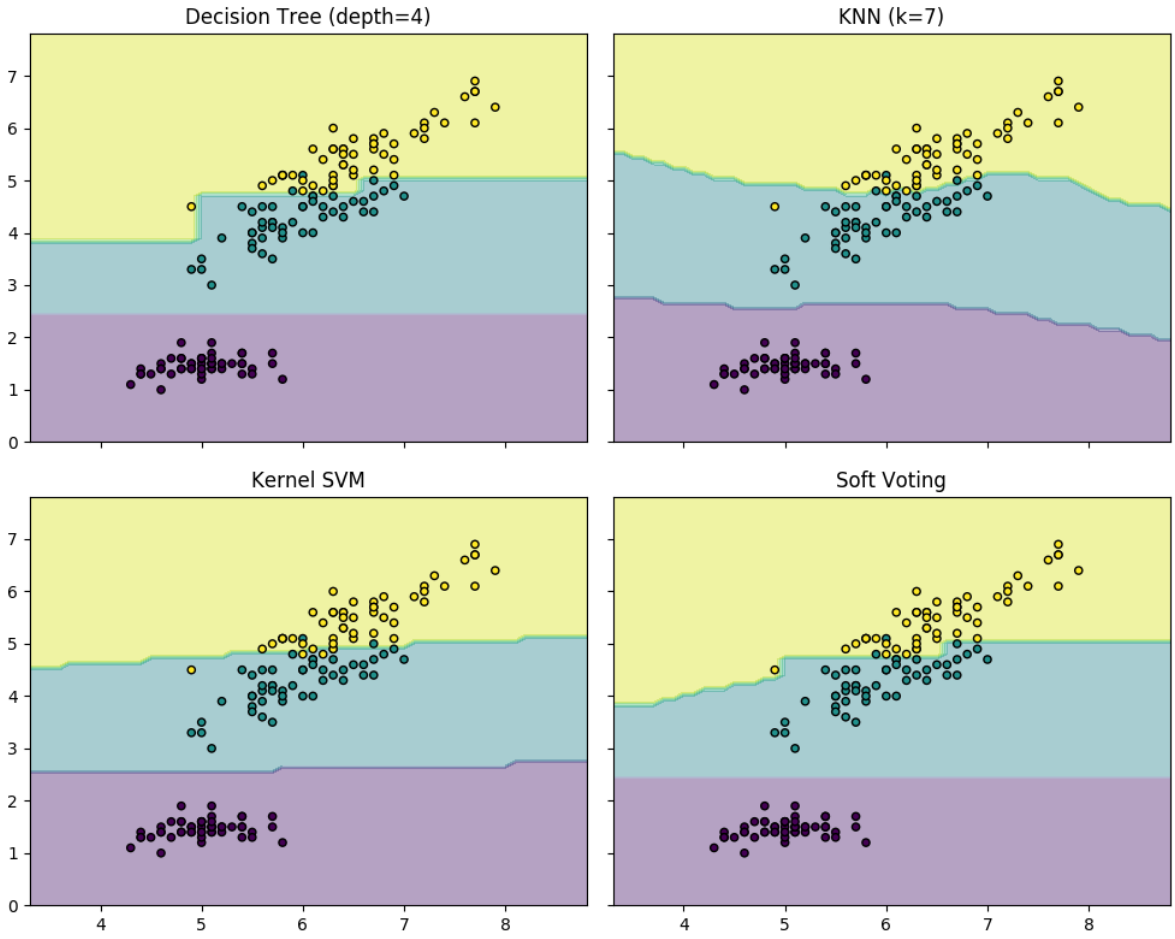
其中ID3是决策树最基本的构建算法，而C4.5和CART是在ID3的基础上进行优化的算法。

- 决策树Demo

```
print(__doc__)

from itertools import product

import numpy as np
import matplotlib.pyplot as plt

from sklearn import datasets
from sklearn.tree import DecisionTreeClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.svm import SVC
from sklearn.ensemble import VotingClassifier

# Loading some example data
iris = datasets.load_iris()
X = iris.data[:, [0, 2]]
y = iris.target

# Training classifiers
clf1 = DecisionTreeClassifier(max_depth=4)
clf2 = KNeighborsClassifier(n_neighbors=7)
clf3 = SVC(gamma=.1, kernel='rbf', probability=True)
eclf = VotingClassifier(estimators=[('dt', clf1), ('knn', clf2),
                                    ('svc', clf3)],
```

```
                          voting='soft', weights=[2, 1, 2])

clf1.fit(X, y)
clf2.fit(X, y)
clf3.fit(X, y)
eclf.fit(X, y)

plt.scatter(X,y)
plt.show()

# Plotting decision regions
x_min, x_max = X[:, 0].min() - 1, X[:, 0].max() + 1
y_min, y_max = X[:, 1].min() - 1, X[:, 1].max() + 1
xx, yy = np.meshgrid(np.arange(x_min, x_max, 0.1),
                     np.arange(y_min, y_max, 0.1))

f, axarr = plt.subplots(2, 2, sharex='col', sharey='row', figsize=
(10, 8))

for idx, clf, tt in zip(product([0, 1], [0, 1]),
                        [clf1, clf2, clf3, eclf],
                        ['Decision Tree (depth=4)', 'KNN (k=7)',
                         'Kernel SVM', 'Soft Voting']):

    Z = clf.predict(np.c_[xx.ravel(), yy.ravel()])
    Z = Z.reshape(xx.shape)

    axarr[idx[0], idx[1]].contourf(xx, yy, Z, alpha=0.4)
    axarr[idx[0], idx[1]].scatter(X[:, 0], X[:, 1], c=y,
                                  s=20, edgecolor='k')
    axarr[idx[0], idx[1]].set_title(tt)

    plt.show()
```

运行结果：


Decision Tree (depth=4)
KNN (k=7)
Kernel SVM
Soft Voting
0
1
2
3
4
5
6
7
4
5
6
7
8