**【量化课堂】决策树入门及Python应用**

**简介：**

决策树是一种非参的监督性机器学习方法。它可以用来做分类判断（classification）和回归预测（regression）。决策树的基本原理是通过学习现有数据的特征，得到简单的决策规律，再根据这些决策规律对目标进行判断。

本文由 JoinQuant 量化课堂推出 。难度标签为入门，理解深度标签：level-1

作者： 大白

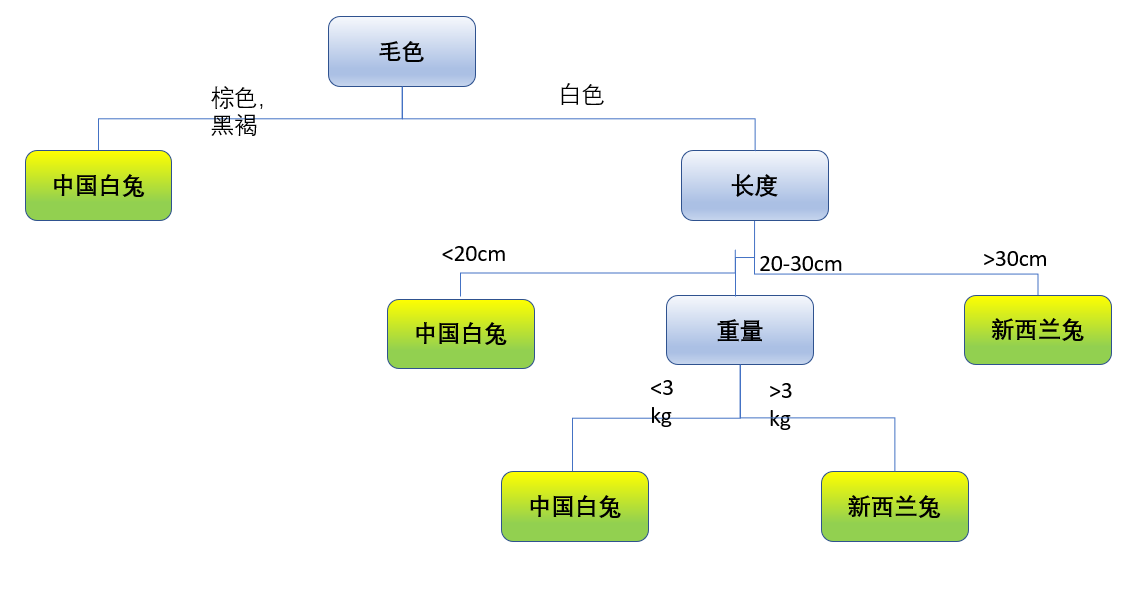
编辑： 肖睿

**本章分为：**  
1，决策树的概念  
2，决策树的生成过程

* 1） 数据纯度判断方法  
  2） 决策树过拟合处理

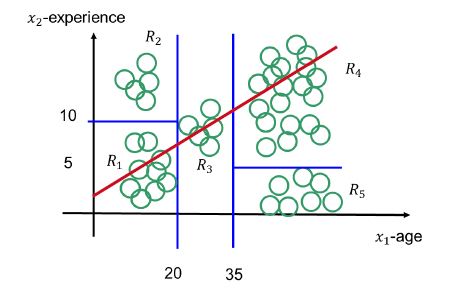
3，决策树的python代码实现（如何让计算机来帮你寻找数据特征）  
4，补充和扩展

**一，决策树的概念**

决策树是在已知各种情况发生概率的基础上，通过构成决策树来判断下一预测点，判断其是否会发生的分析方法，是直观运用概率分析的一种图解法。由于这种决策树分支的图形很像树的枝干，故称决策树。下面是一个决策树的例子：  


在上面例子中我们需要对新的兔子进行判断，根据这一问题建立一棵决策树，上图也展示了决策树的组成部分：根节点，分支和叶子。首先上述决策树的根节点是“毛色”，在这一基础上延伸出两个分支“黑褐色”和“白色”，倘若为黑褐色，进入叶子得到结论这是一只“中国白兔；倘若为“白色”则进入下一个节点判断，一直到没有节点的叶子为止。对于一只40厘米的白色成年兔子，首先判断其毛色为白色，进入长度节点，然后长度大于30cm，基于此我们可判断这只成年兔子为新西兰兔。

决策树本质上是基于过去的特征信息对过去目标结论的判断的总结，运用于金融预测领域直观的解释就是：由于这件事在过去相同特征的环境下大概率发生，所以此时这件事也很可能会发生，及“历史总是惊人的相似”。决策树在机器学习领域的应用在于可以把人从浩如烟海的数据中解放出来，让计算机去寻找数据之间的内在规律用于下一步研究，也可以直接使用决策树的结论进行判断，但不需要可视化的情况下目前更多的会去选择Randomforest，Boosting等决策树的改进。决策树基于的假设较少，适用于大部分情况。

但其缺点也很明显：  
1，精度较低  
2，他每次只会根据单一特征划分数据，不会根据数据组合切分。如下图, 他只会根据年龄或经验区分数据，但当两者有联系时，决策树只会用多次切分拟合这一情况。  
  
3，受误差值影响较大

**二，决策树的生成过程**

了解了决策树的原理，那如何根据问题生成决策树呢？我们知道决策树是由根节点，分叉，叶组成的，决策树的生成就是寻找根节点、节点和如何生成分叉了。决策树生成的核心思想就是找出更加纯净的子集，最好每个子集里都是结论极其单一的数据。判断纯度的方法不同决策树的生成也不同，常用的判断方法有：1，ID3算法使用信息增益作纯度判断；2，C4.5算法使用信息增益率作纯度判断；3，CART算法使用基尼系数作纯度判断。

**决策树生成过程：**

1， 寻找最适合分割的特征

2， 根据纯度判断方法，寻找最优的分割点，基于这一特征把数据分割成纯度更高的两部分数据

3， 判断是否达到要求，若未达到，重复步骤一继续分割，直到达到要求停止为止。

4， 剪枝，防止过拟合。

纯度判断和过拟合的具体做法在下面两部分讲到

**数据纯净度的判别**

决策树生成时，我们希望每个分支的数据尽量属于同一类别，也就是寻找纯净的分类，判断数据纯净程度的方法也决定了我们如何生成分支，目前主流的方法包括ID3，C4.5和CART三种。

在介绍这三种方法之前先简单介绍信息熵：  
**信息熵：**  
熵是衡量系统混乱程度的一个指标，熵值越大表示对应的系统越混乱，决策树中引入信息熵的概念来衡量数据的纯净程度。假定目标集合S中有n种样本，第k种样本所占比例为pk（k=1,2,3,…,n）,则S的信息熵为：

Ent(S)=−∑k=1npklog2pk

Ent(S)=−∑k=1npklog2pkEnt(S)=−∑k=1npklog2⁡pk

**ID3算法：**

ID3算法用信息增益来衡量混乱程度  
**信息增益:**  
Gain(k)=分裂前目标变量的信息熵-对特征值k分裂后的目标变量信息熵  
若总数据量为D，分裂后数据成为有M个叶节点的分叉树，那么Gain（k）为

Gain(k)=Ent(S)−∑m=1MEnt(Dm)

Gain(k)=Ent(S)−∑m=1MEnt(Dm)Gain(k)=Ent(S)−∑m=1MEnt(Dm)

**分叉点的选择：**遍历全部现有可能分叉点，选择信息增益最大的分叉点建立新的分叉。  
**停止分裂的条件：**  
1，节点中全部样本的目标变量同属于一个类别  
2，如果分叉后的样本数目小于给定的阀值，也停止进行分叉。

**叶节点的属性：**选择叶节点里数量最多的类别作为叶节点的类别

**C4.5算法：**

C4.5算法用信息增益率来衡量混乱程度  
**信息增益率:**Gain\_ratio(k)=信息增益率/被选分叉点本来的熵

Gain\_ratio(k)=Gain(S)SplitEntropy(Dk)

Gain\_ratio(k)=Gain(S)SplitEntropy(Dk)Gain\_ratio(k)=Gain(S)SplitEntropy(Dk)

其中，

SplitEntropy(k)=Ent(Dk)

SplitEntropy(k)=Ent(Dk)SplitEntropy(k)=Ent(Dk)

Ent(Dk)Ent(Dk)Ent(Dk) 和Ent(S)Ent(S)Ent(S) 的区别是：一个计算的是总集合S的熵，一个计算的是总集合S中节点k的熵。

**分叉点的选择：**遍历全部现有可能分叉点，选择信息增益率最大的分叉点建立新的分叉。  
**停止分裂的条件：**  
1 节点中全部样本的目标变量同属于一个类别  
2 如果分叉后的样本数目小于给定的阀值，也停止进行分叉。

**叶节点的属性：**选择叶节点里数量最多的类别作为叶节点的类别

**特征：**比起ID3算法来，C4.5更倾向于选择区直更少的特征值分叉。

**CART算法：**

CART算法用基尼指数来衡量数据集混乱程度  
**基尼值：**

Gini(S)=1−∑k=1np2k

Gini(S)=1−∑k=1np2kGini(S)=1−∑k=1npk2

基尼值(S)指的是从数据S中随机取两个不同类别值的概率，因此Gini值越大意味着数据越混乱。  
若总数据量为D，分裂后数据成为有M个叶节点的分叉树，那么Gini\_index(k)为

**基尼指数：**

Gini-index(k)=∑m=1MGini(Dm)

Gini-index(k)=∑m=1MGini(Dm)Gini-index(k)=∑m=1MGini(Dm)

**分叉点的选择：** 遍历全部现有可能分叉点，选择基尼指数最小的分叉点建立新的分叉。  
**停止分裂的条件：**  
1，决策树到达最大深度  
2，分叉节点的样本数小于阀值  
3，分叉的叶内的样本数小于阀值  
**叶节点的属性：**选择叶节点里数量最多的类别作为叶节点的类别

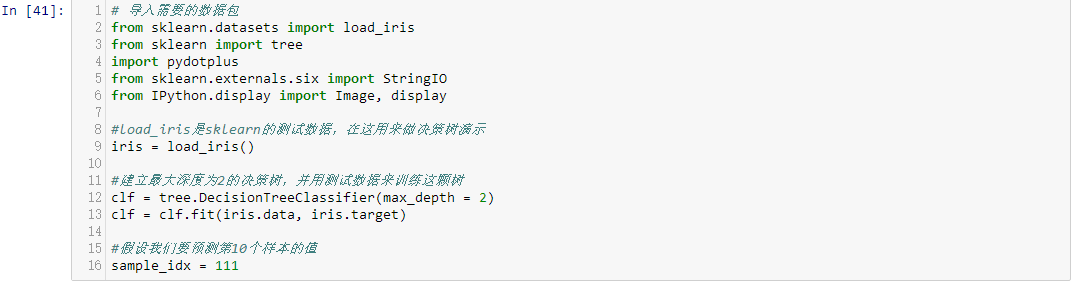
**过拟合处理**

在决策树学习过程中，因为随着子集样本越小，混乱程度必然下降，这意味着决策树分叉总是向着过拟合的方向。为了降低过拟合风险，需要主动去除一些分类效果不明显的分叉来防止过拟合，这一过程被称为剪枝。  
剪枝主要有两种方法：预剪枝和后剪枝  
**预剪枝：**  
预剪枝是在生成决策树过程中，对节点划分先进性预估，若节点分叉不能使决策树得到泛化提升，则停止分叉生成叶。代表算法是悲观错误剪枝法PEP（Pesimistic-Error Pruning）  
**后剪枝：**  
后剪枝是生成一颗完整的决策树后，自下而上对每个叶片和节点进行评估，若减少分叉可以使决策树得到泛化提升，则转化分叉为一个叶。代表算法有错误率降低剪枝REP（Reduced-Error Pruning），代价复杂度剪枝CCP（Cost-Complexity Pruning）

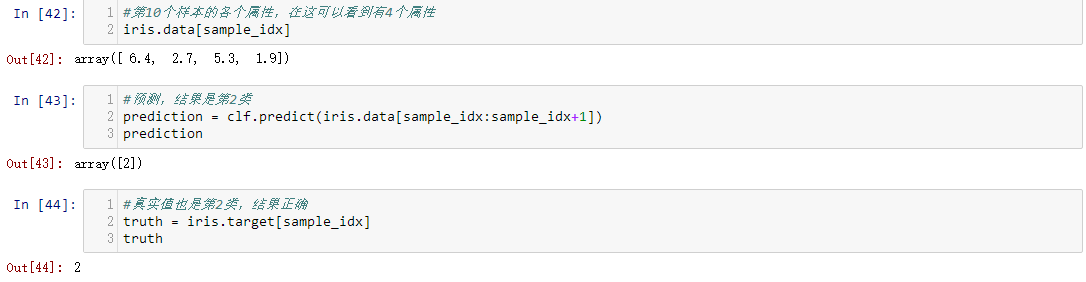
**三、决策树的python代码实现**

决策树在实际中一般有两个用处：1，直接用于预测。2，借助决策树的实现图形寻找规律。  
第一种很好理解，和大多数模型一样，建立模型预测。  
第二种的实现是基于决策树的原理与人类思考接近。决策树的实现过程本来就是纯化子集的过程，生成决策树后，他的每一个叶子都应该是比较纯净的，及它可以找到在某些特征条件下结论较为一致的情境。研究者可以根据决策树找到的一些纯净度比较高的情境，省去大量找规律的时间。

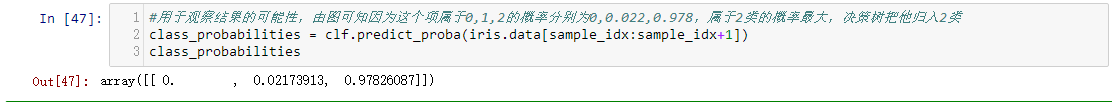
**1，学习**

首先我们使用机器学习库sklearn中的数据来介绍决策树的主要功能。  


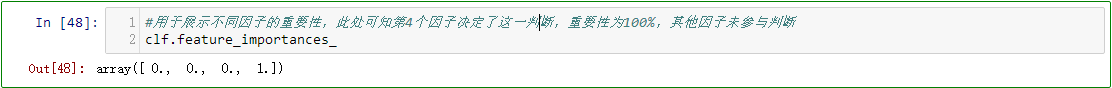
首先导入必要的包和数据，在这我们以预测第111个数据作为例子



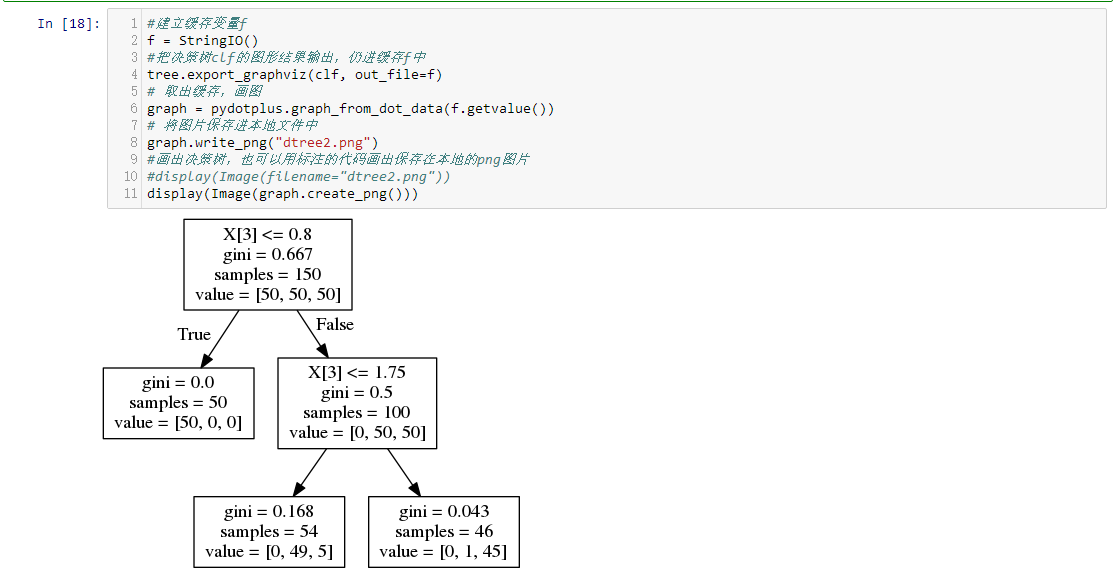
查看用作例子的数据的各个特征值，将其特征值带入建立好的决策树预测，可以看到结果与真实值相同。



predict\_proba可以用来观察决策树判断的依据。决策树的结论是： 这一数据属于0,1,2的概率分别为0,0.22,0.978，选取最大概率，这一数据被认为属于第2类。



feature*impertances*可以用来查看各个因子的重要性。显示因子4重要性占所有因子的100%，整个决策树完全依赖于因子4.



画出决策树，有两个节点3片叶子，叶子的纯度都比较高，都大于90%。

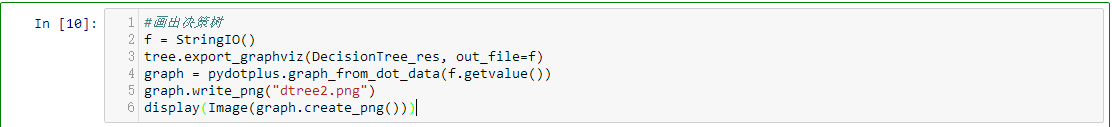
**2，实验**

接下来举个简单的例子展示决策树在实际股市分析预测中的运用。  
已下是一个使用决策树预测HS300涨跌测试函数，用市场平均的'PCF','turnover\_ratio', 'PE\_rate','PB\_rate','PS\_rate',以及宏观数据'pmi', 'pmi\_rate', 'leading\_idx\_rate', 'M1\_rate', 'M2\_rate', 'CPI\_rate', 'GDP\_rate', 'LOAN6MONTH', 'LARGEFINANCIAL', 'SMALLFINANCIAL', 'dcg', 'URUR' 这些数据使用决策树算法对HS300的涨跌进行预测，预测结果有四种，分别是：‘下跌大于5%’，‘下跌介于5%与0%之间’，‘上涨介于0%和5%’，‘上涨大于5%’，。数据取了2005年5月到2017年11月的全部月数据，以其中70%做训练，以30%数据做测试，所有宏观数据都根据发布时间取了滞后两个月或者一个季度的数据来做预测，以确保实际中可以取到这些值。

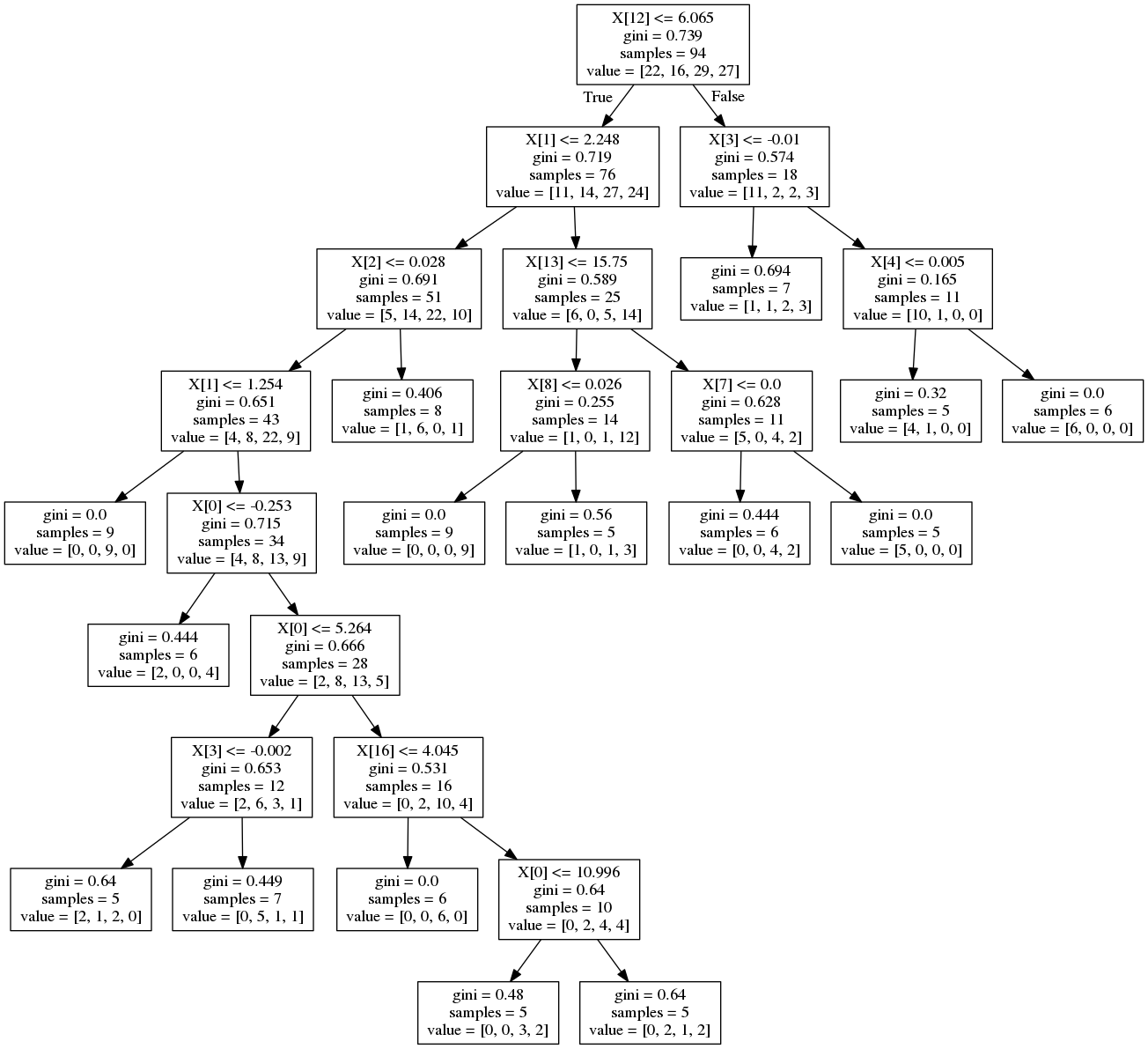


以上是决策树建立的代码，我们用CART的方法用‘gini值’确定叶数据纯度，并要求节点最少应该有10个数据，每片叶最少有5个数据。  
决策树模型的主要参数有：  
**criterion**: 决定判断节点纯度的方法。目前sklearn支持”gini”和”entropy”两种，分别对应用基尼指数和信息增益的CART和ID3两种算法，默认算法是“gini”。  
**max\_depth**: 最大树深度，达到最大数深度即停止分叉，最常用的约束，默认值为None  
**spliter**: 决定如何分叉每个节点。目前sklearn支持“best”和“random”两种，指的是选取特征的最优分叉点分叉还是随机分叉点分叉，默认值是“best”  
**max\_features**: 使用特征值的最大数量，默认为None  
**max\_leaf\_nodes**: 叶节点的数量，默认值为None  
**min\_impurity\_split**: 最小纯度门槛，假如纯度小于最小纯度，则会继续切分，默认值为None  
**min\_samples\_leaf**: 最小叶数据，达到则不再分叉，默认值为1  
**min\_samples\_split**: 最小节点数据，达到则不再分叉，默认值是2

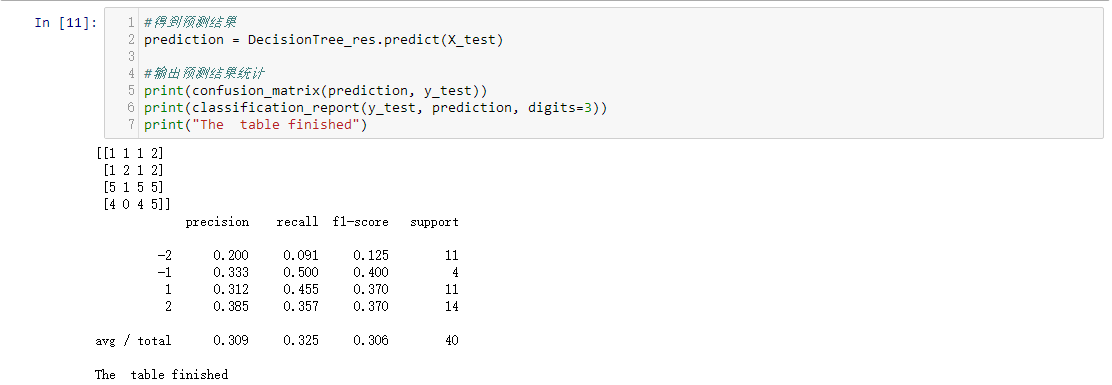
决策树的参数一般依情况而定，可以根据需要不断调整。也可以用sklearn的包GridSearchCV用cross validation的方法暴力遍历指定范围参数确定结果。调参在其他机器学习方法如神经网络模型中也是核心问题，一般是使用者的经验加计算机尝试不同的值得到一个相对较好的结果。对于决策树，初学者一般限定树深度即可，其他设置可使用默认参数。

接下来即可使用决策树进行预测  


上述代码即可使用计算机自动生成的决策树图解，其叶片都是比较纯净的子集，可以帮助研究者筛选规律。



以上是决策树的整个图解，挖掘到了一些特征，例如最右边的叶表示当市场平均的'PB\_rate'<=-0.01, ‘PS\_rate’<=0.05时，历史上6次这种情况下HS300都发生大跌。



这是取2005年5月到2017年11月全部数据后，取70%做训练集，30%做测试验证结果，得到的精度统计，表格如下所示：

| **实际变化\预测** | **<-0.5%** | **-0.5%~0%** | **0%~0.5%** | **>0.5%** |
| --- | --- | --- | --- | --- |
| <-0.5% | 1 | 1 | 1 | 2 |  |
| -0.5%~0% | 1 | 2 | 1 | 2 |  |
| 0%~0.5% | 5 | 1 | 5 | 5 |  |
| >0.5% | 4 | 0 | 4 | 5 |  |

精度很差，只是略高于一半而已。简单的决策树过于基础，识别能力有限，用于预测股市显然很一般。

不过决策树是很多机器学习方法的基石，其改进模型往往会有较好表现。此外决策树便于可视化，他的产生的分布图可以用于数据研究，例如决策树的每一片页都是相对比较纯净的数据集，使用者可以抽出一些纯度比较高的叶来研究为何在训练集中这些叶上的数据集纯度较高，从而发现新规律。

**补充和扩展**

**写在最后：**  
决策树由于理论简单，易于理解，在数据可视化，利用计算机寻找数据规律中有不小的作用。但其模型过于简单，对于复杂的模型一般来说只是优于线性模型而已，对比其他算法并没有太大优势，其简单的结构也不能充分利用目前强大的计算机计算能力，已经不怎么活跃了。但是作为极易可视化，逻辑清晰的一种白盒算法，其不仅活跃于数据研究，而且也是其他高级算法的基石。决策树的两个主要改进方向是Randomforest 和 boosting，都在机器学习中有举足轻重的地位。

本文由JoinQuant量化课堂推出，版权归JoinQuant所有，商业转载请联系我们获得授权，非商业转载请注明出处。