决策树学习笔记（二）：剪枝，ID3，C4.5

原创 wLsq Python数据科学 2019-01-10

收录于话题
#Python数据科学 60 #机器学习 18

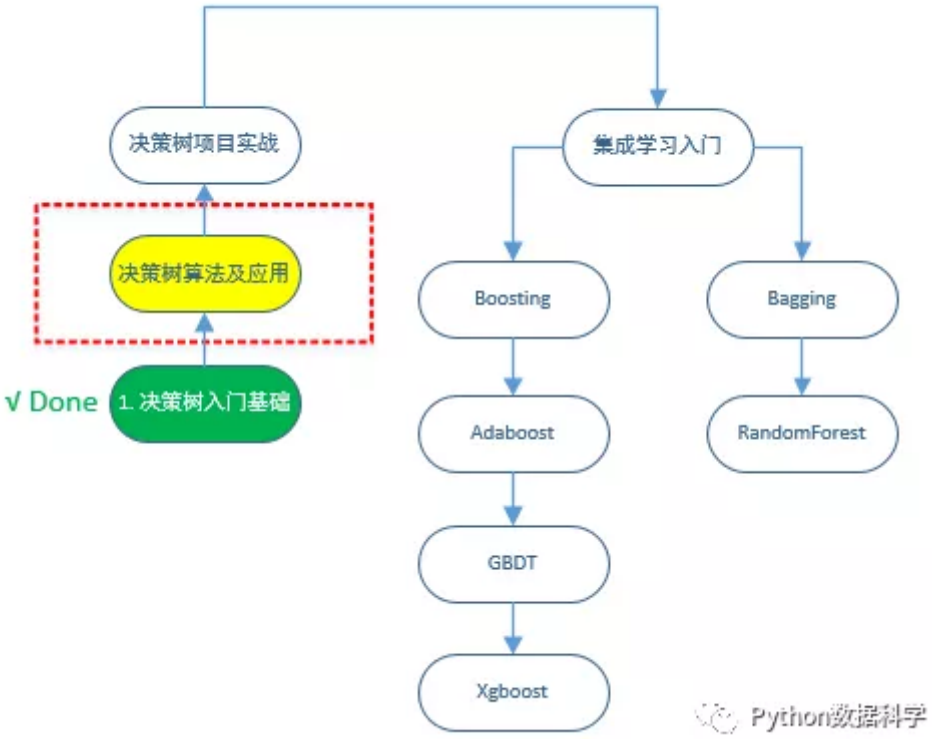
点击上方**“Python数据科学”**，选择**“星标公众号”**

关键时刻，第一时间送达！

作者：xiaoyu

介绍：一个半路转行的数据挖掘工程师

推荐导读：本篇为树模型系列第二篇，旨在从最简单的决策树开始学习，循序渐进，最后理解并掌握复杂模型GBDT，Xgboost，为要想要深入了解机器学习算法和参加数据挖掘竞赛的朋友提供帮助。



「树模型学习系列」进度追踪

上一篇主要介绍了决策树的一些基础概念，以及特征选择的三个度量指标：信息增益，增益率，基尼指数，传送门： 决策树学习笔记（一）：特征选择。

本篇将详细介绍决策树常用的三种算法，剪枝处理，缺失值，决策树优缺点，以及常见的应用场景。

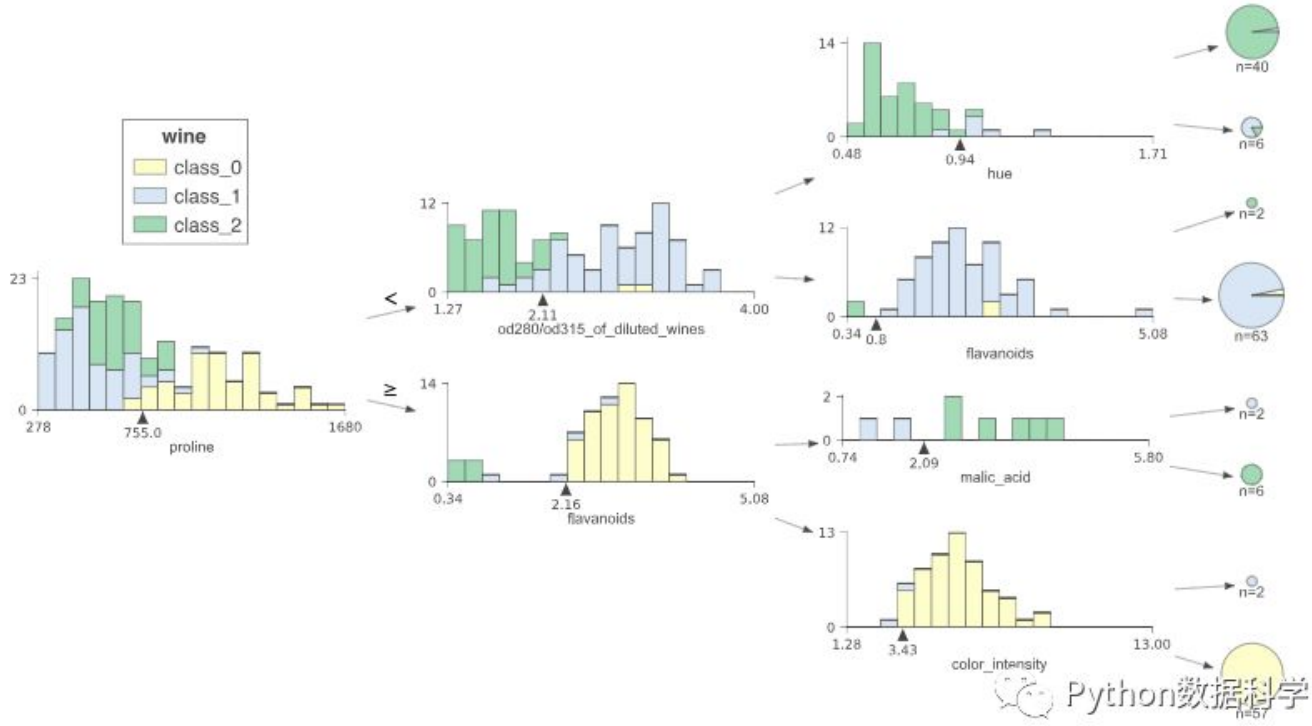
- 决策树的生成
- 决策树的剪枝
- 决策树三种算法概况
- 总结

决策树的生成

决策树的生成其实就是不断地向下构建决策树节点，最终形成一颗完整的决策树模型。其中，节点建立的度量标准可以是信息增益，增益率，基尼指数等。那么如何通过已有的度量标准不断地构建决策树节点呢？

我们可以用数学上的**递归**方法解决，就如**数据结构二叉树**一样。设置判断标准，设置递归的停止条件，归纳并实现决策树的不断生成。递归方面的内容也可以参考：如何用Python递归地思考问

题？下图就是用递归生成一颗完整决策树的过程。



递归生成决策树的伪代码如下：

```
Input:  训练集D={(x1, y1), (x2, y2), ..., (xm, ym)};
        属性集A={a1, a2, ..., ad}.
Output: 以node为根节点的一个决策树

Process:
## 通过给定样本集D和属性集A构建决策树
TreeGenerate(D, A){
    1: 生成结点node;
    2: if D 中样本全属于同一类别C then
    3:      将node标记为 C类 叶节点; return
    4: end if
    5: if A = ∅ OR D中样本在A上取值相同 then
    6:      将node标记为叶节点，其类别标记为D中样本数最多的类; return
    7: end if
    8: 从 A 中选择最优化分属性 a*
    9: for a* 的每一值a[i] do
   10:      为node生成一个分支; 令Dv表示D中在 a* 上取值为 a[i] 的样本子集;
   11:      if Dv is empty then
   12:          将分支结点标记为叶节点，其类别为D中样本最多的类; return
   13:      else
   14:          以 TreeGenerate(Dv, A\{a*}) 为分支结点;
   15:      end if
   16: end for
}
```

使用Python实现的递归构建决策树如下：

```
def treeGrow(dataset,features):
    # 停止条件(1)(2)
```

```
    if  len(classList) == classList.count(classList[0]):  #no more feature
        return classifyMaxLabel(classList)
    if  len(dataSet[0]) == 1:  # pure dataset
        return classList[0]

    # 特征选择
    bestFeature = findBestSplit(dataset)
    del bestFeature

    # 划分特征并递归调用
    SplitData = SplitDataset(dataset, bestFeature)
    tree = treeGrow(SplitData, features)
    return tree
```

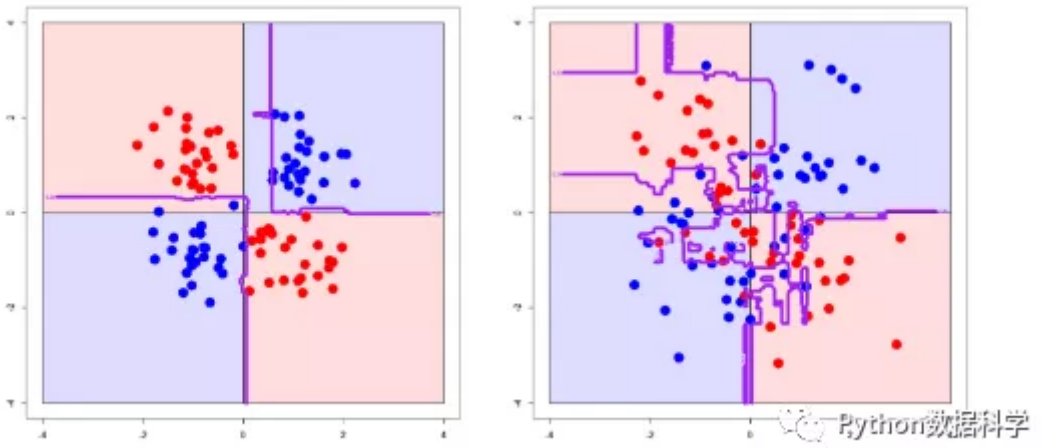
上面递归函数的过程是：

- 先定义停止条件：(1)没有更多特征供选择了；(2)数据集本身就已经分类好了，纯数据集。满足这两个中任何一个条件树生成就停止。
- 特征选择：根据自己选择的度量标准来选择特征。
- 递归地调用treeGrowth函数并根据选择特征不断地生成子树，直到达到停止条件。

> 注：上面代码只是一个决策树递归生成的框架示例，细节部分不完整，具体实现还需要补充。

决策树的剪枝

决策树是一个非常容易发生过拟合的模型，因为如果没有任何限制，在生成的阶段，它将会穷尽所有的特征，直到停止条件。这时叶子节点的数目最多，而叶子节点越多则越容易发生过拟合缺少泛化能力。如下图所示，右图就是未剪枝后的过拟合情况，显然对于新的数据集效果将会很差。



那么如何对一颗决策树进行一些限制呢？

可以通过**正则化**来解决，我们之前提到过正则化的问题：【机器学习笔记】：解读正则化，LASSO回归，岭回归。在决策树中被称为**剪枝**，就是将树生成的不必要子树剪掉，减少叶子节点数量，降低树模型复杂度。

总的来说，剪枝可分为：**预剪枝，后剪枝**两类。

预剪枝(pre-pruning)

预剪枝的重点在“**预**“字。它是指在完全正确分类之前，决策树会较早地停止树的生长。而终止树继续向下生长的方法有很多，我把停止生长的方法总结为通用的停止和更严格的停止两种。

通用的停止

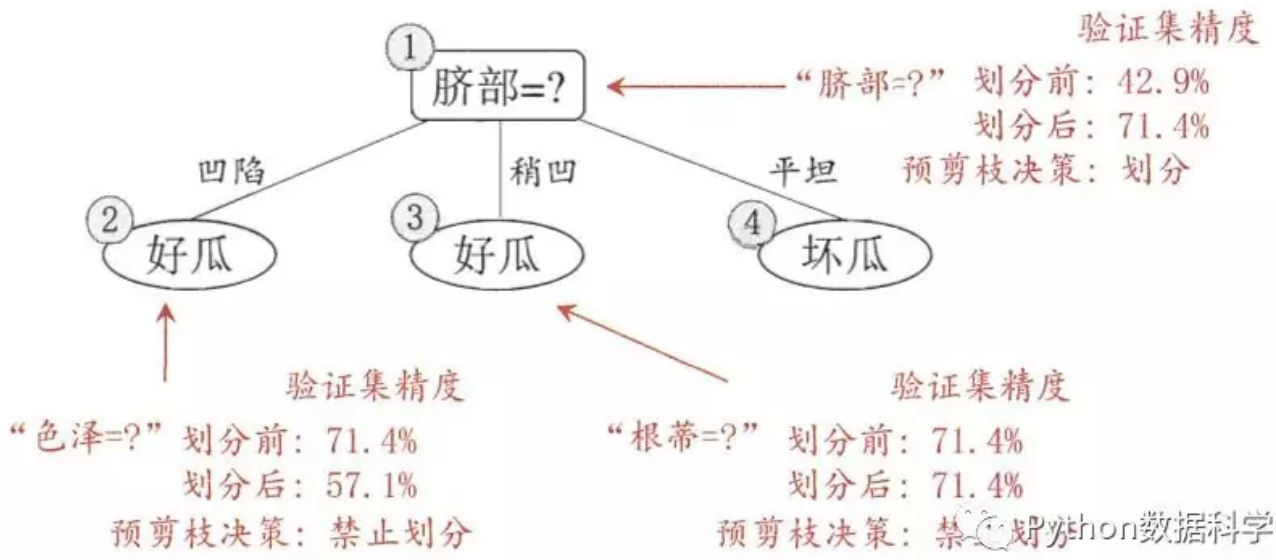
通用的停止其实就是前面递归生成示例中的终止判定条件：

- 如果所有样本均属同一类，终止递归。
- 如果样本的所有的特征值都相同，终止递归。

更严格的终止

- 如果树到达一定高度
- 如果节点下包含的样本点小于指定的阈值
- 如果样本的类分布是独立于可用特征的（使用卡方检验）
- 如果扩展当前节点不会改善信息增益，即信息增益小于指定的阈值

周志华老师的"机器学习"一书中采用对每个节点划分前用**验证集**进行估计，通过比较划分前后的验证集精度来判断是否剪枝。若当前节点的划分不能带来决策树泛化能力的提升，则停止划分并标记当前节点为叶子也点。下图是"周志华机器学习"中的西瓜示例，描述了该方法预剪枝过程。



利用验证集对节点进行评估，如果划分后的正确率比划分前还低，那就禁止划分，如图中的 "色泽" 特征。如果划分后的正确率大于划分前，则同意划分，如图中的 "脐部" 特征。

> 注：很多博客在学习周志华老师的书籍过程中，将预剪枝方法局限于上面这个方法。我个人认为这只是其中的一种，还有很多其它方法可以使用，只要满足这个"预"的含义，都可以算作预剪枝处理。

后剪枝(post-pruning)

与预剪枝不同，后剪枝首先通过完全分裂构造完整的决策树，允许过拟合，然后采取一定的策略来进行剪枝，个人的简单理解就是**"先斩后奏"**。常用的后剪枝策略包括:

- 降低错误剪枝 REP
- 悲观错误剪枝 PEP
- 基于错误剪枝 EBP
- 代价-复杂度剪枝 CCP
- 最小错误剪枝 MEP

比较项目和枝剪方法	CCP	REP	PEP	MEP
独立剪枝集	CV 方式：不需要	需要	不需要	不需要
剪枝方式	自顶向上	自底向上	自顶向下	自底向上
误差估计	使用 CV 或标准误差	利用剪枝集	使用连续性校正	基于 m2 概率估计
计算复杂性	$O(n^2)$	$O(n)$	$O(n)$	$O(n)$

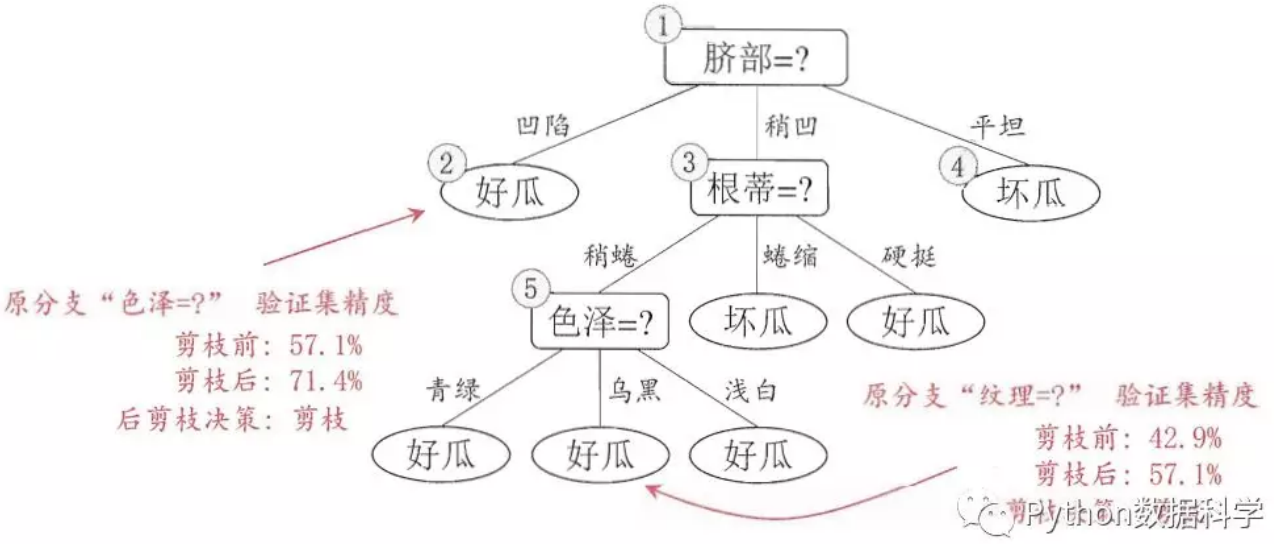
几种后剪枝算法的对比情况

网上大部分博客都是参考周志华老师的**"机器学习"**和李航老师的**"统计学习方法"**来介绍的，并没有从概况上说明属于哪一种。我在本篇对于两本书的方法做个总结。

统计学习方法中的剪枝方法属于CCP，也就是代价-复杂度剪枝方法。就像其他的模型最小化风险结构函数一样，在原有的经验损失函数（经验熵）基础上加入了正则项，正则参数是树的叶子节点个数，公式如下：

$$C_\alpha(T) = C(T) + \alpha|T|$$

机器学习中的剪枝方法属于REP，也就是降低错误剪枝，它是最简单粗暴的一种后剪枝方法，其目的减少误差样本数量。下图是书中西瓜示例的后剪枝过程，通过对比验证集前后的误差精度来判断是否剪枝。



这里仅对这两本书中的方法进行一个总结，其它剪枝方法不在这里展开。如果对这两本书中的剪枝方法感兴趣，建议好好翻一翻，写得很详细。

预剪枝和后剪枝对比

虽然都是剪枝，但预剪枝与后剪枝最终达到的效果是不一样的。了解两种方法并进行比较有助于我们更好地选择。我们来看一下二者的区别：

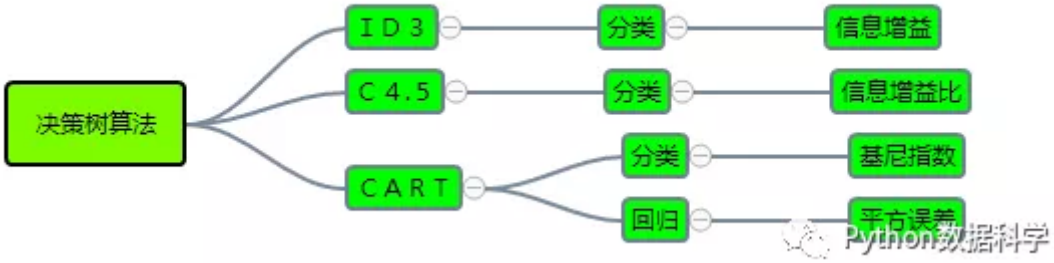
预剪枝：预剪枝提前使很多分支都没有展开，降低了过拟合的风险，但是这个分支下的后续划分可能是非常有用的。从这点考虑，预剪枝是基于”贪心“的本质来禁止分支以及后续的展开，在降低过拟合的同时也有欠拟合的风险。

后剪枝：相比预剪枝，后剪枝的**优点**是：1)后剪枝决策树通常比预剪枝决策树保留了更多的分支；2)后剪枝决策树的欠拟合风险很小，泛化性能往往优于预剪枝决策树。后剪枝的**缺点**是：1)决策树训练时间开销比未剪枝决策树和预剪枝决策树都要大的多。

以上就是决策树中比较关键的三个步骤：**特征选择，树生成，树剪枝。**当然，决策树还有很多其它方面的问题需要考虑，比如连续值处理，缺失值处理，以及如何用于回归等。这些问题我们将通过决策树的三种算法来深入探讨。

决策树算法概况

前面提到的三个步骤其实就基本构成了一个决策树的算法。决策树经典有三种常用的算法有：ID3，C4.5，CART。在对每个算法深入介绍之前，我们先从总体了解一下这几个算法的功能。



每个算法对应着不同的度量准则，其中只有CART算法可以用于回归和分类，分类基于基尼指数，回归基于平方误差最小化。

决策树算法：ID3

ID3算法由Ross Quinlan于1986年提出，它的核心是根据**信息增益**(Information gain)来选取Feature作为决策树分裂的节点。特征对训练数据集的信息增益定义为集合D的经验熵(所谓经验熵，指的是熵是有某个数据集合估计得到的) H(D) 与特征A给定条件下的经验条件熵 H(D|A) 之差，记为：

$$Gain(D, A) = H(D) - H(D|A)$$

实际上就是特征A和D的**互信息**

```
# 统计学习方法：ID3生成决策树算法
输入：训练数据集D，特征集A，阈值e
输出：决策树T
1：若D中所有实例属于同一类Ck，则T为单结点树，并将类Ck作为该结点的类标记，返回T；
2：若A=空，则T为单结点树，将D中实例数最多的类Ck作为结点类标记，返回T；
3：否则，计算A中各特征对D的信息增益，选择信息增益值最大的特征Ag；
4：如果Ag的信息增益小于阈值e，则T为单结点树，将D中最多的类Ck作为结点类标记，返回T；
5：否则，对Ag的每一可能值ai，依Ag=ai将D分割为若干子集Di，将Di中实例数最大多的类作为类标记，构建子
6：对于第i个子结点，以Di为训练集，以A-Ag为特征集，递归调用步骤（1）~（5），得到子树Ti，返回Ti。
```

整个代码部分很长，只显示核心信息增益部分：

```
# 筛选出信息增益最大的特征，返回特征索引,该特征的信息增益
def Maxinformation_gain(data,labels):
    feature_gain ={}
    data_labels = [y[-1] for y in data]
    entropyD = entropy(data_labels)
    # 计算每一个feature的信息增益
    for f in range(len(labels)):
        featureVal = [value[f] for value in data]
        entropyF = ConditionalEntropy(featureVal,data_labels)
        feature_gain[f] = entropyD-entropyF

    result = max(feature_gain.items(),key=lambda x:x[1])
    return result[0],result[1]
```

ID3算法只有树的生成，所以该算法生成的树容易产生过拟合。

决策树算法：C4.5

ID3算法有很多局限性，Quinlan针对这些局限性给出了ID3的一个扩展算法：即**C4.5算法**。C4.5是ID3算法的改进版本，针对四个主要的不足进行改进：

- 不能处理连续特征
- 用信息增益作为标准容易偏向于取值较多的特征
- 不能处理缺失值
- 容易发生过拟合问题

不能处理连续特征：C4.5的思路是将连续的特征离散化，采用**”二分法“**对连续属性进行处理。具体的做法是：将a特征的连续值从小打大进行排列，生成n-1个切分选择，遍历每个切分，根据增益率(信息增益比)选择最优的切分。下图引自机器学习中的内容。

给定训练集D和连续属性a，假定a在D上出现了n个不同的取值，先把这些值从小到大排序，记为$\{a^1, a^2, \ldots, a^n\}$. 基于划分点$t$可将$D$分为子集$D_t^-$和$D_t^+$，其中$D_t^-$是包含那些在属性$a$上取值不大于$t$的样本，$D_t^+$则是包含那些在属性$a$上取值大于$t$的样本。显然，对相邻的属性取值$a^i$与$a^{i+1}$来说，$t$在区间$[a^i, a^{i+1})$中取任意值所产生的划分结果相同。因此，对连续属性$a$，我们可考察包含$n-1$个元素的候选划分点集合

$$T_a = \left\{ \frac{a^i + a^{i+1}}{2} \mid 1 \leq i \leq n-1 \right\}$$

即把区间$[a^i, a^{i+1})$的中位点$\frac{a^i + a^{i+1}}{2}$作为候选划分点。然后，我们就可以像前面处理离散属性值那样来考虑这些划分点，选择最优的划分点进行样本集合的划分，使用的公式如下：

$$Gain(D,a) = \max_{t \in T_a} Gain(D,a,t) = \max_{t \in T_a} \left(Ent(D) - \sum_{\lambda \in \{-,+\}} \frac{|D_t^\lambda|}{|D|} Ent(D_t^\lambda) \right)$$

其中$Gain(D,a,t)$是样本集D基于划分点t二分后的信息增益。划分的时候，选择使$Gain(D,a,t)$最大的划分点。

信息增益作为标准容易偏向于取值较多的特征：引入信息增益比，特征数越多的特征对应的特征熵越大，它作为分母，可以校正信息增益容易偏向于取值较多的特征的问题。

$$Gain\_ratio(D,a) = \frac{Gain(D,A)}{Split\_Information(D,A)}$$

$$Split\_Information(D,A) = -\sum_{i=1}^{n} \frac{|D_i|}{|D|} log_2 \frac{|D_i|}{|D|}$$

具体代码实现如下：

```
def Maxinformation_gain_ratio(data,labels):
    # 计算每个特征的信息增益
    result = {}
    data_labels = [y[-1] for y in data]
    entropyD = entropy(data_labels)
    # 计算每一个feature的信息增益
    for f in range(len(labels)):
        featureVal = [value[f] for value in data]
        entropyF = ConditionalEntropy(featureVal,data_labels)
        feature_gain = entropyD-entropyF
        feature_data_en = FeatureDataEntropy(featureVal)
        result[f] = feature_gain/feature_data_en
    return max(result,key=result.get)[0],max(result,key=result.get)[1]
```

不能处理缺失值：主要需要解决的是两个问题：

1）如何在属性值缺失的情况下进行划分属性选择？

2）给定了划分属性，若样本在该属性上的值缺失，如何对样本进行划分？

引自机器学习的内容，具体解释如下：

给定训练集 D 和属性 a，令$\tilde{D}$表示 D 中在属性 a 上没有缺失值的样本子集（比如，假设 $a=$ 色泽，则$\tilde{D}=\{2,3,4,6,7,8,9,10,11,12,14,15,16,17\}$）。

对于第一个问题，我们可以根据$\tilde{D}$（即在该属性上没有缺失的样本集）来计算属性 a 的信息增益或者其它指标。我们只要再给根据$\tilde{D}$计算出来的值一个权重，就可以表示训练集 D 中属性 a 的优劣。具体来讲，假定属性 a 有 V 个可取值$\{a^1,a^2,\ldots\ldots,a^V\}$，令$\tilde{D}^v$表示$\tilde{D}$中在属性 a 上取值为a^v的样本子集，$\tilde{D}_k$表示$\tilde{D}$中属于第 k 类$(k=1,2,3,\ldots,|\mathcal{Y}|)$的样本子集，则显然有 $\tilde{D}=\bigcup_{k=1}^{|\mathcal{Y}|}\tilde{D}_k$，$\tilde{D}=\bigcup_{v=1}^{V}\tilde{D}^v$。假定我们为每个样本 $\boldsymbol{x}$ 赋予一个权重w_x（**在决策树学习的初始阶段，根节点中各样本的权重初始化为 1**），并定义：

$$\rho=\frac{\sum_{x\in\tilde{D}}w_x}{\sum_{x\in D}w_x}$$

$$\tilde{p}_k=\frac{\sum_{x\in\tilde{D}_k}w_x}{\sum_{x\in\tilde{D}}w_x}\qquad(1\le k\le|\mathcal{Y}|)$$

$$\tilde{r}_v=\frac{\sum_{x\in\tilde{D}^v}w_x}{\sum_{x\in\tilde{D}}w_x}\qquad(1\le v\le V)$$

观察以上公式，能够发现，ρ表示无缺失值样本所占的比例，$\tilde{p}_k$表示无缺失值样本中第 k 类所占的比例，$\tilde{r}_v$表示无缺失值样本中在属性 a 上取值a^v的样本所占的比例。则$\sum_{k=1}^{|\mathcal{Y}|}\tilde{p}_k=1$，$\sum_{v=1}^{V}\tilde{r}_v=1$。

因此，可以把前面用到的信息增益公式改一下：

$$Gain(D,a)=\rho\times Gain(\tilde{D},a)=\rho\times\left(Ent(\tilde{D})-\sum_{v=1}^{V}\tilde{r}_v Ent(\tilde{D}^v)\right)$$

$$Ent(\tilde{D})=-\sum_{k=1}^{|\mathcal{Y}|}\tilde{p}_k\log_2\tilde{p}_k$$

第一个问题根据上面的公式就可以计算出来，**对于第二个问题，若样本 x 在划分属性 a 上的取值未知，则将 x 同时划入所有子节点，只不过此刻要调整该样本 x 的权重值为：$\tilde{r}_v\cdot w_x$。直观的看，其实就是让同一个样本以不同的概率划入到不同的子节点中去。**（这个下面举的例子会更加形象的体现出来是怎么回事）**（C4.5 就是采用上述解决方案）。**

容易发生过拟合问题：C4.5引入了正则化系数进行初步的剪枝。剪枝参考前面解释部分。

总结

本篇介绍了决策树的生成，剪枝两个步骤，然后介绍了前两种算法ID3，C4.5。下一篇将介绍非常经典的**CART算法**，它是集成学习常用的基础算法，可以说是十大经典算法中的一员了，因此