1. K近邻算法和决策树的区别：
2. K近邻算法可以完成多分类任务，但是它最大的缺点是无法给出数据的内在含义，决策树的主要优势就在于数据形式非常容易理解。
3. 决策树的重要任务是：理解数据中所蕴含的知识信息，因此决策树可以使用不熟悉的数据集合，并从中抽取出一系列的规则，这些机器根据数据集创建规则的过程，就是机器学习的过程。

二、决策树的优缺点：

优点：计算复杂度不高，易于理解，对中间值的缺失不敏感，可以处理不相关特征的数据。

缺点：可能会产生过度匹配问题。

使用数据类型：数值型和标称型。

三、 决策树的构造

1. 需要解决的第一个问题就是，当前数据集上哪个特征在划分数据分类时起作用。为了找到决定性的特征，划分出最好的结果，我们必须评估每个特征。

2. 如果某个分支下的数据属于同一类型，则当前无需进一步对数据集进行分割。如果数据子集内的数据不属于同一类型，则需要重复划分子集的过程。

3. 划分数据子集的算法和划分原始数据集的方法相同，直到所有具有相同类型的数据都在一个数据集内。

createBranch()

检测数据集中的每个子项是否属于同一分类：

If so return 类标签

Else

寻找划分数据集的最好特征

划分数据集

创建分支节点

For每个划分的子集

调用函数createBranch并增加返回结果到分支节点中

Return 分支节点

4. 信息增益

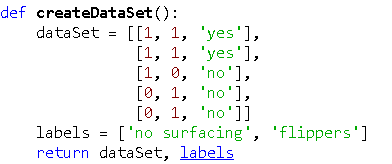
在划分数据集之前之后信息发生的变化称为信息增益，知道如何计算信息增益，就可以计算每个特征值划分数据集获得的信息增益，获得信息增益最高的特征就是最好的选择。

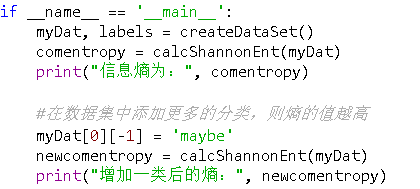
熵定义为信息的期望值，如果待分类的事物可能划分在多个分类中，那么符号的信息定义为：

需要计算所有类别所有可能值包含的信息期望值(n是分类的数目)：

5. 熵越高，则混合的数据也越多。因此可以按照获取最大信息增益的方法划分数据集。







输出结果：



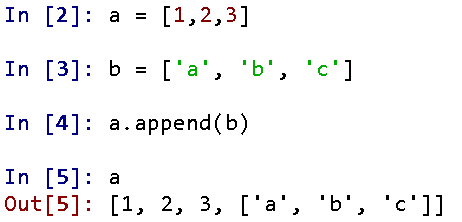
6. 划分数据集

对每个特征划分数据集的结果计算一次信息熵，然后判定哪个特征划分数据集是最好的划分方式。

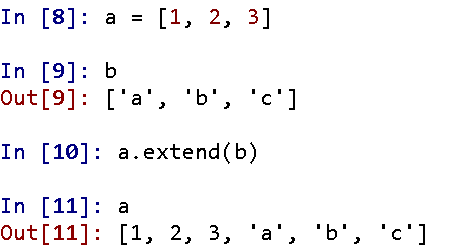
当我们按照某个特征划分数据集时，就需要将所有符合要求的元素抽取出来。

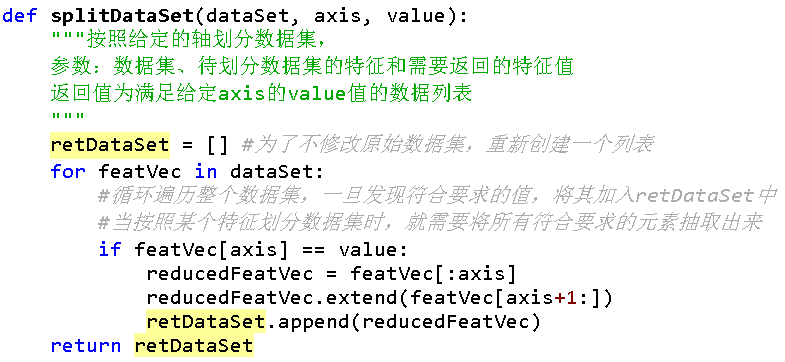
Extend()和append()方法：

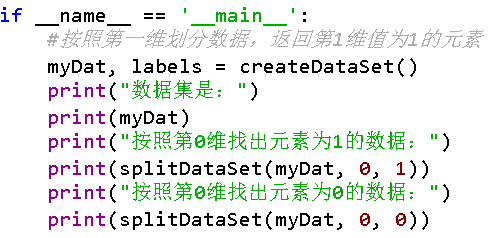
使用append()，当添加一个列表元素时，在列表中也存储为列表。

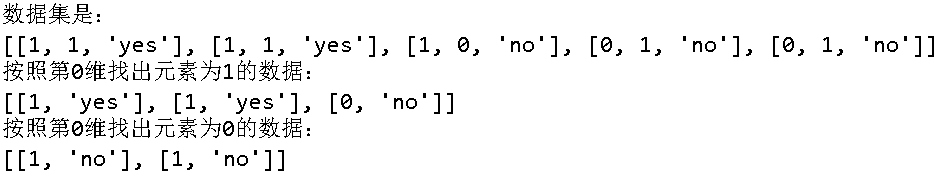


使用extend()，当添加一个列表元素时，将这个列表中的每个元素都加入当前列表。



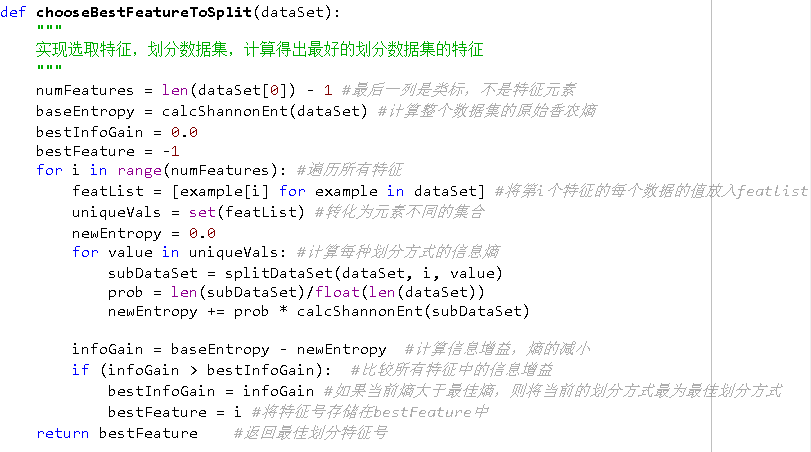


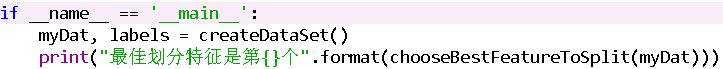




7. 选择最好的数据集划分方式

集合和列表类型相似，不同之处在于集合类型中的每个值都互不相同。



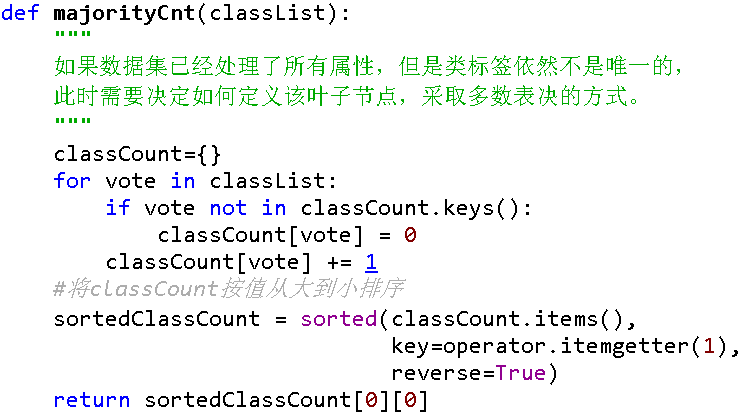




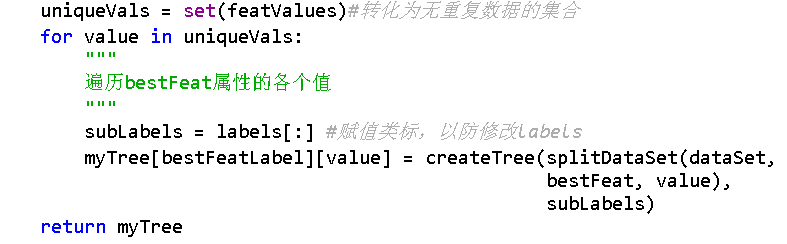
8. 一般而言，信息增益越大，则意味着用该属性进行划分时所获得的“纯度提升”越大。

9. 递归构建决策树

递归结束的条件是：程序遍历完所有划分数据集的属性，或者每个分支下的所有实例都具有相同的分类。任何到达叶子节点的数据必然属于叶子节点的分类。





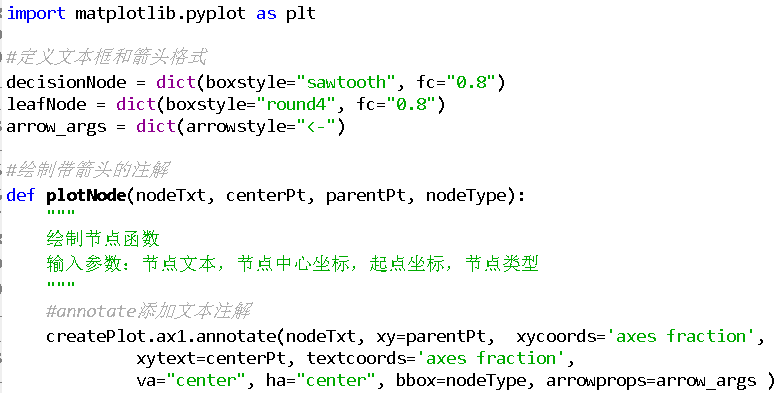




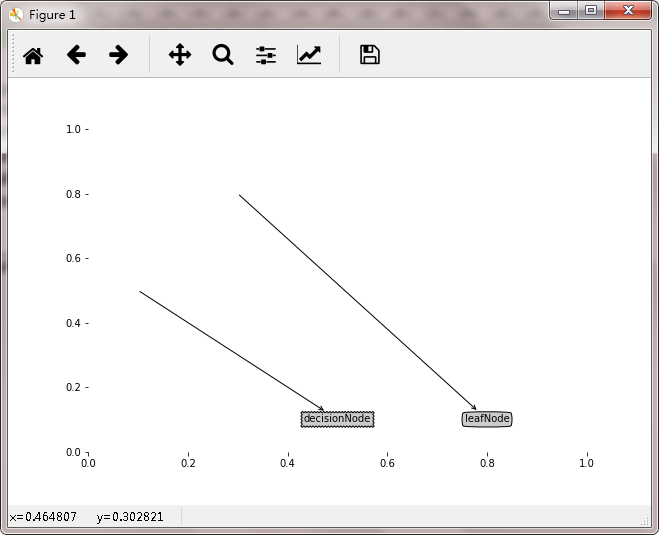
Tree中包含了很多代表树结构信息的嵌套字典，这些值可能是类标签，也可能是另一个字典。如果值是类标签，则该子节点是叶子节点；如果值是另一个数据字典，则子节点是一个判断节点。

四、在Python中使用Matplotlib注解绘制树形图

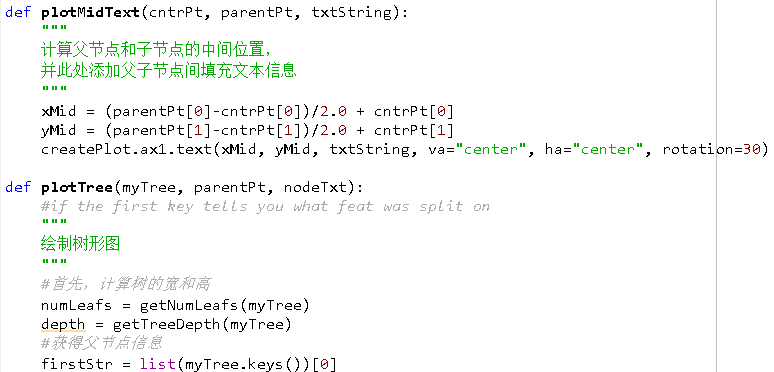
1. 使用Matplotlib提供的注解工具annotations，可以在数据图形上添加文本注解（用于解释数据的内容）。

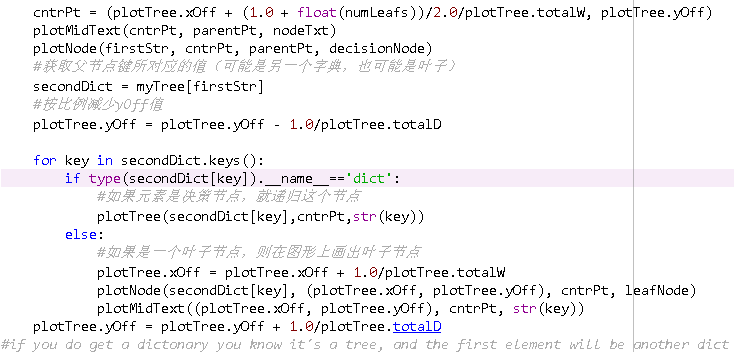




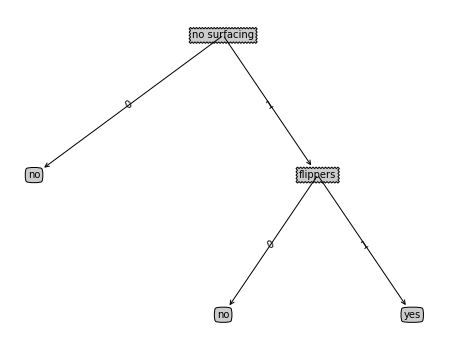


2. 绘制决策树：



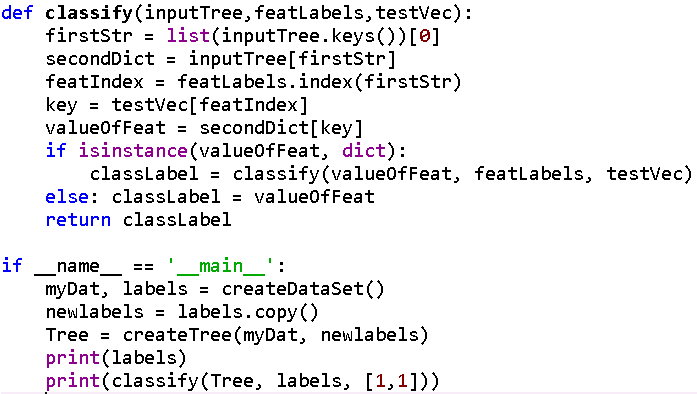






四、 测试算法

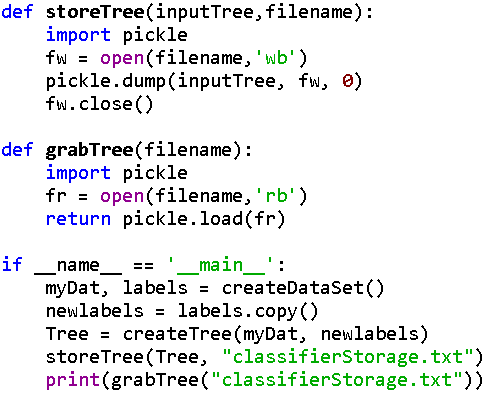
在执行数据分类时，需要使用决策树以及用于构造决策树的标签向量，然后，程序比较测试数据与决策树上的数值，递归执行该过程直到到达叶子节点；最后将测试数据定义为叶子节点所属的类型。





五、 决策树的存储

使用pickle序列化对象，可以在磁盘上保存对象，并在需要的时候读取出来。任何对象都可以执行序列化操作，字典对象也不例外。





可以将分类器存储在硬盘上，而不用每次对数据分类时重新学习一遍。