**中山大学数据科学与计算机学院**

**移动信息工程专业-人工智能**

**本科生实验报告**

**（2017-2018学年秋季学期）**

课程名称：**Artificial Intelligence**

# 实验题目

1. **实现ID3, C4.5, CART三种决策树**

# 实验内容

1. 算法原理
2. **决策树**

决策树可以认为是一种流程图，输入一个测试样例，在每个节点处根据当前样例的某一个属性进行判决，直到最后将样例分到对应的类中。

决策树模型是一种对样例进行分类的树形结构。决策树由结点和有向边组成。结点有两种类型：内部节点和叶节点，内部节点表示一个特征或属性，叶节点表示一个类。

决策树从根到叶的路径代表分类规则。决策树可以被线性化为决策规则，其中结果是叶节点的内容，其决策方式可以简化为if condition1 and condition2 and condition3 and … then outcome，其中的condition就是一个内部节点，outcome就是叶节点。

使用决策树进行分类，其优点在于决策树易于理解，其决策流程和流程图的执行流程相似，且决策树构建也比较简单，同时决策树模型的修改非常方便，因为其有树结构的特性，对于新的属性我们可以很容易的添加对应的内部节点和叶节点。相应的决策树缺点也是很明显的，其构建的时候根据数据集的大小可能会使得构建过程变得非常复杂，同时当决策树过于复杂时，其判决效果可能会下降。

实验中我们需要做的事情就是构建决策树，然后使用决策树对测试样例进行判决。

决策树的构建不是唯一的，最优决策树的构建属于NP问题。所以我们几乎没有办法构建最优决策树，所以就需要利不同的算法选择决策点来进行决策树的构建。

1. **ID3算法**

ID3算法的核心是在决策树各个结点上应用信息增益准则选择特征，递归地构建决策树。

具体方法是：从根结点开始，对结点计算所有可能的特征的信息增益，选择信息增益最大的特征作为结点的特征，由该特征的不同取值建立子结点。再对子结点递归地调用以上方法，构建决策树。直到所有特征的信息增益均很小或没有特征可以选择为止。最后得到一棵决策树。

这个算法中我们就是要实现**信息增益**的计算。

首先是熵：信息熵是信息论中用于度量信息量的一个概念。一个系统越是有序，信息熵就越低。

对于一个可能有n种取值的随机变量：

其熵为：

其中pi表示第i个类别在整个训练元组中出现的概率，可以用属于此类别元素的数量除以训练元组元素总数量作为估计。

X的熵与X的值无关，只与分布有关，所以也可以将X的熵记作H(p):

然后是条件熵：对于随机变量(Y,X)其条件熵为：

根据定义信息增益：

此时我们选择信息增益最大的点作为当前的内部节点。

1. **C4.5算法**

C4.5实际上ID3算法的一种优化，C4.5在生成的树过程中，用信息增益率来选择特征。其在ID3的基础上进行计算信息增益率。

信息增益比：

其中公式分母是特征A的split information也可以看成A的熵:

此时我们选择信息增益率做大的点最为决策点。

1. **CART算法**

分类回归树，实验中我们只需要进行分类。CART本质是对特征空间进行二元划分(即CART生成的决策树是一棵二叉树)，不过本次实验中不要求生成二叉树，实验中给出的要求是多路分裂。

多路分裂和二元分裂的不同主要在于离散值得处理上，对于连续值我们都可以取属性相邻值的平均值，然后使用Gini指数选择一个最优的平均值处理。其主要的不同在于离散值的复杂度上，多路分裂我们可以直接根据离散值中有多少中属性值来确定有多少路进行分裂，也就是说一个属性值对应一个子节点，但是二元分裂就会比较复杂，所有属性值作为一个集合，之后将这个集合分成两部分，遍历所有可能的分成两部分的分法，计算所有分法的Gini指数，选择最优的分类方式作为最终的分裂方式。实验中并没有实现这种二路分裂因为其复杂度是组合数级别的，会造成算法跑不完。

建树的方式和之前的ID3,C4.5一致，也都是选择某个特征值作为节点进行分类，与ID3，C4.5不同的是其使用的是Gini指数作为特征值的选择指标。

Gini指数反映了样本集合的不确定性程度。Gini指数越大，样本集合的不确定性程度越高。

其中：

分类学习过程的本质是样本不确定性程度的减少，所以应选择最小Gini指数的特征分裂。

此处我们为了统一化建树标准，所以此处使用Gini指数的变化量也就是最为标准，所以此时就是选择最大的。

1. **K折交叉验证**

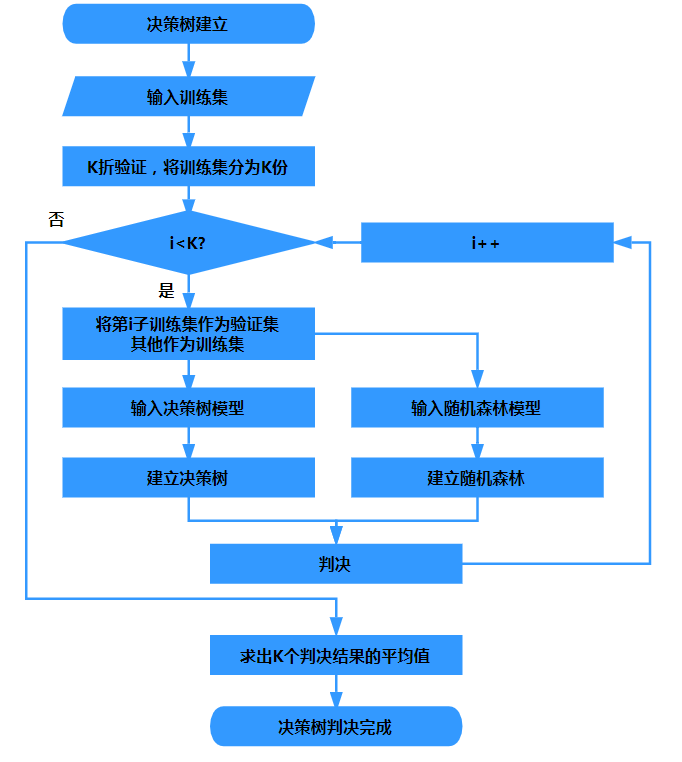
在样本量不充足的情况下，为了充分利用数据集对算法效果进行测试，将数据集随机分为K个子数据集，每次将其中一个子数据集验证集，剩下K-1个子数据集作为训练集进行训练。交叉验证重复K次，每份验证一次，平均K次作为最终的结果。

这个方法的优势在于，同时重复运用随机产生的子样本进行训练和验证，得到可靠稳定的模型。

剪枝以及随机森林见拓展部分

1. 伪代码&流程图
   1. 总流程

此处包括了K折交叉验证(随机的分成K份)。



以下三种节点选择方法中的计算可以直接从数据集中获取相应的P(i)，参见原理中的公式

* 1. ID3

|  |
| --- |
| **In:**经验熵  **for** i=0 to 总特征数  特征值i的条件熵=0  **for** j=0 to 第i个特征所有值数  特征值i的条件熵+=特征值i的第j个值的条件熵  第i个特征的信息增益=经验熵-特征值i的条件熵  **return**所有特征值的信息增益 |

* 1. C4.5

|  |
| --- |
| **In:** 所有特征值的信息增益  **for** i=0 to 总特征数  特征值i的SplitInfo=0  **for** j=0 to 第j个特征所有值数  特征值i的SplitInfo+= p(j)\*log2(p(j))  第i个特征的信息增益率=第i个特征的信息增益/特征值i的SplitInfo  **return**所有特征值的信息增益率 |

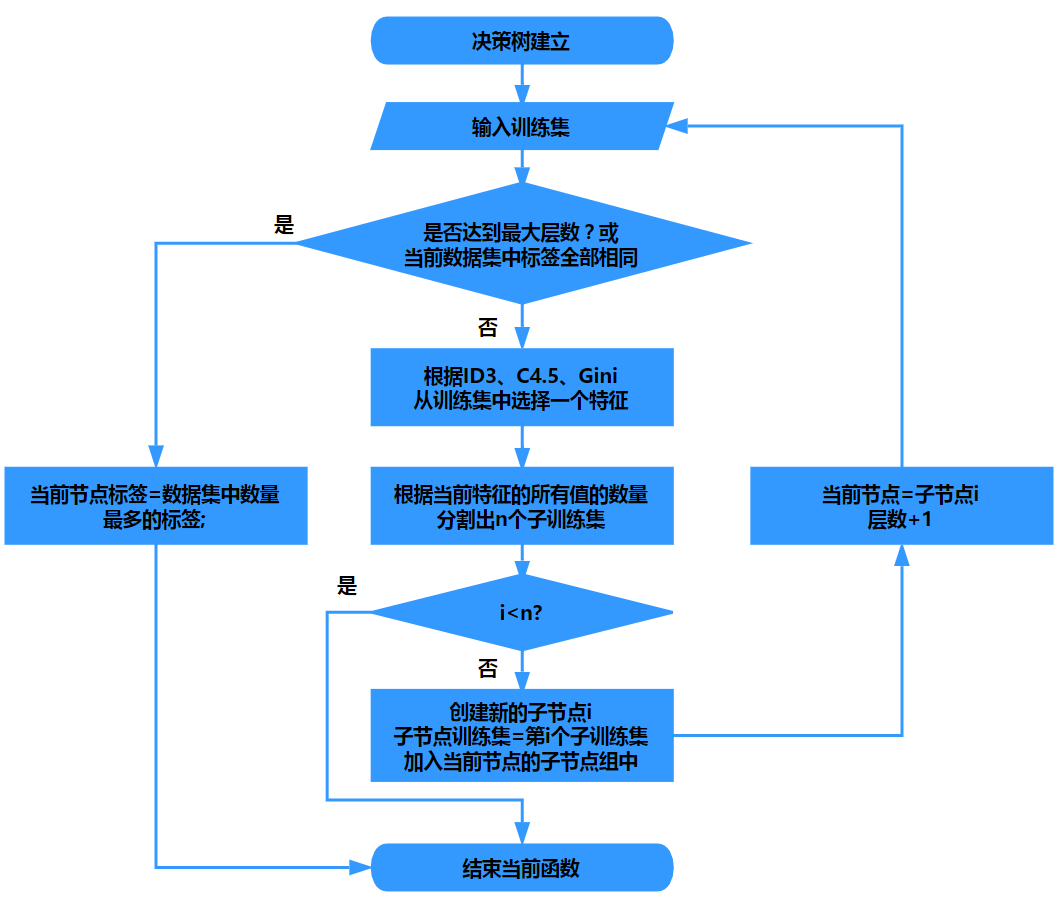
* 1. Gini

|  |
| --- |
| 计算原始Gini指数  **for** i=0 to 总特征数  特征值i的Gini指数=0  **for** j=0 to 第j个特征所有值数  特征值i的Gini指数+=特征值i的第j个值的Gini指数  △特征值i的Gini指数=原始Gini指数-特征值i的Gini指数  **return** △Gini指数 |

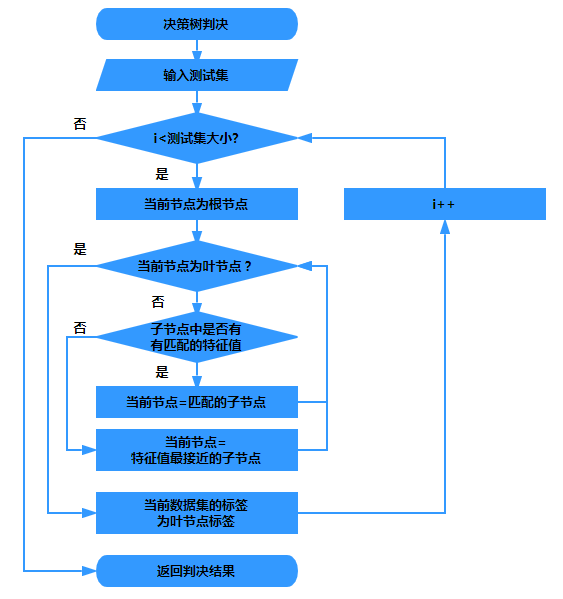
* 1. 决策树的建立

这里主要就是递归终止的条件

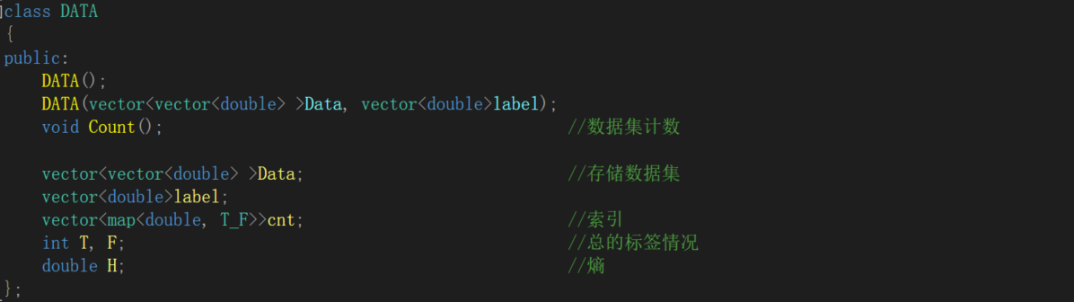
|  |
| --- |
| Build\_Tree(Train\_Data){  **if**(到达递归层数||当前数据集中的标签全部一致){  当前节点为叶节点;  当前节点标签=数据集中数量最多的标签;  **return**;  }  根据ID3、C45、Gini选择一列特征值;  根据特征值的所有值划分子数据集;  **for** i=0 to 子数据集数{  新建一个子节点;  子节点数据集=第i个子数据集;  子节点加入当前节点的子节点集中;  当前节点=子节点;  **if**(第i个子数据集为空){  子节点为叶节点;  子节点标签=父数据集中数量最多的标签;  }  **else**{  递归层数+1;  Build\_Tree(第i个子数据集);  递归层数-1;  }  }  } |



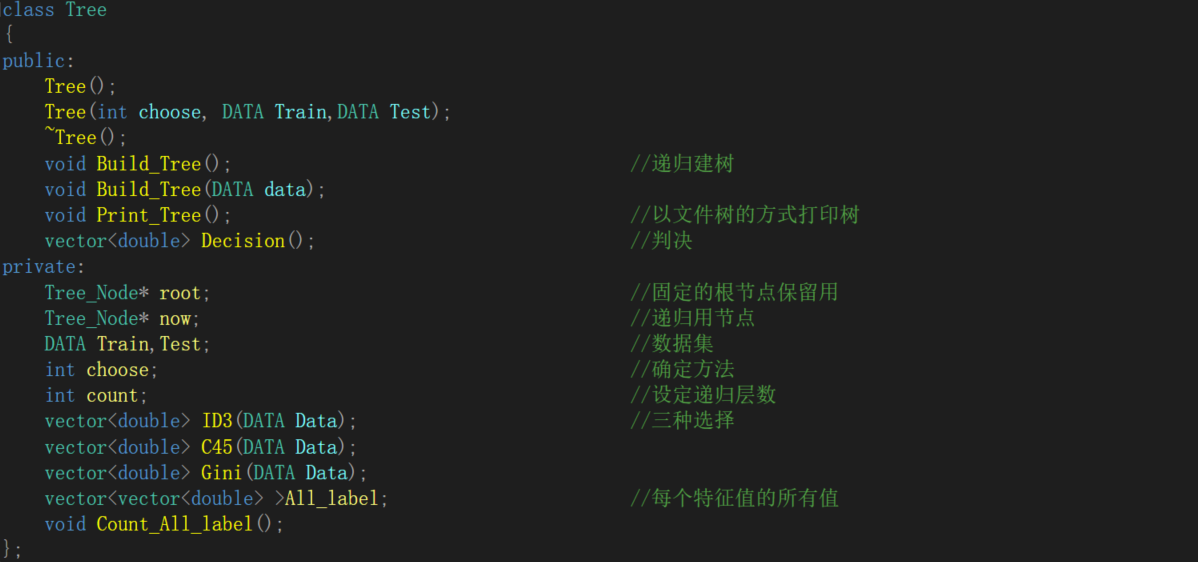
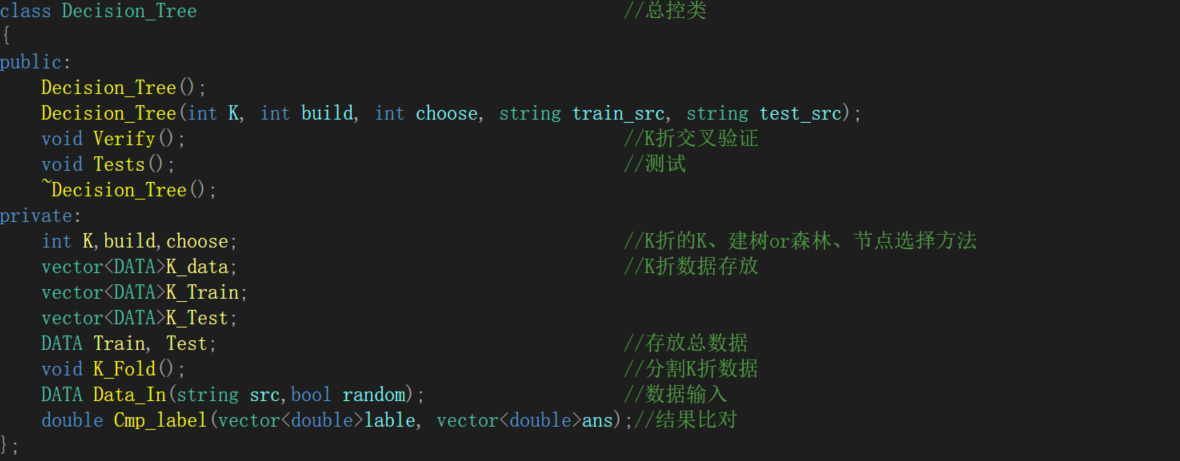
* 1. 决策树的判决



1. 关键代码截图（带注释）
   1. 数据集统计部分ID3,C4.5,Gini计算时需要的各项数据的准备。



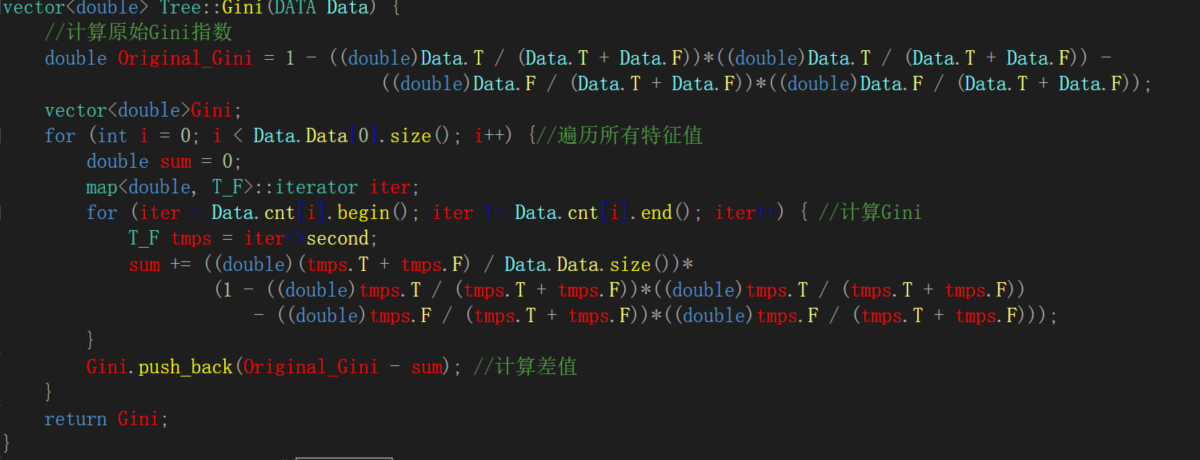
* 1. 建树部分 此处的K折交叉验证代码比较简单不展示



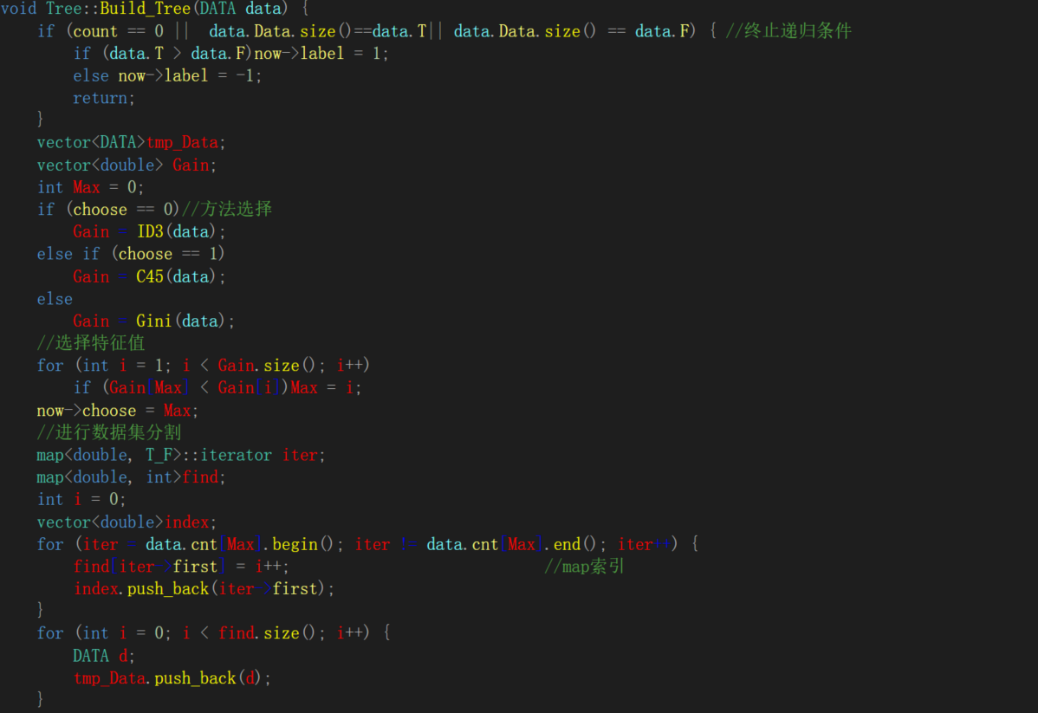
1. **ID3**
2. **C4.5**



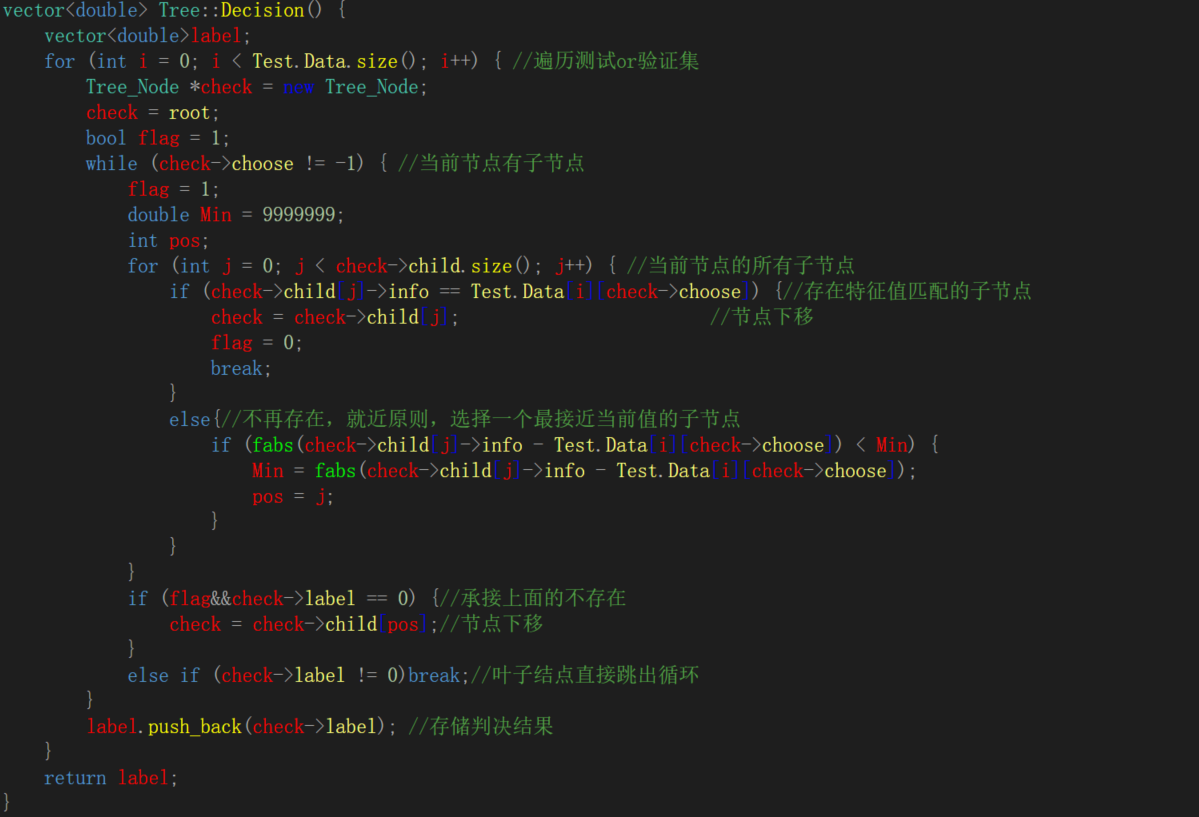
1. **Gini**



1. **递归建树**



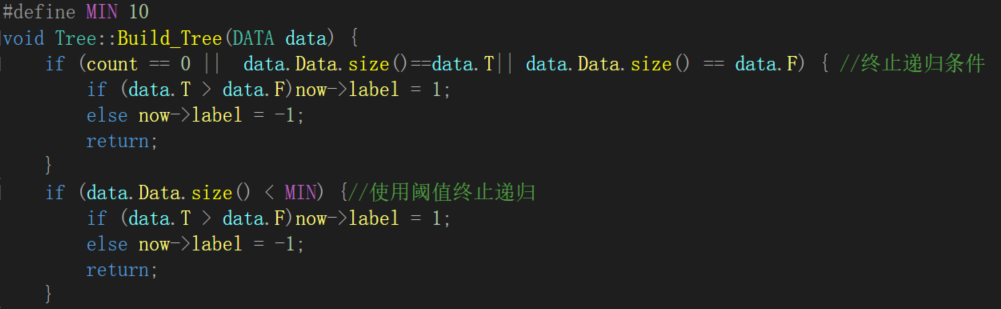
* 1. 决策树判决



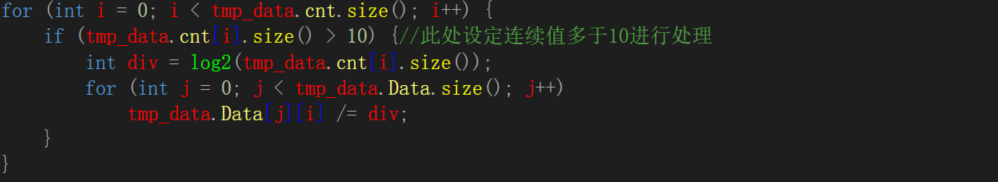
1. 创新点&优化
2. **预剪枝**

主要是通过提前停止树的构建而对树剪枝，主要目的是为了防止决策树过拟合。此处实现了3中预剪枝方法：

* 定义一个高度，当决策树达到该高度时就停止决策树的生长，参见之前建树部分的的结束递归条件中的count，通过调节count我们可以控制决策树的高度。(此段代码在提交版本中为放任生长)
* 定义一个阈值，当达到某个节点的数据集个数小于阈值时就可以停止决策树的生长。(此段代码在提交版本中注释，因为后期实现的随机森林不需要剪枝)



* 再输入时对连续值进行划分，将其划分成多个区间，从而减少子节点数量。此处通过使用除法取整实现区间。区间个数使用log实现。(此段代码位于输入处理函数Data\_In中)。



1. **后剪枝**

后剪枝也是为了防止数据过拟合。此处使用误差降低剪枝(Reduced Error Pruning)，过程就是去掉这个节点，使其称为叶节点看正确率是否提升，如果没有提升就还原该节点。

|  |
| --- |
| REP(){  **if**(叶节点) **return**;  创建一个临时节点;  将当前节点所有属性赋予临时节点;  清空当前节点所有属性,使得当前节点成为叶节点;  设置当前节点的标签为其数据集中最多的标签;    调用判决函数检测正确率;  **if**(正确率>原正确率){  正式删除当前节点;  }  **else** {  用临时节点还原当前节点;  **for** i=0 to 当前节点的子节点数{  当前节点=第i个子节点;  REP();  }  }  } |

此处代码因为之后的随机森林而弃用。

1. **随机森林**

随机森林，指的是利用多棵树对样本进行训练并预测的一种分类器。本质上就是可用通过之前的建树算法进行构造随机森林，也就是构造多棵决策树，然后通过多棵决策树进行投票表决。

随机森林的实现主要在于随机。首先假设训练集大小为N，随机且有放回地从训练集中的抽取N个训练样本；然后假设每个样本的特征维度为M，指定一个常数m<<M，随机地从M个特征中选取m个特征子集。这样就可以组成一个N个样本m个特征的训练集，然后我们使用这个训练集进行树的构建。重复多次我们得到一片决策森林，之后使用这片森林进行判决。同时一点，每棵树都尽最大程度的生长，并且没有剪枝过程(所以之前的剪枝基本作废，为了代码兼容)。

森林建造如下：



判决时只需要调用单棵决策树的判决函数然后进行投票即可。

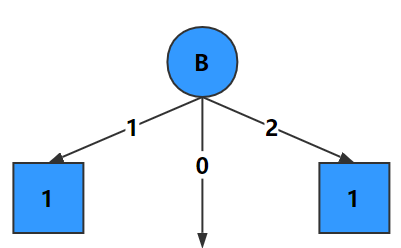
# 实验结果及分析

1. 实验结果展示示例

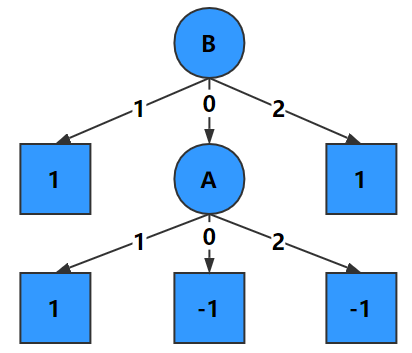
小数据测试，此处不使用K折交叉验证，只是普通的进行决策树的建立，节点选择方法为ID3，主要目的为验证建树结果的正确性。训练集如下

|  |  |  |  |
| --- | --- | --- | --- |
| A(日志密度) | B(好友密度) | C(是否真实头像) | label |
| **0** | **0** | **0** | **-1** |
| **0** | **2** | **1** | **1** |
| **2** | **1** | **1** | **1** |
| **1** | **1** | **1** | **1** |
| **2** | **1** | **1** | **1** |
| **1** | **2** | **0** | **1** |
| **1** | **0** | **0** | **-1** |
| **2** | **1** | **0** | **1** |
| **1** | **0** | **0** | **1** |
| **0** | **0** | **1** | **-1** |

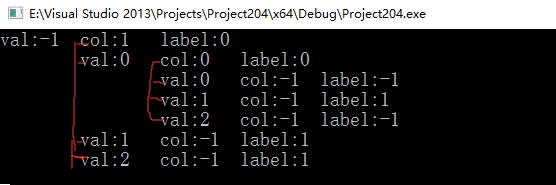
此处选用SNS社区中不真实账号检测的一个例子，其中0、1和2分别表示密度小、中和大。

* 根节点选取：
  + 此时G(D|B)最大。
  + 所以选择B为根节点。
  + 

|  |  |  |
| --- | --- | --- |
| A | C | label |
| **0** | **0** | **-1** |
| **1** | **0** | **1** |
| **1** | **0** | **-1** |
| **0** | **1** | **-1** |

* 计算过程省略直接给出选择节点B值为0是第二次选择A节点作为子节点，此时A=1时C在分割不会降低熵，所以不必再分，直接取1。此时A=2的值不存在所有A=2的节点为父节点中最多的类。
* 

代码运行结果(仿文件树)



实验中主要验证树的建立是否正确。

测试一下

|  |  |  |  |
| --- | --- | --- | --- |
| A(日志密度) | B(好友密度) | C(是否真实头像) | label |
| **1** | **2** | **1** |  |
| **0** | **0** | **1** |
| **2** | **0** | **1** |

1. **评测指标展示即分析**
   1. **决策树无任何优化，使用10折验证**

|  |  |  |
| --- | --- | --- |
|  |  |  |
| ID3 | C4.5 | CART |

* 1. **决策树使用剪枝，使用10折验证**

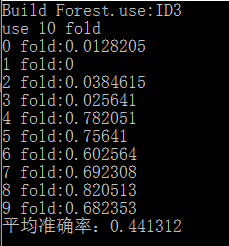
|  |  |  |
| --- | --- | --- |
|  |  |  |
| ID3 | C4.5 | CART |

总结一下剪枝这个相对于没有剪枝只有略微提升，也有可能不会提升，主要原因我认为主要是使用的剪枝方法并不是和高级，像预剪枝这类完全是在砍树进行大范围的控制树的形状，并不是针对每一个节点操作的，所以没什么正确率的提升也正常，主要还是要防止过拟合。

* 1. **随机森林，使用10折验证**

|  |  |  |
| --- | --- | --- |
|  |  |  |
| ID3 | C4.5 | CART |

随机森林在这个训练集上看起效果平平，但是实际使用中，随机森林每次运行的时候准确率波动更小，因为使用了K折验证随机划分训练集，所以还是会有小幅的波动。其实可也以看出决策树的效果训练集的还是有很大关系的，这也造成了决策树抗噪性不足，但是随机森林的波动情况来看其增强了抗噪性。

实际上使用随机森林的时候发现这个训练集有点奇怪，在不打乱训练集(按顺序而不是随机)的情况下我们会出现如右图结果，前几折的交叉验证效果很差，后面有出奇的好，后来测试了一下发现，这个训练集，前面300多个和后面400多个好像不是一个数据集中取出的，主要问题出现在这个数据，前面全是1后面全是-1。如果有前面或者后面的分别训练，发现他们自己的训练效果都很好，甚至能在分出来的验证集上跑出1。

在这种训练集情况下使用K折交叉验证的重要性就比较明显，如果只是单纯的划分很容易陷入过拟合。

# 思考题

* + - 1. **决策树有哪些避免过拟合的方法？**

过拟合主要有两种原因，一是样本问题，样本中噪音数据干扰过大或者是样抽样错误、样本数量不足。二是构建决策树的方法问题，在构建决策树的时候我们对于决策树的生长没有合理限制的时候，此时决策树可以做到完美拟合训练数据，但是到了测试数据的时候结果就会变差。

针对样本原因，我们可以合理、有效地抽样，例如使用交叉验证的方式，进行抽样验证。

针对决策树构建的原因，主要是使用剪枝，可以是限制树的生长(预剪枝)，或者是对已生成的决策树按一定规则进行修剪(后剪枝)。

* + - 1. **C4.5相比于ID3的优点是什么？**

用信息增益率来选择属性，克服了用信息增益来选择属性时偏向选择值多的属性的问题，在有些情况下这类属性可能不会提供太多有价值的信息。这是两种最本质的区别，也是C4.5优于ID3的最根本地方。

当然在看算法原理的时候发现C4.5的应该还包含了剪枝和连续数据处理等，但是这些同样可以修改已改一下应用到原始ID3上，所以不能说是优点了。

* + - 1. **如何用决策树来判断特征的重要性？**

根据我们选择节点方式来看，越重要的特征会被越先选择到，也就是越重要的特征，离根节点越近，根结点的特征即是最重要的。