# 人工智能-第3次课程作业报告

授课教师：张宇 作者：蒋雨初-58121102

## 1 问题描述

### 1.1题目介绍

请利用决策树判断是否需要在餐馆等座位，本问题主要基于如下属性列表基础上决策的。

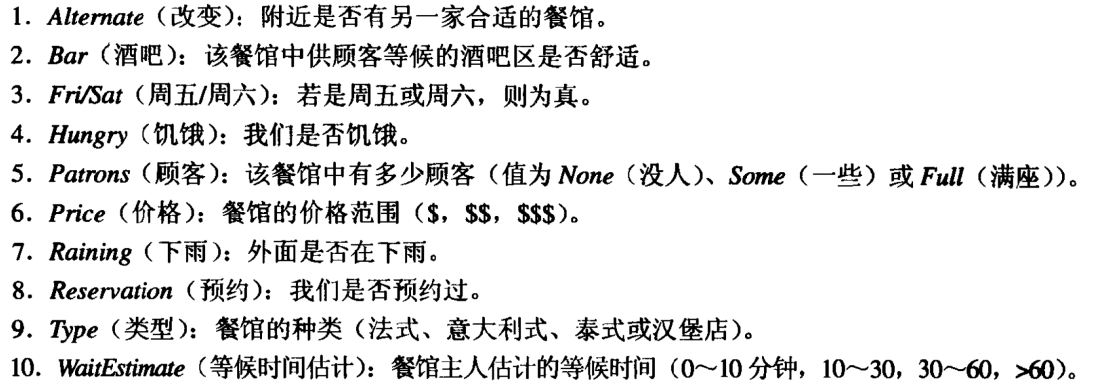


图1 餐馆等座位问题属性列表

### 1.2任务说明

restaurant\_willwait数据集共有100条数据，按照3:1划分比例分为训练集和测试集，现提供训练集数据，即75条数据。保存路径为“Assignment3\_Refactor\dataset”大家可根据需要自行分为训练集和测试集。为了方便程序编写，数据集属性值取值做了如下对应改写：

表1 属性值离散表示

|  |  |
| --- | --- |
| 属性 | 离散取值 |
| Alternate | 1 0 |
| Bar | 1 0 |
| Fri/sat | 1 0 |
| Hungry | 1 0 |
| Patrons | 0 0.5 1 |
| Price | 0 0.5 1 |
| Raining | 1 0 |
| Reservation | 1 0 |
| Type | 1 0.67 0.33 0 |
| Waitestimate | 0 0.17 0.5 1 |

表2 是否等待离散表示

|  |  |
| --- | --- |
| 属性 | 离散取值 |
| willwait | 1 0 |

### 1.3实验环境

### 设备规格

设备名称 LAPTOP-TFMBQKQ8

处理器 AMD Ryzen 7 4800H with Radeon Graphics 2.90 GHz

机带 RAM 16.0 GB (15.4 GB 可用)

设备 ID 6CC35513-8821-49C9-A60B-C44F31F302A7

产品 ID 00342-35891-56086-AAOEM

系统类型 64 位操作系统, 基于 x64 的处理器

笔和触控 没有可用于此显示器的笔或触控输入

### 系统规格

版本 Windows 11 家庭中文版

版本 22H2

安装日期 ‎2022/‎9/‎28

操作系统版本 22621.521

体验 Windows Feature Experience Pack 1000.22634.1000.0

### 开发环境

Microsoft Visual Studio Professional 2022

Version 17.2.4

VisualStudio.17.Release/17.2.4+32602.215

Microsoft .NET Framework

Version 4.8.09032

Installed Version: Professional

Visual C++ 2022 00483-00000-00004-AA929

Microsoft Visual C++ 2022

### 语言标准

C++20 or latest.

### 1.4评价标准

最终评测将在测试集上采用**正确率(A, Accuracy)**指标进行评测。

正确率与得分情况关系如下表所示：

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 正确率 | 60%<70% | 70%A<80% | 80%A<90% | A90% |
| 分数 | 2 | 4 | 6 | 8 |

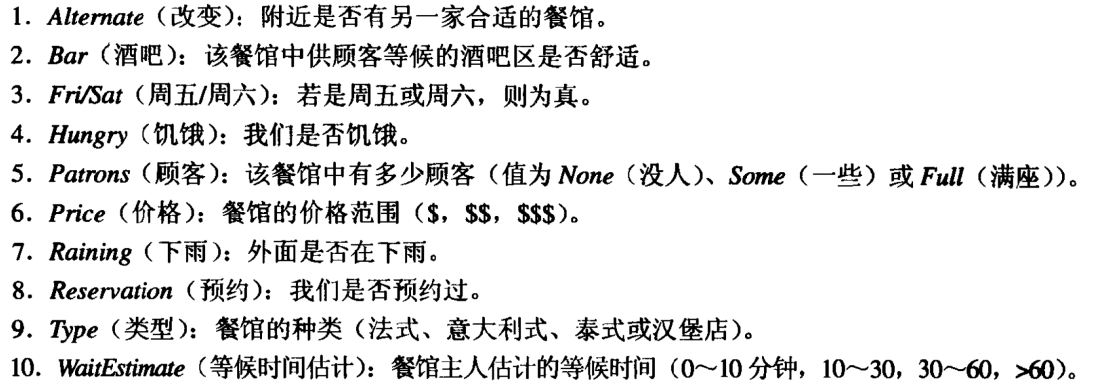
## 2 实验方案

### 2.1 决策树

决策树表示一个函数，以属性值向量作为输入，返回一个“决策”一一简单输出值。输入值和输出值即可以是离散的，也可以是连续的。此时此刻，我们将聚焦于输入值是离散的和输出值为二值的情况。这是布尔分类，其中样例输入被分类为真（正例）或假（反例）。

决策树通过执行一系列测试达到决策。树中内部结点代表对输入属性之值的一个测试，从结点射出的分支用属性可能值标识。树中叶结点指定函数的一个返回值。

作为一个例子，我们将构造一个决定在饭店中是否等待餐桌的决策树，目的是学习目标谓词WillWait的定义。首先列出输入属性：

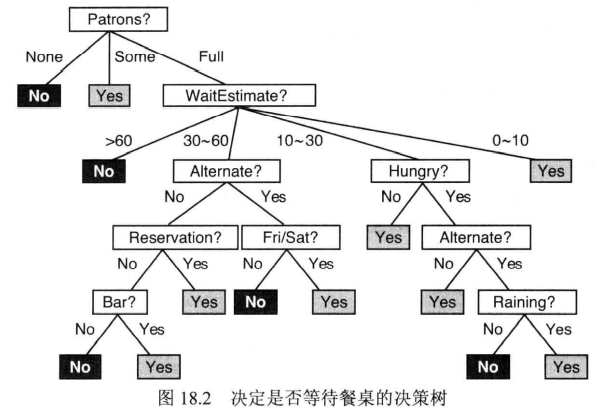


如图18.2是手动画出的决策树。

### 2.2 从样例归纳决策树

作为一个例子，考虑由对组成的布尔决策树，其中是输入属性的值向量，y是单一布尔输出值。12个训练样例显示在图18.3中，其中目标WillWait为真的是正例()，为假的是反例（）。

所需之决策树当与样例一致，且规模尽可能小。不幸的是，无论用何种方法度量规模寻找极小一致树都是一个难以对付的问题，在棵树中进行搜索，不存在高效方法。然而，利用简单启发式能够发现良好的近似解：小规模的一致树。DECISION-TREE-LEARNING算法采取贪婪“分化-征服”（divide-and-conquer）策略：总是优先测试最重要属性。测试将问题分解为更小的子问题，这些子问题又可被递归求解。“最重要属性”意指对于样例分类具有最大差异的属性。如此做的目的是希望通过较少测试达到正确分类，即树中所有路径都较短，整个树较浅。



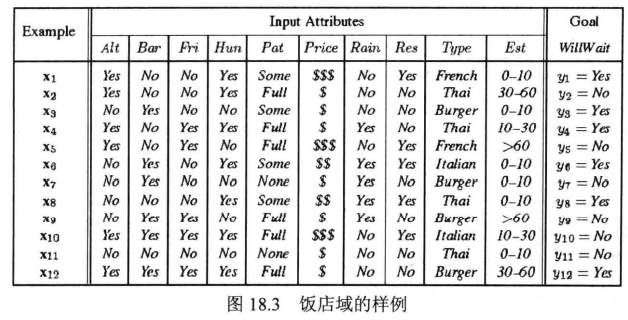


图18.4（a）显示*Type*是弱分类属性，因为它产生的四个可能输出的每一个都包含同等数目的正反样例。另一方面，在（b）中，*Patrons*是一个极其重要的属性，因为当其值为*None*或*Some*时，留下的样例集可以得到肯定的回答（分别为*No*和*Yes*）。如果其值是*Full*，留下一个混合样例集。一般来说，当第一个属性测试将样例集分裂后，每一个结果本身又是新的决策树学习问题，只不过其样例稍少，且属性减少了一个。这种递归问题要考虑四种情况：

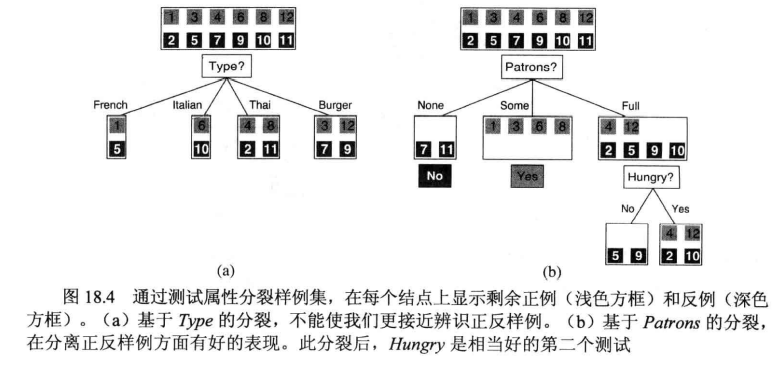
（1）如果剩余样例都是正例（或反例），则事情完毕，可回答*Yes*或*No*。图18.4（b）显示了发生在*None*和*Some*分支上的这种情形。

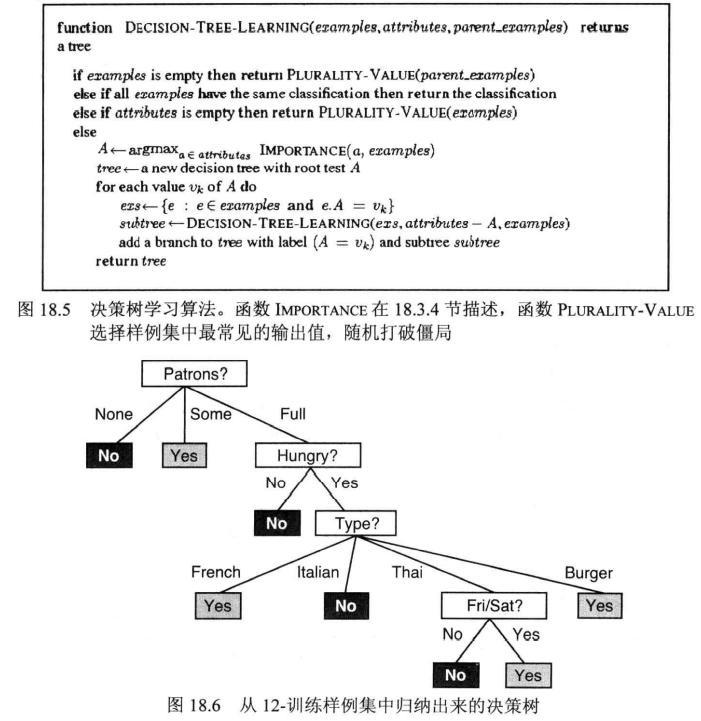
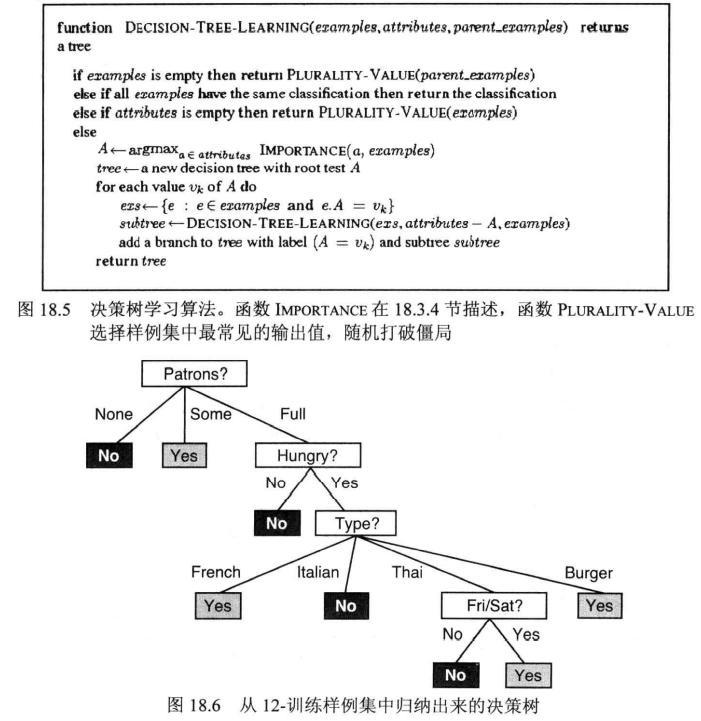
（2）如果既有正例又有反例，则选择最好属性继续分裂之。图18.4（b）显示了用*Hungry*分裂剩余样例的情况。

（3）如果没有留下任何样例，则表明对于这个属性值组合，没有观察到样例。此种情况下返回一个缺省值，该值是构造其父结点用到的所有样例中得票最多的分类。它们通过变量*parent\_examples*传递。

（4）如果留下的样例既有正例又有反例，而没有属性可用，这意味着这些样例的描述

相同而分类相异。这种情况的发生是由于数据中存在错误或噪音，或由于领域是非确定性的，或由于没有观察到能够区分样例的属性。此时最好的方法是返回剩余样例中得票最多的分类。



DECISION-TREE-LEARNING算法在图18.5中给出。注意，样例集对于树的构造是至关重要的，然而样例本身并不在树中出现。树仅由内结点上的属性测试、分支上的属性值和叶结点上的输出值组成。函数IMPORTANCE的细节在18.3.4中描述。对于图18.3所示的样本训练集，学习算法的输出显示在图18.6中，此树显然不同于在图18.2所示的原树。鉴于此，也许有人认为学习算法在学习正确函数方面做得并不好，但这个推论是错误的。学习算法探究的是examples而不是正确函数。事实上它的假说（见图18.6）不仅仅与所有样例一致，而且比原树简单多了！学习算法没有理由考虑对Raining和Reservation进行测试，原因是没有它们也能分类所有的样例。当给定更多训练样例时，学习程序会纠正该错误。

### 2.3 选择测试属性

决策树学习的搜索策略被设计成贪婪搜索策略，该策略近似于极小化最终树的深度。主要思想是挑选一个属性，以便尽可能对样例进行确切分类。一个完美属性将样例划分为每一个都是正例或者反例的集合，这些集合对应树的叶结点。Patrons属性不是理想的，但相当好。真正无用的属性是那些分裂出的样例集中正反样例的比例大约与原集合中的比例相同的属性，如Type。

现在需要形式化度量“相当好”和“真正无用”，并实现图18.5中的函数IMPORTANCE。

我们将使用信息收益的概念，并用熵的术语定义之。熵是信息论中的基本量（Shannon和Weaver，1949)。

熵是随机变量的不确定性度量，信息的获取对应于熵的减少。

一般地，设随机变量具有值，的概率为，则的熵定义为：

现返回决策树学习。如果训练集包含p个正例和n个反例，则目标属性在整个样例集上的熵是

图18.3中的饭店训练集有，因此对应的熵是，或确切为1比特。单一属性A上的测试也许只给出这个1比特的一部分，通过考虑属性测试后的剩余，可以测量这部分熵是多少。

带有d个不同值的属性A将训练集E划分为子集，每个子集有个正例和个反例。如果沿着该分支前进，需要额外的比特信息来回答问题。设从训练集中随机挑选的一个样例以概率具有属性的第k个值，则在测试属性A之后，剩余的期望熵是：

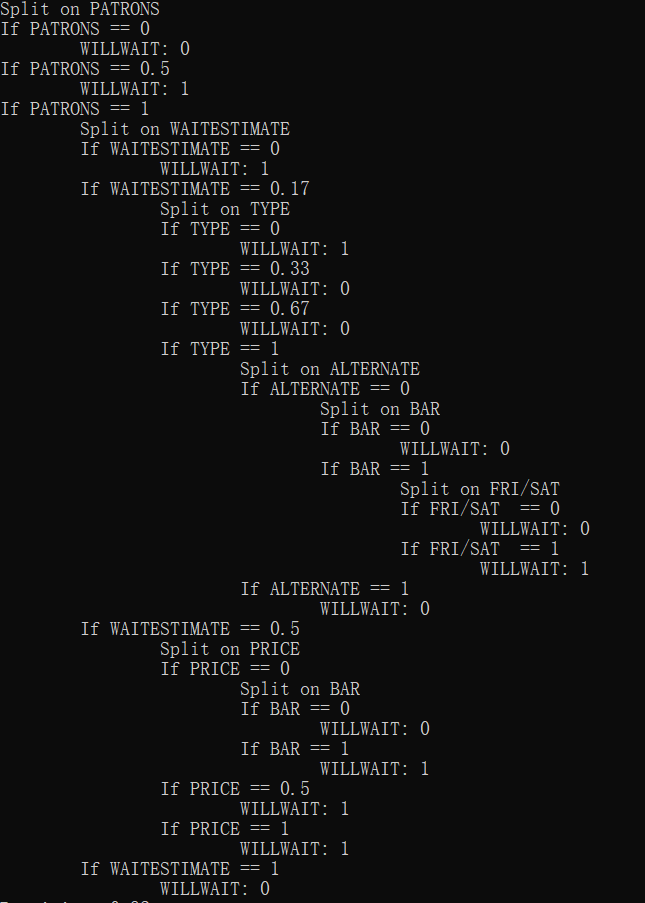
从对属性A的测试获得的信息收益是熵的期望减少：

事实上，正是实现函数IMPORTANCE所需要的。回到图18.4中考虑的属性，有

这进一步肯定了我们的直觉：Patrons是用于分裂的更好属性。事实上Patrons在所考虑的属性中具有最大的信息收益，因此被决策树学习算法选作根结点。

## 3实验结果

最终得到了如图所示的决策树。



在测试集上，准确率为0.92。

## 4实验分析

接下来讨论决策树的优化策略。

### 4.1 信息增益比

在上面的介绍中我们有意忽略编号一列。若把“编号”也作为一个候选划分属性，则可计算出它的信息增益为0.673，大于其他候选划分属性.这很容易理解：“编号”将产生10个分支，每个分支结点仅包含一个样本，这些分支结点的纯度已达最大。然而，这样的决策树显然不具有泛化能力，无法对新样本进行有效预测。

实际上，信息增益准则对可取值数目较多的属性有所偏好。这是因为：信息增益表示一个属性对当前熵（即不确定度）的减少能力。取值数目越多，则被分类后的样本数越少，也就是对熵的减少能力越强（在编号的例子中，直接使得Remainder变为0），此时信息增益很大，根据信息增益准则会优先选择之。然而这种决策树的泛化能力是很差的。所以为减少这种偏好可能带来的不利影响，著名的C4.5决策树算法[Quinlan，1993]不直接使用信息增益，而是使用“增益率”（gain ratio）来选择最优划分属性。增益率定义为

其中

称为属性A的“固有值”（intrinsic value）[Quinlan，1993]．属性A的可能取值数目越多（即d越大)，则的值通常会越大.

需注意的是，增益率准则对可取值数目较少的属性有所偏好，因此，C4.5算法并不是直接选择增益率最大的候选划分属性，而是使用了一个启发式[Quinlan，1993]：先从候选划分属性中找出信息增益高于平均水平的属性，再从中选择增益率最高的。

### 4.2 剪枝

对于某些问题，当不存在欲寻找的模式时，DECISION-TREE-LEARNING算法将生成一颗庞大的树。现考虑预测骰子的滚动是否出现6的问题。假设用各种骰子做实验，描述训练样例的属性包括骰子的颜色、重量、滚动时间和实验者的手指是否交叉。如果骰子是公正的，能够学到的正确东西应该是一个带单一结点的树，该结点说“no”。但是DECISION-TREE-LEARNING算法将抓住在输入数据中能够发现的任何模式。如果以手指交叉方式投掷7克重的蓝色骰子两次，两次滚动都出现6，则算法也许会构造一条路径，它在此情况下预测为6。这个问题称之为**过度拟合**。当假说空间和属性数目增长时，过度拟合更可能出现，而随着训练样例的增加，过度拟合的可能性逐步降低。

对于决策树，一种称为**决策树剪枝**的技术将减轻过度拟合。通过删除不明显相关的结点来实现剪枝。它开始于DECISION-TREE-LEARNING生成的一棵完全树，然后考查只有叶结点作为后代的测试结点。如果测试结点是不相关的一一仅侦知数据中的噪音一一则删除该测试，用叶结点替代它。在所有只有叶结点作为后代的测试上重复此过程，直到每个这样的测试或被剪枝或保持不变。

怎样发现一个结点正在测试不相关的属性？假设现正处于有p个正例和n个反例组成的结点上。如果属性是不相关的，可以预期它将样例分裂为多个子集，其中每一个所含正例的比例与整个集合的比例大致相同，为，此时信息收益接近零。因此信息收益对于不相关性来说是一个好线索。现在的问题是，为了确定用于分裂的特定属性，所需收益应该是多大？

使用统计重要性测试可回答该问题。这样的测试首先假定不存在基础模式（所谓**空假说**），然后分析实际数据，以计算它们偏离模式完全缺席的程度。如果偏离程度是统计不可能的（一般指5%的概率或更小），则认为它是数据中存在重要模式的有力证据。这些概率从偏离量的标准分布中计算出来，而对于标准分布人们期望在随机取样中能够看到。

在这种情况中，空假说是“属性是不相关的，因而对于一个无穷大的样本，信息收益将为零”。我们需要计算，在空假说之下，大小为的样本所展现出来的与正反例的期望分布的观察偏离的概率。假定真实不相关性为：

其中和分别是每个子集中正反样例的实际数目，和分别是它们的期望数目，通过比较实际数目和期望数目，能测量偏离。

总偏离的一个很方便的度量由下式给出：

在空假说之下，值的分布是带个自由度的分布（卡方分布）。我们能够用表或标准统计库例程，来判断一个特定值是肯定空假说还是拒绝之。例如，考虑饭店属性Type。的一个值将以1%的量级拒绝）。该剪枝称为**剪枝**。

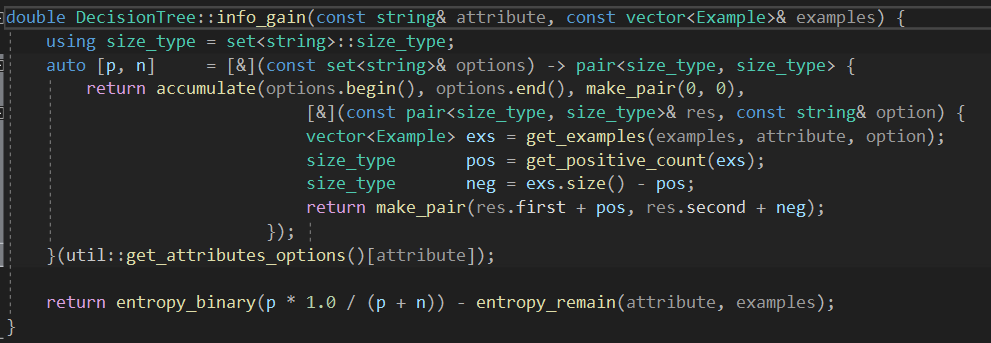
剪枝的使用可容忍样例中的噪音。样例标注的错误（例如，样例 实际上应该是）使预测误差线性增长，而样例描述的错误（例如，描述中，而实际上）具有渐近线效果，当树收缩至更小集合时，预测变得更坏。当数据中包含大量噪音时，相比于非剪枝树，剪枝树的性能有重要的改进。同时，剪枝树通常更小，因此更易理解。

最后一个警告：也许你会认为，看起来剪枝与信息收益相似，为什么不使用所谓早期终止的途径将它们合并？早期终止是指，当不存在可用于分裂的好属性时，决策树算法停止生成结点，而不是辛辛苦苦产生结点，然后剪除它们。使用早期终止途径存在的问题是，它阻止我们识别这样的情形，其中不存在好属性，但存在富含信息的属性组合。例如，考虑两个二值属性的XOR函数。如果对于输入值的四种组合，样例数目都大致相等，则没有任何属性是富含信息的。这时可以做的正确事情是，用一个属性（无论哪一个）进行分裂，然后在下一级将得到富含信息的分裂。早期终止失去了这一好处，但是“生成-然后-剪枝”（generate-and-then-pruning）能够正确处理这种情况。

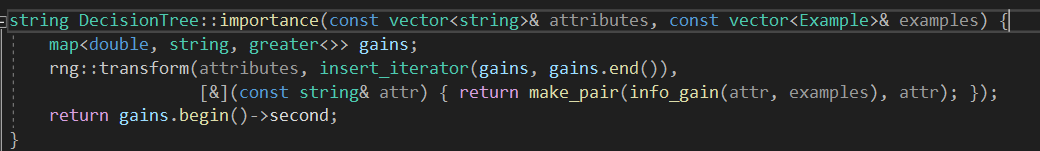
## 5 结论

可以看到程序正确分类了大部分的测试样本。

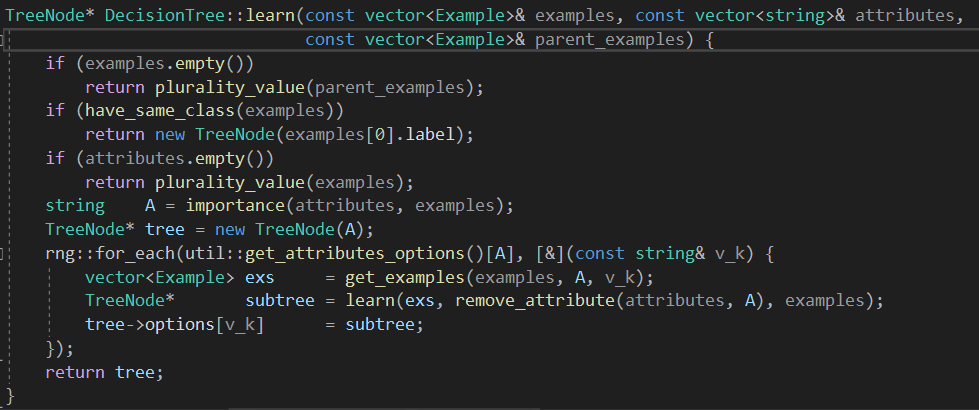
## 6 代码



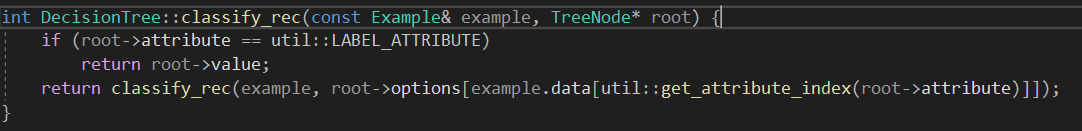
info\_gain



Importance



learn



classify\_rec