**决策树**

摘要：本文介绍了决策树算法的基本原理，给了算法的伪代码，以应届大学生求职为案例逐步解释决策树在属性划分选择、剪纸处理方面上的计算流程。本文介绍了三种属性划分指标的选择，分别为“信息增益”、“增益率”和“基尼指数”。在剪枝处理方面上可以分为预剪枝和后剪枝两张剪枝策略，并都分别给出了案例分析。最后为了更好地说明决策树在分类方面的作用，以kaggle平台上的蘑菇数据集作为学习样本，用python3对数据进行处理分析，最后得出决策树模型图。

**Decision Tree Classifier**

Abstract: This paper introduces the principle of decision tree algorithm, gives the pseudo code of the algorithm, and gradually explains the calculation process of the decision tree in terms of attribute division selection and paper cutting processing. This paper introduces the choice of three attribute partitioning indicators, namely “information gain”, “gain rate” and “Gini index”. In the aspect of pruning treatment, it can be divided into two pruning strategies: pre-pruning and post-pruning, and case studies are given respectively. Finally, in order to better explain the role of decision tree in classification, the mushroom data set on the kaggle platform is used as the learning sample, and the data is processed and analyzed by python3, and finally the decision tree model is obtained.

**一、介绍**

决策树是一种常见的机器学习方法。以二分类任务为例，我们希望从给定训练集数据集学得一个模型以对新示例进行分类，这个把样本分类的任务，可以看作对“当前样本属于正类吗？”这个问题的“决策”或“判定”过程。顾名思义，决策树树是基于树结构来进行决策的，这恰是人类在面临解决问题时一种很自然的处理机制。

以企业在选择职员为案例，面试官首先关心的是面试者的一些基本情况，如年龄、学历、专业等，比如这家企业希望找到一个年轻气盛的、本科学历以上的、专业对口的人才，如果这些基本情况都满足了，再去深入了解面试者的综合技能，评价指标可能会有学生在校成绩处于哪个水平、是否获得过什么荣誉之类的。假设面试者跟面试官有如下对话：

HR：年龄多大了？

求职者：22岁，刚大学毕业。

HR：哪一家大学的？是全日制本科吗？

求职者：毕业于广东第二师范小学，是全日制本科。

HR：学什么专业的？专业跟你应聘的岗位相符吗？

求职者：学信息计算的，跟应聘的岗位对口。

HR：学习成绩如何，在专业排名属于哪个层次？

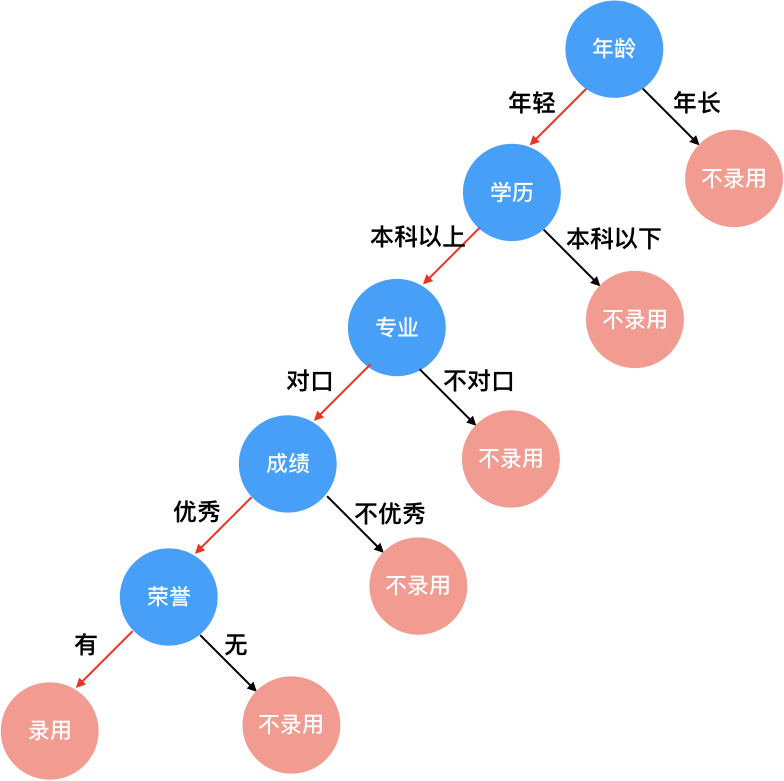
求职者：一直在班级名列前茅，属于上等水平。

HR：拿你应该拿过不少奖项吧？

求职者：是的，每年我都拿奖学金，参加好多比赛都拿奖了。

HR：好了，明天9点到这里来报道。

HR的决策过程就是典型的分类决策树。相当于对求职者的年龄、学历、专业、成绩水平、是否获得过荣誉等特征将这次面试结果分为两个类别录用和不录用。HR的决策逻辑如下：



上图表达了HR是否录用求职者的策略，其中蓝色结点（内部结点）表示判断条件，粉红结点（叶结点）表示决策结果，箭头表示在一个判断条件在不同情况下的决策路径，图中红色箭头表示了上面例子中HR的决策过程。

以上只是举一个例子，在实际应用中并不止考虑以上几种情况，这时候决策树会不停延伸，直至能作出决策为止。

**二、算法流程**

一般地，一棵决策树包含一个根结点、若干个内部结点和若干个叶结点。叶结点对应于决策结果，其他结点则对应于一个属性测试；每个结点包含的样本集合根据属性测试的结果被划分到子结点中；根结点包含样本全集。从根结点到每个叶结点的路径对应了一个判定测试序列。决策树学习的目的就是为了产生一棵泛化能力强，即处理未见示列能力强的决策树，其基本流程遵循简单且直观的“分而治之(divide-and-conquer)”策略[1]。算法流程如下：

---------------------------------------------------------------------

输入：训练集

属性集.

过程：函数

1:生成结点node；

2:if 中样本全属同一类别 then

3: 将node 标记为类叶结点；return

4: end if

5:if 为空集或中样本在上取值相同 then

6: 将node标记为叶结点，其类别标记为中样本数最多的类；retuen

7: end if

8: 从中选择最优划分属性；

9: for 的每一个值 do

10: 为node生成一个分支；令表示中在上取值为的样本子集；

11: if 为空集 then

12: 将分支结点标记为叶结点，其类别标记为中样本最多的类；

13: else

14: 以为分枝结点

15: end if

16: end for

输出：以node为根结点的一颗决策树

在上面举例的求职录用例子中，算法输入的训练集为的为年龄、学历、专业、成绩、是否有荣誉等数据，为HR是否录用求职者的数据。

显然，决策树的生成时一个递归过程。在决策树基本算法中，有三种情形会导致递归返回：

* 当前结点包含的样本全属同一类别，无需划分；
* 当前属性集为空，或是所有样本在所有属性上取值相同，无法划分；
* 当前结点包含的样本集合为空，不能划分。

在第二种情形下，我们把当前结点标记为叶结点，并将其类别设定为该结点所含样本最多的类别；在第三种情形下，同样把当前结点标记为叶结点，但将其类别设定为父结点所含样本类别最多的类别。

**三、属性划分选择**

由决策树算法流程可以看出，决策树学习的关键是第8步，即如何选择最优划分属性。一般而言，随着划分过程不断进行，我们希望决策树的分支结点所包含的样本尽可能属于同一类别，即结点的“纯度（purity）”越来越高。

“信息熵”是度量样本集合纯度最常用的一种指标。假定当前样本集合种第k类样本所占的比例为,则的信息熵定义为

的值越小，则的纯的越高。

假定离散属性有个可能的取值,若使用来对样本集进行划分，则会产生个分支结点，其中第个分支结点包含了中所有在属性上取值为的样本，记为我们可以根据信息熵的定义计算出的信息熵，再考虑不同的分支结点所包含的样本数不同，给分支结点赋予权重,即样本数越多的分支结点的影响越大，于是可计算出用属性对样本集进行划分所获得的“信息增益”

一般而言，信息增益越大，则意味着使用属性来进行划分所获得的“纯度提升”越大。因此，我们可用信息增益来进行决策树的划分属性选择，即在决策树算法第8步选择属性.

以信息增益为准则来选择划分属性的决策树算法叫ID3决策树学习算法。

以下表的数据作为一个案例，该数据集包含了20个训练样例，用以学习一棵能判别求职者是否录用的决策树。

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 编号 | 求职者是否年轻 | 是否本科学历以上 | 是否专业对口 | 综合能力是否突出 | 判定 |
| 1 | 是 | 否 | 否 | 否 | 不录用 |
| 2 | 是 | 否 | 否 | 是 | 录用 |
| 3 | 是 | 否 | 是 | 否 | 不录用 |
| 4 | 否 | 否 | 是 | 否 | 录用 |
| 5 | 是 | 否 | 否 | 是 | 不录用 |
| 6 | 否 | 否 | 是 | 否 | 不录用 |
| 7 | 是 | 是 | 是 | 是 | 录用 |
| 8 | 否 | 是 | 否 | 是 | 录用 |
| 9 | 否 | 否 | 否 | 否 | 不录用 |
| 10 | 是 | 否 | 否 | 否 | 不录用 |
| 11 | 否 | 否 | 否 | 是 | 不录用 |
| 12 | 是 | 否 | 否 | 否 | 不录用 |
| 13 | 是 | 否 | 否 | 否 | 不录用 |
| 14 | 否 | 否 | 是 | 否 | 不录用 |
| 15 | 是 | 是 | 是 | 否 | 不录用 |
| 16 | 否 | 否 | 是 | 否 | 不录用 |
| 17 | 是 | 否 | 否 | 否 | 不录用 |
| 18 | 否 | 否 | 否 | 否 | 不录用 |
| 19 | 否 | 否 | 否 | 否 | 不录用 |
| 20 | 是 | 否 | 否 | 否 | 不录用 |

表1 决策树学习训练集数据

显然判别的结果只有两个，录用和不录用，即=2。在决策树学习开始时，根结点包含中的所有样例，其中正例（在此案例中为录用的案例）有4个，占，即 ；反例（在此案例中为不录用的案例）占 。于是，根据信息熵的计算公式可以计算出根结点的信息熵为

然后，我们要计算出当前属性集合求职者是否年轻，是否本科学历以上，是否专业对口，综合能力是否突出 中每个属性的信息增益。以属性“求职者是否年轻”为例，它有两种可能的取值：。若使用该属性对进行划分，则可得到2个子集，分别记未：，。

子集包含编号为的11个样例，其中正例（判别录用）占，反例占；子集包含编号为的9个样例，其中正例（判别为录用）占，反例占。根据信息熵的计算公式可计算出用属性“求职者是否年轻”划分之后所获得2个分支结点的信息熵为

于是，可以根据信息增益的计算公式计算出属性“求职者是否年轻”的信息增益为

类似地，我们可以计算出其他属性的信息增益：

显然，属性“”的信息增益最大，于是它被选为划分属性。下图为基于“”对根结点进行划分的结果，各分支结点所包含的样例子集显示在结点中。

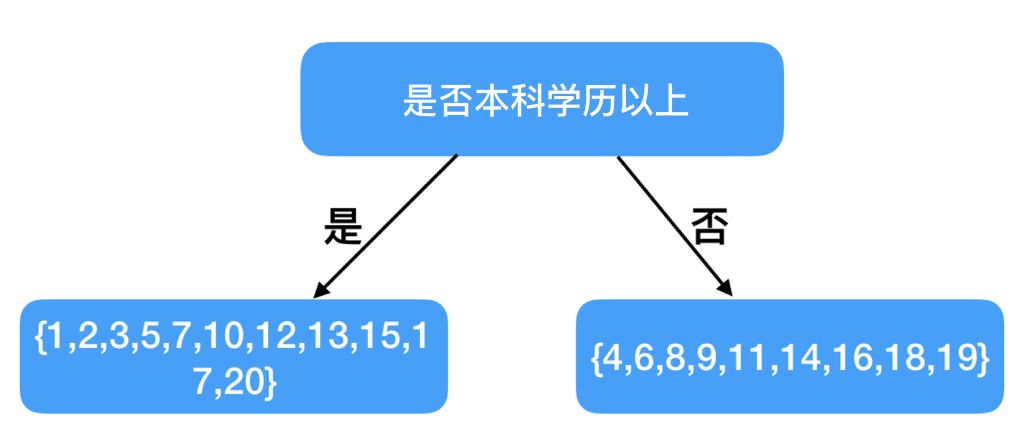
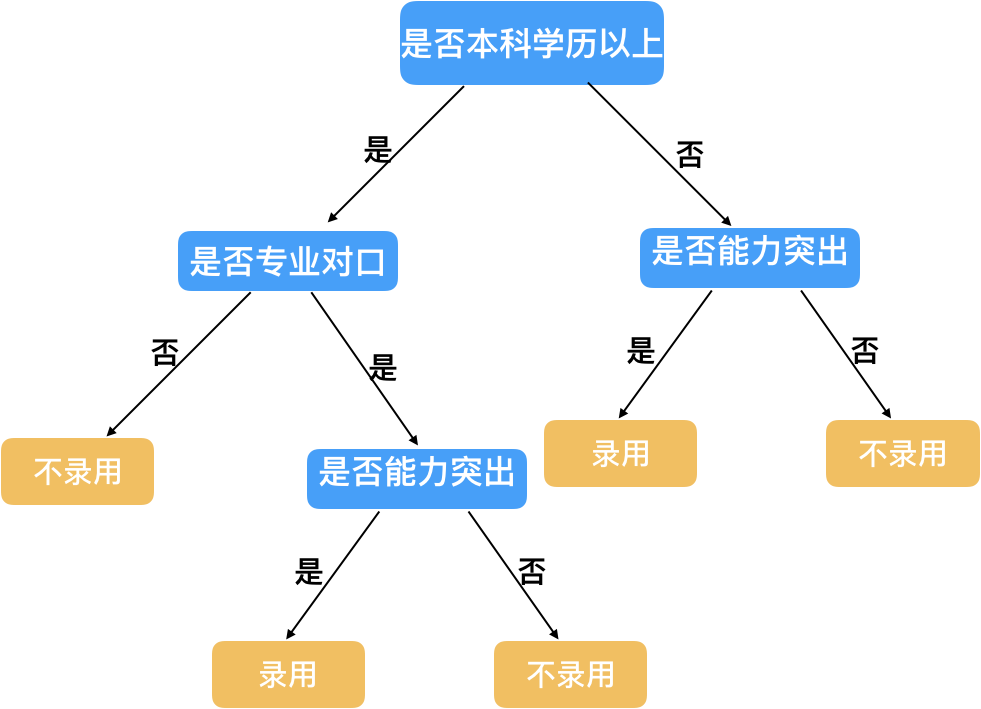


图1 基于“”属性对根结点划分

然后，决策树学习算法将每个分支结点进一步划分。以上图左边的分支结点（“是”）为例，该结点包含对样例集合中有编号为的11个样例，可用属性集合为。基于计算出各属性的信息增益：

“”、“”两个属性均取得了最大的信息增益，可任选其中之一作为划分属性。类似地，对每个分支结点进行了上述操作，最终会得到一棵完整的决策树。下图由上面的案例根据ID3决策树算法生成的一棵决策树，从决策树中看到最深只有3个属性分支，只是因为在第三层结点包含的样本全属同一类别，无需继续划分。



除了以信息增益最大为准则来选择划分属性外，还有其他准则，比如增益率。实际上，信息增益准则对可取值数目较多的属性有所偏好，为减少这种偏好可能带来的不利影响，C4.5决策树算法不直接使用信息增益，而是使用“增益率”来选择最优划分属性。增益率定义为

其中

称为属性的“固有值（intrinsic value）”。属性的可能取值数目越多（即越大），则的值通常会越大。

此外，CART决策树使用的是“基尼指数（Gini index）”来选择划分属性，数据集的纯度可用基尼值来度量：

直观来说，反映了从数据集中随机抽取两个样本，其类别标记不一致的概率。因此，越小，则数据集的纯度越高。

属性的基尼指数定义为

于是，我们在候选属性集合中，选择那个使得划分后的基尼指数最小的属性作为最优划分属性，即.

**三、剪枝处理[2]**

剪枝是决策树学习算法对付“过拟合”的主要手段。在决策树学习中，为了进可能正确分类训练样本，结点划分过程将不断重复，有时会造成决策树分支过多，这时就可能因训练样本学得“太好”了，以致于把训练集自身的一些特点当作所有数据都具有的一般性质而导致过你和。因此，可通过主动去掉一些分支来降低过拟合的风险。

决策树剪枝的基本策略有“预剪枝”和“后剪枝”。预剪枝是指在决策树生成过程中，对每个结点在划分前先进行估计，若当前结点的划分不能带来决策树泛化性能提升，则停止划分并将当前结点标记为叶结点；后剪枝则是先从训练集生成一棵完整的决策树，然后自低向上地对非叶结点进行考察，若将该结点对应的子数替换为叶结点能带来决策树泛化性能提升，则将该子树替换为叶结点。

如何判断决策树泛化能力性能是否提升呢？在这里我们采用留出法，即预留一部分数据作为“验证集”以进行性能评估。上面表1只是学习决策树时所用到的训练数据，下表2是进行性能测试的验证数据，我们将用下面的数据来判断已训练好的决策树是否需要剪枝处理。

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 编号 | 求职者是否年轻 | 是否本科学历以上 | 专业是否对口 | 综合能力是否突出 | 判定 |
| 21 | 是 | 否 | 否 | 否 | 不录用 |
| 22 | 是 | 否 | 是 | 否 | 不录用 |
| 23 | 否 | 是 | 否 | 是 | 录用 |
| 24 | 是 | 否 | 是 | 否 | 不录用 |
| 25 | 是 | 否 | 否 | 否 | 不录用 |
| 26 | 是 | 是 | 否 | 否 | 录用 |

表2 决策树泛化能力验证数据

**3.1 预剪枝**

在前面的案例中，基于信息增益准则，我们会先选取属性“是否有本科学历以上”来对训练集进行划分，并产生了2个分支。然而，是否应该进行这个划分呢？预剪枝要对划分前对划分前后的泛化性能进行估计。

在划分之前，所有样例集中在根结点。若不进行划分，则根据决策树算法第6步，该结点将被标记为叶结点，其类别标记为训练集样例最多的类别。而样例最多的类别为不录用，也即是所有求职者都不需要去考虑各种指标了，全部判别为非不录用。先不去考虑这样做是否符合实际，如果用验证集对这个单结点决策树进行评估，则编号{21,22,24,25}的样例被分类正确，另外2个样例分类错误，于是验证集精度为。

在用属性“是否本科学历以上”划分之后，下图结点、分别包含编号为{7，8，15}、{1,2,3,4,5,6,9,10,11,12,13,14,15,16,17,18,19,20}的训练样例，因此这两个结点分别被标记为叶结点“录用”、“不录用”。此时，验证集中编号为{21,22,23,24,25,26}全部样例被分类正确，验证集精度为。于是用“是否本科学历以上”进行划分得以确定。

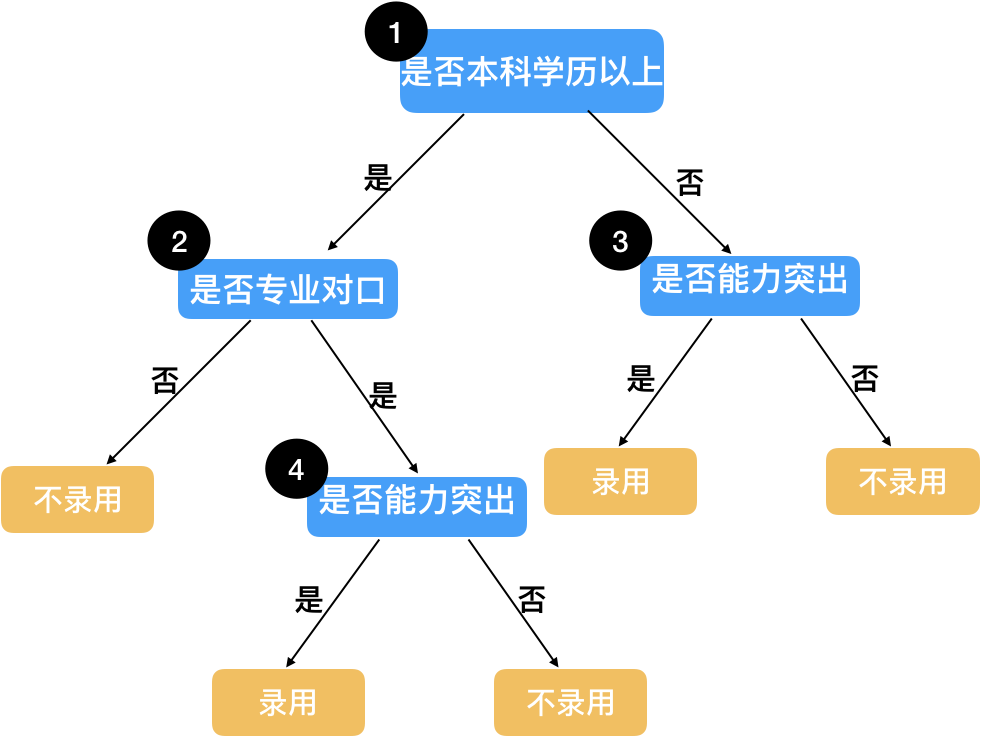


图3 基于表1生成的未剪枝决策树

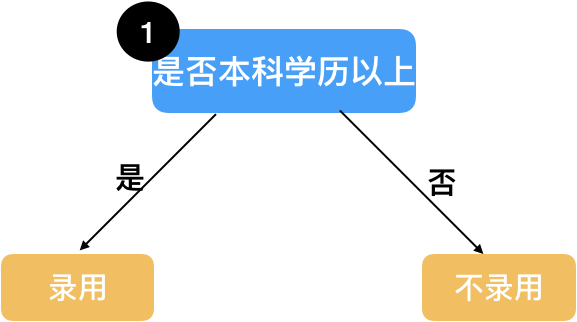


图4 基于表1生成的预剪枝决策树

然后，决策树算法应该对结点进行划分，基于信息增益准则将挑选出划分属性“综合能力是否突出”。如果在使用“专业是否对口”划分后，使得验证集精度下降了，那么预剪枝策略将禁止结点被划分。在上图中，通过预剪枝，原本训练出来的决策树被减到只剩下结点，因为只需判断“学历是否本科以上”验证集的准确率已经100%了，再针对结点、继续划分也不会提高验证集的准确率，故没必要划分。

对比图4和图3可看出，预剪枝使得决策树的很多分支都没有“展开”，这不仅降低了过拟合的风险，还显著减少了决策树的训练时间开销和测试时间开销。

**3.2 后剪枝**

后剪枝先从训练集生成一棵完整决策树，例如基于表1的数据，我们得到图3所示决策树。易知，该决策树的验证集精度为83.33%。

后剪枝首先考察图3中的结点。若将其领衔的分支剪除，则相当于把替换成叶结点。替换后的叶结点包含样例中类别最多的类别为“不录用”，那么叶结点类别应该标记为“不录用”，此时决策树验证精度仍为83.33%。于是可以不进行剪枝。当然，为了节省计算开销 ，也可以进行剪枝。

再考虑上一层的结点，若将其领衔的子树替换成叶结点，则替换后的叶结点包含样例中类别最多的类别为“不录用”，那么该叶结点应该标记为“不录用”，此时，决策树验证精度提升到100%。于是，后剪枝策略决定剪枝。

因为经过后剪枝后的决策树验证精度已经到达100%，不能再提升了，所以再对结点、进行剪枝也没意义。当然，如果在其他验证集数据中验证得继续再对或进行剪枝能提高准确率，那应该剪到精确度不能再提高为止。

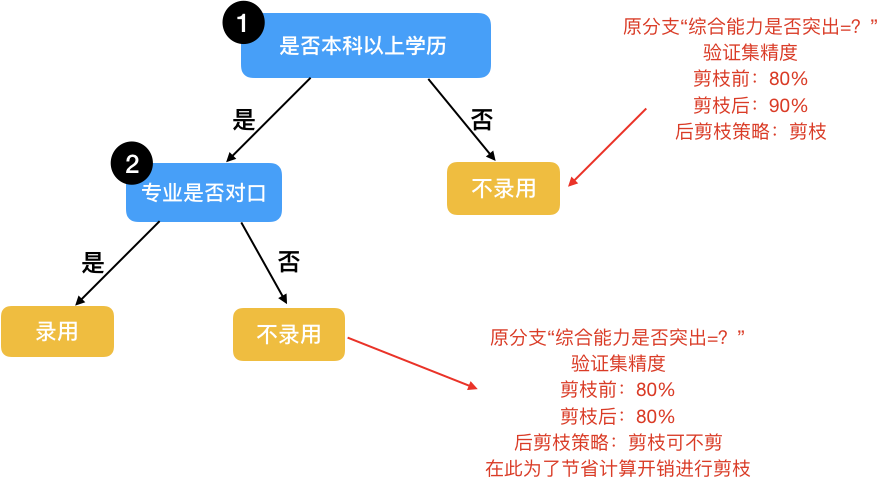


图5 基于表1生成的后剪枝决策树

**四、连续值处理**

以上仅讨论了基于离散属性来生成决策树。现实学习任务中常会遇到连续属性，以下部分讨论如何在决策树学习中使用连续属性。

由于连续属性的可取值数目不再有限，因此，不能根据连续属性的可取值来对结点进行划分。此时，应该先对连续属性进行离散化处理。最简单的策略是采用二分法对连续属性进行处理，这也是C4.5决策树算法中采用的机制。

给定样本集和连续属性，假定在上出现了个不同的取值，将这些值从小到大进行排序，记为,基于划分点可将分为子集和，其中包含那些在属性上取值不大于的样本，而包含那些在属性上取值大于的样本。显然，对相邻的属性取值与来说，在区间中取任何值所阐述的划分结果相同。因此，对连续属性,我们可考察包含个元素的后划分点集合

（4.1式）

即把区间作为候选划分点。然后，我们就可像离散属性值一样来考察这些划分点，选取最优的划分点进行样本集合的划分。对于ID3算法，信息增益的属性划分选择变为

（4.2式）

其中是样本集基于划分点二分后的信息增益。于是，我们就可选择使最大化的划分点。

继续以前面的案例，我们在表1的训练集熵增加一个连续属性“空箱数(**表**MANIFEST\_HEAD)”，得到下表3所示的数据集。下面我们用这个数据集生成一棵决策树。

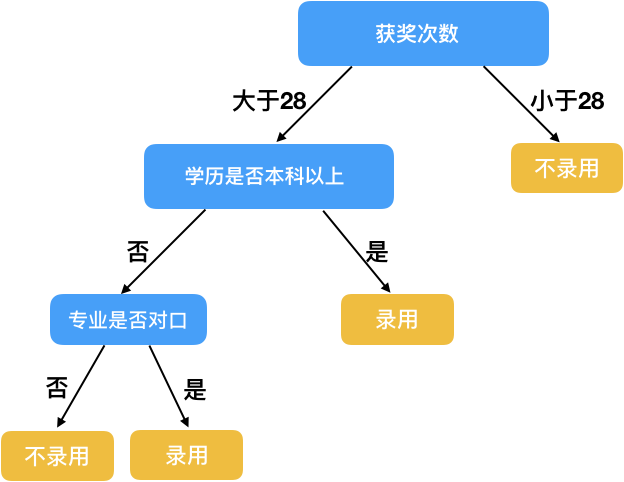
|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 编号 | 求职者是否年轻 | 是否本科学历以上 | 专业是否对口 | 综合能力是否突出 | 获奖次数 | 判定 |
| 1 | 是 | 否 | 否 | 否 | 0 | 不录用 |
| 2 | 是 | 否 | 否 | 是 | 24 | 录用 |
| 3 | 是 | 否 | 是 | 否 | 0 | 不录用 |
| 4 | 否 | 否 | 是 | 否 | 12 | 录用 |
| 5 | 是 | 否 | 否 | 是 | 2 | 不录用 |
| 6 | 否 | 否 | 是 | 否 | 30 | 不录用 |
| 7 | 是 | 是 | 是 | 是 | 76 | 录用 |
| 8 | 否 | 是 | 否 | 是 | 0 | 录用 |
| 9 | 否 | 否 | 否 | 否 | 56 | 不录用 |
| 10 | 是 | 否 | 否 | 否 | 11 | 不录用 |
| 11 | 否 | 否 | 否 | 是 | 0 | 不录用 |
| 12 | 是 | 否 | 否 | 否 | 26 | 不录用 |
| 13 | 是 | 否 | 否 | 否 | 4 | 不录用 |
| 14 | 否 | 否 | 是 | 否 | 10 | 不录用 |
| 15 | 是 | 是 | 是 | 否 | 0 | 不录用 |
| 16 | 否 | 否 | 是 | 否 | 0 | 不录用 |
| 17 | 是 | 否 | 否 | 否 | 0 | 不录用 |
| 18 | 否 | 否 | 否 | 否 | 0 | 不录用 |
| 19 | 否 | 否 | 否 | 否 | 0 | 不录用 |
| 20 | 是 | 否 | 否 | 否 | 0 | 不录用 |

表3 决策树训练数据集

对属性“获奖数”，在决策树学习开始时，根结点包含11个不同的属性值。根据4.1式，该属性的候选划分点集合包含10个候选值：。由4.2式可计算出属性“获奖数”的信息增益为0.2262，对应划分点为28。

再计算出各属性的信息增益为

于是，“获奖次数”被选作跟结点划分属性，此后结点划分过程递归进行，最终生成下图所示的决策树。



**五、决策树在蘑菇数据集上的应用**

在前面介绍的案例中，为了方便计算和尽量使大家通俗地理解决策树的演算过程，用到的数据都是作者本人自编的，而且数据量较小。以下为了说明决策树在实际中的应用，引用kaggle平台上的mushrooms（蘑菇）数据集作为模型的学习样本（数据下载地址：<https://www.kaggle.com/uciml/mushroom-classification>），在python3上调用sklearn库进行训练模型。

该数据集共有8214个样本，22个特征和1个分类标签。特征变量全为英文代号，为了方便计算，在数据分析前先要对数据标签全部采用整数替代。另外把学习样本分为训练集和测试集，比例为6:4，以测试集验证训练集训练出来的决策树模型效果。在本次训练中，采用的是以“基尼指数”作为属性划分选择指标，另外不限制决策树的最大深度，最后把结果输出到out.png文件中，其结果看见下图6.

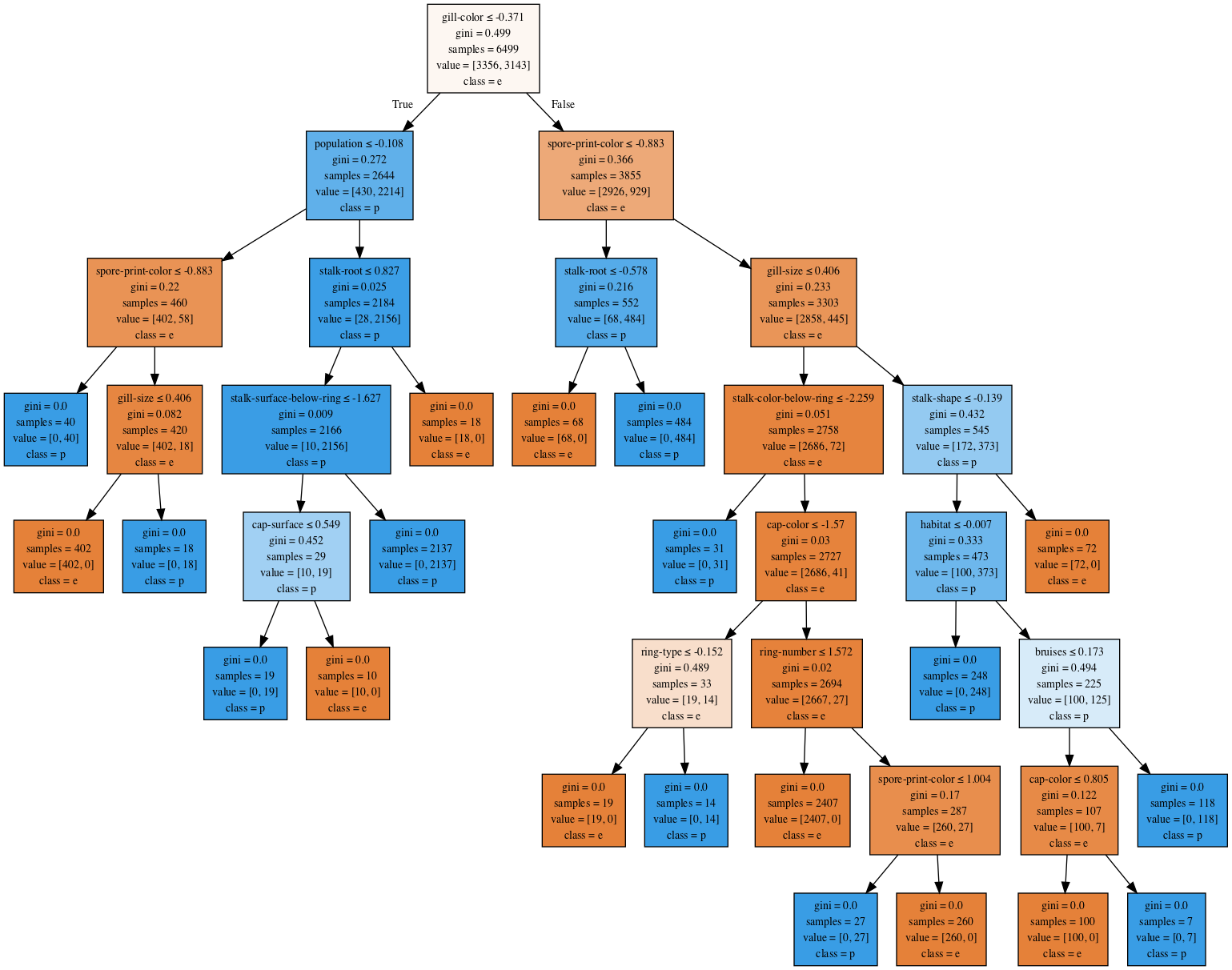


图6 蘑菇数据集的决策树模型图

**参考文献**

[1]周志华. 机器学习 : = Machine learning[M]. 清华大学出版社, 2016.

[2]李航. 统计学习方法[M]. 清华大学出版社, 2012.