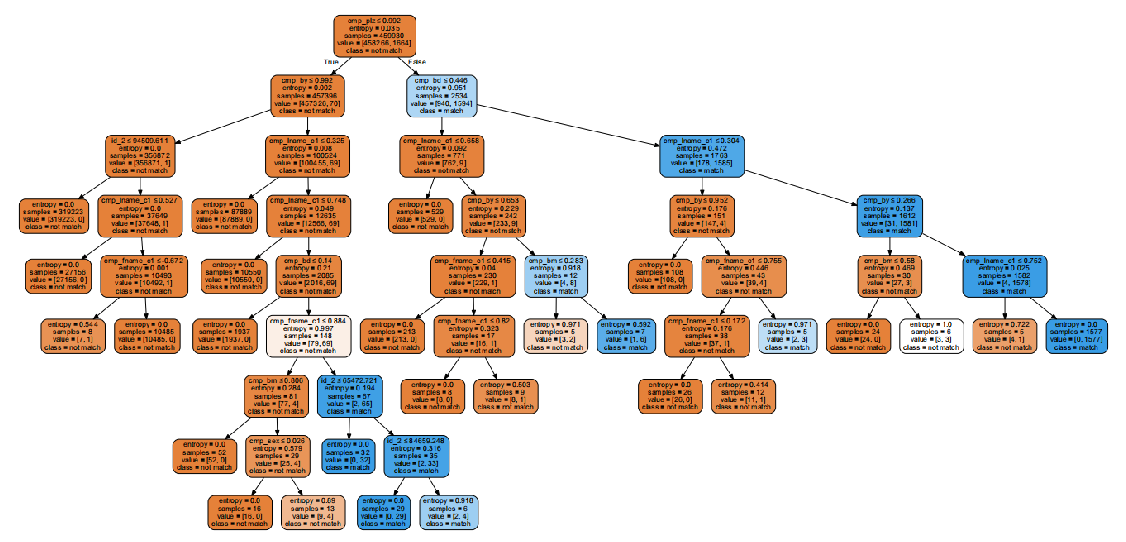
分类号 密级

UDC

“大数据分析”课程结课论文



利用改进信息熵决策树算法C4.5实现对记录链接模式匹配问题的分类

作 者 姓 名： 梁子

学 号： 20163933

单 位 名 称： 东北大学信息科学与工程学院

专 业 名 称： 自动化

东 北 大 学

2018年11月

# **课程设计（论文）任务书**

|  |
| --- |
| **课程设计（论文）题目：**  **利用改进多变量信息熵决策树算法C4.5实现对记录链接模式匹配问题的分类** |
| **报告(论文)的基本内容：**  使用数据集： Record Linkage Comparison Patterns[[1]](#footnote-0)  要求使用算法： decision tree C4.5  本论文工作：  (1)给出了针对数据集种缺省值的特殊处理方法；  (2)提出一种解决决策树无法处理同属性不同结果的数据的解决方案；  (3)将数据集进行分类处理，并细致分析了本算法的精确度、错误率、过拟合与欠拟合问题。 |
| **报告（论文）部分：**  **题目：利用改进多变量信息熵决策树算法C4.5实现对记录链接模式匹配问题的分类**  **设计或论文专题的基本内容：** |
| **学生接受设计（论文）题目日期**  **第　　周**  **指导教师签字：**  **年　　月　　日** |

利用改进多变量信息熵决策树算法C4.5实现对记录链接模式匹配问题的分类[[2]](#footnote-1)

摘要

决策树算法是机器学习系列理论算法中的一种，它采用数据的属性值来进行树状结构生成，因而表征了一种对象属性到对象值的映射。树中的每一个节点表示对象属性的判断条件，其分支表示符合节点条件的对象，叶节点表示对象所属的预测结果。因次，决策树算法主要用于进行目标任务的预测和分类。

本论文选取了德国美因茨约翰内斯古腾堡大学医学中心的癌症病人身份数据集作为目标分类数据集，按照老师要求，以数据挖掘中的有监督学习决策树预测算法C4.5为基础，搭建分类预测模型。基于该任务完成了如下工作：

1）利用高维度映射的方法对数据的属性缺省值进行了预处理，使得在未引进人为噪声的前提下保留了原始数据的全部信息；

2）在数据集的利用上，将全部的5749132组数据按照匹配与否的相同频率均分成了十组，并分别采用训练集、测试集、验证集的多种比例来验证本算法的鲁棒性和泛化能力，分别在测试集上达到了99.9956%、99.9810%、99.9977%的精度，并对是否存在欠拟合问题和过拟合问题以及最佳数据尺度进行了分析；

3）在决策树算法C4.5的基础上，权衡了悲观错误剪枝法、交叉复杂剪枝法等多种方法进行剪枝处理；

本程序使用python3编写，在读取目标数据时借用了函数库pandas，在进行缺省值处理时借用了函数库numpy，在生成决策树和划分数据集时借用了机器学习函数库scikit-learn，在进行数据可视化时借用了函数库graphviz和matplotlib，在进行源代码编写时参考了C4.5作者的源代码。此处一并表示感谢。

实验证明，本模型在解决该问题上的准确率稳定在99.99%，可以说完全可以胜任该数据集任务的分类和预测。

**关键词**： 决策树模型；C4.5算法；信息熵；缺省值留存；剪枝处理

# 目录

课程设计（论文）任务书 II

摘要 III

目录 IV

1. 决策树算法C4.5 5

1.1. ID3算法(Iterative Dichotomiser 3 Algorithm) 5

1.1.1. 信息熵(Information Entropy Theory)和信息增益(Information Gain) 5

1.1.2. ID3算法具体流程 6

1.1.3. 决策树ID3算法的缺陷 6

1.2. 以ID3为基础——C4.5决策树算法 6

1.2.1. 连续属性值离散处理 6

1.2.2. 信息增益率（Information Gain Rate） 7

1.2.3. 缺省值处理 7

1.2.4. 剪枝处理：基于错误的剪枝法（Error-based Pruning） 7

2. 以C4.5为基础——自定义的改进 8

2.1. 数据预处理——属性值高维度映射 8

2.2. 抑制过拟合——借鉴CART算法的剪枝策略 9

2.2.1. PEP悲观错误剪枝策略(Pessimistic Error Pruning) 10

2.2.2. 代价复杂剪枝策略(Cost-complexity Pruning) 10

2.3. 解决数据矛盾问题 12

3. 实验验证与结果分析 12

3.1. 数据集介绍 12

3.2. 实验验证与结果分析 13

4. 总结 17

4.1. 结题 17

4.2. 不足之处 17

5. 参考文献 17

# 

# 决策树算法C4.5

## ID3算法(Iterative Dichotomiser 3 Algorithm)[[3]](#endnote-0)

决策树算法的本质为根据已有数据的属性对数据进行分类。因而所有的决策树算法都会面临选择哪一个属性如何进行条件选择的难题。

针对以上问题，昆兰于1970年提出了III型迭代二叉树算法（ID3），该算法建立在奥卡姆剃刀理论的基础上，即越是小型的决策树越优于大的决策树的简单理论。该算法属于启发式算法，采用香农信息论中的熵来度量决策树的决策选择过程，从而实现了简单高效的决策。

### 信息熵(Information Entropy Theory)和信息增益(Information Gain)[[4]](#endnote-1)

熵（entropy）的概念来自于物理，1948年，克劳德·艾尔伍德·香农将热力学中的熵引入到信息论中。依据玻尔兹曼热力学定理(Boltzmann's H-theorem)，香农将一个随机过程X的熵H定义为：



其中P为随机过程X的概率质量函数(probability mass function)，E为数学期望函数，属于X的自信息量。

当取自有限多的离散样本时，一个随机过程的熵可以表述为



其中为对数的底，通常取2，此时熵的单位为比特(bit)。

根据单变量的熵公式，可以过渡到多变量的联合熵表达式，下面给出X，Y双变量时的联合熵表达式



基于以上的联合熵和条件概率的公式，条件熵可以表述为



根据条件概率的定义，对于一个具有多变量的随机过程，可以使用信息熵来表示变量X的不确定度，可以使用信息熵来表示当变量Y确定时变量X带来的不确定度，因而，若定义为



代表随机变量Y确定后对X带来的确定度的提升，称之为信息增益。

ID3算法即是采用信息增益来作为判定当前节点应该采用数据的哪一个属性来作为分类依据。信息增益越大，则该属性在当下越适合用来进行分类。

### ID3算法具体流程

设共有m个训练样本，输出集合为D，每个样本中有n个离散属性，属性集合记为A，输出为决策树T。则ID3算法的流程可以归结为：

1. 计算输出集合D的信息熵;
2. 判断n是否大于2，如果不是，则程序结束，返回决策树T, 如果大于2，进行步骤三；
3. 计算集合A中的所有属性的条件信息熵，继而求出A中所有属性的信息增益，取最大者作为当前决策树的分裂依据，生成新的决策树T，并将该属性从属性集合A中剔除，将记录属性数目的变量n减一，跳回到步骤二。

### 决策树ID3算法的缺陷

ID3算法给出了一种选择决策属性的衡量方法，首次解决了如何选择属性作为决策树分裂节点的问题，但是它仍然有很多不足之处，主要包含以下几个方面：

1）ID3算法没有给出属性值为连续值的属性的解决方案，这限制了其应用；

2）ID3算法采用信息增益最大的方式来进行决策属性的选择，而往往包含属性值多的属性其信息增益总是偏大，但是他们包含的不确定度却往往是相同或者其他结果的；

3）ID3算法没有针对缺省值给出解决方案；

4）ID3算法没有考虑过拟合的问题。

## 以ID3为基础——C4.5决策树算法[[5]](#endnote-2)

为了解决以上四个缺陷，罗斯·昆兰(Ross Quinlan)于1992年提出了C4.5决策树算法。该算法在ID3算法的基础上，采用将连续值属性离散化的方法来解决属性值为连续值的数据分类问题[[6]](#endnote-3)，利用信息增益率代替信息增益来改善属性中可选属性值数目对信息不确定度关系影响的问题，并且给出了对缺省值的处理思路，针对过拟合问题提出了剪枝处理方案。下面对这四个方面进行详细讨论：

### 连续属性值离散处理

对于m个样本的分类任务，若属性A为属性值连续变化的特征，可以通过求取m个样本中相邻样本的平均值作为划分点，得到m-1个划分点。其中第i个划分点可以表示为，因此就可以依据这m-1个划分点得到生成二叉树的思路。仍然是计算采用不同的划分点作为划分依据时得到的系统信息增益，选择信息增益最大的划分点作为当前的决策分支点。

与离散值的分支不同的是，连续属性值离散处理后在被作为决策分支点之后仍然不会消失，而是继续作为二叉树的的一个判断依据，和其他属性的信息增益进行比较，直到该属性下的所有样本属性值相同为止。

### 信息增益率（Information Gain Rate）

信息增益具有偏向于属性值个数多的属性的特征，为了针对这个问题进行修正，首先引入特征熵的概念。

对于一个输出属性集合为D的随机过程，定义其中的一个属性A的特征熵为



其中，n为属性A的属性值个数，为属性A的第i个取值对应的样本个数。

通过引入特征熵，就可对信息增益进行校正，从而得到信息增益率：



这样就使得对决策节点的选取更加地客观科学。

### 缺省值处理

C4.5中对缺省值的处理方式给出的解决思路是将缺省的数据降低权重加入信息熵计算中，之后将缺省数据视为经过了若干次决策之后的数据。

C4.5算法认为，对于输出特征集合为D的m个样本，属性A出现缺省的样本收集为m1,其特征集合记为D1，属性A未出现的缺省的样本收集为m2，其特征的集合记为D2。则在进行条件信息熵的计算时，公式可以修正为



其中为其加权平均值，即未缺省的样本在整体样本中的所占比例。

进行决策之后，如果属性A被选作决策节点，则属性A在下面的节点计算中将会消失，因而缺省属性A数值的数据将会被同样地施加到两个子决策树中，从而实现对缺省值的处理。

### 剪枝处理：基于错误的剪枝法（Error-based Pruning）

剪枝处理是生成决策树之后必不可少的环节，在2.2中将会给出详细介绍。1992年，在昆兰提出C4.5算法时，也给出了一种剪枝处理的策略：基于错误的剪枝法（error-based pruning）。

该策略属于后剪枝法，即对于每一个节点，都要计算其在剪枝前后在验证集上的误判个数，如若剪枝处理有助于减小误判（包括相等的情况），则删掉该节点所在的分支，反之则保留。

这种方案属于中规中矩的处理方法，本论文在实际使用时进行了改进。

# 以C4.5为基础——自定义的改进

## 数据预处理——属性值高维度映射

在对数据集Record Linkage Comparison Patterns 进行处理时，注意到数据中存在着大量的缺省值。这在档案记录中出现是十分正常的，如图2.1所示。

决策树算法属于稳定性较差的机器学习算法，因而属性值的变化极有可能会引起决策树结构上的根本变化，因而合理处理这些缺失的属性值十分必要。在1.2.3中已经提到过，C4.5算法将有无缺省值看作一种权重，之后却又在决策完成后将数据取回。这样做对缺失的属性值进行了一定的模拟和预测，但却存在着很多问题：首先，我们并不知道实际的数据中这属性上的取值到底如何，因而利用已知的属性值去推断未知的属性值就存在着将局部特征视为全局特征的危险，从而使得最终的预测结果产生偏差。此外，通过在子决策树中加入相同部分的缺省属性样本集合，将会使得整体的决策树规模增大，也会使得其计算量增大，且易产生过拟合等其他的问题。

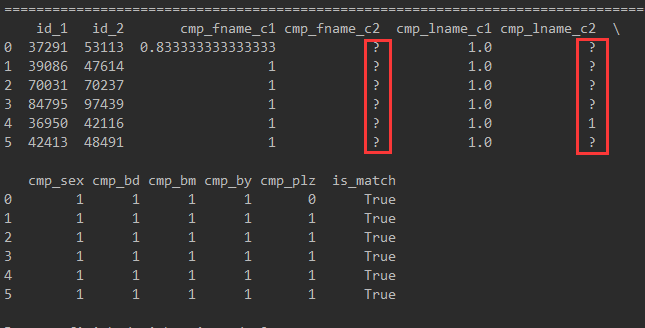


图2.1 读取Record Linkage Comparison Patterns数据集的前6组数据剪影，红框标出的记为数据的缺省值

针对以上问题，本论文并没有采用加权求取信息熵的策略，而是将每一个属性的属性值均映射到高维空间，即把缺省值也作为一种属性值进行实际处理。采用这种方案后，缺省值将代表一个特征的相同程度，这样实现了对数据的无损修正，也去掉了对缺省数值的预测环节。这样做的优点是保证了对数据的本原的利用，缺点是有可能缺省的数据来源于相同属性下不同的属性值，但是这种来源在数据中是无法表征的，因而理论上讲这样处理可以达到对数据的保守最佳利用。

对相同的属性下的每一个缺失的属性值都用相同的一个符合代替，尔后将这种符号作为决策中的一个属性值去使用，从而实现了在不改变数据原意的基础上保留下来缺省属性值的意义，相关的处理代码和效果如图2.2所示。

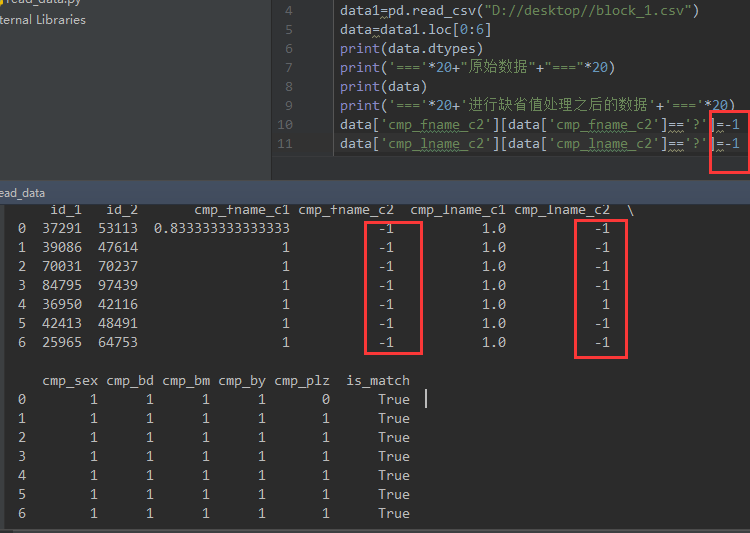


图2.2 对缺省值进行的处理

## 抑制过拟合——借鉴CART算法的剪枝策略

过拟合(over fitting)是指模型在训练集上误差极小，但却在验证集上具有较大的误差。通常过拟合问题是由于模型过于复杂产生的，如图2.3所示。因此为了抑制模型的急剧扩大，就必须采取一些特定的措施。决策树算法由于其参量多的特点在使用过程中极易出现过拟合问题，所以必须采取适当的措施对其进行处理。

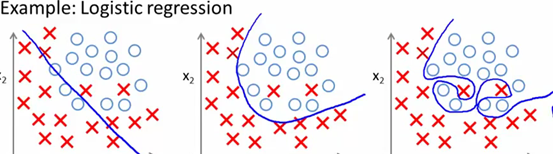


图2.3 三幅图片分别代表：欠拟合、拟合、过拟合

通常，决策树算法如ID3、C4.5都会对决策树的规模进行尽可能的压缩，这也是信息熵被引入的重要因素。但是，尽管如此，决策树算法的饱和问题仍是不尽人意。因而几乎所有的决策树算法都需要进行剪枝处理。

目前已有的决策树剪枝策略主要有：误差降低剪枝(Reduced error pruning)，悲观错误剪枝(pessimistic error pruning)，最小误差剪枝(minimum error pruning)，代价复杂剪枝(cost-complexity pruning)，基于错误的剪枝(error-based pruning)和评估值剪枝(critical value pruning)。

由于基于错误的剪枝已经在上文中给出了介绍，所以下面主要讨论悲观错误剪枝和代价复杂剪枝法两种。

### PEP悲观错误剪枝策略(Pessimistic Error Pruning)

PEP剪枝属于自上而下的修剪。由于修建过程中仅使用了训练集误差，因而不需要额外准备验证集。同时，为了改善修剪过程偏向于训练数据而带来的错分误差，常常添加0.5作为修正因子。

具有T个节点的树的误差率衡量为



表示t节点下误测样本点数目，N(t)表示节点t之下所有的样本点数。

对于上面的决策树，当去掉了节点K之后的个节点的误差率衡量为



若设，则在去除节点K前后，有



即去掉一个节点后的误差率变化量等于去掉之后的误测点数减去去掉之前的误测样本点数之后再处以去掉之前该节点下的总节点数。

剪枝的条件为



### 代价复杂剪枝策略(Cost-complexity Pruning)

本论文中采用的方案是代价复杂剪枝法，在布莱曼（Breiman）于1984年提出的CART决策树算法中首次出现。

该剪枝法的思路是：先生成决策树，然后产生所有可能的剪枝后的决策树，使用交叉验证来检验各种剪枝的效果，最后选择泛化能力最好的剪枝策略。因而剪枝法主要可以分为两步。第一步即从原始决策树生成各种剪枝后的决策树，第二步是采用交叉验证来检测剪枝后的预测能力，选择泛化预测能力最好的剪枝后的树作为最终的决策树。

因而这里需要解决两个问题：一是如何剪枝，二是如何评估剪枝之后的决策树的泛化预测能力。

代价复杂剪枝法的剪枝思路来源于线性回归中的正则化(regularized linear regression）操作。

线性回归正则化的基本表达式为



其中的为超参数，即通过控制的大小来使得多项式的阶数维持在合理的范围内。

与之类似，CP剪枝法也采用一个超参数来控制决策树的深度和广度。在CART算法中，剪枝的损失函数度量被定义为



其中为正则化参数，等同于上文中提到的，而表示的是所得决策树的训练误差，在CART中采用的是基尼系数：



在C4.5算法中，训练误差损失函数仍然采用香农信息熵公式。损失度量函数中的表示的是子树中叶子节点的数量。

如此一来，就完成了对剪枝损失函数的定义。

可以看出，类似于线性回归中对高阶次数的限制，CP剪枝法约束了剪枝的广度。因而通过调节超参数的大小，就可以改善决策树的尺寸，越大，剪枝后得到的超参数便越小。

利用这个度量进行剪枝，即计算出每一个子决策树是否剪枝的阈值



之后，针对不同的所对应的剪枝后的子树进行交叉验证，便可以找到一个最优的值，利用这个，便可以选择得到一个最优子树。

将上述步骤总结起来如下：

1. 初始化，最优子树集合；
2. 从叶子节点开始自下而上计算各个内部节点t的训练误差损失函数，叶子节点数，并记录此时的正则化阈值，更新的值；
3. 得到所有节点的正则化参数值的集合M;
4. 从M中选择最大的值，并自上而下访问子树t的内部节点，如若满足，则进行剪枝处理。这样便可以得到对应的最优子树；
5. 将获得的最优子树加入最优子树集合，将使用过的最大值从集合M中剔除；
6. 运行直至集合M为空；
7. 采用交叉验证在最优子树集合中选择出最优子树。

## 解决数据矛盾问题

数据矛盾问题是指如果对于相同属性的两组数据其结果属性值截然矛盾的现象。决策树算法无法借用这样的数据生成决策树模型。当然，在生成了决策树模型之后，便可以通过剪枝之后的数据进行判别。一般采用最大匹配原则，即在叶子节点上的不同结果中选取数目最多的作为预测结果。但是在生成的时候这种模棱两可的叶子节点会给剪枝过程中的计算带来困难，因而本论文提出一种虚拟叶子节点的方式。

虚拟叶子节点是指对于在决策树生成之初就存在的矛盾问题，采用在矛盾的叶子节点上再增添一层新的叶子节点的方式进行解决。当访问到该叶子节点时，其节点个数认为是包含着的所有样本个数，其节点的预测值则采用样本在不同结果属性值的数目作为基础，采用随机概率的方式得到最终的预测值。

经由这种方式，既解决了数据中广泛存在的矛盾问题，也给数据的矛盾现象以合理的解释，从而在总体上兼顾了模型的精确度和数据的保真性。

# 实验验证与结果分析

## 数据集介绍

本篇作业采用的数据集是德国美因茨约翰内斯古腾堡大学医学中心(Institute for Medical Biostatistics, Epidemiology and Informatics (IMBEI), University Medical Center of Johannes Gutenberg University, Mainz, Germany)的癌症病人身份数据集作为目标分类数据集。该数据集的记录来源于多次进行登记的相同个体，通过他们每次登记的各项特征之间的相关程度来确定两项记录是否来源于相同的病人，因而该数据集被称为记录链接模式匹配问题（Record Linkage Comparison Patterns），该问题属于分类预测型问题，因而符合老师对处理数据集的要求。该数据集按照老师要求取源于UCI库。

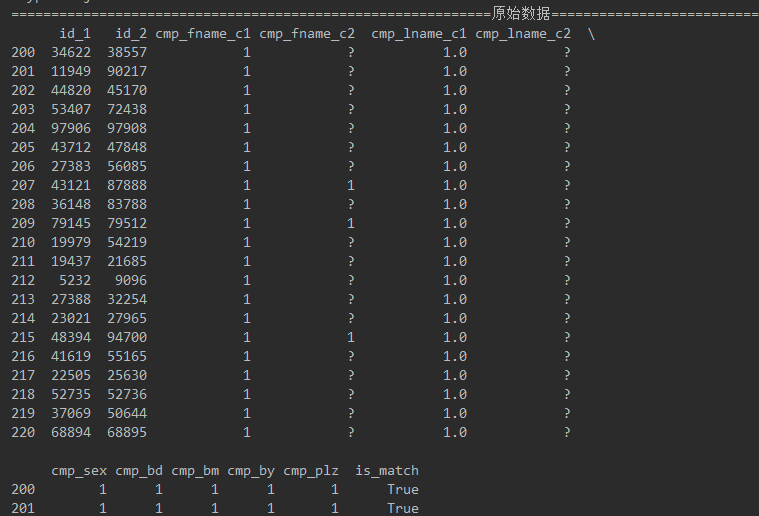


图3.1原始数据图

据图3.1，3.2可知，该数据集的输入属性主要有11个，分别代表第一记录的账号、第二次记录的账号、姓的第一部分的相似程度、姓的第二部分的相似程度、名的第一部分的相似程度、名的第二部分的相似程度、性别的是否相同、出生日是否相同、出生月是否相同、出生年份是否相同、邮政编码是否相同等。输出属性仅有一个，即两个账号是否匹配。

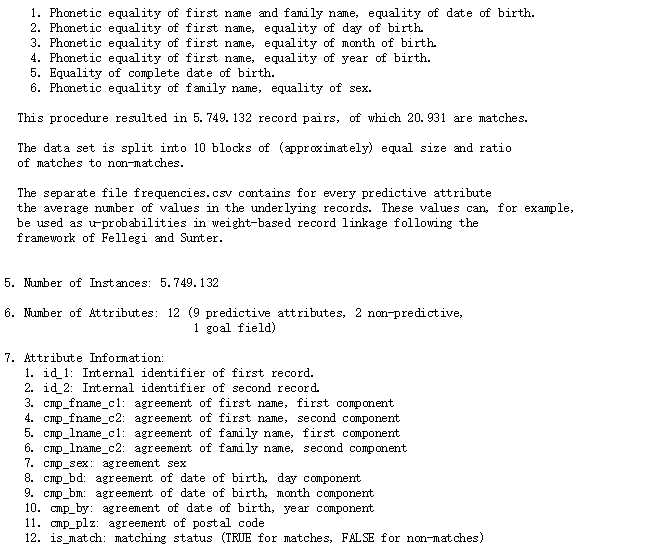


图3.2 数据集属性解释

在这些输入输出的属性特征中可以发现，里面既有连续变化的数值量，也有离散的布尔量，既有完全填满的数据，也有缺省的属性值，因而这个数据集的处理难度并不低，且对数据的处理和整体的策略有一定要求，有助于将课上的知识和额外的知识进行运用。

在进行实际运用时，本作业采用了机器学习中常用的处理方案，将整个数据集分为三个部分：

1. 训练集。用于生成决策树；
2. 验证集。用于剪枝处理，判断模型是否过拟合；
3. 验证集。用于评估模型的错误率、精确程度等等。

并且，由于决策树算法C4.5并非深度神经网络那样的大模型，所以本作业在三个数据集的数据条数划分上采用了不同的比例：我将全部的5749132组数据按照匹配与否的相同频率均分成了十组，并分别采用训练集、测试集、验证集的个数分别为7组、2组、1组，5组、4组、1组，1组、8组、1组等多种比例来验证本算法的鲁棒性和泛化能力，并对是否存在欠拟合问题和过拟合问题以及最佳数据尺度进行了分析。分析结果见3.2。

## 实验验证与结果分析

首先，本实验先采用了数据集中的第一个分块，仅将其分为了训练集和测试集，经过测试，生成的决策树[[7]](#footnote-2)如图3.3，图3.4所示。

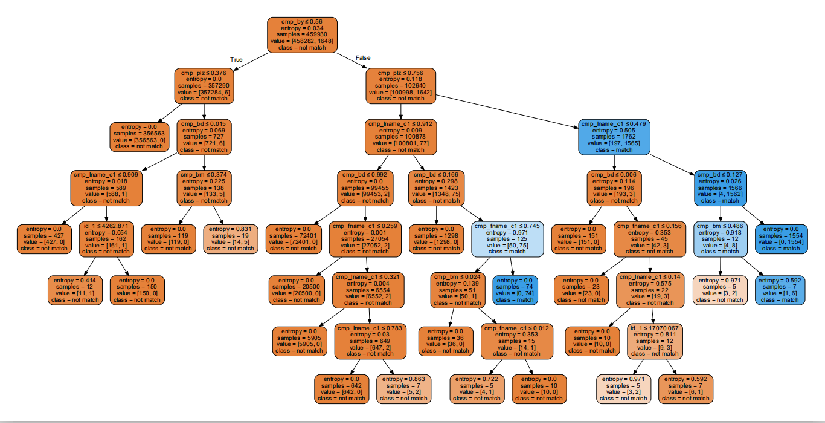


图3.3 决策树掠影（由于决策树较大，所以先放一张掠影，下图是局部放大图）

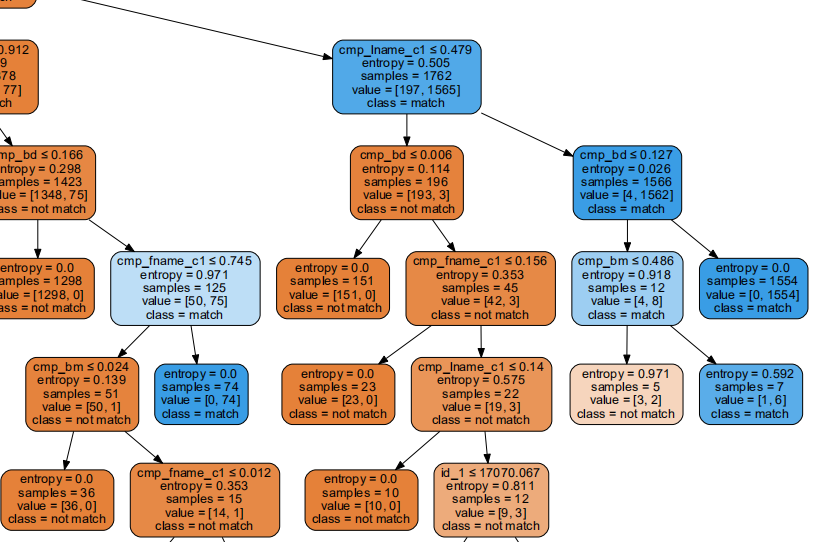


图3.4 决策树局部放大图（每个框里均包括：分裂属性值、信息熵值、样本数、样本在目标属性下的分布，根据分布得到的是否匹配的结论）

注意，对于每一个节点，朝左生成表示True，朝右表示False。节点下的结果表示在最后一行，如若匹配，则是“match”，反之，则是“not match”。由于在程序中我限制了叶子节点的最小样本数目（目的同样也是为了限制决策树的广度），所以图中所有叶子节点包含的最小样本数为5。

利用上述决策树对测试集进行测试，可以得到图3.5所示结果。

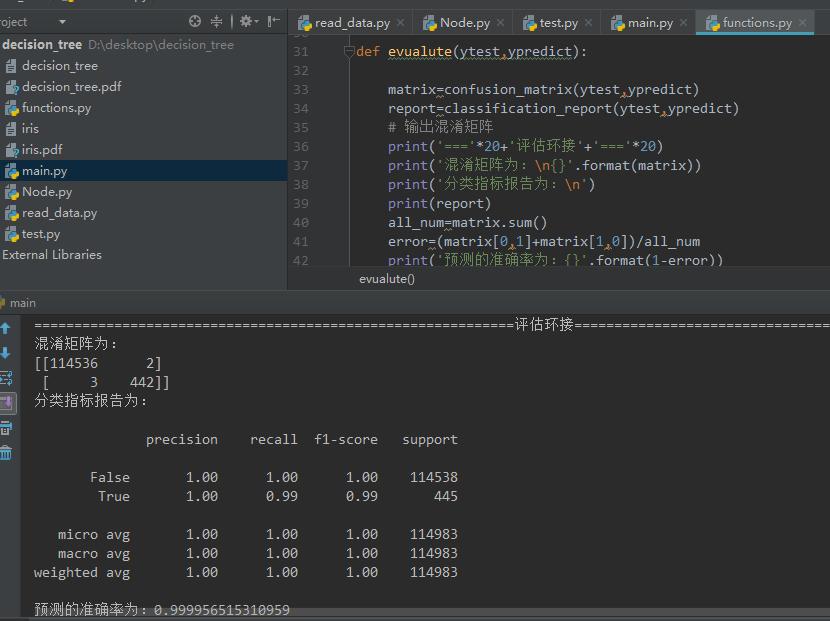


图3.5 测试结果图

可以看出，在被测试的整整11万余条数据中，仅仅有2个数据本来是“不匹配”但是被标记为了“匹配”，有3个数据本来是“匹配”却被分成了不匹配，可以看出，即使是使用决策树算法这种极易过拟合、对属性变化敏感的算法， 经过预处理和剪枝之后算法的精度仍然达到了99.99565153%的程度，可谓是足够强大！

接下来，本实验尝试再次增加样本数目。

当使用上全部的样本时，可以得到图3.6所示的决策树

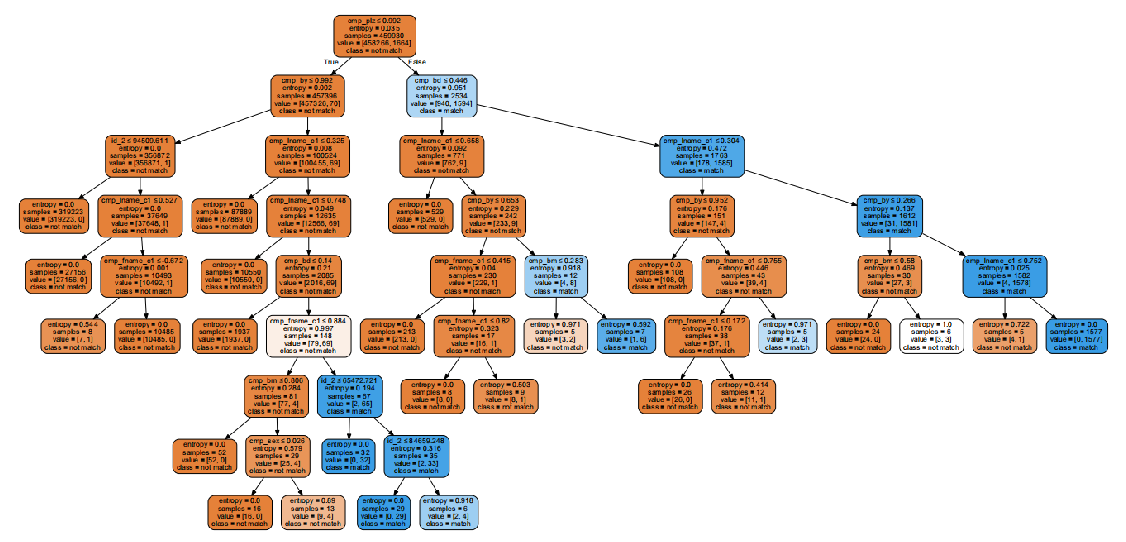


图3.6 使用80%的数据进行训练生成的决策树模型剪影

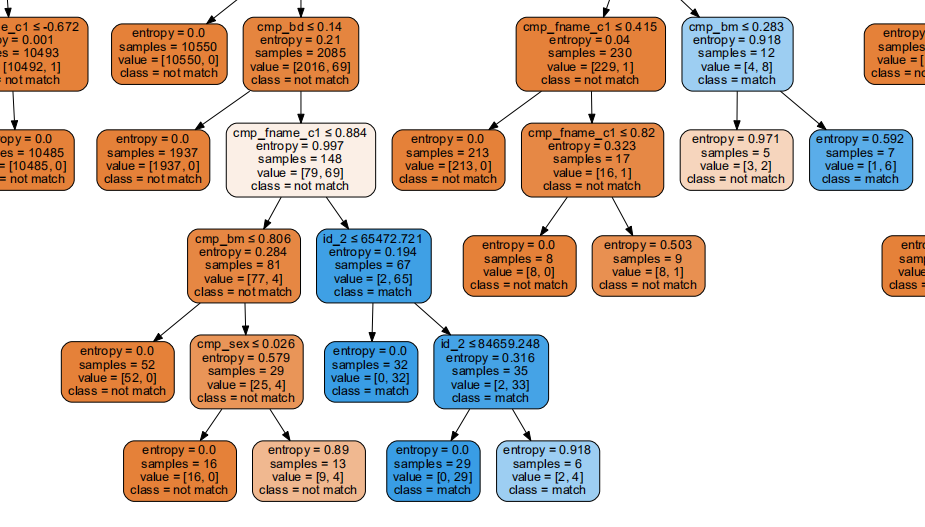


图3.7 决策树局部

可以看出，使用了80%的数据进行训练后，生成的决策树和之前的决策树（见图3.4）在整体结构上十分相似，仅仅是再最后生成叶子节点的部位有些区别。由此可以看出，该决策树具有明显的可解释性（white box），相较于神经网络等黑箱系统更能够给以合理的解释。该模型的评估结果如图3.8所示。

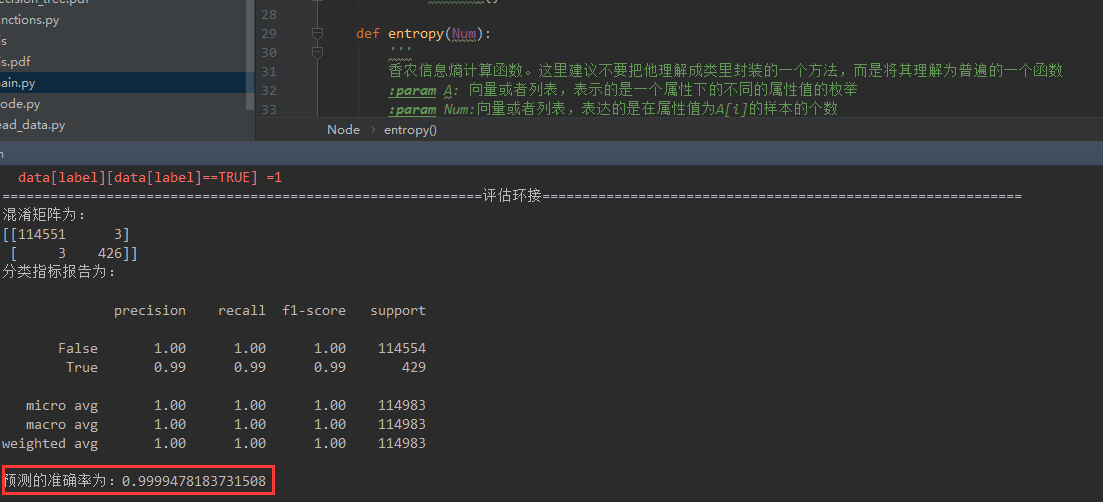


图3.8 运行评估图

从图中可以看出，在测试集上模型的准确率达到了99.9947818%,仅仅有6个样本出现了分类错误。准确率已经极高。

# 总结

## 结题

在认真学习了C4.5决策树算法的各个部分之后，本论文根据现实数据集的具体需求，通过查阅资料确定了目前使用最广的映射到高维度的缺省值处理方法，使得C4.5算法能够实现对数据集的处理。并在此基础上，本论文分析、学习了决策树算法常用的几种剪枝策略，并以此为基础选取了最适合于本数据集的一种，并经过多次调参确定了训练集和测试集的比例选取。

最终，经过处理，本论文的算法实验精度全部超过了99.99%，可以说完满完成了二分类任务。

## 不足之处

本算法当训练样本偏向于某一个特征比例时，可能在实际应用时便会表现不佳，这个问题在本算法中已经通过施加修正因子进行过修正，但是需要多次调参；

本算法难以处理复杂的逻辑，比如异或，这是所有决策树算法的通病。根据“天下没有免费的午餐”的算法原理，决策树算法具有在别处的优势，便会拥有例如在此处的劣势。因而需要对实际的分类任务进行分析，如果逻辑过于复杂，可以考虑更换算法，例如神经网络。

# 参考文献

1. 该数据集按照要求取自UCI，网址为：http://archive.ics.uci.edu/ml/machine-learning-databases/00210/ [↑](#footnote-ref-0)
2. 本结课论文研究算法的所有源代码和训练模型请见：https://github.com/liangzid/DecisionTreeC4.5 [↑](#footnote-ref-1)
3. 《统计学习方法》，李航著。清华大学出版社，2012，3 [↑](#endnote-ref-0)
4. Douglas Robert Stinson; Maura Paterson. 第2.4节“熵”. Cryptography Theory and Practice [密码学理论与实践] [↑](#endnote-ref-1)
5. Quinlan, J. R. C4.5: Programs for Machine Learning. Morgan Kaufmann Publishers, 1993. [↑](#endnote-ref-2)
6. S.B. Kotsiantis, "Supervised Machine Learning: A Review of Classification Techniques", Informatica 31(2007) 249-268, 2007

   此外，还参考了大量的博客和源代码，难以一一列出，表示歉意。

   # 附录

   程序源代码：

   完成源代码请见文件夹附件，或者参考网址：（https://github.com/liangzid/DecisionTreeC4.5）

   下面仅仅给出主函数的程序源代码：

   import read\_data

   import functions

   from sklearn.model\_selection import train\_test\_split

   import sklearn.tree as tree

   import sklearn

   import graphviz as gvz

   train\_path= ['D://desktop//dataset//block\_1.csv','D://desktop//dataset//block\_2.csv',

   'D://desktop//dataset//block\_3.csv','D://desktop//dataset//block\_4.csv',

   'D://desktop//dataset//block\_5.csv','D://desktop//dataset//block\_6.csv',]

   test\_path = ['D://desktop//dataset//block\_7.csv','D://desktop//dataset//block\_8.csv',

   'D://desktop//dataset//block\_9.csv',]

   vali\_path =['D://desktop//dataset//block\_10.csv',]

   target\_label='is\_match'

   # 读取数据

   DataSet=read\_data.ReadAData(train\_path[0])

   for i in range(9):

   ii=i+2

   path1='D://desktop//dataset//block\_'

   path2='.csv'

   dataset=read\_data.ReadAData(path1+str(ii)+path2)

   DataSet.append(dataset)

   DataSett=functions.fromNaNToLable(DataSet)

   #从数据中提取训练数据和标签

   X=DataSett.drop(target\_label,axis=1)

   Y=DataSett[target\_label]

   #将数据分为训练集和测试集

   Xtrain,Xtest,ytrain,ytest=train\_test\_split(X,Y,test\_size=0.2)

   #print('++++++++++++++++++++++++++++++++++++++++++++++++++++++++')

   #print(DataSett.dtypes)

   #============================================生成决策树================================================

   clf=tree.DecisionTreeClassifier(criterion='entropy', #采用信息熵的计算方式

   splitter='random', #随机在部分划分点中寻找局部最优

   max\_features=None, #最大特征数

   max\_depth=100, #最大深度

   min\_samples\_split=5, #分割所需要的最小样本数

   min\_samples\_leaf=5, #身为叶子所需要的最小样本数

   min\_weight\_fraction\_leaf=0,

   max\_leaf\_nodes=None,

   min\_impurity\_decrease=0,

   #min\_impurity\_split=0

   )

   clf=clf.fit(Xtrain,ytrain)

   #======================================保存决策树=====================================================

   dot\_data=tree.export\_graphviz(clf, out\_file=None,

   feature\_names=DataSet.columns[:-1],

   class\_names=['not match','match'],

   filled=True,rounded=True,

   special\_characters=True)

   graph=gvz.Source(dot\_data)

   graph.render('decision\_tree')

   #进行预测

   predict=clf.predict(Xtest)

   #进行评估

   functions.evualute(ytest,predict) [↑](#endnote-ref-3)
7. 决策树的绘制使用了python开源数据可视化库graphviz，感谢它让我画出了决策树 [↑](#footnote-ref-2)