Artificial Intelligence

决策树扩展问题

顾涵雪
刘　洁
庞婧璇
涂剑凯

Outlines

➢Part I: Missing data 缺失值处理

- Given a decision tree, classify a example with missing data
- How to generate a tree with missing data

➢Part II: Continuous and integer-valued 连续值处理

- Introduce to C4.5
- Improvement I: Pre-Pruning
- Improvement II: Segment

➢Part III: Combination and improvement 连续值缺失与改进

Part I: Missing data

Missing data: In many domains, not all the attribute values will be known for every example. The values might have gone unrecorded, or they might be too expensive to obtain. This gives rise to two problems: First, given a complete decision tree, how should one classify an example that is missing one of the test attributes? Second, how should one modify the information-gain formula when some examples have unknown values for the attribute?

Two Stages
- the training process： the missing data (training set) are used to construct the tree
- the testing process： the tree is put into use and classify the testing data

Part I: Missing data – Training Process

- **Main methods**

1.exclude the missing value:

a) Complete case method: deletes all observations that contain missing values in any of the predictors in the training phase

b) Complete variable method: simply deletes all variables that contain missing values

2. separate class: treats the missing values as a new class of the predictor

3. filling in missing attribute values :

a) Surrogate split: Uses surrogate variable within a node if the variable for the next split contains missing values, missing data imputation

b) Missing data imputation: EM algorithm, k-means

c) Grand mode imputation: imputes the missing value with the grand mode of that of that variable if it is categorical

4. looking at the probability distribution of values of attributes

Probabilistic split: split the example with probability

Part I: Missing data – Training Process

- **Separate Class**

  Method: To treats the missing values as **a new class (category)** of the predictor

  Advantage: The **best** method to use **when** missing **data exist both the training phase and the testing phase**

- **Imputation Method**

  Method: Mean imputation / random imputation / regression imputation / k-means / **EM algorithm**

  Missing-data mechanisms:

  1. missing completely at random
     $$f(M|Y,\theta) = f(M|\theta)$$
  2. missing at random
     $$f(M|Y,\theta) = f(M|Y_{obs},\theta)$$
  3. not missing at random

  1. Define the distribution function and initialize the parameter
  2. E step: Calculate the missing data
  3. M step: Maximum Likelihood Estimation to calculate parameter
  4. Iteration

Part I: Missing data – Training Process

- **Probability split:**

Given the training set D, and the attribute a. Let $\widetilde{D}$ represent the subset of D which has no missing value on the attribute a.

We assume that a has V options $\{a^1, a^2, \ldots, a^V\}$, thus we let $\widetilde{D^v}$ represent the subset of $\widetilde{D}$ which the attribute a has the value a^v. And we let $\widetilde{D_k}$ represent the subset of $\widetilde{D}$ which belongs to the k class($k = 1,2, \ldots, K$).

And we give each example x a weight w_x, which was initially set to 1.

We define:

$$\rho = \frac{\sum_{x \in \widetilde{D}} w_x}{\sum_{x \in D} w_x},$$ ρ represents the ratio of none missing data

$$\widetilde{p_k} = \frac{\sum_{x \in \widetilde{D_k}} w_x}{\sum_{x \in \widetilde{D}} w_x},$$ $\widetilde{p_k}$ represents ratio of the k class of the none-missing sample

$$\widetilde{r_v} = \frac{\sum_{x \in \widetilde{D^v}} w_x}{\sum_{x \in \widetilde{D}} w_x},$$ $\widetilde{r_v}$ represents ratio of the samples with a^v value of the none-missing sample

Part I: Missing data – Training Process

Thus, we can give the modified information-gain formula:

$$Gain(D,a) = \rho \times Gain\left(\widetilde{D},a\right) = \rho \times \left(Ent\left(\widetilde{D}\right) - \sum_{v=1}^{V} \widetilde{r_v} Ent(\widetilde{D^v})\right)$$

$$Ent\left(\widetilde{D}\right) = -\sum_{k=1}^{K} \widetilde{p_k} \log \widetilde{p_k}$$

After calculating the modified information-gain, we apply the same method of ID3, another problem is how to divide the samples with missing data into branches:

It's inappropriate to randomly group this sample into one class, that's why we define the weight of each example.

We put this example into each class but with different weight: $\widetilde{r_v} \cdot w_x$

Part I: Missing data – Example

- Example: 数据摘自 周志华 机器学习 （西瓜书）

编号	色泽	根蒂	敲声	纹理	脐部	触感	好瓜
1	–	蜷缩	浊响	清晰	凹陷	硬滑	是
2	乌黑	蜷缩	沉闷	清晰	凹陷	–	是
3	乌黑	蜷缩	–	清晰	凹陷	硬滑	是
4	青绿	蜷缩	沉闷	清晰	凹陷	硬滑	是
5	–	蜷缩	浊响	清晰	凹陷	硬滑	是
6	青绿	稍蜷	浊响	清晰	–	软粘	是
7	乌黑	稍蜷	浊响	稍糊	稍凹	软粘	是
8	乌黑	稍蜷	浊响	–	稍凹	硬滑	是
9	乌黑	–	沉闷	稍糊	稍凹	硬滑	否
10	青绿	硬挺	清脆	–	平坦	软粘	否
11	浅白	硬挺	清脆	模糊	平坦	–	否
12	浅白	蜷缩	–	模糊	平坦	软粘	否
13	–	稍蜷	浊响	稍糊	凹陷	硬滑	否
14	浅白	稍蜷	沉闷	稍糊	凹陷	硬滑	否
15	乌黑	稍蜷	浊响	清晰	–	软粘	否
16	浅白	蜷缩	浊响	模糊	平坦	硬滑	否
17	青绿	–	沉闷	稍糊	稍凹	硬滑	否

D 为左边的全集

针对‘色泽’属性，$\tilde{D} = \{2,3,4,6,7,8,9,10,11,12,14,15,16,17\}$

K有两个取值：是好瓜 / 不是好瓜

V有三个取值：乌黑、青绿、浅白

初始化各样本权重为1

$$\rho = \frac{\sum_{x\in\tilde{D}} w_x}{\sum_{x\in D} w_x} \qquad \widetilde{p_k} = \frac{\sum_{x\in\widetilde{D_k}} w_x}{\sum_{x\in\tilde{D}} w_x} \qquad \widetilde{r_v} = \frac{\sum_{x\in\widetilde{D^v}} w_x}{\sum_{x\in\tilde{D}} w_x}$$

$$Gain(D,a) = \rho \times Gain\left(\tilde{D},a\right) = \rho \times \left(Ent\left(\tilde{D}\right) - \sum_{v=1}^{V} \widetilde{r_v} Ent(\widetilde{D^v})\right)$$

$$Ent\left(\tilde{D}\right) = -\sum_{k=1}^{K} \widetilde{p_k} \log \widetilde{p_k}$$

Part I: Missing data – Example

➢ 色泽

$$\rho = \frac{\sum_{x\in\tilde{D}} w_x}{\sum_{x\in D} w_x} = \frac{14}{17} \qquad \widetilde{p_1} = \frac{\sum_{x\in\widetilde{D_1}} w_x}{\sum_{x\in\tilde{D}} w_x} = \frac{6}{14}\text{(是好瓜)};\ \widetilde{p_2} = \frac{\sum_{x\in\widetilde{D_2}} w_x}{\sum_{x\in\tilde{D}} w_x} = \frac{8}{14}\text{(不是好瓜)}$$

$$\widetilde{r_v} = \frac{\sum_{x\in\widetilde{D^v}} w_x}{\sum_{x\in\tilde{D}} w_x} = \left(\frac{6}{14}, \frac{4}{14}, \frac{4}{14}\right)\text{(乌黑、青绿、浅白)}$$

$$Ent(\tilde{D}) = -\sum_{k=1}^{K} \widetilde{p_k} \log \widetilde{p_k} = -\frac{6}{14}\log\frac{6}{14} - \frac{8}{14}\log\frac{8}{14} = 0.985$$

$$Ent(\widetilde{D^1}) = -\frac{4}{6}\log\frac{4}{6} - \frac{2}{6}\log\frac{2}{6} = 0.918$$
$$Ent(\widetilde{D^2}) = -\frac{2}{4}\log\frac{2}{4} - \frac{2}{4}\log\frac{2}{4} = 1$$
$$Ent(\widetilde{D^3}) = -0\log 0 - 1\log 1 = 0$$

$$Gain(D, \text{色泽}) = \rho \times Gain(\tilde{D}, a) = \rho \times \left(Ent(\tilde{D}) - \sum_{v=1}^{V} \widetilde{r_v} Ent(\widetilde{D^v})\right)$$
$$= \frac{14}{17}\left(0.985 - \frac{6}{14}\cdot 0.918 - \frac{4}{14}\cdot 1 - \frac{4}{14}\cdot 0\right) = \mathbf{0.252}$$

编号	色泽	根蒂	敲声	纹理	脐部	触感	好瓜
1	–	蜷缩	浊响	清晰	凹陷	硬滑	是
2	乌黑	蜷缩	沉闷	清晰	凹陷	–	是
3	乌黑	蜷缩	–	清晰	凹陷	硬滑	是
4	青绿	蜷缩	沉闷	清晰	凹陷	硬滑	是
5	–	蜷缩	浊响	清晰	凹陷	硬滑	是
6	青绿	稍蜷	浊响	清晰	–	软粘	是
7	乌黑	稍蜷	浊响	稍糊	稍凹	软粘	是
8	乌黑	稍蜷	浊响	–	稍凹	硬滑	是
9	乌黑	–	沉闷	稍糊	稍凹	硬滑	否
10	青绿	硬挺	清脆	–	平坦	软粘	否
11	浅白	硬挺	清脆	模糊	平坦	–	否
12	浅白	蜷缩	–	模糊	平坦	软粘	否
13	–	稍蜷	浊响	稍糊	凹陷	硬滑	否
14	浅白	稍蜷	沉闷	稍糊	凹陷	硬滑	否
15	乌黑	稍蜷	浊响	清晰	–	软粘	否
16	浅白	蜷缩	浊响	模糊	平坦	硬滑	否
17	青绿	–	沉闷	稍糊	稍凹	硬滑	否

Part I: Missing data – Example

- **根蒂:** $Gain(D, 根蒂) = \rho \times Gain(\tilde{D}, a) = \rho \times \left(Ent(\tilde{D}) - \sum_{v=1}^{V} \tilde{r}_v Ent(\tilde{D}^v) \right)$

  $= \frac{15}{17}\left(0.997 - \frac{7}{15} \cdot 0.863 - \frac{6}{15} \cdot 1 - \frac{2}{15} \cdot 0\right) = \mathbf{0.171}$

- **敲声:** $Gain(D, 敲声) = \rho \times Gain(\tilde{D}, a) = \rho \times \left(Ent(\tilde{D}) - \sum_{v=1}^{V} \tilde{r}_v Ent(\tilde{D}^v) \right)$

  $= \frac{15}{17}\left(0.997 - \frac{8}{15} \cdot 0.954 - \frac{5}{15} \cdot 0.971 - \frac{2}{15} \cdot 0\right) = \mathbf{0.145}$

- **纹理:** $Gain(D, 纹理) = \rho \times Gain(\tilde{D}, a) = \rho \times \left(Ent(\tilde{D}) - \sum_{v=1}^{V} \tilde{r}_v Ent(\tilde{D}^v) \right)$

  $= \frac{15}{17}\left(0.997 - \frac{7}{15} \cdot 0.592 - \frac{5}{15} \cdot 0.722 - \frac{3}{15} \cdot 0\right) = \mathbf{0.424}$

- **脐部:** $Gain(D, 脐部) = \rho \times Gain(\tilde{D}, a) = \rho \times \left(Ent(\tilde{D}) - \sum_{v=1}^{V} \tilde{r}_v Ent(\tilde{D}^v) \right)$

  $= \frac{15}{17}\left(0.997 - \frac{7}{15} \cdot 0.863 - \frac{4}{15} \cdot 1 - \frac{4}{15} \cdot 0\right) = \mathbf{0.289}$

- **触感:** $Gain(D, 触感) = \rho \times Gain(\tilde{D}, a) = \rho \times \left(Ent(\tilde{D}) - \sum_{v=1}^{V} \tilde{r}_v Ent(\tilde{D}^v) \right)$

  $= \frac{15}{17}\left(0.997 - \frac{10}{15} \cdot 1 - \frac{5}{15} \cdot 0.971\right) = \mathbf{0.006}$

编号	色泽	根蒂	敲声	纹理	脐部	触感	好瓜
1	–	蜷缩	浊响	清晰	凹陷	硬滑	是
2	乌黑	蜷缩	沉闷	清晰	凹陷	–	是
3	乌黑	蜷缩	–	清晰	凹陷	硬滑	是
4	青绿	蜷缩	沉闷	清晰	凹陷	硬滑	是
5	–	蜷缩	浊响	清晰	凹陷	硬滑	是
6	青绿	稍蜷	浊响	清晰	–	软粘	是
7	乌黑	稍蜷	浊响	稍糊	稍凹	软粘	是
8	乌黑	稍蜷	浊响	–	稍凹	硬滑	是
9	乌黑	–	沉闷	稍糊	稍凹	硬滑	否
10	青绿	硬挺	清脆	–	平坦	软粘	否
11	浅白	硬挺	清脆	模糊	平坦	–	否
12	浅白	蜷缩	–	模糊	平坦	软粘	否
13	–	稍蜷	浊响	稍糊	凹陷	硬滑	否
14	浅白	稍蜷	沉闷	稍糊	凹陷	硬滑	否
15	乌黑	稍蜷	浊响	清晰	–	软粘	否
16	浅白	蜷缩	浊响	模糊	平坦	硬滑	否
17	青绿	–	沉闷	稍糊	稍凹	硬滑	否

- **色泽:**

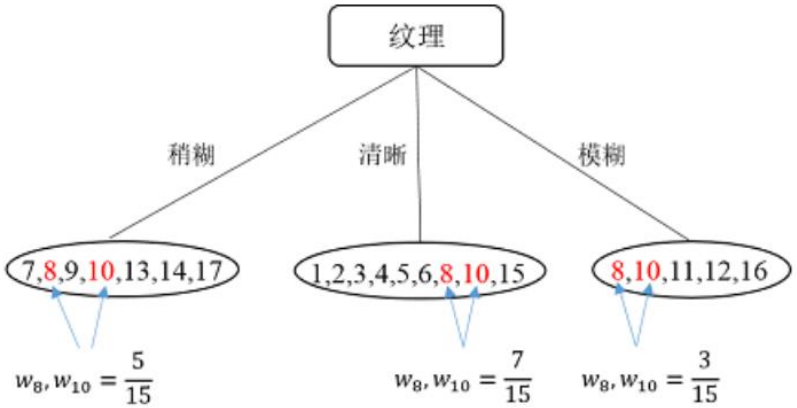
$Gain(D, 色泽) = \rho \times Gain(\tilde{D}, a) = \rho \times \left(Ent(\tilde{D}) - \sum_{v=1}^{V} \tilde{r}_v Ent(\tilde{D}^v) \right)$

$= \frac{14}{17}\left(0.985 - \frac{6}{14} \cdot 0.918 - \frac{4}{14} \cdot 1 - \frac{4}{14} \cdot 0\right) = \mathbf{0.252}$

第一层扩展 **纹理**

Part I: Missing data – Example

$$\tilde{r}_1 = \frac{5}{15} \qquad \tilde{r}_2 = \frac{7}{15} \qquad \tilde{r}_3 = \frac{3}{15}$$



编号	色泽	根蒂	敲声	纹理	脐部	触感	好瓜
1	–	蜷缩	浊响	清晰	凹陷	硬滑	是
2	乌黑	蜷缩	沉闷	清晰	凹陷	–	是
3	乌黑	蜷缩	–	清晰	凹陷	硬滑	是
4	青绿	蜷缩	沉闷	清晰	凹陷	硬滑	是
5	–	蜷缩	浊响	清晰	凹陷	硬滑	是
6	青绿	稍蜷	浊响	清晰	–	软粘	是
7	乌黑	稍蜷	浊响	稍糊	稍凹	软粘	是
8	乌黑	稍蜷	浊响	–	稍凹	硬滑	是
9	乌黑	–	沉闷	稍糊	稍凹	硬滑	否
10	青绿	硬挺	清脆	–	平坦	软粘	否
11	浅白	硬挺	清脆	模糊	平坦	–	否
12	浅白	蜷缩	–	模糊	平坦	软粘	否
13	–	稍蜷	浊响	稍糊	凹陷	硬滑	否
14	浅白	稍蜷	沉闷	稍糊	凹陷	硬滑	否
15	乌黑	稍蜷	浊响	清晰	–	软粘	否
16	浅白	蜷缩	浊响	模糊	平坦	硬滑	否
17	青绿	–	沉闷	稍糊	稍凹	硬滑	否

Part I: Missing data – Example

Weight: $w = \{1, \frac{5}{15}, 1, \frac{5}{15}, 1, 1, 1\}$

编号	色泽	根蒂	敲声	纹理	脐部	触感	好瓜
7	乌黑	稍蜷	浊响	稍糊	稍凹	软粘	是
8	乌黑	稍蜷	浊响	–	稍凹	硬滑	是
9	乌黑	–	沉闷	稍糊	稍凹	硬滑	否
10	青绿	硬挺	清脆	–	平坦	软粘	否
13	–	稍蜷	浊响	稍糊	凹陷	硬滑	否
14	浅白	稍蜷	沉闷	稍糊	凹陷	硬滑	否
17	青绿	–	沉闷	稍糊	稍凹	硬滑	否

- **色泽:**

$$\rho = \frac{\sum_{x\in\tilde{D}} w_x}{\sum_{x\in D} w_x} = \frac{14}{17} \qquad \widetilde{p_1} = \frac{\sum_{x\in\widetilde{D_1}} w_x}{\sum_{x\in\tilde{D}} w_x} = \frac{4}{14}\text{(是好瓜)};\ \widetilde{p_2} = \frac{\sum_{x\in\widetilde{D_2}} w_x}{\sum_{x\in\tilde{D}} w_x} = \frac{10}{14}\text{(不是好瓜)}$$

$$\widetilde{r_v} = \frac{\sum_{x\in\widetilde{D^v}} w_x}{\sum_{x\in\tilde{D}} w_x} = \left(\frac{7}{14}, \frac{4}{14}, \frac{3}{14}\right)\text{(乌黑、青绿、浅白)}$$

$$Ent(\tilde{D}) = -\sum_{k=1}^{K} \widetilde{p_k} \log \widetilde{p_k} = -\frac{4}{14}\log\frac{4}{14} - \frac{10}{14}\log\frac{10}{14} = 0.863$$

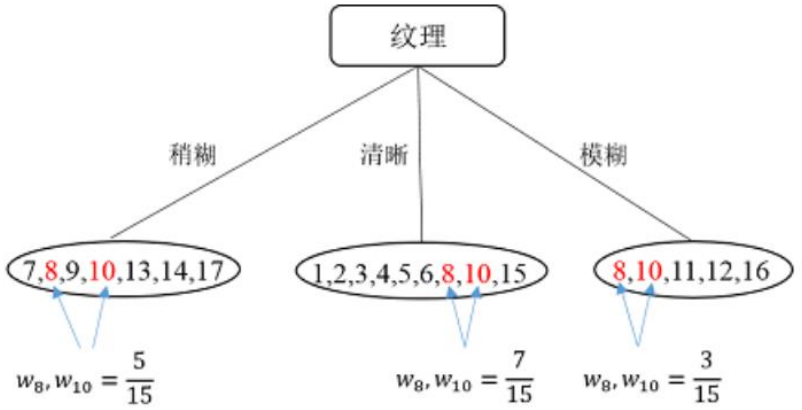
$$Ent(\widetilde{D^1}) = -\frac{4}{7}\log\frac{4}{7} - \frac{3}{7}\log\frac{3}{7} = 0.985$$

$$Ent(\widetilde{D^2}) = 0$$

$$Ent(\widetilde{D^3}) = 0$$

$$Gain(D, \text{色泽}) = \rho \times Gain(\tilde{D}, a) = \rho \times \left(Ent(\tilde{D}) - \sum_{v=1}^{V} \widetilde{r_v} Ent(\widetilde{D^v})\right)$$

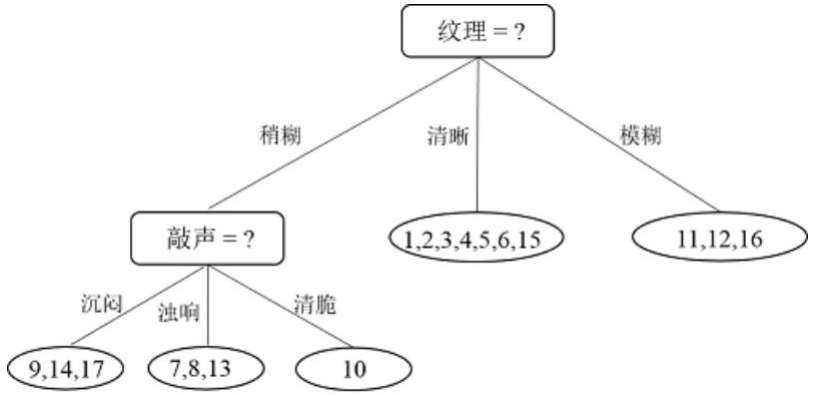
$$= \frac{14}{17}\left(0.863 - \frac{7}{14}\cdot 0.985 - \frac{4}{14}\cdot 0 - \frac{3}{14}\cdot 0\right) = \mathbf{0.305}$$



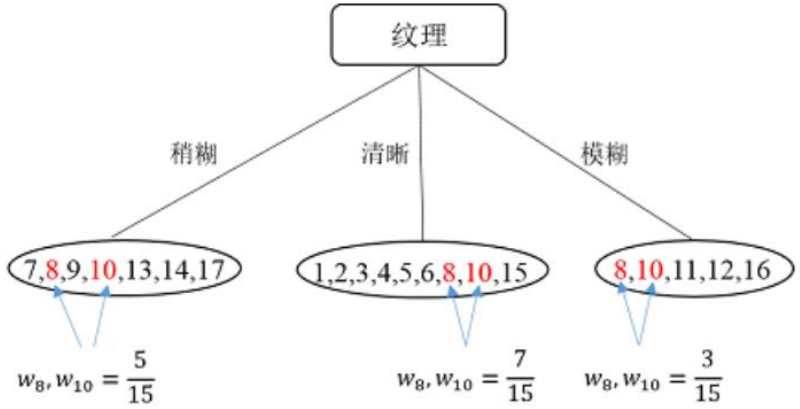
Part I: Missing data – Example

- **色泽:** $Gain(D, \text{色泽}) = \frac{14}{17}\left(0.863 - \frac{7}{14}\cdot 0.985 - \frac{4}{14}\cdot 0 - \frac{3}{14}\cdot 0\right) = \mathbf{0.305}$
- **根蒂:** $Gain(D, \text{根蒂}) = \frac{11}{17}\left(0.946 - \frac{10}{11}\cdot 0.971 - \frac{1}{11}\cdot 0\right) = \mathbf{0.041}$
- **敲声:** $Gain(D, \text{敲声}) = 1\cdot\left(0.787 - \frac{7}{17}\cdot 0.985 - \frac{9}{17}\cdot 0 - \frac{1}{17}\cdot 0\right) = \mathbf{0.381}$
- **脐部:** $Gain(D, \text{脐部}) = 1\cdot\left(0.787 - \frac{10}{17}\cdot 0.971 - \frac{6}{17}\cdot 0 - \frac{1}{17}\cdot 0\right) = \mathbf{0.216}$
- **触感:** $Gain(D, \text{触感}) = 1\cdot\left(0.787 - \frac{4}{17}\cdot 0.811 - \frac{13}{17}\cdot 0.391\right) = \mathbf{0.297}$

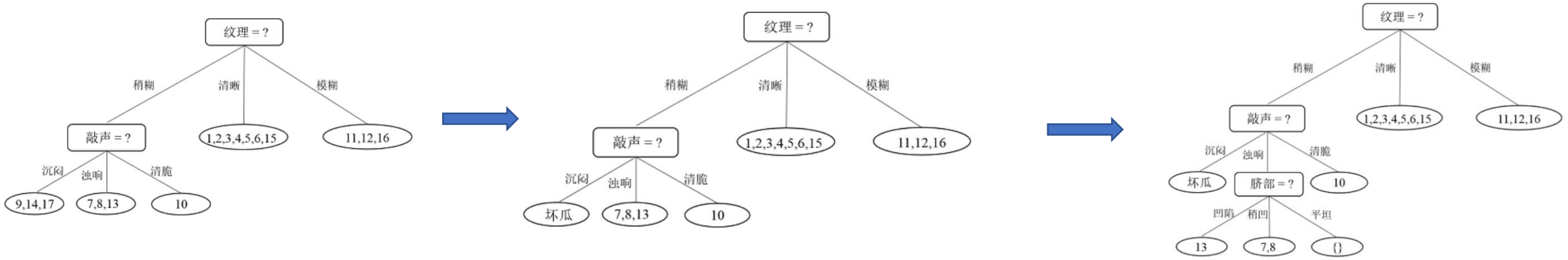
依照 **敲声** 扩展



编号	色泽	根蒂	敲声	纹理	脐部	触感	好瓜
7	乌黑	稍蜷	浊响	稍糊	稍凹	软粘	是
8	乌黑	稍蜷	浊响	–	稍凹	硬滑	是
9	乌黑	–	沉闷	稍糊	稍凹	硬滑	否
10	青绿	硬挺	清脆	–	平坦	软粘	否
13	–	稍蜷	浊响	稍糊	凹陷	硬滑	否
14	浅白	稍蜷	沉闷	稍糊	凹陷	硬滑	否
17	青绿	–	沉闷	稍糊	稍凹	硬滑	否

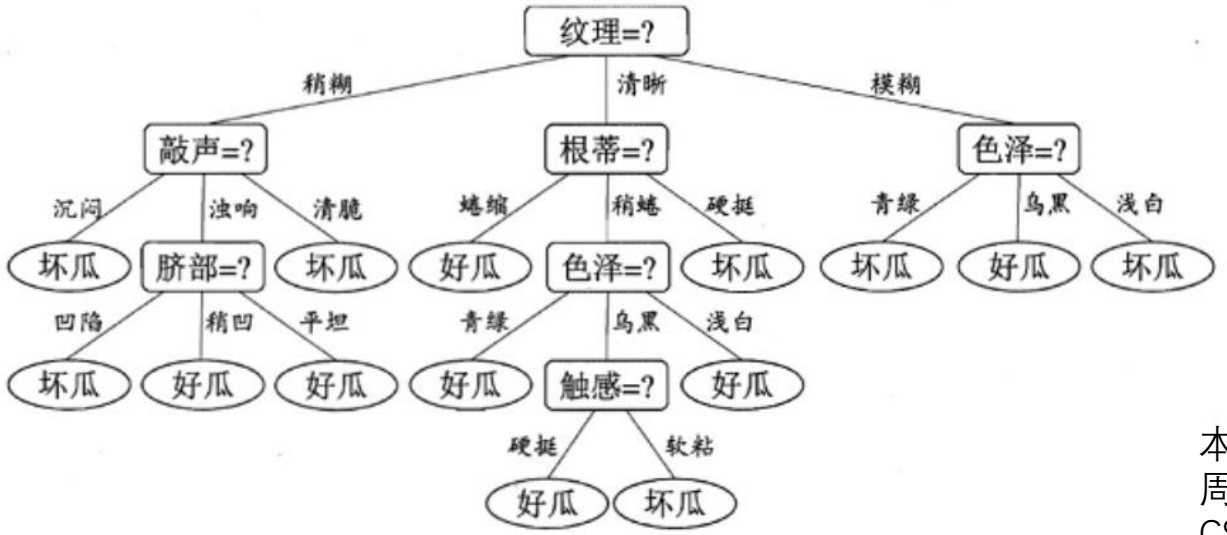


Part I: Missing data – Example



平坦 为空，则以父节点中多数类别确定为**好瓜**

迭代往复：



本示例参考:
周志华，《机器学习》；
CSDN博客，决策树（decision tree）（四）——缺失值处理
https://blog.csdn.net/u012328159/article/details/79413610

Part I: Missing data - Codes Implementation

➢ Codes / Implementation

Decision tree is an traditional algorithm. It's easy to find the implantation code based on ID3, but the one considering missing data with the probability split method is seldom.
So we improved the codes base on traditional ID3 algorithm.

```
01. def createDataSet():
02.     dataSet = [[-1, 1, 1, 1, 1, 1, 'yes'],          #数据集
03.                [1, 1, 2, 1, 1, -1, 'yes'],          # -1 代表数据缺失
04.                [1, 1, -1, 1, 1, 1, 'yes'],
05.                [2, 1, 2, 1, 1, 1, 'yes'],
06.                [-1, 1, 1, 1, 1, 1, 'yes'],
07.                [2, 2, 1, 1, -1, 2, 'yes'],
08.                [1, 2, 1, 2, 2, 2, 'yes'],
09.                [1, 2, 1, -1, 2, 1, 'yes'],
10.                [1, -1, 2, 2, 2, 1, 'no'],
11.                [2, 3, 3, -1, 3, 2, 'no'],
12.                [3, 3, 3, 3, 3, -1, 'no'],
13.                [3, 1, -1, 3, 3, 2, 'no'],
14.                [-1, 2, 1, 2, 1, 1, 'no'],
15.                [3, 2, 2, 2, 1, 1, 'no'],
16.                [1, 2, 1, 1, -1, 2, 'no'],
17.                [3, 1, 1, 3, 3, 1, 'no'],
18.                [2, -1, 2, 2, 2, 1, 'no']]
19.     '''''
20.     色泽：1：乌黑；2：青绿；3：浅白
21.     根蒂：1：蜷缩；2：稍蜷；3：硬挺
22.     敲声：1：浊响；2：沉闷；3：清脆
23.     纹理：1：清晰；2：稍糊；3：模糊
24.     脐部：1：凹陷；2：稍凹；3：平坦
25.     触感：1：硬滑；2：软粘
26.     '''
27.     labels = ['色泽', '根蒂', '敲声', '纹理', '脐部', '触感']        #分类属性
28.
29.     num = len(dataSet)
30.     weights = [1 for i in range(1,num+1)]
31.     return dataSet, labels,weights                  #返回数据集和分类属性和权重
```

```
01. def createTree(dataSet, labels, featLabels, weights):
02.     classList = [example[-1] for example in dataSet]              #取分类标签
03.     if classList.count(classList[0]) == len(classList):           #如果类别完全相同则停止继续划分
04.         return classList[0]
05.     if len(dataSet[0]) == 1:                                      #遍历完所有特征时返回出现次数最多的类标签
06.         return majorityCnt(classList)
07.     bestFeat, bestr = chooseBestFeatureToSplit(dataSet, weights)                    #选择最优特征
08.     bestFeatLabel = labels[bestFeat]                              #最优特征的标签
09.     featLabels.append(bestFeatLabel)
10.     myTree = {bestFeatLabel:{}}                                   #根据最优特征的标签生成树
11.     del(labels[bestFeat])                                         #删除已经使用特征标签
12.     featValues = [example[bestFeat] for example in dataSet]       #得到训练集中所有最优特征的属性值
13.     ## 有缺失的情况下会出现-1，想办法解决
14.     uniqueVals = set(featValues)                                  #去掉重复的属性值
15.     if -1 in uniqueVals:
16.         uniqueVals.remove(-1)
17.     for value in uniqueVals:
18.         subLabels=labels[:]
19.         #递归调用函数createTree(),遍历特征，创建决策树。
20.         subdataSet, subweights = splitDataSet(dataSet, bestFeat, value, weights, bestr)
21.         myTree[bestFeatLabel][value] = createTree(subdataSet, subLabels, featLabels, subweights)
22.     return myTree
```

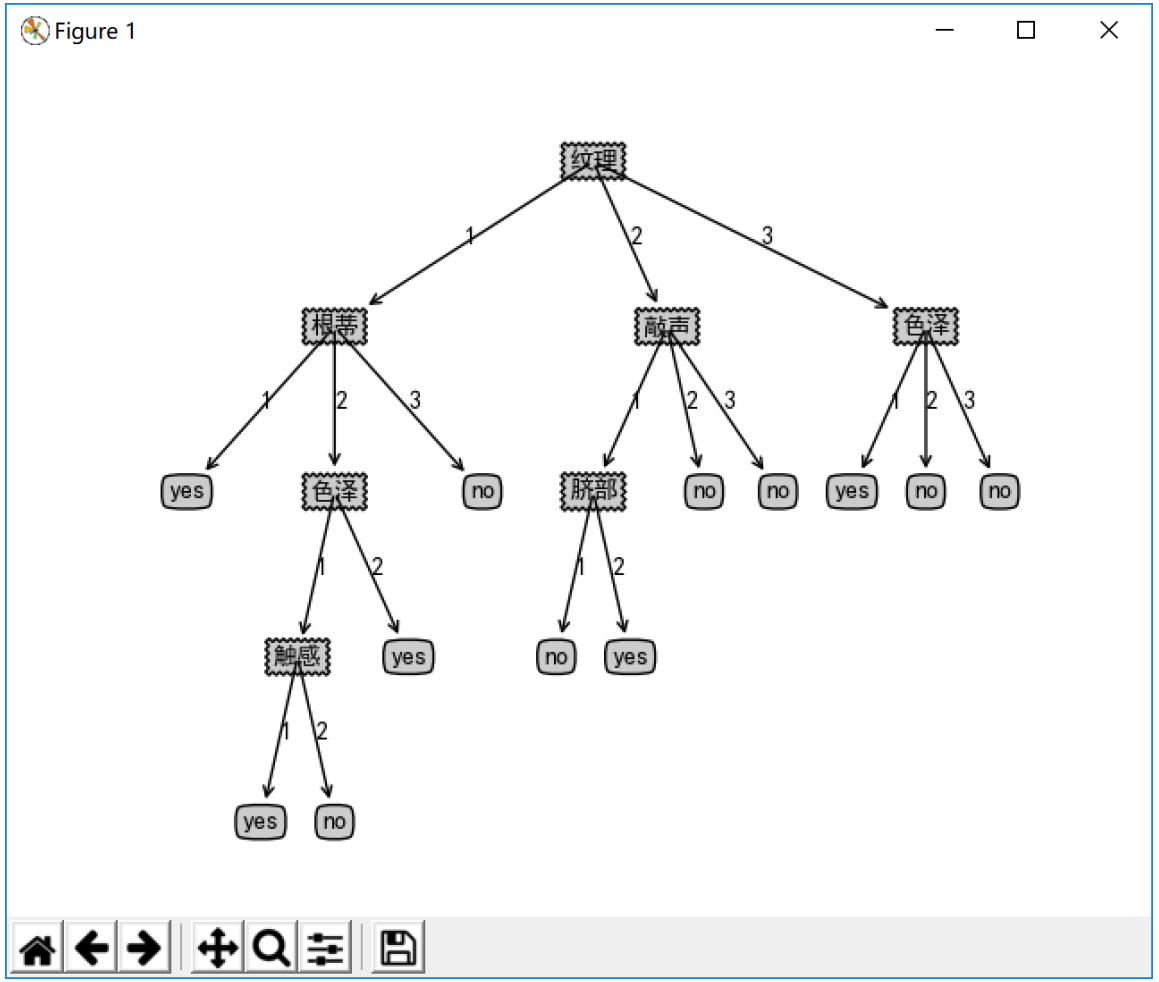
Part I: Missing data - Codes Implementation

➢ **Codes / Implementation**

```
01. def chooseBestFeatureToSplit(dataSet, weights):
02.     numFeatures = len(dataSet[0]) - 1                    #特征数量
03.
04.     bestInfoGain = 0.0                                   #信息增益
05.     bestFeature = -1                                     #最优特征的索引值
06.     for i in range(numFeatures):                         #遍历所有特征
07.         #获取dataSet的第i个所有特征
08.         nmdataSet, nmweights = selectNoMissingData(dataSet,i,weights)
09.         rho = sum(nmweights) / sum(weights)
10.         baseEntropy = calcShannonEnt(nmdataSet, nmweights)              #计算数据集的香农熵
11.
12.         featList = [example[i] for example in nmdataSet]
13.         uniqueVals = set(featList)                       #创建set集合{},元素不可重复
14.         newEntropy = 0.0                                 #经验条件熵
15.         flag = 0
16.         num = len(uniqueVals)
17.         r = [1 for i in range(1,num+1)]
18.         for value in uniqueVals:                         #计算信息增益
19.             subDataSet, subweights = splitDataSet(nmdataSet, i, value, nmweights)        #subDataSet划分后的子集
20.             r[flag] = sum(subweights) / float(sum(nmweights))          #计算子集的概率
21.             newEntropy += r[flag] * calcShannonEnt(subDataSet,subweights)  #根据公式计算经验条件熵
22.             flag += 1
23.         infoGain = rho * ( baseEntropy - newEntropy )                    #信息增益
24.         print("第%d个特征的增益为%.3f" % (i, infoGain))            #打印每个特征的信息增益
25.         if (infoGain > bestInfoGain):                    #计算信息增益
26.             bestInfoGain = infoGain                      #更新信息增益，找到最大的信息增益
27.             bestFeature = i                              #记录信息增益最大的特征的索引值
28.             bestr = r
29.     return bestFeature, bestr                                       #返回信息增益最大的特征的索引值
```

Part I: Missing data - Codes Implementation

- Codes / Implementation

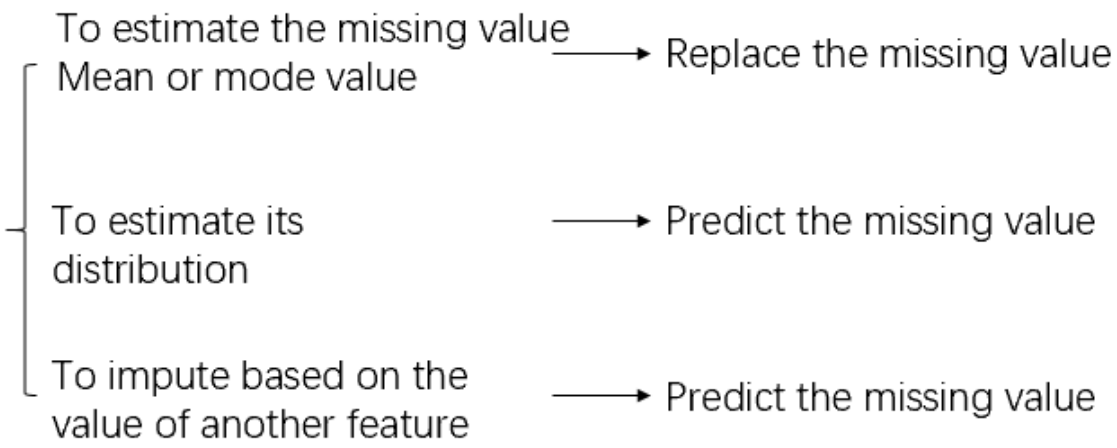


Part I: Missing data –Testing Process

- **the testing process:**

1. Imputation or classification

(a) Predictive Value Imputation



(b) Distribution-based Imputation → **Same as probability split in training**

(c) Unique-value imputation. **(Separate Class)**

Part I: Missing data –Testing Process

2. Reduced-feature models

Method: classify the example with the decision tree which reduced this missing feature

Two implementation:

- **Induce a model online**

  Limitation: need computation time and storage of the training data

- **Pre-compute and store models**

  Limitation: need storage of models, which in the worst case is exponential in the number of attributes.

3. Hybrid models（Combination of the first two methods)

Target：Balance of storage and computation

- **reduced-feature models** are stored for the **most important** patterns;
- **lazy learning or imputation** could be applied for **less-important** patterns.

Part I: Missing data – Example

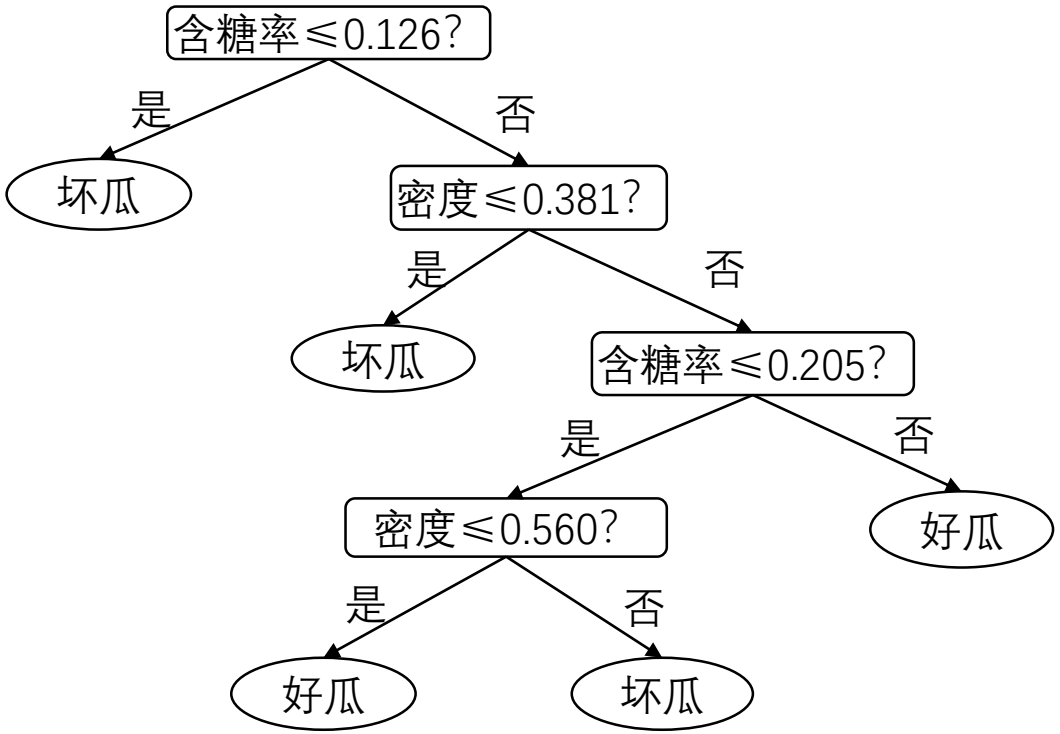
- **Example:**

DataSet

编号	密度	含糖率	好瓜
1	0.697	0.400	是
2	0.774	0.376	是
3	0.634	0.264	是
4	0.608	0.318	是
5	0.556	0.215	是
6	0.403	0.237	是
7	0.481	0.149	是
8	0.437	0.211	是
9	0.666	0.091	否
10	0.243	0.267	否
11	0.245	0.057	否
12	0.343	0.099	否
13	0.639	0.161	否
14	0.657	0.198	否
15	0.360	0.370	否
16	0.593	0.042	否
17	0.719	0.103	否

《机器学习》 89页表4.5 西瓜数据集3.0α

Decision Tree



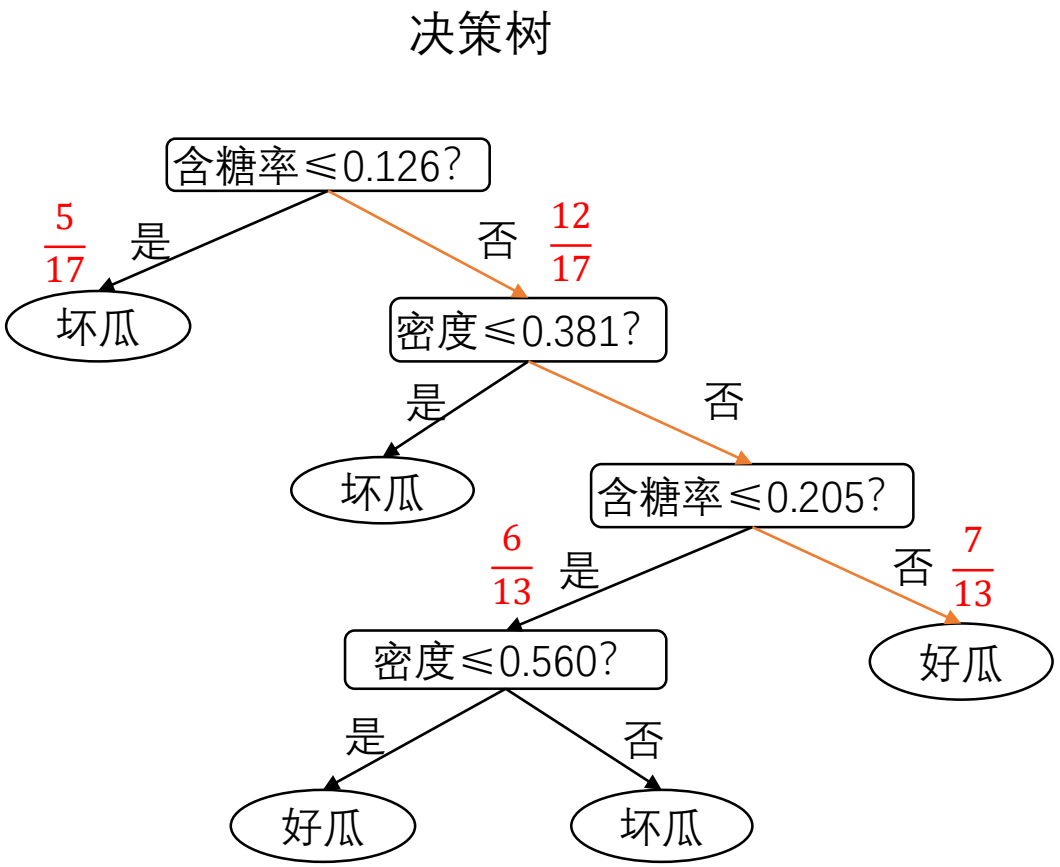
Testing Sample

密度0.400；含糖量未知

好瓜or坏瓜?

（例子只用两个属性说明不同方法的实现过程，实际情况属性很少有缺失，再预测就没意义了…）

Part I: Missing data – Example



测试样本：密度0.400；含糖率未知

- **Predictive Value Imputation：**
  含糖量： Mean Value = 0.213
  Impute the missing data with the mean value :0.213

  Classification： 好瓜

- **Distribution-Based Imputation：**
  Classify the testing sample based on the distribution of the training data, which provide the weight of each sub-tree on the current arribute

  Classification： 好瓜

- **Reduced Model**
  Delete the missing attribute from the training data, and put the new tree into use to predict the classification.

Part I: Missing data – Example

- **决策树的构建及测试——Probabilistic split, Complete Case Method, Complete Variable Method**

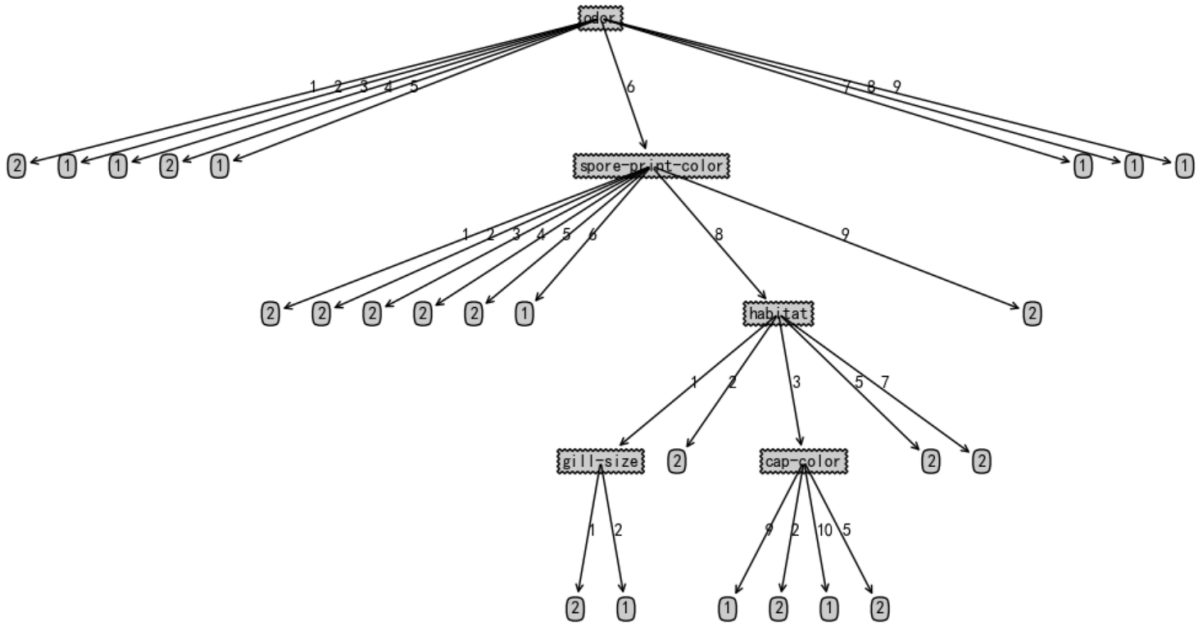
  DataSet: mushrooms.csv　**8124 samples**，**23 attributes**

  TrainingSet：**70% of the DataSet，data is missing with probability of 'missing_ratio'**

  TestingSet：30% of the DataSet

  Results：the Tree Structure and predicting accuracy of each method

  No missing data，accuracy = 1.0，Tree Structure:



Part I: Missing data – Example

1. 利用同一训练集训练，不同缺失率下，三种处理方法得到的预测准确率：

缺失率	Probabilistic split	Complete case method	Complete variable method
0.0001	1.0	1.0	0.9997538158542589
0.0005	0.9992614475627769	0.999507631708518	0.7914820285573609
0.001	0.9982767109798129	0.9955686853766618	Empty FeatureSet
0.005	0.9923682914820285	0.8549975381585426	Empty FeatureSet
0.01	0.966272772033481	0.8313638601674053	Empty FeatureSet
0.05	0.8705071393402265	0.8737075332348597	Empty FeatureSet
0.1	0.708025603151157	0.6361398325947809	Empty FeatureSet

Complete case method和weight方法都能获得较高准确率

Complete variable method 速度最快，但最不可靠，而且只适用于缺失值较少的情况

随着**缺失数据增多**，Probabilistic split方法优势更加明显

Part I: Missing data – Example

2. 同一缺失率0.001下，不同训练集规模，三种处理方法对同一测试集预测得到的准确率:

训练集规模	Probabilistic split	Complete case method	Complete variable method
0.5%	0.9497784342688331	0.8025603151157066	0.9497784342688331
1%	0.9347612013786312	0.7968980797636632	0.9347612013786312
5%	0.9977843426883308	0.9694731659281143	0.9864598719842442
10%	0.9977843426883308	0.9928606597735106	0.9522402757262433
50%	0.9972919743968488	0.9881831610044313	0.7218119153126539

训练集规模较少时，Complete variable method和weight方法可利用少量样本获得准确率较高的决策树模型，Complete case method效果较差

Probabilistic split方法在**训练样本较少**的情况下依然表现良好

Part II: Continuous and integer-valued input attributes

Continuous and integer-valued input attributes: Continuous or integer-valued attributes such as Height and Weight, have an infinite set of possible values. Rather than generate infinitely many branches, decision-tree learning algorithms typically find the ***split point*** that gives the highest information gaim. Efficient methods exist for finding good split points: start by sorting the values of the attribute, and then consider only split points that are between two examples in sorted order that have different classifications, while keeping track of the running totals of positive and negative examples on each side of the split point. Splitting is the most expensive part of real-world decision tree learning applications.

Decision Trees-C4.5

- C4.5 is an algorithm developed by Ross Quinlan that generates Decision Trees (DT)
- can be used for classification problems
- improves (extends) the **ID3 algorithm** by **dealing with both continuous and discrete attributes**, missing values and pruning trees after construction

  pruning trees:略去挂着几个元素的结点，防止over-fitting
- Its commercial successor is **C5.0/See5**, a lot faster that C4.5, more memory efficient and used for building smaller decision trees

Decision Trees-C4.5

- **Gain** $\mathrm{Gain}(S, A) = \mathrm{Entropy}(S) - \sum_{v \in \mathrm{Values}(A)} \frac{|S_v|}{|S|} \mathrm{Entropy}(S_v)$
- **Gain Ratio** $\mathrm{GainRatio}(S, A) = \frac{\mathrm{Gain}(S,A)}{\mathrm{SplitInformation}(S,A)}$
- **Split Information** $\mathrm{SplitInformation}(S, A) \equiv -\sum_{i=1}^{c} \frac{|S_i|}{|S|} \log_2 \frac{|S_i|}{|S|}$

（克服ID3用信息增益选择属性时偏向选择取值多的属性的不足）

Example of C4.5

Day	Outlook	Temp.	Humidity	Wind	Decision
1	Sunny	85	85	Weak	No
2	Sunny	80	90	Strong	No
3	Overcast	83	78	Weak	Yes
4	Rain	70	96	Weak	Yes
5	Rain	68	80	Weak	Yes
6	Rain	65	70	Strong	No
7	Overcast	64	65	Strong	Yes
8	Sunny	72	95	Weak	No
9	Sunny	69	70	Weak	Yes
10	Rain	75	80	Weak	Yes
11	Sunny	75	70	Strong	Yes
12	Overcast	72	90	Strong	Yes
13	Overcast	81	75	Weak	Yes
14	Rain	71	80	Strong	No

- humidity is a continuous attribute
- We need to convert continuous values to nominal ones
- C4.5 proposes to perform binary split based on a threshold value
- Threshold should be a value which offers maximum gain for that attribute

Example of C4.5

Day	Humidity	Decision
7	65	Yes
6	70	No
9	70	Yes
11	70	Yes
13	75	Yes
3	78	Yes
5	80	Yes
10	80	Yes
14	80	No
1	85	No
2	90	No
12	90	Yes
8	95	No
4	96	Yes

- Firstly, we need to **sort humidity values smallest to largest**
- iterate on all humidity values and seperate dataset into two parts
- calculate the gain or gain ratio for every step
- The value which maximizes the gain would be the threshold

Example of C4.5

Day	Humidity	Decision
7	65	Yes
6	70	No
9	70	Yes
11	70	Yes
13	75	Yes
3	78	Yes
5	80	Yes
10	80	Yes
14	80	No
1	85	No
2	90	No
12	90	Yes
8	95	No
4	96	Yes

- Check 65 as a threshold for humidity
- $Entropy(Decision|Humidity \leq 75) = -P(No) * \log_2 P(No) - P(Yes) * \log_2 P(Yes) = -\left(\frac{0}{1}\right) * \log_2 \frac{0}{1} - \frac{1}{1} * \log_2 \frac{1}{1} = 0$
- $Entroy(Decision|Humidty > 65) = -\left(\frac{5}{13}\right) * \log_2 \frac{5}{13} - \left(\frac{8}{13}\right) * \log_2 \frac{8}{13} = 0.530 + 0.431 = 0.96$
- $Entroy(Decision) = \sum -P(l) \log_2 P(l) = -P(Yes) * \log_2 P(Yes) - P(No) * \log_2 P(No) = -\left(\frac{9}{14}\right) * \log_2 \frac{9}{14} - \frac{5}{14} * \log_2 \frac{5}{14} = 0.940$
- $Gain(Decision, Humidty \neq 65) = 0.940 - \left(\frac{1}{14}\right) * 0 - \left(\frac{13}{14}\right) * (0.961) = 0.048$
- $\text{Split}Info(Decision, Humidty \neq 65) = -\left(\frac{1}{14}\right) * \log_2 \frac{1}{14} - \left(\frac{13}{14}\right) * \log_2 \frac{13}{14} = 0.371$
- $GainRatio(Decision, Humidty \neq 65) = 0.126$

Example of C4.5

Day	Humidity	Decision
7	65	Yes
6	70	No
9	70	Yes
11	70	Yes
13	75	Yes
3	78	Yes
5	80	Yes
10	80	Yes
14	80	No
1	85	No
2	90	No
12	90	Yes
8	95	No
4	96	Yes

- Check 70 as a threshold for humidity
- $Entropy(Decision|Humidity \leq 70) = -\left(\frac{1}{4}\right) * \log_2 \frac{1}{4} - \left(\frac{3}{4}\right) * \log_2 \frac{3}{4} = 0.811$
- $Entropy(Decision|Humidity > 70) = -\left(\frac{4}{10}\right) * \log_2 \frac{4}{10} - \left(\frac{6}{10}\right) * \log_2 \frac{6}{10} = 0.970$
- $Gain(Decision, Humidity \neq 70) = 0.940 - \left(\frac{4}{14}\right) * (0.811) - \left(\frac{10}{14}\right) * (0.970) = 0.940 - 0.231 - 0.6922 = 0.014$
- $SplitInfo(Decision, Humidity \neq 70) = -\left(\frac{4}{14}\right) * \log_2 \frac{4}{14} - \left(\frac{10}{14}\right) * \log_2 \frac{10}{14} = 0.863$
- $\text{GainRatio}(Decision, Humidity \neq 70) = 0.016$

Example of C4.5

Day	Humidity	Decision
7	65	Yes
6	70	No
9	70	Yes
11	70	Yes
13	75	Yes
3	78	Yes
5	80	Yes
10	80	Yes
14	80	No
1	85	No
2	90	No
12	90	Yes
8	95	No
4	96	Yes

- Check 75 as a threshold for humidity
- $Entropy(Decision|Humidity \leq 75) = -\left(\frac{1}{5}\right) * \log_2 \frac{1}{5} - \left(\frac{4}{5}\right) * \log_2 \frac{4}{5}$
- $Entropy(Decision|Humidity > 75) = -\left(\frac{4}{9}\right) * \log_2 \frac{4}{9} - \left(\frac{5}{9}\right) * \log_2 \frac{5}{9} = 0.991$
- $(Decision, Humidity \neq 75) = 0.940 - \left(\frac{5}{14}\right) * (0.721) - \left(\frac{9}{14}\right) * (0.991) = 0.940 - 0.2575 - 0.637 = 0.045$
- $SplitInfo(Decision, Humidity \neq 75) = -\left(\frac{5}{14}\right) * \log_2 \frac{4}{14} - \left(\frac{9}{14}\right) * \log_2 \frac{10}{14} = 0.940$
- $\text{GainRatio}(Decision, Humidity \neq 75) = 0.047$

Example of C4.5

Day	Humidity	Decision
7	65	Yes
6	70	No
9	70	Yes
11	70	Yes
13	75	Yes
3	78	Yes
5	80	Yes
10	80	Yes
14	80	No
1	85	No
2	90	No
12	90	Yes
8	95	No
4	96	Yes

- $Gain(Decision, Humidty \neq 78) =$
  $0.090. GainRatio(Decision, Humidity \neq 78) = 0.090$
- $\boldsymbol{Gain(Decision, Humidty \neq 80) =}$
  $\boldsymbol{0.101, GainRatio(Decision, Humidity \neq 80) = 0.107}$
- $Gain(Decision, Humidty \neq 85) =$
  $0.024, GainRatio(Decision, Humidity \neq 80) = 0.027$
- $Gain(Decision, Humidty \neq 90) =$
  $0.010, GainRatio(Decision, Humidity \neq 80) = 0.016$
- $Gain(Decision, Humidty \neq 95) =$
  $0.048, GainRatio(Decision, Humidity \neq 80) = 0.128$

- Value 96 can be ignored

Example of C4.5

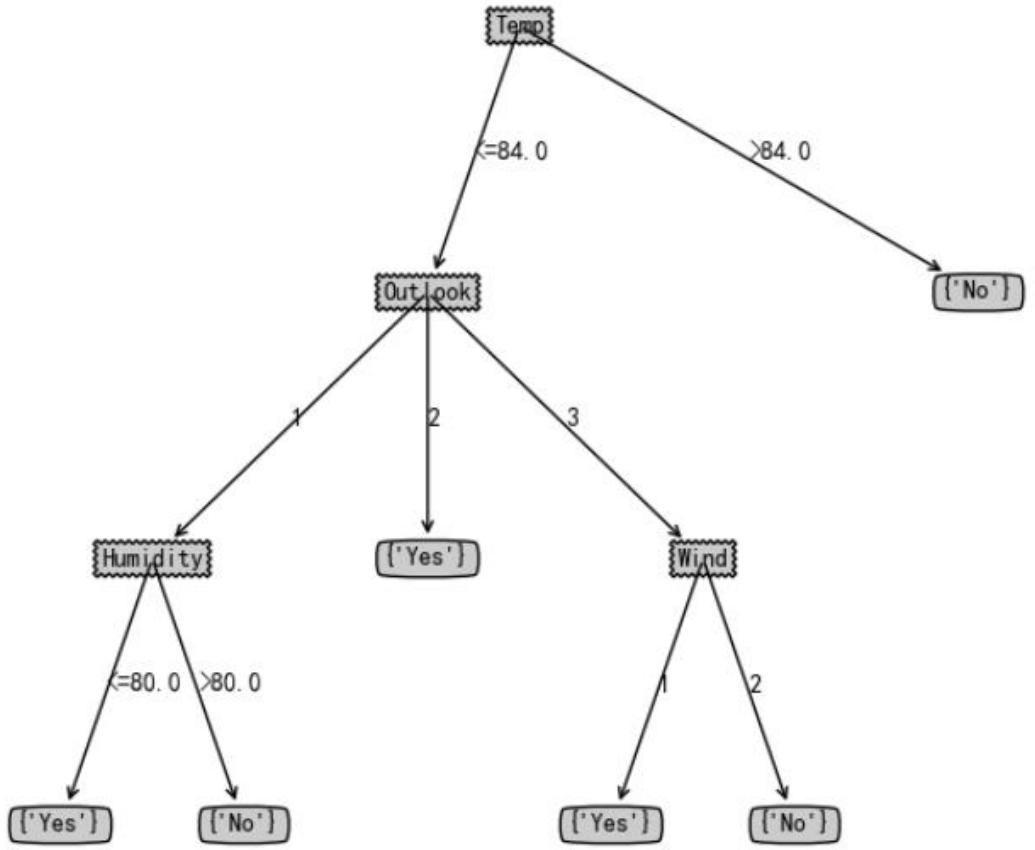
Attribute	Gain	GainRatio
Wind	0.049	0.049
Outlook	0.246	0.155
Humidity <> 80	0.101	0.107
Temperature<>84	0.113	0.305

- Temperature attribute comes with both maximized gain ratio.
- This means that we need to put temperature decision in root of decision tree.

Implement of C4.5

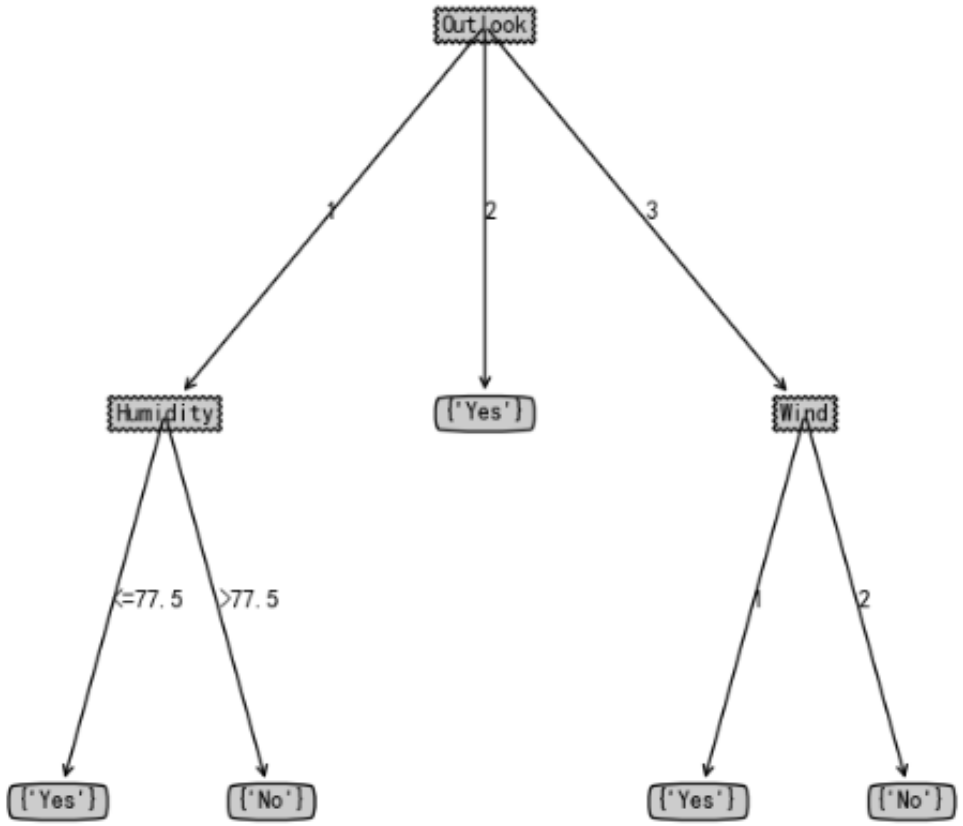
```
The attributes now is: Outlook, the gainratio is: 0.156
The attributes now is: Temp, the best threshold is: 84.000000, the gainratio is: 0.305
The attributes now is: Humidity, the best threshold is: 95.500000, the gainratio is: 0.129
The attributes now is: Wind, the gainratio is: 0.049
Then we choose the attibute:Temp, the gainratio is 0.305
The attributes now is: Outlook, the gainratio is: 0.133
The attributes now is: Humidity, the best threshold is: 95.500000, the gainratio is: 0.109
The attributes now is: Wind, the gainratio is: 0.111
Then we choose the attibute:Outlook, the gainratio is 0.133
The attributes now is: Humidity, the best threshold is: 80.000000, the gainratio is: 0.384
The attributes now is: Wind, the gainratio is: 0.000
Then we choose the attibute:Humidity, the gainratio is 1.000
The attributes now is: Humidity, the best threshold is: 75.000000, the gainratio is: 0.237
The attributes now is: Wind, the gainratio is: 1.000
Then we choose the attibute:Wind, the gainratio is 1.000
```

Implement of C4.5

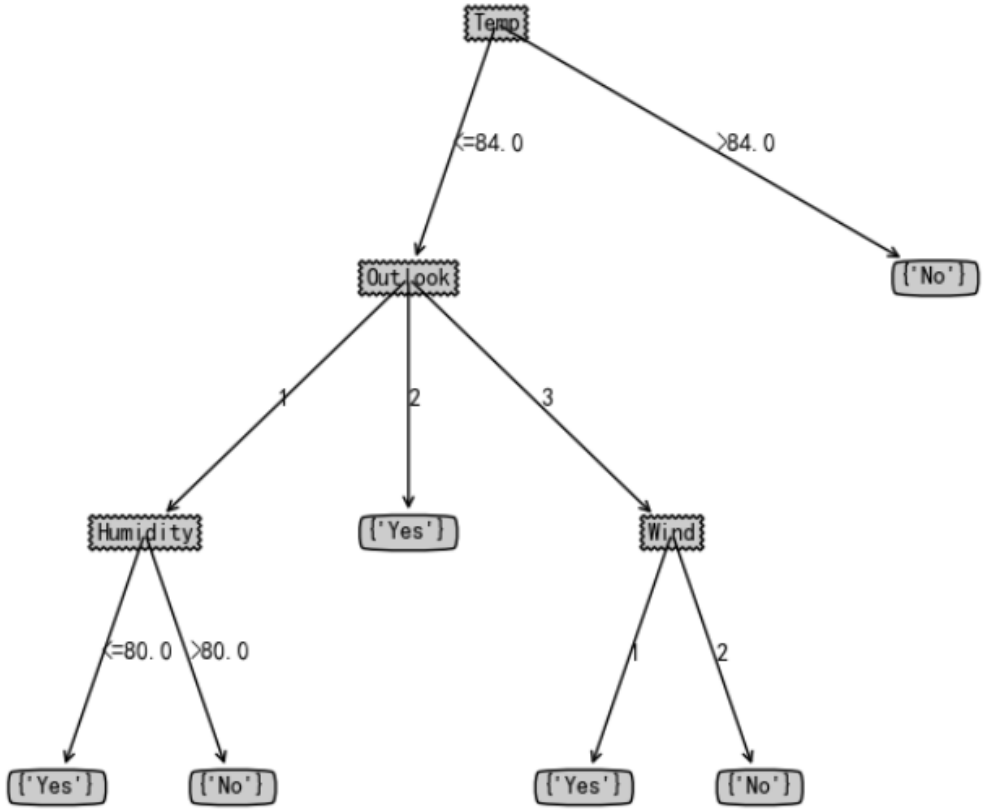


Implement of C4.5

Gain



Gain Ratio



Gain will tend to use the discrete attributes with most different values

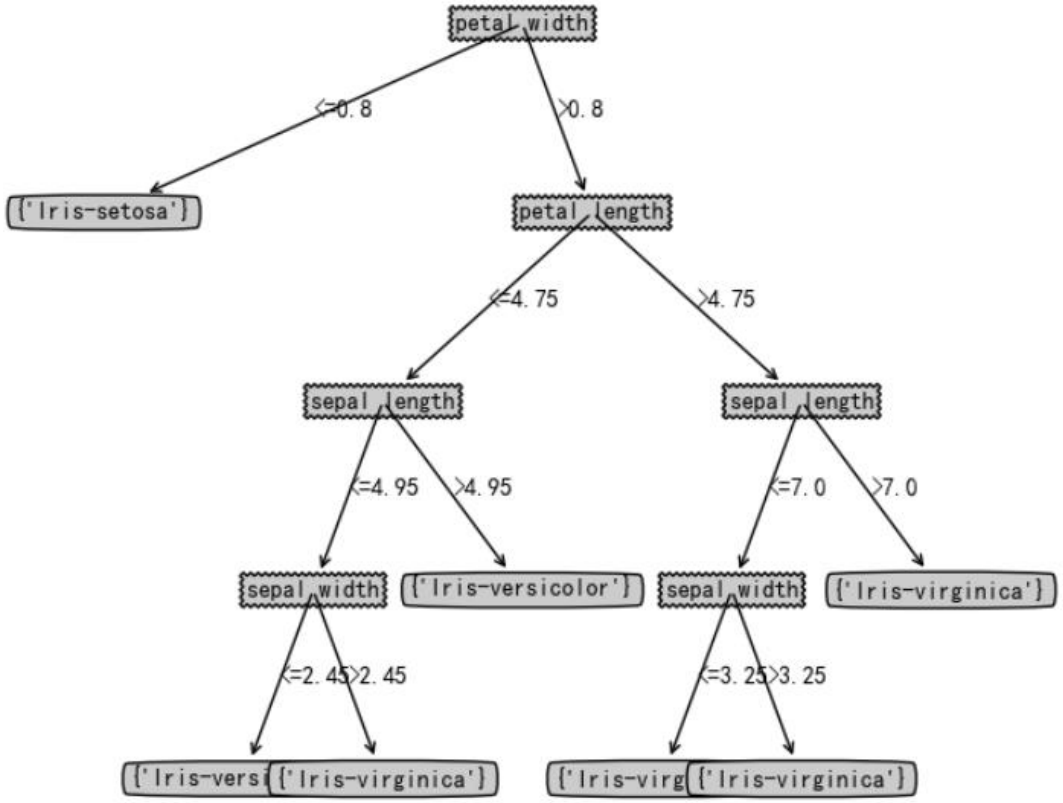
Implement of C4.5

Dataset: iris

```
The attributes now is: sepal length, the best threshold is: 5.450000, the gainratio is: 0.184
The attributes now is: sepal width, the best threshold is: 3.350000, the gainratio is: 0.184
The attributes now is: petal length, the best threshold is: 2.450000, the gainratio is: 0.184
The attributes now is: petal width, the best threshold is: 0.800000, the gainratio is: 0.228
Then we choose the attibute:petal width, the gainratio is 1.000
The attributes now is: sepal length, the best threshold is: 7.050000, the gainratio is: 0.125
The attributes now is: sepal width, the best threshold is: 3.500000, the gainratio is: 0.143
The attributes now is: petal length, the best threshold is: 4.750000, the gainratio is: 0.125
Then we choose the attibute:petal length, the gainratio is 0.662
The attributes now is: sepal length, the best threshold is: 4.950000, the gainratio is: 0.005
The attributes now is: sepal width, the best threshold is: 2.550000, the gainratio is: 0.005
Then we choose the attibute:sepal length, the gainratio is 0.417
The attributes now is: sepal width, the best threshold is: 2.450000, the gainratio is: 1.000
Then we choose the attibute:sepal width, the gainratio is 1.000
The attributes now is: sepal length, the best threshold is: 7.000000, the gainratio is: 0.023
The attributes now is: sepal width, the best threshold is: 3.250000, the gainratio is: 0.027
Then we choose the attibute:sepal length, the gainratio is 0.055
The attributes now is: sepal width, the best threshold is: 3.250000, the gainratio is: 0.038
Then we choose the attibute:sepal width, the gainratio is 0.052
```

```
self.traindata = self.data[0: math.floor(0.9*lendata)]
self.testdata = self.data[math.floor(0.9*lendata):lendata]
```

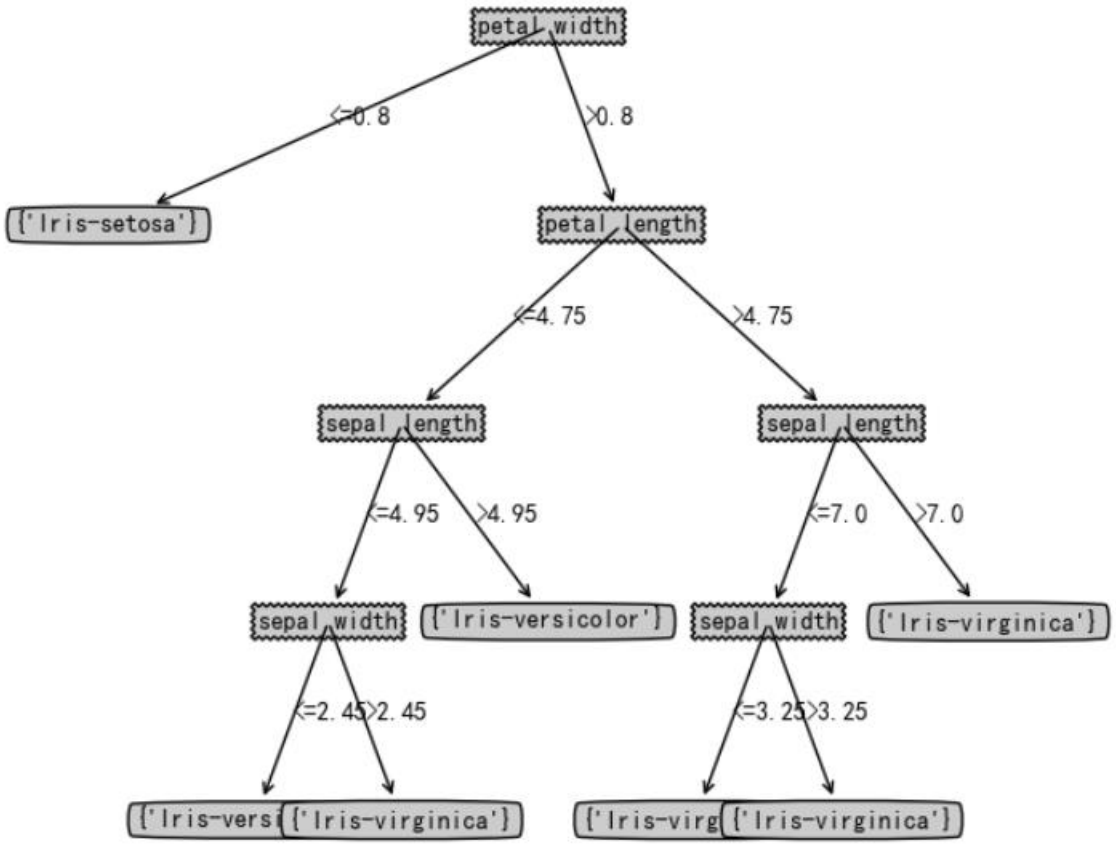
```
Accuracy:    1.0
```



Implement of C4.5

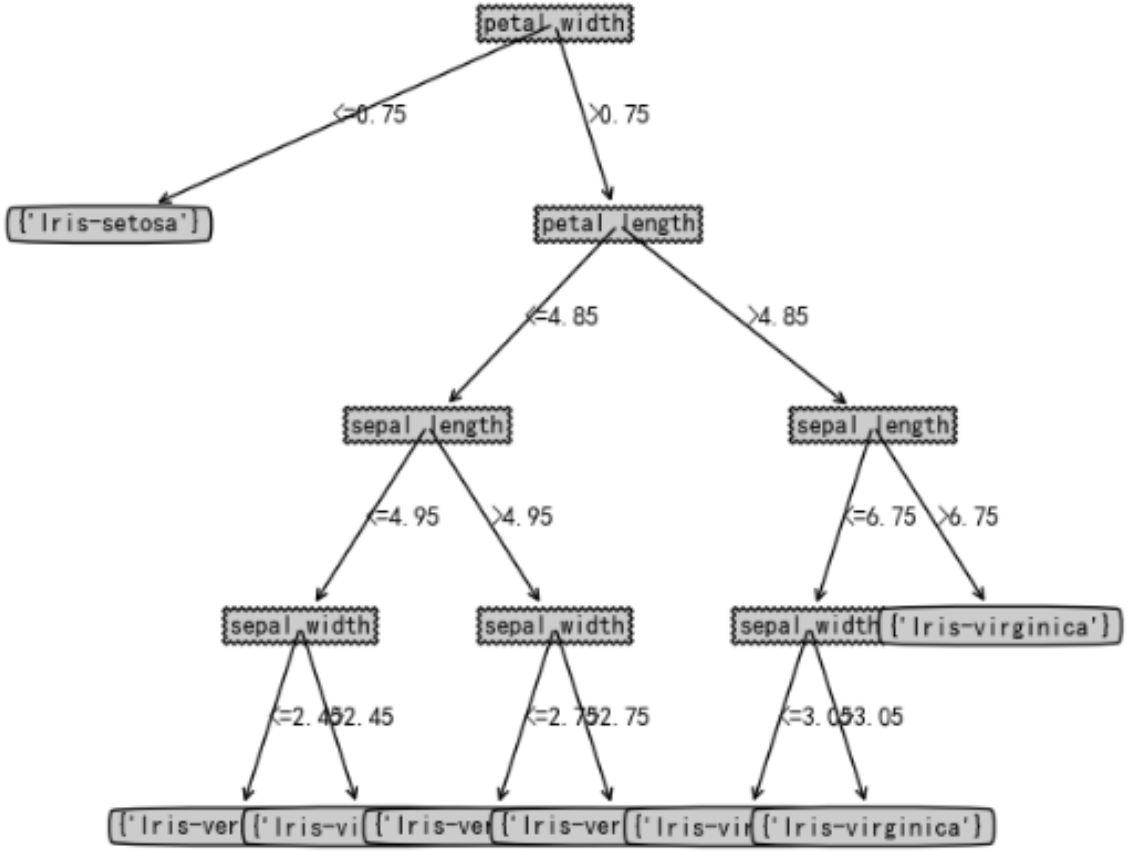
Dataset: iris

Gain Ratio



Accuracy: 1.0

Gain



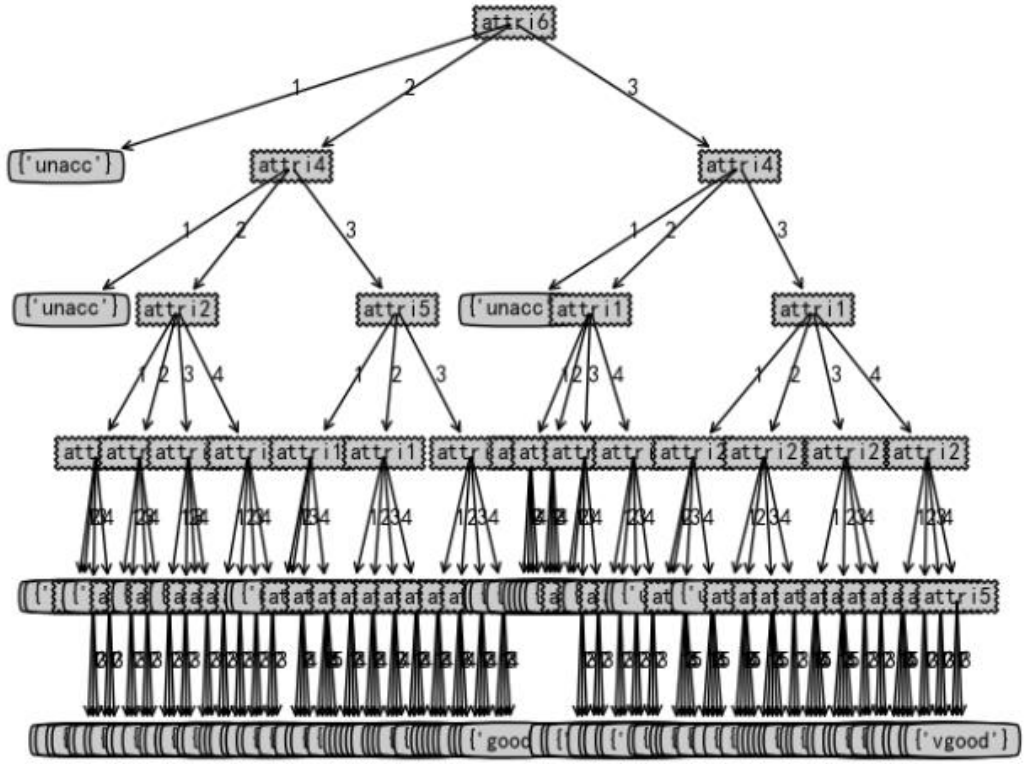
Accuracy: 0.9333333333333333

Part II: Continuous and integer-valued input attributes

➢ Improvement I: Pre-Pruning

pruning trees:节点中example小于一定阈值时停止划分，防止over-fitting

Attributes can be used many times



Dataset: CAR EVALUATION DATASET

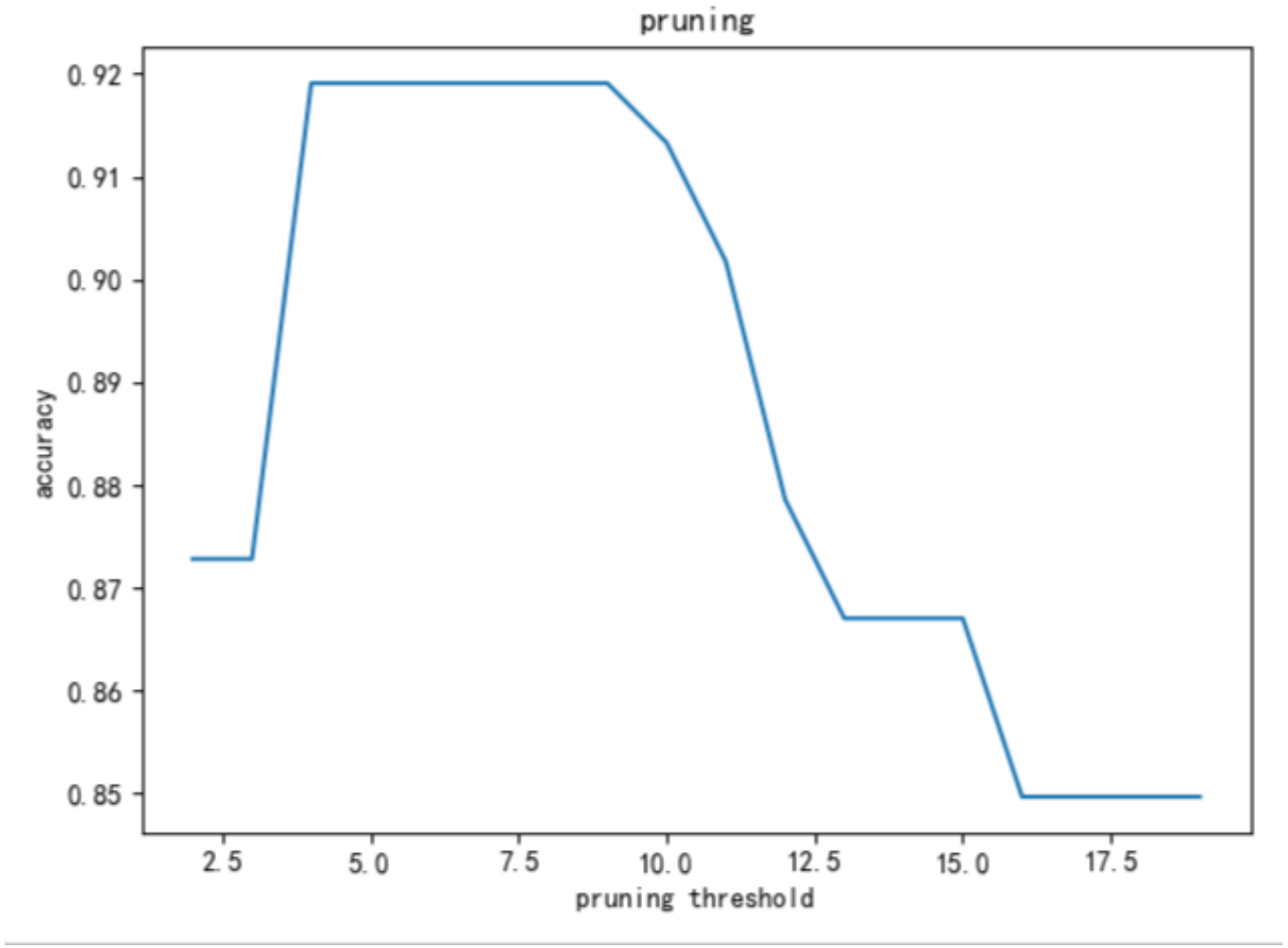
- URL: https://archive.ics.uci.edu/ml/datasets/Car+Evaluation
- Number of Instances: 1728 Number of Attributes: 7

```
Accuracy without prunning:  0.8901734104046243
```

Part II: Continuous and integer-valued input attributes

➢ Improvement I: Pre-Pruning

```
Accuracy with prunning:   0.8728323699421965 threshold =  2
Accuracy with prunning:   0.8728323699421965 threshold =  3
Accuracy with prunning:   0.930635838150289 threshold =  4
Accuracy with prunning:   0.930635838150289 threshold =  5
Accuracy with prunning:   0.930635838150289 threshold =  6
Accuracy with prunning:   0.930635838150289 threshold =  7
Accuracy with prunning:   0.930635838150289 threshold =  8
Accuracy with prunning:   0.930635838150289 threshold =  9
Accuracy with prunning:   0.9190751445086706 threshold =  10
Accuracy with prunning:   0.8901734104046243 threshold =  11
Accuracy with prunning:   0.8728323699421965 threshold =  12
Accuracy with prunning:   0.8728323699421965 threshold =  13
Accuracy with prunning:   0.8728323699421965 threshold =  14
Accuracy with prunning:   0.861271676300578 threshold =  15
Accuracy with prunning:   0.861271676300578 threshold =  16
Accuracy with prunning:   0.861271676300578 threshold =  17
Accuracy with prunning:   0.861271676300578 threshold =  18
Accuracy with prunning:   0.861271676300578 threshold =  19
```



Part II: Continuous and integer-valued input attributes

➢ Improvement II: Segment+C4.5

1. Background:

Because of the Occam's razor(奥卡姆剃刀原理) we prefer a smaller tree to a bigger tree when both of them are acceptable.

2. Motivation:

S_1: 1 1 1 1 1 1 2 2 2 2 √

S_2: 1 2 1 2 1 2 1 2 1 1

These are two different permutations of examples with different attribution, we obviously prefers S1 over S2. So, there is some useful information for classification.

Wang R, Kwong S, Wang X Z, et al. Segment based decision tree induction with continuous valued attributes[J]. IEEE transactions on cybernetics, 2015, 45(7): 1262-1275.

Part II: Continuous and integer-valued input attributes

3. Definitions:

- **Segment**

Let Q be a permutation of examples in S. Then a sub-queue of Q, i.e. SQ = ($\mathbf{e}_r$, $\mathbf{e}_{r+1}$, … , $\mathbf{e}_t$), $1 \leqslant r \leqslant t \leqslant N$, is called a segment if and only if it satisfies the following requirements:

1) C ($\mathbf{e}_r$) = C ($\mathbf{e}_{r+1}$) = … = C ($\mathbf{e}_t$).

2) C ($\mathbf{e}_r$) ≠ C ($\mathbf{e}_{r-1}$) if and only if r ≠ 1.

3) C ($\mathbf{e}_t$) ≠ C ($\mathbf{e}_{t+1}$) if and only if t ≠ N.

Seg(Q(A_j)), also denoted as Seg(S, A_j), is called a set of segments in S induced by A_j.

- **Number of Segments in a Node**

The number of segments in node S is defined as $Seg\#(S) = \min_j |Seg(S, A_j)|$

where | | denotes the number of elements in a set.

Part II: Continuous and integer-valued input attributes

- **Bar**

Suppose there is a duplicated value of A_j at some examples in S, and let this value be a, then the set of examples $\{e \in S \mid A_j(e) = a\}$ is called a bar in S with respect to A_j, denoted by $Bar(S, A_j = a)$. The value a is called a bar point.

- **Number of Segments in a Bar**

Let B be a bar in S with respect to Aj, and Q* be a permutation of examples in B whose class labels are most chaotic. Then the number of segments in bar B is defined as $Seg\#(B) = |Seg(Q^*)|$

Input: All the examples in a bar $\mathcal{B}$.
Output: The number of segments in bar $\mathcal{B}$: $bSeg\#(\mathcal{B})$.

Get the numbers of examples belonging to each class: $n_1, n_2, \ldots, n_L$;
Sort $n_1, n_2, \ldots, n_L$ in descending order. Suppose the sorted values are $n'_1, n'_2, \ldots, n'_L$;
Set initial value: $bSeg\#(\mathcal{B}) = 0$;
while $n'_2 > 0$ **do**
    $bSeg\#(\mathcal{B}) = bSeg\#(\mathcal{B}) + 2n'_2$;
    Set $n_1 = n'_1 - n'_2$, $n_2 = 0$ and $n_i = n'_i$ for each $i(3 \leq i \leq L)$, thus we get new values of $n_1, n_2, \ldots, n_L$;
    Get the sorted values $n'_1, n'_2, \ldots, n'_L$ of $n_1, n_2, \ldots, n_L$ in descending order;
end
if $n'_1 \neq 0$ **then**
    $bSeg\#(\mathcal{B}) = bSeg\#(\mathcal{B}) + 1$;
end
return $bSeg\#(\mathcal{B})$.

Number of Segments in a Bar

Part II: Continuous and integer-valued input attributes

4. The step to compute the number of segments in a node induced by an attribute

1) Sort the examples with ascending order according to the attribute, and find all the bar points.

2) Divide the order into several bar sub-queues and nonbar sub-queues based on the bar points.

3) Get the number of segments in each nonbar sub-queue directly, and compute the number of segments in each bar sub-queue based on last slide.

4) Sum up the numbers of segments in all the sub-queues, and return it as the final number of segments.

5. Segment+C4.5

1) Calculating the best cut point of every attributes.

2) Sort the candidate cut point in descending order.

3) Select the first K candidate cut point x_{ij} to form the subset X

4) The optimal candidate cut point is derived by

$$x_{ij}{}^{*} = arg \min_{x_{ij} \in X} \frac{|S_1|}{|S|} \left|Seg\left(S_1; \mathrm{x}_{ij}\right)\right| + \frac{|S_2|}{|S|} \left|Seg\left(S_2; \mathrm{x}_{ij}\right)\right|$$

Part II: Continuous and integer-valued input attributes

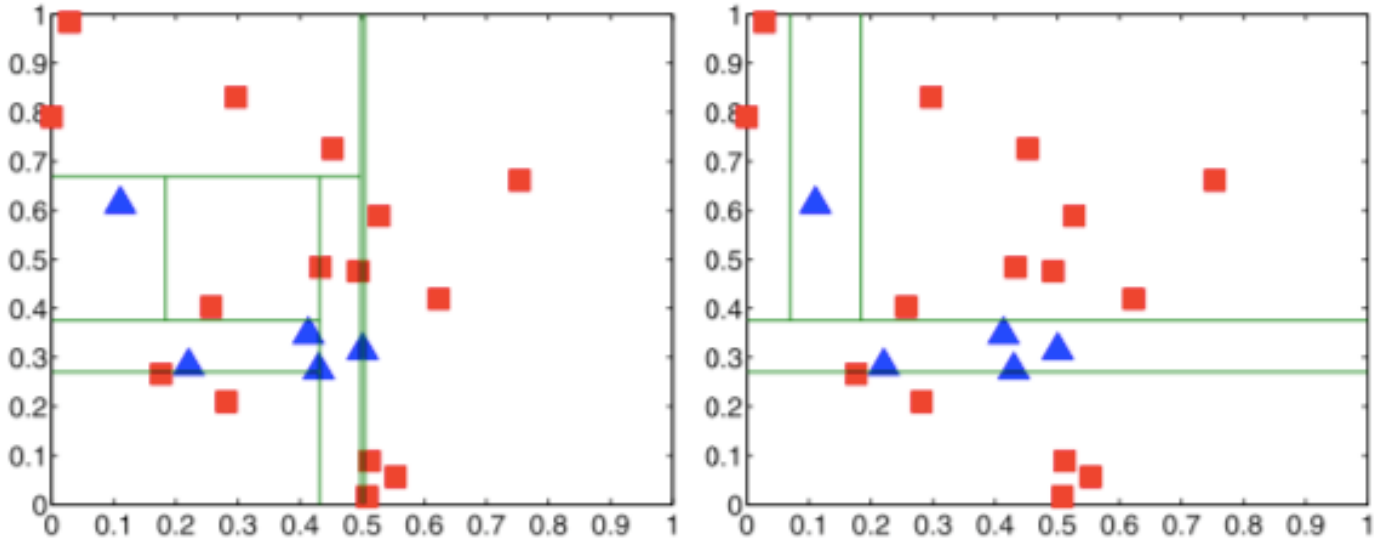
- EXAMPLE

sample	feature1	feature2	label	sample	feature1	feature2	label
1	0.0000	0.7903	2	11	0.1106	0.6129	1
2	0.2207	0.2823	1	12	0.6232	0.4194	2
3	0.4136	0.3468	1	13	0.5009	0.3145	1
4	0.5274	0.5887	2	14	0.4302	0.2742	1
5	0.4327	0.4839	2	15	0.2815	0.2097	2
6	0.5120	0.0887	2	16	0.4524	0.7258	2
7	0.7535	0.6613	2	17	0.0289	0.9839	2
8	0.5083	0.0161	2	18	0.2569	0.4032	2
9	0.2969	0.8306	2	19	0.5538	0.0565	2
10	0.1764	0.2661	2	20	0.4935	0.4758	2

$X_{1,14}$ gives the information gain ratio of 0.1737 and the segment measure value of 5.90.
$X_{2,5}$ gives the information gain ratio of 0.1511 and the segment measure value of 3.25.

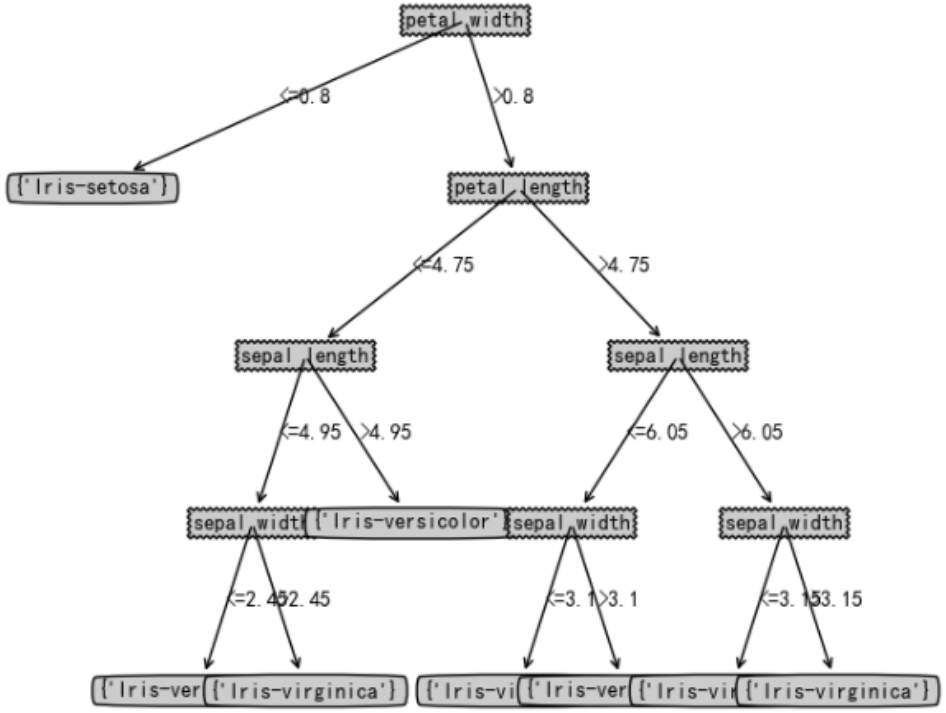
Part II: Continuous and integer-valued input attributes

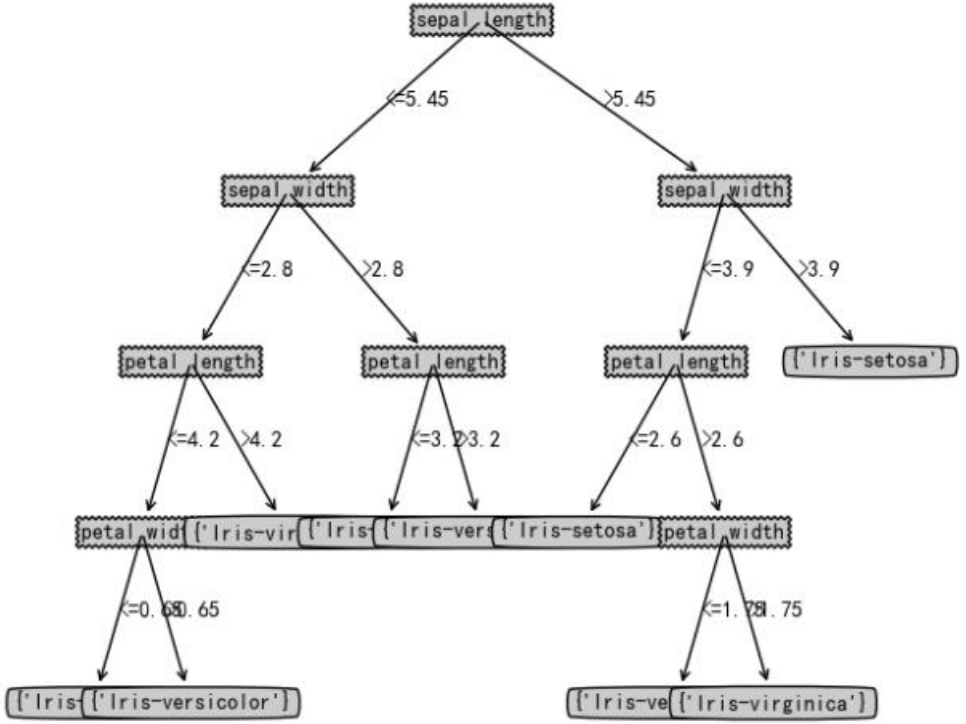
- EXAMPLE



Part II: Continuous and integer-valued input attributes

- EXAMPLE





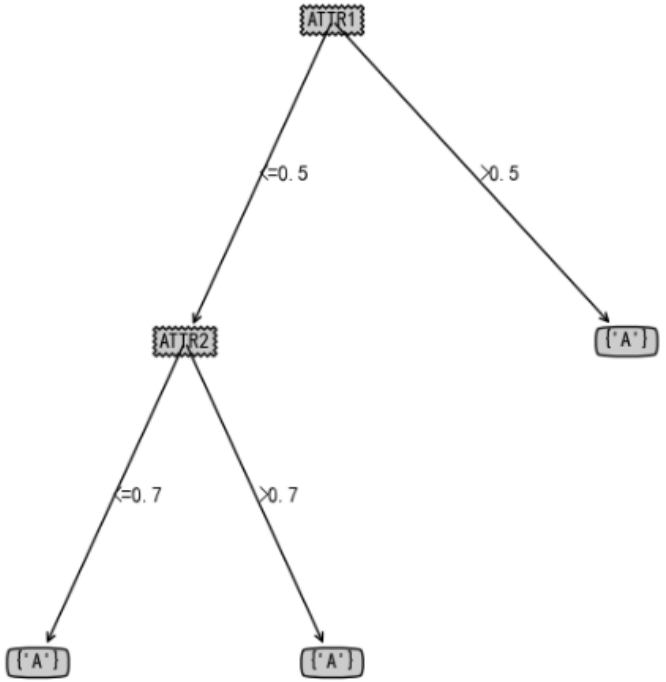
[True, True, False, True, True, True, True, True, True, False, True, True, True, True, True]
strategy without segment: Accuracy: 0.8666666666666667

[True, True, True, True, True, True, True, True, True, True, True, True, True, True, True]
strategy with segment: Accuracy: 1.0

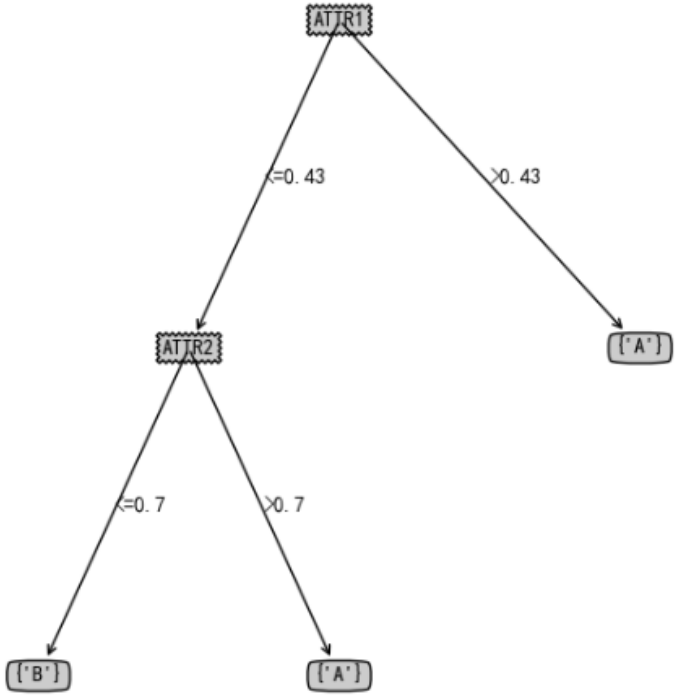
Part III: Combination of missing and continuous

Based on continuous data algorithm, we added the probability split method. Thus, we successfully **solved these two problems together**. The test result are as follow:

0	0.7903	A
0.2207	0.2823	B
0.4136	0.3468	B
0.5274	0.5887	A
0.4327	0.4839	A
0.512	0.0887	A
0.7535	0.6613	A
0.5083	0.0161	A
0.2969	0.8306	A
0.1764	0.2661	A
0.1106	0.6129	B
0.6232	0.4194	A
0.5009	0.3145	B
0.4302	0.2742	B
0.2815	0.2097	A
0.5624	0.7258	A
0.0289	0.9839	A
0.2569	0.4032	A
0.5538	0.0565	A
0.4935	0.4758	A



0	0.7903	A
0.2207	0.2823	B
0.4136	0.3468	B
0.5274	0.5887	A
0.4327	0.4839	A
0.512	0.0887	A
0.7535	0.6613	A
0.5083	0.0161	A
0.2969	0.8306	A
0.1764	0.2661	A
0.1106	0.6129	B
0.6232	0.4194	A
-1	0.3145	B
0.4302	0.2742	B
0.2815	-1	A
0.5624	0.7258	A
0.0289	0.9839	A
0.2569	0.4032	A
-1	0.0565	A
0.4935	0.4758	A



Summary

There are numerous methods dealing with missing data and continuous data in decision tree in theory. And we introduced **probability split method** and **segment+C4.5 method** in detail. After that, we implemented these two methods in Python and combined them together to solve missing continuous data problem.

The presentation material and codes has been uploaded to Github:

URL：https://github.com/ljwztc/decision-tree/tree/master

You are welcomed to testify the results and bring forward any opinion.

Reference

[1]Quinlan J R. Induction of decision trees[J]. Machine learning, 1986, 1(1): 81-106.

[2]Boullé M. MODL: A Bayes optimal discretization method for continuous attributes[J]. Machine learning, 2006, 65(1): 131-165.

[3]Wang R, Kwong S, Wang X Z, et al. Segment based decision tree induction with continuous valued attributes[J]. IEEE transactions on cybernetics, 2015, 45(7): 1262-1275.

Ending

Thanks for watching.

顾涵雪

刘　洁

庞婧璇

涂剑凯