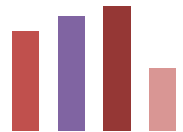
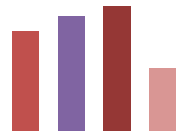


决策树



内容概要

- 一、决策树模型原理
- 二、Sklearn中的决策树分类模型
- 三、特征选择
- 四、线性模型与非线性模型对比



一、决策树模型原理

思考右图所示的鸢尾花数据集分类任务

➤ 目的：区分 c_1 (Iris-setosa, 用圆圈表示) 和 c_2 (其他类型的鸢尾花, 用三角形表示);

• 属性：萼片长度(X_1)和萼片宽度(X_2)属性;

• 数据集 D 有 $n=150$ 个数据点;

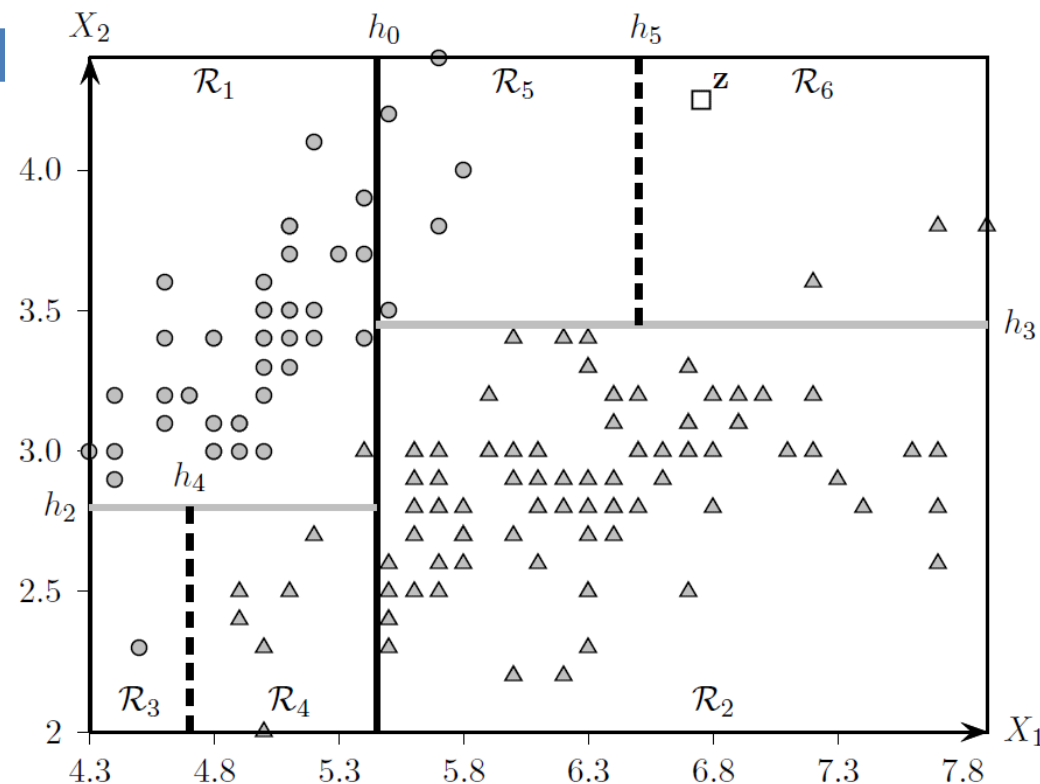
• 数据空间 $\mathcal{R} = range(X_1) \times range(X_2)$
 $= [4.3, 7.9] \times [2.0, 4.4]$

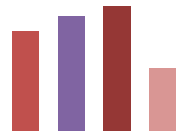
➤ 分类方法：通过平行于坐标轴的直线对空间 \mathcal{R} 不断进行分割，直到数据点的标签接近一致。

- 第一次分割：对应于 h_0 ，对应决策点 $X_1 \leq h_0$ 得到的左半空间和右半空间;
- 第二次分割：分别对应 h_2 和 h_3 (显示为灰线)，对应决策点 $X_2 \leq h_2$, $X_2 \leq h_3$;
- 第三次分割：分别对应 h_4 和 h_5 ，对应决策点 $X_1 \leq h_4$, $X_1 \leq h_5$;

在二维空间中，分割直线被称为超平面。

➤ 最终共得到 $\mathcal{R}_1, \mathcal{R}_2, \mathcal{R}_3, \mathcal{R}_4, \mathcal{R}_5, \mathcal{R}_6$ 6 个数据区域，它们和超平面一起构成决策树模型。





一、决策树模型原理

1. 决策树模型构建和预测

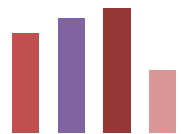
➤ 决策树模型构建：递归地对数据空间 \mathcal{R} 进行分区。

- 使用平行于坐标轴的超平面，将 \mathcal{R} 分成两个半空间区域 \mathcal{R}_1 或 \mathcal{R}_2 ，输入数据划分为 D_1 或 D_2 ；
- \mathcal{R}_1 或 \mathcal{R}_2 被平行于坐标轴的超平面继续分割；
- 直到新生成的区域内的点的类标签相对一致，该区域作为叶子节点。

➤ 决策树模型预测

- 递归地计算它所属的半空间，直到遇到叶子节点，叶子节点的标签作为该点的类标签。

相对一致的判断标准是什么？

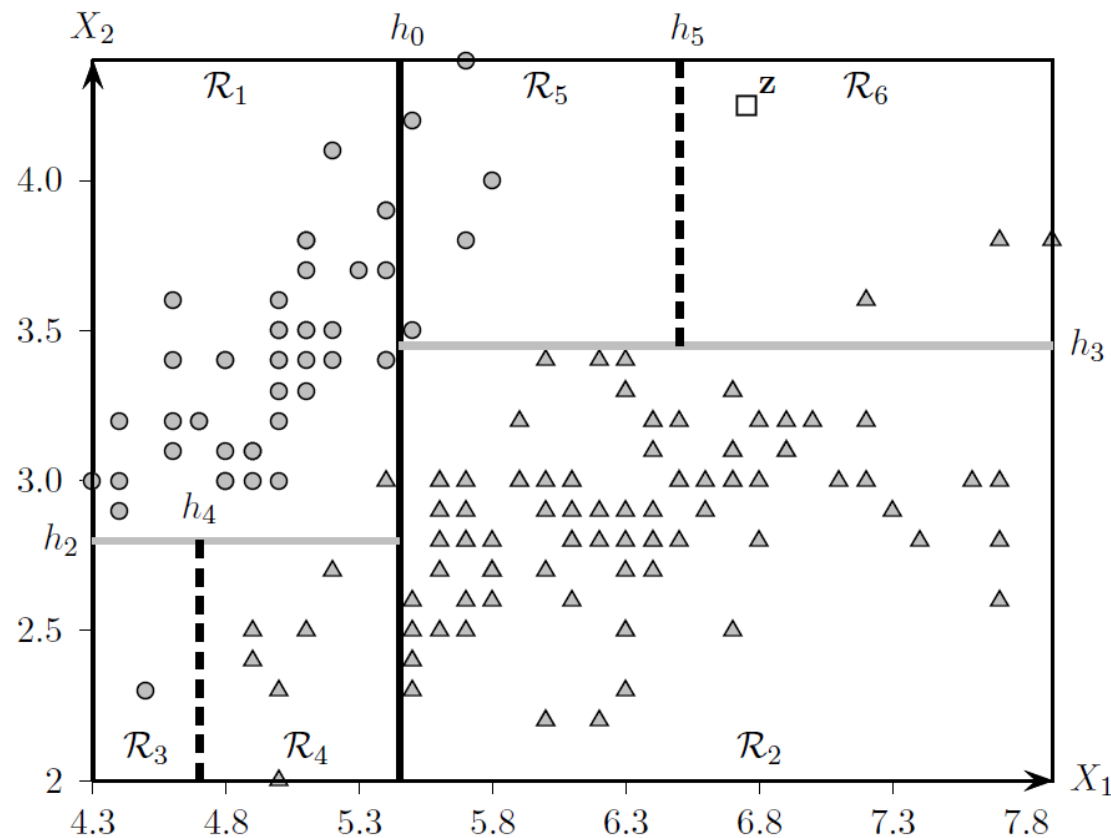
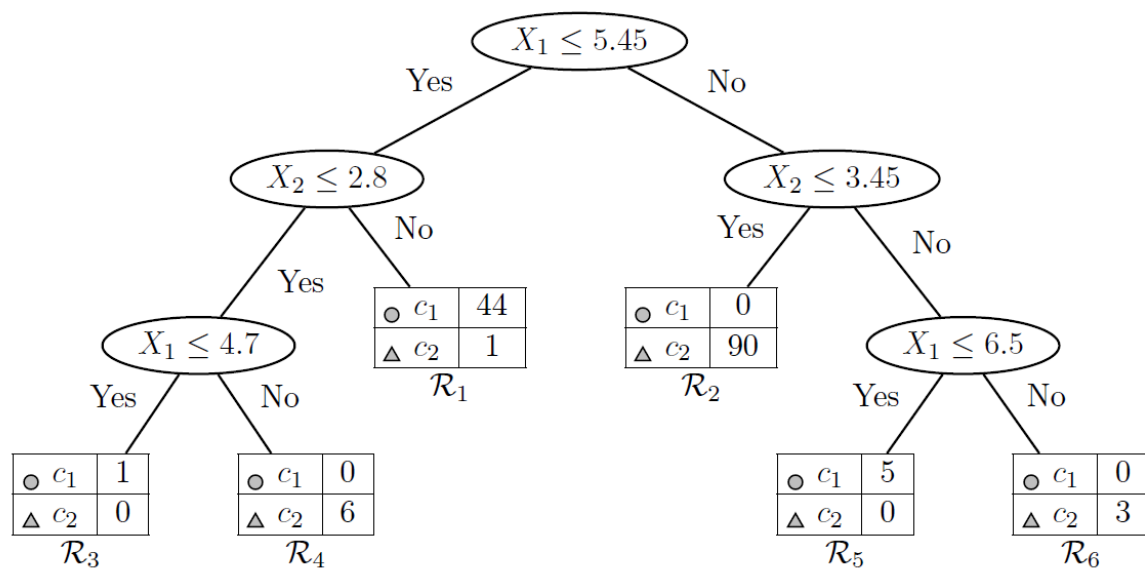


一、决策树模型原理

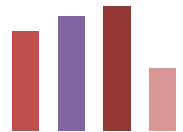
➤ 纯度：多数标签点的比例

$$purity(\mathbf{D}_j) = \max_i \left\{ \frac{n_{ji}}{n_j} \right\}$$

- 其中， $n_j = |\mathbf{D}_j|$ 是 \mathbf{R}_j 中数据点的总数， n_{ji} 是 \mathbf{D}_j 中类标签为 c_i 的点的数目。



用例1所示的超平面递归划分生成的**决策树**。其中区域大小阈值为5（区域至少包含5个点）且纯度小于0.95时才继续分割该区域。



一、决策树模型原理

2. 分割点的形式

➤ 超平面对应分割点，而分割点的形式由数据集的 D 的属性 X_j 的 **类型** 决定。

2.1 对于 **数值型属性**

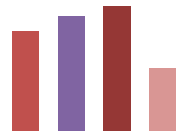
➤ 假设 X_j 为 **数值型属性**，其分割点的形式为：

$$X_j \leq v$$

- 其中， v 是属性 X_j 定义域中的 **某个值**；
- 分割点 $X_j \leq v$ 将输入数据空间 \mathcal{R} 分成两个区域 \mathcal{R}_Y 和 \mathcal{R}_N ，对应着 **输入数据 D 的二分区**：

$$\mathbf{D}_Y = \{\mathbf{x} \mid \mathbf{x} \in \mathbf{D}, x_j \leq v\}$$

$$\mathbf{D}_N = \{\mathbf{x} \mid \mathbf{x} \in \mathbf{D}, x_j > v\}$$

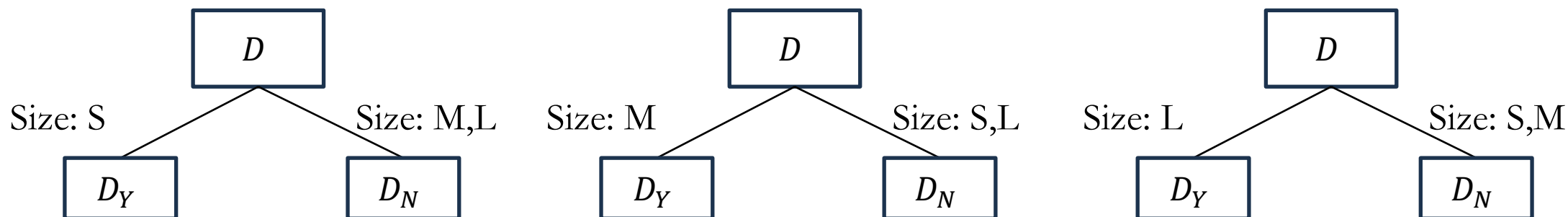


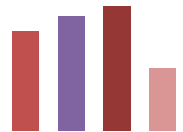
一、决策树模型原理

如何评估分割点的优劣呢？

2.2. 对于类别型属性

- 如果 X 是类别型属性，则分割点形式为： $X \in V$ ， V 为 X 定义域的子集
- 将输入数据空间 \mathcal{R} 分成两个区域 \mathcal{R}_Y 和 \mathcal{R}_N ，对应着输入数据 D 的二分区 D_Y 和 D_N 。
- 一般为限制分割点的数量，通常设定为二元分割。
- 假设某数据集有属性 $Size$ 表示衣服尺寸，其定义域为 $\{S,M,L\}$ ， $Size$ 上的可能得分割点有： $\{S\}$ ， $\{M\}$ ， $\{L\}$





一、决策树模型原理

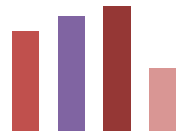
3、分割点评估度量指标

3.1 熵

- 熵衡量系统中的无序度或不确定性。在分类场景中，如果所划分数据区域的多数点具有相同的标签，则它的熵较低；反之，如果区域内点的类标签混杂，它的熵较高。一组带标签的数据集 D 的熵定义如下：

$$H(D) = - \sum_{i=1}^k P(c_i|D) \log_2 P(c_i|D)$$

- 其中， $P(c_i|D)$ 是 D 中类 c_i 的概率， k 是不同类的数目。
- 如果所有点都来自一类，则 $H(D)$ 为0。
- 如果所有类混合在一起，并且每个类以相同的概率 $P(c_i|D) = \frac{1}{k}$ 出现，则熵最大，为 $H(D) = \log_2^k$ 。



一、决策树模型原理

➤ 分割熵

假设一个分割点将 D 分割为 D_Y 和 D_N ，分割熵定义为每个区域的加权熵，如下所示：

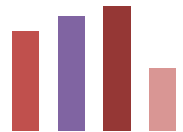
$$H(D_Y, D_N) = \frac{n_Y}{n}H(D_Y) + \frac{n_N}{n}H(D_N)$$

- 其中， $n = |D|$, $n_Y = |D_Y|$ 和 $n_N = |D_N|$ 是 D_Y 和 D_N 中的点数。

➤ 给定分割点的信息增益：分割点是否导致整体熵下降

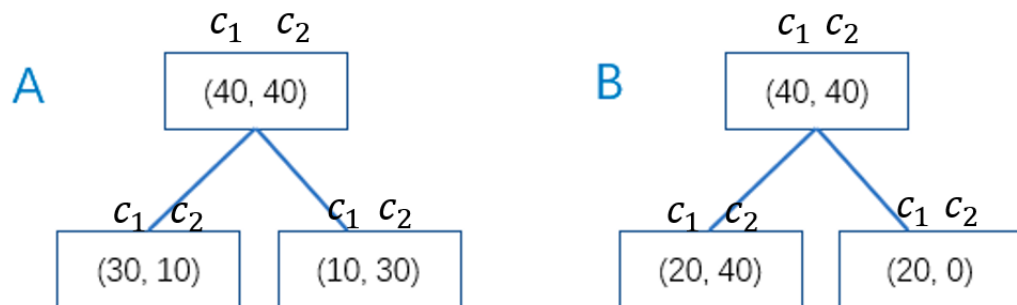
$$Gain(D, D_Y, D_N) = H(D) - H(D_Y, D_N)$$

- 信息增益越高，说明熵降得越低，分割点越好。
- 因此，给定分割点及相应的分区，对每个分割点进行评分，并选择信息增益最高的分割点。在ID3决策树算法中采用信息增益作为节点优劣评估指标。



一、决策树模型原理

例1：如图所示，数据集 D 中的类标签分别为 c_1 和 c_2 ，有A和B两个分割点，计算：



(1) 计算分割前数据集 D 的熵；

数据集 D 的标签为：

- $c_1 = 40$
- $c_2 = 40$
- 总数： $|D| = 40 + 40 = 80$

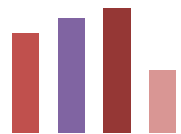
代入计算：

$$P(c_1|D) = \frac{40}{80} = 0.5, \quad P(c_2|D) = \frac{40}{80} = 0.5$$

$$H(D) = -(0.5 \log_2 0.5 + 0.5 \log_2 0.5) = -(0.5 \times (-1) + 0.5 \times (-1)) = 1$$

根据熵的定义公式：

$$H(D) = - \sum_{i=1}^k P(c_i|D) \log_2 P(c_i|D)$$



一、决策树模型原理

(2) 计算A节点和B节点的分割熵 $H(D_Y, D_N)$

分割点 A:

上图中, A 分成两部分:

- 左子集: $D_Y = (30, 10)$, 总数 $n_Y = 40$
- 右子集: $D_N = (10, 30)$, 总数 $n_N = 40$
- 总数 $n = 80$

Step 1: 计算 $H(D_Y)$

$$P(c_1|D_Y) = \frac{30}{40} = 0.75, \quad P(c_2|D_Y) = \frac{10}{40} = 0.25$$

$$H(D_Y) = -(0.75 \log_2 0.75 + 0.25 \log_2 0.25) = -(0.75 \times -0.415 + 0.25 \times -2) = 0.811$$

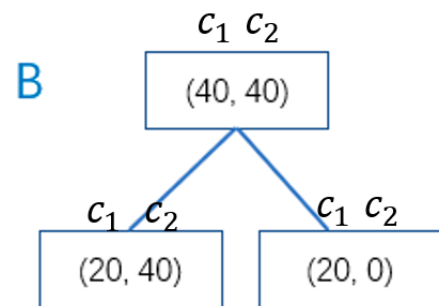
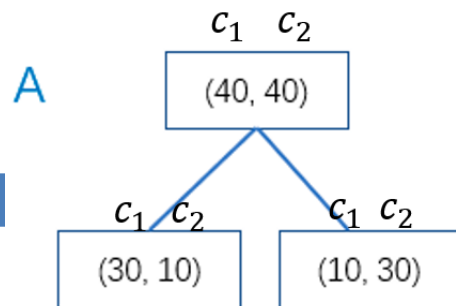
Step 2: 计算 $H(D_N)$

$$P(c_1|D_N) = \frac{10}{40} = 0.25, \quad P(c_2|D_N) = 0.75$$

$$H(D_N) = H(D_Y) = 0.811 \quad (\text{因为概率对称})$$

Step 3: 条件熵

$$H(D_Y, D_N) = \frac{40}{80} \cdot 0.811 + \frac{40}{80} \cdot 0.811 = 0.811$$



分割点 B:

B 分成两部分:

- 左子集: $D_Y = (20, 40)$, 总数 $n_Y = 60$
- 右子集: $D_N = (20, 0)$, 总数 $n_N = 20$

Step 1: $H(D_Y)$

$$P(c_1|D_Y) = \frac{20}{60} = \frac{1}{3}, \quad P(c_2|D_Y) = \frac{2}{3}$$

$$H(D_Y) = -\left(\frac{1}{3} \log_2 \frac{1}{3} + \frac{2}{3} \log_2 \frac{2}{3}\right) \approx -(0.333 \times -1.585 + 0.667 \times -0.585) \approx 0.918$$

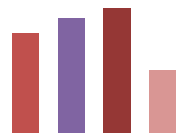
Step 2: $H(D_N)$

只有 c_1 : 纯节点

$$H(D_N) = -(1 \log_2 1 + 0 \log_2 0) = 0$$

Step 3: 条件熵

$$H(D_Y, D_N) = \frac{60}{80} \cdot 0.918 + \frac{20}{80} \cdot 0 = 0.6885$$



一、决策树模型原理


(3) 计算节点的信息增益，并判断是选择A节点还是B节点作为分裂节点？

 信息增益公式：


$$\text{Gain}(D, A) = H(D) - H(D_Y, D_N)$$

我们已知：

- $H(D) = 1$
- $H_A(D_Y, D_N) = 0.811$
- $H_B(D_Y, D_N) = 0.6885$

 对分裂点 A：

$$\text{Gain}(D, A) = 1 - 0.811 = \boxed{0.189}$$

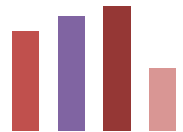
 对分裂点 B：

$$\text{Gain}(D, B) = 1 - 0.6885 = \boxed{0.3115}$$

 结论：

- 分裂点 B 的信息增益更大，即：

选择 B 作为分裂节点



一、决策树模型原理

3.2 基尼指标 (Gini index)

➤ 是另一种衡量分割点纯度的常用方法，定义如下：

$$G(D) = 1 - \sum_{i=1}^k P(c_i|D)^2$$

- 如果所划分区域是纯的，则所有点来自一类，基尼指标是0。
- 如果每一类出现的概率均等，均为 $P(c_i|D) = \frac{1}{k}$ 时，基尼指标为 $\frac{k-1}{k}$ 。
- 因此，基尼指标越高，类标签越无序，基尼指标越低，类标签越有序。

➤ 分割点的加权基尼指标

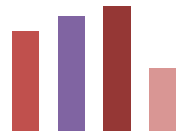
$$G(D_Y, D_N) = \frac{n_Y}{n} G(D_Y) + \frac{n_N}{n} G(D_N)$$

- 其中 n 、 n_Y 、 n_N 分别表示区域 D 、 D_Y 、 D_N 中的点数。

➤ 分割点的基尼增益

$$G(D, X) = Gini(D) - \sum_{i=1}^n w_i \times Gini(D_i)$$

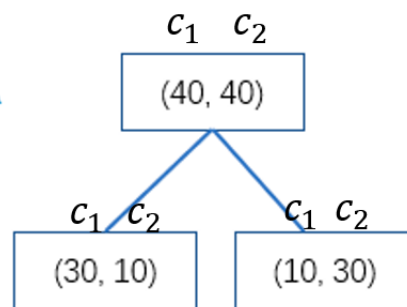
- 加权后基尼指数越低，基尼增益越高。



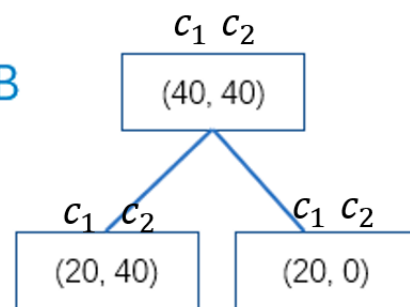
一、决策树模型原理

例2：计算上例分割点A和B的加权基尼指标。

A



B



● 分裂点 A:

- 左子集: $D_Y = (30, 10)$, 总数 $n_Y = 40$
- 右子集: $D_N = (10, 30)$, 总数 $n_N = 40$
- 总数: $n = 80$

Step 1: Gini(D_Y)

$$P(c_1) = \frac{30}{40} = 0.75, \quad P(c_2) = 0.25$$

$$Gini(D_Y) = 1 - (0.75^2 + 0.25^2) = 1 - (0.5625 + 0.0625) = 0.375$$

Step 2: Gini(D_N)

$$P(c_1) = 0.25, \quad P(c_2) = 0.75$$

$$Gini(D_N) = Gini(D_Y) = 0.375 \quad (\text{对称})$$

Step 3: Gini(A)

$$Gini_A = \frac{40}{80} \cdot 0.375 + \frac{40}{80} \cdot 0.375 = 0.375$$

● 分裂点 B:

- 左子集: $D_Y = (20, 40)$, $n_Y = 60$
- 右子集: $D_N = (20, 0)$, $n_N = 20$

Step 1: Gini(D_Y)

$$P(c_1) = \frac{20}{60} = \frac{1}{3}, \quad P(c_2) = \frac{2}{3}$$

$$Gini(D_Y) = 1 - \left(\left(\frac{1}{3} \right)^2 + \left(\frac{2}{3} \right)^2 \right) = 1 - \left(\frac{1}{9} + \frac{4}{9} \right) = 1 - \frac{5}{9} = \frac{4}{9} \approx 0.444$$

Step 2: Gini(D_N)

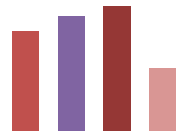
纯节点: 只有 c_1

$$Gini(D_N) = 1 - 1^2 = 0$$

Step 3: Gini(B)

依然选择 B 作为分裂节点 (Gini 更小)

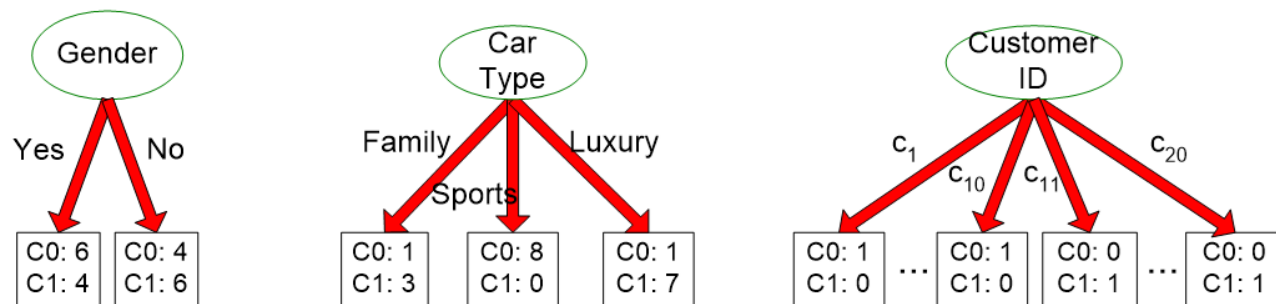
$$Gini_B = \frac{60}{80} \cdot 0.444 + \frac{20}{80} \cdot 0 = 0.333$$



一、决策树模型原理

3.3 信息增益率(Gain Ratio)

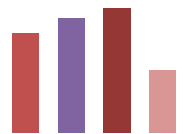
比较二元分裂与多元分裂



- 每个样本都有一个CustomerID，如果用来划分样本，每一类都是“纯”的，信息增益就会很高，但这样的分组毫无意义。
- 可见，多元分裂的信息增益会比二元分裂大，因为分裂后节点更纯。因此，如果根据信息增益来选择特征，会偏向于选择取值较多的特征，但该特征可能并不是一个好的特征。

➤ 解决策略

- 限制判断条件只能二元分裂，如**CART算法**
- 不仅考虑信息增益，还要考虑这个特征本身的取值有多分散，即在信息增益的基础上，惩罚特征熵（特征本身的取值越多，特征熵越大。），如**C4.5算法**。



一、决策树模型原理

3.3 信息增益率(Gain Ratio)

◆ 信息增益率公式:

$$\text{GainRatio}(A) = \frac{\text{Gain}(D, A)}{\text{IV}(A)}$$

其中:

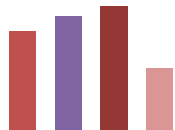
- $\text{Gain}(D, A)$ 是信息增益 (前面已经算过)
- $\text{IV}(A)$ 是**固有值 (Intrinsic Value)** , 用于衡量分裂的“复杂度”

◆ IV (固有值) 计算公式:

$$\text{IV}(A) = - \sum_{i=1}^k \frac{|D_i|}{|D|} \log_2 \left(\frac{|D_i|}{|D|} \right)$$

其中:

- D_i 是分裂后第 i 个子集
- $|D_i|$ 是它的样本数
- $|D|$ 是原始样本总数



一、决策树模型原理

➤ 结合上例进行GainRatio计算

分裂点 A:

- 子集1: 40个样本 (30,10)
- 子集2: 40个样本 (10,30)
- 总样本数: 80

1. 信息增益 (已知) :

$$\text{Gain}(A) = 0.189$$

2. 计算 IV(A):

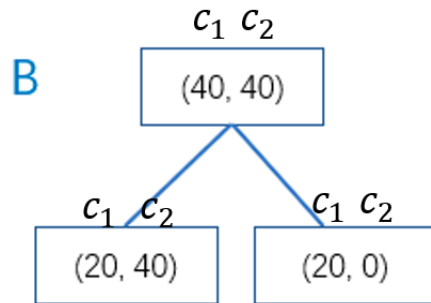
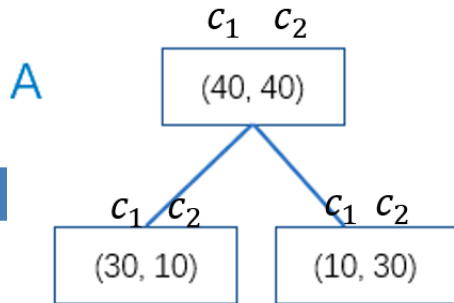
$$IV(A) = - \left(\frac{40}{80} \log_2 \frac{40}{80} + \frac{40}{80} \log_2 \frac{40}{80} \right) = -2 \cdot 0.5 \cdot \log_2 0.5 = -2 \cdot 0.5 \cdot (-1) = 1$$

3. GainRatio(A)

$$\text{GainRatio}(A) = \frac{0.189}{1} = 0.189$$

对比总结

分裂点	信息增益 Gain	IV (固有值)	信息增益率 GainRatio
A	0.189	1	0.189
B	0.3115	0.81125	0.384



分裂点 B:

- 子集1: 60个样本 (20,40)
- 子集2: 20个样本 (20,0)
- 总样本数: 80

1. 信息增益 (已知) :

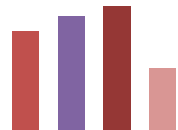
$$\text{Gain}(B) = 0.3115$$

2. 计算 IV(B):

$$IV(B) = - \left(\frac{60}{80} \log_2 \frac{60}{80} + \frac{20}{80} \log_2 \frac{20}{80} \right) = - (0.75 \log_2 0.75 + 0.25 \log_2 0.25) = 0.81125$$

3. GainRatio(B)

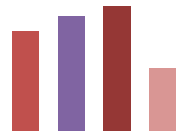
$$\text{GainRatio}(B) = \frac{0.3115}{0.81125} \approx \boxed{0.384}$$



一、决策树模型原理

➤ 常见决策树算法对比

算法名称	节点选择指标	特点	是否容易过拟合
ID3	信息增益 (Information Gain)	偏向取值较多的特征，构造简单	是 
C4.5	信息增益率 (Gain Ratio)	修正了 ID3 的偏向问题，适合多值属性，支持剪枝	不容易 
CART	基尼指数 (Gini Index)	二叉树结构，可用于分类或回归（CART回归树用的是MSE）	适中 

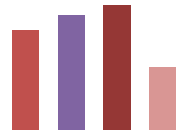


一、决策树模型原理

练习1：有如下简单的数据集，用于预测是否购买产品。数据集包含“收入、是否已婚”特征。

- (1) 编写函数分别计算信息增益、Gini增益和信息增益率等指标。
- (2) 写出“收入”和“是否已婚”两个特征的信息增益、Gini增益和信息增益率的计算结果。
- (2) 根据这些指标函数，确定最优分裂特征。

收入	是否已婚	是否购买
高	是	1
中	否	0
低	是	1
高	是	0
中	否	1



一、决策树模型原理

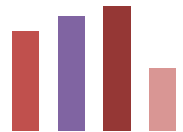
4 分割点的选择

- 对于每一个 v ，使用评价指标（信息增益，Gini增益或信息增益率）对其进行评分，记录最佳分割点和分数并返回。

4.1 数值型属性分割点的选择

- 如果 X 是数值型属性，则必须评估形式为 $X \leq v$ 的分割点，如果 v 在属性 X 的范围内取值，则 v 有无限多的选择。
- 排序选择法**：只考虑样本 D 中 X 的**两个连续不同值的中点**。因为 X 最多可以有 n 个不同的值，所以最多要考虑 **$n-1$ 个中点**。

各分割区域 类分布计数		Cheat	No		No		No		Yes		Yes		Yes		No		No		No		No			
		值排序	Annual Income																					
			60		70		75		85		90		95		100		120		125		220			
					65		72		80		87		92		97		110		122		172			
			分割点			<=	>	<=	>	<=	>	<=	>	<=	>	<=	>	<=	>	<=	>			
Yes				0	3	0	3	0	3	1	2	2	1	3	0	3	0	3	0	3	0			
No				1	6	2	5	3	4	3	4	3	4	3	4	4	3	5	2	6	1			



一、决策树模型原理

➤ 例3：基于二维鸢尾花数据集 D (数据点 $n = 150$ ，其中点标签分别为 c_1 (Iris - setosa, 50个点)、 c_2 (其它鸢尾, 100个点))，考虑 X_1 (萼片长度)和 X_2 (萼片宽度)两个属性找到最佳分割点（采用信息增益指标）。

(1) 计算数据集 D 的熵

- 计算数据集 D 的 c_i 类出现的概率

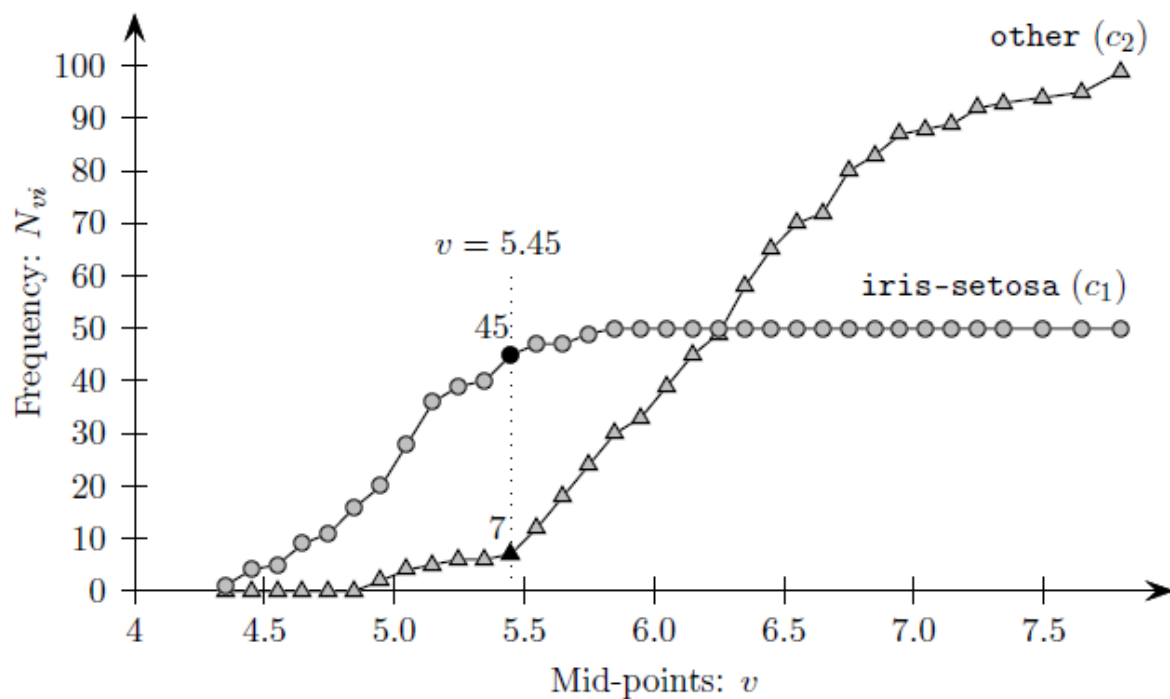
$$\hat{P}(c_1) = 50/150 = 1/3$$

$$\hat{P}(c_2) = 100/150 = 2/3$$

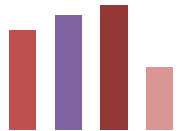
- 计算数据集 D 的熵 $H(D)$

$$H(D) = - \left(\frac{1}{3} \log_2 \frac{1}{3} + \frac{2}{3} \log_2 \frac{2}{3} \right) = 0.918$$

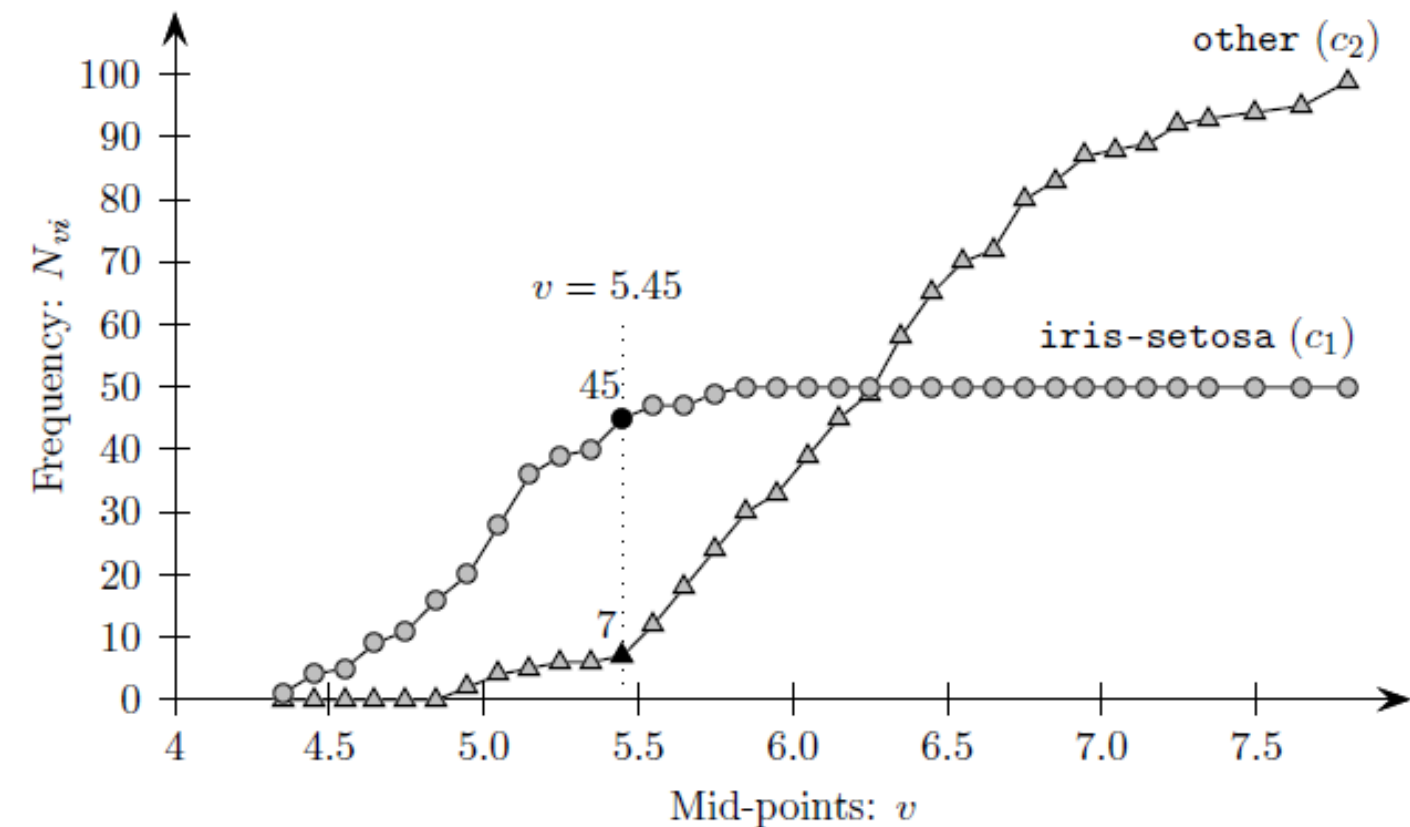
(2) 排序第1个属性 X_1 (萼片长度)的所有分割点 v ，在属性判断条件 $X_1 \leq v$ 上，考虑两个样本类的分布频率 N_{vi} 。



萼片长度所有分割点上样本的类分布



一、决策树模型原理



- 例如：思考分割点 $X_1 \leq 5.45$

$$N_{v1} = 45$$

$$N_{v2} = 7$$

$$\hat{P}(c_1|\mathbf{D}_Y) = \frac{N_{v1}}{N_{v1} + N_{v2}} = \frac{45}{45 + 7} = 0.865$$

$$\hat{P}(c_2|\mathbf{D}_Y) = \frac{N_{v2}}{N_{v1} + N_{v2}} = \frac{7}{45 + 7} = 0.135$$

$$\hat{P}(c_1|\mathbf{D}_N) = \frac{n_1 - N_{v1}}{(n_1 - N_{v1}) + (n_2 - N_{v2})} = \frac{50 - 45}{(50 - 45) + (100 - 7)} = 0.051$$

$$\hat{P}(c_2|\mathbf{D}_N) = \frac{n_2 - N_{v2}}{(n_1 - N_{v1}) + (n_2 - N_{v2})} = \frac{(100 - 7)}{(50 - 45) + (100 - 7)} = 0.949$$

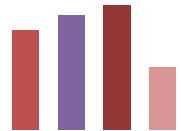
$$H(\mathbf{D}_Y) = -(0.865 \log_2 0.865 + 0.135 \log_2 0.135) = 0.571$$

$$H(\mathbf{D}_N) = -(0.051 \log_2 0.051 + 0.949 \log_2 0.949) = 0.291$$

鸢尾花：对于属性 X_1 (萼片长度)，对应类 c_1 类和 c_2 类的频率 N_{vi}

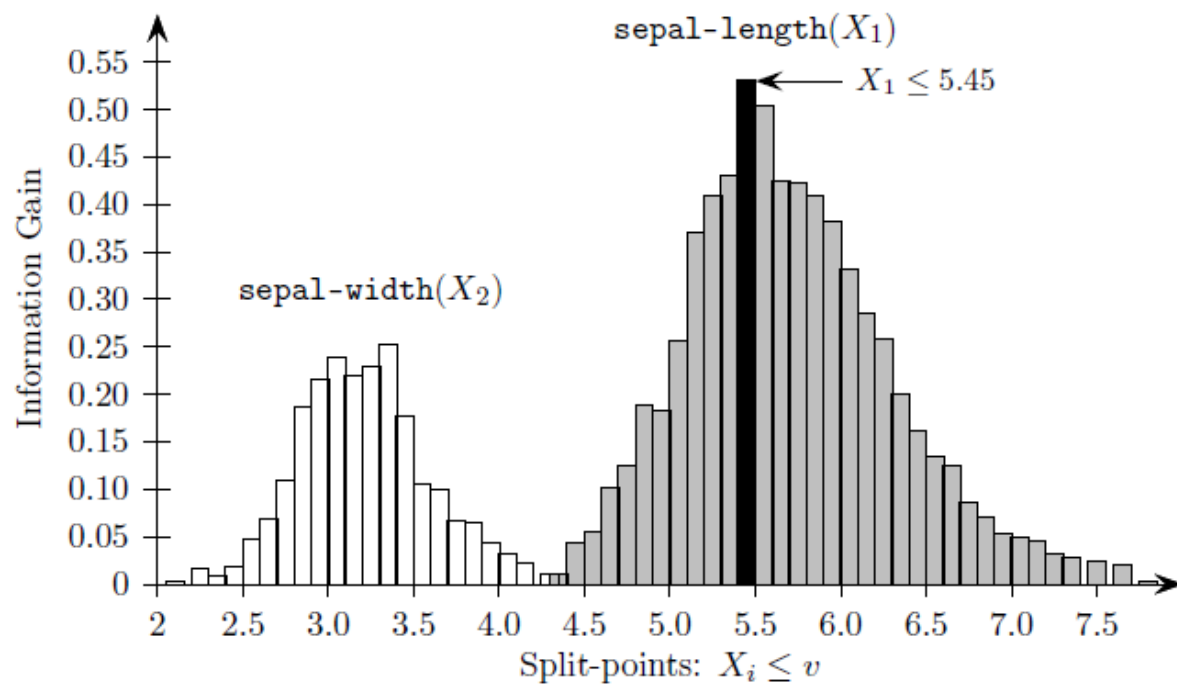
$$H(\mathbf{D}_Y, \mathbf{D}_N) = \frac{52}{150} H(\mathbf{D}_Y) + \frac{98}{150} H(\mathbf{D}_N) = 0.388$$

$$Gain = H(\mathbf{D}) - H(\mathbf{D}_Y, \mathbf{D}_N) = 0.918 - 0.388 = 0.53$$



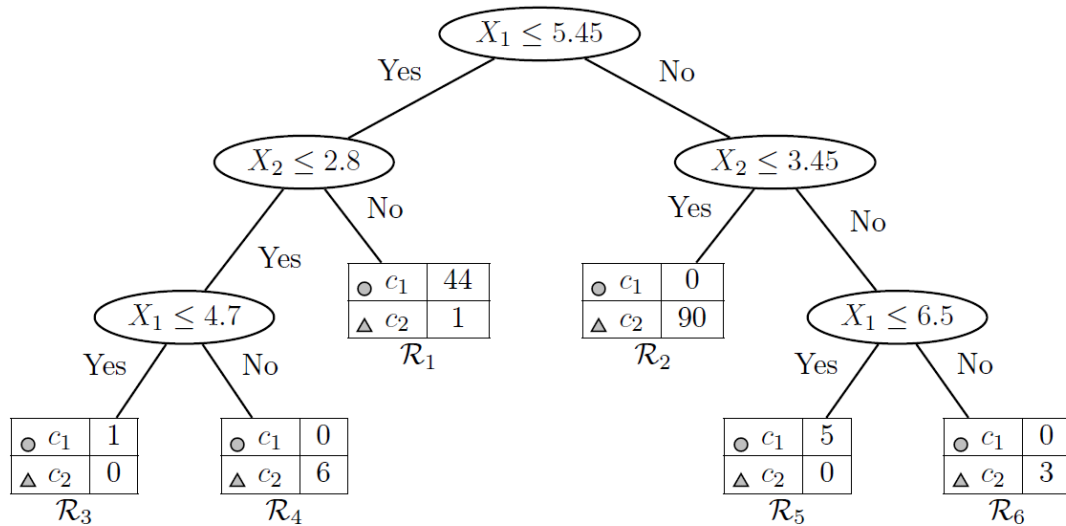
一、决策树模型原理

(3) 以类似的方式，评估属性 X_1 和 X_2 的所有分割点。



- 不同分割点关于萼片长度和萼片宽度的信息增益

- 可以观察到 $X_1 \leq 5.45$ 是最佳分割点，因此选择其作为决策树的根。



- 使用叶子节点大小阈值为5，纯度阈值为0.95

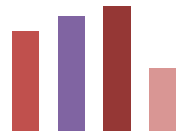
(4) 树的递归生长不断进行，直到产生最终决策树和分割点。

DECISIONTREE (\mathbf{D}, η, π): • η : 叶子节点大小阈值; π : 叶子节点纯度阈值

```
1  $n \leftarrow |\mathbf{D}|$  // partition size
2  $n_i \leftarrow |\{\mathbf{x}_j | \mathbf{x}_j \in \mathbf{D}, y_j = c_i\}|$  // size of class  $c_i$ 
3  $\text{purity}(\mathbf{D}) \leftarrow \max_i \{\frac{n_i}{n}\}$ 
4 if  $n \leq \eta$  or  $\text{purity}(\mathbf{D}) \geq \pi$  then // stopping condition
5      $c^* \leftarrow \arg \max_i \{\frac{n_i}{n}\}$  // majority class 找到数据集中占比最大的类别  $c^*$ 
6     create leaf node, and label it with class  $c^*$  创建一个叶节点, 并用类别  $c^*$  标记,
7     return

8  $(\text{split-point}^*, \text{score}^*) \leftarrow (\emptyset, 0)$  // initialize best split-point
9 foreach (attribute  $X_j$ ) do
10     if ( $X_j$  is numeric) then 如果  $x_j$  是数值型属性
11          $(v, \text{score}) \leftarrow \text{Evaluate-Numeric-Attribute}(\mathbf{D}, X_j)$  调用函数来评估基于该数值属性划分数据集的情况
12         if  $\text{score} > \text{score}^*$  then  $(\text{split-point}^*, \text{score}^*) \leftarrow (X_j \leq v, \text{score})$ 
13     else if ( $X_j$  is categorical) then 如果  $x_j$  是分类型属性
14          $(V, \text{score}) \leftarrow \text{Evaluate-Categorical-Attribute}(\mathbf{D}, X_j)$  调用函数来评估基于该分类型属性划分数据集的情况,
15         if  $\text{score} > \text{score}^*$  then  $(\text{split-point}^*, \text{score}^*) \leftarrow (X_j \in V, \text{score})$ 

    // partition  $\mathbf{D}$  into  $\mathbf{D}_Y$  and  $\mathbf{D}_N$  using  $\text{split-point}^*$ , and call
    recursively
16  $\mathbf{D}_Y \leftarrow \{\mathbf{x} \in \mathbf{D} \mid \mathbf{x} \text{ satisfies } \text{split-point}^*\}$  根据找到的最佳划分点  $\text{split-point}^*$ , 将数据集  $\mathbf{D}$  划分为两个子集  $\mathbf{D}_Y$  和  $\mathbf{D}_N$ 
17  $\mathbf{D}_N \leftarrow \{\mathbf{x} \in \mathbf{D} \mid \mathbf{x} \text{ does not satisfy } \text{split-point}^*\}$ 
18 create internal node  $\text{split-point}^*$ , with two child nodes,  $\mathbf{D}_Y$  and  $\mathbf{D}_N$  创建一个内部节点 其两个子节点分别为  $\mathbf{D}_Y$  和  $\mathbf{D}_N$ 
19  $\text{DecisionTree}(\mathbf{D}_Y); \text{DecisionTree}(\mathbf{D}_N)$  对划分得到的两个子集  $\mathbf{D}_Y$  和  $\mathbf{D}_N$  分别递归调用 DecisionTree 函数, 继续进行划分, 直到满足停止条件.
```

二、Sklearn中的决策树分类模型

- 使用 Sklearn (scikit-learn) 中的决策树分类模型 (DecisionTreeClassifier) 做分类任务时，通常包括以下几个具体操作步骤：

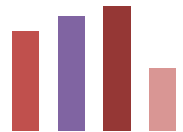
📌 一、导入必要的库

```
from sklearn.datasets import load_iris # 示例数据集
from sklearn.tree import DecisionTreeClassifier
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
```

📌 二、准备数据

```
# 加载数据
data = load_iris()
X = data.data # 特征
y = data.target # 标签

# 拆分训练集和测试集（例如70%训练，30%测试）
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)
```



二、Sklearn中的决策树分类模型



三、创建并训练模型

创建决策树分类器

```
clf = DecisionTreeClassifier(criterion='gini', max_depth=None, random_state=42)
```

训练模型

```
clf.fit(X_train, y_train)
```

• 常用参数说明:

- `criterion='gini'` 或 `'entropy'`: 用于衡量节点纯度的指标
- `max_depth`: 控制树的最大深度, 防止过拟合
- `random_state`: 设置随机种子, 保证结果可复现



四、模型预测

```
y_pred = clf.predict(X_test)
```

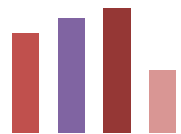


五、模型评估

准确率评估

```
accuracy = accuracy_score(y_test, y_pred)
```

```
print("准确率:", accuracy)
```

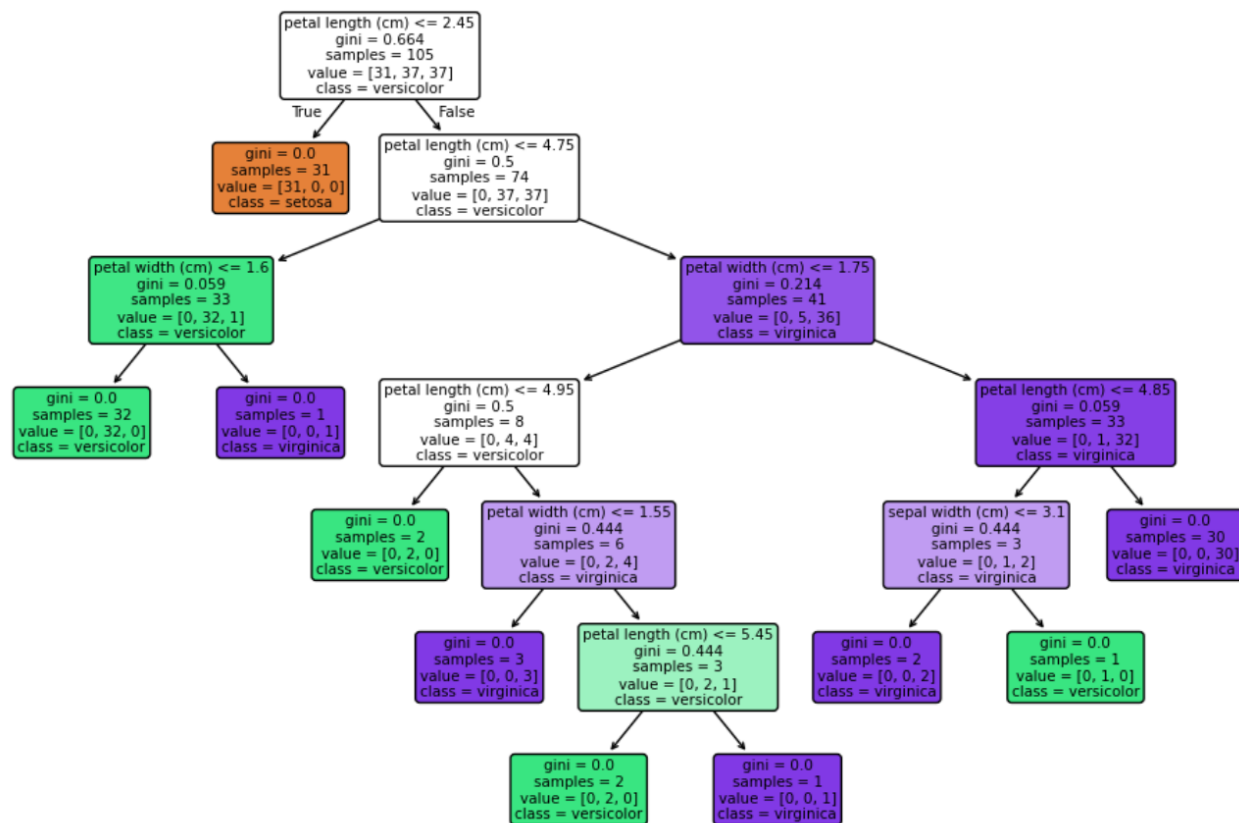
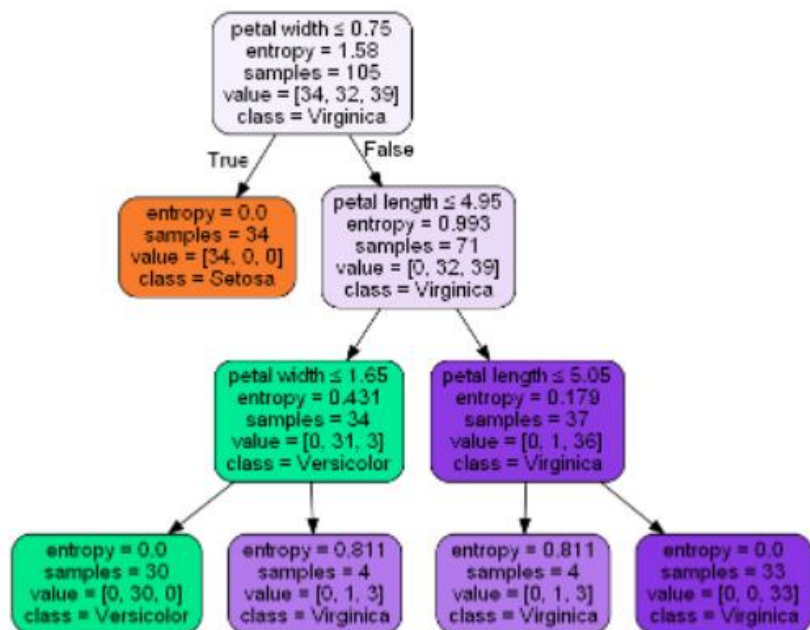


二、Sklearn中的决策树分类模型

六、可选：可视化决策树（可视化树结构）

```
from sklearn.tree import plot_tree
import matplotlib.pyplot as plt
```

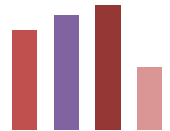
```
plt.figure(figsize=(12, 8))
plot_tree(clf, filled=True, feature_names=data.feature_names,
class_names=data.target_names)
plt.show()
```



```
print("决策树的深度为:", clf.get_depth())
```

因为 `max_depth=None`，模型会自动增长到纯叶节点（直到不能再分）

```
clf = DecisionTreeClassifier(max_depth=3)
```

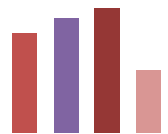


二、Sklearn中的决策树分类模型

练习：

在白葡萄酒数据集winequality-white.csv上增加一个quality_label质量标签，质量分数为3、4和5的葡萄酒质量较低；6分和7分为中等质量；8分和9分是高质量的葡萄酒。然后，构建决策树模型对葡萄酒的质量进行判断。

- (1) 报告模型在训练集和测试集的准确率。
- (2) 报告决策树的深度。
- (3) 分析目前模型存在的问题。



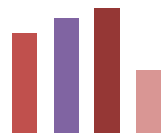
三、特征选择

在决策树模型中，**特征选择**的目的是从众多特征中挑选出那些最有助于模型预测的特征，以提高模型的性能、减少计算量，并避免过拟合。决策树本身具有内建的特征选择机制，通过计算每个特征的重要性来评估特征对预测结果的贡献。

★使用决策树的特征重要性

决策树算法（如 `DecisionTreeClassifier` 或 `DecisionTreeRegressor`）会根据每个特征对决策过程的贡献来自动评估特征的重要性。特征重要性衡量了每个特征对模型预测的影响，它基于特征在各个分裂节点上对不纯度（如Gini Impurity或信息增益）的减少。

- **特征重要性**：在训练完决策树模型后，可以使用 `feature_importances_` 属性获取每个特征的重要性值。值越大，特征对模型的贡献越大。
- **筛选特征**：根据特征重要性值，可以选择那些重要性值较高的特征来构建模型，从而进行特征选择。



三、特征选择

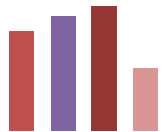
在决策树模型中，`feature_importances_` 属性用来评估每个特征对决策树模型预测的贡献程度。具体来说，`feature_importances_` 是通过衡量每个特征在决策树训练过程中所引入的**不纯度减少量**来计算的。不纯度的减少量（或者说是分裂增益）反映了一个特征在分裂数据时的有效性，即它对模型预测的影响程度。

1. 什么是“不纯度”

决策树模型通过节点分裂来对数据进行分类或回归。在每个节点上，决策树都会选择一个特征来划分数据，选择的标准是使得子节点比父节点的不纯度更低。

常见的“不纯度”度量标准有：

- **基尼不纯度 (Gini Impurity)**：用于分类问题
- **信息增益 (Entropy)**：用于分类问题
- **均方误差 (MSE)**：用于回归问题



三、特征选择

2. 特征重要性的计算方法

特征重要性基于决策树在训练过程中对每个特征的贡献度进行评估。其计算过程如下：

步骤 1: 计算分裂时的不纯度减少量（信息增益或基尼不纯度的减少）

- 在训练过程中，每次决策树分裂节点时，会选择一个特征，并基于该特征的某个值（如一个阈值）将数据分成两个子集。
- 不纯度减少量**（Information Gain 或 Gini Impurity reduction）即为分裂前后的不纯度差值。分裂前是父节点的不纯度，分裂后是两个子节点的不纯度加权平均值。

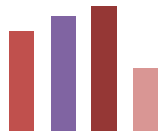
步骤 2: 计算每个特征的贡献

每次使用某个特征进行节点分裂时，计算该特征带来的不纯度减少量。然后，累积所有使用该特征的分裂节点的不纯度减少量。

具体来说，假设某个特征在树的多个节点中参与了分裂，特征对模型的贡献（即重要性）是通过在所有节点中计算该特征引起的不纯度减少量的总和来评估的。

步骤 3: 归一化

为了让所有特征的重要性值的和为1，最终的特征重要性是将每个特征的贡献度除以所有特征的贡献度之和，得到一个**归一化的值**。



三、特征选择

3. 举个例子

假设我们有一个数据集，决策树通过以下步骤来进行训练：

1. **根节点**：在根节点上计算数据集的基尼不纯度为 0.5。
2. **第一次分裂**：根据特征 A 进行分裂，分裂后子节点的基尼不纯度分别为 0.3 和 0.4。基尼不纯度减少了 $0.5 - 0.35 = 0.15$ 。
3. **第二次分裂**：根据特征 B 进行分裂，分裂后子节点的基尼不纯度分别为 0.2 和 0.1。基尼不纯度减少了 $0.5 - 0.15 = 0.35$ 。

在这个例子中，特征 A 和特征 B 都对模型的预测做出了贡献，但是特征 B 对决策树的影响更大，因为它的分裂导致了更大的不纯度减少。

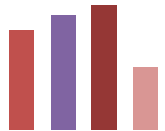
特征 A 和特征 B 的重要性计算：

- **特征 A**：贡献了 0.15 的基尼不纯度减少。
- **特征 B**：贡献了 0.35 的基尼不纯度减少。

最终，特征 A 和特征 B 的重要性将分别为：

$$\text{Feature importance of A} = \frac{0.15}{0.15 + 0.35} = 0.3$$

$$\text{Feature importance of B} = \frac{0.35}{0.15 + 0.35} = 0.7$$



三、特征选择

```
from sklearn.tree import DecisionTreeClassifier
import pandas as pd
```

```
# 假设已加载了数据集
# X = df.drop(['target'], axis=1) # 特征
# y = df['target'] # 目标标签
```

```
# 训练决策树模型
clf = DecisionTreeClassifier(random_state=42)
clf.fit(X, y)
```

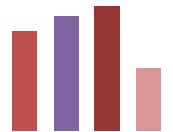
```
# 获取特征的重要性
importances = clf.feature_importances_
feature_names = X.columns
```

```
# 筛选重要特征（假设重要性大于0.05）
selected_features = [feature for feature, importance in zip(feature_names, importances) if importance > 0.05]
```

```
print("重要特征: ", selected_features)
```

步骤:

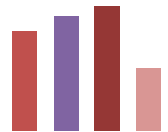
1. 使用 `clf.feature_importances_` 获得每个特征的重要性。
2. 按照特征的重要性进行筛选，选择对模型影响最大的特征。



三、特征选择

练习3：在例2的基础上，根据特征重要性 >0.1 。

- (1) 被选中的重要特征是（）
- (2) 选择重要的特征后，重新训练模型，报告模型在训练集和测试集上的准确率。
- (3) 绘制特征重要性。



四、线性模型与非线性模型对比

什么是线性模型？

输入和输出之间的关系可以用一条直线（或高维的超平面）表达出来。

数学表达：

以二维为例，最简单的线性分类模型：

$$y = w_1x_1 + w_2x_2 + b$$

- 其中 w_1, w_2 是权重, b 是偏置;
- 整体是一个“线性组合”。

直观理解：

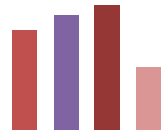
- 把输入特征乘上权重，加起来，输出结果；
- 画出来就是一条“直线”或“超平面”；所有的“分界线”都是直的。

优点：

- 简单，训练快；
- 适合特征和输出之间是“线性关系”的问题；
- 不容易过拟合（但容易欠拟合）。

缺点：

- 表达能力有限，无法处理复杂的非线性关系；
- 对异常值比较敏感。



四、线性模型与非线性模型对比



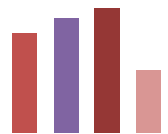
什么是非线性模型？

非线性模型就是：输入和输出之间不是线性关系，不能用一条直线（或超平面）来分开数据。

常见的非线性模型包括：

模型	非线性来源
决策树	切割特征空间
KNN	距离度量 + 局部判断
SVM（非线性核）	核函数变换
神经网络	激活函数（如ReLU、Sigmoid）

思考：Logistic 回归是线性模型还是非线性模型？



四、线性模型与非线性模型对比

✓ 为什么 Logistic 回归是线性模型?

💡 本质逻辑:

Logistic 回归的决策边界是基于特征的**线性组合**建立的。

它的模型形式如下:

$$P(y = 1 | \mathbf{x}) = \frac{1}{1 + e^{-(\mathbf{w}^\top \mathbf{x} + b)}}$$

- $\mathbf{w}^\top \mathbf{x} + b$: 是特征的**线性组合**;
- 整个模型对输入特征是**线性的**, 只不过通过 sigmoid 函数把输出映射到了 0 到 1 之间 (概率);
- 最终分类是通过设置一个**阈值 (比如 0.5) **进行划分。

🔧 决策边界是“线”的

如果你设定:

$$P(y = 1 | \mathbf{x}) = 0.5$$

求出 \mathbf{x} 必须满足的解析形式

$$\mathbf{w}^\top \mathbf{x} + b = 0$$

这是一条标准的**线性超平面** (二维是直线, 三维是平面, 更高维是超平面)。

代数推导

设定分类阈值 0.5:

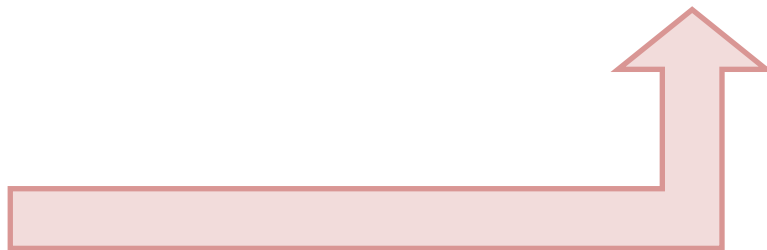
$$\frac{1}{1 + \exp(-(\mathbf{w}^\top \mathbf{x} + b))} = 0.5.$$

两边取倒数再移项:

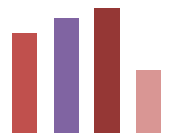
$$1 + \exp(-(\mathbf{w}^\top \mathbf{x} + b)) = 2 \implies \exp(-(\mathbf{w}^\top \mathbf{x} + b)) = 1.$$

对数两边:

$$-(\mathbf{w}^\top \mathbf{x} + b) = \ln 1 = 0 \implies \boxed{\mathbf{w}^\top \mathbf{x} + b = 0}.$$

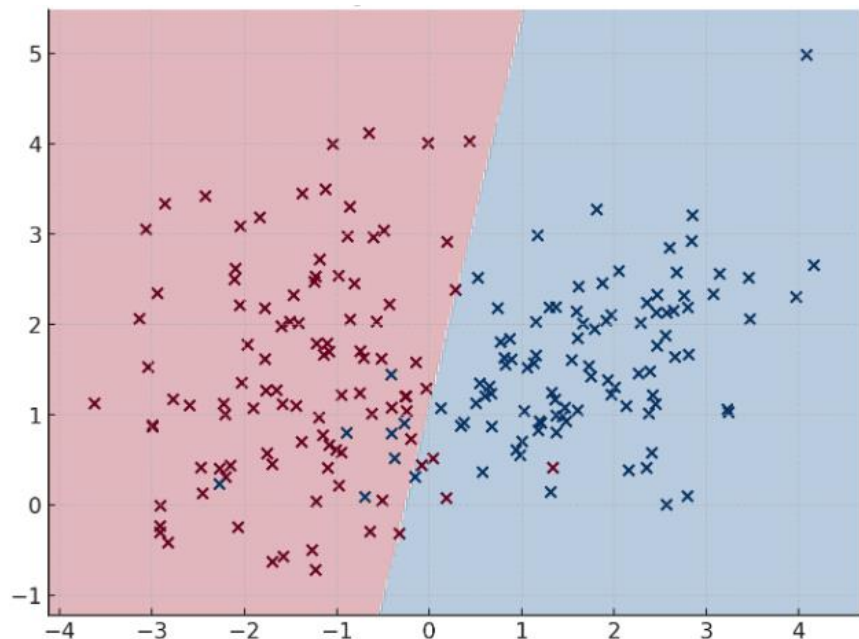


尽管 sigmoid 函数是非线性的, 但那只是对输出的非线性变换, 模型本身依然是对输入特征的线性函数。



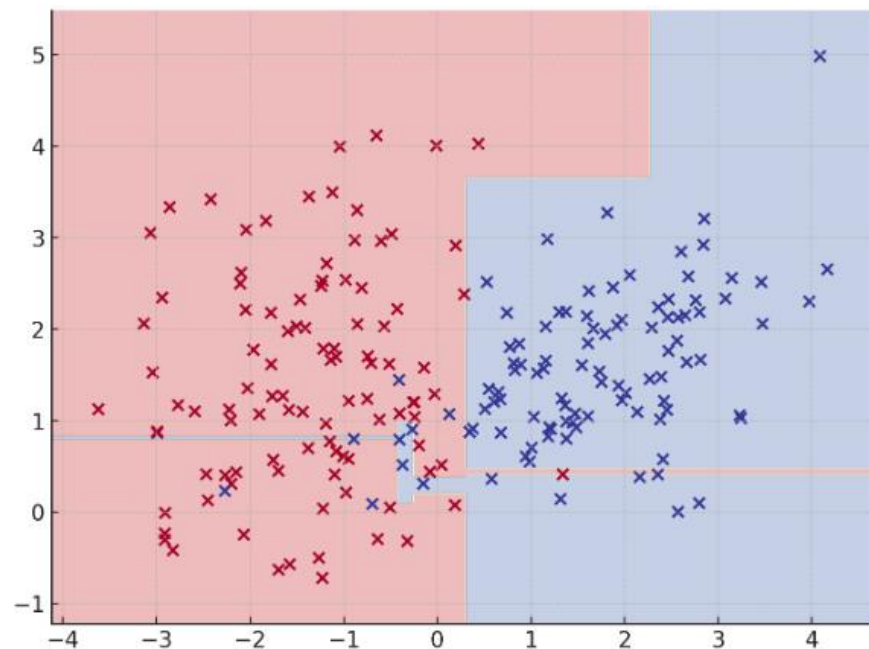
四、线性模型与非线性模型对比

Logistic 回归的决策边界



- 背景颜色代表模型预测的分类区域；
- 点表示样本，颜色表示它们的真实类别；
- 中间那条“分界线”其实是一个线性超平面（在二维空间里就是一条直线）。

决策树的分类边界



- 决策边界是非线性的，呈现出**阶梯状、块状**的分割方式；
- 决策树是通过特征空间“逐层切割”来分类的，所以形成的是“**矩形块**”；
- 与 Logistic 回归那种“直线型”分界不同，决策树能更好处理复杂、非线性结构的数据。