交通大数据

决策树

- 刘志远教授
- zhiyuanl@seu.edu.cn

决策树

❑ 学习目标

- 学习决策树特征选择的指标
  - ➢ 了解信息熵的由来和意义
  - ➢ 熟悉信息的测量方法
  - ➢ 掌握熵的表达式及其证明
- 掌握经典的决策树算法
- 了解决策树剪枝的原因、策略及其异同点
- 掌握应用决策树算法解决实际问题的方法

决策树

❑ 学习目标

- **学习决策树特征选择的指标**
  - 了解信息熵的由来和意义
  - 熟悉信息的测量方法
  - 掌握熵的表达式及其证明
- 掌握经典的决策树算法
- 了解决策树剪枝的原因、策略及其异同点
- 掌握应用决策树算法解决实际问题的方法

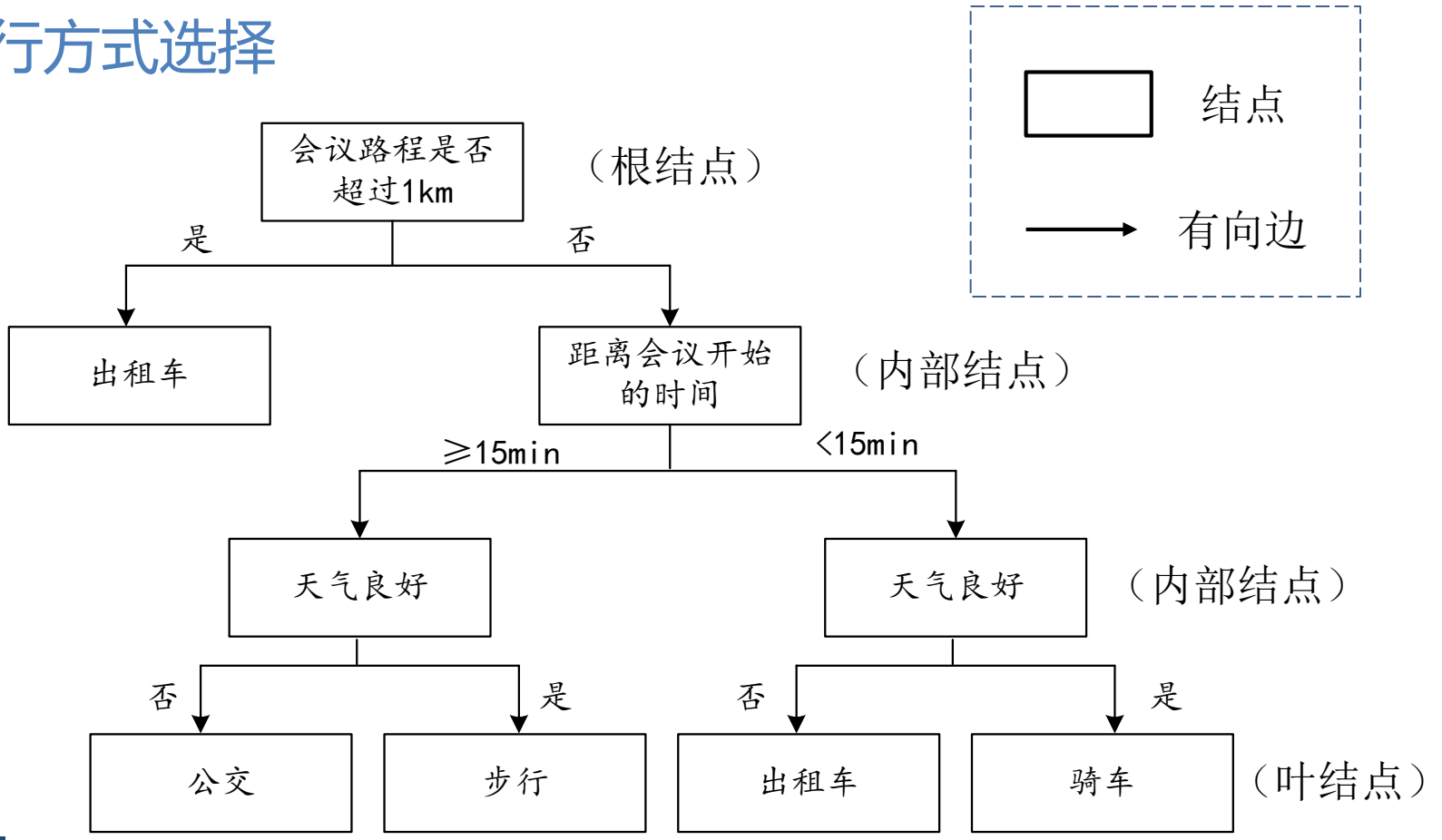
背景

- 出行方式选择

编号	会议路程（是否大于1km）	距离会议开始的时间（是否大于15min）	天气是否良好	出行方式
1	否	是	否	公交
2	否	是	是	步行
3	否	否	否	出租车
4	否	否	是	骑车
5	是	是	否	出租车
6	是	是	是	出租车
7	是	否	否	出租车
8	是	否	是	出租车

背景

出行方式选择

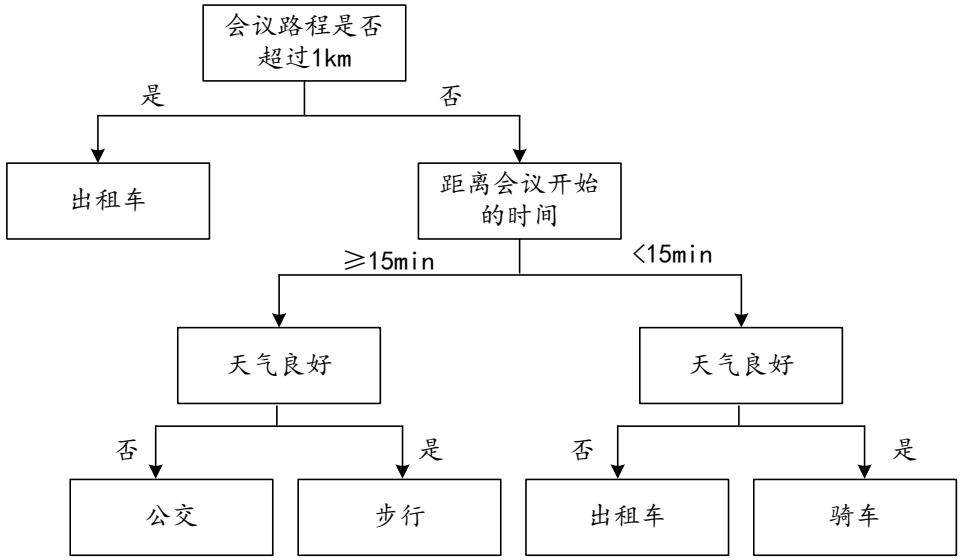


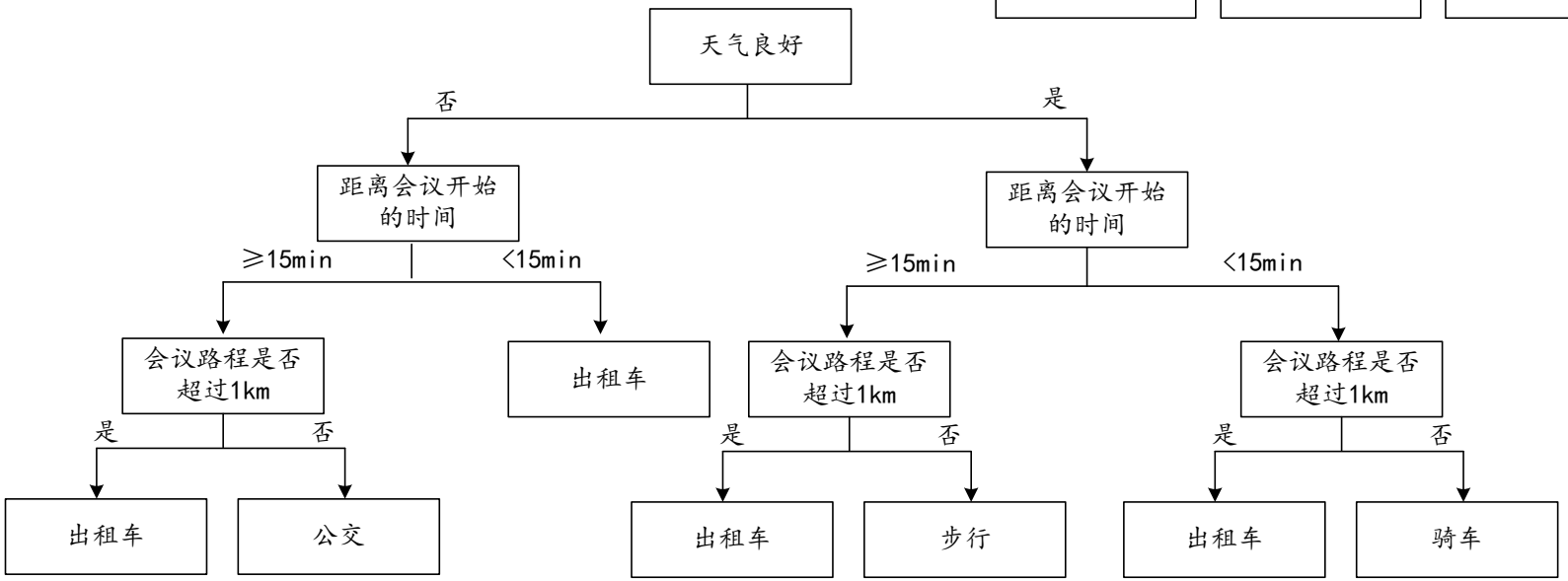
试一试

表征这一数据集，树的结构是唯一的吗？

背景

- 出行方式选择（特征选择）



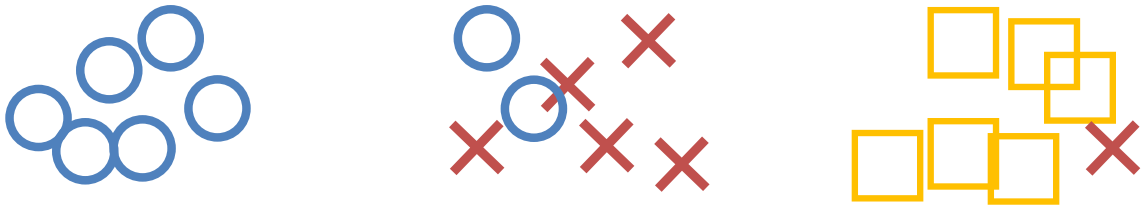


特征选择的顺序不同，则完成同样的分类，需要进行的分叉次数不同

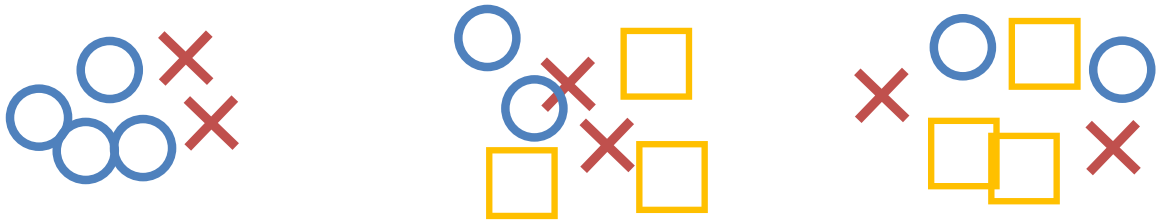
特征选择

- 为提高学习效率，决策树每一步需要选择**最优特征**

基于特征A的分类结果



基于特征B的分类结果



试一试

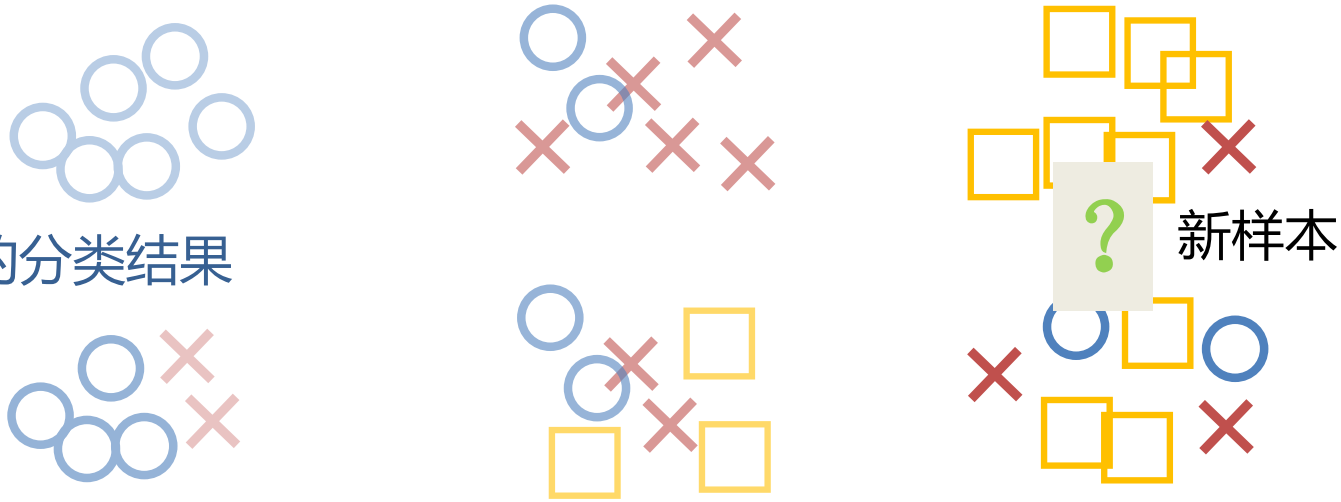
特征A和特征B哪一个分类效果更优？如果要构造特征选择标准，你是否有思路？

特征选择

- 为提高学习效率，决策树每一步需要选择**最优特征**
  - 什么是最优特征?
    - 剩余的特征中，对数据的分类能力最佳的特征。
  - 怎样体现分类能力最佳?
    - 在该特征空间划分下，各个子集（该特征的各个取值）所包含的样本尽可能属于同一类（纯度最高）。
  - 为什么纯度越高越好（见第一个和第二个例子）？

特征选择

基于特征A的分类结果



基于特征B的分类结果

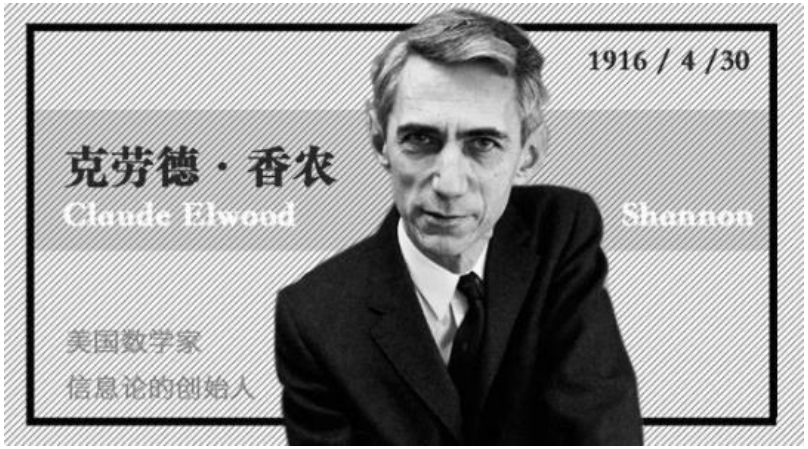
根据特征A，新样本比较有可能是 □；根据特征B则无法猜测类别

我们可以认为特征B为分类任务提供的**信息**更小，基于特征B得到的分类结果**不确定性**更大

思考：如何度量一个集合的纯度、某个特征分类出来的多个集合的纯度？

答：信息熵、条件熵、熵增益。

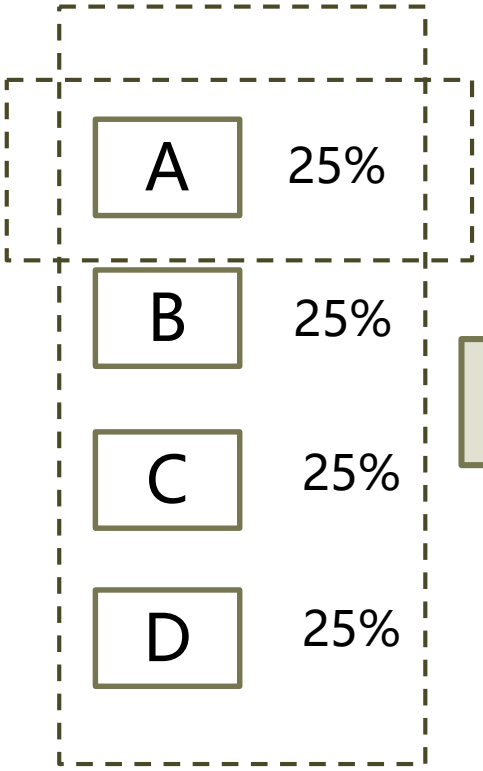
信息熵——不确定性的度量



- 基本概念（信息论）
  - 信息熵（information entropy）
    - 不确定性的度量
  - 信息（information）
    - 能够消除不确定性的事物叫做信息
  - 噪音（noise）
    - 不能帮助消除不确定性，是信息获取的干扰。
  - 数据（data）
    - 噪音+信息。需要用知识将其分离。

信息熵——不确定性的度量

- 基本概念：以选择题为例



概率：某件事情某个可能情况的确定性

熵最大 — **熵最小**

信息

0%

0%

100%

0%

熵：当一件事情有多种可能情况时，这件事情对某人而言究竟是哪种情况发生的不确定性叫做熵。

获取信息 = 消除熵

信息熵——不确定性的度量

❑ 基本概念

- 信息与**媒介无关**。例如：“C为正确答案”的信息，通过视觉或者听觉传播，是一样的。
- 信息是**相对于个体的**，接收到的信息是相对于观察者已经对于该事件的实际了解程度而言的。例如：小红已知C时正确答案，“C为正确答案”无法为她提供信息。
- 信息是**相对于事件**的。例如：小明对于“正确答案属于A,B,C,D”的熵与他对于“正确答案属于AB还是CD”的熵不同（25%*4 vs. 50%*2）。

信息熵——不确定性的度量

❑ 如何测量信息

质量测量案例：

参照物体：
B, 定义为1kg

待测物体：
m= ? kg

$$m = n \times B$$
$$n = m/B\ (kg)$$

类比

信息测量：

$$m = 2^n$$
$$n = log_2 m\ (bit)$$

抛硬币正反信息作为参照：
2种情况，定义为1bit

待测信息：
m= ? bit

指数关系：抛3枚硬币共2^3，即8种情况

A	25%
B	25%
C	25%
D	25%

熵=$log_2 4$=2 bit

由于参照的是等可能事件，这种计算方法仅对等可能事件成立

信息熵——不确定性的度量

❑ 如何测量信息

A	1/6
B	1/6
C	1/2
D	1/6

- 对于非等可能事件，分别测量每种可能情况的信息量，乘以各自的概率之后再相加。

$$\text{熵}=\frac{1}{6}log_2 m_A+\frac{1}{6}log_2 m_B+\frac{1}{2}log_2 m_C+\frac{1}{6}log_2 m_D$$

- 概率的倒数 = 等可能情况的个数 ： $m_A=\frac{1}{p_A}$
- 总信息量（信息熵）：

$$H=\sum_i p_i log_2\frac{1}{p_i}=-\sum_i p_i log_2 p_i$$

$$-\frac{1}{6}\log\frac{1}{6}\times 3$$

$$-\frac{1}{2}\log\frac{1}{2}=1.79\ bit$$

- 由于$x\to 0, xlogx\to 0$，约定$0log0=0$

知识1：如果有人告诉你C是答案。
知识2：如果有人告诉你C是答案的可能是1/2，其他几个答案等可能。
两个知识的信息量如何度量？

❑ 如何测量信息

事件1：信息熵=2bit

A	1/4
B	1/4
C	1/4
D	1/4

知识1

事件3：信息熵=0bit

A	0
B	0
C	1
D	0

知识1提供的信息量

$= 2 - 0 = 2bit$

= 事件1的信息熵

试一试

知识2提供的信息量是多少？

知识2

事件2：信息熵=1.79bit

A	1/6
B	1/6
C	1/2
D	1/6

获取信息 = 消除熵

信息熵=当前事件→不确定度为0所需的信息量

信息熵的另一种阐述方式

❑ 随机事件自身属性的视角——自信息

自信息（self-information）或**信息量**主要描述随机事件集合中，某一个**事件自身的属性。**

下面哪条信息提供的信息量（价值量）更大：

- 太阳明天将会从东边升起（概率$\mathrm{p}=1$）
- 明天的足球世界杯决赛，意大利队将会赢巴西队（概率$\mathrm{p}=0.5$）
- 由10个数字组成的彩票大奖，明天开奖，中奖号码将会是736 542 976 3 （概率$\mathrm{p}=0.0000000001$）

信息熵的另一种阐述方式

- 随机事件自身属性的视角——自信息

描述信息量的函数f应该满足以下条件：

1. 非负性：$f \geq 0$
2. $f(p_i)$应是概率的单调递减函数，即当$p_1 > p_2$时$f(p_1) < f(p_2)$，
3. 当$p_1 = 1$时，$f(p_1) = 0$
4. 当$p_1 = 0$时，$f(p_1) = \infty$
5. 两个独立事件的联合信息量应等于它们分别的信息量之和

一个满足上述条件的函数：$I(x_i) = -\log(p(x_i))$

信息熵的另一种阐述方式

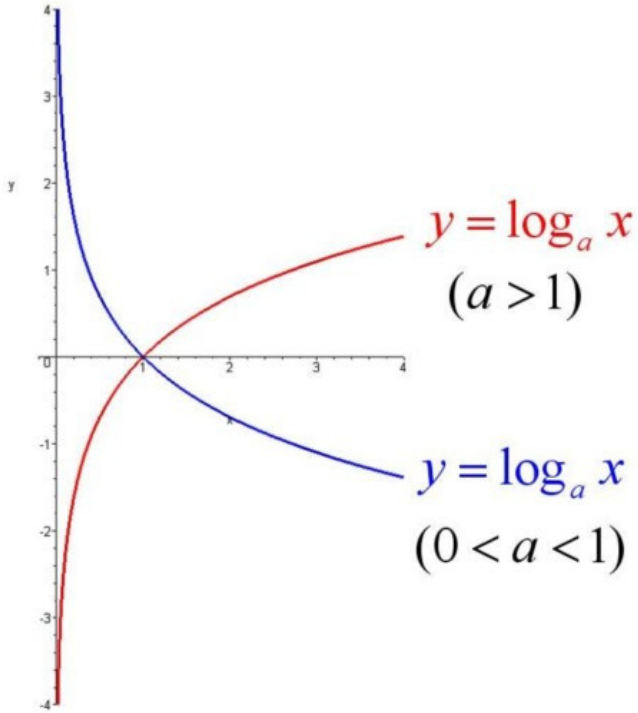
- 随机事件自身属性的视角——自信息

自信息的**定义**：

$$I(x_i) = -\log(p(x_i))$$

自信息的特征：

- 自信息是随机变量
- 自信息的含义包含两个方面：
  - 事件发生前：事件发生的不确定性
  - 事件发生后：事件包含的信息量



信息熵的另一种阐述方式

- 随机事件自身属性的视角——自信息

随机变量的熵可定义为自信息的平均值，记为

$$H(X) = \mathop{\mathrm{E}}_{p(x)}[I(x)] = -(x+a)^n = -\sum_x p(x) log p(x)$$

简记为 $H(X) = H(p_1, p_2, \dots, p_n)$

信息熵的含义：

- 输出样本前：随机变量的平均不确定性
- 输出样本后：一个样本所提供的平均信息量
- 表示随机性大小：熵越大，随机性越大
- 输出样本后，不确定性就解除：解除信源不确定性所需的信息量

信息熵——不确定性的度量

❑ 信息熵计算公式推导（补充阅读）

记熵的表达式为$H(p_1, p_2, \dots, p_n)$，其中一共有n种可能的情况，每种情况的概率为p_i

这一函数需要满足以下条件：

- $H(p_1, p_2, \dots, p_n) \leq H(\frac{1}{n}, \frac{1}{n}, \dots, \frac{1}{n})$
- $H(\frac{1}{n}, \frac{1}{n}, \dots, \frac{1}{n})$是自然数$n$的单调递增函数
- 如果一个不确定事件分解为几个持续的事件，则原先事件的不确定性等于持续事件不确定性的加权和
- 对于固定自然数n，不确定性函数H是$(p_1, p_2, \dots, p_n)$的一个连续函数

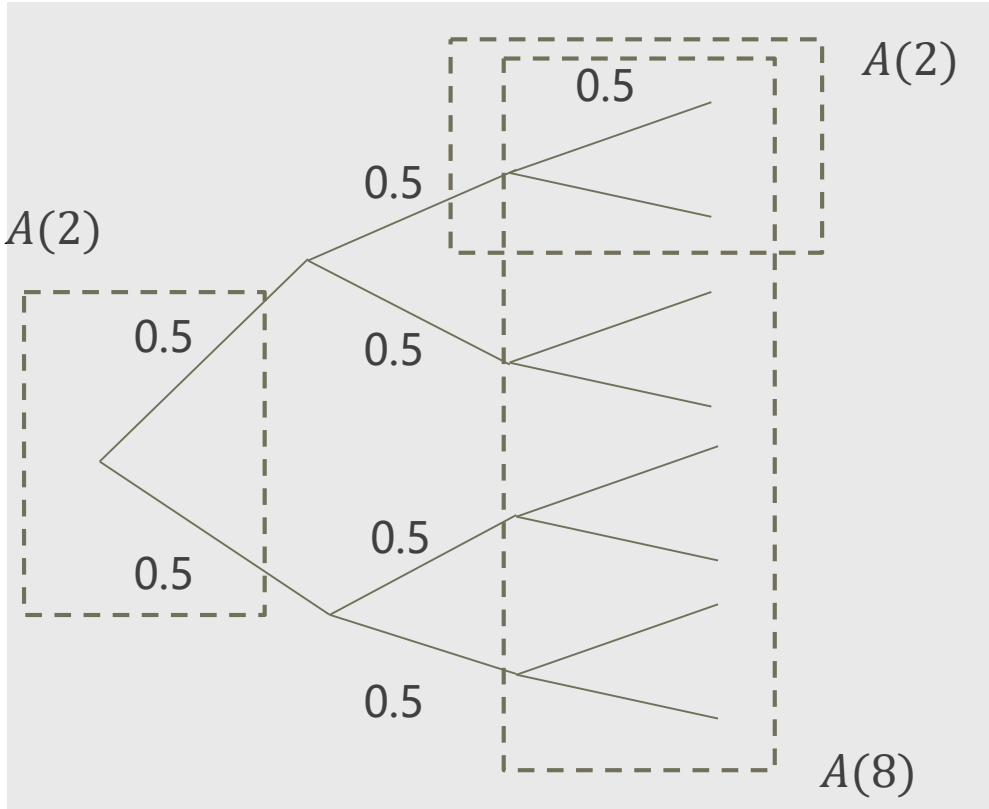
信息熵——不确定性的度量

❑ 信息熵计算公式推导（补充阅读）

首先证明$H(\frac{1}{n},\frac{1}{n},\dots,\frac{1}{n}) = -\sum \frac{1}{n} log_2 \frac{1}{n}$

- 记$H\left(\frac{1}{n},\frac{1}{n},\dots,\frac{1}{n}\right) = A(n)$
- 如果一个不确定事件分解为几个持续的事件，则原先事件的不确定性等于持续事件不确定性的加权和：

$$A(S^m) = mA(S) \quad (1)$$



$$A(2^3) = A(2) + \left[\frac{1}{2}A(2) + \frac{1}{2}A(2)\right] + \left[\frac{1}{4}A(2) + \frac{1}{4}A(2) + \frac{1}{4}A(2) + \frac{1}{4}A(2)\right] = 3A(2)$$

信息熵——不确定性的度量

❑ 信息熵计算公式推导（补充阅读）

首先证明$H(\frac{1}{n},\frac{1}{n},\dots,\frac{1}{n})=-\sum\frac{1}{n}log_2\frac{1}{n}$

- 假设四个正整数t,s,n,m满足不等式：$s^m\le t^n<s^{m+1}$，
  - 对其求对数得：$mlog_2s\le nlog_2t<(m+1)log_2s$
  - 即：$\frac{m}{n}\le\frac{log_2t}{log_2s}<\frac{m+1}{n}$，所以$\left|\frac{m}{n}-\frac{log_2t}{log_2s}\right|<\frac{1}{n}$ (2)
- 由熵的条件（2），$A(k)$是k的递增函数，所以$mA(s)\le nA(t)<(m+1)A(s)$，类比公式（2）同理可得$\left|\frac{m}{n}-\frac{A(t)}{A(s)}\right|<\frac{1}{n}$ (3)

信息熵——不确定性的度量

❑ 信息熵计算公式推导（补充阅读）

首先证明$H(\frac{1}{n},\frac{1}{n},\dots,\frac{1}{n})=-\sum\frac{1}{n}log_2\frac{1}{n}$

- 结合公式（2）（3），$\left|\frac{A(t)}{A(s)}-\frac{log_2t}{log_2s}\right|<\left|\frac{m}{n}-\frac{log_2t}{log_2s}\right|+\left|\frac{m}{n}-\frac{A(t)}{A(s)}\right|=\frac{2}{n}$
- 由于n是任意的，$\left|\frac{A(t)}{A(s)}-\frac{log_2t}{log_2s}\right|=0$，$A(t)=Clog_2t$
- 取$C=1$，$H\left(\frac{1}{n},\frac{1}{n},\dots,\frac{1}{n}\right)=\log_2 n=-\sum\frac{1}{n}log_2\frac{1}{n}$

信息熵——不确定性的度量

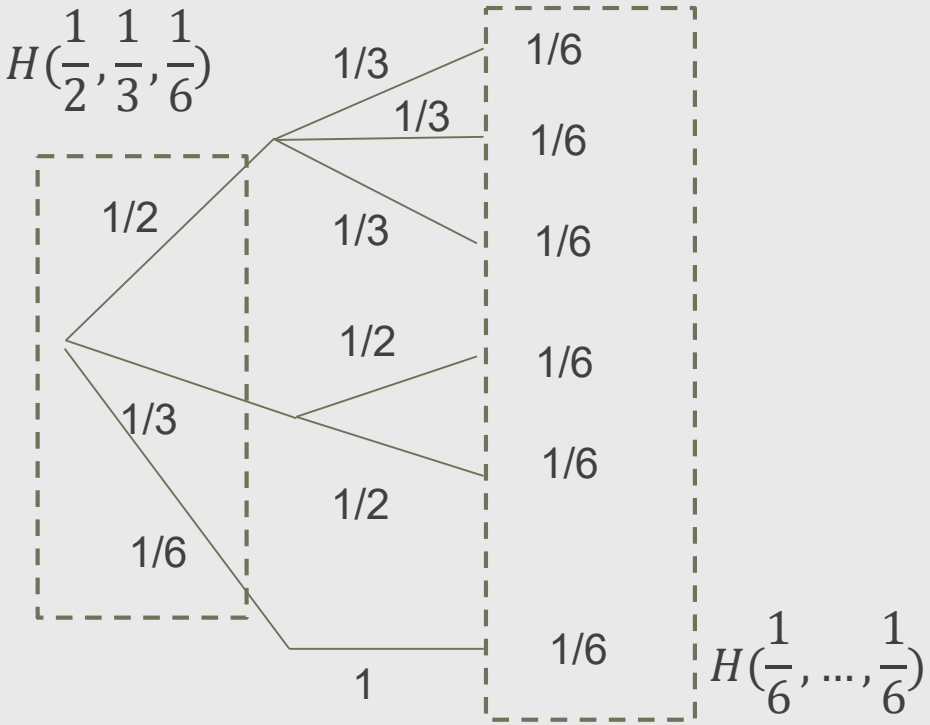
❑ 信息熵计算公式推导（补充阅读）

然后证明H公式对于所有和为1的一组正有理数都成立

根据右边例子，

$$H\left(\frac{1}{6},\dots,\frac{1}{6}\right) = H\left(\frac{1}{2},\frac{1}{3},\frac{1}{6}\right) + \frac{1}{2}H\left(\frac{1}{3},\frac{1}{3},\frac{1}{3}\right) + \frac{1}{3}H\left(\frac{1}{2},\frac{1}{2}\right) + \frac{1}{6}H(1)$$

- 对于$p_1, p_2, p_3 (p_1 + p_2 + p_3 = 1)$，总能找到$n_1, n_2, n_3$使$p_1 = \frac{n_1}{n_1+n_2+n_3}, p_2 = \frac{n_2}{n_1+n_2+n_3}, p_3 = \frac{n_3}{n_1+n_2+n_3}$，此例中$n_1 = 3, n_2 = 2, n_3 = 1$



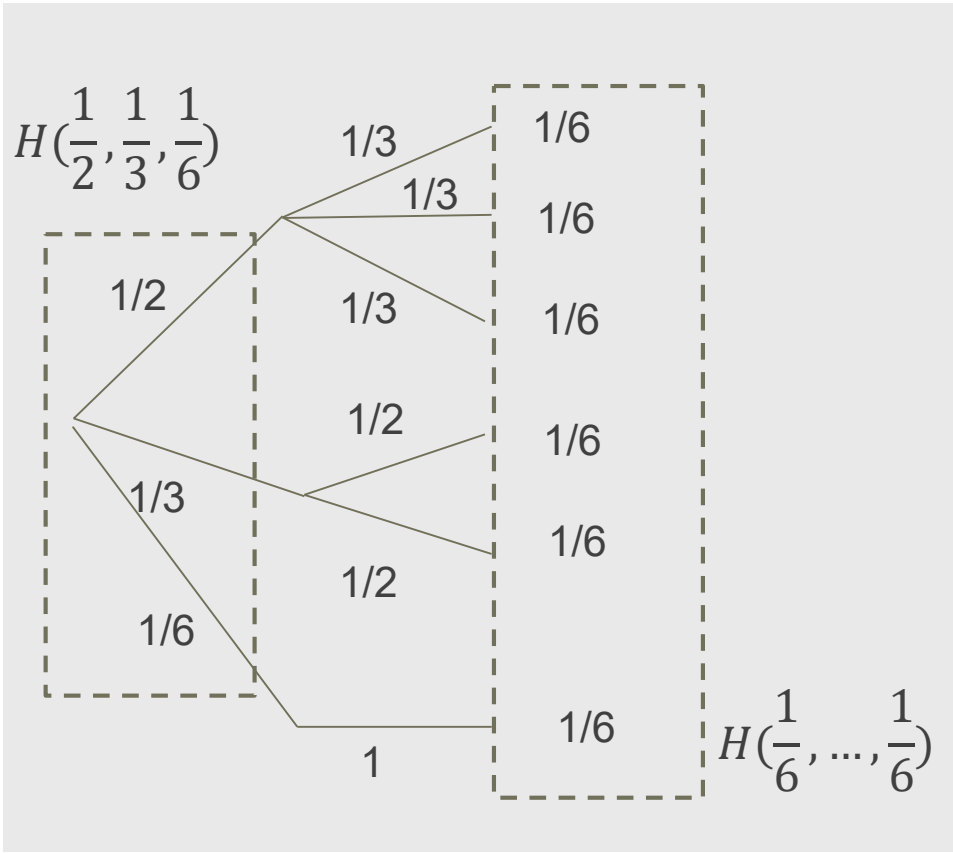
信息熵——不确定性的度量

❑ 信息熵计算公式推导（补充阅读）

然后证明H公式对于所有和为1的一组正有理数都成立

- 可以总结

$$H(p_1, p_2, p_3) = A(n_1 + n_2 + n_3) - [p_1 A(n_1) + p_2 A(n_2) + p_3 A(n_3)]$$



信息熵——不确定性的度量

❑ 信息熵计算公式推导（补充阅读）

然后证明H公式对于所有和为1的一组正有理数都成立

- 总结

$$H(p_1, p_2, p_3) = A(n_1 + n_2 + n_3) - [p_1 A(n_1) + p_2 A(n_2) + p_3 A(n_3)]$$

- 对于一般情形则有

$$H(p_1, p_2, \dots, p_k)$$

$$= A(n_1 + n_2 + \cdots + n_k) - [p_1 A(n_1) + p_2 A(n_2) + \cdots + p_k A(n_k)]$$

$$= (p_1 + p_2 + \cdots p_k)\log_2(n_1 + n_2 + \cdots + n_k) - [p_1 \log_2 n_1 + \cdots p_k \log_2 n_k]$$

$$= p_1 \log_2 \frac{n_1 + \cdots n_k}{n_1} + p_2 \log_2 \frac{n_1 + \cdots n_k}{n_2} + \cdots + p_k \log_2 \frac{n_1 + \cdots n_k}{n_k}$$

$$= -(p_1 \log_2 p_1 + \cdots p_k \log_2 p_k)$$

信息熵——不确定性的度量

- 信息熵计算公式推导（补充阅读）

熵公式对所有和为1的正有理数成立，由条件（4）（即，连续性条件）可以推出它对所有和为1的非负实数成立。

由此完成证明：

$$H(p_1, p_2, \dots p_n) = -\sum_i p_i log_2 p_i$$

信息熵——不确定性的度量

❑信息论与其他学科的关系

- 通信理论：数据压缩的临界值 & 数据传输的临界值
- 概率论：极限定理、大偏差理论
- 统计学：假设检验、费希尔信息
- 数学：不等式
- 经济学：投资组合理论、Kelly博弈论
- 计算机科学：科尔莫戈洛夫复杂度
- 物理学：AEP热力学、量子信息论

信息熵的概念及其计算方法是《信息论》的核心内容之一

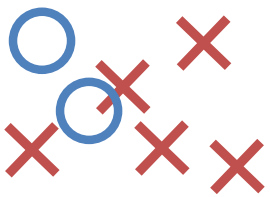
特征选择

❑ 信息熵（information entropy）

数据集$D = \{(\boldsymbol{x_1}, y_1), (\boldsymbol{x_2}, y_2), \dots, (\boldsymbol{x_m}, y_m)\}$包含$m$个样本。样本标签$y_i$共有$K$个可能取值（即概率论中的样本空间）。根据标签取值，要将所有样本分为K个类别，其中第k类样本所占的比例为$p_k (k = 1,2, \dots, K)$，则D的信息熵定义为

$$Ent(D) = -\sum_{k=1}^{K} p_k \log_2(p_k)$$

- 若$p_k = 0$，则令$p_k \log_2(p_k) = 0$

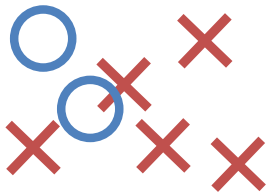


试一试

根据标签计算熵

特征选择

❑ 信息熵（information entropy）



试一试

根据标签计算熵

```
import math
print (-(5/7)*math.log((5/7),2)-(2/7)*math.log((2/7),2))
```

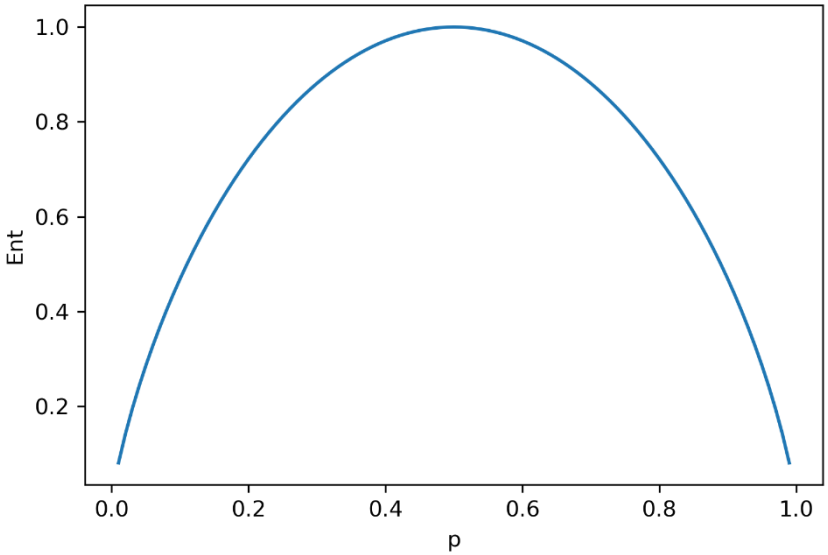
```
#0.863120568566631
```

特征选择

❑ 信息熵（information entropy）

试一试

编程绘制某二分类问题的熵$Ent = -p\log_2 p - (1-p)\log_2(1-p)$随$p$的变化趋势，并理解信息熵的特征



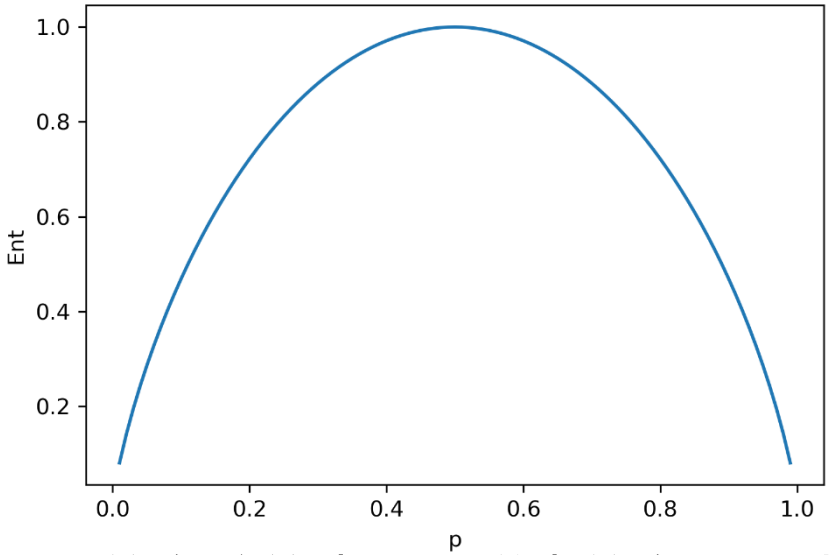
- 信息熵的本质是样本信息量（或自信息）$-\log_2(p_k)$的期望
- 在分类问题中，各类别比例相等时，信息熵最大，即混乱程度最大

特征选择

❑ 信息熵（information entropy）

试一试

编程绘制某二分类问题的熵$Ent = -p\log_2 p - (1-p)\log_2(1-p)$随$p$的变化趋势，并理解信息熵的特征



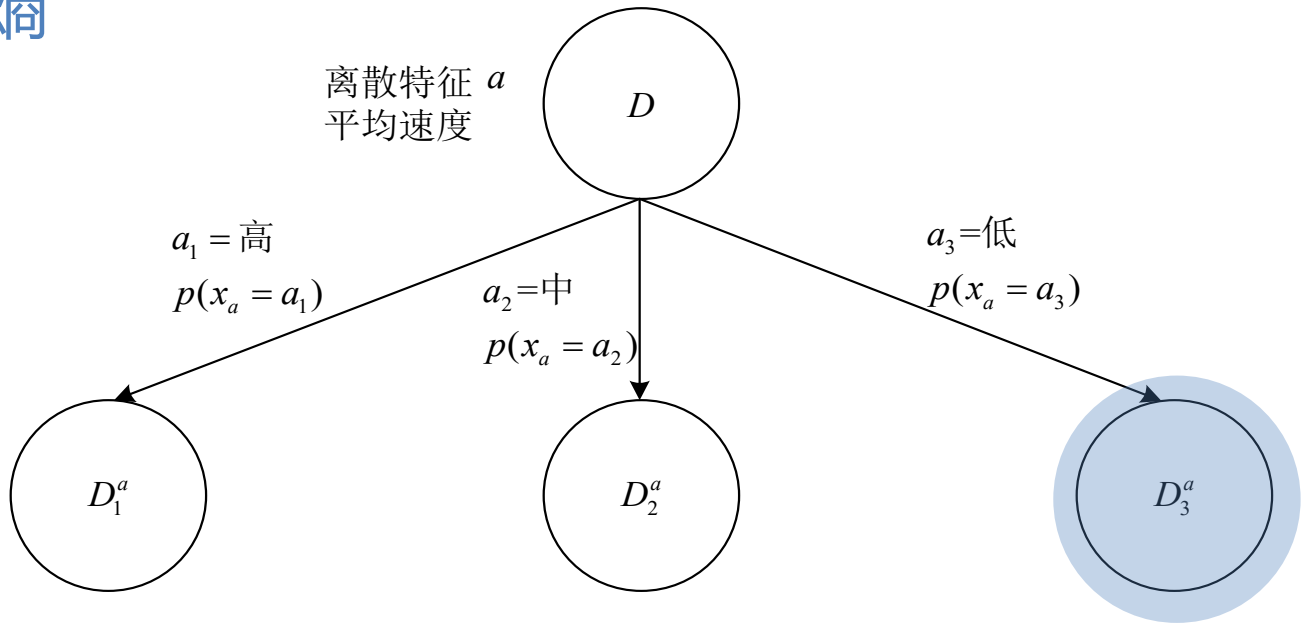

```
import numpy as np
import matplotlib.pyplot as plt
x = np.arange(0,1,0.01)
y = -x*np.log2(x)-(1-x)*np.log2(1-x)
plt.plot(x,y)
plt.xlabel("p")
plt.ylabel("Ent")
```

- 信息熵的本质是样本信息量（或自信息）$-\log_2(p_k)$的期望
- 在分类问题中，各类别比例相等时，信息熵最大，即混乱程度最大

特征选择

假设数据集D具有某个离散特征a，其可能的取值有N个，记为$\{a_1, a_2, \ldots, a_N\}$. D_j^a为D中样本在特征a上取值为a_j的样本子集，这些样本在整个样本集中所占的比例记为$p(x_a = a_j)$。

❑ 条件熵



D_j^a

试一试

计算集合D_3^a的信息熵$Ent(D_3^a)$

特征选择

❑ 条件熵

假设数据集D具有某个离散特征a，其可能的取值有N个，记为$\{a_1, a_2, \ldots, a_N\}$. D_j^a为D中样本在特征a上取值为a_j的样本子集，这些样本在整个样本集中所占的比例记为$p(x_a = a_j)$。

试一试

计算集合D_3^a的信息熵$Ent(D_3^a)$

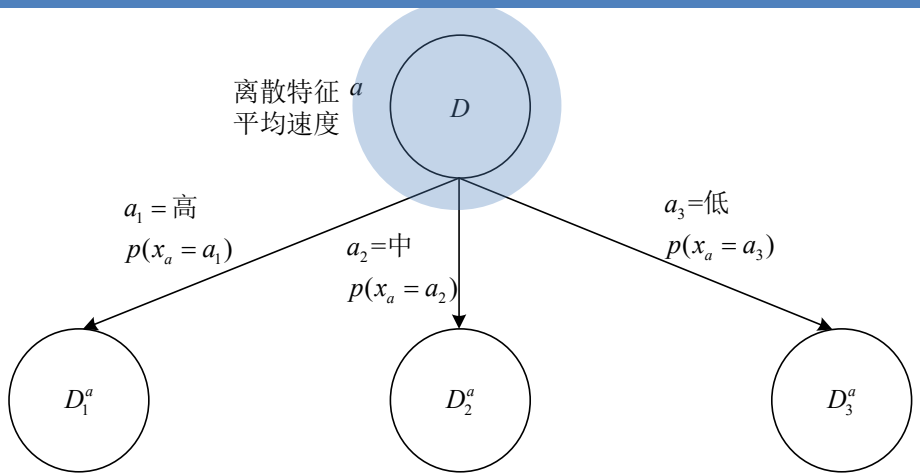
$$-\frac{5}{7}\log\frac{5}{7}-\frac{2}{7}\log\frac{2}{7}=0.863$$

```
import math
print (-(5/7)*math.log((5/7),2)-(2/7)*math.log((2/7),2))
```

```
#0.863120568566631
```

特征选择

❑ 条件熵



集合D_j^a的信息熵$Ent(D_j^a)$为

$-\sum_{k=1}^{K} p(y_k|x_a = a_j)\log_2 p(y_k|x_a = a_j).$

D_j^a

样本集D对于特征a的条件熵$Ent_a(D)$定义如下：

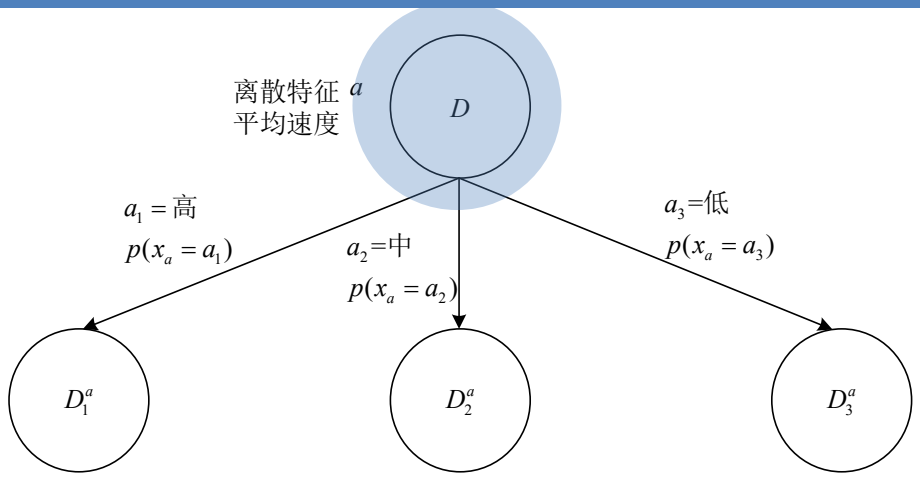
$$-\sum_{j=1}^{N} p(x_a = a_j)\sum_{k=1}^{K} p(y_k|x_a = a_j)\log_2 p(y_k|x_a = a_j) = \sum_{j=1}^{N}\frac{|D_j^a|}{|D|}Ent(D_j^a)$$

试一试

样本集D对于特征a的条件熵$Ent_a(D)$

特征选择

❑ 条件熵



试一试

样本集D对于特征a的条件熵$Ent_a(D)$

D_j^a

```
import math
print(
6/20*0
+7/20*(-2/7*math.log(2/7,2)-1/7*math.log(1/7,2)-4/7*math.log(4/7,2))
+7/20*(-5/7*math.log(5/7,2)-2/7*math.log(2/7,2))
)
```

```
#0.7846664217184822
```

特征选择

❑ 信息增益（information gain）

决策树算法在所有的备选特征中进行比选的时候，应该优先挑选那些**条件熵最小的特征（分类后，各分支的样本纯度高的特征）**。这一选择原则，也可以进一步使用如下的：

信息增益（information gain）：

$$Gain(D|a) = Ent(D) - Ent_a(D)$$

信息增益（information gain）表示“得知特征a的信息后，D的不确定性减少的程度”。在备选的特征中，信息增益大的特征则具有更强的分类能力（各分支的样本纯度最高），这是决策树学习选择特征的重要准则。

思考：能让分类结果“纯度更高”的“最优特征”，条件熵大or小？信息增益大or 小？

特征选择

❑ 信息增益（information gain）

试一试

根据以下数据集，以信息增益准则，选择用于路况分类的最优特征

编号	平均速度	流量	有无停车	交通状态	编号	平均速度	流量	有无停车	交通状态
1	低	小	有	拥堵	11	低	大	无	缓行
2	低	大	有	拥堵	12	中	大	无	畅通
3	高	大	无	畅通	13	低	小	有	拥堵
4	中	小	无	缓行	14	高	大	无	畅通
5	中	大	无	畅通	15	高	大	有	畅通
6	低	小	无	缓行	16	中	大	有	缓行
7	高	小	无	畅通	17	高	小	有	畅通
8	中	小	有	拥堵	18	高	小	无	畅通
9	低	小	有	拥堵	19	低	小	有	拥堵
10	中	小	无	缓行	20	中	小	无	缓行

首先手动计算一下特征“平均速度”对于数据集的信息增益

特征选择

❑ 信息增益（information gain）

试一试

根据以下数据集，以信息增益准则，选择用于路况分类的最优特征

分别用*sp*，*vo*，*st*表示平均速度、流量与有无停车3个特征

$$Gain(D|sp)=Ent(D)-\left[\frac{7}{20}Ent(D_1^{sp})+\frac{7}{20}Ent(D_2^{sp})+\frac{6}{20}Ent(D_3^{sp})\right]$$

$$=1.571-\left[\frac{7}{20}\left(-\frac{5}{7}\log_2\frac{5}{7}-\frac{2}{7}\log_2\frac{2}{7}\right)+\frac{7}{20}\left(-\frac{1}{7}\log_2\frac{1}{7}-\frac{4}{7}\log_2\frac{4}{7}-\frac{2}{7}\log_2\frac{2}{7}\right)+\frac{6}{20}\left(-\frac{6}{6}\log_2\frac{6}{6}\right)\right]$$

$$=1.571-0.785=0.786$$

类似地，可以计算其他特征对应的信息增益

$$Gain(D|vo)=Ent(D)-\left[\frac{12}{20}Ent(D_1^{vo})+\frac{8}{20}Ent(D_2^{vo})\right]=1.571-1.452=0.119$$

$$Gain(D|st)=Ent(D)-\left[\frac{11}{20}Ent(D_1^{st})+\frac{9}{20}Ent(D_2^{st})\right]=1.571-1.098=0.473$$

特征选择

❑ 信息增益（information gain）

输入：DataFrame，最后一列为标签

chooseBestFeatureToSplit(dataset)

```
baseEntropy = cal_Ent(dataset)  #计算当前数据集的信息熵
bestInfoGain = -1.0 #初始化最优信息增益和最优的特征
for name in dataset.columns[:-1]:
    uniquevals=set(dataset[name])# 获取当前特征的所有可能取值
    newEntropy = 0.0
    for value in uniquevals:# 计算各样本子集的信息熵
        subDataSet = splitDataSet(dataset, name, value)
        prob = len(subDataSet)/float(len(dataset))
        newEntropy += prob * cal_Ent(subDataSet)
    infoGain = baseEntropy - newEntropy #计算信息增益
    if (infoGain > bestInfoGain):      #比较每个特征的信息增益
        bestInfoGain = infoGain
        bestFeature = name
return bestFeature
```

特征选择

❑ 信息增益（information gain）

splitDataSet(data, name, value)

```
    """split the dataset
    Args:
        data: the original dataset
        name: the name of the feature for dataset splitting
        value: one of the values of the selected feature

    Returns:
        reducedFeatureVec: the subset of "data" whose "name" feature equa
ls to "value"
        (the "name" column should be dropped)
    """
    reducedFeatVec=data.loc[data[name]==value].drop(name,axis=1)
    return reducedFeatVec
```

特征选择

❑ 信息增益（information gain）

cal_Ent(data)

```
    """ calculate information entropy
    Args:
        data: a dataframe with header

    Returns:
        Ent: information entropy calculated based on sample labels ("
交通状态")
    """
    pass
```

特征选择

❑ 信息增益（information gain）

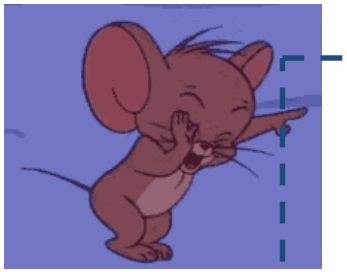
```
from math import log
import pandas as pd
def cal_Ent(data):
    sample_size = len(data)  # 样本数
    labelCounts = {}   #key是交通状态类别，value是属于该类别的样本个数
    for index,data in data.iterrows(): # 遍历整个数据集，每次取一行
        currentLabel = data['交通状态']  #取标签的值
        if currentLabel not in labelCounts.keys():
            labelCounts[currentLabel]=0
        labelCounts[currentLabel] += 1
    Ent = 0.0  # 初始化信息熵
    for key in labelCounts:
        prob = float(labelCounts[key])/sample_size
        Ent -= prob * log(prob,2) #计算信息熵
    return Ent
dataset=pd.read_csv('example8-1.csv',encoding='gbk')
print(cal_Ent(dataset))
```

```
#1.5709505944546684
```

特征选择

❑ 信息增益（information gain）

以信息增益作为特征选择依据的缺点：



试一试

如果我们把 “编号”列也作为一个特征，会有什么样的结果？

总结信息增益作为特征选择依据会带来什么问题？

对于这一问题，你是否有解决思路？

特征选择

❑ 信息增益率（information gain ratio）

信息增益倾向于选择那些取值较多的特征，容易造成过拟合。为了解决这个问题，引入**信息增益率**的概念：

$$GainRate(D|a) = \frac{Gain(D|a)}{IV(a)}$$

$$IV(a) = -\sum_{j=1}^{N} \frac{|D_j^a|}{|D|} \log_2 \frac{|D_j^a|}{|D|}$$

$IV(a)$称为特征a的固有值。可见，特征a的固有值实际上是以特征a的取值为标签来得到的一种信息熵。某个特征的取值越多，通常其固有值的取值也越大。

特征选择

❑ 信息增益率（information gain ratio）

试一试

根据以下数据集，计算各个特征的信息增益率

编号	平均速度	流量	有无停车	交通状态	编号	平均速度	流量	有无停车	交通状态
1	低	小	有	拥堵	11	低	大	无	缓行
2	低	大	有	拥堵	12	中	大	无	畅通
3	高	大	无	畅通	13	低	小	有	拥堵
4	中	小	无	缓行	14	高	大	无	畅通
5	中	大	无	畅通	15	高	大	有	畅通
6	低	小	无	缓行	16	中	大	有	缓行
7	高	小	无	畅通	17	高	小	有	畅通
8	中	小	有	拥堵	18	高	小	无	畅通
9	低	小	有	拥堵	19	低	小	有	拥堵
10	中	小	无	缓行	20	中	小	无	缓行

特征选择

❑ 信息增益率（information gain ratio）

EntGainRatio(dataset)

```
    """calculate information gain ratio
    Args:
        dataset: the original dataset

    Returns:
        GainRatio: a dictionary with the name of feature as key and t
he corresponding information gain ratio as value
    """
    InfoGain = EntGain(dataset)
    GainRatio = {}
    for name in dataset.columns[:-1]:
        GainRatio[name] = InfoGain[name] / cal_EntName(dataset, name)
    return GainRatio
```

特征选择

❑ 信息增益率（information gain ratio）

EntGain(dataset)

```
    """calculate information gain
    Args:
        dataset: the original dataset
    Returns:
        InfoGain: a dictionary with the name of feature as key and th
e corresponding information gain as value
    """
    pass
```

支持函数：

cal_Ent(dataset)

splitDataSet(dataset, name, value)

特征选择

❑ 信息增益率（information gain ratio）

cal_EntName(dataset, name)

```
""" calculate information entropy based on given feature
Args:
    data: a dataframe with header
    name: the feature name

Returns:
    Ent: information entropy (the intrinsic value, IV)
"""
pass
```

特征选择

❑ 信息增益率（information gain ratio）

EntGain(dataset)

```
baseEntropy = cal_Ent(dataset)  #计算当前数据集的信息熵
InfoGain = {} #记录不同特征下的信息增益
for name in dataset.columns[:-1]:
    uniquevals=set(dataset[name])# 获取当前特征的所有可能取值
    newEntropy = 0.0
    for value in uniquevals:#计算每种划分方式的信息熵
        subDataSet=splitDataSet(dataset,name,value)
        prob = len(subDataSet)/float(len(dataset))
        newEntropy += prob * cal_Ent(subDataSet)
    infoGainFeature = baseEntropy - newEntropy #计算信息增益
    InfoGain[name] = infoGainFeature # 记录信息增益
return InfoGain
```

特征选择

❑ 信息增益率（information gain ratio）

cal_EntName(dataset, name)

```
sample_size = len(dataset)  # 样本数
labelCounts = {}
for index,data in dataset.iterrows(): #遍历整个数据集，每次取一行
    currentLabel = data[name]
    if currentLabel not in labelCounts.keys():
        labelCounts[currentLabel]=0
    labelCounts[currentLabel] += 1
Ent = 0.0  # 初始化信息熵
for key in labelCounts:
    prob = float(labelCounts[key])/sample_size
    Ent -= prob * log(prob,2) #计算信息熵
return Ent
```

特征选择

❑ 基尼指数（Gini index）

对于数据集D，首先定义其“**基尼值**”：

$$\mathrm{Gini}(D) = \sum_{k=1}^{K} \sum_{k' \neq k} p_k p_{k'}$$

试一试

化简上式，消去$p_{k'}$；假如$K = 2$，写出基尼值的表达式；

并考虑基尼值越小，分类效果越好还是越差

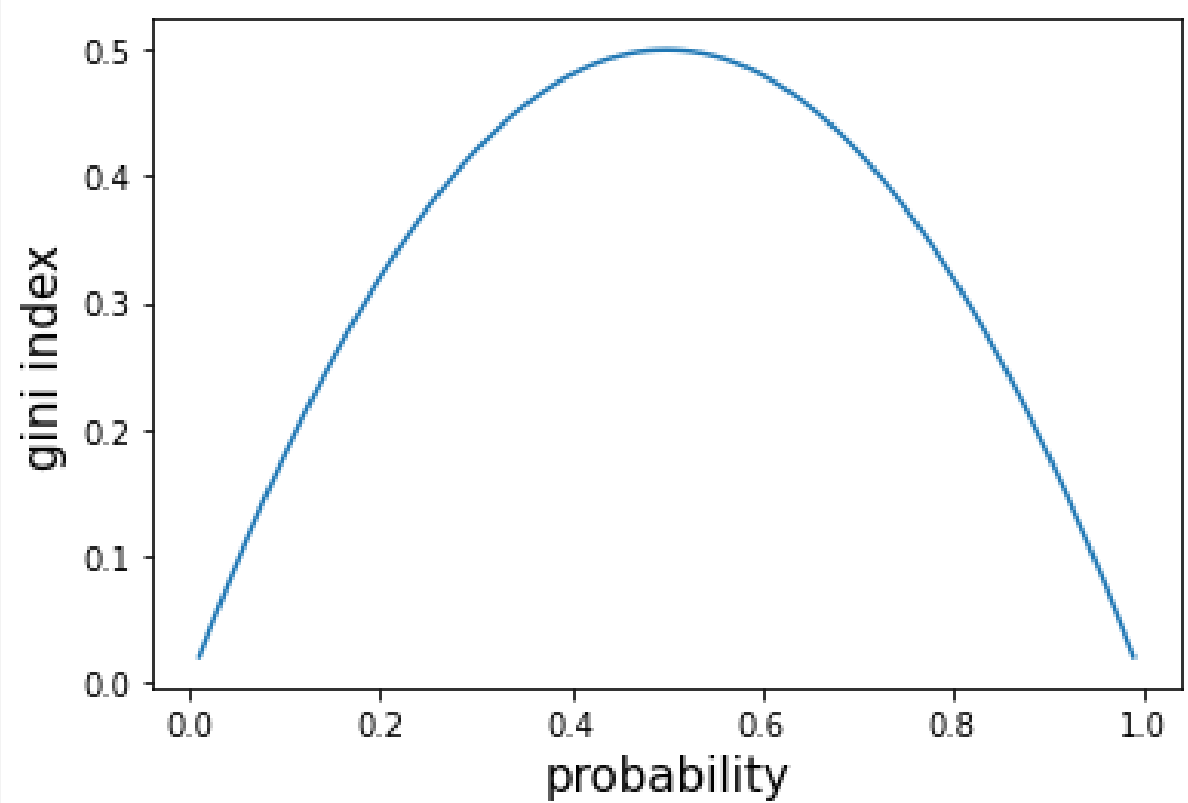
$$\sum_{k=1}^{K} \sum_{k' \neq k} p_k p_{k'} = \sum_{k=1}^{K} p_k \sum_{k' \neq k} p_{k'}$$

$$= \sum_{k=1}^{K} p_k (1 - p_k) = \sum_{k=1}^{K} p_k - p_k^2 = 1 - \sum_{k=1}^{K} p_k^2$$

特征选择

下图为二分类问题的基尼指数$2p(1-p)$随p的变化趋势

基尼值的物理意义就是从数据集D中随机抽取两个样本，他们**类别不一样**的概率，因此基尼指数越小表明数据集D的中同一类样本的数量越多， 其纯度越高。



特征选择

❑ 基尼指数（Gini index）

$$Gini(D) = \sum_{k=1}^{K} \sum_{k' \neq k} p_k p_{k'} = 1 - \sum_{k=1}^{K} p_k^2$$

对于特征a划分下的各个子集$D_j^a (j = 1,2,\dots,N)$，用$Gini(D_j^a)$表示其基尼值，则特征a的基尼指数定义为：

$$Gini\_index(D,a) = \sum_{j=1}^{N} \frac{|D_j^a|}{|D|} Gini(D_j^a)$$

基尼指数越小，则特征选择效果越好。

特征选择

❑ 基尼指数（Gini index）

试一试

根据以下数据集，计算各个特征的基尼指数

编号	平均速度	流量	有无停车	交通状态	编号	平均速度	流量	有无停车	交通状态
1	低	小	有	拥堵	11	低	大	无	缓行
2	低	大	有	拥堵	12	中	大	无	畅通
3	高	大	无	畅通	13	低	小	有	拥堵
4	中	小	无	缓行	14	高	大	无	畅通
5	中	大	无	畅通	15	高	大	有	畅通
6	低	小	无	缓行	16	中	大	有	缓行
7	高	小	无	畅通	17	高	小	有	畅通
8	中	小	有	拥堵	18	高	小	无	畅通
9	低	小	有	拥堵	19	低	小	有	拥堵
10	中	小	无	缓行	20	中	小	无	缓行

特征选择

❑ 基尼系数（Gini index）

GiniIndex(dataset)

```
    """calculate Gini index for all features
    Args:
        dataset: the original dataset
    Returns:
        Gini:
a dictionary with the name of feature as key and the corresponding Gi
ni index as value
    """
    pass
```

支持函数：

splitDataSet(dataset, name, value)

cal_Gini(subDataSet)

特征选择

- 基尼系数（Gini index）

cal_Gini(subDataSet)

```
"""calculate Gini index
Args:
    dataset: the original dataset
Returns:
    Gini: Gini index for dataset based on the label ("交通状态“)
"""
pass
```

特征选择

❑ 基尼系数（Gini index）

GiniIndex(dataset)

```
Gini = {}
for name in dataset.columns[:-1]:
    uniquevals=set(dataset[name])# 获取当前特征的所有可能取值
    GiniIndex = 0.0
    for value in uniquevals:#计算每种划分方式的信息熵
        subDataSet=splitDataSet(dataset,name,value)
        prob = len(subDataSet)/float(len(dataset))
        GiniIndex += prob * cal_Gini(subDataSet)
    Gini[name] = GiniIndex
return Gini
```

特征选择

❑ 基尼系数（Gini index）

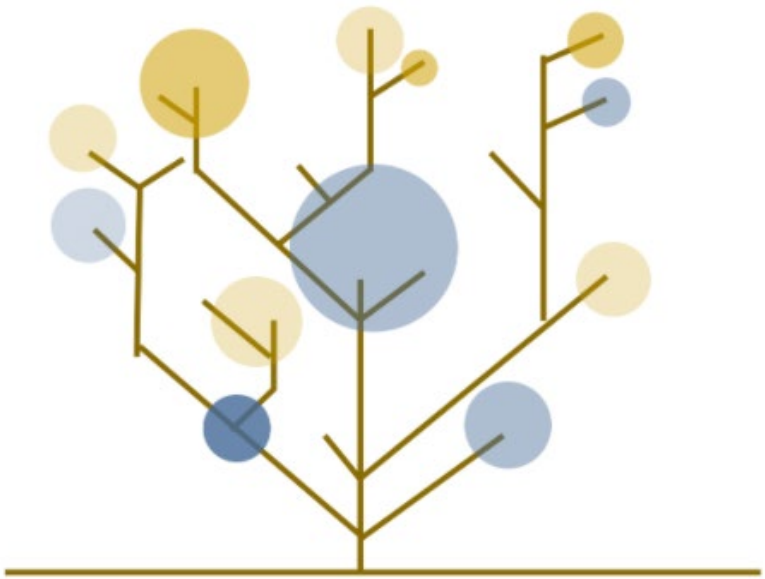
cal_Gini(subDataSet)

```
sample_size = len(dataset)  # 样本数
labelCounts = {}   #key是交通状态类别，value是属于该类别的样本个数
for index,data in dataset.iterrows(): # 遍历整个数据集，每次取一行
    currentLabel = data['交通状态']  #取标签的值
    if currentLabel not in labelCounts.keys():
        labelCounts[currentLabel]=0
    labelCounts[currentLabel] += 1
Gini = 1
for key in labelCounts:
    prob = float(labelCounts[key])/sample_size
    Gini -= prob**2
return Gini
```

决策树

❑ 学习目标

- 学习决策树特征选择的指标
  - 了解信息熵的由来和意义
  - 知道信息的测量方法
  - 掌握熵的表达式及其证明
- **知道经典的决策树算法**
- 了解决策树剪枝的原因、策略及其异同点
- 掌握应用决策树算法解决实际问题的方法
- 超参数优化方法



决策树生成

- 经典的决策树生成算法
  - ID3
  - C4.5
  - CART (classification and regression tree)算法

决策树生成

ID3算法

- 输入：训练数据集D，特征集A，阈值ε；
- 输出：决策树T。

1. 若D中所有样本属于同一类C_k，则T为单结点树，并将类C_k作为该结点的类标记，返回T；
2. 若$A = \emptyset$，则T为单结点树，并将D中样本数最多的类C_k作为该结点的类标记，返回T；
3. 否则，计算A中每个特征对D的信息增益，选出信息增益最大的特征a，并将该特征从集合A中删除；

决策树生成

ID3算法

- 输入：训练数据集D，特征集A，阈值ε；
- 输出：决策树T。

4. 如果a的信息增益小于阈值ε（例如0.1），则置T为单结点树，并将D中样本数量最多的类C_k作为该结点的类标记，返回T；
5. 否则，对a 的每一可能值a_i，依据其样本特征值将D分割为若干非空子集D_i，将D_i中样本数量最多的类作为标记，构建子结点，由结点及其子结点构成树T，返回T；
6. 对第i个子结点，以D_i为训练集，以更新后的A为特征集，递归地执行步骤（1）~（5），得到子树T_i，返回T_i。

决策树生成

ID3算法

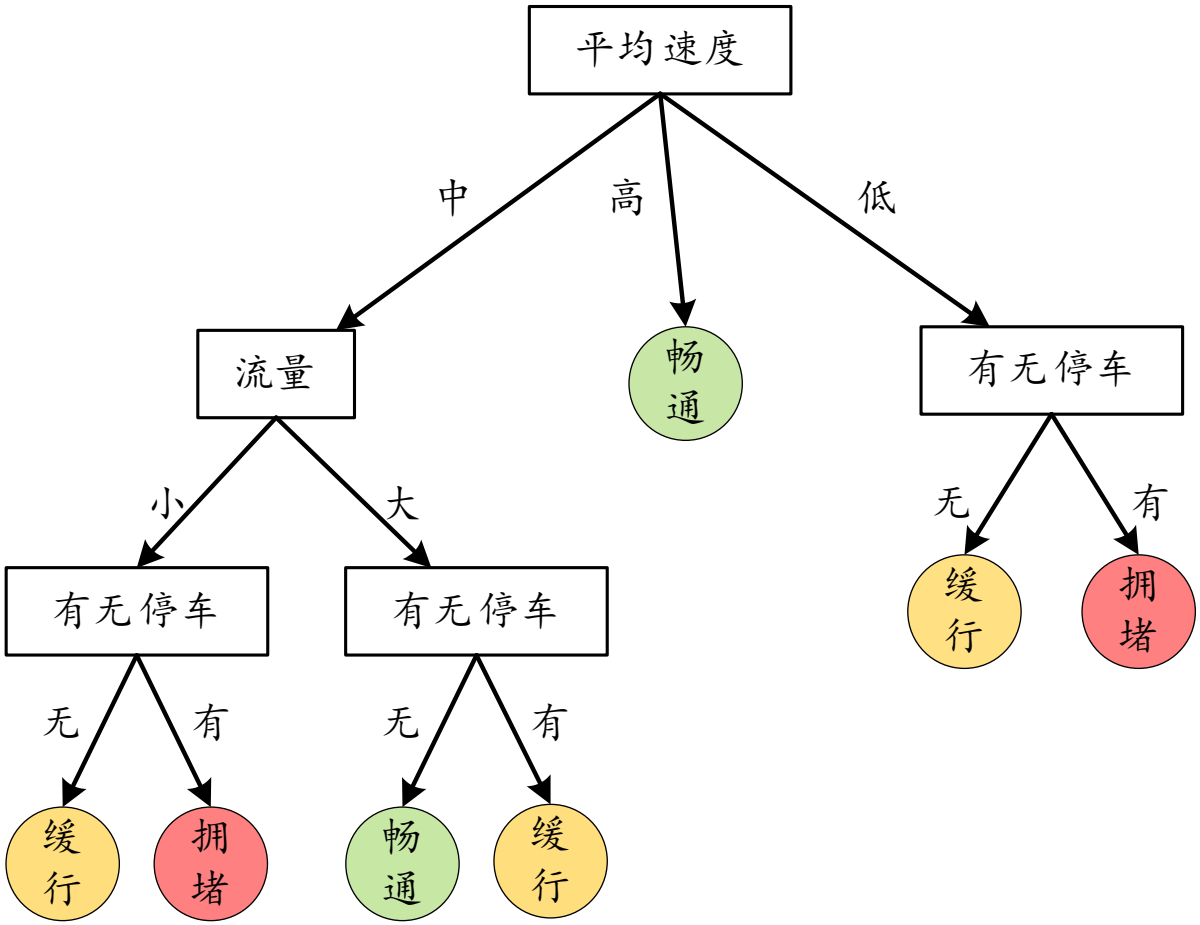
试一试

根据以下训练集，基于信息增益规则和ID3算法流程步骤，通过代码递归生成一棵决策树，以字典的方式对结果进行存储。

编号	平均速度	流量	有无停车	交通状态	编号	平均速度	流量	有无停车	交通状态
1	低	小	有	拥堵	**11**	低	大	无	缓行
2	低	大	有	拥堵	**12**	中	大	无	畅通
3	高	大	无	畅通	**13**	低	小	有	拥堵
4	中	小	无	缓行	**14**	高	大	无	畅通
5	中	大	无	畅通	**15**	高	大	有	畅通
6	低	小	无	缓行	**16**	中	大	有	缓行
7	高	小	无	畅通	**17**	高	小	有	畅通
8	中	小	有	拥堵	**18**	高	小	无	畅通
9	低	小	有	拥堵	**19**	低	小	有	拥堵
10	中	小	无	缓行	**20**	中	小	无	缓行

决策树生成

- ID3算法



决策树生成

ID3算法

createTree(dataSet,featureName)

```
    classList = dataSet['交通状态'].tolist()
    # 停止分类条件：
    ### 当类别完全相同则停止继续划分
    ### 当只剩一个特征的时候，无法继续划分，返回数量最多的类别：
return majorityCnt(classList)
    bestFeatLabel= chooseBestFeatureToSplit(dataSet)#最佳特征
    myTree ={bestFeatLabel:{}}  # 字典结构，且key为featureLabel
    featureName.remove(bestFeatLabel)
    featValues = dataSet[bestFeatLabel] # 找到需要分类的特征子集
    uniqueVals = set(featValues)
    for value in uniqueVals:
        subLabels = featureName[:] # 复制操作
        ### 递归，给myTree[bestFeatLabel]赋值
    return myTree
```

chooseBestFeatureToSplit(dataset)

datasetmajorityCnt(classList)

决策树生成

ID3算法

majorityCnt(classList)

```
    """return the class with the majority of samples
    Args:
        classList: the list of sample labels
    Returns:
        sortedClassCount[0][0]: the class with the majority of samples
    note that:
  classCount records different labels in classList and their occurrences
  sortedClassCount is a dictionary sorted by value in descending order b
ased on classCount
    """
    classCount={}
    for vote in classList:
        if vote not in classCount.keys(): classCount[vote] = 0
        classCount[vote] += 1
    sortedClassCount = sorted(classCount.items(), key=itemgetter(1), rev
erse=True)
    return sortedClassCount[0][0]
```

决策树生成

ID3算法

createTree(dataSet,featureName)

```
    classList = dataSet['交通状态'].tolist()
    if classList.count(classList[0]) == len(classList): # 统计属于类别
classList[0]的个数
        return classList[0] # 当类别完全相同则停止继续划分
    if len(dataSet.iloc[0]) ==1: # 当只有一个特征的时候
        return majorityCnt(classList)
    bestFeatLabel= chooseBestFeatureToSplit(dataSet)#最佳特征
    myTree ={bestFeatLabel:{}}  # 字典 结构，且key为featureLabel
    featureName.remove(bestFeatLabel)
    featValues =dataSet[bestFeatLabel] # 找到需要分类的特征子集
    uniqueVals = set(featValues)
    for value in uniqueVals:
        subLabels = featureName[:] # 复制操作
        myTree[bestFeatLabel][value] = createTree(splitDataSet(dataSet,b
estFeatLabel,value),subLabels)
    return myTree
```

决策树生成

❑ C4.5算法

- ID3算法的缺点：
  1. 采用信息增益来筛选特征，容易倾向于选择那些取值较多的特征
  2. 只能处理离散变量
- C4.5算法：
  1. 采用信息增益率筛选特征
  2. 在现实任务中经常会遇到连续型变量（特征），比如交通流参数中的速度、流量等，为了在决策树中对连续型变量进行分支化归类，需要对其进行离散化处理。该算法采用的是：**二分法（bi-partition）**

决策树生成

❑ C4.5算法

- 连续型变量的离散化处理：二分法（bi-partition）

1. 对于某个连续特征a，将所有数据按a的取值做升序排列，得到一个序列$\{a_1, a_2, \dots, a_N\}$。
2. 对于任意$1 \le i \le N-1$，都有两个相邻的属性取值为a_i和a_{i+1}，定义$t_i = \frac{a_i + a_{i+1}}{2}$，则可将数据集分为两个子集。
3. 计算采用每个t_i划分后集合D的信息增益率，并选择信息增益率最大的t_i^*作为最优划分点。

决策树生成

C4.5算法

试一试

根据以下训练集，基于二分法，写一个函数找到两个连续特征各自的最佳划分点

ID	平均速度	流量	有无停车	交通状态	ID	平均速度	流量	有无停车	交通状态
1	6.25	5	有	拥堵	11	10.56	10	无	缓行
2	5.70	8	有	拥堵	12	11.47	12	无	畅通
3	12.15	2	无	畅通	13	3.98	7	有	拥堵
4	6.61	3	无	缓行	14	14.71	21	无	畅通
5	16.02	16	无	畅通	15	16.02	21	有	畅通
6	5.52	3	无	缓行	16	11.11	5	有	缓行
7	15.49	21	有	畅通	17	15.37	4	有	畅通
8	9.48	15	有	拥堵	18	12.03	15	有	畅通
9	7.16	21	有	拥堵	19	6.19	6	有	拥堵
10	6.85	9	有	缓行	20	8.68	6	无	缓行

决策树生成

C4.5算法

试一试

根据以下训练集，基于二分法，写一个函数找到两个连续特征各自的最佳划分点

平均速度特征划分点选择结果

划分点	信息增益	划分点	信息增益	划分点	信息增益
4.75	0.091	7.01	0.359	11.75	0.717
5.61	0.079	7.92	0.476	12.09	0.557
5.95	0.139	9.08	0.522	13.43	0.430
6.22	0.227	10.02	0.725	15.04	0.322
6.43	0.341	10.84	0.798	15.43	0.228
6.73	0.330	11.29	0.971	15.75	0.144

平均速度的最优划分点即为11.29

决策树生成

C4.5算法

试一试

根据以下训练集，基于二分法，写一个函数找到两个连续特征各自的最佳划分点

流量特征划分点选择结果

划分点	信息增益	划分点	信息增益	划分点	信息增益
2.5	0.069	6.5	0.095	11	0.371
3.5	0.117	7.5	0.093	13.5	0.277
4.5	0.122	8.5	0.125	15.5	0.216
5.5	0.062	9.5	0.198	18.5	0.145

流量的最优划分点即为11

决策树生成

C4.5算法

chooseBestFeatureToSplit_c(dataSet)

```
baseEntropy = cal_Ent(dataSet)  # 计算根结点的信息熵
bestInfoGain = {}
bestPartValue = {} # 连续的特征值，最佳划分值
for name in dataSet.columns[:-2]:#取平均速度和流量两个特征
    bestInfoGain[name] = 0
    bestPartValue[name] = None
    uniqueVals=set(dataSet[name])# 获取当前特征的所有可能取值
    sortedUniqueVals = list(uniqueVals)
    sortedUniqueVals.sort()  # 对特征值排序
    # 未完
```

splitDataSet_c(dataSet, name, partValue)

cal_Ent(dataSet)

决策树生成

C4.5算法

chooseBestFeatureToSplit_c(dataSet)

```
for name in dataset.columns[:-2]:#取平均速度和流量两个特征
    ## 省略 ##
    for j in range(len(sortedUniqueVals) - 1):
        ### 计算划分点partValue，借助splitDataSet_c和cal_Ent计算对应的信息熵
        infoGain = baseEntropy - Entropy    # 计算信息增益
        if infoGain > bestInfoGain[name]:
            bestInfoGain[name] = infoGain
            bestPartValue[name] = partValue
return bestInfoGain, bestPartValue
```

决策树生成

C4.5算法

chooseBestFeatureToSplit_c(dataSet)

```
for name in dataset.columns[:-2]:#取平均速度和流量两个特征
    ## 省略 ##
    for j in range(len(sortedUniqueVals) - 1):
        partValue = (float(sortedUniqueVals[j]) + float(sortedUniqueVals
[j + 1]))/2 #计算划分点
        dataSetLeft,dataSetRight,probLeft,probRight = splitDataSet_c(dat
aSet, name, partValue)
        Entropy = probLeft * cal_Ent(dataSetLeft) + probRight * cal_Ent(
dataSetRight) #计算信息熵
        infoGain = baseEntropy - Entropy    # 计算信息增益
        if infoGain > bestInfoGain[name]:
            bestInfoGain[name] = infoGain
            bestPartValue[name] = partValue
return bestInfoGain, bestPartValue
```

决策树生成

❑CART(classification and regression tree)算法

- 可将连续值离散化（其处理思想与C4.5相同，即将连续特征值离散化）
- 采用基尼系数作为特征选择标准
- 采用**二叉树**（即每个结点只有两个分叉）
  - ✓ ID3和C4.5：对于取值数量>2的离散特征，在结点上划分出多叉树，在分叉之后的“子树”中该特征不会再次起作用。
  - ✓ CART：采用二叉树，使决策树的深度增加，且每步重复使用所有特征；使决策树的深度增加，提升树的结构复杂度，且每步重复使用所有特征。
- 停止条件：结点中的样本个数小于预定阈值，或样本集的基尼指数小于预定阈值。

❑CART(classification and regression tree)算法

- CART分类树的分枝规则：二叉树的构建方法
  - ✓ 对于有多个取值的特征，CART树会**选择其中一个特征取值分为一类，其它特征取值汇总为另外一类。**例如，特征a的取值包括$\{a_1, a_2, \dots, a_N\}$，CART树首先将a_1作为一类，而把其它取值全部作为另外一类。
  - ✓ 之后，计算该分类下数据集D的基尼指数，记为$Gini_index\ (D, a_1)$。
  - ✓ 以此遍历所有N个取值，进行这样的二分类处理，并逐次计算其对应的基尼指数。
  - ✓ 最终选择基尼指数最小的一种划分方法，其取值记为a_j^*，来作为该特征的划分结果，即特征a的基尼指数值$Gini_index\ (D, a) = Gini_index\ (D, a_j^*)$。

❑CART(classification and regression tree)算法

- CART分类树的算法流程：

对于训练数据集D，从根结点开始，递归地对每个结点进行如下操作：

① 用$\bar{D}$来表示是目前结点待分析的数据集，供选择的特征构成集合$\bar{A}$。对任一个特征$a \in \bar{A}$，根据分支规则，计算基尼指数$Gini_{index}(D,a)$。选择$\bar{A}$中基尼指数最小的特征a^*作为最优特征，并以该特征的最优划分方法来建立决策树分支（二叉树），从现结点生成两个子结点，将训练数据集依特征分配到子结点中去，得到子集$\bar{D}_1$和$\bar{D}_2$，并将对应子结点的标签分别标记为$\bar{D}_1$和$\bar{D}_2$中样本数最多的类别。

② 判断目前的决策树是否满足停止条件，例如结点中的样本个数小于预定阈值，或样本集的基尼指数$Gini\ (\bar{D})$小于预定阈值（样本基本属于同一类）。如果不满足，则对新生成的子结点，递归地执行步骤①中的操作，要注意的是，此时通常会保留$\bar{A}$中的所有特征到子树中，即特征a^*在子树中可能会重复出现；若满足停止条件则输出生成的CART决策树。

决策树生成

❑ CART(classification and regression tree)算法

试一试

根据以下训练集，调用Scikit-learn中的DecisionTree模块，以平均速度和流量作为特征来生成一棵决策树

ID	平均速度	流量	有无停车	交通状态	ID	平均速度	流量	有无停车	交通状态
1	6.25	5	有	拥堵	11	10.56	10	无	缓行
2	5.70	8	有	拥堵	12	11.47	12	无	畅通
3	12.15	2	无	畅通	13	3.98	7	有	拥堵
4	6.61	3	无	缓行	14	14.71	21	无	畅通
5	16.02	16	无	畅通	15	16.02	21	有	畅通
6	5.52	3	无	缓行	16	11.11	5	有	缓行
7	15.49	21	有	畅通	17	15.37	4	有	畅通
8	9.48	15	有	拥堵	18	12.03	15	有	畅通
9	7.16	21	有	拥堵	19	6.19	6	有	拥堵
10	6.85	9	有	缓行	20	8.68	6	无	缓行

决策树生成

- CART(classification and regression tree)算法

提示：

1. **pd.Categorical(data[‘交通状态’]).codes** returns the category codes of the categorical variable
2. **DecisionTreeClassifier(criterion='gini')** – build the model;
3. **model.fit(x, y)** – fit the model
4. Graphviz是一个开源的图形可视化软件，用于表达有向图或无向图的连接关系，也是在Python中进行决策树模型建立时一个非常有力的可视化工具，可以直观地展示决策树结构

决策树生成

- CART(classification and regression tree)算法

```
import os
os.environ[“PATH”]+=os.pathsep+'D:/F/Graphviz/bin/'#修改为安装的路径
import numpy as np
import pandas as pd
from sklearn import tree
from sklearn.tree import DecisionTreeClassifier  #导入决策树模块
import pydotplus
data = pd.read_csv('example8-2.csv',encoding='gbk')
x1= data['平均速度']
x2= data['流量']
x=np.stack((x1,x2),axis=1)
y=pd.Categorical(data['交通状态']).codes
model = DecisionTreeClassifier(criterion='gini')     #决策树模型建立
model.fit(x, y)
```

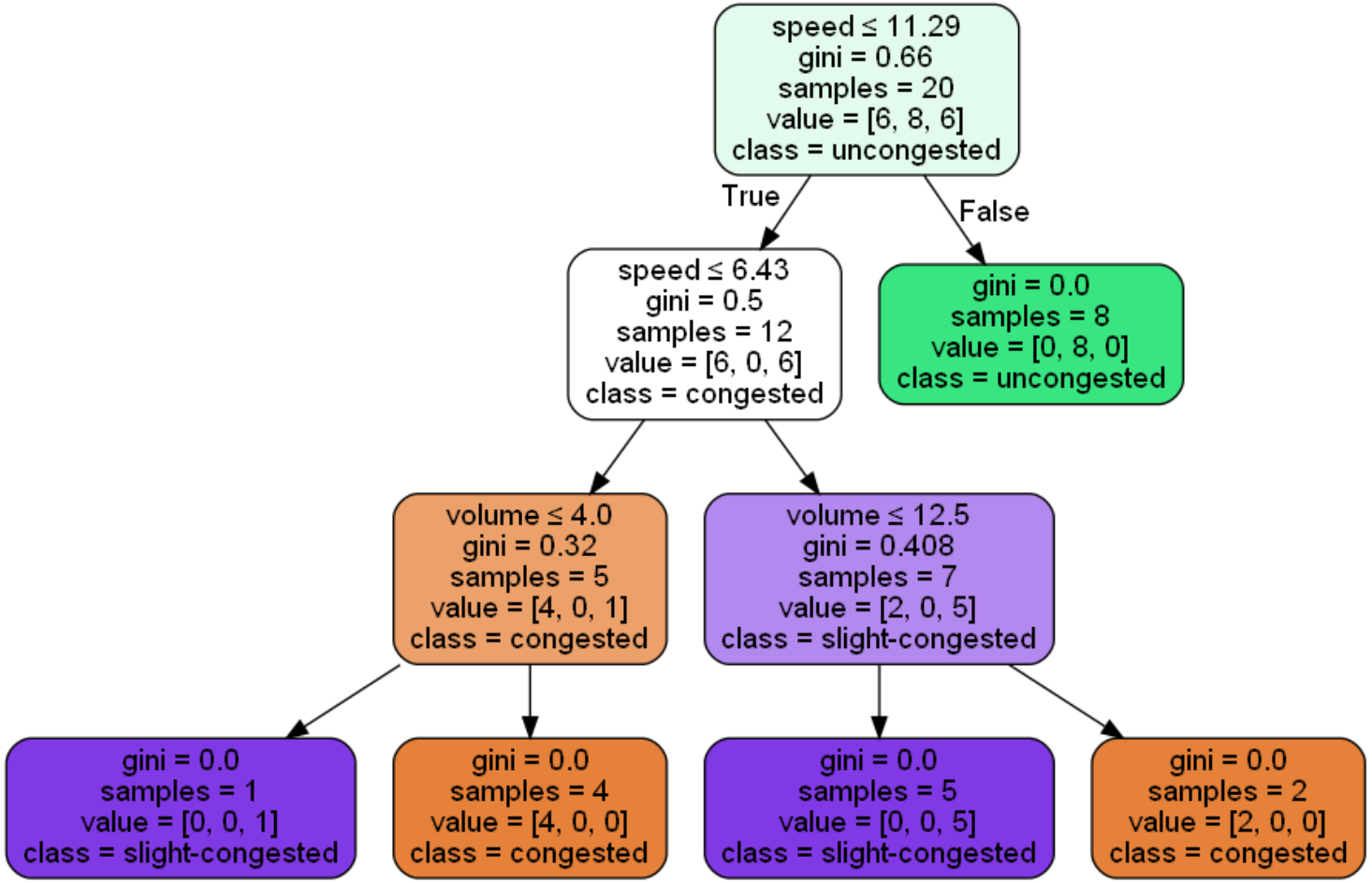
决策树生成

- CART(classification and regression tree)算法

```
traffic_feature_E = 'speed', 'volume'
label = 'congested', 'uncongested','slight-congested'
with open('traffic_condition.dot','w', encoding='utf-8') as f:
    tree.export_graphviz(model, out_file=f)
    dot_data = tree.export_graphviz(model, out_file=None,
                                    feature_names=traffic_feature_E,
                                    class_names=label,
                                    filled=True, rounded=True,
                                    #颜色表示结点纯度
                                    special_characters=True)
    graph = pydotplus.graph_from_dot_data(dot_data)
    f = open('traffic_condition_decision_tree.png', 'wb')
    f.write(graph.create_png())  #输出决策树结构图
    f.close()
```

决策树生成

- CART(classification and regression tree)算法



决策树生成

❑CART(classification and regression tree)算法

CART回归树

- CART回归树的特征可以是离散值或连续值，它沿用了CART分类树中对连续特征的处理方法和二叉树建树规则，但不同之处在于回归树的输出变量（样本标签y）是连续值，比如预测一条道路的交通量（在车流量比较大的时候，交通量可近似认为是一种连续值）。因此，回归树在选择划分特征的度量标准和决策树建立后预测的方式上，也和分类树存在不同。
- 数据集$D = \{(\boldsymbol{x_1}, y_1), (\boldsymbol{x_2}, y_2), \dots, (\boldsymbol{x_m}, y_m)\}$包含$m$个样本。**$\boldsymbol{y_i}$是连续变量**，回归问题的目标是构造一个函数$f(\boldsymbol{x})$来拟合数据，使得误差最小。以均方误差为损失函数时：

$$\min \frac{1}{m} \sum_{i=1}^{m} (f(\boldsymbol{x_i}) - y_i)^2$$

决策树生成

❑CART(classification and regression tree)算法

CART回归树

- CART分类树的每一个叶结点的类别标签是由该结点中样本量最大的类所决定。而回归树同样需要确定这样的“标记”来作为叶结点的预测值。一棵有N个叶节点的决策树，将输入空间χ划分成N个子集$\chi_1, \chi_2, \ldots, \chi_N$。对于叶节点$n$，它的**预测值（标签）$c_n$即为该节点所有样本的平均值**，则CART回归树的**均方误差损失函数**为：

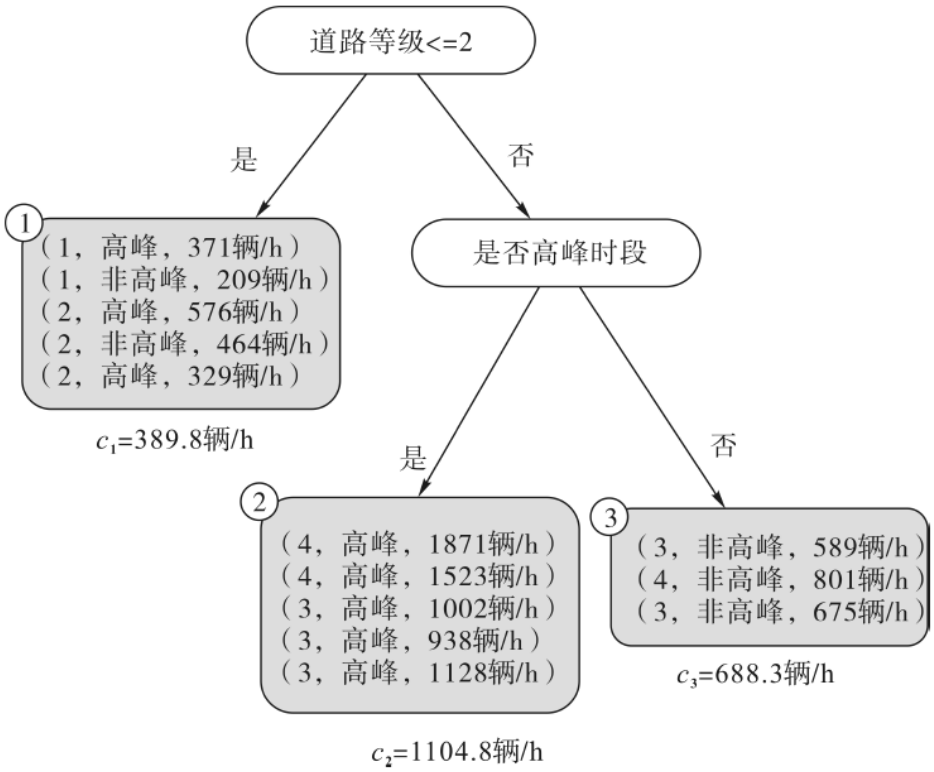
$$\min \frac{1}{m} \sum_{n=1}^{N} \sum_{x_i \in \chi_n} (c_n - y_i)^2$$

决策树生成

❑CART(classification and regression tree)算法

CART回归树

- 以道路交通量预测问题为例，建立包含两个特征（道路等级rl和观测时段op）的回归树模型，进行交通量y的预测。现有13个样本构成的数据集，生成的回归树结构如图。可见，标签y为连续值，那么节点1的标签为：



$$c_1 = (371 + 209 + 576 + 464 + 329)/5 = 389.8辆/h$$

决策树生成

❑CART(classification and regression tree)算法

CART回归树

- 在确定了c_n值后，即可计算各叶结点的均方误差，通过选择最优划分特征和划分点，使得各个叶结点的均方误差和最小。CART回归树与分类树的算法步骤基本一致，每步都需要遍历所有输入特征a和其划分点a_j进行二分叉，将输入空间划分为$\overline{D}_1{}^a(a_j)$和$\overline{D}_2{}^a(a_j)$。
- 对于连续变量来说，划分结果是$\overline{D}_1{}^a(x_a > a_j)$和$\overline{D}_2{}^a(x_a \le a_j)$的形式；对于离散变量，划分结果是$\overline{D}_1{}^a(x_a = a_j)$和$\overline{D}_2{}^a(x_a \ne a_j)$的形式。

决策树生成

❑CART(classification and regression tree)算法

CART回归树

- 为了使整个决策树的均方误差最小，此处的划分点a_j的选择也应该以均方误差为目标（替代CART分类树中的基尼指数），使得

$$\min_{a,a_j}\left[\sum_{x_i\in\bar{X}_1(a,a_j)}(y_i-c_1)^2+\sum_{x_i\in\bar{X}_2(a,a_j)}(y_i-c_2)^2\right]$$

决策树生成

- CART(classification and regression tree)算法

试一试

根据以下训练集，调用Scikit-learn中的DecisionTreeRegressor模块，以估计交通量

ID	道路等级	是否高峰	交通量	ID	道路等级	是否高峰	交通量
1	1	是	371	**8**	3	是	1002
2	1	否	209	**9**	3	是	938
3	2	是	576	**10**	3	是	1128
4	2	否	464	**11**	3	否	589
5	2	是	329	**12**	4	否	801
6	4	是	1871	**13**	3	否	675
7	4	是	1532				

决策树生成

- CART(classification and regression tree)算法

```
from sklearn.tree import DecisionTreeRegressor
from sklearn.metrics import mean_squared_error
data = pd.read_csv('example8-3.csv',encoding='gbk')
model = DecisionTreeRegressor(
                        criterion='squared_error')  #决策树模型建立
x1= data['道路等级']
x2= pd.Categorical(data['是否高峰']).codes
x=np.stack((x1,x2),axis=1)
model.fit(x, data[['交通量']])
pred = model.predict(x)
print (mean_squared_error(data[['交通量']], pred))
#8488.74358974359
```

决策树

❑ 学习目标

- 学习决策树特征选择的指标
  - 了解信息熵的由来和意义
  - 知道信息的测量方法
  - 掌握熵的表达式及其证明
- 知道经典的决策树算法
- **了解决策树剪枝的原因、策略及其异同点**
- 掌握应用决策树算法解决实际问题的方法

决策树剪枝

❑问题概述

- 原因：减少过拟合，提升泛化能力

在决策树学习中，为了尽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“过好”**，以至于把训练集中的部分噪声当作所有数据都具有的一般性质而导致**过拟合**。

因此，在决策树构建过程中，可通过主动去掉一些分支的方法来降低过拟合的风险，即**需要对已生成的决策方案进行简化**，这种过程称为**剪枝（pruning）**。

决策树剪枝

❑问题概述

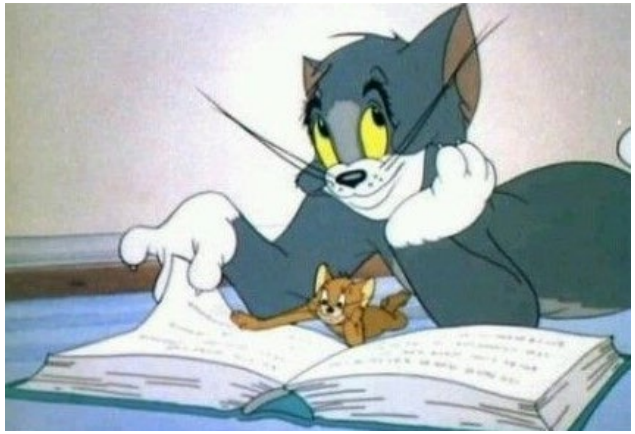
- 方法：预剪枝（pre-pruning）和后剪枝（post-pruning）

预剪枝是通过设定一些控制条件终止树的生长；

后剪枝则在使树充分生长后，通过一定条件将树进行收缩以达到剪枝的目的。

试一试

你会怎样设计预剪枝和后剪枝策略？



决策树剪枝

❑预剪枝的几种常用方法

1. 控制决策树的最大**深度**（即根结点到叶结点的最大结点数，但不包含根结点）。如果决策树的层数已经达到指定深度，则停止生长；
2. 控制树中**父结点**的最少样本量或比例。如果结点的样本量低于最小样本量或比例，则不再对该结点进一步划分；
3. 控制树中**叶结点**的最小样本量或比例。如果结点划分后生成的任意一个子结点的样本量低于最小样本量或比例，则不对该结点进行划分。

决策树剪枝

https://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeClassifier.html

❑预剪枝

优点

- 通过阻止分支“展开”的方法，降低了过拟合的风险
- 提高了训练效率

缺点

- 预剪枝方法存在欠拟合的风险

试一试

Scikit-learn决策树算法库中的DecisionTreeClassifier，就运用了预剪枝的思想，可以通过设置一些重要参数来防止过拟合。请自行查阅官方文档进行学习。

决策树剪枝

❑后剪枝

1. 从所有样本中选择一部分作为验证集，计算决策树T在验证集上的验证误差$\mathcal{L}(T)$
2. 考察决策树T的内部节点i，是否要在此进行剪枝。将在结点i处剪枝得到的决策树记为T_i，计算其验证误差$\mathcal{L}(T_i)$，若$\mathcal{L}(T) > \mathcal{L}(T_i)$，则剪枝，更新$T = T_i$.
3. 自下而上遍历决策树T的所有内部结点，执行步骤（2）；直至分析完所有的内部结点后，输出此时的决策树T

决策树剪枝

❑后剪枝

- 上述方法是后剪枝方法中最简单的“**错误率降低剪枝法**”（reduced-error pruning）
- 由于其使用了独立的验证集，其缺点在于：
  1. 修改后的决策树可能偏向于过度修剪；
  2. 在数据总量有限的时候，设置验证集浪费了宝贵的数据资源。
- 为了克服这些问题，产生了另外两种剪枝方法，**悲观错误剪枝法**（pessimistic error pruning，PEP）和**代价复杂度剪枝法**（cost-complexity pruning，CCP），分别在C4.5算法和CART算法中被采用了。

C4.5算法还用到了error-based pruning (EBP)，感兴趣的同学了解一下

决策树剪枝

❑悲观错误剪枝法（PEP）

- PEP自上而下地估算每个内部结点t所覆盖的样本集的错误率P_t，通过比较剪枝前后这部分样本的错误率来决定是否进行剪枝。
- 特点：
  1. 完全使用训练数据来生成决策树，又用这些训练数据来完成剪枝。
  2. 决策树生成和剪枝都使用训练集，所以有错分风险。

决策树剪枝

❑悲观错误剪枝法（PEP）

- 叶结点错误率估计与二项分布

样本的类别判断可以看作一个随机事件，它只有两种可能的结果：

A：类别判断正确，

$\bar{A}$：类别判断错误。

对于n个独立同分布的样本进行类别判断，可以认为这n次独立试验为n重**伯努利试验**，事件A 和$\bar{A}$的发生次数服从**二项分布（binomial distribution）**。

假设一个叶结点i覆盖N_i个训练集样本，其中误判个数为$\overline{N_i}$，$\overline{N_i} \sim b(N_i, P_i)$，其中$P_i$是误判的概率，均值和方差分别为$N_i P_i$，$N_i P_i(1-P_i)$。

❑悲观错误剪枝法（PEP）

- 叶结点错误率估计与二项分布

对于从内部结点t处分枝出的子树T_t，假设其样本总数为N_{T_t}，误判样本总数为$\overline{N_{T_t}}$。其拥有L_{T_t}个叶结点，N_i是第i个叶结点的样本数，$\overline{N_i}$是第i个叶结点的误判样本数。

错误率:

$$r(t) = \frac{\overline{N_t}}{N_t} \text{ 对于叶结点 } t,$$

$$r(T_t) = \frac{\sum_{i=1}^{L_{T_t}} \overline{N_i}}{\sum_{i=1}^{L_{T_t}} N_i} = \frac{\overline{N_{T_t}}}{N_{T_t}} \text{ 对于子树} T_t$$

因为决策树本身是由同一个训练集生成的，因此上述误差无法准确评估模型在新数据集上的错误率（被低估了），因此引入连续性修正项（continuity correction）$\beta = 0.5$

$$P_t = \frac{\overline{N_t} + 0.5}{N_t} \text{对于叶结点} t, P_{T_t} = \frac{\beta L_{T_t} + \sum_{i=1}^{L_{T_t}} \overline{N_i}}{\sum_{i=1}^{L_{T_t}} N_i} = \frac{\overline{N_{T_t}} + \beta L_{T_t}}{N_{T_t}} \text{对于子树} T_t$$

决策树剪枝

❑悲观错误剪枝法（PEP）

- 子树样本误判数的期望与标准差

根据二项分布的性质，可以得到子树样本误判数的期望和方差

$$E\left(\overline{N_{T_t}}\right) = N_{T_t} \times P_{T_t} = \overline{N_{T_t}} + \beta L_{T_t}$$

$$std\left(\overline{N_{T_t}}\right) = \sqrt{N_{T_t} \times P_{T_t} \times (1 - P_{T_t})}$$

其中，$\overline{N_{T_t}} + \beta L_{T_t}$可以看作一种树的复杂度测度，让每个叶结点有一个值为β的代价

- 剪枝策略

当剪枝后的误差小于剪枝前误差的上限时，剪枝成立。即

$$E\left(\overline{N_t'}\right) \leq E\left(\overline{N_{T_t}}\right) + \mathrm{std}\left(\overline{N_{T_t}}\right)$$

决策树剪枝

❑悲观错误剪枝法（PEP）

- 剪枝策略

  当剪枝后的误差小于剪枝前误差的上限时，剪枝成立。即

$$E\left(\overline{N_t'}\right) \leq E\left(\overline{N_{T_t}}\right) + \mathrm{std}\left(\overline{N_{T_t}}\right)$$

- 拓展阅读：A Comparative Analysis of Methods for Pruning Decision Trees

As expected, the subtree T_t makes less errors on the training set than the node t when t becomes a leaf, but sometimes it may happen that $n'(t) \leq n'(T_t)$ due to the continuity correction, in which case the node t is pruned. Nonetheless, this rarely occurs, since the estimate $n'(T_t)$ of the number of misclassifications made by the subtree is still quite optimistic. For this reason, Quinlan weakens the condition, requiring that:

$$e'(t) \leq e'(T_t) + SE(e'(T_t))$$

where

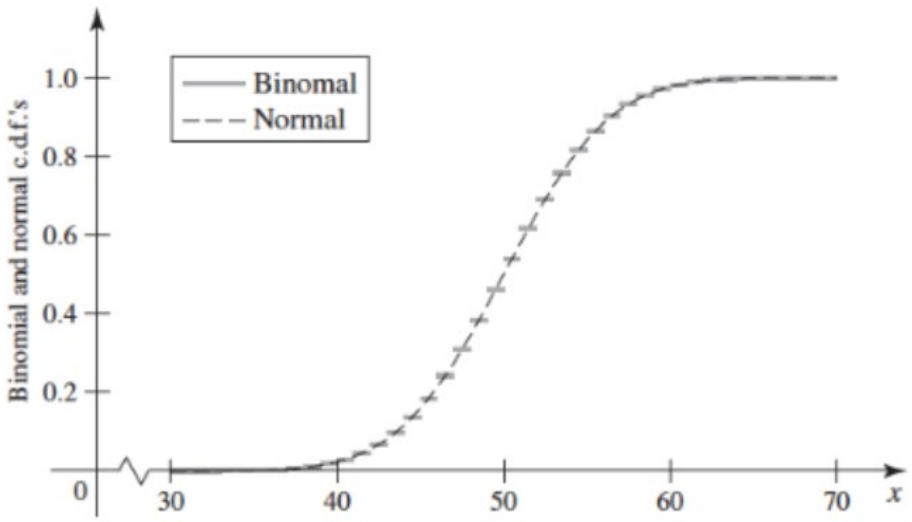
$$SE(e'(T_t)) = [e'(T_t) \cdot (n(t) - e'(T_t)) / n(t)]^{1/2}$$

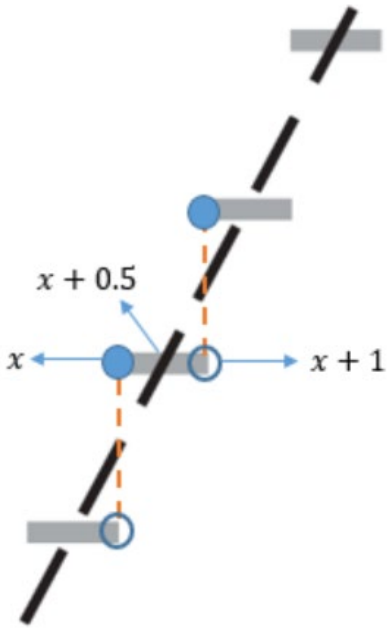
决策树剪枝

❑悲观错误剪枝法（PEP）

- 拓展阅读：连续性修正项

（概率论中）当$np(1-p) \geq 10$时，二项分布近似于均值为np、方差为$np(1-p)$的正态分布。这里特别强调了“近似于”，是因为二项分布的随机变量是离散型的，它的分布函数是阶梯状的，而正态分布的曲线穿过了每个阶梯的中心点。因此引入连续性修正项$\beta = 0.5$.





决策树剪枝

❑悲观错误剪枝法（PEP）

- 拓展阅读：连续性修正项

（PEP中）用于引入树的复杂度测度，参考文献：Esposito F，D Malerba，Semeraro G．**A Comparative Analysis of Methods for Pruning Decision Trees** [J]. IEEE Transactions on Pattern Analysis & Machine Intelligence, 1997, 19(5):476-491.

The introduction of the continuity correction in the estimation of the error rate has no theoretical justification. In statistics, it is used to approximate a binomial distribution with a normal one, but it was never applied to correct overoptimistic estimates of error rates. Actually, the continuity correction is useful only to introduce a tree complexity factor. Nonetheless, such a factor is improperly compared to an error rate, and this may lead to either underpruning or overpruning. Indeed, if T_{max} correctly classifies all training examples, then:

决策树剪枝

❑代价复杂度剪枝法（CCP）（补充阅读）

- CCP综合考虑了预测误差与复杂度，可同时降低偏差和方差
- 代价复杂度：

$$R_\alpha(T) = R(T) + \alpha L_T$$

式中，$R(T)$是预测误差（子树T在验证集上的错误率或均方误差）；L_T表示叶结点的个数，可用来衡量树的复杂度；$\alpha \geq 0$为复杂度参数，衡量了每多一个叶结点带来的过拟合风险，进而追求在偏差和方差之间的平衡。

- 用T表示一棵充分生长的决策树，对于给定的α我们都可以找到使得$R_\alpha(T)$最小的最优子树T^α，且α值越大，得到的子树的复杂度越小.

决策树剪枝

❑代价复杂度剪枝法（CCP）（补充阅读）

- CCP算法的关键是获取上述的最优子树序列。即找到一个从小到大排列的α值序列$(\alpha_1, \alpha_2, \ldots, \alpha_n)$使得其对应的最优子树序列$\{T^{\alpha_1} \supseteq T^{\alpha_2} \supseteq \cdots T^{\alpha_n}\}$具有以下两个特征：
  1. $T^{\alpha_{i+1}}$是由T^{α_i}剪枝而得；
  2. T^{α_i}在这$\alpha \in [\alpha_i, \alpha_{i+1})$区间中始终为最优子树

决策树剪枝

❑代价复杂度剪枝法（CCP）（补充阅读）

- α临界值的确定
  1. 内部结点t的代价复杂度为：$\mathrm{R}_\alpha(t) = R(t) + \alpha$
  2. 以t为根结点的子树T_t的代价复杂度为：$R_\alpha(T_t) = R(T_t) + \alpha L_{T_t}$
  3. 令两式相等，临界值：$\bar{\alpha} = \frac{R(t) - R(T_t)}{L_{T_t} - 1}$
- 此时，t与T_t具有相同的代价复杂度，但t的结点更少，因此在t处执行剪枝
- 其它每一个内部结点都可以算出这样的临界值，我们选临界值最小的一个内部结点进行修剪，便可以获得对应的最优子树。
- 采用交叉验证法在$\{T^{\alpha_0}, T^{\alpha_1}, T^{\alpha_2}, \ldots T^{\alpha_n}\}$中选择唯一最优子树

决策树

❑ 学习目标

- 学习决策树特征选择的指标
  - 了解信息熵的由来和意义
  - 知道信息的测量方法
  - 掌握熵的表达式及其证明
- 知道经典的决策树算法
- 了解决策树剪枝的原因、策略及其异同点
- **掌握应用决策树算法解决实际问题的方法**

应用实例

❑数据集介绍

- 本例将采用2016.11.01的网格交通流数据，同时选取“平均速度”、“流量”、“速度标准差”、“平均停车次数”这4个特征作为模型输入，进行分类研究

平均速度	流量	速度标准差	平均停车次数	交通状态
6.61	3	3.412	0.00	缓行
9.47	6	2.614	0.00	缓行
8.12	5	4.485	0.60	缓行
11.92	15	6.444	0.40	畅通
12.04	6	6.204	2.70	拥堵

应用实例

❑建模预处理

- 首先对研究范围内的数据进行载入并进行训练集与测试集的划分。由于决策树算法完全不受数据缩放的影响，每个特征被单独处理，而且数据的划分也不依赖于缩放，因此这里不需要对不同交通流参数做标准化处理。

```
1.  import numpy as np
2.  import pandas as pd
3.  from sklearn.tree import DecisionTreeClassifier
4.  from sklearn.model_selection import train_test_split
5.  # 加载数据文件
6.  data = pd.read_csv('DATASET-B.csv',nrows=200000)
7.  x = data[['aveSpeed','stopNum','volume','speed_std']]  #特征提取
8.  y = pd.Categorical(data['labels']).codes  #类别特征处理
9.   #划分训练集测试集（7:3）
10. x_train, x_test, y_train, y_test = train_test_split(x, y, train_size=0.7,random_state=1)
```

应用实例

❑模型训练

- 我们用DecisionTreeClassifier类中的fit()函数对训练集进行建模，构建决策树模型

```
1.  # 决策树参数估计
2.  model = DecisionTreeClassifier(criterion='gini', min_samples_split=10,
3.                                 min_samples_leaf=40,max_depth=10,
4.                                 class_weight='balanced')
5.  model.fit(x_train, y_train)
```

应用实例

❑分类准确性及可视化

- 训练完毕的决策树模型在测试集上的分类正确率为93.88%

```
# 测试集上的预测结果
y_test_hat = model.predict(x_test)# 测试数据
y_test = y_test.reshape(-1)
result = (y_test_hat == y_test)# True 则预测正确，False 则预测错误
acc = np.mean(result)
print('准确度: %.2f%%' % (100 * acc))
```

- 通过model.feature_importances_计算各个特征的贡献度。选取贡献度较大的平均速度和平均停车次数重新建模，通过meshgrid可视化的形式对特征空间划分和样本分类效果进行直观展示。

应用实例

❑分类准确性及可视化