



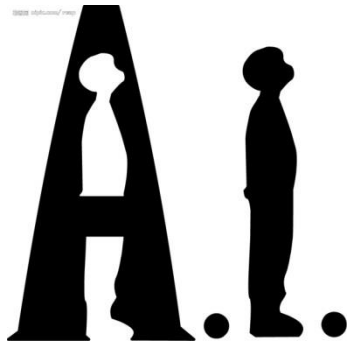
# 机器学习



讲师：曾江峰



E-mail: [jfzeng@ccnu.edu.cn](mailto:jfzeng@ccnu.edu.cn)



大数据，成就未来

# 决策树



# outline

---



1.1 引言

1.2 决策树简介

1.3 决策树的核心

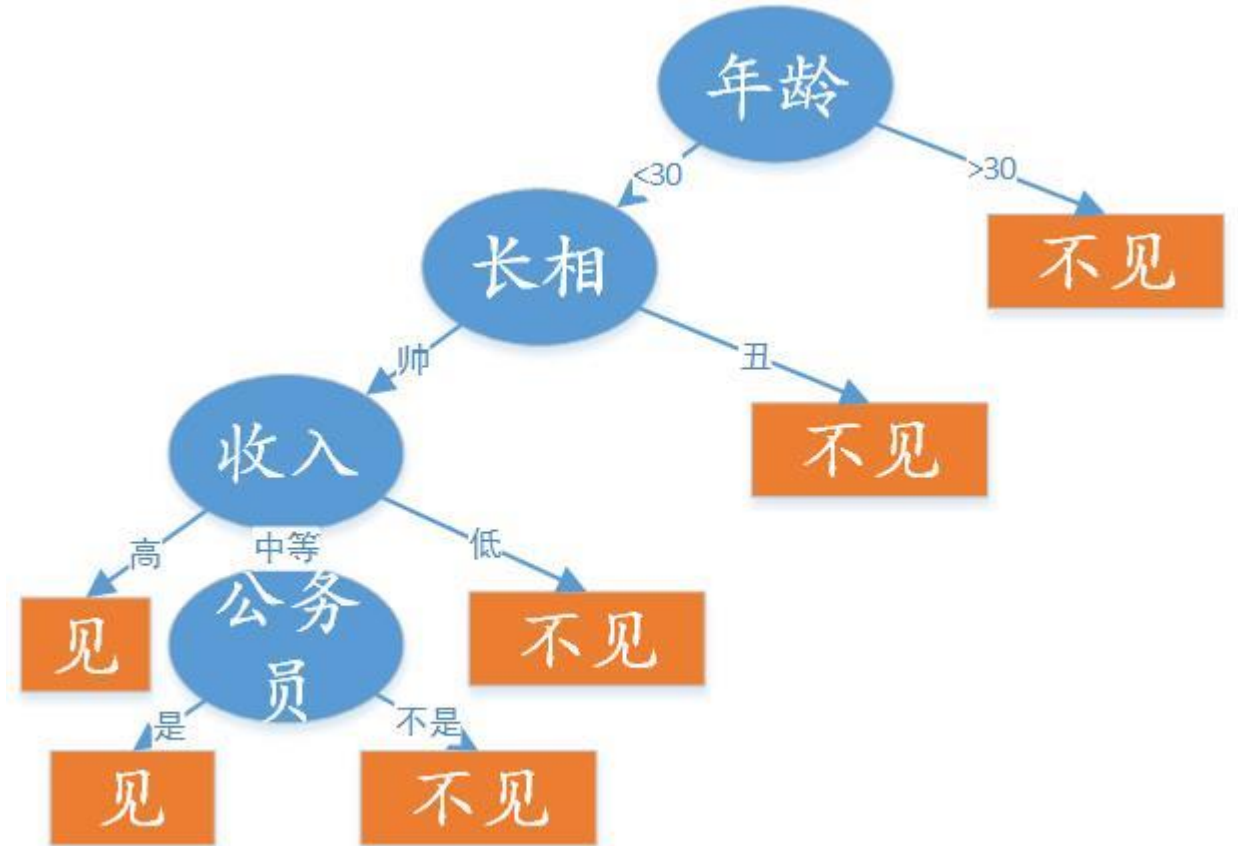
1.4 案例分析

1.5 决策树优缺点



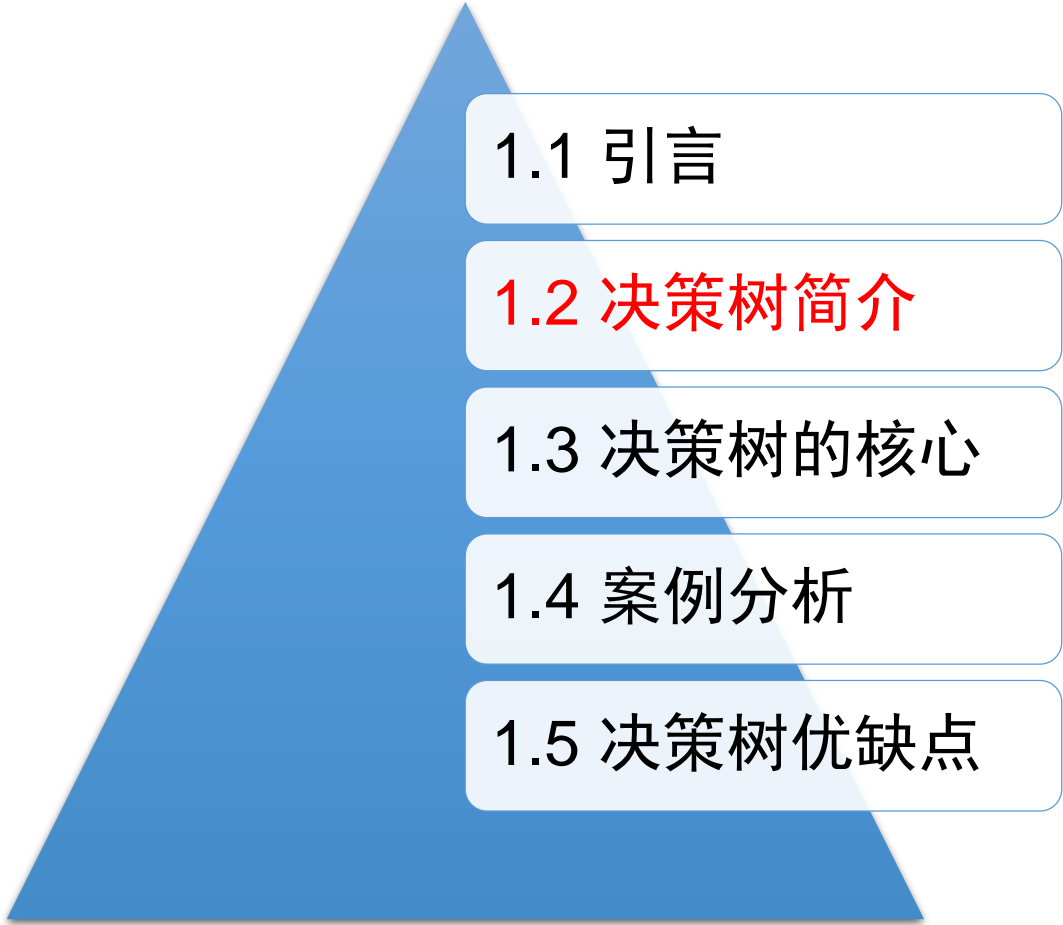
# 一个通俗易懂的例子——相亲

- 女儿：年纪多大了？
- 母亲：26
- 女儿：长相如何？
- 母亲：挺帅的
- 女儿：收入如何？
- 母亲：不算很高，中等情况
- 女儿：是公务员不？
- 母亲：是，在税务局上班
- 女儿：那好，我去见见



# outline

---



1.1 引言

1.2 决策树简介

1.3 决策树的核心

1.4 案例分析

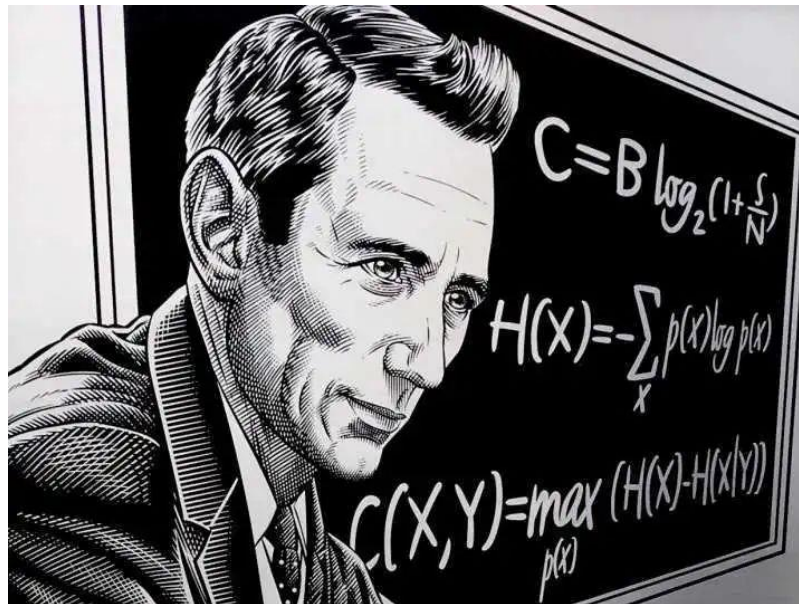
1.5 决策树优缺点



# 决策树

- 决策树的思想主要来源于Quinlan在1986年提出的ID3算法和1993年提出的C4.5算法，以及有Breiman等人在1984年提出的CART算法。
- 和朴素贝叶斯算法一样，散发着浓厚的数学气味。

信息论基础



# 决策树简史

- 第一个决策树算法：CLS (Concept Learning System)  
[E. B. Hunt, J. Marin, and P. T. Stone's book "*Experiments in Induction*" published by Academic Press in 1966]
- 使决策树受到关注、成为机器学习主流技术的算法：ID3  
[J. R. Quinlan's paper in a book "*Expert Systems in the Micro Electronic Age*" edited by D. Michie, published by Edinburgh University Press in 1979]
- 最常用的决策树算法：C4.5  
[J. R. Quinlan's book "*C4.5: Programs for Machine Learning*" published by Morgan Kaufmann in 1993]

# 决策树简史

- 可以用于回归任务的决策树算法： CART (Classification and Regression Trees)  
[L. Breiman, J. H. Friedman, R. A. Olshen, and C. J. Stone's book "Classification and Regression Trees" published by Wadsworth in 1984]
- 基于决策树的最强大算法： RF (Random Forest)  
[L. Breiman's MLJ'01 paper "Random Forest"]  
这是一种“集成学习”方法

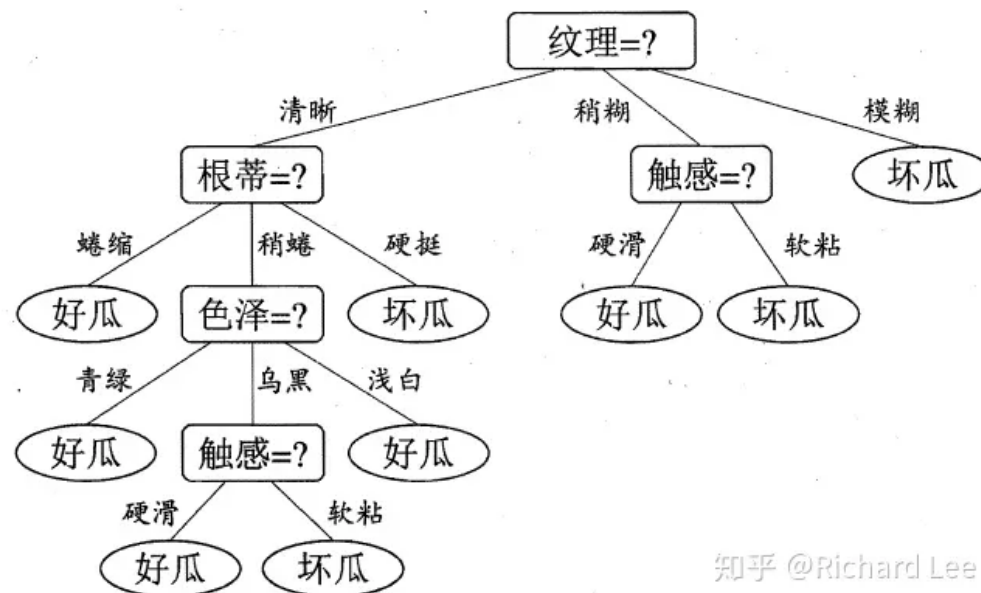


# 决策树的概念

- 决策树通过树形结构对数据进行分类。
- 一颗完整的结构树由结点和有向边构成，其中内部结点表示特征，叶子结点表示类别。
- 基本概念：决策树基于树结构，从顶往下，依次对样本的（一个或多个）属性进行判断，直到决策树的叶节点并导出最终结果。

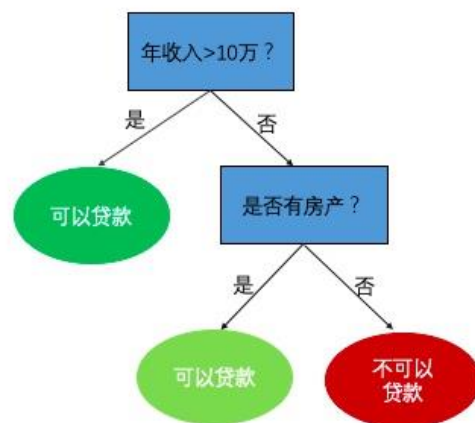
# 决策树的工作原理

- 决策树从根结点开始，选取数据中某一特征，根据特征取值对实例进行分配，通过不断地选取特征进行实例分配，决策树可以达到对所有实例进行分类的目的。
- 整个决策树的学习过程就是一个递归地选择最优特征，并根据该特征对数据集进行划分，使得各个样本都得到一个最好的分类的过程。

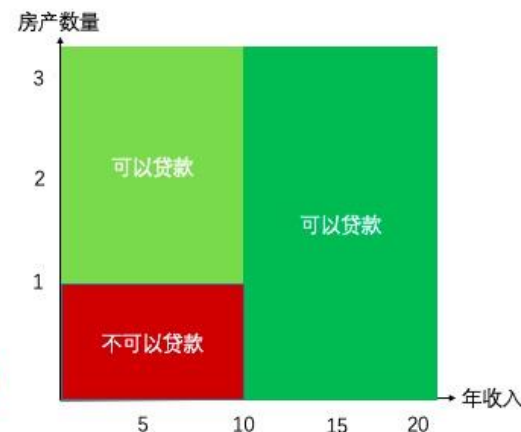


# 决策树的本质

- 决策树模型通过一系列if then决策规则的集合，将特征空间划分成有限个不相交的子区域，对于落在相同子区域的样本，决策树模型给出相同的预测值。
- 这些if then决策规则之间的层次关系形成一个树形结构，称之为决策树，这些不相交的子区域和树结构的叶子节点一一对应。



if then决策规则的层次结构



特征空间的不相交子区域划分

# outline

---



1.1 引言

1.2 决策树简介

1.3 决策树的核心

1.4 案例分析

1.5 决策树优缺点



# 决策树的核心

- 特征选择方法
  - 衡量各个特征的分类能力
  - 制定标准选取最优特征来划分数据集
- 决策树构建过程
  - 决策树生成算法
  - 通过特征选择方法递归地选择最优特征进行决策树的构造
- 决策树剪枝策略
  - 决策树生成算法能够最大限度地拟合训练集，但极有可能导致过拟合现象。
  - 对决策树进行剪枝，缓解过拟合现象

# 特征选择方法

# 什么是特征选择

- 为了能够构建一颗分类性能良好的决策树，我们需要从训练集中不断选取具有较强分类能力的特征。
- 决策树的特征选择就是从数据集中选取具备较强分类能力的特征对数据集进行划分。

那么什么样的特征才具备较强分类能力呢？或者说，我们按照什么标准来选择去最优特征？



# :: Decision Tree

## Measures that can be used to capture the purity of split

Information Gain, Gain Ratio and Gini Index are the most common methods of *attribute selection*

信息增益

Information Gain

信息增益比

Information Gain Ratio

基尼系数

Gini Index



# :: Decision Tree

## **We need to understand Entropy first before we move on**

ID3 uses *Entropy* and *Information Gain* to construct a decision tree. It's necessary to understand what Entropy means.

---

What's Entropy ?

---



# :: Decision Tree

## Generally entropy is a measure of disorder or uncertainty

Entropy is a concept used in Physics, mathematics, computer science (information theory) and other fields of science.<sup>1</sup> The concept of entropy originated in thermodynamics as a measure of molecular disorder: entropy approaches zero when molecules are still and well ordered.<sup>2</sup>

### Entropy in Physics:



Low



Medium



High

Entropy, so far, had been a concept in physics. If the particles inside a system have many possible positions to move around, then the system has **high entropy**, and if they have to stay rigid, then the system has **low entropy**.<sup>2</sup>

For example, water in its three states, solid, liquid, and gas, has different entropies. The molecules in ice have to stay in a lattice, as it is a rigid system, so ice has low entropy. The molecules in water have more positions to move around, so water in liquid state has medium entropy. The molecules inside water vapor can pretty much go anywhere they want, so water vapor has high entropy.<sup>3</sup>

# :: Decision Tree

**More uncertainty, more entropy!**

entropy is a measure of  
**disorder** or **uncertainty**



High Entropy


Increase entropy




Low Entropy

- High Disorder
- Messy
- High Randomness;
- High uncertainty
- Low certainty

- Well- ordered
- Neat
- Low Randomness
- Low uncertainty
- High certainty



那么什么样的特征才具备较强分类能力呢？或者说，我们按照什么标准来选择去最优特征？



如果用一个特征对数据集进行划分，使得分支结点中的样本尽可能属于同一类别，即该结点有着较高的纯度，那么该特征对数据集而言就具备较强的分类能力。

# 信息熵 **information entropy**

- 香农将“信息”定义为：信息是用来消除随机不确定性的东西。
- 将熵引入信息论中，是为了更好地量化信息。
- 在信息论和概率统计中，熵是一种描述随机变量不确定性的度量方式，也可以描述样本的纯度。
- 信息熵越低，样本不确定性越小，相应的纯度就越高。

# 信息熵的公式

$$D = \{(x_1, y_1), (x_2, y_2), \dots, (x_n, y_n)\}$$

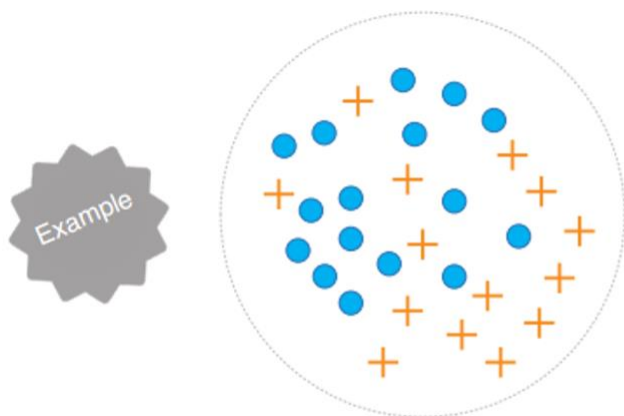
$$x_i = (x_{i1}, x_{i2}, \dots, x_{id})$$

$$c \in (c_1, c_2, \dots, c_m)$$

- 样本数据集D的熵可以定义为：

数据集D关于类别标签的熵  $E(D) = - \sum_{k=1}^m p(c_k) \log p(c_k)$

$$p(c_k) = P(Y = c_k) = \frac{1}{n} \sum_{i=1}^n I(y_i = c_k) = \frac{n_{c_k}}{n}$$



Entire population (30 instances)

16/30 are blue circles:

$$\log_2 \left( \frac{16}{30} \right) = -0.9;$$

14/30 are orange crosses:

$$\log_2 \left( \frac{14}{30} \right) = -1.1;$$

$$Entropy = - \sum_{i=1}^n p_i \log_2 p_i = - \left( \frac{16}{30} \right) (-0.9) - \left( \frac{14}{30} \right) (-1.1) = 0.99$$



Video: Shannon Entropy and Information Gain (21mins)

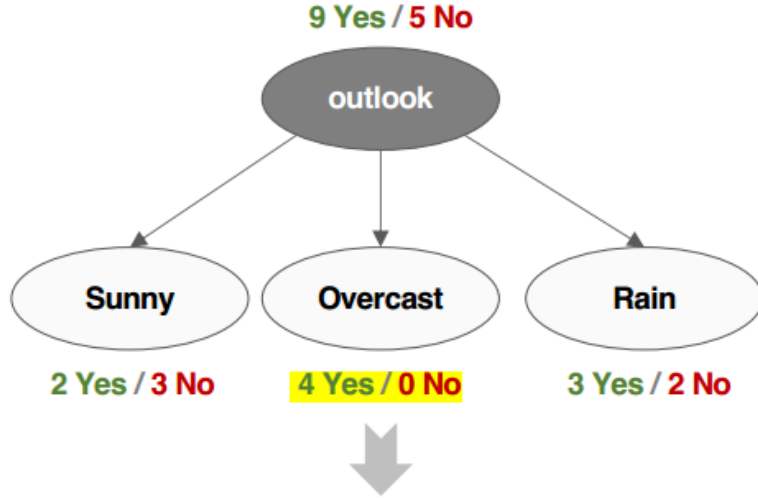
[https://www.youtube.com/watch?time\\_continue=1247&v=9r7FIXEAGvs](https://www.youtube.com/watch?time_continue=1247&v=9r7FIXEAGvs)

# :: Decision Tree

## Another example of calculating entropy

Use entropy to measure the “purity” of the split

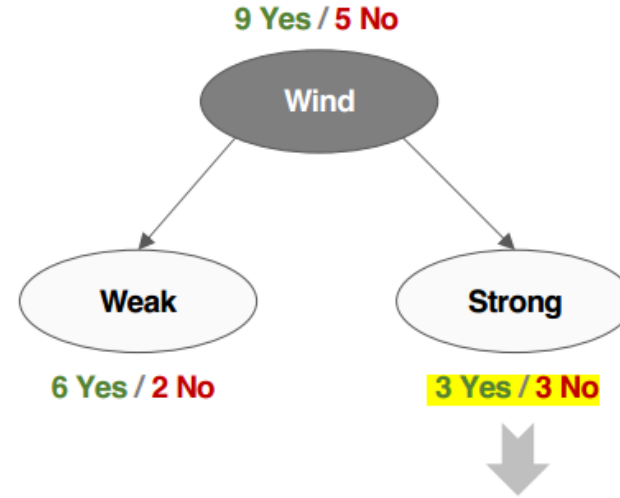
$$Entropy = - \sum_{i=1}^n p_i \log_2 p_i$$



**Pure set** (4 yes / 0 No)  $\Rightarrow$  completely certain(100%)

The entropy is 0 If the sample is completely homogeneous

$$Entropy = -\frac{4}{4}\log_2 \frac{4}{4} - \frac{0}{4}\log_2 \frac{0}{4} = 0$$



**Impure set** (3 yes / 3 No)  $\Rightarrow$  completely uncertain(50%)

The entropy is 1 If the sample is an equally divided

$$Entropy = -\frac{3}{6}\log_2 \frac{3}{6} - \frac{3}{6}\log_2 \frac{3}{6} = 1$$

# 信息增益

Information Gain

i.e. entropy reduction

$$\text{Information Gain} = \text{Entropy}_{\text{before}} - \text{Entropy}_{\text{after}}$$

- 信息增益是指由于得到特征  $X$  的信息而使得类别  $Y$  的信息不确定性（熵）减少的程度。
- 信息增益是一种描述目标类别确定性增加的量，特征的信息增益越大，目标类的确定性越大。
- 给定特征  $X^j, j=1, 2, \dots, d$ ，其中，特征  $X^j$  的取值空间为  $t$ ，则样本数据集  $D$  的条件熵为：

$$E(D|X^j) = \sum_{i=1}^{|t|} p(X^j = t_i) E(D|X^j = t_i) \quad p(X^j = t_i) = \frac{n_{t_i}}{n}$$

$$E(D|X^j = t_i) = E(D_{t_i}) = - \sum_{k=1}^m p(c_k^{t_i}) \log p(c_k^{t_i})$$



# 信息增益

- 给定训练集 $D$ ，根据特征  $X^j, j=1, 2, \dots, d$ ，划分数数据集，信息增益可定义为经验熵  $E(D)$  与经验条件熵  $E(D|X^j)$  之差：

$$g(D, X^j) = E(D) - E(D|X^j)$$

- 每个特征一般会有不同的信息增益，信息增益越大，代表对应的特征分类能力越强。
- 构造决策树时，可以使用信息增益进行特征选择。
- ID3 算法是基于信息增益进行特征选择的。

## Example : Can we play tennis today ?

Lets say we have a table that decided if we should play tennis under certain circumstances. These could be the **outlook** of the weather; the **temperature**; the **humidity** and the strength of the **wind**:

Day	Outlook	Temperature	Humidity	Wind	Play Tennis ?
1	Sunny	Hot	High	Weak	No
2	Sunny	Hot	High	Strong	No
3	Overcast	Hot	High	Weak	Yes
4	Rain	Mild	High	Weak	Yes
5	Rain	Cool	Normal	Weak	Yes
6	Rain	Cool	Normal	Strong	No
7	Overcast	Cool	Normal	Strong	Yes
8	Sunny	Mild	High	Weak	No
9	Sunny	Cool	Normal	Weak	Yes
10	Rain	Mild	Normal	Weak	Yes
11	Sunny	Mild	Normal	Strong	Yes
12	Overcast	Mild	High	Strong	Yes
13	Overcast	Hot	Normal	Weak	Yes
14	Rain	Mild	High	Strong	No

*there are 5 cases of not being able to play a game, and 9 cases of being able to play a game.*



If we were given a new instance :

X = (**Outlook** = Sunny, **Temperature** = Cool,  
**Humidity** = High, **Wind** = Strong)

We want to know if we can play a game or not ?

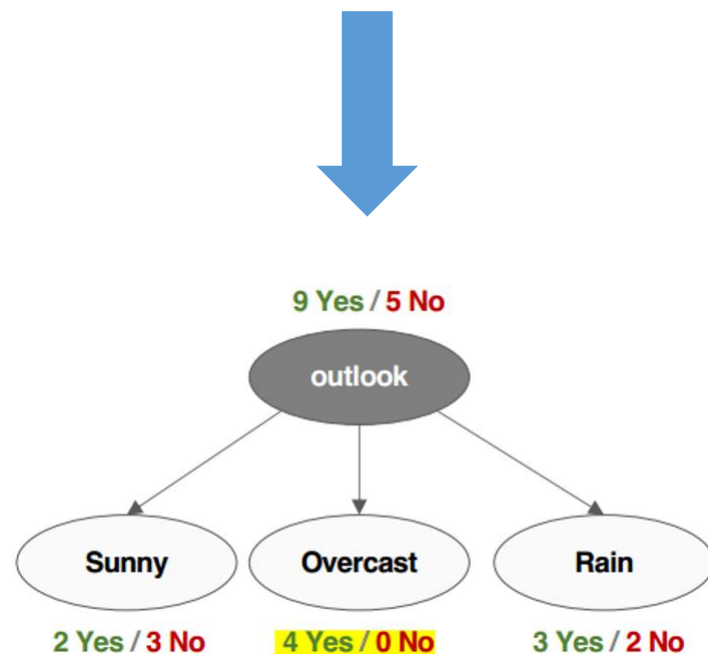
```
def entropy(ele):
    probs = [ele.count(i)/len(ele) for i in set(ele)]
    entropy = -sum([prob*log(prob, 2) for prob in probs])
    return entropy
```

### 根据特征的取值来切分数据集

```
def split_dataframe(data, col):
    unique_values = data[col].unique()
    result_dict = {elem : pd.DataFrame for elem in unique_values}
    for key in result_dict.keys():
        result_dict[key] = data[:, data[col] == key]
    return result_dict
```

```
def info_gain(df_data, label, featureX):
    entropy_D = entropy(df_data[label].tolist())
    splited_set = split_dataframe(df_data, featureX)
    entropy_DX = 0
    for subset_col, subset in splited_set.items():
        entropy_DXj = entropy(subset[label].tolist())
        entropy_DX += len(subset)/len(df_data) * entropy_DXj
    info_gain = entropy_D - entropy_DX
    print('根据 ' + featureX + ' 特征进行划分, 信息增益为: ', info_gain)
    return info_gain
```

根据 windy 特征进行划分, 信息增益为: 0.04812703040826927  
 根据 outlook 特征进行划分, 信息增益为: 0.2467498197744391  
 根据 humidity 特征进行划分, 信息增益为: 0.15183550136234136  
 根据 temp 特征进行划分, 信息增益为: 0.029222565658954647



# The bias of information gain

- 信息增益存在的问题：当某个特征取值较多时，该特征的信息增益计算结果就会较大。
- 举个极端的案例，样本数据集 $n$ 条，某个特征取值个数也为 $n$ ，根据这个特征划分数数据集将得到 $n$ 个分支，每个分支包含1个样本，每个结点的纯度为1，与其他特征相比，这个特征的信息增益绝对最大。这样构建的决策树是无效的。
- 因此，**基于信息增益选择特征时，会偏向于取值多的特征。**

# 信息增益比

## Information Gain Ratio

- 给定训练集 $D$ ，根据特征  $X^j, j=1, 2, \dots, d$ ，划分数数据集，信息增益比其信息增益  $g(D, X^j)$  与数据集 $D$ 关于特征  $X^j$  的熵  $E_{X^j}(D)$  的比值：

$$g_R(D, X^j) = \frac{g(D, X^j)}{E_{X^j}(D)}$$

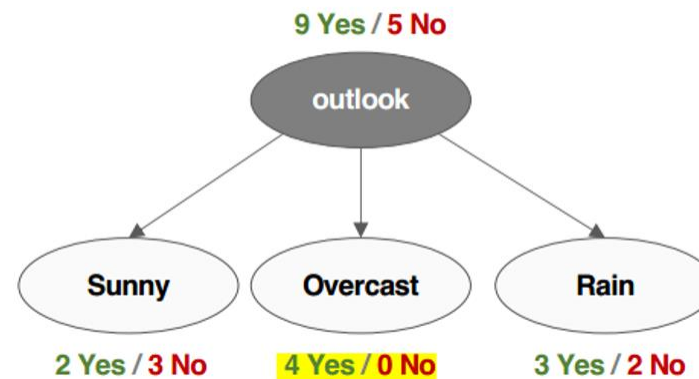
$$g(D, X^j) = E(D) - E(D|X^j)$$

$$E_{X^j}(D) = - \sum_{i=1}^{|t|} p(X^j = t_i) \log p(X^j = t_i) \quad p(X^j = t_i) = \frac{n_{t_i}}{n}$$

- 跟信息增益一样，在基于信息增益比进行特征选择时，我们选择信息增益比最大的特征作为决策树分裂点。
- C4.5是基于信息增益比进行特征选择的。

```
def info_gain_rate(df_data, label, featureX):
    entropy_D = entropy(df_data[label].tolist())
    splited_set = split_dataframe(df_data, featureX)
    entropy_DX = 0
    for subset_col, subset in splited_set.items():
        entropy_DXj = entropy(subset[label].tolist())
        entropy_DX += len(subset)/len(df_data) * entropy_DXj
    info_gain = entropy_D - entropy_DX
    entropy_DXj = entropy(df_data[featureX].tolist())
    info_gain_rate = info_gain/entropy_DXj
    print('根据 ' + featureX + ' 特征进行划分, 信息增益比为: ', info_gain_rate)
    return info_gain_rate
```

根据 windy 特征进行划分, 信息增益比为: 0.048848615511520595  
根据 outlook 特征进行划分, 信息增益比为: 0.15642756242117517  
根据 humidity 特征进行划分, 信息增益比为: 0.15183550136234136  
根据 temp 特征进行划分, 信息增益比为: 0.01877264622241867



# 基尼系数



Gini Index

- 基尼系数（英文：Gini index、Gini Coefficient），是国际上通用的、用以衡量一个国家或地区居民收入差距的常用指标之一。
- 基尼系数最大为“1”，最小等于“0”。
- 基尼系数越接近0表明收入分配越是趋向平等。有不少人认为基尼系数小于0.2时，居民收入过于平均，0.2-0.3之间时较为平均，0.3-0.4之间时比较合理，0.4-0.5时差距过大，大于0.5时差距悬殊。
- 基尼系数也是一种较好的特征选择方法。

# 基尼系数

- 给定数据集D，则该数据集的基尼系数可定义为：

$$Gini(D) = \sum_{k=1}^m p(c_k) * (1 - p(c_k)) = 1 - \sum_{k=1}^m (p(c_k))^2 \quad p(c_k) = \frac{n_{c_k}}{n}$$

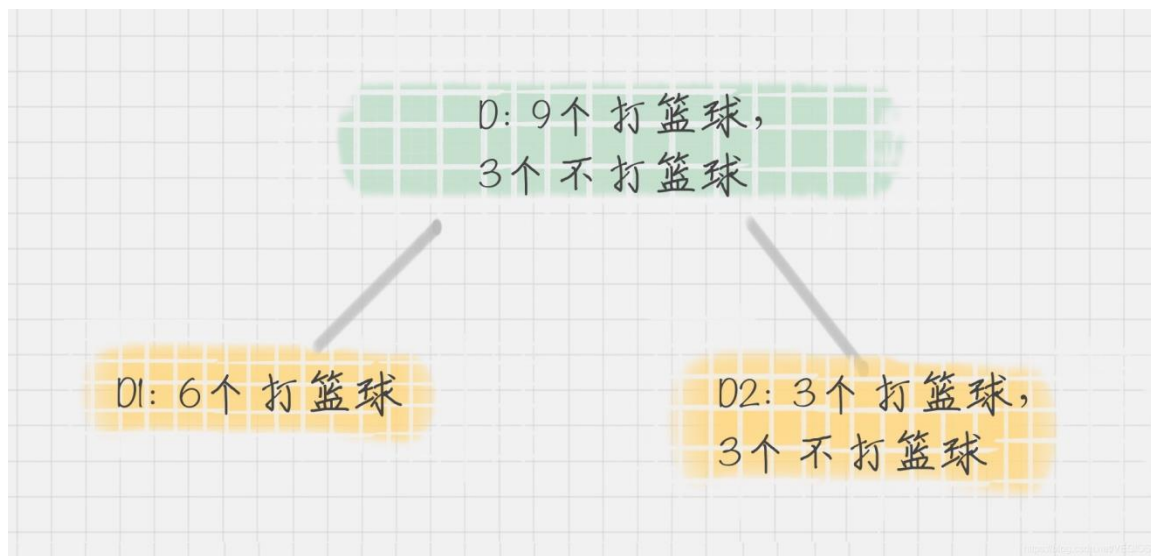
- 给定训练集D，根据特征  $X^j, j=1, 2, \dots, d$ ，划分数数据集为  $|t|$  部分，则在该特征条件下训练集的基尼系数可定义为：

$$Gini(D, X^j) = \sum_{i=1}^{|t|} \frac{n_{t_i}}{n} Gini(D_i) \quad Gini(D_i) = 1 - \sum_{k=1}^m (p(c_k^i))^2 \quad p(c_k^i) = \frac{n_{c_k^i}}{n_{t_i}}$$

- $Gini(D, X^j)$  越小，对应的特征对训练样本的分类能力越强。
- CART算法是基于基尼系数进行特征选择的。



# 基尼系数



$$GINI(D, A) = \frac{D_1}{D} GINI(D_1) + \frac{D_2}{D} GINI(D_2)$$

$$GINI(D_1) = 0$$

$$GINI(D_2) = 0.5$$

$$GINI(D, A) = \frac{6}{12} GINI(D_1) + \frac{6}{12} GINI(D_2) = 0.25$$

# 决策树构建过程

# ID3

- 核心是基于信息增益递归地选择最优特征构造决策树。
- 给定训练集 $D$ 、特征集合 $X = \{X^1, X^2, \dots, X^d\}$ 以及信息增益阈值 $\mathcal{E}$ ，ID3算法的流程如下：
  - (1) 如果 $D$ 中所有实例属于同一类别  $C_k$ ，那么所构建的决策树 $T$ 为单节点树，并且类 $C_k$ 即为该结点的类标记，返回 $T$ 。
  - (2) 计算特征集合 $X$ 中各特征对 $D$ 的信息增益，选择信息增益最大的特征 $X^*$ 。
  - (3) 如果 $X^*$ 的信息增益小于阈值 $\mathcal{E}$ ，则将 $T$ 视为单结点树，并将 $D$ 中所属数量最多的类 $C_k$ 作为该结点的类标记，并返回 $T$ 。
  - (4) 否则，根据 $X^*$ 的不同取值将 $D$ 划分为若干个非空子集 $D_i$ ，以 $D_i$ 中所属数量最多的类作为标记构建子结点，由结点和子结点构成树 $T$ 并返回。
  - (5) 对第 $i$ 个子结点，以 $D_i$ 为训练集，以 $X - X^*$ 为特征值，递归地调用(1) ~ (4)步，即可得到决策树子树 $T_i$ 并返回。

```

import numpy as np
import pandas as pd
from math import log

def entropy(ele):
    probs = [ele.count(i)/len(ele) for i in set(ele)]
    entropy = -sum([prob*log(prob, 2) for prob in probs])
    return entropy

### 根据特征的取值来切分数据集
def split_dataframe(data, col):
    unique_values = data[col].unique()
    result_dict = {elem : pd.DataFrame for elem in unique_values}
    for key in result_dict.keys():
        result_dict[key] = data[:, [data[col] == key]]
    return result_dict

def choose_best_col(df, label):
    entropy_D = entropy(df[label].tolist())
    cols = [col for col in df.columns if col not in [label]]
    max_value, best_col = -999, None
    max_splited = None
    for col in cols:
        splited_set = split_dataframe(df, col)
        entropy_DA = 0
        for subset_col, subset in splited_set.items():
            entropy_Di = entropy(subset[label].tolist())
            entropy_DA += len(subset)/len(df) * entropy_Di
        info_gain = entropy_D - entropy_DA

        if info_gain > max_value:
            max_value, best_col = info_gain, col
            max_splited = splited_set
    return max_value, best_col, max_splited

if __name__ == "__main__":
    df = pd.read_csv('./example_data.csv', dtype={'windy': 'str'})
    treel = ID3Tree(df, 'play')
    treel.construct_tree()
    treel.print_tree(treel.root, "")

```

```

class ID3Tree:
    class Node:
        def __init__(self, name):
            self.name = name
            self.connections = {}

        def connect(self, label, node):
            self.connections[label] = node

    def __init__(self, data, label):
        self.columns = data.columns
        self.data = data
        self.label = label
        self.root = self.Node("Root")

    def print_tree(self, node, tabs):
        print(tabs + node.name)
        for connection, child_node in node.connections.items():
            print(tabs + "\t" + "(" + connection + ")")
            self.print_tree(child_node, tabs + "\t\t")

    def construct_tree(self):
        self.construct(self.root, "", self.data, self.columns)

    def construct(self, parent_node, parent_connection_label, input_data, columns):
        max_value, best_col, max_splited = choose_best_col(input_data[columns], self.label)

        if not best_col:
            node = self.Node(input_data[self.label].iloc[0])
            parent_node.connect(parent_connection_label, node)
            return

        node = self.Node(best_col)
        parent_node.connect(parent_connection_label, node)

        new_columns = [col for col in columns if col != best_col]

        for splited_value, splited_data in max_splited.items():
            self.construct(node, splited_value, splited_data, new_columns)

```

## C4.5

- 核心是基于信息增益比递归地选择最优特征构造决策树。
- 给定训练集 $D$ 、特征集合 $X = \{X^1, X^2, \dots, X^d\}$ 以及信息增益阈值比 $\mathcal{E}$ ，C4.5算法的流程如下：
  - (1) 如果 $D$ 中所有实例属于同一类别  $\mathbf{C}_k$ ，那么所构建的决策树 $T$ 为单结点树，并且类 $\mathbf{C}_k$ 即为该结点的类标记，返回 $T$ 。
  - (2) 计算特征集合 $X$ 中各特征对 $D$ 的信息增益比，选择信息增益比最大的特征  $X^*$ 。
  - (3) 如果 $X^*$ 的信息增益比小于阈值 $\mathcal{E}$ ，则将 $T$ 视为单结点树，并将 $D$ 中所属数量最多的类 $\mathbf{C}_k$ 作为该结点的类标记，并返回 $T$ 。
  - (4) 否则，根据 $X^*$ 的不同取值将 $D$ 划分为若干个非空子集 $D_i$ ，以 $D_i$ 中所属数量最多的类作为标记构建子结点，由结点和子结点构成树 $T$ 并返回。
  - (5) 对第 $i$ 个子结点，以 $D_i$ 为训练集，以 $X - X^*$ 为特征值，递归地调用(1) ~ (4)步，即可得到决策树子树 $T_i$ 并返回。

# CART

- CART算法的全称为Classification And Regression Tree，顾名思义，CART既可以用于分类，也可以用于回归。
- CART算法不仅包括决策树生成算法，还包括决策树剪枝算法。
- CART生成的决策树是二叉决策树，内部结点取值为“是”和“否”，这种结点划分方法等价于递归地二分每个特征，将特征空间划分为有限个单元。

# CART分类树

- 核心是基于基尼系数递归地选择最优特征构造分类决策树。
- 给定训练集 $D$ 、特征集合 $\mathbf{X} = \{\mathbf{X}^1, \mathbf{X}^2, \dots, \mathbf{X}^d\}$ ，CART分类树生成算法的流程如下：
  - (1) 对于每个特征 $\mathbf{X}^j, j = 1, 2, \dots, d$ 及其所有取值 $V_i$ ，根据 $\mathbf{X}^j = V_i$ 将数据集划分为 $D_1$ 和 $D_2$ 两个部分，计算 $\mathbf{X}^j = V_i$ 时的基尼系数。
  - (2) 取基尼系数最小的特征及其对应的划分点作为最有特征和最优切分点，据此将当前结点划分为两个子结点，将训练集根据特征分配到两个子结点中。
  - (3) 对两个子结点递归地调用(1)和(2)，直到满足停止条件。
  - (4) 最后即可生成CART分类决策树。
- 待预测样本落至某一叶子节点，则输出该叶子节点中所有样本所属类别最多的那一类（即叶子节点中的样本可能不是属于同一个类别，则多数为主）。

```
from sklearn import datasets
from sklearn.tree import DecisionTreeClassifier
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score, mean_squared_error
data = datasets.load_iris()
X, y = data.data, data.target
y = y.reshape(-1,1)
X_train, X_test, y_train, y_test = train_test_split(X, y.reshape(-1,1), test_size=0.3)

clf = DecisionTreeClassifier()
clf.fit(X_train, y_train)
y_pred = clf.predict(X_test)

print(accuracy_score(y_test, y_pred))
```



# CART回归树

- 回归树用于目标变量为连续型的建模任务，其特征选择准则是平方误差最小化。
- 待预测样本落至某一叶子节点，则输出该叶子结点中所有样本的均值。

```
from sklearn import datasets
from sklearn.tree import DecisionTreeRegressor
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score, mean_squared_error
X, y = datasets.load_boston(return_X_y=True)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3)

reg = DecisionTreeRegressor()
reg.fit(X_train, y_train)
y_pred = reg.predict(X_test)
mse = mean_squared_error(y_test, y_pred)

print("Mean Squared Error:", mse)
```

# 决策树剪枝策略

# 分类决策树损失函数

- 叶子结点个数为 $|T|$ ， $t$ 为树 $T$ 的叶子结点，每个叶子结点有 $N_t$ 个样本。
- $H_t(T)$ 为叶子结点上的经验熵。

$$C_\alpha(T) = \sum_{t=1}^{|T|} N_t H_t(T) + \alpha |T|$$

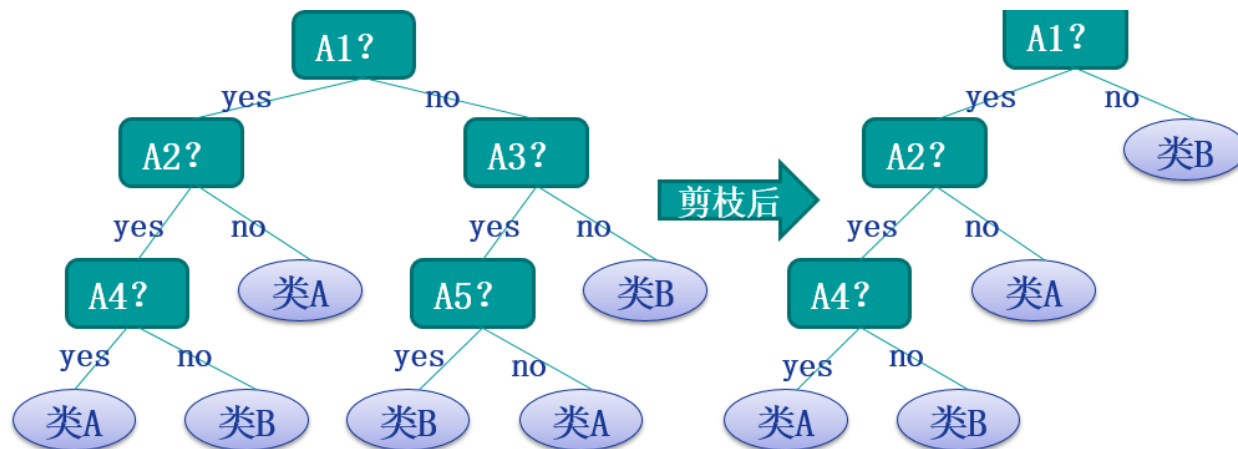
知乎 @雷神

$$H_t(T) = - \sum_{k=1}^M p_{tk} \log p_{tk}$$

知乎 @雷神

# 决策树剪枝

- 一个完整的决策树算法，不仅包括决策树生成算法，还包括决策树剪枝算法。
- 决策树生成算法递归地生成决策树，生成的决策树大而全，但很容易导致过拟合现象。
- 决策树剪枝则是对已生成的决策树进行简化的过程，通过剪掉一些子树或者叶子结点，并将根节点或父结点作为新的叶子结点，从而提升决策树的泛化性能。



# 剪枝策略——预剪枝

- 预剪枝：在决策树生成过程中提前停止树的生长。
- 主要思路是在决策树结点分裂之前，计算当前结点划分能否提升模型泛化能力，如果不能，则决策树在该结点停止生长。
- 简单高效，适用于大规模求解问题。
- 但提前停止树的生长，一定程度上存在欠拟合风险。

# 剪枝策略——后剪枝

- 后剪枝：在决策树生成后，从最底端的叶子结点进行剪枝，自底向上对完全树进行逐结点剪枝。
  - (1) 计算每个叶子结点的经验熵 $H_t(T)$ 。
  - (2) 递归地自底向上回缩，假设一组叶子结点会到父结点前后的树分别为 $T_{\text{before}}$ 和 $T_{\text{after}}$ ，分别计算对应的损失函数值 $L(T_{\text{before}})$ 和 $L(T_{\text{after}})$ ，如果 $L(T_{\text{after}}) \leq L(T_{\text{before}})$ ，则进行剪枝，将父结点变为新的叶子结点。
  - (3) 重复第(2)步，直至得到损失函数最小的子树 $T^*$ 。
- CART算法的剪枝正是后剪枝方法。

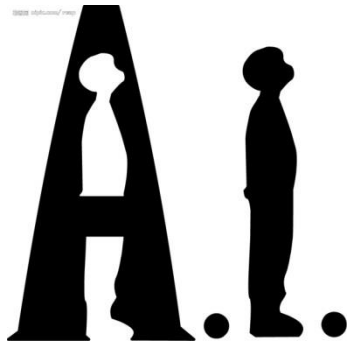
# ID3、C4.5、CART 区别

- 划分标准的差异：信息增益、信息增益比、基尼系数。
- 使用场景的差异：ID3 和 C4.5 都只能用于分类问题，CART 可以用于分类和回归问题。
- ID3 和 C4.5 是多叉树，速度较慢，CART 是二叉树，计算速度很快。
- 样本数据的差异：ID3 只能处理离散数据且缺失值敏感，C4.5 和 CART 可以处理连续性数据且有多种方式处理缺失值；从样本量考虑的话，小样本建议 C4.5、大样本建议 CART。
- ID3 和 C4.5 层级之间只使用一次特征，CART 可多次重复使用特征；
- 剪枝策略的差异：CART 采用后剪枝策略。



# 基于**scikit-learn**的模型实现

- 课后作业：基于sklearn的红酒分类
- `from sklearn.datasets import load_wine`
- 要求：
  - 按照7/3分训练和测试；
  - 调用DecisionTreeClassifier模型，使用ID3决策树算法；
  - 打印出准确率；
  - 从sklearn.metrics里导入classification\_report用于详细的分类性能报告；
  - 使用graphviz可视化构建的决策树。



大数据，成就未来



# Thank you!