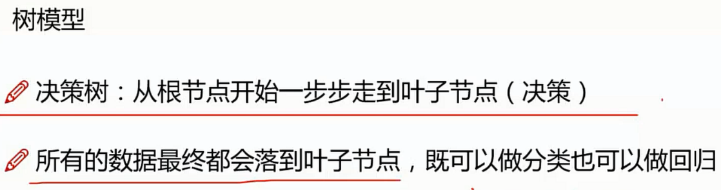
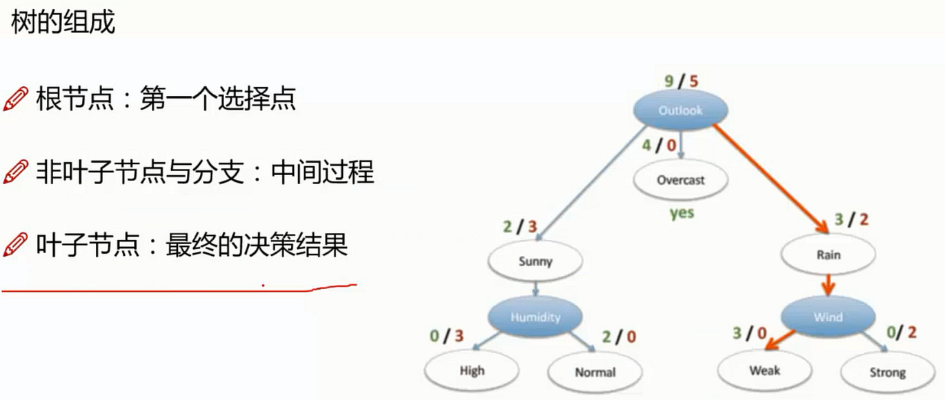
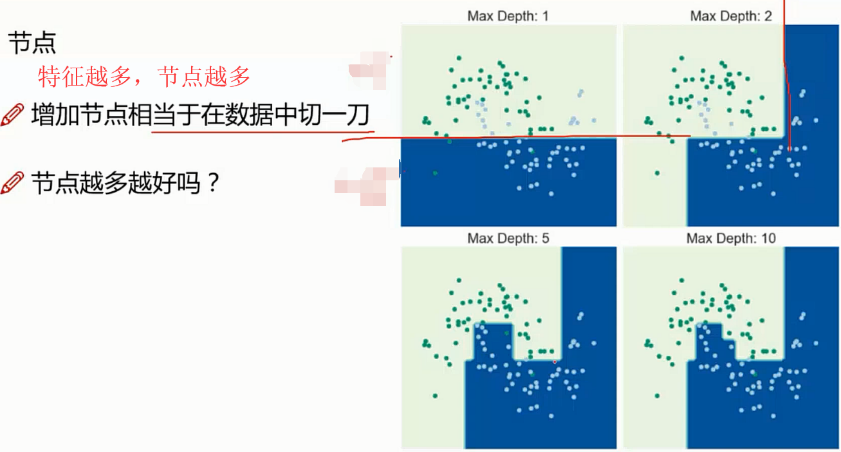
1. 决策树结构



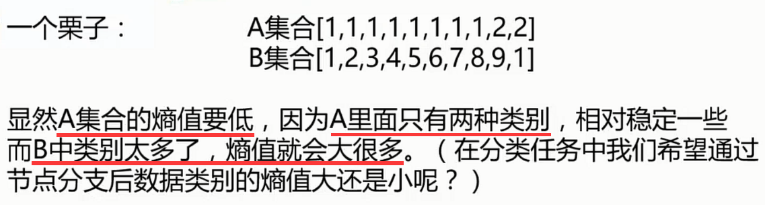


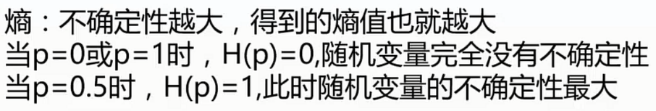


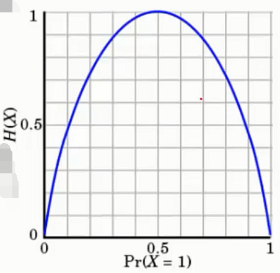
1. 节点的选择
2. **根节点选择**：必须选择分类效果最好的特征
3. **衡量标准：熵(定义：信息的期望值)**

**熵**：表示随机变量不确定性的度量。即**：数据混乱程度，数据差异越大，熵越大**。

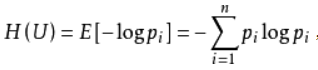
eg：



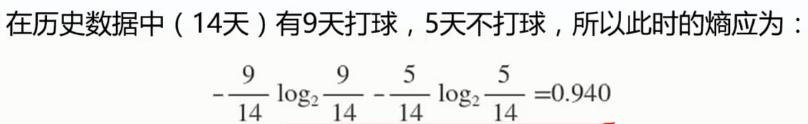




**公式：**（一般以2为底数，P(i)：为概率）



**eg：**



1. **信息增益：**特征X区分数据的能力。**信息增益越大**，该特征**区分**数据**能力越强**。

即：该特征的**分类效果越好**。

**1）信息增益** = 信息熵H(U)的变化大小

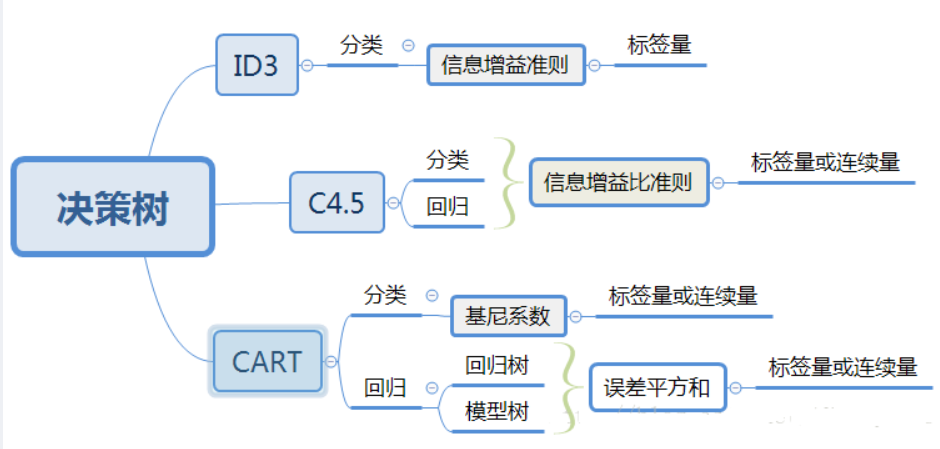
= 系统的熵值 – 某个特征下的系统熵值

**2）信息增益率** = 信息增益 / 自身熵值

**(可以用来，选择最优划分属性)**

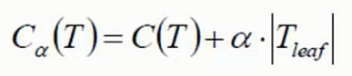
**注：**

1. **C4.5算法：**先找出信息增益高于平均水平的属性，再从中找出信息增益率最大的属性
2. **CART决策树算法：**用基尼指数来选择划分属性，基尼指数越小，数据纯度越高
3. **ID3：**使用信息增益划分属性
4. **三种决策树算法**



1. 决策树减枝策略：防止过拟合
2. **策略：预减枝**（边建立边减枝——常用）、**后剪枝**（先建立后减枝）
3. **预减枝**：限制深度、叶子节点数、叶子节点样本数、信息增益等

**后剪枝**：通过衡量标准C（叶子节点越多，损失越大，即：C越小越好）



**1）公式说明：**C(T) = 

Tleaf = 对应叶子节点的样本数T

**2）判断标准：**该节点分裂前、分裂后，C(T)值的大小

分裂前C(T)< 分裂后C(T)：不分裂

3. python下使用决策树

可以使用sklearn库来计算：

eg：（housing：sklearn里面的数据）

from sklearn.datasets.california\_housing import fetch\_california\_housing

from sklearn import tree

housing = fetch\_california\_housing()

dtr = tree.DecisionTreeRegressor(max\_depth = 2)

dtr.fit(housing.data[:, [6, 7]], housing.target)

#可视化，需要安装graphviz（使用anaconda就可以直接安装对应的库）

dot\_data = \

tree.export\_graphviz(

dtr,

out\_file = None,

feature\_names = housing.feature\_names[6:8],

filled = True,

impurity = False,

rounded = True

)

import pydotplus

graph = pydotplus.graph\_from\_dot\_data(dot\_data)

graph.get\_nodes()[7].set\_fillcolor("#FFF2DD")

from IPython.display import Image

Image(graph.create\_png())

#结果：

