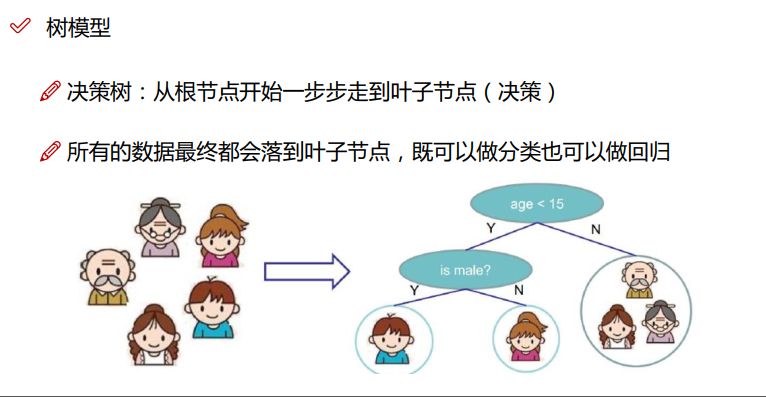
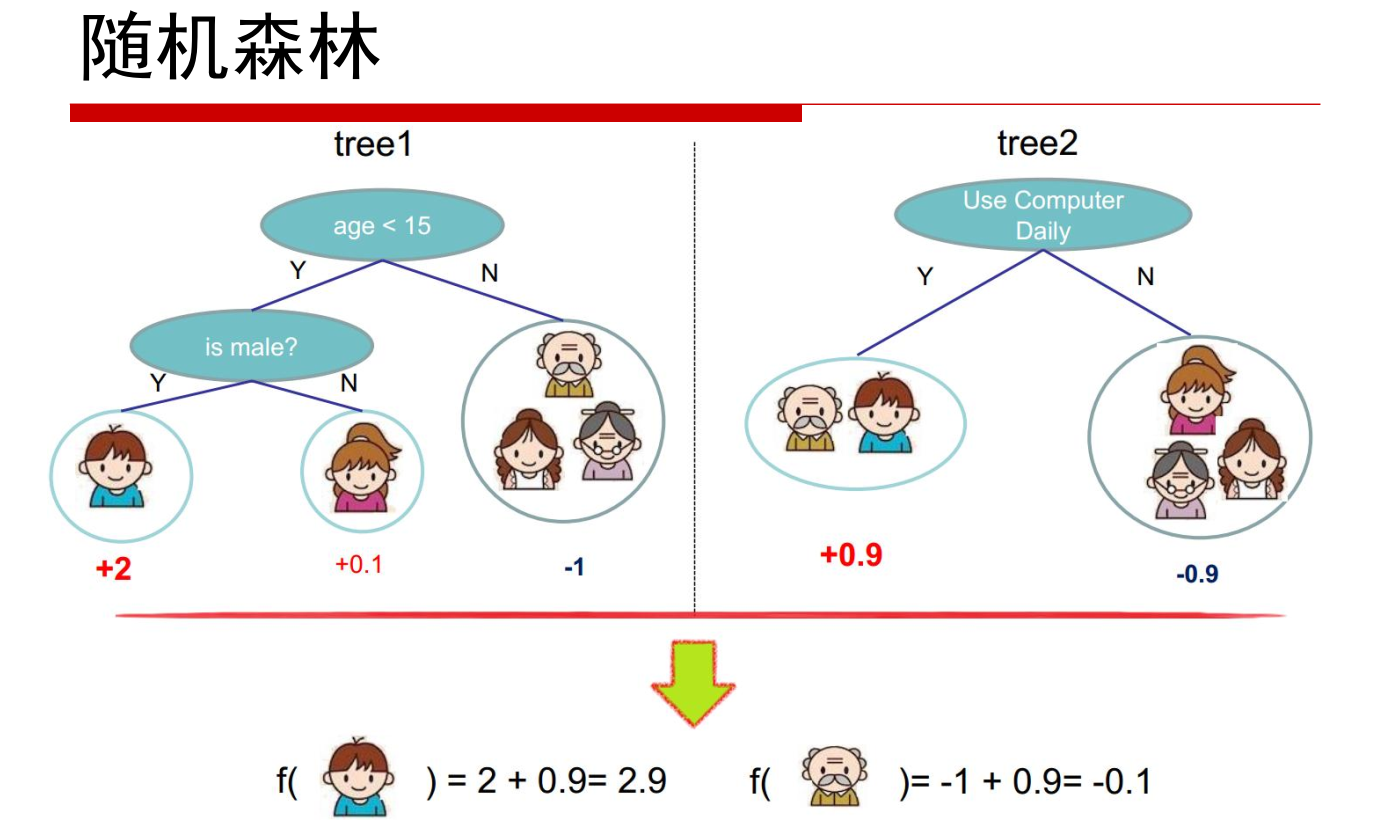
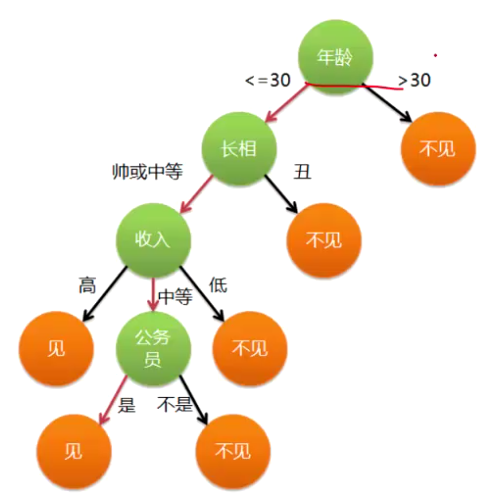
CART( 其意思就是分类and 回归)



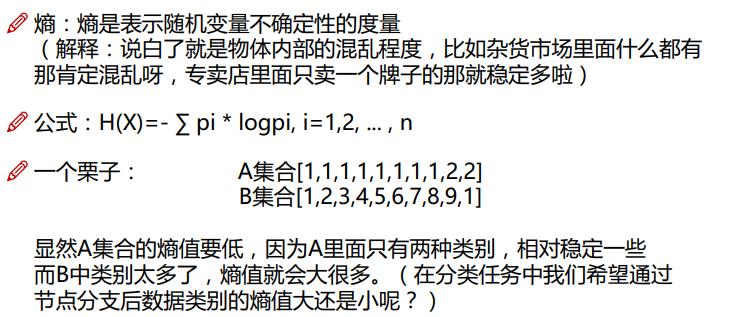
树模型即可做回归又可以做分类任务，在回归任务中，因为叶子节点中都是我们的样本，对于新的样本，通过决策树游走到某一个叶子节点，可以利用叶子节点每一个样本的权值，加权平均做回归任务，在分类任务中可以用该叶子节点中样本数最多的标签做为分类结果。



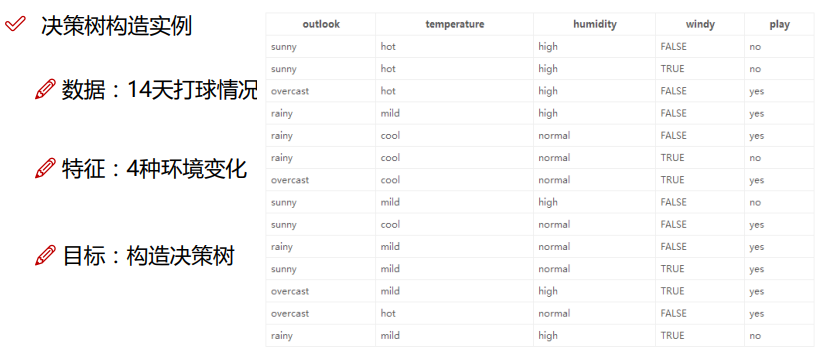
随机森林（RF）相对于决策树，相对于构造了多个决策树（DT），不同的决策树结构不同，最终的样本的分类或回归可以将多个树的决策结果相加得到结果



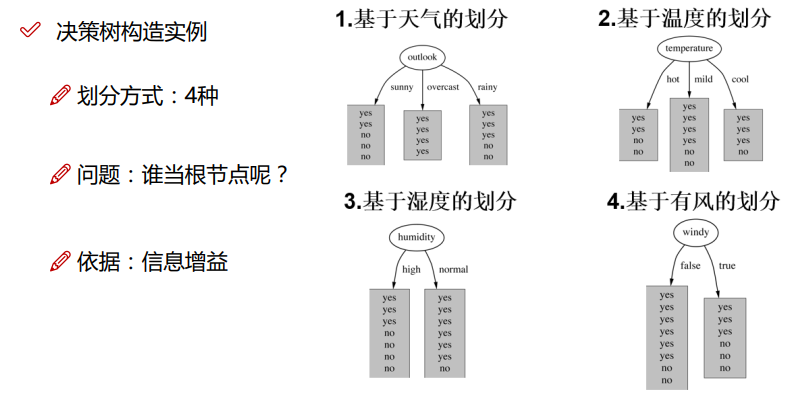
决策树的分裂过程：思考决策树生成过程，从父节点到若干子节点，我们按照正常的逻辑去解决这个问题，无非就是找到从若干个特征中找出一个特征，再从这个特征的分布中找出一个或多个切割点，不同的子树代表该特征不同的取值范围，不同的子树分得的样本有好有坏，假设一个二分类问题，一个好的决策树其叶子节点中的好坏样本比越高越好，因为样本的决策树在最终的预测中，效果会越好，那么怎么能让每次分裂的子节点的好坏样本比更高，这是我们追求的目标。决策树引入了一个衡量标准-熵，熵可以评价子树中样本的不确定性程度，对于A集合明显的我们确定1事件发生的概率很大，对于B我们不太确定哪个事件可能会发生，显然我们追求高确定性，那么熵越小越好

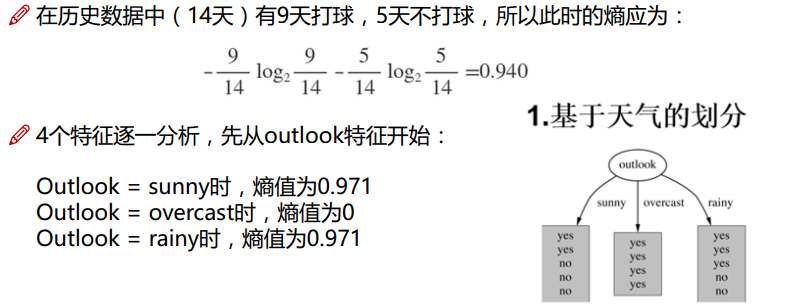


显然我们可以通过比较父节点和子树中样本的熵值来评估这次划分的好还，即信息增益，信息增益当然越大越好。p代表该事件发生的概率



这里特征全部是离散的且类别较少，对于连续特征需要进行离散化



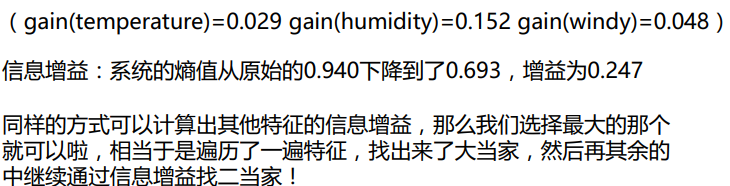


这里是根节点划分，故父节点的熵应该是所有样本的熵值

Eall = 0.94

子节点的样本的熵值分别是0.972、0、0.971，取加权平均即可求出子节点整体的信息熵





在根节点的分裂中发现，outlook划分之后的信息增益最大，故按照outlook进行划分。

评估标准：





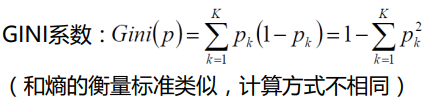
缺点：思考有这么一列特征，是连续特征，从1到n，如果不按照范围划分的话，每一个值都可以划分为一个子树，而且该子树中只有对应的一个样本，该层可以划分为n个子树，而且这次划分的信息熵是0，因为任何一个子树的样都只有一个，这样的划分其信息增益无疑相比于其他特征的划分都是最好的，那么算法就会这么划分，但是显然是不合理的，试想你的特征数据中混入了样本的ID数据，其结果就是这样的，还有一些连续特征比如年龄，如果不进行离散化，其结果也是这样的，这是信息增益所带来的问题



Hx = 信息增益 / 信息熵

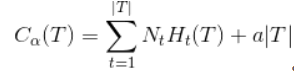
针对上述问题，虽然其信息增益很大，但是该列本身的信息熵非常大，用信息增益率来代替信息增益，这样可以解决这种问题





基尼系数用来表征序列的纯度，效果与熵类似

损失函数：



其中T代表叶子节点个数，Nt代表该叶子节点样本数，Ht代表该叶子节点的信息熵，最终再加一个正则项。

直观理解的话，使得损失函数最小，那么每一个叶子节点的信息熵应该尽可能的小，这样想是合理的，但是无法解释决策树损失函数的来源。

<https://blog.csdn.net/wzk1996/article/details/81941572>

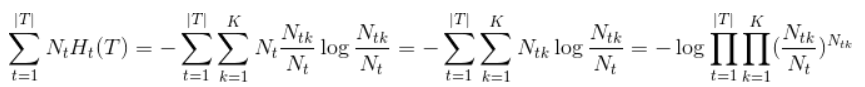
我们知道正则化的损失函数中前一项代表经验误差，而在概率模型中(决策树模型是一种概率模型)，经验误差函数的获得往往通过将极大似然函数取反，即将求极大化为求极小而获得。因此，在概率模型中，极大似然函数与经验误差函数可以认为是相同的概念，那么必然就可以通过经验误差函数来推导出极大似然函数，以此来加深对决策树损失函数的理解。

表面上来看：将每个叶节点的实例个数与其信息熵的乘积相加，这究竟代表个什么玩意呢？现在，我将利用该损失函数反向推导出极大似然函数，将信息熵Ht分解

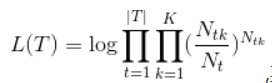
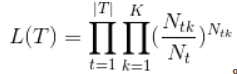


Pi = Ntk / Nt 该类别占比



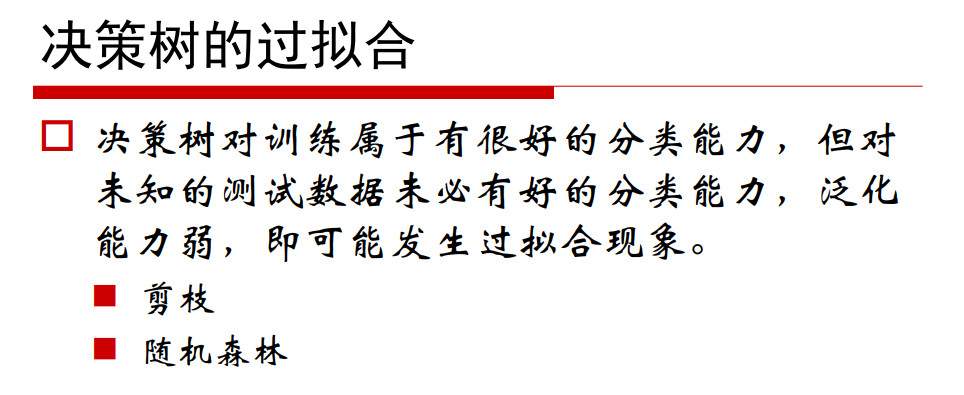


至此，损失函数被化简为这样，可以看到它与我们所要求的极大似然函数仅一步之遥：将求极小转为求极大，即去掉上式的负号，我们得到对数极大似然函数

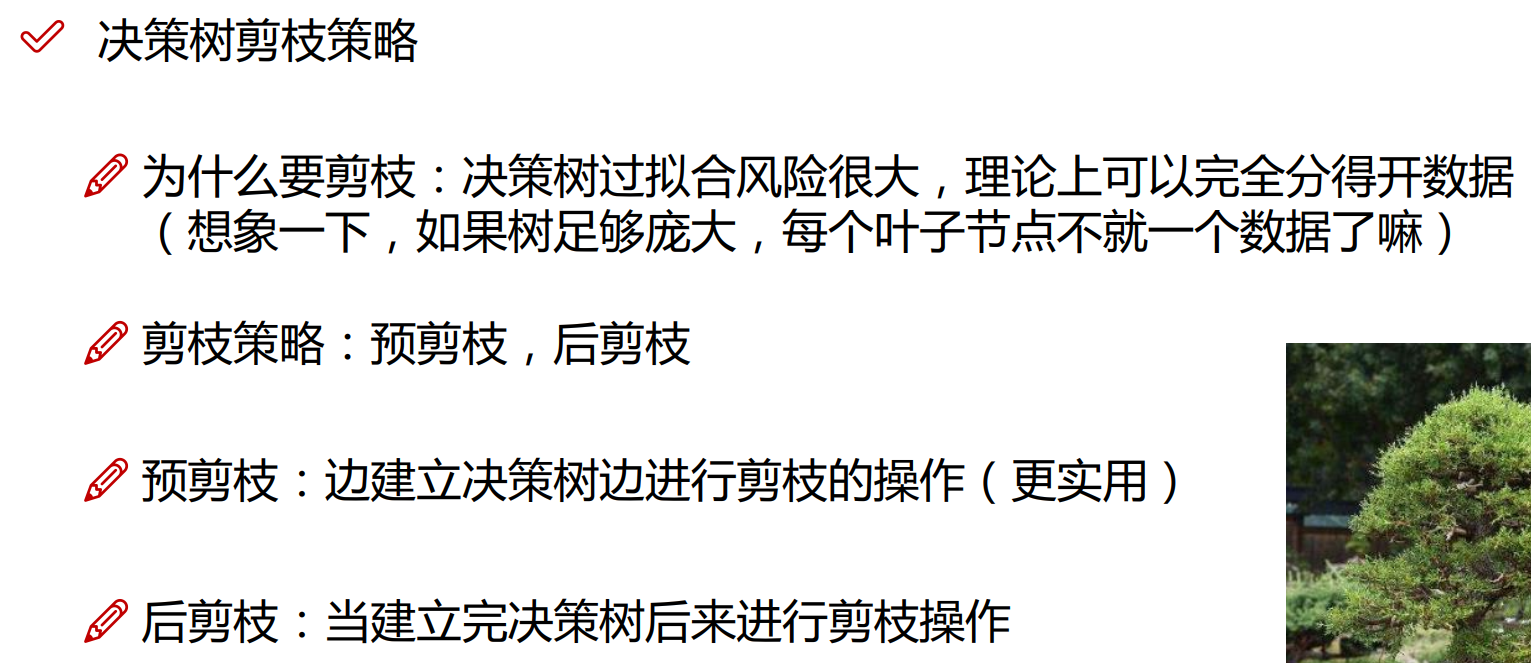
 ---🡪 

从极大似然的角度思考,

决策树的过拟合问题



决策树剪枝

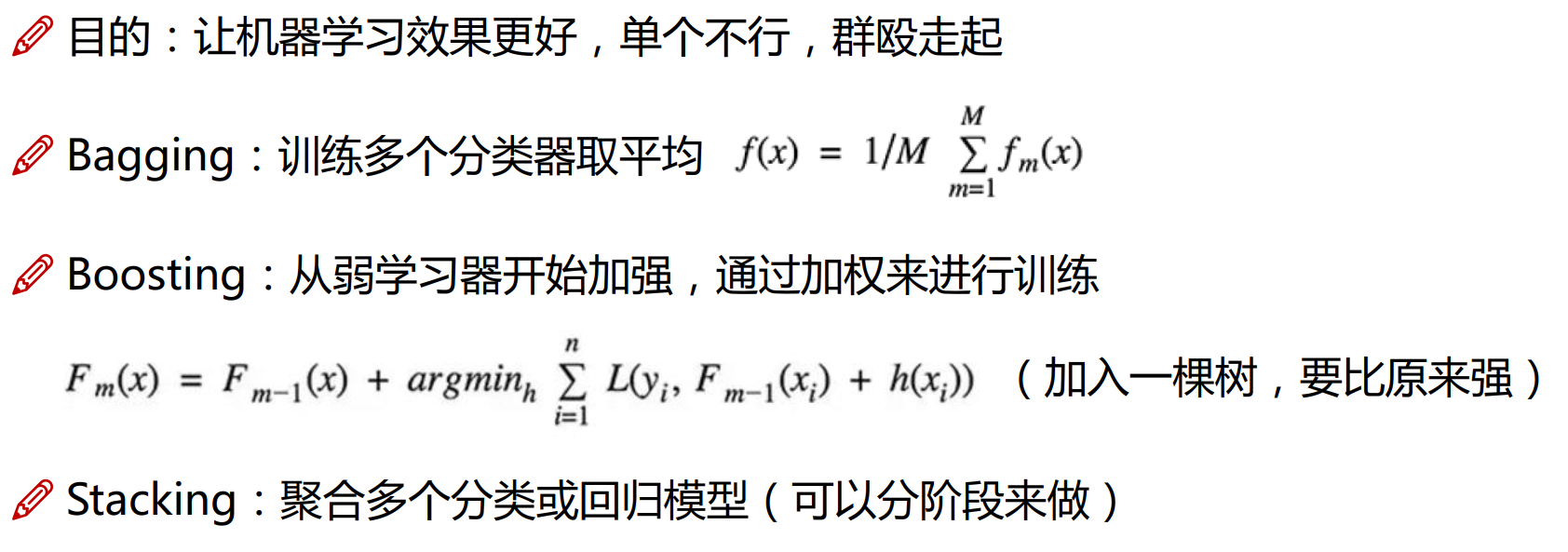


树的深度过深，叶子节点的样本数过少，都会导致决策树过拟合，剪枝是一种防止决策树过拟合的方式之一。

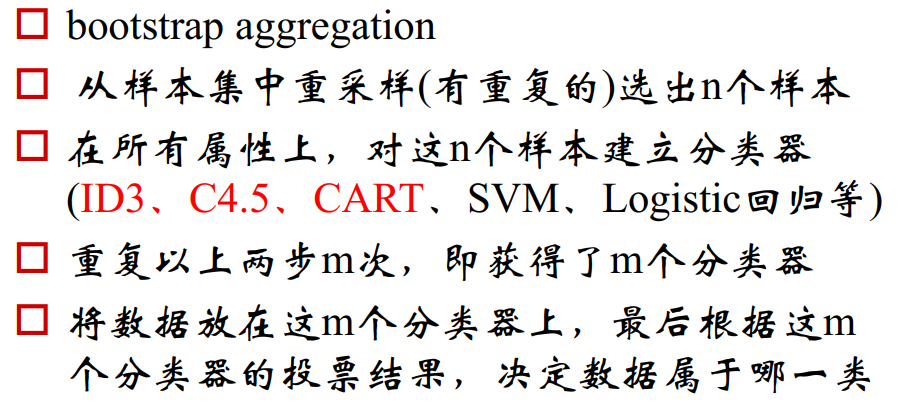
随机森林采用多个决策树投票的方式做决策，往往泛化能力更好。

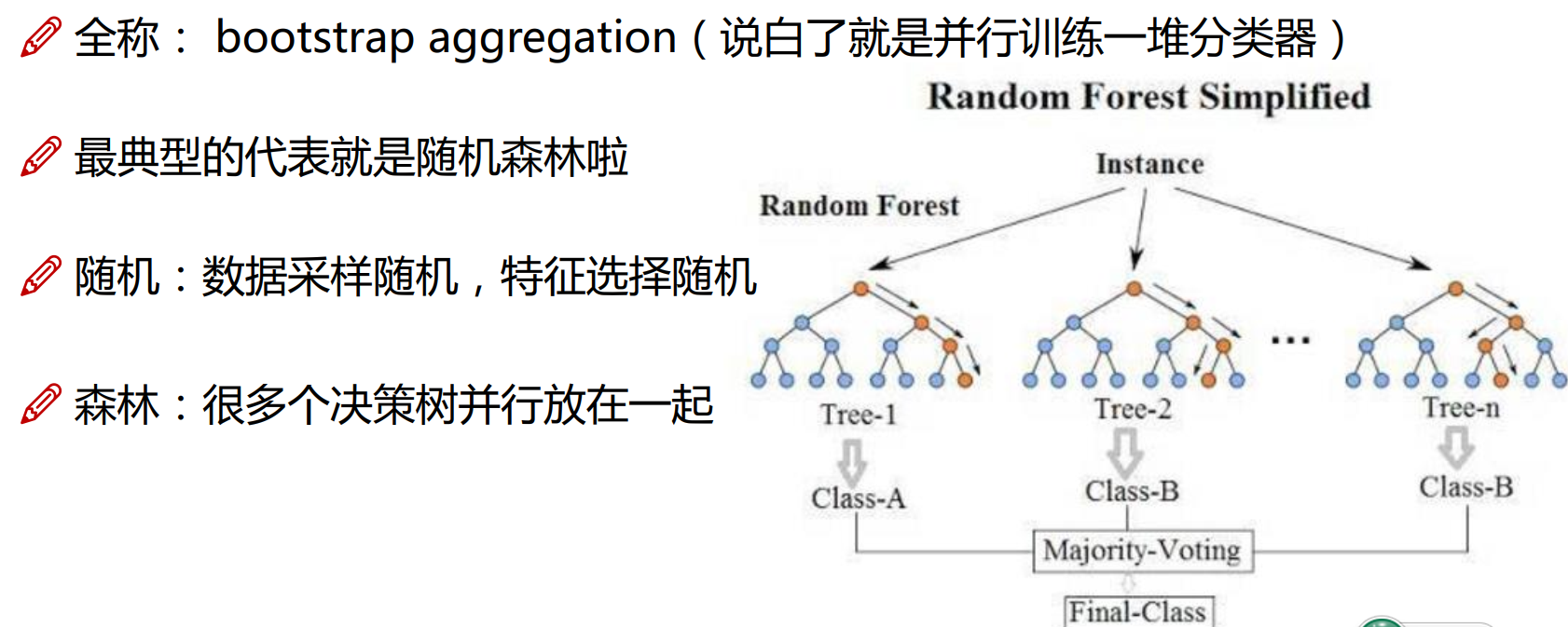
# 集成学习

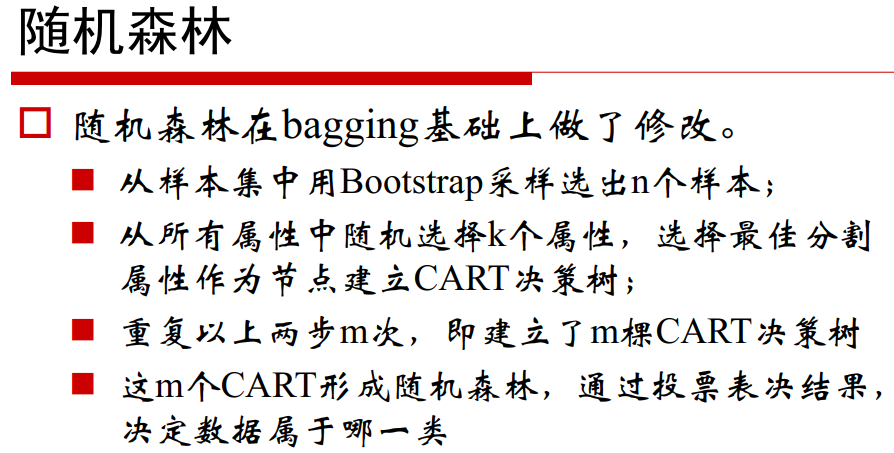
主要分为bagging、boosting、stacking三种算法



## Bagging的策略





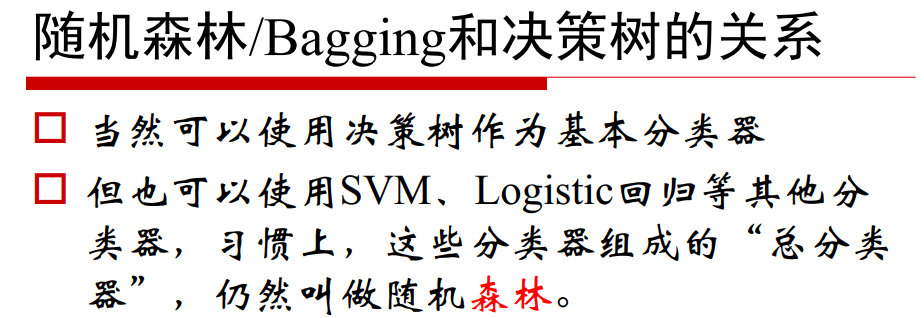


特点：每次从总样本中抽出n个来构造这个决策树，并不是所有样本，

每次从总特征中抽出k个来构造这个决策树，并不是所有特征

通过m次的构造出m课树，但是每个树的结构都不相同，这是由于有放回的样本抽取和有放回的特征抽取，这二重随机性保证了m个数之间的不同，试想如果每次构造的样本和特征一样的，那构造的树结构应该也是一样的，二重随机性使得随机森林不容易陷入过拟合，并且具有很好得抗噪能力（比如：对缺省值不敏感）

随机森林集成了所有的分类投票结果，将投票次数最多的类别指定为最终的输出，这就是一种最简单的 Bagging 思想。



**为什么要随机抽样训练集？**

如果不进行随机抽样，每棵树的训练集都一样，那么最终训练出的树分类结果也是完全一样的，这样的话完全没有bagging的必要

**为什么要有放回地抽样？**

如果不是有放回的抽样，那么每棵树的训练样本都是不同的，都是没有交集的，这样每棵树都是"有偏的"。

如果每个样本的特征维度为M，指定一个常数m<<M，随机地从M个特征中选取m个特征子集，每次树进行分裂时，从这m个特征中选择最优的，每棵树都尽最大程度的生长，并且没有剪枝过程。

**随机森林分类效果（错误率）与两个因素有关：**

* 森林中任意两棵树的相关性：相关性越大，错误率越大；
* 森林中每棵树的分类能力：每棵树的分类能力越强，整个森林的错误率越低。

减小特征选择个数m，树的相关性和分类能力也会相应的降低；增大m，两者也会随之增大。所以关键问题是如何选择最优的m（或者是范围），这也是随机森林唯一的一个参数。

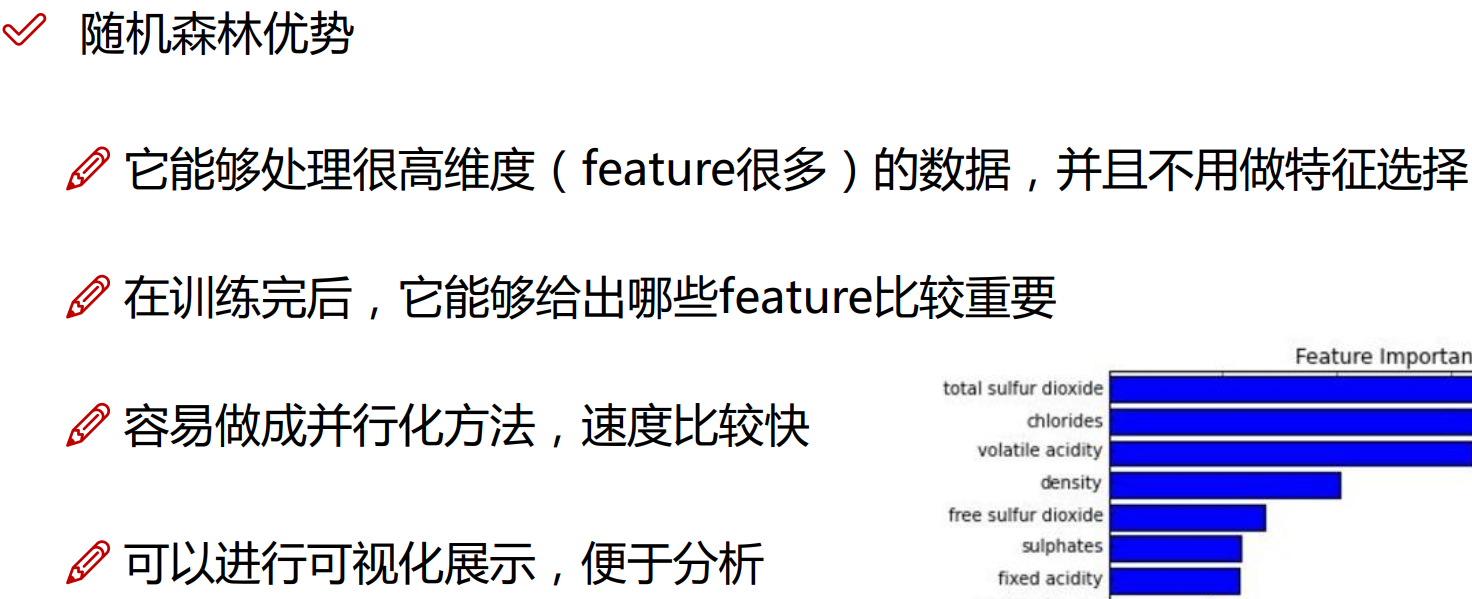
**袋外错误率（oob error）**

构建随机森林的关键问题就是如何选择最优的m，要解决这个问题主要依据计算袋外错误率oob error（out-of-bag error）。随机森林有一个重要的优点就是，没有必要对它进行交叉验证或者用一个独立的测试集来获得误差的一个无偏估计。它可以在内部进行评估，也就是说在生成的过程中就可以对误差建立一个无偏估计。我们知道，在构建每棵树时，我们对训练集使用了不同的bootstrap sample（随机且有放回地抽取）。所以对于每棵树而言（假设对于第k棵树），大约有1/3的训练实例没有参与第k棵树的生成，它们称为第k棵树的oob样本。

我们最终把oob样本汇集起来，让m个决策树给出每一个oob样本的分类结果，

袋外错误率= 有误样本个数 / oob样本数

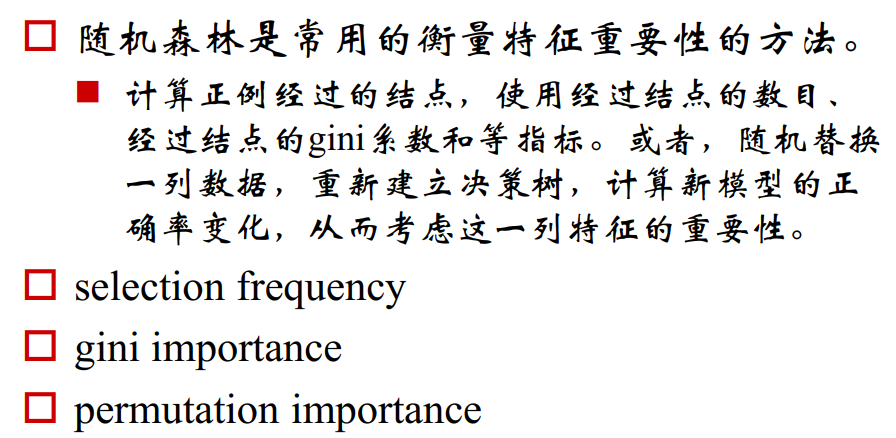
oob误分率是随机森林泛化误差的一个无偏估计，它的结果近似于需要大量计算的k折交叉验证。



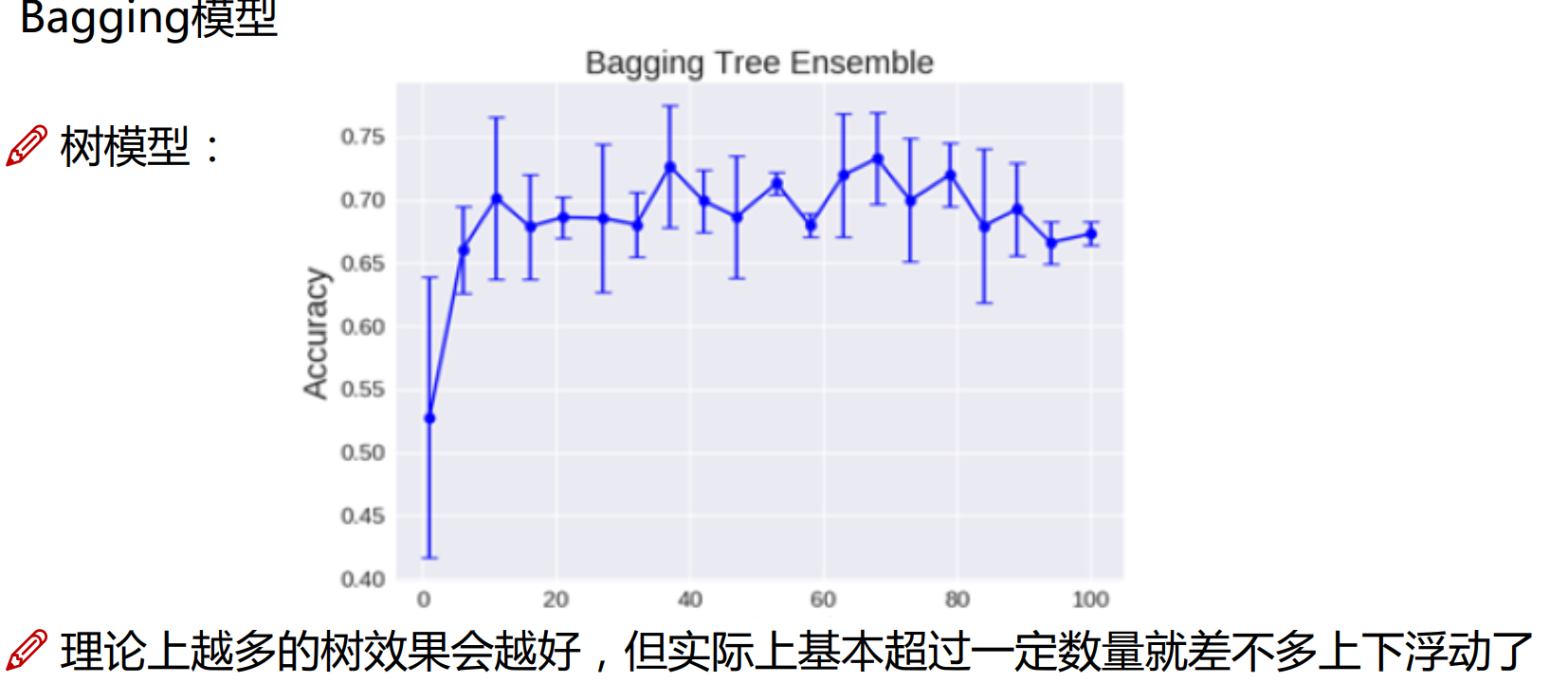
### RF评估特征重要性--基于袋外数据

### 对于一棵树 ，用OOB样本可以得到误差 e1，然后随机改变OOB中的第 j 个特征，保持其他特征不变，对第 j 个特征的样本值进行随机的上下置换，得到误差 e2。至此，可以用 e1-e2 来刻画特征 j 的重要性。其依据就是，如果特征j很重要，那么其变动后会非常影响测试误差，如果测试误差没有怎么改变，则说明特征j不重要。

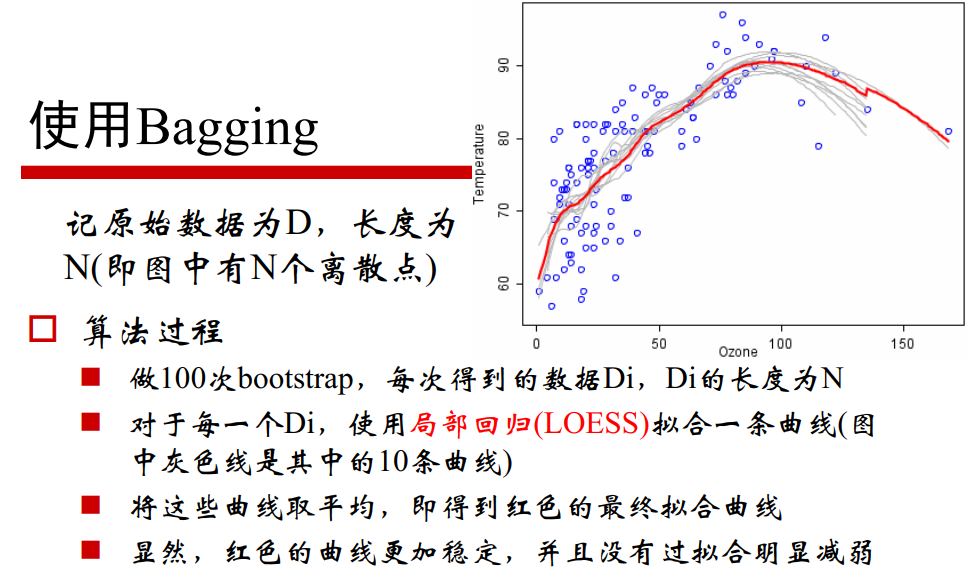
当然RF不止一种评估特征重要性的办法，其他的还有

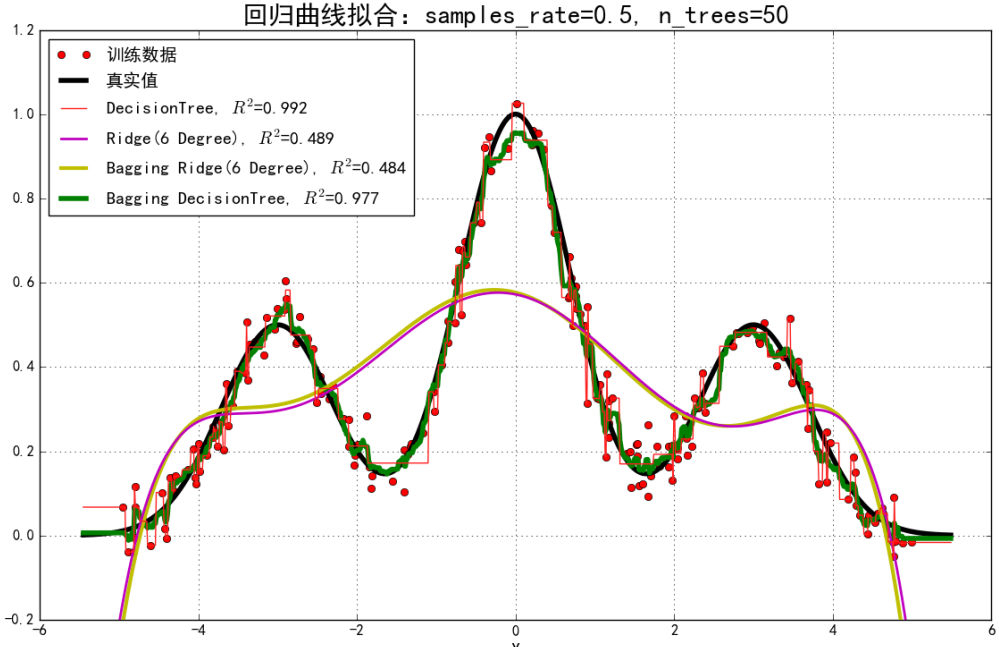


1. 该特征被决策树选用的频率。
2. 基尼重要性
3. 置换重要性，即基于袋外数据的误差变化情况做的评估

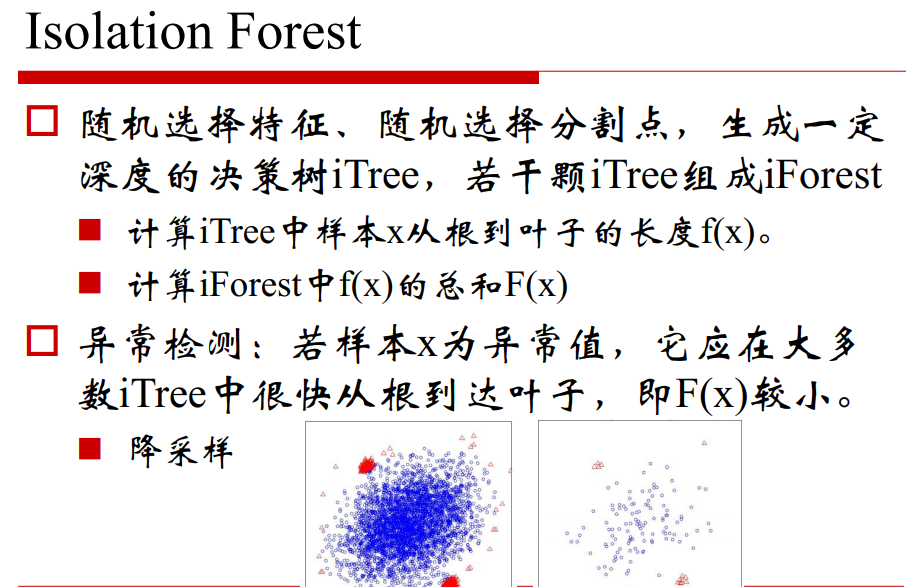








单决策树、岭回归、以岭回归为弱回归模型的bagging、随机森林，可以看出在该数据上随机森林效果最佳



孤立森林