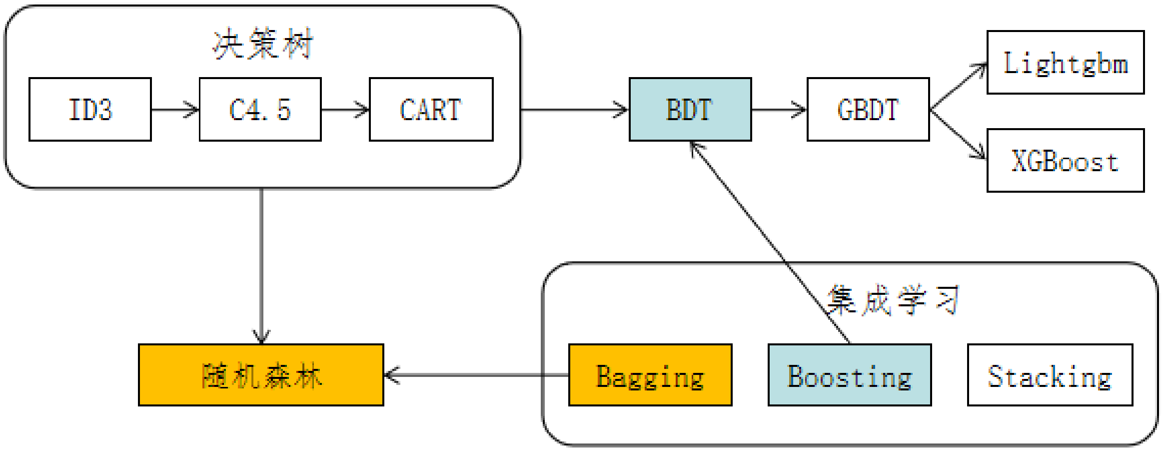
从DT到GBDT

1. 算法架构



名词解释：

- DT：decision tree，决策树
- BDT：boosting decision tree，集成决策树
- GBDT：gradient boosting decision tree，梯度提升决策树

2. 预备知识

2.1. 信息量

对某个事件发生概率的度量。一般情况下，概率越低，则事件包含的信息量越大。衡量事件信息量的公式如下：

$$I = log\frac{1}{p} = -logp \tag{1}$$

2.2. 熵（Entropy）

熵是随机变量不确定性的度量。

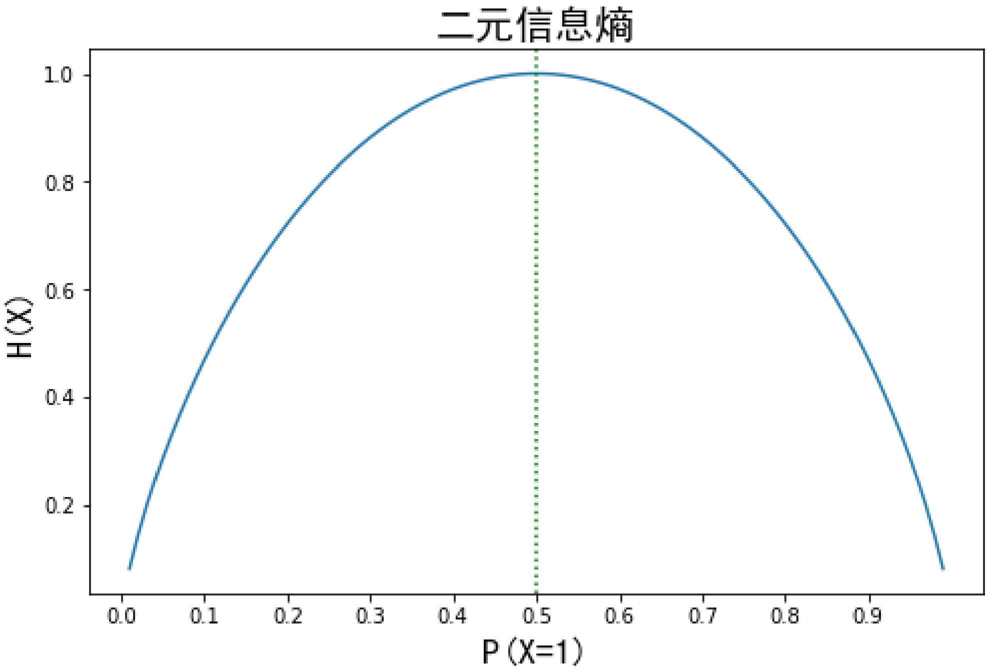
设X是一个取值个数为n的离散随机变量，其概率分布为：

$$P(X = x_i) = p_i, i = 1, 2, \ldots, n \tag{2}$$

则随机变量X的熵定义为：

$$H(X) = -\sum_{i=1}^{n} p_i * logp_i \tag{3}$$

熵越大，随机变量取值的不确定性越大，反之越小。当随机变量的分布为均匀分布时，该随机变量的熵最大。下图为二分类时熵与概率的变化曲线，可以看出，当P(X=1)=0.5时，H(X)最大。



2.3. 条件熵

表示在已知随机变量X的条件下随机变量Y的不确定性。

设随机变量（X,Y），其联合概率分布为：

$$P(X,Y) = p_{ij}(i = 1,2,\ldots,n; j = 1,2,\ldots,m) \tag{4}$$

给定随机变量X的条件下随机变量Y的条件熵为H(Y|X)，定义为X给定条件下Y的条件概率分布的熵对X的数学期望：

$$\begin{aligned} H(Y|X) &= \sum p_i * H(Y|X = x_i) \\ p_i &= P(X = x_i), i = 1,2,3,\ldots,n \end{aligned} \tag{5}$$

2.4. 信息增益

表示得知**特征X的信息**从而使得**类Y的信息的不确定性**减少的程度。

特征A对训练数据集D的信息增益$g(D,A)$，定义为集合D的经验熵H(D)与特征A给定条件下D的经验条件熵H(D|A)之差，即

$$g(D,A) = H(D) - H(D|A) \tag{6}$$

特征A的取值越多，$g(D,A)$越大，反之越小。

2.5. 信息增益比

特征A对训练数据集D的信息增益比$g_R(D,A)$定义为其**信息增益**$g(D,A)$与**训练数据集D关于特征A的值的熵**$H_A(D)$之比，即：

$$g_R(D,A) = \frac{g(D,A)}{H_A(D)} \tag{7}$$

$$\text{其中，} H_A(D) = -\sum_{i=1}^{n} \frac{|D_i|}{|D|} log_2 \frac{|D_i|}{|D|}\text{，} n\text{是特征}A\text{取值的个数} \tag{8}$$

特征A的取值越多，$g_R(D,A)$越小，反之越大。

2.6. 基尼指数

分类问题中，假设有K个类，样本点属于第k类的概率为p_k，则概率分布的基尼指数定义为：

$$Gini(p)=\sum_{k=1}^{K}p_k(1-p_k)=1-\sum_{k=1}^{K}p_k^2 \tag{9}$$

对于给定的样本集合D，其基尼指数为：

$$Gini(D)=1-\sum_{k=1}^{K}\left(\frac{|C_k|}{|D|}\right)^2 \\ \text{其中，}C_k\text{是}D\text{中属于第}k\text{类的样本子集，}K\text{是类的个数} \tag{10}$$

如果样本集合D根据特征A是否取某一可能值a被分割成D1和D2两部分，即：

$$D1=\{(x,y)\in D|A(x)=a\} \\ D2=D-D1 \tag{11}$$

则在特征A是否取a的条件下，集合D的基尼指数定义为：

$$Gini(D,A)=\frac{|D1|}{|D|}Gini(D_1)+\frac{|D2|}{|D|}Gini(D_2) \tag{12}$$

Gini(D)表示集合D的不确定性，Gini(D,A)表示将A=a分割后集合D的不确定性。

基尼指数越大，样本集合的不确定性（不纯度）越大，与熵类似。

下式表明基尼指数为熵的一阶泰勒近似值：

$$\begin{aligned} &\text{将}f(x)=-lnx\text{在}x=1\text{处进行一阶泰勒展开：} \\ f(x)&=f(x_0)+f^{'}(x-x_0)+o(\cdot) \\ &=f(1)+f^{'}(1)(x-1)+o(\cdot) \\ &\approx 1-x \end{aligned} \tag{13}$$

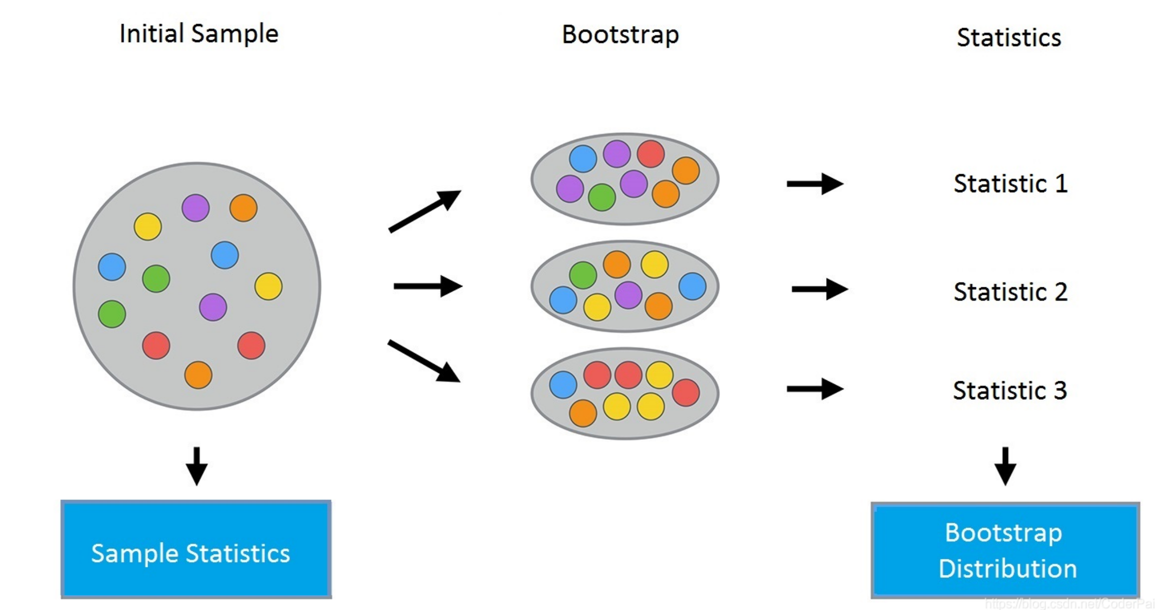
$$\begin{aligned} &\text{因此，随机变量}X\text{的熵近似于其概率分布的基尼指数：} \\ H(X)&=\sum_{i=1}^{n}p_i*(-logp_i) \\ &\approx\sum_{i=1}^{n}p_i*(1-p_i) \\ &=Gini(p) \end{aligned} \tag{14}$$

2.7. 自助法

自助法(bootstrapping)是一种数据采样方法，过程如下：

给定包含m个样本的数据集D，采用有放回的方法，每次从D中挑选一个样本放入$D^{'}$，执行m次后，生成一个包含m个样本的数据集$D^{'}$。根据计算，初始数据集D中约36.8%的样本未出现在数据集$D^{'}$中，因此，可以将$D^{'}$作为训练集，$D-D^{'}$用作测试集。

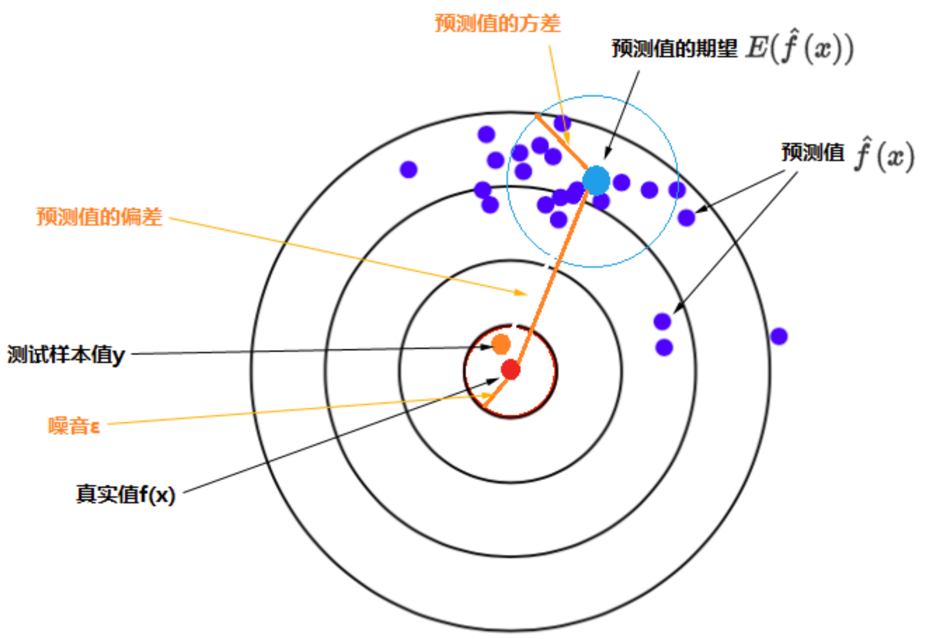
将上述过程重复N次，便可以得到N个采样后的样本集。



2.8. Bias-Variance Trade-off

符号	含义
x	测试样本
D	数据集
y	x的真实标记
y_D	x在数据集中的标记
$f(x;D)$	基于数据集D上学到的模型f在x上的**预测输出**
$\bar{f}(x)$	不同$f(x;D)$对x的期望预测输出
ε	数据集中标记的噪声，等于$y - y_D$

符号图形化展示如下：



- 期望预测输出

$$\bar{f}(x)=E[f(x;D)] \tag{15}$$

- 噪声

  数据采集、存储、加工等过程中引入的误差。为便于讨论，假设噪声ε的期望为0，所以：

$$\begin{aligned} E(\varepsilon) &= E(y-y_D)=0 \\ Var(\varepsilon) &= Var(y-y_D)^2 \\ &= E(y-y_D)^2-(E(y-y_D))^2 \\ &= E(y-y_D)^2 \end{aligned} \tag{16}$$

噪声用于刻画学习问题本身的难度。

- 方差

  使用**样本数相同**的**不同训练集**训练出的模型，对同一x预测输出值之间的差异程度：

$$Var(x)=E\left[(f(x;D)-\bar{f})^2\right] \tag{17}$$

方差用于刻画数据扰动对模型的影响，即不同训练数据集训练出的不同模型对同一个待预测x的预测结果的离散程度。

- 偏差

  期望输出（所有可能的训练数据集训练出的所有模型的输出的平均值）与**真实标记**的差别称为偏差（bias），即：

$$bias^2(x)=(\bar{f}-y)^2 \tag{18}$$

偏差用于刻画学习算法本身的拟合能力。

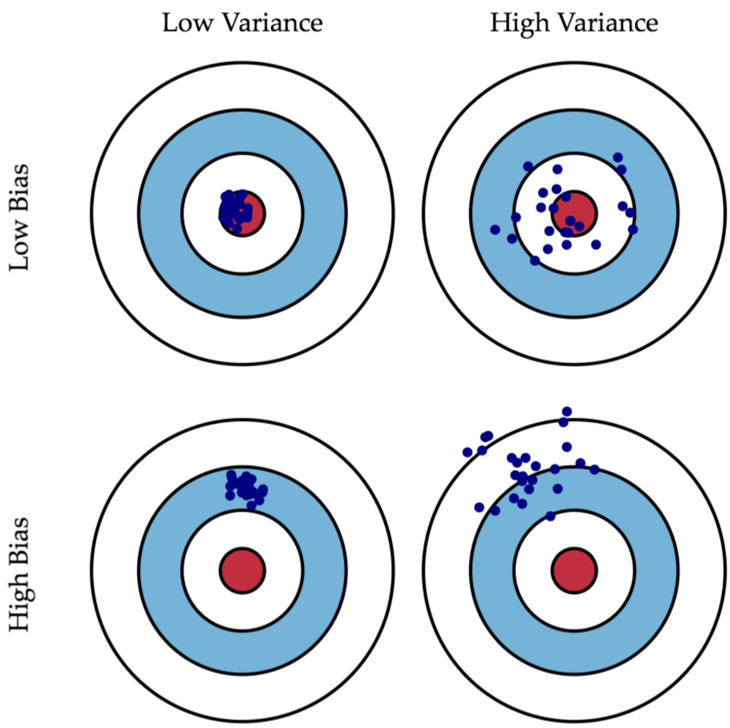
- 期望泛化误差

$$\begin{aligned} E(f;D) &= E\left[(f(x;D)-y_D)^2\right] \\ &= E\left[(f(x;D)-\bar{f}+\bar{f}-y_D)^2\right] \\ &= E\left[(f(x;D)-\bar{f})^2\right]+E\left[(\bar{f}-y_D)^2\right]+2*E\left[(f(x;D)-\bar{f})*(\bar{f}-y_D)\right] \\ &= Var(x)+E\left[(\bar{f}-y+y-y_D)^2\right] \\ &= Var(x)+E\left[(\bar{f}-y)^2\right]+E\left[(y-y_D)^2\right]+2*E\left[(\bar{f}-y)(y-y_D)\right] \\ &= Var(x)+(\bar{f}-y)^2+Var(\varepsilon) \\ &= Var(x)+bias^2(x)+Var(\varepsilon) \end{aligned} \tag{19}$$

偏差-方差分解说明，泛华性能由学习算法的能力、数据的充分性以及学习任务本身的难度共同决定。

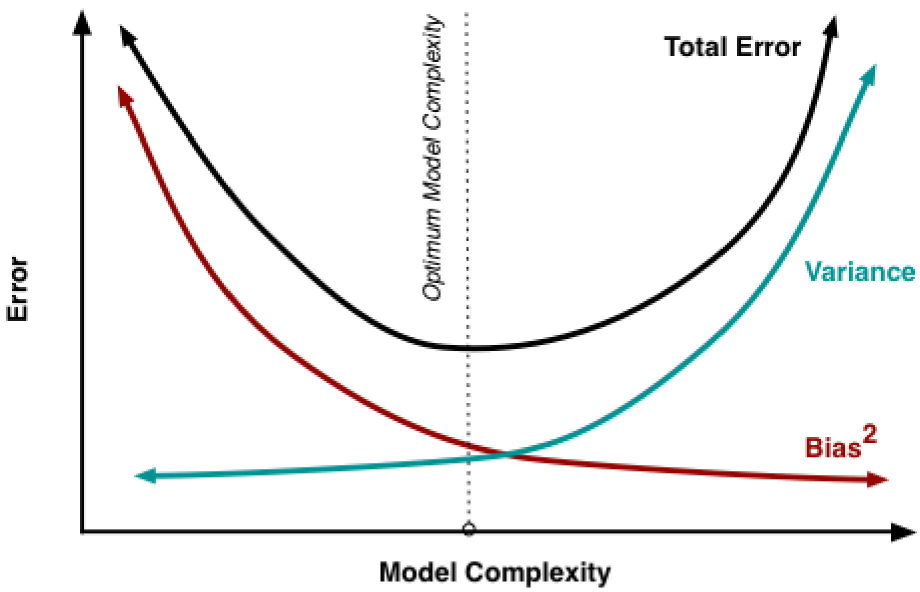
- 直观展示

  下图将机器学习任务描述为一个「打靶」的活动：根据相同算法、不同数据集训练出的模型，对同一个样本进行预测；每个模型作出的预测相当于是一次打靶。



- **左上角**：低偏差，低方差。如果有无穷的训练数据，以及完美的模型算法，有希望达成这样的情况。然而，现实中的工程问题，通常数据量是有限的，而模型也是不完美的。因此，这只是一个理想状况。
- **右上角**：低偏差，高方差。靶纸上的落点都集中分布在红心周围，它们的期望落在红心之内，因此偏差较小。另外一方面，落点虽然集中在红心周围，但是比较分散，这是方差大的表现。
- **左下角**：高偏差，低方差。靶纸上的落点非常集中，说明方差小。但是落点集中的位置距离红心很远，这是偏差大的表现。
- **右下角**：高偏差，高方差。最差的情况。

- 总结
  - 泛化误差由偏差、方差、噪声构成。
  - 模型训练的起始阶段，拟合效果差，偏差较大，数据集的变化对于模型的影响也很小，因此方差较小。此时模型表现为欠拟合。
  - 随着训练得深入，模型的拟合能力越来越强，偏差逐渐减小，方差逐渐增大。
  - 当模型训练到一定程度时，它的拟合能力非常强，这时所有样本都可以很好地被拟合，偏差很小，但是训练集细微的变化都会对模型的效果产生很大的影响，方差很大，将发生过拟合。

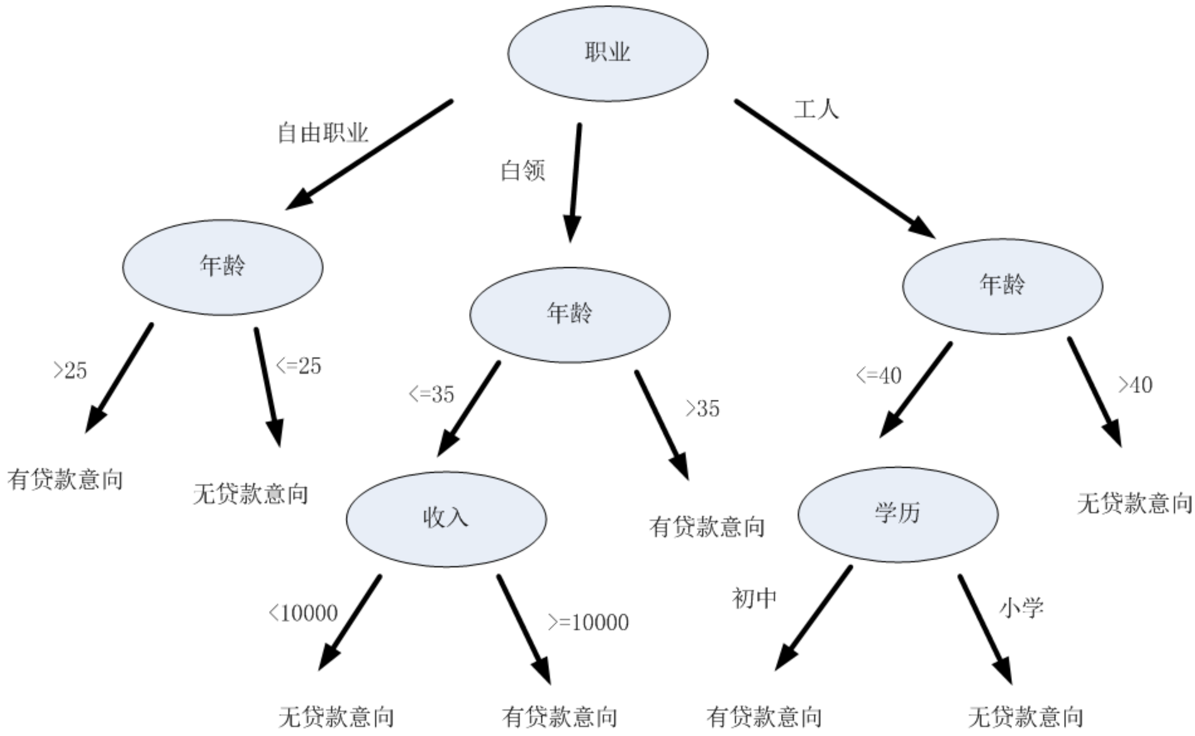


3. 决策树

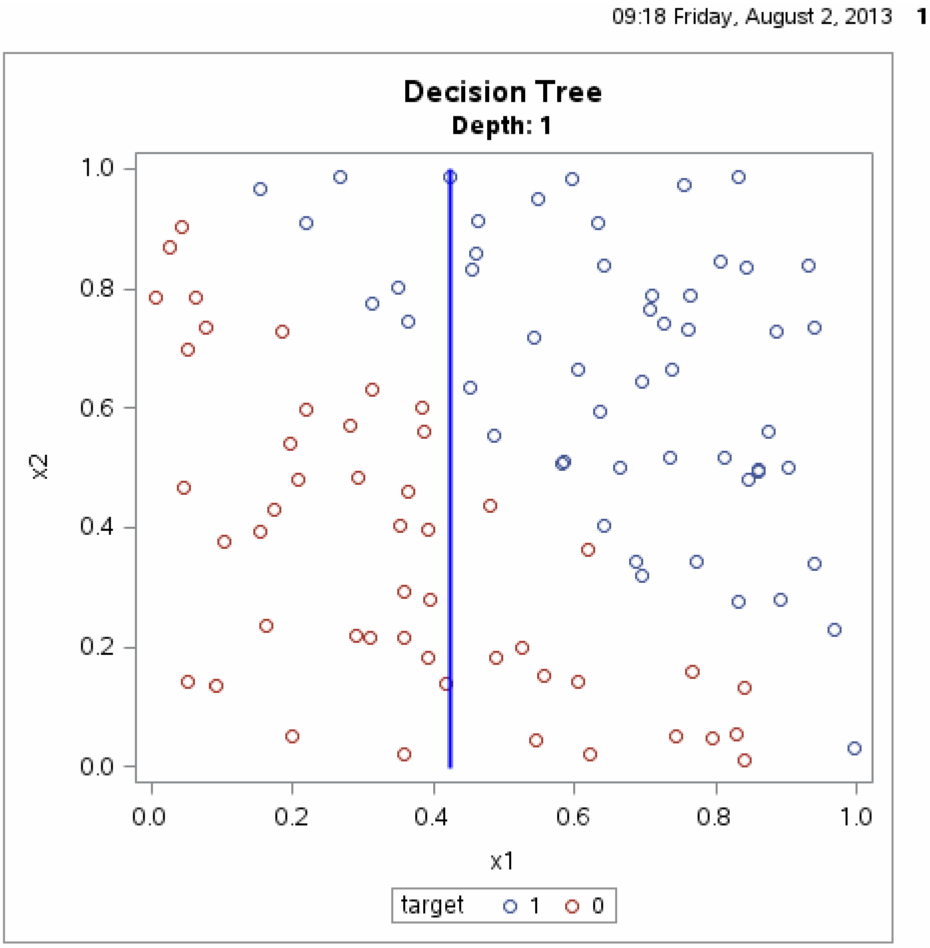
3.1. 概述

决策树是一种基本的分类与回归方法，它可以认为是一种if-then规则的集合。决策树由节点和有向边组成，内部节点代表特征，叶子节点代表类别。

下图为决策树的一个图例，判断用户是否有贷款意向：



决策树递归地选择特征并对整个特征空间进行划分，从而对样本进行分类，其过程如下所示：



从上图可以看出，决策树划分的方式有无数种，如何得到最优的决策树？即对训练数据有较好分类效果，同时对测试数据有较低的误差率。

根据特征选择依据的不同，决策树有三种生成算法，包括：ID3、C4.5、CART(Classification and Regression Tree)。

3.2. ID3

一、特征选择依据：选择信息增益最大的特征。

二、构建过程：

输入：训练数据集D，特征集A，阈值ε

输出：决策树T

主要过程：

1、计算A中各特征对D的信息增益，即G(D, A)，选择信息增益最大的特征A_g；

2、若A_g的信息增益小于ε，则置T为单节点树，并将D中实例数最大的类C_k作为该节点的类标记，返回T；

3、对A_g的每一可能值a_i，根据$A_g = a_i$将D分割为非空子集D_i，将D_i中实例数最大的类作为标记，构建子节点，由节点及其子节点构成树T，返回T；

4、对第i个子节点，以D_i为训练集，以$A - \{A_g\}$为特征集，递归地调用步骤1~3，得到子树T_i，返回T_i。

三、优点

1、构建决策树的速度比较快，算法实现简单，生成的规则容易理解。

四、缺点

1、倾向选择取值较多的特征。

2、只能处理离散特征，不能处理连续特征。

3、无修剪过程。

3.3. C4.5

一、特征选择依据：选择信息增益比最大的特征。

二、构建过程

输入：训练数据集D，特征集A，阈值ε

输出：决策树T

主要过程：

1、计算特征A中各特征对D的信息增益比，即$g_R(D,A)$，选择信息增益比最大的特征A_g；

2、若A_g的信息增益比小于ε，置T为单节点树，并将D中实例数最大的类C_k作为该节点的类标记，返回T；

3、对A_g的每一可能值a_i，根据$A_g=a_i$将D分割为非空子集D_i，将D_i中实例数最大的类作为标记，构建子节点，由节点及其子节点构成树T，返回T；

4、对第i个子节点，以D_i为训练集，以$A-\{A_g\}$为特征集，递归地调用步骤1~3，得到子树T_i，返回T_i。

三、优点

1、能够处理缺失值

a、计算信息增益比时缺失：忽略；将此属性出行频率最高的值赋予该样本。

b、按该属性创建分支时缺失：忽略；将此属性出行频率最高的值赋予该样本；为缺失值创建一个分支。

c、预测时，待分类样本的属性缺失：到达该属性时结束，将该属性所对应子树中概率最大的类别作为预

测类别；将此属性出行频率最高的值赋予该样本，然后继续预测。

2、能够处理离散值和连续值（按属性值排序，按二分法枚举两两属性值之间的阈值点进行离散化）。

3、构造树有后有剪枝操作，防止过拟合。

四、缺点

1、倾向选择取值较少的特征。

2、针对连续值特征，计算效率低。

3.4. CART

一、特征选择依据：选择基尼指数最小的特征。

二、构建过程 - 回归树

输入：训练数据集D

输出：回归树f(x)

主要过程：

1、遍历特征j，对特征j遍历其切分点s，按照下式，求解最优的(j,s)

$$\min_{j,s}\left[\min_{c_1}\sum_{x_i\in R_1(j,s))}(y_i-c_1)^2+\min_{c_1}\sum_{x_i\in R_2(j,s))}(y_i-c_2)^2\right] \tag{20}$$

2、用选定的(j,s)划分区域并决定相应的输出值：

$$\begin{aligned}
&R_1(j,s)=\left\{x|x^{(j)}\leqslant s\right\}\\
&R_2(j,s)=\left\{x|x^{(j)}>s\right\}\\
&\hat{c}_m=\frac{1}{N_m}\sum_{x_i\in R_m(j,s)}y_i, x\in R_m, m=1,2, N_m\text{为}R_m\text{元素个数}
\end{aligned} \tag{21}$$

3、继续对两个子区域调用步骤1~2，直到满足停止条件。

4、将输入空间划分为M个区域$R_1,R_2,\ldots,R_M$，生成决策树：

$$f(x)=\sum_{m=1}^{M}\hat{c}_m I(x\in R_m) \tag{22}$$

三、构建过程 - 分类树

输入：训练数据集D，停止计算的条件

输出：CART决策树

主要过程：

1、对训练数据集D，对每个特征A，对A可能取的每个值a，根据对$A=a$的测试为“是”或“否”，将D分成D_1和D_2两部分，计算$A=a$时的$Gini(D,A)$。

2、对所有可能的特征及其取值，选择基尼指数最小的特征及其取值，作为最优特征及最优切分点。根据最优特征及最优切分点，从现节点生成两个子节点，将训练数据集依特征分配到两个子节点中去。

3、继续对两个子区域调用步骤1~2，直到满足停止条件。

4、生成CART决策树。

3.5. 小结

算法	场景	树结构	特征选择	连续值	缺失值	剪枝
ID3	分类	多叉树	信息增益	不支持	不支持	不支持
C4.5	分类	多叉树	信息增益比	支持	支持	支持
CART	分类，回归	二叉树	基尼指数，MSE	支持	支持	支持

4. 集成学习

集成学习是通过训练若干个弱学习器，通过一定的组合策略，从而形成一个强学习器。按照基学习器之间是否存在依赖关系，可以分为两类：

- 基学习器之间不存在强依赖关系：基学习器可以并行生成，代表算法是bagging系列算法。
- 基学习器存在强依赖关系：基学习器需要串行生成，代表算法是boosting系列算法。

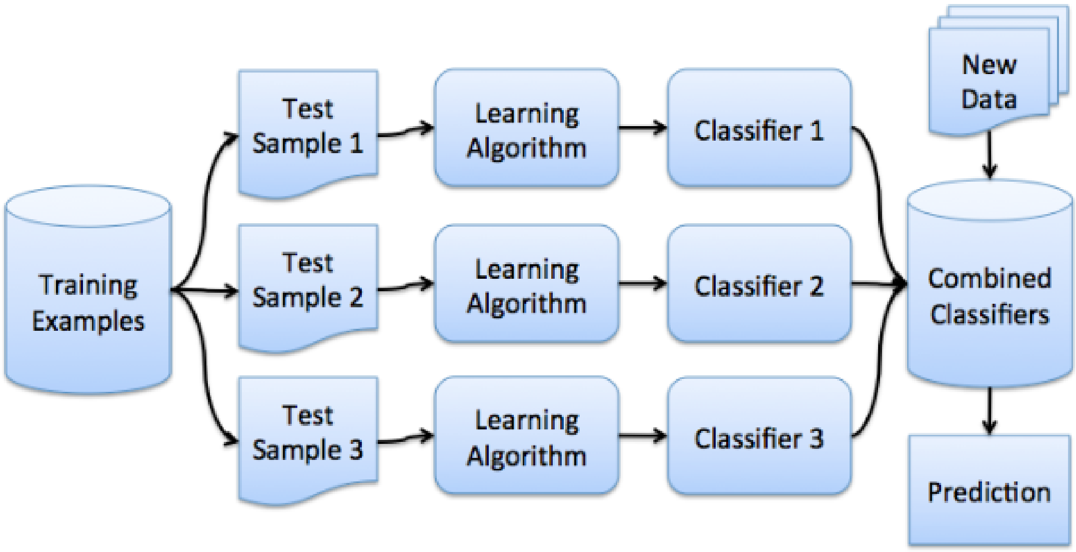
4.1. Bagging

Bagging(bootstrap aggregating)是并行式集成学习方法的代表。

以分类为例，假设训练集为X，利用自助法，可以生成N个样本集$X_1, X_2, \ldots X_N$，针对每个样本集训练一个单独的分类器$f_i(x)$，最终分类结果由N个分类器投票得出：

$$f(x) = sign\left(\frac{1}{N}\sum_{i=1}^{N} f_i(x)\right) \tag{23}$$

算法流程如下：



假设N个模型预测的结果分别为$Y_1, Y_2, \ldots, Y_N$，则组合后的预测结果为$Y = \frac{1}{N}\sum_{i=1}^{N} Y_i$。设单模型预测结果的期望是$\mu$，方差是$\sigma^2$。则bagging的期望预测为：

$$E(Y) = E(\frac{1}{N}\sum_{i=1}^{N} Y_i) = \frac{1}{N}E(\sum_{i=1}^{N} Y_i) = E(Y_i) \approx \mu \tag{24}$$

说明bagging模型预测的期望近似于单模型的期望，意味着bagging模型的bias与单模型的bias近似，所以bagging通常选择偏差低的强学习器。

bagging的抽样是有放回抽样，因此数据集之间会有重复的样本，模型的预测结果不独立。假设单模型之间具有一定相关性，相关系数为$0 < \rho < 1$，则bagging模型的方差为：

$$
\begin{aligned}
Var(Y) &= Var(\frac{1}{N}\sum_{i=1}^{N}Y_i) \\
&= \frac{1}{N^2}Var(\sum_{i=1}^{N}Y_i) \\
&= \frac{1}{N^2}Cov(\sum_{i=1}^{N}Y_i, \sum_{i=1}^{N}Y_i) \\
&= \frac{1}{N^2}\left(\sum_{i=1}^{N}Var(Y_i) + \sum_{i\neq j}^{N}Cov(Y_i, Y_j)\right) \\
&= \frac{1}{N^2}\left(N\sigma^2 + N(N-1)\rho\sigma^2\right) \\
&= \rho\sigma^2 + \frac{1-\rho}{N}\sigma^2
\end{aligned} \tag{25}
$$

所以，当N较大时，$Var(Y) \approx \rho\sigma^2$，bagging能降低整体预测结果的variance，而对bias优化有限。

4.2. Boosting

4.2.1. Adaboost

需要解决的两个问题：

- 每一轮如何改变训练数据的权值或概率分布？

  提高前一轮被错误分类的样本的权值，降低被正确分类的样本的权值。

- 如何将弱分类器组合成一个强分类器？

  加权多数表决。加大误差率低的分类器的权值，较小误差率高的分类器的权值。

4.2.1.1. 分类

输入：训练数据集$T=(x_1,y_1),(x_2,y_2),\dots,(x_N,y_N)$，其中$x_i \in X \subseteq R^N, y_i \in Y=\{-1,+1\}$；弱学习器算法。

输出：最终分类器$G(x)$。

算法步骤如下：

1. 初始化训练数据的权值分布（该值影响分类误差率）

   $D_1 = (w_{11},\dots,w_{1i},\dots,w_{1N})$
   $w_{1i} = \frac{1}{N}$
   $i = 1,2,\dots,N$

2. 对$m=1,2,\dots,M$

- 使用具有权值分布D_m的训练数据集学习，得到基本分类器：

$$G_m(x): X \rightarrow \{-1,+1\} \tag{26}$$

- 计算分类误差率

  计算$G_m(x)$在训练数据集上的分类误差率：

$$e_m = \sum_{i=1}^{N}P(G_m(x_i)\neq y_i) = \sum_{i=1}^{N}w_{mi}I(G_m(x_i)\neq y_i) \tag{27}$$

- 计算弱分类器$G_m(x)$的系数：

$$\alpha_m = \frac{1}{2}\log\frac{1-e_m}{e_m} \tag{28}$$

- 更新训练数据集的权值分布

$$D_{m+1} = (w_{m+1,1}, \dots, w_{m+1,i}, \dots, w_{m+1,N})$$
$$w_{m+1,i} = \frac{w_{mi}}{Z_m} e^{-\alpha_m y_i G_m(x_i)}, i = 1, 2, \dots, N \tag{29}$$

其中，Z_m是规范化因子：

$$Z_m = \sum_{i=1}^{N} w_{mi} e^{-\alpha_m y_i G_m(x_i)} \tag{30}$$

由上式可知，规范化后使得D_{m+1}成为一个概率分布。

3. 构建弱分类器的线性组合

$$f(x) = \sum_{m=1}^{M} \alpha_m G_m(x) \tag{31}$$

4. 最终分类器

$$G(x) = sign(f(x)) = sign\left(\sum_{m=1}^{M} \alpha_m G_m(x)\right) \tag{32}$$

4.2.1.2. 回归

输入：训练数据集$T = (x_1, y_1), (x_2, y_2), \dots, (x_N, y_N)$，其中$x_i \in X \subseteq R^N, y_i \in Y \subseteq R$；弱学习器算法。

输出：最终分类器$G(x)$。

算法步骤如下：

1. 初始化训练数据的权值分布（该值影响分类误差率）

$D_1 = (w_{11}, \dots, w_{1i}, \dots, w_{1N})$
$w_{1i} = \frac{1}{N}$
$i = 1, 2, \dots, N$

2. 对迭代次数$m = 1, 2, \dots, M$
   - 使用具有权值分布D_m的训练数据集学习，得到基本分类器：$G_m(x)$
   - 计算训练集上的最大误差

$$E_m = max|y_i - G_m(x_i)| \tag{33}$$

   - 计算单个样本的相对误差

$$\text{如果是线性误差，则} e_{mi} = \frac{|y_i - G_m(x_i)|}{E_m}$$
$$\text{如果是平方误差，则} e_{mi} = \frac{(y_i - G_m(x_i))^2}{E_m^2}$$
$$\text{如果是指数误差，则} e_{mi} = 1 - e^{\frac{-|y_i - G_m(x_i)|}{E_m}} \tag{34}$$

   - 计算弱分类器$G_m(x)$在训练数据集上的误差率：

$$e_m = \sum_{i=1}^{m} w_{mi} e_{mi} \tag{35}$$

- 计算弱分类器的系数

$$\alpha_m = \frac{e_m}{1-e_m} \tag{36}$$

- 更新训练数据集的权值分布

$$\begin{aligned} D_{m+1} &= (w_{m+1,1}, \ldots, w_{m+1,i}, \ldots, w_{m+1,N}) \\ w_{m+1,i} &= \frac{w_{mi}}{Z_m}\alpha_m^{1-e_{mi}}, i = 1, 2, \ldots, N \end{aligned} \tag{37}$$

其中，Z_m是规范化因子：

$$Z_m = \sum_{i=1}^{m} w_{mi}\alpha_m^{1-e_{m,i}} \tag{38}$$

由上式可知，规范化后使得D_{m+1}成为一个概率分布。

3. 构建弱分类器的线性组合，得到最终的强学习器

$$f(x) = \sum_{m=1}^{M}(\ln\frac{1}{\alpha_m})G_m(x) \tag{39}$$

4.2.2. BDT

BDT（boosting decision tree），提升决策树。

4.2.2.1. 分类

将Adaboost分类算法中的基本分类器限定为**二分类树模型**即可。

4.2.2.2. 回归

输入：训练数据集$T = (x_1, y_1), (x_2, y_2), \ldots, (x_N, y_N), x_i \in X \subseteq R^n, y_i \in Y \subseteq R$

输出：提升树$f_M(x)$，基模型为树模型

算法步骤：

1. 定义模型

$$f_M(x) = \sum_{m=1}^{M} T_m(x;\Theta_m) \tag{40}$$

$$T_m(x;\Theta_m) = \sum_{j=1}^{J_m} c_j I(x \in R_j)$$

其中，参数$\Theta_m = (R_1, c_1), (R_2, c_2), \ldots, (R_{J_m}, c_{J_m})$表示树的划分及各区域上的输出值。$J_m$为第$m$棵回归树的叶子节点数（也可认为是树的复杂度）。

2. 定义损失函数

损失函数使用MSE，因此，第m轮的损失为：

$$\begin{aligned} L[y, f_m(x)] &= L[y, f_{m-1}(x) + T_m(x;\Theta_m)] \\ &= [y - f_{m-1}(x) - T_m(x;\Theta_m)]^2 \\ &= [r - T_m(x;\Theta_m)]^2 \\ &\text{其中，} r = y - f_{m-1}(x) \text{是当前模型拟合数据的残差} \end{aligned} \tag{41}$$

因此，对回归提升树来讲，只需拟合当前模型的残差即可。

3. 对$m = 1, 2, \ldots, M$

- 计算残差：

$$r_{mi} = y_i - f_{m-1}(x), i = 1, 2, \ldots, N \tag{42}$$

- 生成第m棵回归树$T_m(x;\Theta_m)$
  根据$(x_1, r_{m1}), (x_2, r_{m2}), \ldots, (x_N, r_{mN})$学习一个回归树，得到$T_m(x;\Theta_m)$
- 更新模型

$$f_m(x) = f_{m-1}(x) + T(x;\Theta_m) \tag{43}$$

4. 生成最终的回归问题提升树

$$f_M(x) = \sum_{m=1}^{M} T_m(x;\Theta_m) \tag{44}$$

4.2.3. GBDT

GBDT（gradient boosting decision tree），梯度提升决策树。

4.2.3.1. 回归

以回归为例，解释GBDT原理。

- 解释一
  对回归问题，给定输入x_i，输出的预测值为y_i：

$$\hat{y_i} = F_M(x_i) = \sum_{m=1}^{M} h_m(x_i) \tag{45}$$

根据boosting算法：

$$F_m(x) = F_{m-1}(x) + h_m(x) \tag{46}$$

在给定已知模型F_{m-1}的情况下，对新加入的模型h_m，期望能使损失l_m降低，即：

$$h_m = \arg\min_h l_m = \arg\min_h \sum_{i=1}^{n} l(y_i, F_{m-1}(x_i) + h(x_i)) \tag{47}$$

根据泰勒展开公式：

$$f(x_0 + \Delta x) \approx f(x_0) + \Delta x * f^{'}(x_0) \tag{48}$$

对l_m一阶展开后可得：

$$\begin{aligned} l(y_i, F_{m-1}(x_i) + h_m(x_i)) &\approx l(y_i, F_{m-1}(x_i)) + h_m(x_i) * \left[\frac{\partial l(y_i, F(x_i))}{\partial F(x_i)}\right]_{F=F_{m-1}} \\ &= l(y_i, F_{m-1}(x_i)) + h_m(x_i) * g_i \\ \text{其中，} \quad g_i &= \left[\frac{\partial l(y_i, F(x_i))}{\partial F(x_i)}\right]_{F=F_{m-1}} \end{aligned} \tag{49}$$

由于$l(y_i, F_{m-1}(x_i))$为常数，为使得l_m最小，则h_m为：

$$\begin{aligned} h_m &\approx \arg\min_h \sum_{i=1}^{n} h(x_i) g_i \\ &= \arg\min_h [h(x_1), h(x_2), \ldots, h(x_n)] \cdot \begin{bmatrix} g_1 \\ g_2 \\ \ldots \\ g_n \end{bmatrix} \\ &= \arg\min_h \overrightarrow{h(x)} \cdot \vec{g} \end{aligned} \tag{50}$$

根据向量点乘定义，当$\overrightarrow{h(x)} = -\vec{g}$时，$\overrightarrow{h(x)} \cdot \vec{g}$取得最小值，即损失$l_m$取得最小值。因此：

在每轮迭代中，h_m用于拟合$-\vec{g}$，残差r_{mi}为：

$$r_{mi} = g_i = \left[\frac{\partial l(y_i, F(x_i))}{\partial F(x_i)} \right]_{F=F_{m-1}} \tag{51}$$

- 解释二

  从函数空间考虑，为了使得l_m下降，根据梯度下降原理：

  $$f_m = f_{m-1} - \frac{\partial l(y, f_{m-1})}{\partial f_{m-1}} \tag{52}$$

  而根据加法模型：

  $$f_m = f_{m-1} + h_m \tag{53}$$

  为了使得l_m下降，仅需使新加入的模型h_m逼近$-\frac{\partial l(y, f_{m-1})}{\partial f_{m-1}}$即可：

  $$h_m \to - \frac{\partial l(y, f_{m-1})}{\partial f_{m-1}} \tag{54}$$

4.2.3.2. 二分类

对二分类问题，可以使用0和1表示预测类别。预测值的区间为$(0, 1)$，预测结果为：

$$class = \begin{cases} 0 & \text{if } \hat{y} < \theta \\ 1 & \text{if } \hat{y} >= \theta \end{cases} \tag{55}$$

其中，θ为二分类判断阈值

- 定义模型

  对二分类问题，给定输入x_i，模型输出为：

  $$\hat{y}_i = \frac{1}{1 + e^{-F_M(x_i)}} \tag{56}$$

  其中，$F_M(x_i) = \sum_{m=1}^{M} h_m(x_i)$，$h_m(x)$为每次迭代的树模型

  模型初始化为：

  $$F_0(x) = h_0(x) = log \frac{\sum_{i=1}^{N} y_i}{\sum_{i=1}^{N} (1 - y_i)} \tag{57}$$

- 定义损失函数

  损失函数使用交叉熵损失，即：

  $$L(y_i, \hat{y}_i) = -y_i log \hat{y}_i - (1 - y_i) log(1 - \hat{y}_i) \tag{58}$$

- 求负梯度

  因为GBDT拟合的是损失函数关于模型的负梯度，求导可得：

  $$\begin{aligned} -\frac{\partial L(y_i, \hat{y}_i)}{\partial F(x_i)} &= -\frac{\partial \left\{ -y_i log \hat{y}_i - (1 - y_i) log(1 - \hat{y}_i) \right\}}{\partial F(x_i)} \\ &= -\left\{ -y_i \frac{1}{\hat{y}_i} \frac{\partial \hat{y}_i}{\partial F(x_i)} - (1 - y_i) \frac{-1}{1 - \hat{y}_i} \frac{\partial \hat{y}_i}{\partial F(x_i)} \right\} \\ &= -\left\{ -\frac{y_i}{\hat{y}_i} \hat{y}_i (1 - \hat{y}_i) - \frac{y_i - 1}{1 - \hat{y}_i} \hat{y}_i (1 - \hat{y}_i) \right\} \\ &= \left\{ (\frac{y_i}{\hat{y}_i} + \frac{y_i - 1}{1 - \hat{y}_i}) \hat{y}_i (1 - \hat{y}_i) \right\} \\ &= y_i - \hat{y}_i \end{aligned} \tag{59}$$

所以，在第m轮，$L(y_i,F(x_i))$关于$F(x_i)$的的负梯度值（伪残差）为：

$$r_{mi}=-\left[\frac{\partial\left\{-y_i log\hat{y}_i-(1-y_i)log(1-\hat{y}_i)\right\}}{\partial F(x_i)}\right]_{F(x)=F_{m-1}(x)}=y_i-\frac{1}{1+e^{-F_{m-1}(x_i)}} \tag{60}$$

- 迭代，对$m=1,2,\ldots,M$
  - 对$i=1,2,\ldots,N$

    计算负梯度，$r_{mi}=y_i-\frac{1}{1+e^{-F_{m-1}(x_i)}}$
  - 拟合回归树，生成第m棵回归树$h_m(x)$

    对(x_i,r_{mi})拟合一棵回归树，得到第m棵树的叶节点区域R_{mj}，$j=1,2,\ldots,J_m$，其中，J_m为第m棵回归树叶子节点的个数。
  - 对于J_m个叶子节点区域$j=1,2,\ldots,J_m$，计算出最佳拟合值

$$\begin{aligned}c_{mj}&=\underset{c}{argmin}\sum_{x_i\in R_{mj}}L(y_i,F_{m-1}(x_i)+c)\\&\approx\frac{\sum_{x_i\in R_{m,j}}r_{m,i}}{\sum_{x_i\in R_{m,j}}(y_i-r_{m,i})(1-y_i+r_{m,i})}\end{aligned} \tag{61}$$

    由于上式没有闭式解，所以采用近似值代替，代替过程如下：

$$\begin{aligned}G(c)&=\sum_{x_i\in R_{mj}}L(y_i,F_{m-1}(x_i)+c)\\&\approx\sum_{x_i\in R_{mj}}L(y_i,F_{m-1}(x_i))+\sum_{x_i\in R_{mj}}c\cdot\frac{\partial L(y_i,F_{m-1}(x_i))}{\partial F_{m-1}(x_i)}+\sum_{x_i\in R_{mj}}\frac{1}{2}\cdot c^2\cdot\frac{\partial^2L(y_i,F_{m-1}(x_i))}{\partial F_{m-1}^2(x_i)}\\\frac{dG}{dc}&=\sum_{x_i\in R_{mj}}\frac{\partial L(y_i,F_{m-1}(x_i))}{\partial F_{m-1}(x_i)}+\sum_{x_i\in R_{mj}}c\cdot\frac{\partial^2L(y_i,F_{m-1}(x_i))}{\partial F_{m-1}^2(x_i)}\\\text{令}\frac{dG}{dc}&=0\text{，此时的}c\text{使得}G(c)\text{最小，得：}\\c&=\frac{\sum_{x_i\in R_{mj}}\frac{\partial L(y_i,F_{m-1}(x_i))}{\partial F_{m-1}(x_i)}}{\sum_{x_i\in R_{mj}}\frac{\partial^2L(y_i,F_{m-1}(x_i))}{\partial F_{m-1}^2(x_i)}}\\&=\frac{\sum_{x_i\in R_{mj}}r_{m,i}}{\sum_{x_i\in R_{mj}}\hat{y}_i(1-\hat{y}_i)}\\&=\frac{\sum_{x_i\in R_{m,j}}r_{m,i}}{\sum_{x_i\in R_{m,j}}(y_i-r_{m,i})(1-y_i+r_{m,i})}\end{aligned} \tag{62}$$

  - 更新强学习器$F_m(x)$

$$F_m(x)=F_{m-1}(x)+\sum_{j=1}^{J_m}c_{m,j}I(x\in R_{m,j}) \tag{63}$$

- 生成最终强学习器$F_M(x)$

$$F_M(x)=F_0(x)+\sum_{m=1}^{M}\sum_{j=1}^{J_m}c_{m,j}I(x\in R_{m,j}) \tag{64}$$

- 结果预测

$$\begin{aligned}P(Y=1|x)&=\frac{1}{1+e^{-F_M(x)}}\\P(Y=0|x)&=1-P(Y=1|x)\end{aligned} \tag{65}$$

4.2.3.3. 多分类

假设有K个类别，对类别进行one-hot处理后，样本集形式为$(x, \underbrace{0,.,1,.,0}_{k})$，label中只有一个维度为1。

模型最终生成K棵集成决策树。

- 定义模型

  给定输入x，属于第k类的概率为：

$$P(y=k|x)=P_k=\frac{e^{F_k(x)}}{\sum_{i=1}^{K} e^{F_i(x)}} \tag{66}$$

$$\text{预测类别}=\underset{k}{argmax}P(y=k|x)=\underset{k}{argmax}\frac{e^{F_k(x)}}{\sum_{i=1}^{K} e^{F_i(x)}}$$

$$\text{其中，}F_k(x)\text{为第}k\text{棵集成树}$$

  对$q=1,2,...,K$，模型初始化为：

$$F_{0q}(x)=log\frac{\sum_{i=1}^{N} y_{iq}}{\sum_{i=1}^{N}(1-y_{iq})} \tag{67}$$

- 定义损失函数

$$L(y_i,\hat{P}_i)=-\sum_{q=1}^{K} y_q logP(y=q|x_i)=-\sum_{q=1}^{K} y_q log\frac{e^{F_q(x_i)}}{\sum_{j=1}^{K} e^{F_j(x_i)}} \tag{68}$$

- 求损失函数关于$F_q(x)$的负梯度

$$\begin{aligned}
-\frac{\partial L(y_q,\hat{P}_q)}{\partial F_q(x)} &= \frac{\partial[\sum_{i=1}^{K} y_i F_i(x)-\sum_{i=1}^{K} y_i\cdot log\sum_{j=1}^{K} e^{F_j(x)}]}{\partial F_q(x)} \\
&= y_q-\sum_{i=1}^{K} y_i\cdot\frac{e^{F_q(x)}}{\sum_{j=1}^{K} e^{F_j(x)}} \\
&= y_q-\hat{P}_q\cdot\sum_{i=1}^{K} y_i \\
&= y_q-\hat{P}_q
\end{aligned} \tag{69}$$

  因此，第m轮第i个样本对应的第q个类别的损失函数负梯度为：

$$\begin{aligned}
r_{miq} &= -\left[\frac{\partial L(y_{iq},F_q(x_i))}{\partial F_q(x_i)}\right]_{F_q(x)=F_{(m-1)q}(x)} \\
&= \left[y_{iq}-\frac{e^{F_q(x_i)}}{\sum_{j=1}^{K} e^{F_q(x_i)}}\right]_{F_q(x)=F_{(m-1)q}(x)}
\end{aligned} \tag{70}$$

- 迭代，对$m=1,2,..,M$
  - 对$i=1,2,..,N$，$q=1,2,..,K$
  - 计算损失函数的负梯度：r_{miq}
  - 对(x,r_{miq})拟合一个回归树，得到第m轮第q类第j个叶节点区域$R_{mqj}, j=1,2,...,J_{mq}$，其中，$J_{mq}$为第m棵第q类回归树叶子节点的个数。
  - 对于J_{mq}个叶子节点区域$j=1,2,...,J_{mq}$，计算出最佳拟合值

$$\begin{aligned}
c_{mqj} &= \underset{c}{argmin}\sum_{x_i\in R_{mqj}} L(y_{iq},F_{(m-1)q}(x_i)+c) \\
&\approx \frac{\sum_{x_i\in R_{mqj}} r_{miq}}{\sum_{x_i\in R_{mqj}}|r_{miq}|(1-|r_{miq}|)}\cdot\frac{K-1}{K}
\end{aligned} \tag{71}$$

- 更新强学习器$F_{mq}(x)$

$$F_{mq}(x)=F_{(m-1)q}(x)+\sum_{j=1}^{J_{mq}}c_{mqj}I(x\in R_{mqj}) \tag{72}$$

- 生成最终强学习器$F_{Mq}, q=1,2...,K$

$$F_{Mq}(x)=F_{0q}(x)+\sum_{m=1}^{M}\sum_{j=1}^{J_{mq}}c_{mqj}I(x\in R_{mqj}) \tag{73}$$

- 结果预测

$$\begin{aligned}P(y=k|x)&=\frac{e^{F_{Mk}(x)}}{\sum_{i=1}^{K}e^{F_{Mi}(x)}}\\ \text{预测类别}&=\underset{k}{argmax}\frac{e^{F_{Mk}(x)}}{\sum_{i=1}^{K}e^{F_{Mi}(x)}}\end{aligned} \tag{74}$$

4.3. 总结

集成方法	优点	缺点	示例
bagging	能处理过拟合； 能够降低variance； 学习器独立，可并行训练；	噪声大时会过拟合； 可能会有很多相似的决策树； 小数据或低维数据效果一般；	随机森林
boosting	能够降低bias和variance；	容易过拟合； 串行训练；	GBDT

5. Bagging实现

5.1. 随机森林

假设数据样本数为N，每个样本的属性个数为M，在每个决策树构造过程中，每个节点随机选择m个属性计算最佳分裂方式进行分裂。具体步骤如下：

1. 有放回地随机选择N个样本，用这N个样本来训练一棵决策树。
2. 每个样本有M个属性，在决策树中需要分裂节点时，从这M个属性中随机选取m个属性，一般来说m << M，然后从这m个属性中采用某种策略选择最佳属性作为当前节点的分裂属性。
3. 每棵决策树的每个节点的分裂都按照步骤（2）进行，直到不能分裂为止。
4. 重复建立K棵决策树，然后对预测结果进行一定组合，即可得随机森林模型。

6. Boosting-GBDT实现

6.1. XGBoost

6.1.1. 模型概述

$$
\begin{aligned}
XGBoost &= eXtreme + GBDT \\
&= eXtreme + (Gradient + BDT) \\
&= eXtreme + Gradient + (Boosting + DecisionTree)
\end{aligned} \tag{75}
$$

$$Boosting \rightarrow BDT \rightarrow GBDT \rightarrow XGBoost$$

6.1.2. 原理推导

输入：训练数据集$D = \{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \ldots, (\mathbf{x}_N, y_N)\}$，其中，$\mathbf{x}_i \in R^M, y_i \in R, |D| = N$

输出：提升决策树模型，由K个基本树模型组成

- 定义模型

$$\hat{y}_i = \phi(\mathbf{x}_i) = \sum_{k=1}^{K} f_k(\mathbf{x}_i) \tag{76}$$

其中，K为决策树个数，$f_k(\mathbf{x})$为第k棵决策树

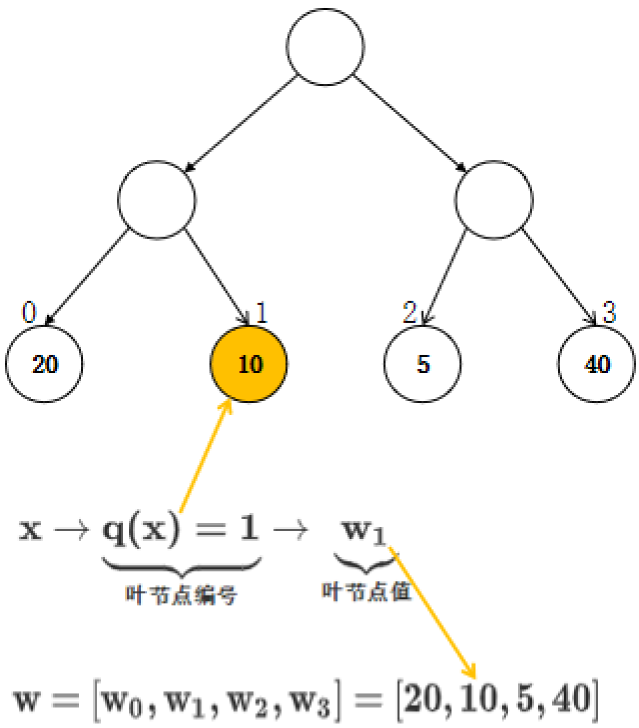
决策树定义为：

$$f(\mathbf{x}) = \mathbf{w}_{q(\mathbf{x})} \tag{77}$$

其中，$q(\mathbf{x})$为输入$\mathbf{x}$向**叶子节点编号**的映射，即$R^m \rightarrow \{1, \ldots, T\}$，$T$为决策树叶子节点数目。

$\mathbf{w} \in R^T$是叶子节点输出值向量，形式为$(w_1, w_2, \ldots, w_T)$。

决策树结构图如下所示：



- 定义损失函数

$$\mathcal{L}(\phi)=\sum_{i} l\left(\hat{y}_{i}, y_{i}\right)+\sum_{k} \Omega\left(f_{k}\right) \tag{78}$$

其中，$l\left(\hat{y}_{i}, y_{i}\right)$ 为经验风险

$\Omega(f)$ 为结构风险，$\Omega(f)=\gamma T+\frac{1}{2} \lambda\|w\|^{2}=\gamma T+\frac{1}{2} \lambda \sum_{j=1}^{T} w_{j}^{2}$

第 t 轮损失函数：

$$\mathcal{L}^{(t)}=\sum_{i=1}^{N} l\left(y_{i}, \hat{y}_{i}^{(t-1)}+f_{t}\left(\mathbf{x}_{i}\right)\right)+\Omega\left(f_{t}\right) \tag{79}$$

第 t 轮损失函数$\mathcal{L}^{(t)}$在$\hat{y}^{(t-1)}$处的二阶泰勒展开为：

$$\begin{aligned}
\mathcal{L}^{(t)} & \simeq \sum_{i=1}^{n}\left[l\left(y_{i}, \hat{y}^{(t-1)}\right)+\partial_{\hat{y}^{(t-1)}} l\left(y_{i}, \hat{y}^{(t-1)}\right) f_{t}\left(\mathbf{x}_{i}\right)+\frac{1}{2} \partial_{\hat{y}^{(t-1)}}^{2} l\left(y_{i}, \hat{y}^{(t-1)}\right) f_{t}^{2}\left(\mathbf{x}_{i}\right)\right]+\Omega\left(f_{t}\right) \\
& =\sum_{i=1}^{n}\left[l\left(y_{i}, \hat{y}^{(t-1)}\right)+g_{i} f_{t}\left(\mathbf{x}_{i}\right)+\frac{1}{2} h_{i} f_{t}^{2}\left(\mathbf{x}_{i}\right)\right]+\Omega\left(f_{t}\right)
\end{aligned} \tag{80}$$

其中，$g_{i}=\partial_{\hat{y}^{(t-1)}} l\left(y_{i}, \hat{y}^{(t-1)}\right), h_{i}=\partial_{\hat{y}^{(t-1)}}^{2} l\left(y_{i}, \hat{y}^{(t-1)}\right)$

第 t 轮目标函数$\mathcal{L}^{(t)}$的二阶泰勒展开移除常数项：

$$\begin{aligned}
\tilde{\mathcal{L}}^{(t)} & =\sum_{i=1}^{N}\left[g_{i} f_{t}\left(\mathbf{x}_{i}\right)+\frac{1}{2} h_{i} f_{t}^{2}\left(\mathbf{x}_{i}\right)\right]+\Omega\left(f_{t}\right) \\
& =\sum_{i=1}^{N}\left[g_{i} f_{t}\left(\mathbf{x}_{i}\right)+\frac{1}{2} h_{i} f_{t}^{2}\left(\mathbf{x}_{i}\right)\right]+\gamma T+\frac{1}{2} \lambda \sum_{j=1}^{T} w_{j}^{2}
\end{aligned} \tag{81}$$

其中，T为叶子节点数

定义叶结点 j 上的 **样本的下标集合** $I_{j}=\left\{i \mid q\left(\mathbf{x}_{i}\right)=j\right\}$，则目标函数可表示为 **按叶结点** 累加的形式：

$$\begin{aligned}
\tilde{\mathcal{L}}^{(t)} & =\sum_{i=1}^{n}\left[g_{i} f_{t}\left(\mathbf{x}_{i}\right)+\frac{1}{2} h_{i} f_{t}^{2}\left(\mathbf{x}_{i}\right)\right]+\gamma T+\frac{1}{2} \lambda \sum_{j=1}^{T} w_{j}^{2} \\
& =\sum_{j=1}^{T}\left[\left(\sum_{i \in I_{j}} g_{i}\right) w_{j}+\frac{1}{2}\left(\sum_{i \in I_{j}} h_{i}\right) w_{j}^{2}\right]+\frac{1}{2} \lambda \sum_{j=1}^{T} w_{j}^{2}+\gamma T \\
& =\sum_{j=1}^{T}\left[\left(\sum_{i \in I_{j}} g_{i}\right) w_{j}+\frac{1}{2}\left(\sum_{i \in I_{j}} h_{i}+\lambda\right) w_{j}^{2}\right]+\gamma T
\end{aligned} \tag{82}$$

由于：

$$w_{j}^{*}=\underset{w_{j}}{\arg \min } \tilde{\mathcal{L}}^{(t)} \tag{83}$$

可令：

$$\frac{\partial \tilde{\mathcal{L}}^{(t)}}{\partial w_{j}}=0 \tag{84}$$

得到每个叶结点 j 的最优输出值为：

$$w_{j}^{*}=-\frac{\sum_{i \in I_{j}} g_{i}}{\sum_{i \in I_{j}} h_{i}+\lambda} \tag{85}$$

代入每个叶结点 j 的输出值，得到第 t 个决策树损失函数的最小值：

$$\tilde{\mathcal{L}}^{(t)}(q) = -\frac{1}{2}\sum_{j=1}^{T}\frac{\left(\sum_{i\in I_j} g_i\right)^2}{\sum_{i\in I_j} h_i+\lambda}+\gamma T \tag{86}$$

其中，T为第t轮决策树的叶节点数

- 节点分裂准则

假设I_L和I_R分别为分裂后左右结点的实例集，令$I = I_L \cup I_R$，则分裂后损失减少量由下式得出：

$$\begin{aligned}\mathcal{L}_{split} &= \tilde{\mathcal{L}}_I^{(t)} - \left(\tilde{\mathcal{L}}_{I_L}^{(t)} + \tilde{\mathcal{L}}_{I_R}^{(t)}\right) \\ &= -\frac{1}{2}\frac{\left(\sum_{i\in I} g_i\right)^2}{\sum_{i\in I} h_i+\lambda}+\gamma - \left[-\frac{1}{2}\frac{\left(\sum_{i\in I_L} g_i\right)^2}{\sum_{i\in I_L} h_i+\lambda}+\gamma\right] - \left[-\frac{1}{2}\frac{\left(\sum_{i\in I_R} g_i\right)^2}{\sum_{i\in I_R} h_i+\lambda}+\gamma\right] \\ &= \frac{1}{2}\left[\frac{\left(\sum_{i\in I_L} g_i\right)^2}{\sum_{i\in I_L} h_i+\lambda}+\frac{\left(\sum_{i\in I_R} g_i\right)^2}{\sum_{i\in I_R} h_i+\lambda}-\frac{\left(\sum_{i\in I} g_i\right)^2}{\sum_{i\in I} h_i+\lambda}\right]-\gamma\end{aligned} \tag{87}$$

使用 $\mathcal{L}_{split}$ 评估待分裂结点。

- XGBoost 节点分裂贪婪查找算法

算法说明：枚举所有特征，对每个特征，枚举每个分裂点，根据 $\mathcal{L}_{split}$ 查找最优分裂点

输入：当前节点实例集I，特征维度d

输出：根据最大score分裂

算法步骤：

$gain \leftarrow 0$

$G \leftarrow \sum_{i\in I} g_i$，$H \leftarrow \sum_{i\in I} h_i$

for $k = 1$ to d do

  $G_L \leftarrow 0$，$H_L \leftarrow 0$

  for j in sorted(I, by $\mathbf{x}_{jk}$) do

    $G_L \leftarrow G_L + g_j$，$H_L \leftarrow H_L + h_j$

    $G_R \leftarrow G - G_L$，$H_R = H - H_L$

    $score \leftarrow \max\left(score, \frac{G_L^2}{H_L+\lambda}+\frac{G_R^2}{H_R+\lambda}-\frac{G^2}{H+\lambda}\right)$

  end

end

6.2. LightGBM

- 分裂准则：

基于决策树某节点的**数据集**O，**特征** j 在**分裂点** d 分裂后的**方差收益**为：

$$V_{j|O}(d)=\frac{1}{n_O}\left(\frac{(\sum_{x_i\in O;x_{ij}\le d} g_i)^2}{n^j_{l|O}(d)}+\frac{(\sum_{x_i\in O;x_{ij}>d} g_i)^2}{n^j_{r|O}(d)}\right) \tag{88}$$

$$\text{其中，} n_O=\sum I[x_i\in O], n^j_{l|O}(d)=\sum I[x_i\in O:x_{ij}\le d], n^j_{r|O}(d)=\sum I[x_i\in O:x_{ij}>d]$$

特征j的最佳分裂点：

$$d^*_j=\underset{d}{argmax}V_j(d) \tag{89}$$

解释一：基于CART树寻找最优分裂点

g_i 为当前步骤要拟合的值，损失函数使用MSE，分裂后使得损失最小，即：

$$\begin{aligned}
&min\left\{\sum_{p\in L}(g_p-\bar{g}_L)^2+\sum_{q\in R}(g_q-\bar{g}_R)^2\right\}\\
&=min\left\{\sum_{p\in L}g_p^2+\sum_{p\in L}\bar{g}_L^2-2\sum_{p\in L}g_p\bar{g}_L+\sum_{q\in R}g_q^2+\sum_{q\in R}\bar{g}_R^2-2\sum_{q\in R}g_q\bar{g}_R\right\}\\
&=min\left\{-\sum_{p\in L}\bar{g}_L^2-\sum_{q\in R}\bar{g}_R^2\right\}\\
&=max\left\{\sum_{p\in L}\bar{g}_L^2+\sum_{q\in R}\bar{g}_R^2\right\}\\
&=max\left\{n(\frac{1}{n}\sum_{p\in L}g_p)^2+m(\frac{1}{m}\sum_{q\in R}g_q)^2\right\}\\
&=max\left\{\frac{(\sum_{p\in L}g_p)^2}{n}+\frac{(\sum_{q\in L}g_q)^2}{m}\right\}\\
&L\text{和}R\text{分别为分裂后的左右子集}\\
&\bar{g}_L\text{和}\bar{g}_R\text{分别为左右子集}g\text{的均值}
\end{aligned} \tag{90}$$

解释二：参考XGBoost的信息增益

$$\begin{aligned}
&max\left[\frac{\left(\sum_{i\in I_L}g_i\right)^2}{\sum_{i\in I_L}h_i+\lambda}+\frac{\left(\sum_{i\in I_R}g_i\right)^2}{\sum_{i\in I_R}h_i+\lambda}-\frac{\left(\sum_{i\in I}g_i\right)^2}{\sum_{i\in I}h_i+\lambda}\right]\\
&=max\left[\frac{\left(\sum_{i\in I_L}g_i\right)^2}{\sum_{i\in I_L}h_i+\lambda}+\frac{\left(\sum_{i\in I_R}g_i\right)^2}{\sum_{i\in I_R}h_i+\lambda}\right]\\
&=max\left[\frac{\left(\sum_{i\in I_L}g_i\right)^2}{\sum_{i\in I_L}h_i}+\frac{\left(\sum_{i\in I_R}g_i\right)^2}{\sum_{i\in I_R}h_i}\right]
\end{aligned} \tag{91}$$

因为使用的是平方损失，那么样本的二阶导数为1，即 h_i 为1，故上式与 $V_{j|O}(d)$ 等价。

- **GOSS(Gradient based One Side Sampling) - 样本下采样**
  - 选择前 $a\%$ 个较大梯度值的样本
  - 从剩余的 $1-a\%$ 个小梯度样本，随机选择 $b\%$ 个样本
  - 对小梯度样本， 在计算信息增益时扩大一定倍数

由于梯度较大的数据实例在信息增益的计算中起着更重要的作用，所以GOSS可以在较小的数据量下获得相 当准确的信息增益估计。

- **EFB(Exclusive Feature Bundling) - 独立特征合并**

  解决数据稀疏的问题。在稀疏特征空间中，许多特征都是互斥的，也就是它们几乎不同时取非0值。因此，可以安全地把这些互斥特征绑到一起形成一个特征。

7. 参考资料

1. https://statisticallearning.org/biasvariance-tradeoff.html
2. https://www.cs.cornell.edu/courses/cs4780/2018fa/lectures/lecturenote12.html
3. https://bcheggeseth.github.io/CorrelatedData/index.html
4. http://www.milefoot.com/math/stat/rv-sums.htm
5. https://stats.stackexchange.com/questions/391740/variance-of-average-of-n-correlated-random-variables
6. https://github.com/Freemanzxp/GBDT_Simple_Tutorial
7. https://scikit-learn.org/stable/modules/ensemble.html
8. https://scikit-learn.org/stable/modules/tree.html#tree-algorithms-id3-c4-5-c5-0-and-cart