决策树：

ID3：以信息增益为准则选择最优划分属性

信息增益计算基于信息熵：

1，计算数据集的经验熵

2，计算某个特征下的条件熵

3，计算信息增益

但是这种分割算法存在一定的缺陷：

假设每个记录有一个属性“ID”，若按照ID来进行分割的话，由于ID是唯一的，因此在这一个属性上，能够取得的特征值等于样本的数目，也就是说ID的特征值很多。那么无论以哪个ID为划分，叶子结点的值只会有一个，纯度很大，得到的信息增益会很大，但这样划分出来的决策树是没意义的。由此可见，ID3决策树偏向于取值较多的属性进行分割，存在一定的偏好。为减小这一影响，有学者提出C4.5的分类算法。

C4.5：基于信息增益率选择最优分割的算法

分子计算与ID3一样，分母是由属性A的特征值个数决定的，个数越多，IV值越大，信息增益率越小，这样就可以避免模型偏好特征值多的属性，但是聪明的人一看就会发现，如果简单的按照这个规则来分割，模型又会偏向特征数少的特征。因此C4.5决策树先从候选划分属性中找出**信息增益高于平均水平**的属性，在从中选择增益率最高的。

对于连续值属性来说，可取值数目不再有限，因此可以采用离散化技术（如二分法）进行处理。将属性值从小到大排序，然后选择中间

值作为分割点，数值比它小的点被划分到左子树，数值不小于它的点被分到又子树，计算分割的信息增益率，选择信息增益率最大的属性值进行分割。

CART：以基尼系数为准则划分，可以用于分类和回归

CART是一棵二叉树，采用二元切分法，每次把数据切成两份，分别进入左子树、右子树。而且每个非叶子节点都有两个孩子，所以CART的叶子节点比非叶子多1。相比ID3和C4.5，CART应用要多一些，既可以用于分类也可以用于回归。CART分类时，使用基尼指数（Gini）来选择最好的数据分割的特征，gini描述的是纯度，与信息熵的含义相似。CART中每一次迭代都会降低GINI系数。

Gini(D)反映了数据集D的纯度，值越小，纯度越高。我们在候选集合中选择使得划分后基尼指数最小的属性作为最优化分属性。

分类树和回归树：

作为对比，先说分类树，我们知道ID3、C4.5分类树在每次分枝时，是穷举每一个特征属性的每一个阈值，找到使得按照feature<=阈值，和feature>阈值分成的两个分枝的熵最大的feature和阈值。按照该标准分枝得到两个新节点，用同样方法继续分枝直到所有人都被分入性别唯一的叶子节点，或达到预设的终止条件，若最终叶子节点中的性别不唯一，则以多数人的性别作为该叶子节点的性别。

回归树总体流程也是类似，不过在每个节点（不一定是叶子节点）都会得一个预测值，以年龄为例，该预测值等于属于这个节点的所有人年龄的平均值。分枝时穷举每一个feature的每个阈值找最好的分割点，**但衡量最好的标准不再是最大熵，而是最小化均方差**--即（每个人的年龄-预测年龄）^2 的总和 / N，或者说是每个人的预测误差平方和 除以 N。这很好理解，被预测出错的人数越多，错的越离谱，均方

差就越大，通过最小化均方差能够找到最靠谱的分枝依据。分枝直到每个叶子节点上人的年龄都唯一（这太难了）或者达到预设的终止条件（如叶子个数上限），若最终叶子节点上人的年龄不唯一，则以该节点上所有人的平均年龄做为该叶子节点的预测年龄。

随机森林：

组合分类器：将多个分类结果进行多票表决或取平均值，作为最终结果。

好处：

1，提升精度

2，处理过大过小数据集，数据集较大时，可以将数据集划分成多个子集，对子集构建分类器；数据集较小时，可通过多种抽样方式（bootstrap）从原始数据集抽样产生多组不同的数据集，构建分类器。

3，若决策边界过于复杂，则线性模型不能很好地描述真实情况。因此先对于特定区域的数据集，训练多个线性分类器，再将它们集成。

4，比较适合处理多源异构数据（存储方式不同（关系型，非关系））类别（时序、离散、连续、网络结构数据）

方法：

1，数据随机选取

第一，从原始的数据集中采取有放回的抽样（bootstrap），构造子数据集，子数据集的数据量是和原始数据集相同的。不同子数据集的元素可以重复，同一个子数据集中的元素也可以重复。

第二，利用子数据集来构建子决策树，将这个数据放到每个子决策树中，每个子决策树输出一个结果。最后，如果有了新的数据需要通过

随机森林得到分类结果，就可以通过对子决策树的判断结果的投票，得到随机森林的输出结果了。如下图，假设随机森林中有3棵子决策树，2棵子树的分类结果是A类，1棵子树的分类结果是B类，那么随机森林的分类结果就是A类。

2，特征随机

与数据集的随机选取类似，随机森林中的子树的每一个分裂过程并未用到所有的待选特征，而是**从所有的待选特征中随机选取一定的特征**，之后再在随机选取的特征中选取最优的特征。这样能够使得随机森林中的决策树都能够彼此不同，提升系统的多样性，从而提升分类性能。

GBDT和xgboost：

bagging：每次从样本集中根据均匀概率分布有放回的抽取和原始数据大小相同的样本集，对每次产生的训练集构造一个分类器，在对分类器进行组合。

boosting：每次抽样的分布不一样，每次迭代根据上次结果，增加错误样本比重，使模型在之后的迭代中注意到难以分类的样本，是一个不断学习不断提升的过程，也是boosting本质所在，将每次分类器集成，进行加权投票。

Adaboost：

1，通过对N个训练样本进行学习得到第一个弱分类器

2，将弱分类器分错的和其他数据构成新的N个训练样本，再学习得到第二个弱分类器

3，重复上面

4，加权投票，得到强分类器

GBDT：梯度下降+boosting+决策树

GBDT是以决策树（CART）为基学习器的GB算法，是迭代树，而不是分类树。

GBDT的核心就在于：每一棵树学的是之前所有树结论和的残差，这个残差就是一个加预测值后能得真实值的累加量。比如A的真实年龄是18岁，但第一棵树的预测年龄是12岁，差了6岁，即残差为6岁。那么在第二棵树里我们把A的年龄设为6岁去学习，如果第二棵树真的能把A分到6岁的叶子节点，那累加两棵树的结论就是A的真实年龄；如果第二棵树的结论是5岁，则A仍然存在1岁的残差，第三棵树里A的年龄就变成1岁，继续学习。

xgboost：boosting+决策树+正则化+二阶泰勒展开

Xgboost相比于GBDT来说，更加有效应用了数值优化，最重要是对损失函数（预测值和真实值的误差）变得更复杂。目标函数依然是所有树的预测值相加等于预测值。

损失函数引入了二阶泰勒展开

对比：

- 传统GBDT在优化时只用到**一阶导数**信息，xgboost则对代价函数进行了**二阶泰勒展开**，同时用到了一阶和二阶导数。顺便提一下，xgboost工具支持自定义代价函数，只要函数可一阶和二阶求导。
- xgboost在代价函数里加入了**正则项**，用于控制模型的复杂度。正则项里包含了树的叶子节点个数、每个叶子节点上输出的score的L2模的平方和。从Bias-variance tradeoff角度来讲，正则项降低

了模型的variance，使学习出来的模型更加简单，防止过拟合，这也是xgboost优于传统GBDT的一个特性。

- Shrinkage（缩减），相当于**学习速率**（xgboost中的eta）。每次迭代，增加新的模型，在前面成上一个小于1的系数，降低优化的速度，每次走一小步逐步逼近最优模型比每次走一大步逼近更加容易避免过拟合现象；
- 列抽样（column subsampling）。xgboost借鉴了随机森林的做法，支持列抽样（即每次的输入特征不是全部特征），不仅能降低过拟合，还能减少计算，这也是xgboost异于传统gbdt的一个特性。
- 忽略缺失值：在寻找splitpoint的时候，不会对该特征为missing的样本进行遍历统计，只对该列特征值为non-missing的样本上对应的特征值进行遍历，通过这个工程技巧来减少了为稀疏离散特征寻找splitpoint的时间开销
- 指定缺失值的分隔方向：可以为缺失值或者指定的值指定分支的默认方向，为了保证完备性，会分别处理将missing该特征值的样本分配到左叶子结点和右叶子结点的两种情形，分到那个子节点带来的增益大，默认的方向就是哪个子节点，这能大大提升算法的效率。
- 并行化处理：在训练之前，预先对每个特征内部进行了排序找出候选切割点，然后保存为block结构，后面的迭代中重复地使用这个结构，大大减小计算量。在进行节点的分裂时，需要计算每个特征的增益，最终选增益最大的那个特征去做分裂，那么各个特征的增益计算就可以开多线程进行，即在不同的特征属性上采用多线程并行方式寻找最佳分割点。

岭回归：L2回归（平方和）

$$S = \sum_{i=1}^{n} (y_i - f(x_i))^2$$

先说L1， L1其实就是把权重向量的每个分量的绝对值相加的和。显然【1，1，1】的L1正则就是3 ，【100，100，-98】的L1正则就是298。L2正则无非就是绝对值平方相加。

倾向于削弱每个特征值的权重

在原来的损失函数前加一个惩罚项，成为L2正则化。

$$S = \sum_{i=1}^{n} (y_i - f(x_i))^2$$

$$f(w) = \sum_{i=1}^{m} (y_i - x_i^T w)^2 + \lambda \sum_{i=1}^{n} w_i^2$$

LASSO回归：L1回归（绝对值和）

L1-norm 损失函数，又被称为 least absolute deviation (LAD，最小绝对偏差)

$$S = \sum_{i=1}^{n} |y_i - f(x_i)|.$$

L1正则化

$$J(\theta) = MSE(\theta) + \alpha \sum_{i=1}^{n} |\theta_i|$$

$$J(\theta) = \frac{1}{m} \sum_{i=1}^{m} \left(\theta^T \cdot x^{(i)} - y^{(i)}\right)^2 + \alpha \sum_{i=1}^{n} \theta_i^2$$