1. GDBT

    1. 是一种迭代的决策回归树算法,也是一种典型的Boosting算法。
    2. 原始的工作流程是：初始化CART回归树，针对每个样本计算残差，将这个残差作为新的真实值，用新的数据去训练下一个CART树。不断迭代直到决策树的个数达到预定义的数量停止，结合所有基分类器的输出作为最后结果的输出。(现在已经有了一个模型F,但是觉得它做的不够好,希望在他的基础上得到一个更好的模型,所以就新加一个模型h(x), 这样最后得到的总结果就是F(x)+h(x))。
    3. 用**缩减率**改进的工作流程:认为每次走一小步比走一大步更能获得好的效果，更能防止过拟合。工作流程：仍然以残差作为学习目标，但是每次只是用step*残差逐步逼近目标，step一般是0.001-0.1.和神经网络的的学习率是一样的。本质上缩减是给每棵树设置了一个权重系数。

2. GDBT的问答

    1. 为什么要使用回归决策树?
        - GDBT的核心在于累加所有树的结果。
        - 分类树用于分类, 最后的结果是类别, 如果男女, 这样累加类别是没有意义的。
        - 回归树用于预测数值, 最后的结果是实数, 累加起来依然是具有实际意义的.
    2. 分类树和回归树的区别是什么?
        - 分类树使用信息增益/信息增益率/基尼指数来划分节点, 中间会穷举所有特征的所有阈值, 最后选择一个合适的划分特征,最后根据叶子节点的投票确定预测样本的类别
        - 回归树使用最小化均方差划分节点, 中间会穷举所有特征的所有划分点, 最后根据叶子节点的样本均值作为预测样本的回归预测值.
    3. 为什么将残差做为真实值?
        - GDBT算法本质是一种加法模型，它的目标函数是MSE，导数是yi~−yi。所以残差就是下一步的最优化的反方向。
        - 每一步的残差计算其实变相的增大了分错样本的权重，而已经分对的样本则都趋向于0。这样后面就更加专注于那些分错的样本。

3. GDBT与AdaBoost的区别

    1. AdaBoost算法是利用前一轮的弱学习器的误差来更新样本权重值，然后一轮一轮的迭代；GDBT也是迭代，但是 GDBT 要求弱学习器必须是 CART 模型（低方差、高偏差）
    2. Adaboost算法使用指数损失函数，GDBT使用平方差损失函数

4. xgboost与GDBT的区别

    1. xgboost不仅支持决策树作为基分类器，还支持线性分类器（相当于引入L1 L2正则惩罚项的LR和线性回归，目标函数公式=误差平方和+正则项，似LR）
    2. loss函数用二阶泰勒展开，因此与损失函数更接近（损失函数：一个样本的误差；代价函数：整个训练集上所有样本误差的平均；目标函数：代价函数 + 正则化项）
    3. 加入了正则项，用来控制模型复杂度。正则化包括叶子结点个数和叶子节点权重的L2范数
    4. 加入缩减系数shrinkage，降低每一步boosting的树的影响，防止过拟合
    5. 列采样，从随机森林的借鉴而来，降低过拟合，减少计算
    6. 支持缺失值处理，要么全放左边，要么全放右边
    7. 支持并行。在每棵树进行节点分裂的时候，需要计算每个特征的增益，选择最大的那个特征作为分裂特征，各个特征的增益计算可以开多线程进行。这是因为xgboost把每一列数据存在一个内存单元block中，分列前，就可以对其进行预排序，然后在每次迭代中重复使用。

    1. 将数据转变成CSC类型（Compression Square Column）（读者一定要明白这是怎么回事）复杂度：O(data)，只需计算一次
    2. CSC类型中的每个block是一列特征，对每个block中的数值排序，并计算每个样本的一阶偏导和二阶偏导。复杂度：O(data)，只需计算一次
    3. 使用多线程来找每个block的最佳切分。复杂度：O(n)
    4. 根据最佳切分点，计算出左子树的索引和右子树的索引。复杂度：O(n)
    5. 遍历CSC把数据(并行处理)分为左子树的CSC数据，和右子树的CSC数据。复杂度O(n)
8. 近似算法。传统的树节点分裂，需要枚举所有可能的分隔点。xgboost中，在树节点分裂是，根据特征值的百分位数提出候选分裂节点，然后把每个点分配到这些桶里。该算法有两种形式：
    1. 全局近似，是在生成一棵树之前，对各个特征计算其分位点并划分样本；
    2. 局部近似，是在每个节点进行分裂时采用近似算法。