1. 简单介绍一下GBDT

GBDT全名是梯度提升树，使用boosting算法作为集成手段，集成多棵决策树的分类或回归结果。在第一颗决策树做出结果后，求出其与真实结果的残差，再将残差作为“真实结果”训练第二棵树，直至使残差最小，这个过程恰好等于对最小均方误差损失函数求一阶导数，因此也被称为梯度提升树GBDT

1. Xgboost有那些改进？

精度更高：GBDT 只用到一阶泰勒展开，而 XGBoost 对损失函数进行了二阶泰勒展开。XGBoost 引入二阶导一方面是为了增加精度，另一方面也是为了能够自定义损失函数，二阶泰勒展开可以近似大量损失函数；

灵活性更强：GBDT 以 CART 作为基分类器，XGBoost 不仅支持 CART 还支持线性分类器，（使用线性分类器的 XGBoost 相当于带 L1 和 L2 正则化项的逻辑斯蒂回归（分类问题）或者线性回归（回归问题））。此外，XGBoost 工具支持自定义损失函数，只需函数支持一阶和二阶求导；

正则化：XGBoost 在目标函数中加入了正则项，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、叶子节点权重的 L2 范式。正则项降低了模型的方差，使学习出来的模型更加简单，有助于防止过拟合；

Shrinkage（缩减）：相当于学习速率。XGBoost 在进行完一次迭代后，会将叶子节点的权重乘上该系数，主要是为了削弱每棵树的影响，让后面有更大的学习空间；

列抽样：XGBoost 借鉴了随机森林的做法，支持列抽样，不仅能降低过拟合，还能减少计算；

缺失值处理：XGBoost 采用的稀疏感知算法极大的加快了节点分裂的速度；可以并行化操作：块结构可以很好的支持并行计算。

1. GBDT与随机森林的异同点？

GBDT与随机森林的基模型都是决策树，不同之处在于GBDT的集成方式为Boosting，随机森林的集成方式是Bagging

1. Xgboost防止过拟合有什么方法，如何调参？

Xgb在目标函数中加入了正则项控制模型复杂度，正则化手段为叶子节点个数和叶子节点权重的L2范式

调参：**lambda** [default=1] L2正则化项的权重

1. xgb为什么对缺失值不敏感，如何处理缺失值的？

采用稀疏感知算法避免对于缺失值过于敏感

如何处理：XGBoost 在构建树的节点过程中只考虑非缺失值的数据遍历，而为每个节点增加了一个缺省方向，当样本相应的特征值缺失时，可以被归类到缺省方向上，分别枚举特征缺省的样本归为左右分支后的增益，选择增益最大的枚举项即为最优缺省方向。我理解就是表面上xgb是二叉树，其实还有第三个不确定分支，把所有缺失值归到这个不确定分支上，最终这个不确定分支归属在那个分支上对结果表现最良好，就归到哪一个分支上。

1. 解释一下GBDT沿着梯度下降方向提升，如何实现的？

GBDT每一次迭代希望残差更小，残差就是最小均方误差的负梯度，求残差的最小值就是让这个梯度越来越小，就是梯度下降的方向，GBDT沿着梯度下降的方向迭代，残差越来越小，损失函数越来越小，结果得到提升。

“提升”是说每一次迭代后是对一个新的残差函数求最小值，即梯度下降，但是最后会把这些结果加在一起，所以叫提升？感觉每次都是上一次的残差求最小值，最后是应该加起来，但是还不是特别理解为什么要加起来。已经写在疑问里面了。