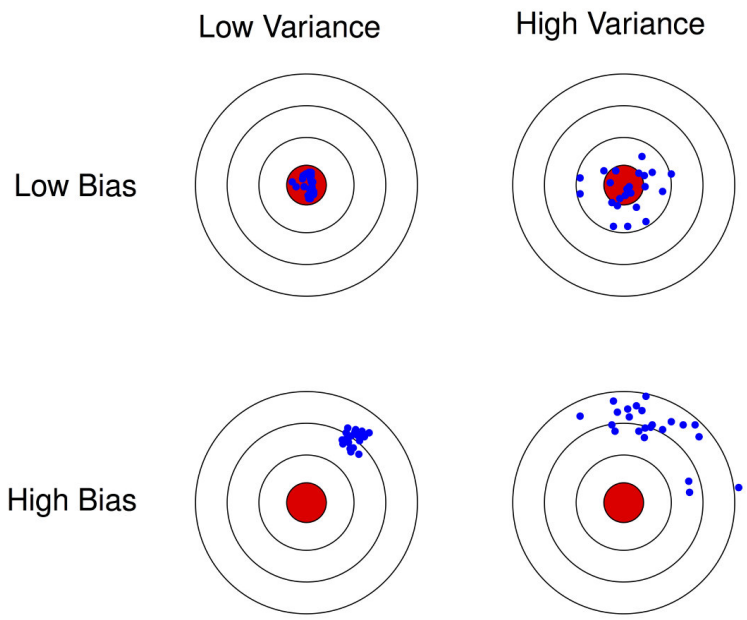
综合

一、请描述一下偏差与方差、过拟合与欠拟合的关系



1 / 1

1、初步定义

- 方差: 预测值的平均值
  - 方差度量了同样大小的训练集的变动所导致的学习性能的变化, 即 **刻画了数据扰动所造成的影响**
- 偏差: 真实值和预测值
  - 偏差度量了学习算法的期望预测与真实结果的偏离程序, 即 **刻画了学习算法本身的拟合能力** .
- 噪声:是真实标记与数据集中的实际标记之间的偏差
  - 噪声表达了在当前任务上任何学习算法所能达到的期望泛化误差的下界, 即 **刻画了学习问题本身的难度** . 巧妇难为无米之炊, 给一堆很差的食材, 要想做出一顿美味, 肯定是很有难度的.

2、四个变量之间的关系

- (方差大的时候,过拟合,和平均值之间求得的值比较大
- 欠拟合的时候,偏差大,即预测值和真实值之间的比较)

二、分类问题总结-交叉熵、对数似然损失、sigmoid、softmax

分类问题名称	输出层使用激活函数	对应的损失函数
二分类	sigmoid函数	二分类交叉熵损失函数（binary_crossentropy）
多分类	Softmax函数	多类别交叉熵损失函数（categorical_crossentropy）
多标签分类	sigmoid函数	二分类交叉熵损失函数（binary_crossentropy）

三、L1和L2的区别

L1范数（L1 norm）是指向量中各个元素绝对值之和，也有个美称叫“稀疏规则算子”（Lasso regularization）。

- 比如 向量A=[1，-1，3]， 那么A的L1范数为 |1|+|-1|+|3|.

简单总结一下就是：

- L1范数: 为x向量各个元素绝对值之和。
- L2范数: 为x向量各个元素平方和的1/2次方，L2范数又称Euclidean范数或者Frobenius范数
- Lp范数: 为x向量各个元素绝对值p次方和的1/p次方.

L1和L2的差别，为什么一个让绝对值最小，一个让平方最小，会有那么大的差别呢?

- 看导数一个是1一个是w便知, 在靠进零附近, L1以匀速下降到零, 而L2则完全停下来了.
- 这说明L1是将不重要的特征(或者说, 重要性不在一个数量级上)尽快剔除, L2则是把特征贡献尽量压缩最小但不至于为零。

四、防止过拟合的方法

- 过拟合的原因是
  - 算法的学习能力过强；
  - 一些假设条件（如样本独立同分布）可能是不成立的；
  - 训练样本过少不能对整个空间进行分布估计。
- 处理方法：
  - 1、早停止：如在训练中多次迭代后发现模型性能没有显著提高就停止训练
  - 2、数据集扩增：原有数据增加、原有数据加随机噪声、重采样
  - 3、正则化，正则化可以限制模型的复杂度
  - 4、交叉验证
  - 5、特征选择/特征降维
  - 6、创建一个验证集是最基本的防止过拟合的方法。我们最终训练得到的模型目标是要在验证集上面有好的表现，而不训练集

五、衡量分类器的好坏?

- 参考讲义

特征工程部分

一、哪些机器学习算法不需要做归一化处理

- 在实际应用中，需要归一化的模型：
  - 1、基于距离计算的模型：KNN。
  - 2、通过梯度下降法求解的模型：线性回归、逻辑回归、支持向量机、神经网络。
- 但树形模型不需要归一化，因为它们不关心变量的值，而是关心变量的分布和变量之间的条件概率，如决策树、随机森林(Random Forest)。

```
衍生题：
为什么树形结构不需要进行归一化处理?

因为数值缩放不影响分裂点位置，对树模型的结构不造成影响。
按照特征值进行排序的，排序的顺序不变，那么所属的分支以及分裂点就不会有不同。而且，树模型是不能进行梯度下降的，因为构建树模型（回归树）寻找最优点时是通过寻找最优分裂点完成的，因此树模型是阶跃的，阶跃点是不可导的，并且求导没意义，也就不需要归一化。

既然树形结构（如决策树、RF）不需要归一化，那为何非树形结构比如SVM、LR、Knn、KMeans之类则需要归一化呢?

对于线性模型，特征值差别很大时，比如说LR，我有两个特征，一个是(0,1)的，一个是(0,10000)的，运用梯度下降的时候，损失等高线是椭圆形，需要进行多次迭代才能到达最优点。
但是如果进行了归一化，那么等高线就是圆形的，促使SGD往原点迭代，从而导致需要的迭代次数较少。
```

逻辑回归

一、简单介绍一下你对逻辑回归的理解

要想掌握逻辑回归，必须掌握两点：

- 逻辑回归中，其输入值是什么
- 如何判断逻辑回归的输出

1、输入

逻辑回归的输入就是一个线性回归的结果。

2、激活函数

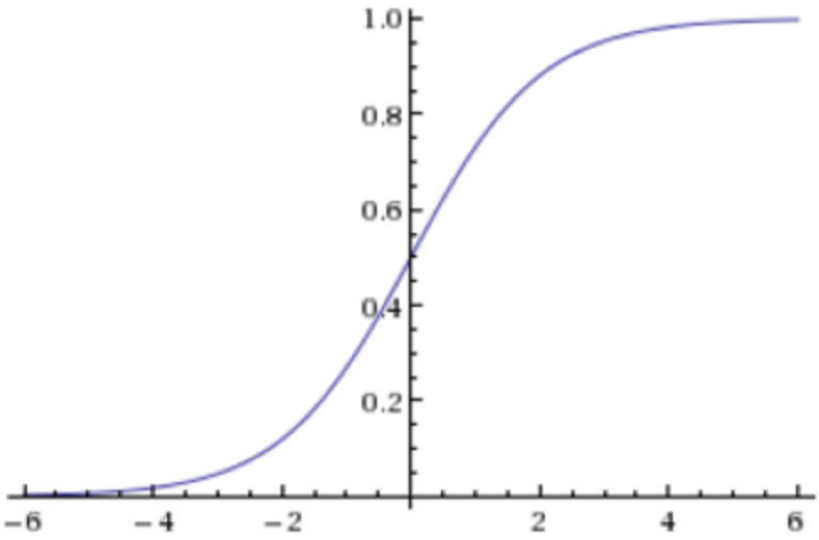
- sigmoid函数

$$g(w^T, x) = \frac{1}{1 + e^{-h(w)}} = \frac{1}{1 + e^{-w^T x}}$$

```
# numpy实现sigmoid

import numpy
y=1/(1+np.exp(-x))
```

- 判断标准
  - 回归的结果输入到sigmoid函数当中
  - 输出结果：[0, 1]区间中的一个概率值，默认为0.5为阈值



3、损失

逻辑回归的损失，称之为**对数似然损失**，公式如下：

- 分开类别：

$$cost(h_\theta(x), y) = \begin{cases} -log(h_\theta(x)) & \text{if y=1} \\ -log(1 - h_\theta(x)) & \text{if y=0} \end{cases}$$

> $h_\theta(x)$为预测值对应的概率值。

- 综合完整损失函数

$$cost(h_\theta(x), y) = \sum_{i=1}^{m} -y_i log(h_\theta(x)) - (1 - y_i) log(1 - h_\theta(x))$$

4、优化

使用梯度下降优化算法，去减少损失函数的值。这样去更新逻辑回归前面对应算法的权重参数，**提升原本属于1类别的概率，降低原本是0类别的概率。**

二、逻辑斯蒂回归为什么要对特征进行离散化。

在工业界，很少直接将连续值作为逻辑回归模型的特征输入，而是将连续特征离散化为一系列0、1特征交给逻辑回归模型，这样做的优势有以下几点：

- 1、离散特征的增加和减少都很容易，易于模型的快速迭代；
- 2、离散化后的特征对异常数据有很强的鲁棒性：比如一个特征是年龄>30是1，否则0。如果特征没有离散化，一个异常数据“年龄300岁”会给模型造成很大的干扰；
- 3、逻辑回归属于广义线性模型，表达能力受限；单变量离散化为N个后，每个变量有单独的权重，相当于为模型引入了非线性，能够提升模型表达能力，加大拟合；
- 4、离散化后可以进行特征交叉，由M+N个变量变为M*N个变量，进一步引入非线性，提升表达能力；
- 5、特征离散化后，模型会更稳定，比如如果对用户年龄离散化，20-30作为一个区间，不会因为一个用户年龄长了一岁就变成一个完全不同的人。当然处于区间相邻处的样本会刚好相反，所以怎么划分区间是门学问；
- 6、特征离散化以后，起到了简化了逻辑回归模型的作用，降低了模型过拟合的风险。

亚马逊AI主任科学家**李沐博士**曾经说过：**模型是使用离散特征还是连续特征，其实是一个“海量离散特征+简单模型” 同 “少量连续特征+复杂模型”的权衡。既可以离散化用线性模型，也可以用连续特征加深度学习。就看是喜欢折腾特征还是折腾模型了。** 通常来说，前者容易，而且可以n个人一起并行做，有成功经验；后者目前看很赞，能走多远还须拭目以待。

三、LR与线性回归的区别与联系

- LR工业上一般指Logistic Regression(逻辑回归)而不是Linear Regression(线性回归).
- LR在线性回归的实数范围输出值上施加sigmoid函数将值收敛到0~1范围, 其目标函数也因此从差平方和函数变为对数损失函数, 以提供最优化所需导数（sigmoid函数是softmax函数的二元特例, 其导数均为函数值的f*(1-f)形式）。
- 请注意, LR往往是解决二元0/1分类问题的, 只是它和线性回归耦合太紧, 不自觉也冠了个回归的名字(马甲无处不在). 若要求多元分类,就要把sigmoid换成大名鼎鼎的softmax了。

SVM

一、请说说支持向量机（support vector machine，SVM）的原理

- 参考资料1：讲义部分-http://52.83.69.131:10003/%E6%94%AF%E6%8C%81%E5%90%91%E9%87%8F%E6%9C%BA/section3/
- 参考资料2：同路径下手推过程

二、LR和SVM的联系与区别

LR和SVM都可以处理分类问题，且一般都用于处理线性二分类问题（在改进的情况下可以处理多分类问题）

- 相同点
  - 1、都是线性分类器。本质上都是求一个最佳分类超平面。
  - 2、都是监督学习算法。
  - 3、都是判别模型。判别模型不关心数据是怎么生成的，它只关心信号之间的差别，然后用差别来简单对给定的一个信号进行分类。常见的判别模型有：KNN、SVM、LR，常见的生成模型有：朴素贝叶斯，隐马尔可夫模型。
- 区别：
  - 1、LR是参数模型，svm是非参数模型；
  - 2、从目标函数来看，区别在于逻辑回归采用的是logistical loss，SVM采用的是hinge loss，这两个损失函数的目的都是增加对分类影响较大的数据点的权重，减少与分类关系较小的数据点的权重。
  - 3、SVM的处理方法是只考虑support vectors，也就是和分类最相关的少数点，去学习分类器。而逻辑回归通过非线性映射，大大减小了离分类平面较远的点的权重，相对提升了与分类最相关的数据点的权重。
  - 4、逻辑回归相对来说模型更简单，好理解，特别是大规模线性分类时比较方便。而SVM的理解和优化相对来说复杂一些，SVM转化为对偶问题后,分类只需要计算与少数几个支持向量的距离,这个在进行复杂核函数计算时优势很明显,能够大大简化模型和计算。
  - 5、logic 能做的 svm能做，但可能在准确率上有问题，svm能做的logic有的做不了。
  - 6、SVM的损失函数就自带正则，而 LR 必须另外在损失函数之外添加正则项。 红框内就是L2正则。

$$\mathcal{L}(w,b,\alpha) = \frac{1}{2}\|w\|^2 - \sum_{i=1}^{n}\alpha_i\left(y_i(w^T x_i + b) - 1\right)$$

```
参数模型和非参数模型

非参数模型（non-parametric model）和参数模型（parametric model）作为数理统计学中的概念，现在也常用于机器学习领域中。
```

在统计学中，参数模型通常假设总体服从某个分布，这个分布可以由一些参数确定，如正态分布由均值和标准差确定，在此基础上构建的模型称为参数模型；非参数模型(类比KNN)对于总体的分布不做任何假设或者说是数据分布假设自由，只知道其分布是存在的，所以就无法得到其分布的相关参数，只能通过非参数统计的方法进行推断。

所以说，参数模型和非参数模型中的“参数”并不是模型中的参数，而是数据分布的参数。

二者大体可以这样进行区分：

一、首先需要明确的是 非参数模型并不是说模型中没有参数！
这里的non-parametric类似单词priceless，并不是没有价值，而是价值非常高，无价，也就是参数是非常非常非常多的！（注意：所谓“多”的标准，就是参数数目大体和样本规模差不多）
而：可以通过有限个参数来确定一个模型，这样的方式就是“有参数模型”，也就是这里说的参数模型，如线性回归、Logistic回归（假定样本维度为N，则假定N个参数 theta1,theta2...thetaN）。

二、其次：参数模型对学到的函数方程有特定的形式，也就是明确指定了目标函数的形式 -- 比如线性回归模型，就是一次方程的形式，然后通过训练数据学习到具体的参数。
所以参数机器学习模型包括两个部分：
1、选择合适的目标函数的形式。
2、通过训练数据学习目标函数的参数。
通常来说，目标函数的形式假设是：对于输入变量的线性联合，于是参数机器学习算法通常被称为“线性机器学习算法”。

三、非参数机器学习算法：对于目标函数形式不作过多的假设的算法称为非参数机器学习算法。通过不做假设，算法可以自由的从训练数据中学习任意形式的函数。
对于理解非参数模型的一个好例子是k近邻算法，其目标是基于k个最相近的模式对新的数据做预测。这种理论对于目标函数的形式，除了相似模式的数目以外不作任何假设。

四、最后：

常见的参数机器学习模型有：
1、逻辑回归（logistic regression）
2、线性回归（linear regression）

参数机器学习算法有如下优点：
1、简洁：理论容易理解和解释结果。
2、快速：参数模型学习和训练的速度都很快。
3、数据更少：通常不需要大量的数据，在对数据的拟合不很好时表现也不错。
参数机器学习算法的局限性：
1、拘束：以指定的函数形式来指定学习方式。
2、有限的复杂度：通常只能应对简单的问题。
3、拟合度小：实际中通常无法和潜在的目标函数完全吻合，也就是容易出现欠拟合。

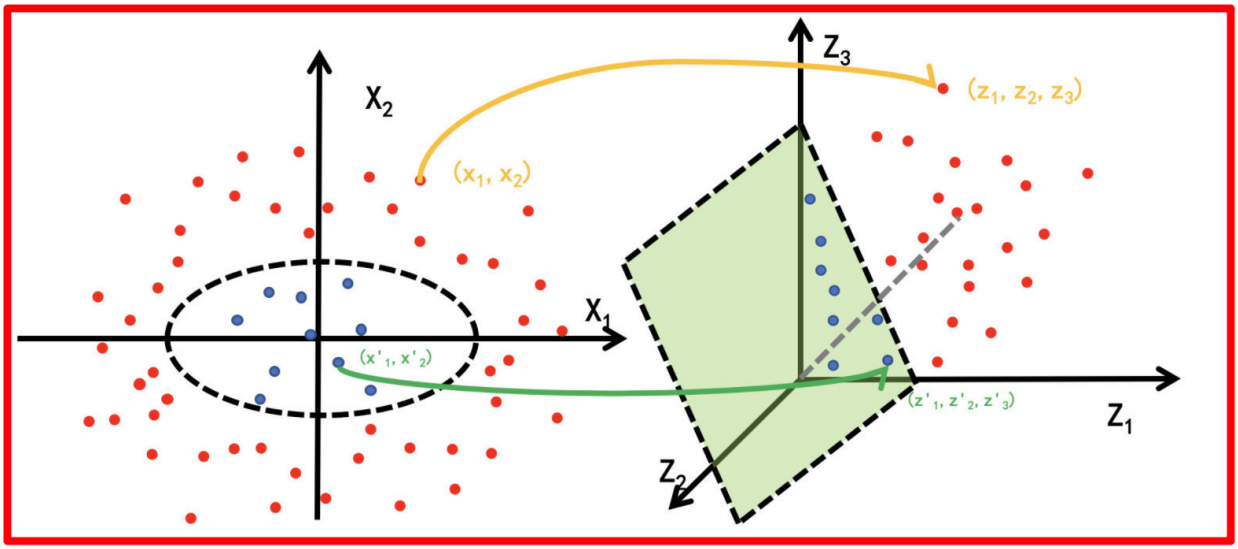
常见的非参数机器学习模型有：
1、决策树
2、朴素贝叶斯

3、支持向量机（SVM的例子中，SVM的参数α数目和样本数目相同，从定义看来，因为参数数目和样本规模相当，所以属于无参数模型。当然，SVM通过得到支撑向量的方式，只有若干样本的参数α不为0，从这个角度，SVM还属于“稀疏模型”，这又属于另外一码事了。）
4、神经网络

非参数机器学习算法的优势有：
1、可变性：可以拟合许多不同的函数形式。
2、模型强大：对于目标函数不做假设或者作出很小的假设。
3、表现良好：对于训练样本数据具有良好的拟合性。
非参数机器学习算法的局限性：
1、需要更多数据：对于拟合目标函数需要更多的训练数据。
2、速度慢：因为需要训练跟多的参数，所以训练过程通常比较慢。
3、过拟合：有较高的风险发生过拟合，对于预测的效果解释性不高。

三、带核的SVM为什么能分类非线性问题?

核函数会把样本点隐射到高维空间，在高维空间非线性问题转化为线性问题, SVM得到超平面是高维空间的线性分类平面, 如图:



其分类结果也视为低维空间的非线性分类结果, 因而带核的SVM就能分类非线性问题。

HMM和CRF

一、简述一下HMM模型构成，三个关键问题以及求解方法

- 参考HMM章节，第三节
  - ps：根据自己实际情况进行复习。

集成学习

一、RF、AdaBoost、GBDT、XGBoost、LinghtGBM原理

见讲义

二、请具体说说Boosting和Bagging的区别

bagging和boosting的区别

- 区别一:数据方面
  - Bagging：对数据进行采样训练；
  - Boosting：根据前一轮学习结果调整数据的重要性。
- 区别二:投票方面
  - Bagging：所有学习器平权投票；
  - Boosting：对学习器进行加权投票。
- 区别三:学习顺序
  - Bagging的学习是并行的，每个学习器没有依赖关系；
  - Boosting学习是串行，学习有先后顺序。
- 区别四:主要作用
  - Bagging主要用于提高泛化性能（解决过拟合，也可以说降低方差）
  - Boosting主要用于提高训练精度 （解决欠拟合，也可以说降低偏差）

- Bagging之随机森林
  - 随机森林改变了决策树容易过拟合的问题，这主要是由两个操作所优化的：
    - 1）Boostrap从袋内有放回的抽取样本值
    - 2）每次随机抽取一定数量的特征（通常为sqr(n)）。
  - 分类问题：采用Bagging投票的方式选择类别频次最高的
  - 回归问题：直接取每颗树结果的平均值。

- Boosting之AdaBoost
  - Boosting的本质实际上是一个加法模型，通过改变训练样本权重学习多个分类器并进行一些线性组合。而Adaboost就是加法模型+指数损失函数+前项分布算法。Adaboost就是从弱分类器出发反复训练，在其中不断调整数据权重或者是概率分布，同时提高前一轮被弱分类器误分的样本的权值。最后用分类器进行投票表决（但是分类器的重要性不同）。
- Boosting之GBDT
  - GBDT（梯度提升决策树）是为了解决一般损失函数的优化问题，方法是用损失函数的负梯度在当前模型的值来模拟回归问题中残差的近似值。 注：由于GBDT很容易出现过拟合的问题，

所以推荐的GBDT深度不要超过6，而随机森林可以在15以上。

- Boosting之Xgboost
  - 这个工具主要有以下几个特点：
    - 支持线性分类器
    - 可以自定义损失函数，并且可以用二阶偏导
    - 加入了正则化项：叶节点数、每个叶节点输出score的L2-norm
    - 在一定情况下支持并行，只有在建树的阶段才会用到，每个节点可以并行的寻找分裂特征。

三、RF与GBDT之间的区别与联系?

- 相同点：
  - 都是由多棵树组成，最终的结果都是由多棵树一起决定。
- 不同点：
  - 1、组成随机森林的树可以分类树也可以是回归树，而GBDT只由回归树组成；
  - 2、组成随机森林的树可以并行生成，而GBDT是串行生成
  - 3、随机森林的结果是多数表决表决的，而GBDT则是多棵树累加之和
  - 4、随机森林是减少模型的方差（防止过拟合），而GBDT是减少模型的偏差（防止欠拟合）
  - 5、GBDT的会累加所有树的结果，而这种累加是无法通过分类完成的，因此GBDT的树都是CART回归树，而不是分类树（尽管GBDT调整后也可以用于分类但不代表GBDT的树为分类树）

四、xgboost如何寻找最优特征？是有放回还是无放回的呢?

xgboost在训练的过程中给出各个特征的增益评分，最大增益的特征会被选出来作为分裂依据, 从而记忆了每个特征对在模型训练时的重要性，从根到叶子中间节点涉及某特征的次数作为该特征重要性排序。

xgboost属于boosting集成学习方法, 样本是不放回的, 因而每轮计算样本不重复. 另一方面, xgboost支持子采样, 也就是每轮计算可以不使用全部样本, 以减少过拟合. 进一步地, xgboost 还有列采样, 每轮计算按百分比随机采样一部分特征, 既提高计算速度又减少过拟合。

五、为什么xgboost不用后剪枝?

后剪枝计算代价太高了，合并一次叶节点就要计算一次测试集的表现，数据量大的情况下非常消耗时间，而且也并不是特别必要，因为这样很容易过拟合测试集。

六、你有自己用过别的模型然后调参之类的吗？能说一下基本的调参流程吗？XGBoost知道吗，以XGBoost为例子说一下调参流程吧。

一般来说采用贝叶斯优化等启发式的优化算法确定相对最佳参数（如果不熟悉的话用随机搜索也是可以的，或者网格搜索但是参数得到步长设置的很大，一步一步确定相对最优参数的区间）；然后再根据实际的模型在验证集上的表现做一些微调。

对于过拟合优先调整max_depth和树的数量，在实际使用过程中这两个参数对于模型的整体效果影响很大很明显。对于欠拟合，反着来就行了。

七、xgboost对特征缺失敏感吗，对缺失值做了什么操作，存在什么问题

- 不敏感，可以自动处理，处理方式是将missing值分别加入左节点、右节点取分裂增益最大的节点将missing样本分裂进这个节点 。
- 这种处理方式的问题在xgboost仅仅在特征的非缺失的值上进行分裂然后missing值直接放入其中一个节点，显然当缺失值很多的情况下，比如缺失80%，那么xgb分裂的时候仅仅在20%的特征值上分裂，这是非常容易过拟合的。

八、RF和xgboost哪个对异常点更敏感

假设当到达某一轮的时候，所有正常样本的计算得到的负梯度都很小而异常样本的负梯度很大，例如：

- 【0.0000001,0.0000001,0.0000001,0.0000001,0.0000001,10】,

这个时候新树会可能会继续进行不正常的分裂为：

- [0.0000001,0.0000001,0.0000001,0.0000001,0.0000001],[10],

而这样的分裂是不合理的，因为异常值本身可能是因为某些人为失误导致的数据记录错误，或者异常样本完全是属于另外一种分布，此时强制要进行模型训练会导致模型的结果有偏从而发生过拟合。当然异常样本数量很少比如10个以内的时候而正常样本有100000000个其实基本没什么影响，但是如果占比较高的话是会产生影响的。

九、XGBoost与GDBT的区别

xgboost类似于gbdt的优化版，不论是精度还是效率上都有了提升。

- 区别一：
    - XGBoost生成CART树考虑了树的复杂度，
    - GDBT未考虑，GDBT在树的剪枝步骤中考虑了树的复杂度。
- 区别二：

    - XGBoost是拟合上一轮损失函数的二阶导展开，GDBT是拟合上一轮损失函数的一阶导展开，因此，XGBoost的准确性更高，且满足相同的训练效果，需要的迭代次数更少。
- 区别三：
    - XGBoost与GDBT都是逐次迭代来提高模型性能，但是XGBoost在选取最佳切分点时可以开启多线程进行，大大提高了运行速度。

十、对比一下XGB和lightGBM在节点分裂时候的区别

- xgb是level-wise，lgb是leaf-wise，
- level-wise指在树分裂的过程中，同一层的非叶子节点，只要继续分裂能够产生正的增益就继续分裂下去，
- 而leaf-wise更苛刻一点，同一层的非叶子节点，仅仅选择分裂增益最大的叶子节点进行分裂。

十一、和xgboost相比，lightGBM 主要基于哪些方面进行了优化

> 不一定要全部答出

1. 基于Histogram（直方图）的决策树算法
2. Lightgbm 的Histogram（直方图）做差加速
3. 带深度限制的Leaf-wise的叶子生长策略
4. 直接支持类别特征
5. 直接支持高效并行

十二、从通过直方图构造数据的角度，说一下Lightgbm相对于xgboost的优缺点

- 优点：直方图算法—更高（效率）更快（速度）更低（内存占用）更泛化（分箱与之后的不精确分割也起到了一定防止过拟合的作用）；
- 缺点：直方图较为粗糙，会损失一定精度，但是在gbm的框架下，基学习器的精度损失可以通过引入更多的tree来弥补。