

风控碎碎念

吴江

2024 年 11 月 24 日

## 1 前言

路漫漫其修远兮，吾将上下而求索。

## 2 基础知识

风控的过程其实就是将具体的问题抽象化，找到你的目标  $Y$ ，然后准备  $X$ ，一步步地去用  $X$  拟合  $Y$ ，可能是存在  $X$  到  $Y$  的映射，也可能是存在  $(X,Y)$  两者之间的联合分布。Who knows?

### 2.1 一些关键的风控指标

#### 2.1.1 二分类模型的评价指标

### 3 风险模型建模流程

实践往往比口述来的实在，一个完整的建模流程只有自己去走完，尝试过中间可能会碰到的每一个坑，下一次能够更为有效地推进自身工作。

简单来说，建模流程涉及到的关键维度包含样本、特征、模型三大块。

#### 3.1 建模样本的准备

合适的样本选择是至关重要的，频繁迭代模型的重要因素就包含外部环境变化 or 策略改变导致的客群的更新，原有的模型一方面随着时间的自然衰减，另一方面伴随客群的变动，会逐渐失效。

建模样本的选择包含客群、时间窗、渠道类型等，以贷前为例，一般收授信 30 天内的首借样本，贷中则包含次新户、老户、睡眠户等，此外为了提升黑样本浓度，还可以加入一些轻资产 or API 用户作为补充，如何选取额外补充的样本，就需要一些方法，例如 PSM 等，这些会在后续进行补充。

时间窗的选择也尤为重要，在不同的阶段，策略的通过率是波动的，可能模型选取的时间窗不够久的话，不能够很好地体现大环境的变化周期。

#### 3.2 建模特征的选取

通常来说，还是以结构化的数据为主。从数据来源可以分为两大类，包括内部数据和外部数据。内部数据比较直接的，就是客户在审完时提供的各类基本信息（贷前），贷中则更为丰富，因为经历的客户生命周期更长，包含了大量客户的借还款行为；外部数据

则更为丰富，最常用的就是客户的人行征信数据，直接从人行报文解析的，此外还有多头类数据，可能有朴道、百融、同盾这种数据供应商，抑或是蚂蚁链、度小满，用户在这些渠道查询时也能获得多头数据，运营商数据、设备信息数据、各色的消费行为、搜索点击行为、欺诈行为也可以通过外部数据推断（腾讯云、蚂蚁链、度小满、友盟等）。

在准备好特征宽表之后，就需要对于特征进行一定的筛选，常规的筛选方法包含对特征的统计指标计算，唯一值数量，缺失率，iv（toad 中对于连续型变量使用决策树进行分箱，对于类别型变量按照唯一值分箱），psi（特征的稳定性，这一指标可能会影响模型的长期表现及衰退情况），feature\_importance（对于树模型来说，一般可以通过 gain 或 split 来刻画，这种情况下，可以采用 embedding 的方法对于特征 list 进行一定的截断，例如在保留 90% 的入模特征时，模型性能只有 0.1pp 的衰退，那就可以考虑剔除重要性后 10% 的变量），进一步的，为了保证入模特征的稳定性，可以进行 k-fold，例如在每一个 fold 上重要性排名靠前的变量的交集，为了避免模型的随机性，还可以采用 shuffle-y（null importance feature selection 如果在打乱 y 时，特征 x 的重要性还是没有很大的衰减，可能这个特征的随机性比较强，注意 xgboost 和 lightgbm 这类 boosting 树模型会倾向于连续型变量）。

还有一些要补充的，对于一些基础信息类变量，可能其重要性，无论是 iv 还是 feature\_importance 未必能达到入模的标准，但是当建模时引入这些变量可以丰富数据维度（有可能方便客户用户画像？比如可以看高风险客群的年龄、性别、在贷余额、多头查询次数等和大盘的差异）。对于一些变量，可能缺失率很高（黑名单类特征），但是将这些变量引入模型当中还是有必要的。还有一点，

例如人行特征，哪怕加工成了子分 or 子模型还是会保留子分、子模型和底层特征混用，这个需要进一步的探究。

### 3.3 建模流程的推进

以下的记录方式以贷前为例，贷中由于存在 uid 重复的情况，可能不完全一致，数据集大体上分为两大部分 dev 和 oot，或者更细一点 ins (train)、oos (valid)、oot (test)，尽管我们想要做出来的模型是在 oot 上有这更好的表现且 by 月衰退在可控范围内，但是为了客观期间我们仅能将模型分打在上面看效果，而不是让它参与早停甚至训练调参的过程（极度危险的行为）。

对于单一的子模型，oot 的时间通常是晚于 dev (ins+oos) 的，这样的话可以看到时间上的自然衰减及 month (衰减包含两种，随着 mob 的和随着 month 的)，但是对于 ins 和 oos 的划分，则未必有定论，以 8-2 划分为例，有两种方法，例如随机选取 20% 作为 oos 或者说可以在时间上进行分位点的切分，两者孰优孰劣需要看具体的场景。因为我们在交付模型分给业务方 or 进行子模型融合时，通常来说 ins 上的模型效果会优于 oos (因为 ins 参与了训练而 oos 参与了早停，ins 其实存在小小的“优势”)，这是难以避免的，我们可以考虑将模型的每个 20% 分别作为 oos，例如 dev 可以划分为 ins1-ins5，轮流使用 ins1-ins5 作为 oos 调参，而用其余的 4 块训练，这样在定版最终的模型分时，dev 上每个样本的分数应该是它作为 oos 没参与训练时那一版模型（上述方法中有 5 个模型，可以采用 1 组参数，也可以采用 5 组各自最优的参数）打分结果，而对于 oot，则可以用 5 版模型 score，这样可以一定程度上保持模型的稳定性，切分样本时建议使用 StratifiedKFold，保证抽样的稳定性，不会有一折数据明显偏离。

通常来说推荐 lightgbm，速度快，性能损失相较于 xgboost 忽略不计，对于类别型变量，需要使用 categorical\_feature 指明，对于调参 hyperopt、bayes 都可以尝试，最后选出的应该是 oos 上效果最好的版本，也需要保证和 ins 之间的 gap 不要太大，会有过拟合的风险。

### 3.4 建模结果的复盘

最直接的还是看那几个指标 ks、auc、lift 以及新旧模型的 swap 分析，ks（一定是越高越好吗？未必，它代表的是累计正样本和累计负样本差值取 max，可能存在跳点），auc 更为全面一点（auc 高的一定好吗？这还涉及到损失，例如我的模型能更好地识别出一些小金额的好样本而不是大金额的好样本，可能未必更好），lift 就很简单了，单调性重要，首尾部能拉开更好，无论是人头还是金额上的（和大盘风险对比区分度，假设一刀切可以带来多少的增益 or 降低多少的损失），swap 主要是看不同版本模型分之间的评级迁移（例如十等分的交叉矩阵，最理想的情况肯定是对角线及次对角线的占比高，原理对角线的占比低），如果模型分迁移很厉害，也是一定的问题。

### 3.5 一些创新的点

模型的参数怎么调，样本的权重怎么设置（weight 怎么给），选取什么样的损失函数（交叉熵全能但是不精，可以考虑 focalloss、ranknet 等等，多看文档），是不是可以增量学习（例如先用主营样本 train，然后混一小部分样本进去，或者将好学的样本及模型分比较低的样本剔除掉然后重新训练），能够探索的东西还有很多。

### 3.6 建模效果的增益点

和之前的思路类似，建模效果的增益可以拆解为如下的几块，样本的增益、特征的增益、算法的增益等。

#### 3.6.1 样本增益

样本的增益从纵向来看主要来自于两块，时间窗的更新及拉长，举个例子，例如此次建模的全部 dev 为 22.07 至 24.01，oot 为 24.02-24.03，上次建模的 dev 为 22.07-23.06，如果模型都看在最新 oot 上的增益，有以下两种方案。

时间窗更新，同样是用一年的样本，上次用的是 22.07-23.06，这次用 23.02-24.01，整体时间窗往后平移了 7 个月，才用的是更新的样本，通常来说取的在 oot 上更好的效果。

时间窗拉长，其实就是用 22.07-24.01 全部样本建模，可以理解为在上一版的 dev 上做了“增量”（这个比喻可能不太恰当，因为我们建模时没有区分 22.01-23.06 和 23.07-24.01 的样本），更大的样本量通常来说是对总体的更好的逼近。

样本上的增益（横向）还包括我们对 Y 的补充，事实上成功交易的样本只占到发起样本的一定比例，对于相当的样本，我们是没办法看到它的表现的，如果能够“推断”出这些样本的表现，我们的模型就学得更加全面，这种就包含了拒绝导轻样本（从其他资方观察其表现，但是会不会受到定价的影响未知，通常来说 API 样本的风险会比较高），如果是贷中样本，缺乏订单维度的 Y，是用历史的订单 Y（相当于聚合到了 uid）来替代。

此外还有横纵交叉的，就是补充近期的样本（暂无长期风险表现但是有短期风险表现，如果对于 3 期订单，其实 3 期表现和 6 期等价），可以考虑在原有模型上做增量。



### 3.6.2 特征增益

特征的增益更为直接，新的特征带来的数据维度的丰富、更为科学地从旧特征池做筛选都是可能的原因。

新的特征来自于两方面，外部数据源的更新迭代和已有数据源的特征衍生。

## 4 lightgbm 参数详解

在风控领域中,lightgbm 以其轻量级的优势(直方图切分、GOSS 单边采样、EFB 互斥特征捆绑)得到了广泛的应用,在实际应用中,有着众多可以调节的参数,下面对于一些重要的参数及其作用进行分析。

<https://lightgbm.readthedocs.io/en/latest/Parameters.html>

下面以 lightgbm.train 为例,开始我们的探索。篇幅所限,不会对所有的参数进行解析,重点在常用的参数及调参思路。

### 4.1 核心参数

1. objective 训练目标参数,对于原生接口,一般 binary 用于二分类, regression 用于回归,还有其他训练目标,不常用;
2. boosting 基学习器,默认 gbdt,还有 rf (随机森林)、dart (带 dropout);
3. data\_sample\_strategy 采样方法,默认 bagging 随机采样,可以使用 goss 提升训练速度;
4. num\_iterations 训练轮数即构建的树的数量,一般就可以设置一个较大的值,配合早停参数使用;
5. learning\_rate 学习率,影响训练速度,一般来说如果模型收敛过快 or 过慢可以调整,可以配合 callbacks 进行 shrinkage 调节;
6. num\_leaves 每棵树的叶片数量,太多容易过拟合,lightgbm 是 leaf-wise 生长策略,通常树深不超过 5,这个参数可以不用调节;

7. `num_threads` 进程数量，阈值为 `cpu 核数-1`，注意，如果设置不合适会拖慢速度，例如设置的值大于阈值上限；
8. `seed` 生成随机数的种子，优先级会低于其他种子。

## 4.2 训练参数

1. `max_depth` 树的深度，控制过拟合，与 `num_leaves` 存在联动；
2. `min_data_in_leaf` 每个叶片的样本量，控制节点分裂；
3. `min_sum_hessian_in_leaf` 每个叶片的 Hessian 和，涉及到损失函数二阶导的计算，这也是 GBDT 模型的重要改进；
4. `bagging_fraction` 每次训练时行采样比例，提升模型的鲁棒性；
5. `bagging_freq` 和参数 `bagging_fraction` 结合使用，如果要进行 bagging，则这个参数必须初始化为一个正值，这样每隔 `k` 轮，都会随机抽 `bagging_fraction` 比例的样本用于下 `k` 轮的模型训练；
6. `feature_fraction` 列采样比例；
7. `early_stopping_round` 早停轮数，可以给 `valid` 传递多个 `metric` 用于早停；
8. `early_stopping_min_delta` 早停需要的 `metric` 增益；
9. `lambda_l1` L1 正则项；
10. `lambda_l2` L2 正则项；
11. `min_gain_to_split` 每次分裂需要的最小增益；

**增益 (Gain) 的计算方法**依赖于损失函数（如平方误差、交叉熵等）以及二阶导数信息。LightGBM 使用基于梯度和 Hessian 的公式来计算每个节点分裂带来的增益。增益的公式如下：

**增益计算公式：**

$$Gain = \frac{1}{2} \left( \frac{G_L^2}{H_L + \lambda} + \frac{G_R^2}{H_R + \lambda} - \frac{G^2}{H + \lambda} \right) - \gamma$$

其中：


- $G$  表示当前节点中所有样本的梯度和（即一阶导数的和）。
- $H$  表示当前节点中所有样本的 Hessian 和（即二阶导数的和）。
- $G_L, G_R$  分别表示左子节点和右子节点的梯度和。
- $H_L, H_R$  分别表示左子节点和右子节点的 Hessian 和。
- $\lambda$  是 L2 正则化参数，用于平滑分母，防止过小的 Hessian 值导致分数不稳定。
- $\gamma$  是 `min_gain_to_split` 的值，用来设置  节点的最小增益。

图 1: Gain 的计算方式

## 5 风险模型建模复盘