客户ML case share 第5集

在我参与的很多计算广告的排序任务和推荐系统的排序任务的项目中（**这些项目的一个通用特点就是数据集都很大**（比如训练集经常都是几百GB以上）），经常会遇到和数据集相关的那些事：

**负样本的生成？**

**数据集的来源？**

**数据集的切分？**

**数据集的采样？**

**数据集的质量？**

**特征的线上线下一致性？**

**对于排序任务来说，最重要的是样本工程和特征工程，模型本身是次之的**。因此值得我们花费大量的时间来关注数据集的这些事。接下来我们就每个问题来展开讨论。

1. **负样本的生成**：

对于个性化搜索/推荐系统/计算广告的CTR/CVR/IVR预估这样的排序任务的**负样本生成问题**，常见有三种方案：

A. 全部曝光但没有点击/转化的作为负样本

B. 把曝光了且位于多个点击/转化的item或者广告素材位置之间的那些作为负样本

C. 对#A和#B折中，即负样本包括两部分：

一部分是#B的全部；

一部分是从最后一次点击/转化item位置之后的那些曝光但是可能没有被用户看到的样本中采样一些。

注意：**曝光并不表示终端用户看到了**（**这个是很重要的**，**也是很容易被忽略的**），这里又分两个情况：

1. **曝光位发生了填充**：

但是由于终端用户在屏幕上的滑动速度太快，其实他并没有看到这些填充的东西**。**

1. **分页的曝光位还没有被填充**（这里指的是超过显示屏幕外的那些曝光位）：

对于个性化搜索和推荐系统来说，假设结果的分页是在客户端侧做的话，搜索结果列表和推荐列表是单个请求的返回结果，这些是从推理服务侧发送给客户端侧的完整的曝光列表。那些不再当前显示屏幕中的需要下滑才看到的下个分页的item就是这里说的还没有被填充的曝光位。

而对于计算广告来说，每个广告位一般会单独发送一个请求，因此这个领域不存在这里提到的分页的曝光位没有填充的问题。

**在我接触的客户中，使用方案A的是绝大多数**（尤其是计算广告的排序任务基本都是用的方案A），但是正如上面讨论的，**其实这里可能会引入噪声**（对于计算广告的排序任务来说，上面的情况#1会带入噪声；对于个性化搜索和推荐系统的排序任务来说，上面的#1和#2两种情况都会带入噪声）:

**因为有一些负样本并不是真的是终端用户不感兴趣的，可能仅仅是因为他们没有看到而已**。

**在某些客户中，据说他们能从客户端的埋点日志中比较准确的判断终端用户是否看到了曝光的东西**（我理解这个判断逻辑可能会涉及到：屏幕的大小，曝光位的大小，手指滑动触屏的速度（太快的滑动速度其实会啥都看不到），手指在曝光位停留的时间（停留的时间越长可能表示看清楚了）等等）。

**如果能有比较靠谱的判断逻辑知道终端用户看了哪些曝光的item或者广告素材，这样获得的负样本可能更可信。**

1. **数据集的来源**：

关于数据集的来源，我遇到过**两个很典型的子问题**：

1. 数据集从多个不同的地方/公司收集
2. 数据集从多个不同的模型/策略收集

先看子问题#B，新模型一般都会经历如下几个阶段（**这里的新模型指的是完全不同的模型，不是同一个模型的不同版本**，比如老模型是xgboost模型，新的模型是wide & deep模型；或者新模型是DeepFM模型，老的用的业务规则）：

1. **起步阶段**：

因为需要有足够的积累的数据让这个新的模型进行**“第一次”训练**，所以这里把它称作新模型的起步阶段。

在起步阶段的话，一般需要用老的策略/规则或者老的模型来收集数据以供新模型使用。

1. **数据积累阶段**：

当新的模型上线以后（包括新模型之后更新的不同版本），通过在线上服务来收集新模型产生的数据。

那么**新模型的第二个版本可以利用的数据集就有了区别**：一部分来自老的模型或者策略/规则，另一部分来自自己产生的数据。

1. **重生阶段**：

当线上的新模型产生的数据足够多，并且特征的覆盖度足够好以后，这个时候可以考虑**用纯的完全新模型产生的数据来从头训练新模型**，也就是尽量让老模型或者老的策略/规则对新模型训练的遗留因素最少。

1. **成长阶段**：

重生后的新模型版本上线以后，通过自给自足的生产数据的方式来进行不同版本的更新。

通过上面介绍的四个阶段可以知道，**在数据积累阶段就会涉及到训练单个模型的数据集会从多个不同的模型或者策略来收集**。

对于子问题#A，**之前的一个客户XXX的数据集是来自三个不同的公司实体**，那么就面临如何使用这样的数据集的问题（具体上下文是这样的：一个计算广告的DSP业务，线上有三个模型做A/B test，三个模型分别来自三个不同的公司实体，客户XXX做其中一个模型，但是它能拿到其他两个公司对该业务每天新增的数据集）：

1. 使用全部三家公司的模型产生的数据来训练一个模型。

**这个也是该客户最开始使用的方法。**

1. 只使用自己家公司的模型产生的数据来训练一个模型。

**这里需要考虑的因素如下**：

自己家公司的线上模型产生数据集的速度？

自己家公司的模型在线上A/B test分配的流量比例带入的bias？

1. 同时用1）和2）中的模型线上做A/B test。

**这个也是该客户目前使用的方法**。

**通过对两个子问题#A和#B的阐述，我们能发现这样的数据集本质上不是“同质“的，都是混血的。从直觉上看，它们的数据分布会不一致，因此用这样的数据集直接建模可能对模型效果有不好的影响**。

因此对于问题#B，可以通过新模型的重生阶段和成长阶段来规避；对于问题#A，可以参考A/B test的结果做调整：

**只要自己家公司模型产生的数据集足够多，并且特征覆盖度也足够好，那么可以尝试给用这样的数据集训练完的模型更大的线上流量，更大的线上流量可能会得到更高的特征覆盖度的数据，从而形成良性循环**。

1. **数据集如何切分**？

**数据集切分又常遇到的两种问题**：

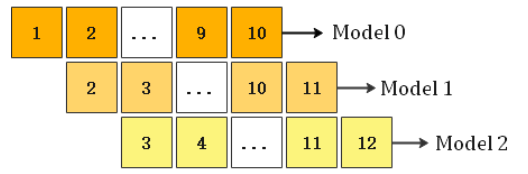
1. 验证集和训练集如何切分
2. 单个数据集如何切分为多个数据集

我们先来看问题#A，在做计算广告的排序模型和推荐系统的排序模型的项目中，发现绝大部分的客户的训练方式就是如下典型的2种方式（这里不考虑有些公司采用的小时级别增量训练结合T+1训练的方式）：

**T+1训练方式**

**每天增量训练方式**

而T+1训练方式占据绝大多数，因此这里的数据集切分就只是讨论这种方式及其变体。参考如下图：



正如上图所示，所谓的T+1训练方式有几个要素：

**全量T天的训练集从头开始训练**；

**紧接着的那一天作为验证集**；

**基于滑动窗口的训练，滑窗为1天**；

**每天都会更新线上模型**。

**虽然排序任务并不是像时间序列预测那样的强时间依赖的任务，但是最常见的T+1训练方式中的T天的训练集和1天的验证集的边界还是按照时间先后顺序来切分的**。

我见过有的客户把8天的数据集随机按照7:1的比例分为训练集和验证集，这个不是建议的方式。

两个思考的点（也是**客户经常问到的问题**）：

**T取多少天比较合适**？

**验证集1天合适吗**？

对于上面的两个问题，会涉及到**数据集的规模问题**。选取几天的数据作为训练集和几天的数据集作为验证集没有什么黄金法则，一般都是所谓的经验（**在ML中 ，太多的东西没有什么定律，就是靠靠谱的直觉以及不断尝试得到的经验**）。在选择的时候下面的几个因素是一定需要斟酌的：

**训练集中的正样本的个数**

**验证集中的正样本的个数**

**工作日和非工作日对于业务的影响区别很大吗**？

之所以关注训练集和验证集中的正样本的个数，是因为对于计算广告和推荐系统的排序任务，点击事件和转化事件相对于曝光事件就是小概率事件，因此这样的数据是非常稀疏的，太少这样的正样本对于模型的学习和评估都不好。

**那到底多少比较合适呢**？

这个没有什么法则，我之前建议过一个客户的验证集的正样本至少1K以上，训练集的正样本至少1W以上。

**那如果1天的验证集的正样本个数很少比如不到1K，怎么办呢**？

这种情况就可以使用**T+N训练方式了**，N就是验证集的时间窗口，为了让验证集的正样本够量，可以把窗口N拉长到大于1。

同样的道理，对于常见的T取值为7的情况，如果你发现7天的训练集的正样本很少比如小于1W，那么把窗口拉长到超过7天是建议的方式。

**对于上面提及的工作日和非工作日对业务的影响是否很大，上下文是这样的**：

这个源自常见的T+1训练方式并且T是7天的情况。连续的7天基本上会涵盖工作日和非工作日，而1天的验证集则有时候是工作日有时候是非工作日。这样的情况下，**如果仍然使用T+1方式训练的话，就要考虑是否是工作日对业务的影响**：

基本对业务的影响是相似的：

这种情况的话，仍然用常规的T+1方式训练。

**对业务的影响差别比较大**：

这种情况的话，就需要考虑验证集的选择了：

1. 选择一：**尝试把验证集窗口拉长**，最好能至少包括将来的一天工作日和将来的一天非工作日的验证集数据。
2. 选择二：**工作日和非工作日单独建模**，用两个模型来分别处理工作日的数据集和非工作日的数据集

**两种选择各有优缺点**：

选择一的优点是只需要维护一个模型，训练集中的特征覆盖度可能会更好；**它的缺点是验证集不好构造**，如果仍然用1天做验证集那么模型效果很可能不好。

选择二的优点是把两个很不一样的数据分布做单独建模，模型效果可能不错，而且验证集容易构造；缺点是可能训练集中的特征覆盖度不太好，而且训练集和验证集的数据的时间维度会间断，而且需要维护更多的模型。

综上所述，在这样的情况下，**尽量用工作日和非工作日单独建模**，其实这个就是问题#B（在一个客户XXX的项目中，也通过实践证明了拆开的效果比合在一起的线上效果好。）

如上所述，上面这个把数据集（包括训练集和验证集）按照工作日和非工作日切分为2个数据集是基于时间维度的。

理论上来说，**基于任何的维度来切分单个数据集为多个数据集都是可以的，只要模型效果持续的好那就是真的好**！

在我参与过的项目中，**还有一种常见的切分数据集为多个数据集的情况是按照地域来切分，或者说每个地域/国家单独一个模型**。

这个其实也可以归结为建模思路的范畴，我的一个客户YYY就验证过，当按照国家尤其是那些数据集足够大并且特征覆盖度足够好的国家来建模，就这个国家的业务线上表现的话，单独建模要比把所有国家的数据联合建模的线上效果好！

1. **数据集的采样**？

对于个性化搜索/推荐系统/计算广告的排序任务来说，主要大的挑战就是每天的正样本（点击样本或者转化样本）相对来说比较少，而负样本则每天都是海量的。那当我们使用T+1训练方式（T经常选择7天）的时候，面临如下的几个问题：

**是否需要对负样本做单独采样**？

对于这三大领域的排序任务来说，负样本和正样本的比例会非常大（比如经常是几千倍的比例），因此为了让模型学习的好一点，经常需要做一些样本类别不均衡的处理，其中一种常见的处理方法就是**下采样**，就是从负样本中采样一定比例的作为最终的训练集中的负样本。（我有某客户ZZZ就是对负样本进行了采样，把负样本与正样本的比例从原始的几千倍变成了10倍）

注意：

1. **如果要做采样，也是针对训练集中的样本进行采样，验证集中的样本需要全部保留**。因为需要让验证集尽量和线上的数据分布保持一致（这里主要指的是验证集中的正负样本的比例）
2. **对于计算广告的排序任务，如果训练集中的负样本进行了采样的话，最后用的ecpm的排序公式中的pctr需要校准**（**因为pctr\*bid的值可能与采样前不同**）；对于个性化搜索和推荐系统的排序任务，即使训练集的负样本进行了采样，也不需要在最后排序的时候进行校准。

**那具体如何对负样本来进行采样呢**?

**每天按照固定采样率做等量采样**

**按照时间的权重来增加采样**

在确定整体采样比例情况下，根据总的负样本量计算出需要采样的量，然后计算每天的采样比重。需要满足的条件：离当前越近，采样比重越大。

**即使做了负样本采样后，总的数据集还是太大**？

业界的一个共识是，数据集越大，需要的模型就会越大/越复杂，训练这样的模型需要的成本也越高。

因此这里可能会想，是否把训练集窗口缩短为比如5天来减少总的数据集的数量吗？这个不是建议的方式，**业界常见的做法还是仍然用7天的数据做训练集，然后用分布式训练来缩短训练时间**。

除了上面两个通用的问题，在计算广告领域还有一个常见的**转化延迟问题**：也就是转化日志上报时间比较晚，比如某些广告当天实际上发生了转化，但是DSP侧要过几天才能拿到这些广告的转化日志，因此就会发生一些负样本其实是正样本的问题。（这个情况几乎是通用的）在这样的情况下，要对CVR预估/IVR预估来建模的话，就**需要权衡样本的正确性和时效性**了：

**完全剔除转化延迟时间窗口内的所有数据，在当前不作为训练集和验证集的一部分**：

这个方法能保证样本的label的正确性，但是因为没有近几天的数据作为训练集和验证集，因此数据的时效性差一些，可能对模型的效果有影响。（这个方法在腾讯2017年计算广告APP CVR转化率预估的决赛TOP10获奖团队的方案中，基本都采用的这个方法。）

注意：**在之后拿到转化延迟的数据后，记得重新给之前对应的同一个样本重新修改label（实现时每个样本可以设置唯一id，然后通过id来对齐）。**

**不管转化延迟问题**：

这个方法主要考虑的就是数据的时效性，不浪费最近新增的数据，但是有些真实的正样本就被wrong label为负样本了。（我接触的DSP客户基本大家都用这个方法）

思考：如果因为转化延迟导致的有问题的样本占的总的负样本的比例很少比如10%，那么是否还需要关注这个问题呢？

**那是否可以对上面两个选择做个折中呢**？

假设：CVR预估建模，用T+1训练方式（T取值为7天），转化延迟时间窗口是3天。那么数据集可以这样来选择：

对于验证集：离当前最近1天的所有正样本和负样本

对于训练集：验证集那天前推7天的所有正样本 + 验证集那天往前推2天的负样本做采样 + 最老的5天的负样本。（注意这里做了采样，在DSP排序任务中使用ecpm公式的时候需要校准pcvr，**采样率按照采样后的负样本数量除以不采样的负样本总数量得到**）

1. **数据集的质量问题**？

数据集的质量是个很大的话题，包括但不限于如下的几种（**我认为以下四个是最重要的**）：

**异常样本以及异常特征的识别和处理（线下必须做）；**

**Label的正确性问题（线下必须做）；**

**特征的覆盖度问题（线下必须做）；**

**样本和特征的正确性检查（线上尽可能做，线下必须做）；**

这里就每个问题在客户项目中遇到的情况来讨论一下：

1. **异常样本以及异常特征的识别和处理**：

用数据集来训练模型，我们希望送入模型的数据是干净合理的，因此**首要的任务就是把原始的数据集进行清洗，把异常样本和异常特征识别出来并做相应的处理**。

异常样本和异常特征的识别属于异常检测的范畴，经常用统计方法，数据分析或者机器学习的方法来识别异常（**三者结合起来做异常检测的效果最好**）。

异常点的归因是特征粒度即异常特征的话，经常用的处理方式就是当做缺失值处理；异常点的归因如果是异常样本的话，常见的方式就是从该数据集中剔除该异常样本（**剔除并不是丢弃，建议的方法是把异常样本单独存储以便之后做异常检测建模**）。

**项目中遇到的几个bad case**：

客户XXX，通过数据分析的方式发现数据集中每天某个时间段的某几个设备ID会有非常多的点击数据，远远超过每个设备每小时平均点击次数。

之后通过把这些有问题的样本从数据集中剔除掉并训练模型，得到的模型的线上效果有明显提升。

客户YYY，通过样本抽查的方式发现很多样本的转化归因有问题（**转化归因会最终影响用户交互时的广告侧与媒体侧的特征配对**）。

之后通过对有问题样本的重新转化归因来调整样本集并重新训练模型，离线效果和线上效果都明显提升。

客户ZZZ，进行常规的连续型特征的取值范围检查，发现数据集中的历史CTR特征的取值非常离谱，对其进行矫正后，离线模型的效果有提升。

强调：项目中发现很多客户都是在发现模型效果（包括离线效果和线上效果）不理想或者在训练过程中遇到一些错误时才会去检查样本和特征，这个是不建议的做法。**最建议的做法还是在开始训练之前对数据集的质量进行严格检查**（**可能这样做的话，前期的投入是比较大的，但是只有这样，之后训练完的模型的离线效果才更可信**；否则遇到问题再回朔来检查数据集的做法看似投入小，实则是定时炸弹，你永远不知道它什么时候爆炸！）。

1. **Label的正确性问题**：

**Label就是模型学习的指导信号，所以保证label的正确性是显而易见并且非常重要的。**但是保证样本的label的正确性不是看起来那么容易的，在搜推广三大领域的排序任务的数据集中经常会遇到如下几种情况**：**

**转化延迟**：

这个主要是计算广告的DSP的CVR/IVR预估模型的数据集中会遇到；前文已经详细讨论了这个问题，这里不再赘述。

**误点击**：

由于终端用户不小心点击item或者广告生成了点击日志。

比如在客户XXX的DSP项目中，客户的开屏广告（开屏广告就容易让人误点击）日志中发现了很多疑似的误点击。

**点击进去后才发现并不是真正的感兴趣**：

这个情况在推荐系统和个性化搜索领域中比较常见。

比如在长视频/电影推荐中，某用户对某个推荐的item点击进入并观看了5秒就关闭，那么可以认为该用户对这个item并不感兴趣，因此这样的点击对应的样本的label需要标注为0而不是1.

**是否re-label的问题**：

使用T+1训练方式时，经常会遇到同一个设备id或同一个userid对同一个item或者广告素材在多次曝光行为下会有不同的点击行为或者转化行为。

由此带来的现象就是在比如7天的训练日志中，除了时间戳，会有其他特征完全一样的样本出现，但是前几次label为0，最后一次label为1。这种情况下，是否需要把前几次的label修改为1，也就是所谓的re-label问题（这是来自一个真实的客户XXX的问题）。

在这种情况下，建议的处理方式：

1. **不要对反复曝光的样本做re-label，也就是遵守该用户的真实点击行为或者转化行为就可以**。

这个情况在很多客户的项目中都出现过，甚至有的客户说，他们做过统计，item或者素材被点击需要平均曝光7次；转化的话需要平均曝光20次。**从这个角度来说，太严格的过滤模块不一定对业务就好**：比如对于某个用户，在召回的结果中是否应该过滤掉该用户最近三个月曝光过或者点击过的item或者广告素材。**太频繁的曝光肯定是需要过滤的（否则对终端用户不友好），这个频率需要具体根据业务来确定**。

1. **可以增加一些其他的特征来让样本有辨识度**。

比如可以利用时间戳来把一天分为24个桶作为一个离散特征；还可以把对应userid/设备id最近半年的安装列表作为一个强特征加入模型。

1. **特征的覆盖度问题：**

特征的覆盖度（**离散特征和连续特征都有覆盖度的问题**）尤其是训练集中的特征的覆盖度，**最终会反映到模型对该特征能了解的程度**。

理想情况下，**我们希望训练集中的每个样本都没有缺失值，并且训练集会把每个特征的所有可能的取值都能覆盖到并且每个取值出现的频次足够高**，这样模型学习的才够充分，之后对验证集以及线上数据做推理才更准确。

现实中，每个特征总是或多或少会有覆盖度的问题。离散特征的覆盖度问题更复杂和常见，因此我们这里主要讨论离散特征的覆盖度问题（**连续特征通过特征归一化或者离散化可以把它的覆盖度问题缓解或者转移**）。而导致特征覆盖度问题的经常会有如下的2种情况：

特征获取困难；

低频的特征值被过滤；

1. **特征获取困难**：

在项目的过程中，经常有很多特征的获取是比较困难的。具体又分为几种情况：

**涉及到隐私的特征**；

这类特征比如身份证号，职级，薪资水平等等是比较隐私的，一般终端用户不会提供给你或者就是提供给你，你也可以认为不是真实的信息，基本可以忽略。

**特征值部分缺失；**

如果某特征的部分值由于某种原因不能获得，那么就当作缺失值来处理。

如果训练集中的某特征的缺失比例很大比如超过60%，那么可以考虑是否剔除该特征。**这也是客户经常遇到的情况，是选择填充缺失值还是简单剔除该特征**？（这个最好还是通过线上AB test来看效果。）

**通过其他模型来对某些特征来建模预测；**

有些特征比如男女性别，年龄段，是否有孩子等等这样的特征对于模型来说比较重要，但是这些比较难由终端用户提供真实的信息。因此有的客户就通过规则/数据分析/机器学习的方式来根据该用户的行为来生成这些属性，也可以认为是该用户的画像。

显然，本身这个特征的正确性就没有办法保证，那么把这个生成的特征放入下游的排序模型中，对模型是有帮助还是有负作用是需要离线评估以及线上AB test来评价的。

客户XXX就在一个计算广告的DSP的排序任务中，通过他们的数据平台来对用户进行男女性别和年龄段的推测，并且把推测的结果作为特征用在排序模型中。

1. **低频的特征值被过滤**：

就是把包含长尾的特征值的整个样本从训练集中干掉了。

对于搜推广三大领域来说，有很多id类的特征，而且像userid/deviceid和itemid这样的特征是典型的高维稀疏特征，因此长尾的id类特征很常见。

那么这些长尾的id类特征常见的处理方式有如下三种：

**过滤掉低频特征的样本**：

**这个方案是我见过的客户用的最多的**。其最简单的方式就是设置一个频次阈值比如8，少于这个频次的样本就不放入当前这次训练集（下一次训练的时候可能这个id的样本的频次够了，所以低频样本不要丢掉，还是需要保存起来的）

优点：

可能会避免了模型对长尾id样本的过拟合

缺点：

需要有相应的逻辑来判断是否是长尾id，因此做线上serving时会带入额外的延迟。

**线上serving时，对于出现的长尾id还需要特殊的处理**（比如走单独的链路，不走模型）。

**使用other-id的方法**：

这个方法同样需要设置一个频次阈值，把低于这个频次的id值替换为other-id，也就是所谓的长尾id和冷启动的id共享一个other-id值。

优点：

因为other-id的存在，线上不会出现id OOV的情况。

缺点：

需要维护一个字典做长尾id到other-id的mapping，并且这个字典会根据训练集中的id的频次变化需要经常更新。

线上推理时，同样需要做这个mapping，带入额外的延迟。

**不考虑频次并使用hash trick的方法：**

对id类特征不做频次过滤，直接对id类特征做hash bucket映射。

优点：

不需要额外的字典来映射；线上不需要处理id OOV的情况；

线上处理逻辑相对比上面两种方案更简单。

缺点：

可能会有hash冲突；

对长尾的id的推理效果可能不好；

由此可以看到，**如果特征的覆盖度问题是因为低频特征被过滤后导致的，那么可以考虑：**

特征的频次阈值是否可以根据业务特点来重新调整；  
重新评估并权衡其他的长尾id类特征的处理方法（可能评估完后会仍然使用低频特征被过滤的方案）。

1. **样本和特征的正确性检查**：

**这个看似最简单的事情，但是确实是最容易忽略，也是经常出问题的地方**。

**我见到的客户中，或多或少都会对样本和特征做一些检查，但是仍然检查力度太少**。

先不说对模型学习效果如何，就是出了问题debug也费劲（尤其是当出现奇怪的问题或者复杂的问题的时候）；而且经常会前功尽弃（比如训练到几个小时因为样本有问题报错退出），对成本带来无谓的消耗。

因此，我建议**至少做如下的正确性检查**：

**样本级别的检查**：

比如tfrecord文件本身的检查，是否包括corrupt的样本；

我遇到过一个客户XXX在使用sagemaker pipe mode训练的时候，训练了挺长时间，后来因为某个tfrecord文件有问题导致整个训练任务失败。

**特征个数的检查**：

**检查每个样本包含的特征的个数是否是期望的个数**。

我遇到过一个客户YYY，在跑训练的时候总是遇到奇怪的报错。客户自己手动抽查了几个样本发现没有问题，后来对其中一个文件逐行肉眼检查发现有些样本的特征个数不对。

**特征类型的检查**：

**检查文件中的每个特征是否是我们期望的类型**（比如整数，浮点数，字符串等等）

**特征的取值范围检查**：

检查连续性特征是否是0，-1，NAN等特殊值；

检查连续性特征是否在期望的取值范围内；

检查离散型特征是否在期望可枚举的范围内（可通过维护一个字典来检查有哪些可枚举的离散特征）。

**单个特征的shape检查**：

**检查每个特征是否与我们期望的shape是一致的**

**多个特征的shape的对齐检查**：

比如一个安装列表的序列特征，填充后的长度是固定长度50；为了让填充的id不对训练造成影响，一种简单方法是提供序列中每个id的weight，填充的id的weight设置为很小的一个值比如0.000001（坑：这里设置为0的话，在TF中使用weight feature column的时候会报错！另外一个坑，weight以后使用share embedding column的话，TF1.X是可以的，但是用TF2.X版本用类似的API tf.feature\_column.shared\_embeddings就会报奇怪的错误，TF2.X使用tf.compat.v1.feature\_column.shared\_embedding\_columns API则是可以的）。因此这里需要检查每个样本中的该序列特征填充的id个数和对应的weight为0.000001的个数是不是相等。

**Label取值的检查**：

**检查每个样本的label的取值是否是期望的范围**。

**建议**：

1. **每个公司对每个数据集都建立和维护一个checklist**.

这个checklist就是对这个数据集的所有的卫生检查。当在日后的迭代中发现了新的检查规则后就加入到这个list中。

1. **对样本和特征的这个卫生检查，建议不要放在训练脚本中去做，而是用一个单独的前置任务去做这个事情，不要浪费宝贵的训练资源**。

在之前的很多项目中，发现很多客户可能为了省事，喜欢把这些检查还有一些统计样本个数等的事情放在训练脚本中。这个不是建议的方式。

1. **对特征的检查，线上和线下都需要尽可能做**。

对于线上的检查来说，尤其是特征取值范围的检查，如果发现不正确，需要根据策略比如使用缺省值来替换。

**总结：**

**线上和线下都应该尽可能对数据集的质量进行检查。线下的时候，这个检查并不只是发生在数据集清洗的开始阶段，其实只要对数据集进行了修改（比如在异常样本/特征的识别和处理后增加了新的交叉特征），那么都需要再做一遍质量检查的。**

1. **特征的线上线下一致性问题**？

在排序任务中，经常会遇到离线评价指标很好，线上评价指标不好的情况发生。而这个**线上线下不一致的一个很常见的原因就是因为特征的线上线下不一致**。

**特征的线上线下不一致，指的是线下训练时样本中的特征的特征值可能会发生变化，并不和该样本在线上生成时的特征值完全一样**。

**这里有个例外**，如果是因为在数据集的质量检查中，探测到有异常特征，应用异常特征处理后（比如用缺失值填充方式）导致的特征的线上线下不一致的话，这个是可以接受的。

在我参与的很多项目中，很多客户是线上特征处理一套逻辑，线下的特征处理又一套逻辑。在不经意间，就很容易发生线上线下特征不一致的情况。

**当前业界建议的方式**：

**线上的时候做特征的获取和拼接（甚至某些特征的生成），并且把拼接得到的特征向量异步落盘（线上落盘的特征向量组成的日志就是原始数据集的特征部分）；**

这样做对线上的系统要求比较高，尤其是对实时或者近实时特征的生成和获取。

**离线训练的时候，把线上落盘的特征向量日志和反馈的label对齐并join得到原始的数据集，然后对原始的数据集进行数据集质量的检查和处理后得到最终的数据集**。

**这里并不是说离线的时候就不需要对数据做额外处理了**。比如一些统计类的数据（比如历史上最近1天的某广告素材的点击次数和曝光次数，某个连续性特征的均值/方差等等，**这些数据会作为meta data用在线上特征的生成**），仍然需要离线做统计，得到的结果存到feature store，线上的时候从feature store获得这样的特征再做处理。**也就是说现在这样类似的离线处理是为线上做准备，而不是为线下训练做直接的准备**。

注意：

这里强调的**特征线上和线下一致性的前提是特征稳定和模型稳定。**

**模型稳定：**

指的是**新模型的成长阶段**，这个阶段新模型的数据完全可以自给自足；而其他阶段的话，离线特征处理和线上特征处理一般还是分为两套的。

**特征稳定：**

**指的是特征的field size不变**。如果需要修改特征（比如增加了新的交叉特征，或者删除了某个特征），那么离线特征和线上特肯定是不一致的。