判定一个用户是否流失有2种思路：

1. 设置一个时间锚点，只要用户最后一次关键行为的时间戳距离时间锚点的长度**大于等于**设定的某个固定时间长度，就认为该用户是流失的。
2. 设置一个观察期时间窗口，只关心用户在观察期时间窗口内是否有关键行为，没有的话该用户就是流失的。（**这个思路可能用的更多一些，我们下面的讨论都是根据这个思路来的**）

关于用户流失/留存预测，用机器学习来建模的话，需要重点考虑下面几个问题（[**https://www.zhihu.com/question/68029067**](https://www.zhihu.com/question/68029067) **这个link中讲解的很清楚**，可以参考，强烈推荐阅读一下）：

**如何定义一个用户是流失的？**

**模型训练好以后，如何使用该模型来预测用户的流失**？

**数据集的时间窗口选择问题**？

**训练集和验证集如何切分**？

**用户流失预测模型的训练迭代问题**？

**关于用户流失预测场景下，强特征的挖掘**？

**在得到了针对用户的流失预测概率以后，如何利用这个概率**？

下面针对每个问题，我们来详细解释：

1. **如何定义一个用户是流失的？**

先解释一下流失周期，它定义的是某用户没有发生某种关键行为的最小时间间隔（也就是说用户已经流失的时间是大于等于流失周期的）

不合理的流失周期的划分会造成预测准确率低下（**因为一定存在回流的用户**），我们需要不断尝试周期划分，才能更为准确地预测流失。在这种情况下，选择恰当的定义方法显得至关重要。

对流失比较经典的定义是“**超过一段时间间隔未进行关键行为的用户”**，**关键点在于如何界定这个时间间隔（即流失周期）和关键行为（即流失行为）**。不同的业务场景，这个关键行为一般会有区别。

**对于游戏场景下用户的流失，可以选择“有效登录”这一行为作为是否流失的关键行为**。

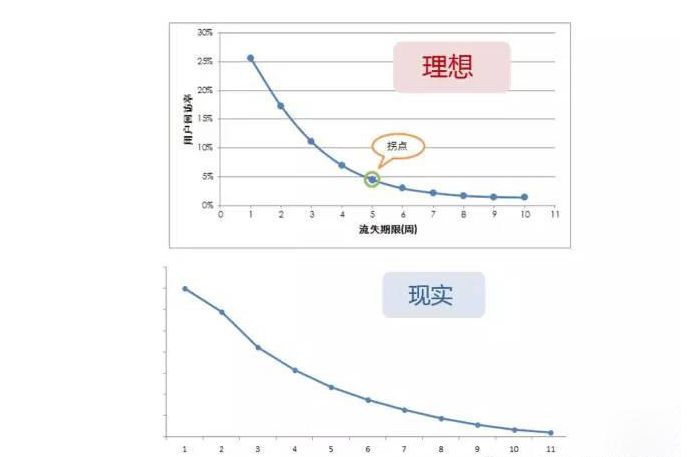
注意：**这里需要考虑如何判定用户为有效登录**。如果登录后马上就退出可能就应该不算做有效登录；或者登录以后马上把这个应用切换到了后台来挂机，这种也不应该算做有效登录。也就是说，**应该把用户有登录动作，并且把应用放在前台，且有足够长的停留时间（比如10分钟）才认为是一次“有效登录**”。

**对于流失周期的界定，常见有两种方法**：

1. **分位数界定法**

**可以使用90%分位数的方式来选择流失周期**。也就是根据用户最近连续两次关键行为（比如登录）的时间间隔从小到大来排序，90%分位数对应的时间间隔就作为流失周期。

1. **拐点界定法**（**可能应该优先用这个方法**）

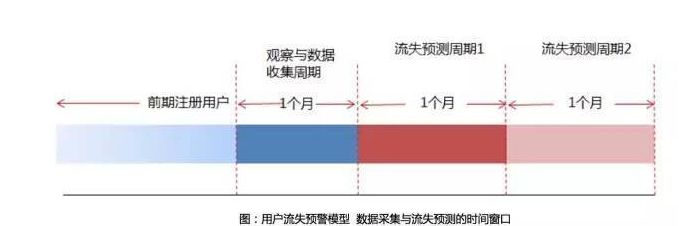
可以参考下图:  


注意上图中理想情况中的那个图，每个流失周期对应的用户回访率的计算需要计算流失用户数和回访用户数（用户回访率 = 回访用户数 ÷ 流失用户数 × 100%），具体计算过程参考如下：

流失用户数计算比较直接，根据流失周期设定观察期窗口的开始和结束位置，在窗口内没有进行登录的用户就认为是流失（**这里为了方便的对比不同的流失周期，观察期窗口的开始位置设置为相同的，不同的流失周期得到不同的结束位置**）；但是**link** [**https://www.zhihu.com/question/68029067**](https://www.zhihu.com/question/68029067)**中没有明确说明如何计算回访用户数**，**我理解应该会首先定义一个比较长的周期**，对于这里的例子来说，因为横坐标中有数据点的最大流失周期为10周，这里就把这个用户**回流周期**也设定为10周，对于流失周期是1周并且被判定为流失的用户如果在后续的10周内只要有有效登录就认为是回流用户；同理，对于流失周期是10周并且判定为流失的用户在后续的10周内只要有有效登录就认为是回流用户。（注意：**我这里是假设对不同的流失周期使用了一致的回流周期，这样设置是否合理以及回流周期具体多长，最好与业务运营人员确认**。）

**如果曲线没有拐点的话，也可以依据产品经验来拍脑袋**，一般产品的回访率5%-10%（通过固定了回访率然后在曲线找到对应的流失周期），不管划分多长的时间周期作为流失周期都会存在回访，误差不可避免。

1. **模型训练好以后，如何使用该模型来预测用户的流失**？



如上图所示，这里设定的流失周期是1个月，那么在准备开始训练的时间点往前推一个月这个时间窗口就是用来观察 **我们****感兴趣的“老用户”** 在这个时间窗口内是否有有效登录行为，如果有的话，表示该用户是留存用户，如 果没有的话说明该用户有超过1个月时间没有有效登录行为了，标记为流失用户。

**模型训练完以后就可以马上针对在观察期窗口内有关键行为的那些用户进行预测**，预测他们从训练结束的时间点开始往将来外推一个时间段来看他们是否有关键行为。这个外推的时间段就是流失预测周期，流失预测周期应该设置为与流失周期一致，因为我们要尽量保证预测时和训练时一致（也就是说，训练时打label给的观察期窗口的长度需要和预测时给的外推时间段一样长才合理）。正如上图中看到的：

在第一次流失预测周期窗口的开始处预测的是观察期窗口内有关键行为的那些用户在第一个流失预测时间窗口内是否有关键行为；第二次流失预测周期窗口的开始处预测的是在第一个流失预测时间窗口内有关键行为的那些用户在第二个流失预测时间窗口内是否有关键行为，以此类推！**我们需要关心的用户是如下三种**：

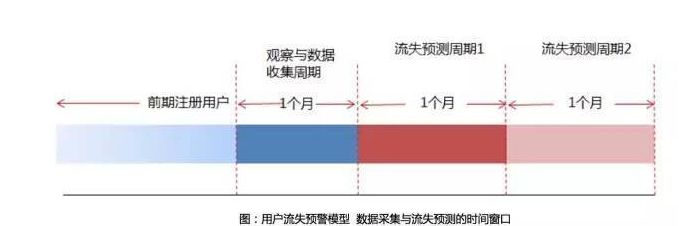
1. 从新用户转化为老用户的这群人；
2. 之前属于留存用户的这群人；
3. 刚回流的用户的这群人。
4. **数据集的时间窗口选择问题**？

在流失周期确定以后，我们需要来选择数据集的时间窗口。这里的时间窗口又包括两个（**这里的时间窗口并不是那么直接且显而易见的**）：

1. **从哪天开始收集用户的登录日志**？
2. **所有用户还是老用户**？

**一般流失预测更多针对的是老用户，而新的用户一般需要时间成长为老用户**。而这个新老用户的判定条件也是根据业务来具体设定的，比如累计玩游戏时间超过1个月的新用户就可以认为是老用户了。

1. **需要纳入数据集来训练的最早的时间如何选择**？



那对于上图提到的前期注册用户也就是我这里说的“老用户”，这个时间窗口往前推多少合适呢？我觉得主要有两个原则：

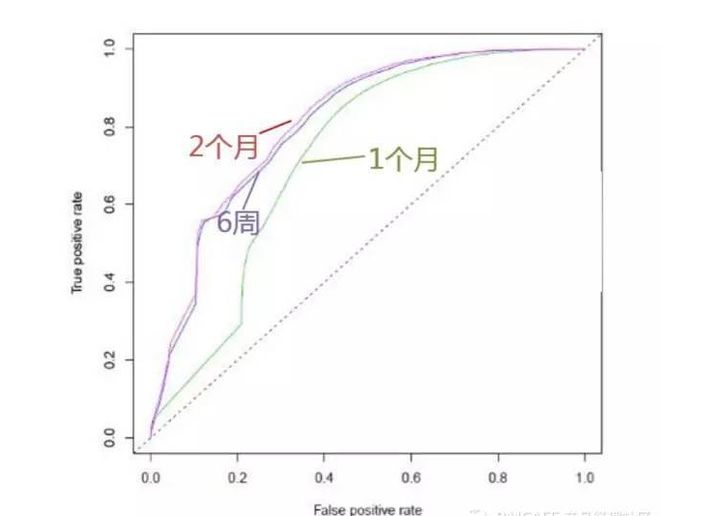
**数据样本要足够**（尤其是正样本也就是流失用户的样本要足够）；

**前推的时间窗口不要太长**（前推时间窗口太长的话，会把一些都已经流失很久的老用户的样本也纳入模型来学习，可能因为时效性等因素这些已经流失很久的老用户并不是我们这个流失预测模型所关心的人群，当然从对流失用户做归因分析的角度来说，这样的数据同样有意义，但那是另一个话题。**有的业务可能只是关心比如从观察期窗口的开始位置往过去推一个月**，看这一个月活跃的用户是否在观察期窗口内有关键行为，也就是前推窗口就固定设置为1个月 ，那这样得到的数据集可能太小，尤其是流失样本太少，**这个时候可以使用多个观察期窗口来得到更多的流失样本**）

1. **需要统计用户多长时间内的行为**？

这个也没有什么定论，**这个时间段是观察期窗口之前的用户last-login最近一次登录时间戳往前推的一段时间**。

**一个方法就像这个link** [**https://www.zhihu.com/question/68029067**](https://www.zhihu.com/question/68029067)**中提到的，通过对多个统计周期的特征来分别建模，然后在验证集上看ROC面积哪个更大**，如下图所示。



正如图所示6周的数据明显优于1个月（曲线右下方面积越大预测准确性越高），而2个月的数据只略优于6周，幅度有限，且时间成本较大，因此选择6周作为数据提取的周期。

注意：之后在进行用户流失的离线预测的时候，会包含那些从新用户转为老用户的人群，因此不适合把这个统计时间窗口弄的很长（由于这批人只有相对比较短的时间内有行为，需要把这个统计时间弄的不能太长，因为太长的话对这批人不公平）。

这里举个例子比较容易理解，比如流失周期是1个月，当前是在3月1日准备进行训练，那么整个2月份的时间窗口是需要预留出来的，数据集中的某个特征比如统计用户离他最近一次登录往前推1个月的累积消费金额。假设某个用户的last -login是1月1号**（注意这个last-login一定是在观察期窗口之外的**），那么这个统计时间窗口对于这个用户就是去年的第12个月（**每个用户的行为统计时间窗口都不一样，因为是按照每个用户的last-login往前推固定长度的时间段来计算的**）。

1. **训练集和验证集如何切分**？

对于用户流失预测这样的场景，可能不需要按照时间线来切分训练集和验证集，因为用户流失预测这样的场景可能并不是一个时间依赖相关的任务，那么对这样的数据集就可以按照比例来切分（**具体考虑到流失/留存类别样本的不均衡，这里做训练集和验证集切分的时候最好做分层按比例切分**）。

当然**对于用户流失预测任务，按照时间线来构造训练集和验证集也是可以的**。举个例子，比如训练集取的是4月1号到4月30号的这些活跃用户，这些用户在5月1号到5月7号这个窗口内是否有关键行为作为他们的label；验证集取5月1号到5月7号的活跃用户，这些用户在5月8号到5月14号这个窗口内是否有关键行为作为他们的label。

1. **用户流失预测模型的训练迭代问题**？

针对这个场景来说，模型的训练迭代可以选择的方式是：

**每隔一段时间做全量数据重新训练**；

**增量训练方式**

如果用增量训练方式的话，每个训练周期的用户的知识可能会学习到（**假设模型的容量足够大**）；如果每隔一段时间做全量数据重新训练的话，虽然可以通过把用户的旧的先验知识作为特征建模进来，但是即使这样也可能对用户旧的知识的学习没有增量训练那么好，而且每隔一段时间做全量数据重新训练更强调的是最近时间段的数据对模型影响更大，对于用户流失预测场景的话，可能这个因素不重要。因此我觉得可能用增量训练方式对这个用户流失预测场景更合适。

另外，**用增量训练还是全量训练和选用的模型还有关系**：像Xgboost，Lightgbm这样的boosting方式的模型，不是很适合做增量训练，他们的模型内涵更适合做全量数据训练（具体解释可以参考：<https://www.zhihu.com/question/343510267> ，<https://zhuanlan.zhihu.com/p/89285046> ）；如果用深度模型的话，天然支持增量训练。

如果选择增量训练的方式，增量训练的频率对于用户流失场景可能不太适合按天来增量做，因为需要等待一定的天数来积累新的样本。增量训练频率（或者我们叫训练周期）和流失预测周期没有什么必然联系，不过把增量训练周期设置为和流失周期以及流失预测周期一致的话，数据处理可能更方便一些。

如果选择全量训练的话，对于用户流失预测这个任务，可能每过几天或者一周来进行训练是比较合适的。

1. **关于用户流失预测场景下，强特征的挖掘**？

**可以把一些能通过业务经验识别到的一些流失影响因素，或者作为特征建模到模型中，或者把这些因素影响到的样本从样本集中剔除**。

对于像产品功能更新了，或者年底积分折半了这样的小概率事件发生时，需要尽可能分析是否这些因素导致了用户的流失，但是**这样的小概率事件是否发生一般不作为特征放入模型，因为这个特征在每次训练的时候基本上都是一个值**（比如对于产品是否升级这个特征，可能在大多数训练的时候，这个特征是0，对目标任务没有什么辨识度，也就是和目标任务不相关）。

对于像比如游戏用户投诉次数这样的特征对于用户流失是一个强相关的特征，这样的特征需要多挖掘一些并放入模型中建模。

前面的讨论中，我们提到了把用户最近一次登录往过去推一段时间的行为统计作为特征来建模。除此之外，**有些业务可能更看重用户最近一段时间的活跃行为，它更能反应出这个用户近期的行为特点，如果这个特征不稀疏的话它是个强特征**（举个例子，5月8号那天要训练，取4月24日到4月30日这7天有登陆行为的那些用户，5月1号到5月7号这个窗口为观察期，观察4月24日到4月30日这个窗口有登陆行为的用户在观察期即5月1号到7号这个窗口内是否有登陆，如果有的话label是0，没有的话label为1。然后把上面关心的那些用户从4月30日往过去推7天的统计行为记为特征A，也就是说关注的是离当前更近的行为，注意这个特征可能对于不同的用户的行为稀疏程度差别很大；预测的时候，比如是5月8号预测，那么关注 的是5月1号到5月7号这个窗口有登陆行为的用户，看他们在5月8号到5月14号这个窗口内是否会登陆，那用户的特征A统计的是5月7号往过去推7天的行为）

1. **在得到了针对用户的流失预测概率以后，如何利用这个概率**？

**如果想简单的判定该用户是否是流失用户，那么需要一个阈值**，这个阈值一般会基于cost成本/代价在验证集上通过对混淆矩阵来遍历阈值找到一个最优的。

具体可以参考AWS sagemaker的一个demo code：<https://github.com/awslabs/amazon-sagemaker-examples/blob/master/sagemaker_neo_compilation_jobs/xgboost_customer_churn/xgboost_customer_churn_neo.ipynb>

利用上面的方法来计算阈值的时候，需要针对不同的metric/指标做一个代价/成本的预估，**但是有些时候对于某些场景/业务不容易预估这个成本，这个时候可以考虑基于策略的方式来做**。

比如将使用App的频率和时长作为用户轻重度的划分标准，结合用户流失预测概率（参考下图，**其实这里仍然需要象限划分的，如何来划分留存和流失的界限文中并没有提及，我理解可能是根据大盘历史上对留存和流失用户数的统计然后基于分位数来做的**：比如大盘一年统计的流失用户的比例为20%，那么对离线预测的这批用户的流失预测概率进行从小到大的排序（不要去重），然后选取80%分位数的概率作为阈值，超过这个阈值的可以认为是流失的潜在用户），将用户划分为高价值、重点发展、重点转化、有待挽留等几种类型，分析每个类型用户不同的行为特点和使用痛点，采取针对性的运营策略。

