变化，无法像样本处理流程那样相对固定下来；三是因为需要在离线训练[1]和在线预测两种场景下生成同一套特征，处理不当会引入不同程度的特征不一致问题，对效果产生严重影响。

基于以上这些考虑，比较推荐的做法是引入特征生成器的概念。所谓特征生成器，顾名思义，其作用是为样本生成要用到的特征。特征生成器包括生成和使用两部分，我们首先来介绍使用部分。特征生成器的本质可理解为一组配置规则，指定了要使用哪些原始特征，以及要对这些原始特征做什么样的特征工程处理，例如指定要使用商品价格特征，具体使用方式是对其进行等频离散化。特征生成器就像是医生开出的处方，而原始特征就像是药房，我们根据处方到药房去抓药，然后按照处方上的使用方式使用药品。具体来说，特征生成器由一组配置数据和一组解析程序组成，配置数据像处方一样描述了要使用哪些特征以及具体的使用方式，解析程序就像药剂师一样，根据配置数据生成一套具体的特征生成逻辑[2]，这套代码对于每条样本都会生成对应的特征。典型地，它会根据样本中用户的key和物品的key到原始特征库中去获取所需要的原始特征，然后将这些特征按照配置要求进行处理，得到最终可用的特征。

所以说特征生成器的生成主要包括两部分，即配置数据的生成和解析程序的生成。配置数据主要包括两类：一类是通过规则指定的，例如使用物品的价格特征；另一类由于维度过高且需要逻辑计算而无法手工指定，所以需要通过代码来计算拟合，典型的如连续特征离散化涉及的分段点。这一类配置数据的生成过程，可以用 Spark MLlib 中的估算器（estimator）概念来类比理解。一个估算器通过在指定数据上调用 fit 方法，生成一个转换器（transformer），这个转换器就是对特征的具体处理方法。例如，我们可以定义一个连续特征离散化的估算器，这个估算器指定了要离散化的分段数量，但没有指定具体的分段区间，那么具体的分段区间就需要在一份样本数据上调用 fit 方法来生成，调用 fit 后生成的转换器就可以接收一个原始连续特征作为输入，并生成对应的离散特征作为输出。TensorFlow 中的特征列（feature column）也是类似的概念。读者可以认真学习 Spark MLlib、TensorFlow 来理解和体会这种做法，其技术细节就不在这里展开介绍了。

在理解了配置数据的生成方法之后，对解析程序的生成也就自然明了了。还是用Spark MLlib中的概念来讲，所谓的解析程序本质上其实就是一组转换器，或者称之为“算子”。在具体的开发中，可以直接使用Spark MLlib中的算子，如果所提供的算子不够用，还可以通过实现

1　引入在线训练后该问题会得到缓解，但是在大部分场景下，都需要从离线训练做起，包括在线训练所依赖的初始离线版本。

2　严格来讲，解析程序不会对每组配置生成不同的代码，而是在同一套代码应用上使用不同的配置。这样做是为了突出每组配置对应着不同的执行逻辑。