组件总结

## 组件化

组件化主要是指解耦复杂系统时将多个功能模块拆分、重组的过程，有多种属性、状态会反映出其中的内部特点。简单的来说组件化是将一个复杂的程序分解为多个组件，组件与组件之间具有低耦合性、组件内部具有高内聚性、组件边界划分明确、通过维护各个组件来方便系统的维护和后续的升级。

在机器学习Pipeline中如果要实现组件化，其组件应该遵循如下规则：

1. 以标准化为前提：所有的方法在进行组件化的时候应该遵循组件划分的标准；
2. 以向Pipeline提供服务为基础：组件是从整个机器学习流程中抽象出来的功能单一且性能较为稳定的部分封装成的公共方法，并将所有的公共方法统一管理，以便于向外提供稳定可靠的计算服务。
3. 以组合式应用为目的：机器学习Pipeline编排的核心就是实现动态拼接已有的组件形成可执行的机器学习全流程Pipeline，并将Pipeline送入调度器中进行调度运行。因此，在组件化时应该准确的划分组件的边界，使其在拼接为Pipeline时有更高的效率。

## 组件

组件是对计算功能的封装，是具有一定计算作用的，能够独立开发和运行，能够动态的组装成为一个复杂的程序，组件可以是软件流程中一个或者是多个小功能的独立封装。

在机器学习中，组件可以理解为实现特定功能的方法。在机器学习领域下，组件的划分一般是按照特定的算法，例如可以将监督学习和非监督学习分类的具体算法封装成不同的组件、将不同的回归算法、聚类算法等封装成特定的组件。

在机器学习组件细粒度的划分上，可以根据划分组件的功能自定义程度将组件划分为三个层次：

* 应用层：应用层可以提供高度封装好的功能接口，接口内部是已经训练好的机器学习模型，使用者只需要输入预测数据进行预测即可。
* 中间层：在中间层提供未训练的机器学习模型，用户需要向模型中输入自己的训练数据对模型进行训练，
* 框架层：框架层可以提供各种已经实现的功能，如深度学习领域下的梯度下降算法、softmax、卷积运算。亦或是机器学习领域下的回归、聚类算法将这些算法进行组件级别的封装。

组件所在的层级越高，接口越简单，使用越方便，但灵活度越低，只适合非常通用的场景。层级越低，其灵活性越高，但编写和训练模型就越复杂。所以处于中间层级的模型组件化能够在复杂性和灵活性之间取得很好的平衡。组件化层次从高到低，其自定义程度从应用层的自主模型选择，到中间层的自主模型选择和自主模型训练，其自定义程度逐步增加。在框架层，可以自主的进行相应算法的选择完成自主模型的搭建和修改，可以实现模型的高度自定义。

同时，组件还需要具备以下几个要素：

1. 组件具备可移植性，在使用组件时，不需要考虑复杂的前提，只要接口符合调用规范，就能够在Pipeline中进行引用，既方便插入也方便使用。
2. 组件以接口为核心，在组件的使用层面上来说只需要关注组件的输入和输出规范，不需要关注组件的底层细节和具体实现。
3. 组件具有较高的可重用性。在不同的程序中，只要能够保证接口的符合规范，就能够在不同的应用程序中调用该组件。

## 组件化的必要性

机器学习的结果严重依赖于数据质量与模型逻辑，所以为了令分析人员能够专注于流程本身，不在分析程序编译、运行、并行化等方面花费精力。在一个机器学习程序编写的过程中，大部分的功能代码可以在其他机器学习代码中重用到。因此，如果将这些可通用的功能代码单独以计算方法的形式进行封装，便得到了类似于sklearn的机器学习方法库。如果我们将这些机器学习方法库进行进一步的封装，使得使用者可以通过拖拽计算组件的方式动态的编排机器学习程序流程，便可以降低使用者的学习和开发成本、提高应用创建效率，使得其在机器学习程序的开发效率上大大提升，机器学习程序开发的门槛大大降低，让使用者能够快速的、高效的构建机器学习程序。

## 组件的划分

在实践中，机器学习流程包括一系列的阶段，包括数据预处理、特征处理、模型拟合以及结果验证或预测。例如，将一组文本文档进行分类包括分词、清理、特征提取、训练分类模型以及输出分类结果。这些阶段都可以看作是黑盒过程，并且可以包装成组件。当然，不同的实现方式便是一个不同的组件，实现同一个功能不同实现方式的组件便可以归为一类。

## 组件的来源

1. 开发的角度：从组件开发人员的角度来说，组件主要是来源于开发人员的手动编写、封装第三方开源库的部分方法、封装第三方开源工程的部分功能方法。开发人员可以将这些方法封装成为组件。
2. 实验应用角度：从实验应用的角度来说，pipeline的编排人员所看到的组件来源于统一的组件仓库，在组件仓库中组件是以组件描述文件（yaml）统一按照版本存放的。在编排Pipeline时直接去组件仓库中取出。
3. 组件仓库角度：从组件仓库角度来看，组件主要是分不同的类别在统一上传接口进行上传的。组件仓库中的组件在组件保存统一接口进行上传以后，再根据不同的功能类别进行分门别类的保存。
4. 迭代角度：同一个组件在进行更新时，从组件统一上传接口进行上传，上传以后根据版本号进行更新。

## 自动组件化的方法