numpy.array (object, dtype = None, copy = True, order = None, subok = False, ndmin = 0)

**参数说明：**

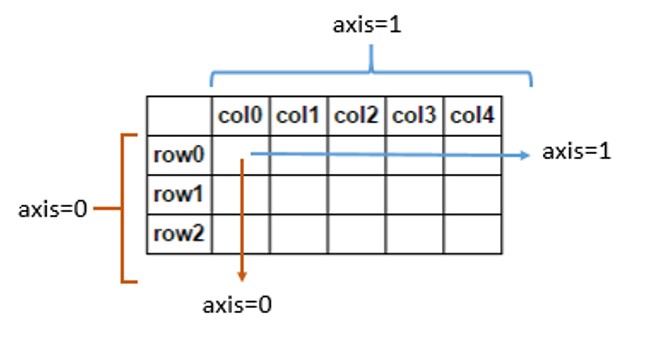
| **名称** | **描述** |
| --- | --- |
| object | 数组或嵌套的数列 |
| dtype | 数组元素的数据类型，可选 |
| copy | 对象是否需要复制，可选 |
| order | 创建数组的样式，C为行方向，F为列方向，A为任意方向（默认） |
| subok | 默认返回一个与基类类型一致的数组 |
| ndmin | 指定生成数组的最小维度 |

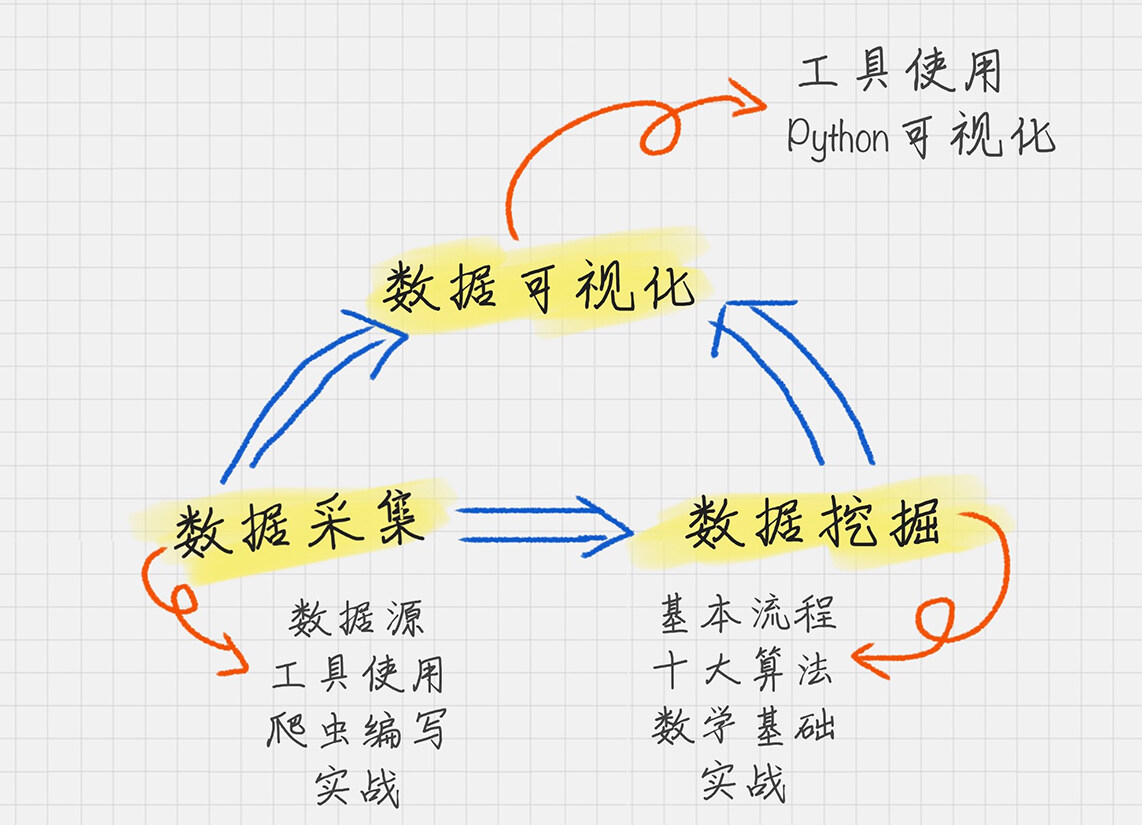
# 理解Python中的axis=0/1指的是什么：

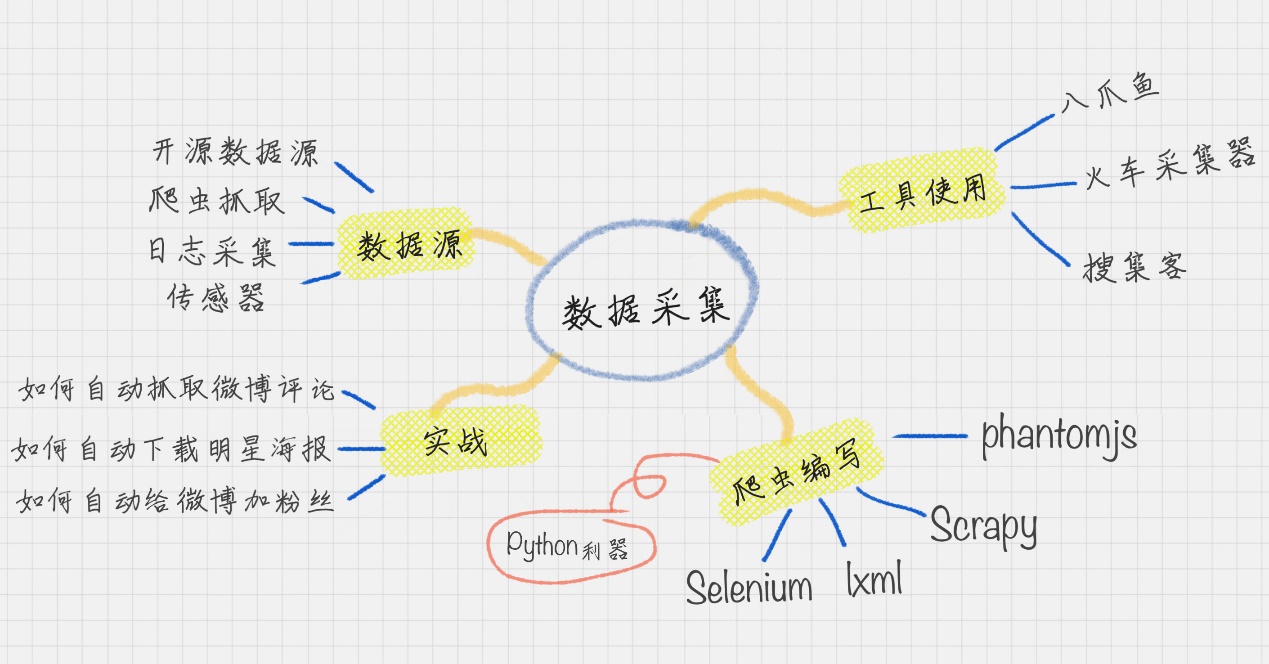
axis=0代表往**跨行（down)**，而axis=1代表**跨列（across)**

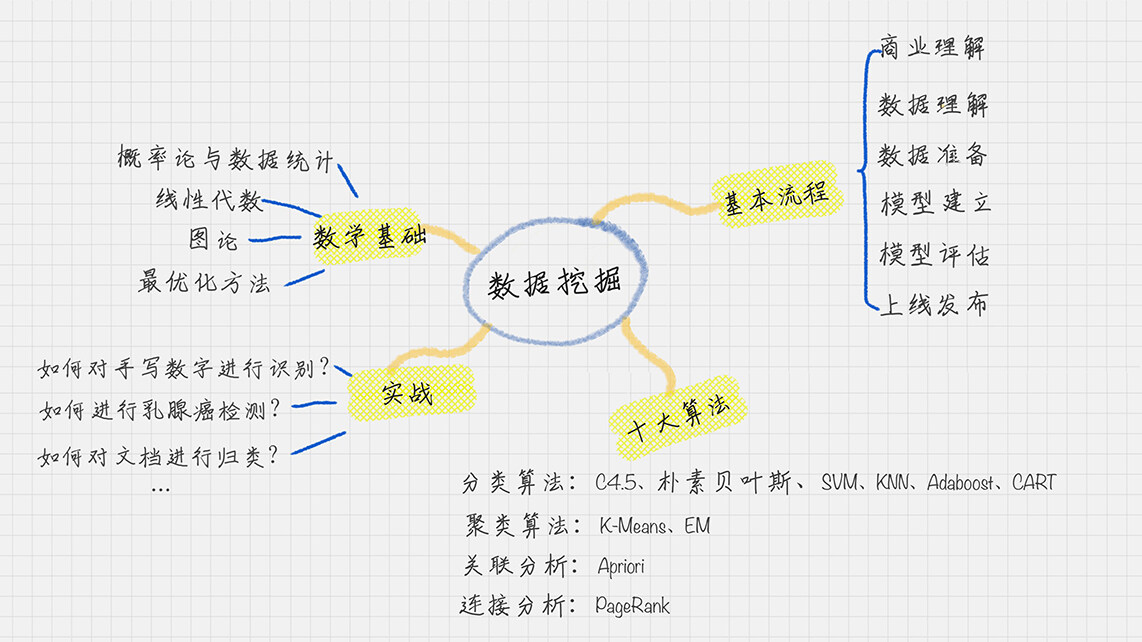
换句话说:

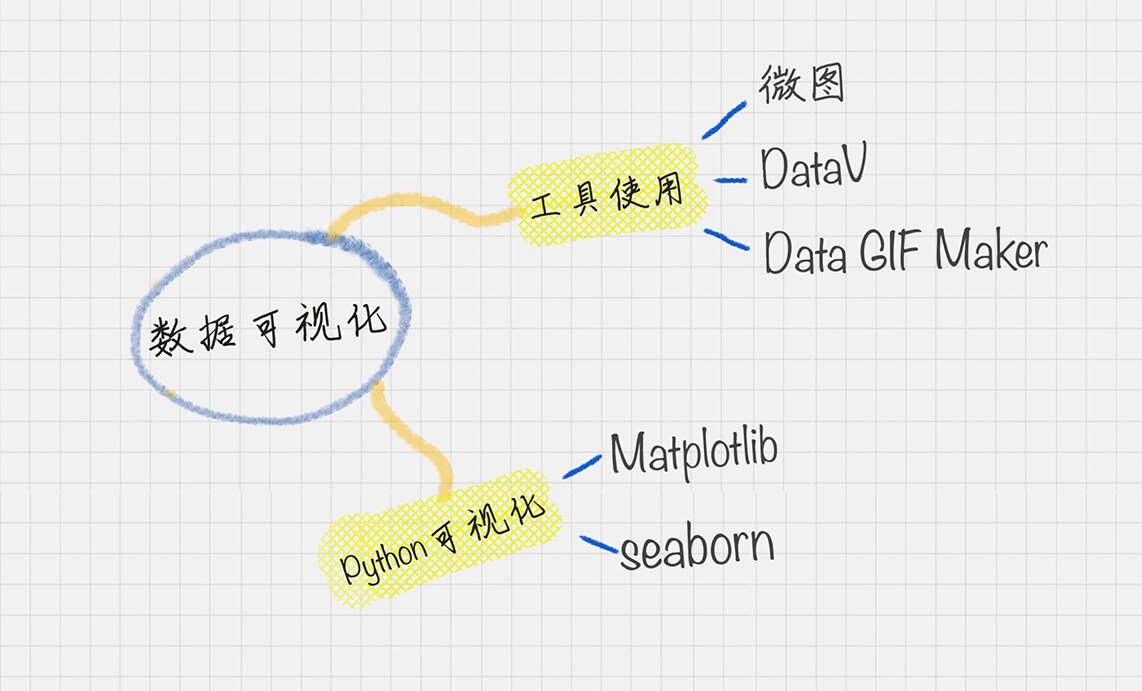
* 使用0值表示沿着每一列或行标签\索引值向下执行方法
* 使用1值表示沿着每一行或者列标签模向执行对应的方法

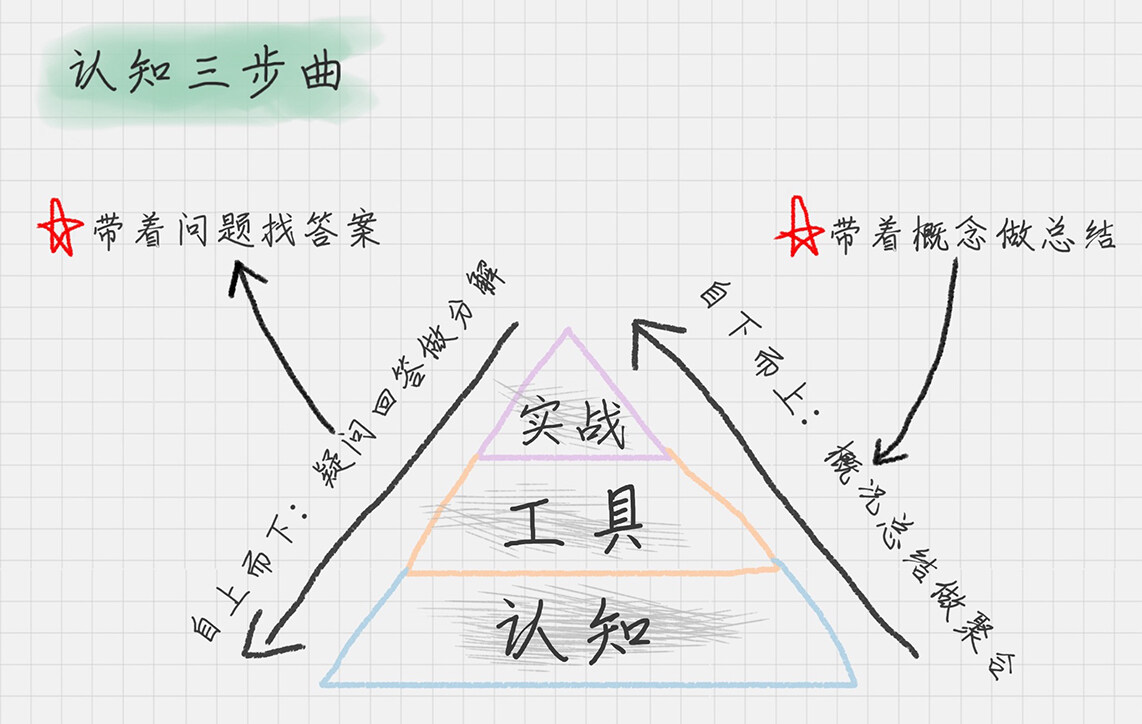


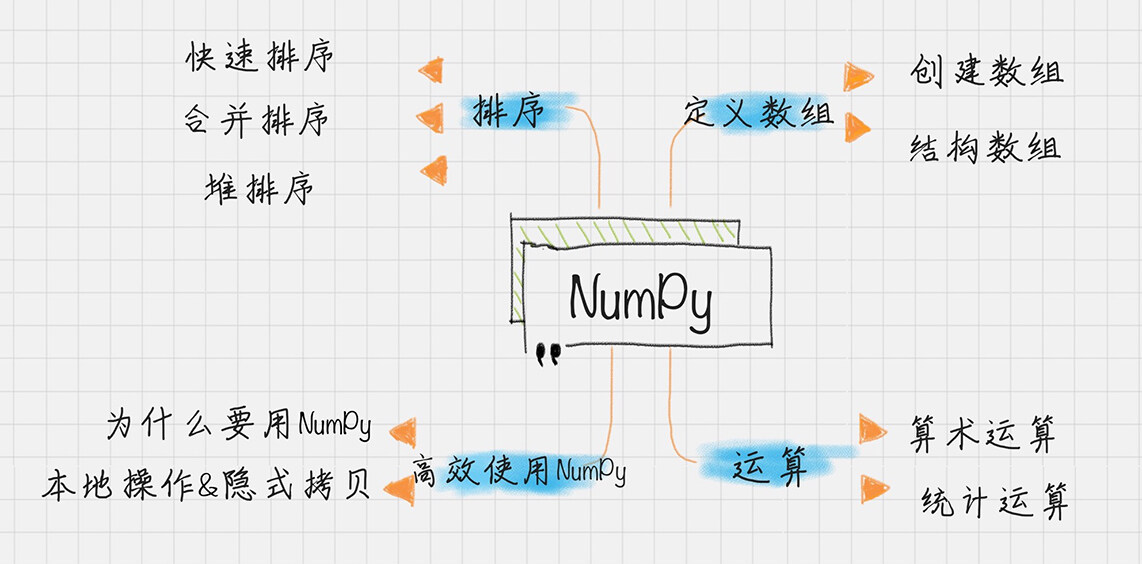


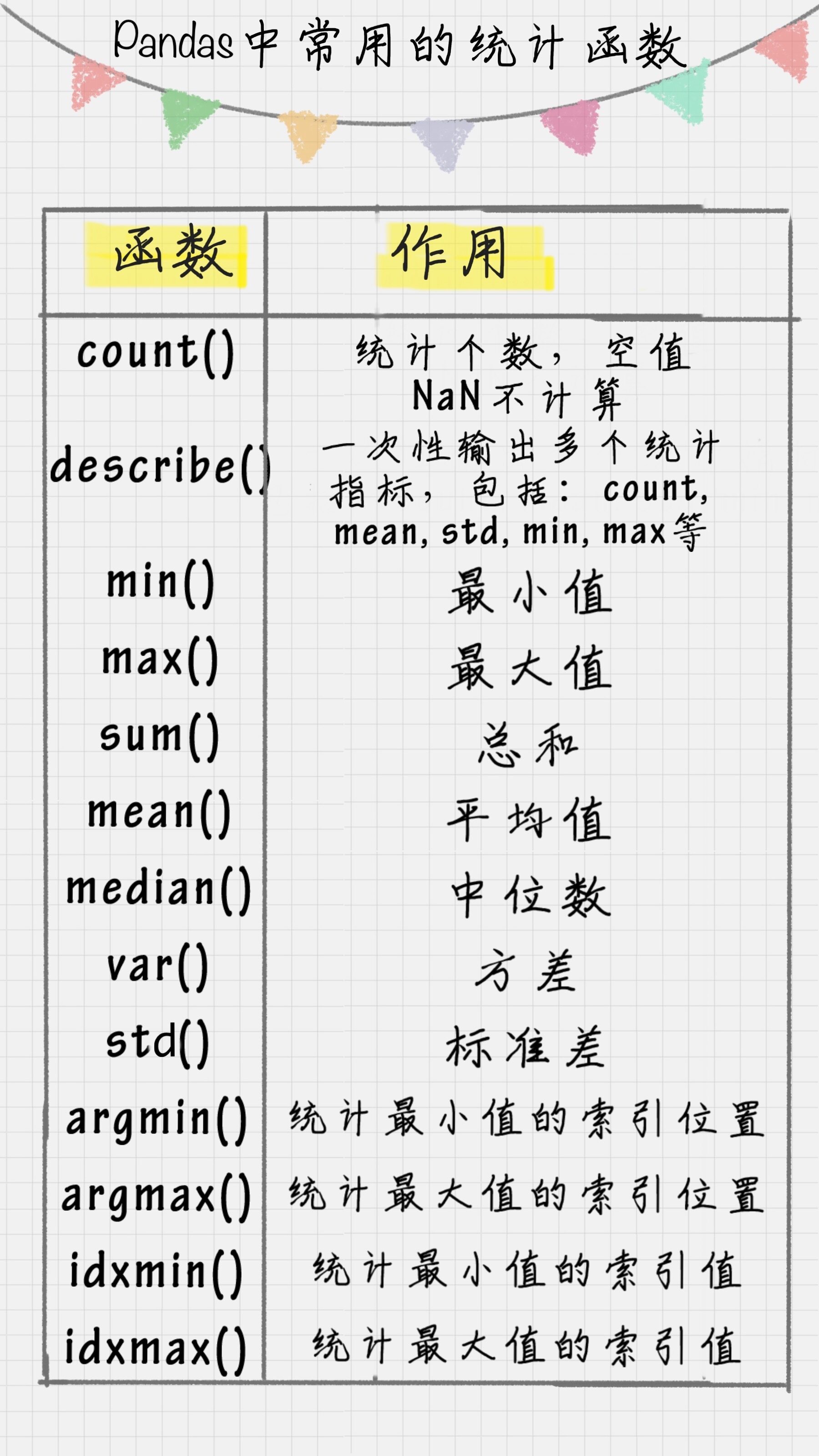




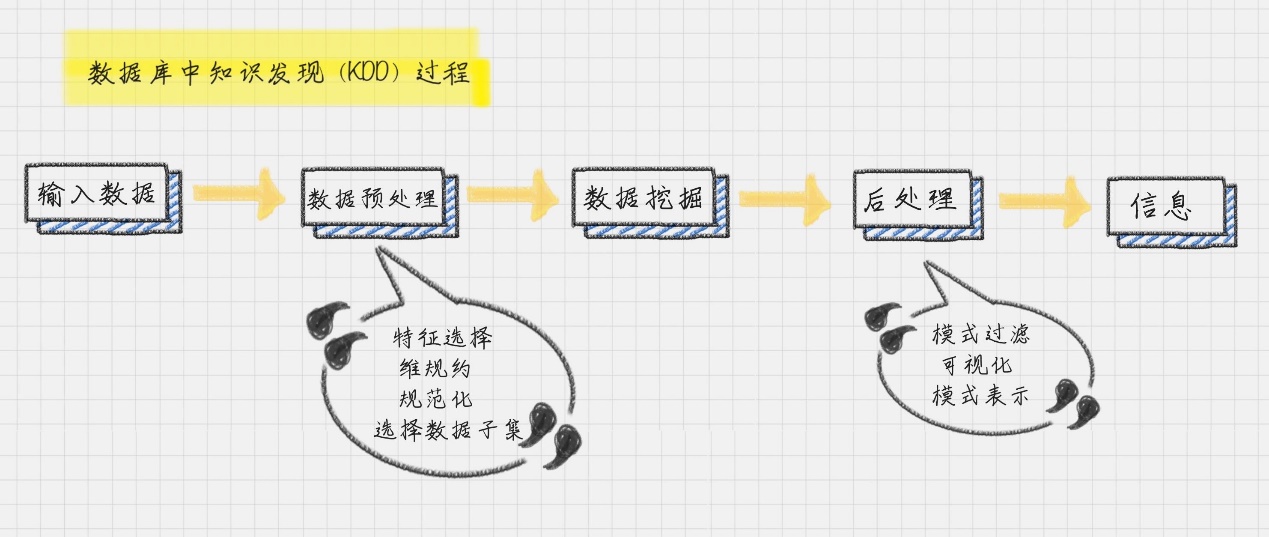






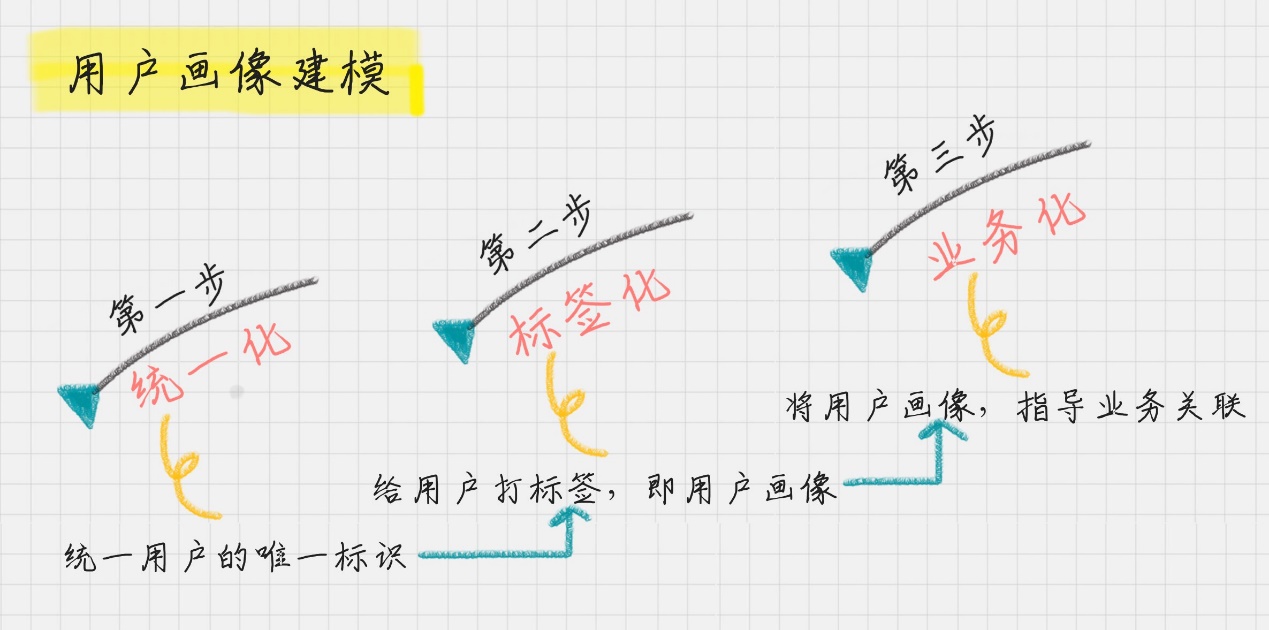


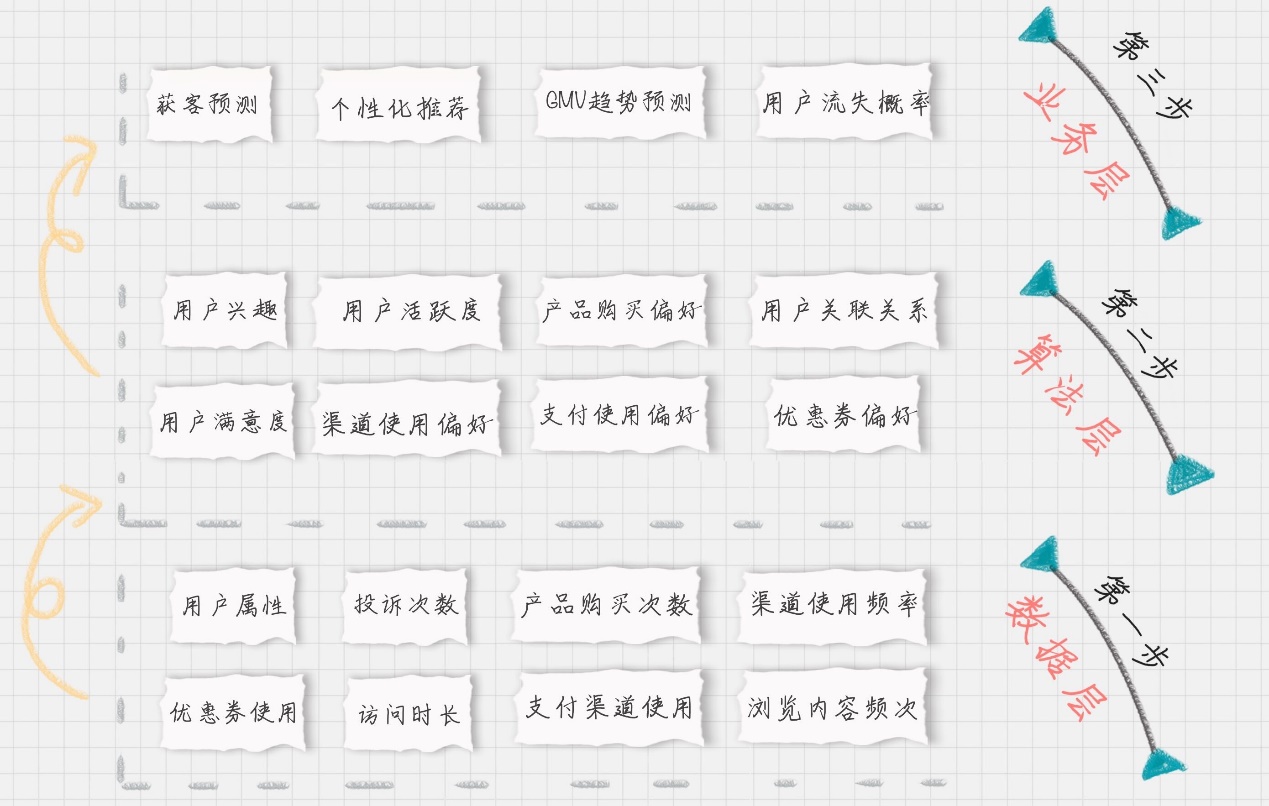




**用户画像的准则**

首先就是将自己企业的用户画像做个白描，告诉他这些用户“都是谁”“从哪来”“要去哪”。你可以这么和老板说：“老板啊，用户画像建模是个系统的工程，我们要解决三个问题。第一呢，就是用户从哪里来，这里我们需要统一标识用户 ID，方便我们对用户后续行为进行跟踪。我们要了解这些羊肉串的用户从哪里来，他们是为了聚餐，还是自己吃宵夜，这些场景我们都要做统计分析。第二呢，这些用户是谁？我们需要对这些用户进行标签化，方便我们对用户行为进行理解。第三呢，就是用户要到哪里去？我们要将这些用户画像与我们的业务相关联，提升我们的转化率，或者降低我们的流失率。”





首先，为什么要设计唯一标识？

用户唯一标识是整个用户画像的核心。我们以一个 App 为例，它把“从用户开始使用 APP 到下单到售后整个所有的用户行为”进行串联，这样就可以更好地去跟踪和分析一个用户的特征。设计唯一标识可以从这些项中选择：用户名、注册手机号、联系人手机号、邮箱、设备号、CookieID 等。其次，给用户打标签。你可能会想，标签有很多，且不同的产品，标签的选择范围也不同，这么多的标签，怎样划分才能既方便记忆，又能保证用户画像的全面性呢？

这里我总结了八个字，叫“用户消费行为分析”。我们可以从这 4 个维度来进行标签划分。

* 用户标签：它包括了性别、年龄、地域、收入、学历、职业等。这些包括了用户的基础属性。
* 消费标签：消费习惯、购买意向、是否对促销敏感。这些统计分析用户的消费习惯。
* 行为标签：时间段、频次、时长、访问路径。这些是通过分析用户行为，来得到他们使用 App 的习惯。
* 内容分析：对用户平时浏览的内容，尤其是停留时间长、浏览次数多的内容进行分析，分析出用户对哪些内容感兴趣，比如，金融、娱乐、教育、体育、时尚、科技等。

可以说，用户画像是现实世界中的用户的数学建模，我们正是将海量数据进行标签化，来得到精准的用户画像，从而为企业更精准地解决问题。最后，当你有了用户画像，可以为企业带来什么业务价值呢？

我们可以从用户生命周期的三个阶段来划分业务价值，包括：获客、粘客和留客。获客：如何进行拉新，通过更精准的营销获取客户。粘客：个性化推荐，搜索排序，场景运营等。留客：流失率预测，分析关键节点降低流失率。如果按照数据流处理的阶段来划分用户画像建模的过程，可以分为数据层、算法层和业务层。你会发现在不同的层，都需要打上不同的标签。数据层指的是用户消费行为里的标签。我们可以打上“事实标签”，作为数据客观的记录。算法层指的是透过这些行为算出的用户建模。我们可以打上“模型标签”，作为用户画像的分类标识。业务层指的是获客、粘客、留客的手段。我们可以打上“预测标签”，作为业务关联的结果。所以这个标签化的流程，就是通过数据层的“事实标签”，在算法层进行计算，打上“模型标签”的分类结果，最后指导业务层，得出“预测标签”。



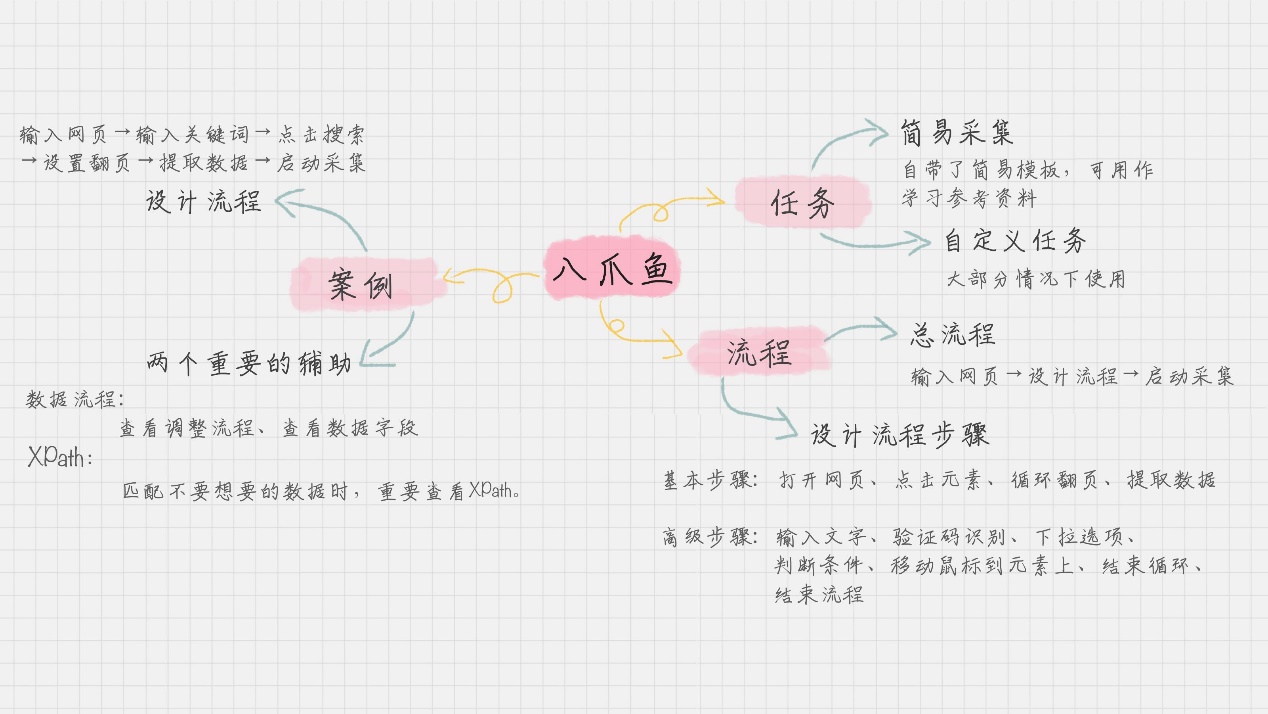


下面我们来看日志采集。为什么要做日志采集呢？日志采集最大的作用，就是通过分析用户访问情况，提升系统的性能，从而提高系统承载量。及时发现系统承载瓶颈，也可以方便技术人员基于用户实际的访问情况进行优化。日志采集也是运维人员的重要工作之一，那么日志都包括哪些呢，又该如何对日志进行采集呢？日志就是日记的意思，它记录了用户访问网站的全过程：哪些人在什么时间，通过什么渠道（比如搜索引擎、网址输入）来过，都执行了哪些操作；系统是否产生了错误；甚至包括用户的 IP、HTTP 请求的时间，用户代理等。这些日志数据可以被写在一个日志文件中，也可以分成不同的日志文件，比如访问日志、错误日志等。日志采集可以分两种形式。通过 Web 服务器采集，例如 httpd、Nginx、Tomcat 都自带日志记录功能。同时很多互联网企业都有自己的海量数据采集工具，多用于系统日志采集，如 Hadoop 的 Chukwa、Cloudera 的 Flume、Facebook 的 Scribe 等，这些工具均采用分布式架构，能够满足每秒数百 MB 的日志数据采集和传输需求。自定义采集用户行为，例如用 JavaScript 代码监听用户的行为、AJAX 异步请求后台记录日志等。埋点是什么埋点是日志采集的关键步骤，那什么是埋点呢？埋点就是在有需要的位置采集相应的信息，进行上报。比如某页面的访问情况，包括用户信息、设备信息；或者用户在页面上的操作行为，包括时间长短等。这就是埋点，每一个埋点就像一台摄像头，采集用户行为数据，将数据进行多维度的交叉分析，可真实还原出用户使用场景，和用户使用需求。那我们要如何进行埋点呢？埋点就是在你需要统计数据的地方植入统计代码，当然植入代码可以自己写，也可以使用第三方统计工具。我之前讲到“不重复造轮子”的原则，一般来说需要自己写的代码，一般是主营核心业务，对于埋点这类监测性的工具，市场上已经比较成熟，这里推荐你使用第三方的工具，比如友盟、Google Analysis、Talkingdata 等。他们都是采用前端埋点的方式，然后在第三方工具里就可以看到用户的行为数据。但如果我们想要看到更深层的用户操作行为，就需要进行自定义埋点。总结一下，日志采集有助于我们了解用户的操作数据，适用于运维监控、安全审计、业务数据分析等场景。一般 Web 服务器会自带日志功能，也可以使用 Flume 从不同的服务器集群中采集、汇总和传输大容量的日志数据。当然我们也可以使用第三方的统计工具或自定义埋点得到自己想要的统计内容。

预测比特币的未来走势，可以从以下维度抓取数据  
1.认知度：社会对比特币的认可，抓取百度指数、谷歌搜索量、微博数据等  
2.比特币依赖的技术：作为数字货币，核心技术的完善和认可度占比重较大，可以从区块链相关技术网站爬取数据，也可以从微博爬取（微博也是技术人活跃的交流平台）  
3.供给平衡：比特币虽说是一种数字货币，但仍逃脱不掉是一种商品的本质，商品必然受市场平衡调节影响，所以爬取买入量、抛售量还有历史价格也是一种预测维度  
4.政府政策：政府政策的影响占很大比重，若国家出台政策强制打压或者支持，那么对价格的影响起了根本性作用，所以需要爬取相关的新闻  
5.竞争数字货币：作为一种商品，必然要考虑竞争品的相关情况，需要抓取其它数字货币相关信息如其它货币的价格、交易量。资本在流入其它市场的时候，与之对应的竞品必然会受影响。抓取数据途径：其它货币交易平台

以上信息，再按能否用二位表结构来进行逻辑表达，分类为结构化数据和非结构化数据。

结构化数据：比特币产量（计算得出）、挖币成本（计算得出）、日/周/月交易数（爬取相关交易网站得到，用以预测热度）、交易手续费走势（爬取相关交易网站得到，用以了解交易成本）。  
非结构化数据：比特币相关新闻（都可爬取比特币垂直资讯网站获得）、比特币相关政策（努力固然重要，还要考虑时代的进程）、比特币自身协议变更情况（可爬取比特币垂直资讯网站获得）。



# 数据清洗

在这里，我将数据清洗规则总结为以下 4 个关键点，统一起来叫“**完全合一**”，下面我来解释下。

**完整性：**单条数据是否存在空值，统计的字段是否完善。

**全面性：**观察某一列的全部数值，比如在 Excel 表中，我们选中一列，可以看到该列的平均值、最大值、最小值。我们可以通过常识来判断该列是否有问题，比如：数据定义、单位标识、数值本身。

**合法性：**数据的类型、内容、大小的合法性。比如数据中存在非 ASCII 字符，性别存在了未知，年龄超过了 150 岁等。

**唯一性：**数据是否存在重复记录，因为数据通常来自不同渠道的汇总，重复的情况是常见的。行数据、列数据都需要是唯一的，比如一个人不能重复记录多次，且一个人的体重也不能在列指标中重复记录多次。在很多数据挖掘的教学中，数据准则通常会列出来 7~8 项，在这里我们归类成了“完全合一”4 项准则，按照以上的原则，我们能解决数据清理中遇到的大部分问题，使得数据**标准、干净、连续，**为后续数据统计、数据挖掘做好准备。如果想要进一步优化数据质量，还需要在实际案例中灵活使用。

## 1 完整性

**问题 1：缺失值**

在数据中有些年龄、体重数值是缺失的，这往往是因为数据量较大，在过程中，有些数值没有采集到。通常我们可以采用以下三种方法：

**删除：**删除数据缺失的记录；

1. df['Age'].dropna(axis=0, how='any'/'all', thresh=None, subset=None, inplace=False)

**均值：**使用当前列的均值；

1. df['Age'].fillna(df['Age'].mean(), inplace=True)

**高频：**使用当前列出现频率最高的数据。

可以先通过 value\_counts 获取 Age 字段最高频次 age\_maxf，然后再对 Age 字段中缺失的数据用 age\_maxf 进行填充：

1. age\_maxf = train\_features['Age'].value\_counts().index[0]
2. train\_features['Age'].fillna(age\_maxf, inplace=True)

**问题 2：空行**

我们发现数据中有一个空行，除了 index 之外，全部的值都是 NaN。Pandas 的 read\_csv() 并没有可选参数来忽略空行，这样，我们就需要在数据被读入之后再使用 dropna() 进行处理，删除空行。

1. # 删除全空的行
2. df.dropna(how='all',inplace=True)

## 2 全面性

**问题：列数据的单位不统一**

观察 weight 列的数值，我们能发现 weight 列的单位不统一。有的单位是千克（kgs），有的单位是磅（lbs）。这里我使用千克作为统一的度量单位，将磅（lbs）转化为千克（kgs）：

1. # 获取 weight 数据列中单位为 lbs 的数据
2. rows\_with\_lbs = df['weight'].str.contains('lbs').fillna(False)
3. **print** df[rows\_with\_lbs]
4. # 将 lbs转换为 kgs, 2.2lbs=1kgs
5. **for** i,lbs\_row **in** df[rows\_with\_lbs].iterrows():
6. # 截取从头开始到倒数第三个字符之前，即去掉lbs。
7. weight = int(float(lbs\_row['weight'][:-3])/2.2)
8. df.at[i,'weight'] = '{}kgs'.format(weight)

## 3 合理性

**问题：非 ASCII 字符**

我们可以看到在数据集中 Firstname 和 Lastname 有一些非 ASCII 的字符。我们可以采用删除或者替换的方式来解决非 ASCII 问题，这里我们使用删除方法：

1. # 删除非 ASCII 字符
2. df['first\_name'].replace({r'[^\x00-\x7F]+':''}, regex=True, inplace=True)
3. df['last\_name'].replace({r'[^\x00-\x7F]+':''}, regex=True, inplace=True)

## 4 唯一性

**问题 1：一列有多个参数**

在数据中不难发现，姓名列（Name）包含了两个参数 Firstname 和 Lastname。为了达到数据整洁目的，我们将 Name 列拆分成 Firstname 和 Lastname 两个字段。我们使用 Python 的 split 方法，str.split(expand=True)，将列表拆成新的列，再将原来的 Name 列删除。

1. # 切分名字，删除源数据列
2. df[['first\_name','last\_name']] = df['name'].str.split(expand=True)
3. df.drop('name', axis=1, inplace=True)
4. # str.split(str="", num=string.count(str)).

**问题 2：重复数据**

我们校验一下数据中是否存在重复记录。如果存在重复记录，就使用 Pandas 提供的 drop\_duplicates() 来删除重复数据。

1. # 删除重复数据行
2. df.drop\_duplicates(['first\_name','last\_name'],inplace=True)

