数据转换

学习计划：

- 了解数据转换的基本概念
- 掌握数据清洗的基本流程
- 掌握利用 Python 包 Pandas 清洗数据的方法
- 了解用于数据转换的几种工具
- 掌握使用 Kettle 工具进行数据清洗的方法
- 理解本章案例

本章主要介绍数据转换，数据转换包括数据清洗、格式转换等。其中对数据清洗做了很大篇幅的说明，包括数据清洗的内容、研究现状、流程等，并利用 Python 工具对数据清洗进行案例分析。介绍了数据转换工具，包括 Data Stage、Kettle、Informatica Powercenter、ETL Automation 和 SSIS；并对其中的 Kettle 工具进行了案例说明。

3.1 数据转换

3.1.1 数据转换的概念

数据转换(Transform)，顾名思义是指将数据从一种表示形式转变为另一种表现形式的过程。在 ETL 中，数据转换之前应该进行数据清洗(Cleaning)。

数据转换的任务主要是进行不一致的数据转换、数据粒度转换，以及一些商务规则的计算。

(1) 不一致的数据转换。

这个过程是一个整合的过程，将不同业务系统的相同类型的数据统一，比如同一个供应商在结算系统的编码是 XX0001，而在 CRM 中编码是 YY0001，这样在抽取过来之后统一转换成一个编码。

(2) 数据粒度转换。

业务系统一般存储非常详细的数据，而数据仓库中的数据是用来分析的，不需要非常详细的数据。一般情况下，会将业务系统数据按照数据仓库粒度进行聚合。

(3) 商务规则的计算。

不同的企业有不同的业务规则和数据指标，这些指标有时不是靠简单的加加减减就

能完成的，需要在 ETL 中将这些数据指标计算好之后存储在数据仓库中，以供分析使用。

3.1.2 数据转换的标准

数据转换不是毫无章法的，而是有一定的标准可依，数据转换标准是依照不同计算机环境间数据转换的一种中间格式。包括一整套使数据按字段、记录和文件要求进行编码的规划，以便通过指定的介质进行转换。数据模型是研制编码规则的先决条件，转换标准的中介性质是主要的特征。

转换标准优化后可使所有的数据进行有效地通信，而对产品和数据库结构进行优化后，可进行有效的存储、应用及维护，是一个将数据从一种表示形式转换为另一种表现形式的过程。例如，软件的全面升级，肯定带来数据库的全面升级，每一个软件对其后面数据库的构架与数据的存诸形式都是不相同的，就需要进行数据转换了。主要是由于数据量不断增加，原来的数据构架不合理，不能满足各方面的要求，需要对数据库进行更换，对数据结构进行更换，从而需要对数据本身的转换。

数据类型是一个值的集合以及定义在这个值集上的一组操作。数据类型包括原始类型、多元组、记录单元、代数数据类型、抽象数据类型、参考类型以及函数类型，下面介绍数据转换要求的数据格式和数据类型。

数据格式(Data Format)是数据保存在文件或记录中的编排格式，可为数值、字符或二进制数等形式，由数据类型及数据长度来描述。

数据类型是与程序中出现的变量相联系的数据形式，常用的数据类型可分为两大类。

(1) 简单类型。

简单类型数据的结构非常简单，具有相同的数学特性和相同的计算机内部表示法，其数据的逻辑结构特点是只包含一个初等项的节点。通常有 5 种基本的简单类型：整数类型、实数类型、布尔类型、字符类型和指针类型。

(2) 复合类型。

复合类型也称组合类型或结构类型，是由简单类型以某种方式组合而成的。根据不同的构造方法，可构成以下不同的数据结构类型。

① 数组类型：所有成分都属于同一类型。

② 记录类型：各成分不一定属于同一类型。

③ 集合类型：它定义的值集合是其基类型的幂集，也就是基类型的值域的所有子集的集合。

④ 文件类型：属于同一类型的各成分的一个序列，这个序列规定各成分的自然次序。

数据长度是可度量的，通常用 8 位二进制数组成一字节，一个键盘上的字母、数字或其他符号用一个 ASCII 码表示，一字节可容纳一个 ASCII 码(含一位奇偶校验位)，计算机存储器的常用编址单位是以字节为单位的。计算机的通信传输单位一般也以字节为基础。

3.1.3 数据转换的方法

进行数据转换时一定要注意，不能不顾一切地进行转换，因为，有时转换会严重扭曲

数据本身的内涵，不能用过于复杂的转换方法。但是，在许多情况下，如果转换得当，就是一种好方法。数据转换方法主要有以下5种。

1. 对数转换

对数转换是将原始数据自然对数值作为分析数据，如果原始数据中有0，就可以在底数中加上一个小数值。

对数转换适用的情况主要包括部分正偏态资料、等比资料，以及各组数值和均值比值相差不大的资料。

2. 平方根转换

平方根转换适用的情况如下。

(1) 服从泊松分布的资料。

(2) 轻度偏态资料。

(3) 样本的方差和均数呈正相关的资料。

(4) 变量的所有个案为百分数，并且取值在0%～20%或者80%～100%的资料。

3. 平方根反正弦转换

平方根反正弦转换适用的情况包括变量所有个案为百分数，并且取值广泛的资料。

4. 平方转换

平方转换适用的情况主要包括方差和均数的平方呈反比；资料呈左偏。

5. 倒数转换

倒数转换和平方转换相反，需要方差和均数的平方呈正比，但是，倒数转换需要资料中没有接近或者小于零的数据。

在“计算变量”对话框对变量转换更加多样和灵活。但是，仍然要小心，不能扭曲变量。

3.1.4 数据之间的关联

在对数据进行转换的同时要注意数据之间的关联，查找存在于项目集合或对象集合之间的频繁模式、关联、相关性或因果结构，因此又称为对数据进行关联分析，下面主要介绍2个算法来分析数据之间关联性。

1. Apriori 算法

Apriori算法是挖掘产生布尔关联规则所需频繁项集的基本算法，也是最著名的关联规则挖掘算法之一。Apriori算法就是根据有关频繁项集特性的先验知识命名的。它使用一种称作逐层搜索的迭代方法，k-项集用于探索$(k+1)$-项集。首先，找出频繁1-项集的集合。记作L1，L1用于找出频繁2-项集的集合L2，再用于找出L3，如此下去，直到不

能找到频繁 k-项集。找每个 L_k 需要扫描一次数据库。

为提高按层次搜索并产生相应频繁项集的处理效率，Apriori 算法利用了一个重要性质，并应用 Apriori 性质来帮助有效缩小频繁项集的搜索空间。

2. FP-growth 算法

由于 Apriori 算法的固有缺陷，即使进行了优化，其效率也仍然不能令人满意。2000 年，韩佳伟(Han Jiawei)等人提出了基于频繁模式树(Frequent Pattern Tree，FP-tree)的发现频繁模式的算法 FP-growth。在 FP-growth 算法中，通过两次扫描事务数据库，把每个事务包含的频繁项目按其支持度降序压缩存储到 FP-tree 中。在以后发现频繁模式的过程中，不需要再扫描事务数据库，而仅在 FP-Tree 中查找，并通过递归调用 FP-growth 算法来直接产生频繁模式，因此在整个发现过程中也不需产生候选模式。该算法克服了 Apriori 算法中存在的问题，在执行效率上也明显好于 Apriori 算法。

3.2 数据清洗

数据清洗一直被视为数据挖掘的一部分，但随着技术的不断发展和研究的深入，数据清洗被单独提出来，作为独立的一部分得到发展及应用。数据清洗是数据仓库技术中的重要环节，其清洗的好坏直接影响进入数据仓库中的数据质量，进而间接影响决策支持。数据仓库中的数据不可避免地存在许多异常，这就使得数据在进入数据仓库之前必须进行清洗。

清洗数据仓库中的数据，可以纠正错误的数据、修复残缺的数据和去除多余的数据，进而将清洗后的数据进行整理，挑选出所需的数据进行集成，将多格式的数据转换为单一的数据格式，消除多余的数据属性，从而达到数据类型同一化、存储集中化、格式一致化和信息精练化。经过处理后的数据，可以尽量减少挖掘系统所付出的代价和提高挖掘知识的有效性，因此数据清洗被认为是建立数据仓库需要解决的最大问题之一。

3.2.1 数据清洗的主要内容

数据清洗一直被人们所熟知，我们需要明白何为数据清洗。按其字义分析，数据清洗就是把“垃圾”洗掉，是纠正数据文件的最后一道程序，包括处理无效值和缺失值、检查数据一致性等。在建立数据仓库时，不可避免有许多冲突，我们不需要的数据，称为“脏数据”，数据清洗的重点就是要按照一定的规则清洗掉“脏数据”。数据清洗的任务就是把那些不符合要求的数据过滤，将结果呈交给主管部门，用于决策和支持决策。

随着企业的不断发展，企业对客户关系的重视度也日渐提升，客户信息的质量也越来越受到企业的重视。名字和地址类数据的清洗是清洗技术在特定领域类的经典应用。用于纠错、标准化、提升数据质量的清洗工具主要有 Trillium Software System、Trillium Software、MatchMaker、Info Tech Ltd、ItraAddress Management、The Computing Group、NADIS、Groupl Software 和 MasterSoft International 等。

如今，数据清洗的相关研究主要集中在以下几个方面。

1. 残缺数据

这类数据主要是指一些应有的数据缺失，使得整个数据不完整，如学校教务系统学生成绩缺失、企业业务系统的报表不相匹配等，对于这一类数据，可以采用直接删除记录和数据填充的方法进行清洗，将补全的数据写入数据仓库中。

2. 错误数据

这类数据主要是指业务系统不够健全，在存入数据仓库中导致冲突或者格式的不一致性所导致，如文字转换为一堆乱码、日期格式不正确和图文匹配错误等。对于这类数据，常用统计方法进行分析，识别该错误值。常用的统计方法有回归方程、正态分布、偏差分析等。

3. 重复数据

清洗数据集中的近似重复的记录问题是目前清洗领域研究较为普遍且广泛的内容，这类数据主要是由于人为因素所致。为了消除这类数据，首先需要解决检测重复记录的问题，在这类问题中基于字符串的匹配问题是检测基础。近似字符串匹配问题研究的主要方法有基本字符串匹配法、递归匹配法、基于动态规划的编辑距离法、N_Grams 距离法、快速过滤法等。针对数据量较大且比较集中的相似重复记录的研究策略中，目前主要的应用有基于近邻排序方法、多趟排序近邻方法以及优先队列方法。对于基本常见的问题，可以直接删除多余项，在维表中出现此类情况可以将重复记录的所有字段导出，让客户层确认并进行处理。对于同一实体的不同种表达，清洗时需要合并数据。

4. 基于大数据增量处理的清洗

并行、增量处理大数据的研究成果主要集中在 ETL 工具上，并没有什么实质性的成果。某些商业 ETL 工具已经可以对数据进行并行集成和清洗，并提供数据的增量复制功能，主要是利用多线程、多进程、流水、多处理器技术来进行。

5. 通用可扩展的清洗模型

商业 ETL 工具提供了一些数据清洗功能，但是都缺乏扩展性，也就为科研人员提供了很好的素材和方向。为此，一些研究人员提出了数据清洗系统的框架，并且围绕该框架，提出了数据清洗语言和模型，在通用 SQL 基础上扩展了新的清洗操作。

目前国内对于数据清洗技术的研究还不是很多，只能在一些学术期刊上见到一些理论文章，相关的书籍也很少，比较系统的书是《干净的数据》。针对数据清洗的中文论文也不是很多。

3.2.2 数据清洗研究现状

在人们眼中，数据清洗通常是数据仓库、数据库中的知识发现和数据/信息质量管理 3 个领域的数据准备阶段的步骤之一。在数据准备阶段，一般需要两种类型的工具：转

换工具和清洗工具。转换工具也就是我们常说的 ETL 工具，主要功能包括数据抽取、转换和载入，它主要是为 OLAP 提供服务。数据清洗工具可以满足一些特殊的要求，它的核心工作是净化和匹配，但不能提供数据抽取、载入和更新等功能。因此，在 ETL 过程中引入数据清洗技术是必然的趋势，两种工具取长补短，发挥各自的优势功能，只有提高数据质量，才能更好地为决策者提供可靠又可信的数据依据。

3.2.3 数据清洗的必要性

“脏数据”会对数据仓库系统造成不良的影响，从而影响数据仓库的运行效果，进一步影响数据挖掘效能，最终影响决策管理。所以数据预处理工作就显得尤为重要，数据清洗作为数据预处理的重要环节，在数据仓库构建过程中占据相当重要的位置，如图 3-1 所示。

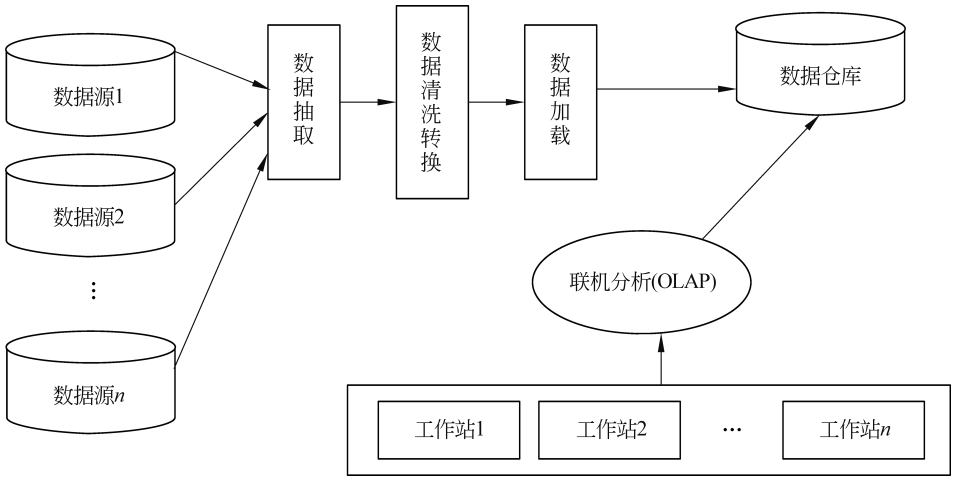

图 3-1 数据清洗在数据仓库中的位置

3.2.4 数据清洗的问题

数据清洗的目的在于检测数据中存在的错误和不一致，消除或者改正它们，进而提高数据的质量。在以前，人们根本没有意识到数据清洗的重要性，而是把数据直接装入数据库中，因而引发了一系列问题。如今，人们逐渐意识到数据清洗对提高数据质量具有正相关性。下面通过案例展示数据清洗的必要性。

工业运用：在企业工厂中，一种线运用“bus(现场总线)”，但这个词在不同的行业内可能有不同的表达方式。

市场交流：客户姓名和电话号码错误、地址无效和时间节点不当都可以导致商品无法投向该客户，进而影响公司的效益。

检测和消除重复记录是数据清洗和提高数据质量要解决的主要问题之一。所谓相似重复记录，是指客观上表达同一实体，但表达方式不同和由于拼写问题使得 DBMS 不能识别，将其视为重复记录。例如，由于拼写问题，导致同一学生在两次登录教务系统时使

用的记录不同，(张三，男，20岁，重庆，2017/09/10)和(张三，男，20岁，渝，2017/09/10)，虽说只有一字之差，表达的是同一学生，但这两条记录已被标为相似重复记录。这样的记录，其影响结果是巨大的，可能导致建立错误的数据挖掘模型。

3.2.5 数据清洗对工具的要求

为了完成数据清洗任务，满足需求，需要应用合理的清洗工具。数据清洗的复杂性，使得清洗涉及多种不同的算法和清洗规则，对工具有以下要求。

(1) 具备检测重复和异常记录的能力，为避免遗漏，提高精度和速率，可同时选择多种检测算法，进行交叉检测。

(2) 提高质量评估的方法，能够实时地对数据质量进行评估，评估其检测算法和清洗策略的效率和可满足程度，以便实时调整清洗流程。

(3) 具备元数据管理能力，能够充分使用元数据，对数据进行集成，提高系统的灵活度，进而保证数据质量。

由于数据清洗的高度复杂性，对不同的数据源，要求数据清洗适应不同的数据类型、数据量及具体业务。一种清洗算法无论多么有效，也不可能在所有问题上都表现出良好的清洗效果，也不可能依靠一种乃至几种清洗算法就能良好地解决所有的清洗问题。所以仍需要探究数据清洗算法和规则，探求一种可扩展和可交互的数据清洗系统框架，进而大大提高数据清洗的综合效果。

3.2.6 数据清洗的流程

数据清洗是一个反复的过程，整个过程枯燥无味，它要求清洗人员有足够的耐心，需要反复发现问题并进行处理。数据清洗占据数据分析工作的80%，这又体现出它的重要性，是数据分析中最为重要的一环。数据清洗过程分为以下几个步骤。

1. 整理原数据

在进行数据清洗时，其原数据可能有多个，且提供的文件格式可能不同，这些原数据可能是网页文件、PDF文件、简单文本文件等类型。为了满足对清洗格式的要求，必须对不同格式的原数据进行整理，清洗数据质量问题。

2. 数据源合并

原数据的格式不一，在合并数据源之前需要分析各个原数据的格式，需要将这些不同的标准和格式进行统一。有些格式需要运用可视交互工具完成，如关系数据库中的数据列；有些只需要简单的手工整理，如电子表格中的数据列。无论使用什么格式，运用不同的工具进行格式统一，必须做到数据标准统一后才能合并数据源。

3. 数据清洗

数据清洗的主要目的是发现并清除数据源中各种数据存在的质量问题，不同的数据质量问题，应用的数据清洗工具和方法各异，甚至有的需要交互完成。面对数据质量问

题,需要反复迭代,直到数据满足可视数据分析要求为止。

4. 可视数据分析

经过前面的工作,数据的各种质量问题基本清除,导入可视化分析工具后就可以分析可视数据。当可视分析工具报错时,需要返回第三步工作,再次审视数据质量问题,反复迭代,直到问题全部解决,满足应用研究的要求时,才可以展示可靠的可视分析结果。

5. 建立操作文档

对可视化处理后的数据,需要评估清洗后数据的质量,必须建立跟踪和记录数据清洗每一个行为的操作文档,且文档中的算法和操作步骤必须能够再现。

3.2.7 数据清洗的原理

数据清洗是一个减少错误和不一致性、解决对象识别的过程。针对不同的数据,采用不同的方法及其模型进行处理,为此需要了解数据清洗的原理。

数据清洗的原理就是首先分析脏数据产生的原因及其存在方式;其次分析整个数据流的过程,最后运用一些技术和结合数理统计的方法将脏数据转换为满足数据质量要求的数据。按数据清洗的方式可将其分为4类。

(1) 手工实现。

用人工来检测和处理所有的错误,并将其改正,但这类方式只针对少量数据。

(2) 专门应用程序实现。

编写程序算法和规则,使计算机按照人为的设定运行,使之完成整个清洗工作。

(3) 特定应用领域的实现。

应用概率统计的原理查找异常的数据,并改正。

(4) 与特定领域无关的实现。

这一类的清洗主要是指对重复数据进行检测进而清洗。

3.2.8 数据清洗的方法

数据清洗的目的是提高数据质量,数据清洗的清洗方法,可以从实例层脏数据和模式层脏数据两方面进行分析。

1. 实例层脏数据清洗方法

实例层脏数据主要包括重复数据检测、孤立点检测和属性值的脏数据检测三大方面,不过就目前的应用来看,主要侧重于对重复数据和孤立点的检测。重复数据检测主要分为基于记录和基于字段的重复检测,基于记录的重复检测算法主要有排序邻居算法、Canopy聚类算法、优先队列算法。排队邻居算法侧重于关键字的选取,Canopy聚类算法最主要的是降低了量,优先队列算法的实用性较好,但对于阈值的设置很关键。基于字段的重复检测算法主要有编辑距离算法、树编辑距离算法、TI Similarity相似匹配算法、Cosine相识度函数算法。编辑距离算法也称Levenshtein算法,主要用来比较两个字符

串的相似度，比较常用且易于实现。树编辑距离算法便于数据交换和机器自动化处理，也是编辑距离算法的一大分支。TI Similarity 相似匹配算法有较好的适用性，时间复杂度也较小，但对字符串的缩写就表现得力不从心。Cosine 相识度函数算法更加侧重于文本字段的重复检测。

2. 模式层脏数据清洗方法

模式层脏数据主要是由于数据结构设计不合理和属性约束不够全面两方面原因造成的，对此提出了结构冲突的清洗方法和噪声数据清洗方法。解决结构冲突有人工干预法和函数依赖法等两大解决方法。人工干预法主要是解决程序运行的弱点——不能识别关键字和数据类型的冲突，该方法比较有效，准确度高，但效率比较低。函数依赖法通过属性之间的依赖关系查找违反函数依赖关系的数据，来进行清洗，该方法效率较高，但必须满足依赖关系条件的局限。噪声数据的清洗方法主要有分箱法、人机结合法和简单规则库法。分箱法主要应用于数字类型的数据，对文本中的噪声数据并不适用。人机结合法也是常用的清洗方法，通过计算机检测出脏数据，再通过人工手动清洗并修正数据，不过，这个清洗方法效率较低，仅对小量数据比较适用。简单规则库法需要首先建立规则库，再通过这些规则库去约束数据，进而达到清洗的目的，不过此方法对规律性不是很强的数据适用性不大。下面介绍 3 种不同情况下的数据清洗。

(1) 基于异常值的数据清洗。

异常数据是指数据库或数据仓库中不符合一般规律的数据对象，又称为孤立点。异常数据可能由执行失误造成，也可能因设备故障而导致结果异常。异常数据可能是去掉的噪声，也可能是含有重要信息的数据单元。因此，在数据清洗中，异常数据的检测十分重要。异常数据的探测主要有基于统计学、基于距离和基于偏离 3 类方法。有的采用数据审计的方法实现异常数据的自动化检测，该方法也称为数据质量挖掘(DQM)。DQM 主要由两步构成：采用数理统计方法对数据分布进行概化描述，自动获得数据的总体分布特征；针对特定的数据质量问题进行挖掘以发现数据异常。DataGlich-es 将数据按距离划分为不同的层，在每一层统计数据特征，再根据定义的距离计算各数据点和中心距离的远近来判断异常是否存在。但是，并非所有的异常数据都是错误数据，在检测出异常数据后，还应结合领域知识和元数据进一步分析，以发现其中的错误。

(2) 基于重复值的数据清洗。

如今，信息集成系统在各行各业中得到广泛应用。对多数据源和单数据源数据进行集成时，多个记录代表同一实体的现象经常存在，这些记录称为重复记录。同时，有些记录并非完全重复，其个别字段存在一定差别，但表示的却是同一对象，此类记录即为相似重复记录。相似重复记录检测是数据清洗研究的重要方面，在信息集成系统中，重复记录不仅导致数据冗余，浪费了网络带宽和存储空间，还提供给用户很多相似信息，起到误导作用。解决该类问题主要基于数据库和人工智能的方法。邻近排序算法(Sorted Neighborhood Method，SNM)是检测重复记录的常用方法，该方法基于排序比较的思想，已得到广泛使用。基于排序比较思想的方法还有多趟排序和优先权队列等算法。另外，有人提出了基于 N-gram 的重复记录检测方法，并给出了改进的优先权队列算法，以准确

地聚类相似重复记录。也有人利用依赖图的概念,计算数据表中的关键属性,根据关键属性值将记录集划分为小记录集,在每个小记录集中检测相似重复记录。还有人提出了分割法,将某一字符串分割成几个组成部分来处理,从一定程度上解决了同一对象多种表示形式的问题。非结构化数据清洗是清洗技术的难点,近年来,针对非结构化数据的重复检测技术也在不断发展。通过介绍复杂数据实体识别的概念和应用,分别就 XML 数据、图数据和复杂网络上实体识别技术进行了讨论,并介绍了相应的技术原理。

(3) 基于残缺值的数据清洗。

残缺值问题是真实数据集中的普遍现象,许多原因都会产生缺失值。例如,设备故障问题导致的测量值丢失和在调查问卷中故意回避某些问题,这些缺失数据经常会带来一些问题。因此,业内提出了很多方法。一种处理缺失值的简单方法是忽略含有缺失值的实例或属性,但是浪费的数据可能相当多,且不完整的数据集可能会导致统计分析偏差。当然,有些数据分析的方法是容忍这些缺失值的。有很多数据挖掘方法用于估计缺失数据。这些方法根据数据间的关联性估计出准确的缺失值,并通过合适的方法对缺失值进行填充。可利用 5 组医疗数据集测试缺失数据对于病情阳性概率的影响,以及对分类结果精确度的影响,并通过最近邻搜索、判别分析和朴素贝叶斯 3 种方法在数据缺失不同比例下,分别对分类效果进行分析比较。

填充缺失数据工作通常以替代值填补的方式进行,它可以通过多种方法实现,如均值填补法使用数据的均值作为替代值。然而,该方法忽略了数据不一致的问题,并且没有考虑属性间的关系,属性间的关联性在缺失值估计过程中非常重要。数据挖掘方法的关键是挖掘属性间的关系,替代缺失值时,利用这些关系非常重要。由此观点出发,填补的目的在于估计正确的替代值,并避免填充偏差问题。如果拥有合适的填补方法,则能得到高质量的数据,数据挖掘结果也会得到改善。基于不完备数据分析的思想,有人提出了基于不完备数据聚类的缺失数据填补方法,针对分类变量不完备数据集定义约束容差集合差异度,从集合的角度判断不完备数据对象的总体相异程度,并以不完备数据聚类的结果为基础填补缺失数据。有的提出了基于进化算法的聚类方法。有的针对缺失数据问题,提出了多元回归方法,弥补了一元回归方法的不足。这些方法都很好地解决了缺失数据的估计问题。

3.3 Python 下的数据清洗

3.3.1 Python 概述

Python 是一种面向对象的解释型计算机程序设计语言,由荷兰人吉多·范·罗斯姆(Guido Van Rossum)于 1989 年发明,第一个公开发行版发行于 1991 年。Python 是纯粹的自由软件,源代码和解释器 CPython 遵循 GPL(General Public License)协议。Python 语法简洁清晰,其特色之一是强制用空白符(White Space)作为语句缩进。

Python 具有丰富和强大的库。它常被称为胶水语言,能够把用其他语言编写的各种模块(尤其是 C/C++)很轻松地联结在一起。常见的一种应用情形是,使用 Python 快速

生成程序的原型(有时甚至是程序的最终界面),然后将其中有特殊要求的部分,用更合适的语言改写,如3D游戏中的图形渲染模块,性能要求特别高,可以用C/C++重写,然后封装为Python可以调用的扩展类库。需要注意的是,在使用扩展类库时,可能需要考虑平台问题,某些可能不提供跨平台的实现。

3.3.2 Python的特点

1. Python的优点

Python有着别具一格的特性,有很多其他语言不具有的优点,具体表现为如下几点。

(1) 简单。

Python是一种代表简单主义思想的语言。阅读一个良好的Python程序就感觉像是在读英语一样。它使用户能够专注于解决问题而不是弄明白语言本身。

(2) 易学。

Python极其容易上手,因为Python有极其简单的说明文档。

(3) 速度快。

Python的底层是用C语言编写的,很多标准库和第三方库也都是用C语言编写,因此,其运行速度非常快。

(4) 免费、开源。

Python是FLOSS(自由/开放源码软件)之一,用户可以自由地发布这个软件的复制版本、阅读它的源代码,对它做改动,把它的一部分用于新的自由软件中。FLOSS是基于一个团体分享知识的概念。

(5) 高级语言。

用Python语言编写程序时无需考虑诸如如何管理程序使用的内存一类的底层细节。

(6) 可移植性强。

由于它的开源本质,Python已经被移植在许多平台上(经过改动使它能够工作在不同平台上)。这些平台包括Linux、Windows、FreeBSD、Mac OS、Solaris、OS/2、Amiga、AROS、AS/400、BeOS、OS/390、Z/OS、Palm OS、QNX、VMS、Psion、Acom RISC OS、VxWorks、PlayStation、Sharp Zaurus、Windows CE、PocketPC、Symbian和Android。

(7) 解释性。

用编译性语言,如C或C++编写的程序可以从源文件(即C或C++语言)转换到用户计算机使用的语言(二进制代码,即0和1)。这个过程通过编译器和不同的标记、选项完成。运行程序时,连接/转载器软件把用户的程序从硬盘复制到内存中并运行。而用Python语言编写的程序不需要编译成二进制代码,可以直接从源代码运行程序。

在计算机内部,Python解释器把源代码转换成称为字节码的中间形式,然后再把它翻译成计算机使用的机器语言并运行。这使得使用Python更加简单,也使得Python程序更加易于移植。

(8) 面向对象。

Python 既支持面向过程的编程,也支持面向对象的编程,在“面向过程”的语言中,程序是由过程或仅仅是可重用代码的函数构建起来的,在“面向对象”的语言中,程序是由数据和功能组合而成的对象构建起来的。

(9) 可扩展性高。

如果需要一段关键代码运行得更快或者希望某些算法不公开,可以部分程序用 C 语言或 C++ 语言编写,然后在 Python 程序中使用它们。

(10) 可嵌入性。

可以把 Python 嵌入 C/C++ 程序,从而向程序用户提供脚本功能。

(11) 丰富的库。

Python 标准库确实很庞大,它可以帮助处理各种工作,包括正则表达式、文档生成、单元测试、线程、数据库、网页浏览器、CGI、FTP、电子邮件、XML、XML-RPC、HTML、WAV 文件、密码系统、GUI(图形用户界面)、Tk 和其他与系统有关的操作。这就是所谓 Python 的“功能齐全”理念。除了标准库以外,还有许多其他高质量的库,如 wxPython、Twisted 和 Python 图像库等。

(12) 规范的代码。

Python 采用强制缩进的方式使得代码具有较好的可读性,而且用 Python 语言编写的程序不需要编译成二进制代码。

2. Python 的缺点

任何语言都有许多不能实现的事务,Python 具有三大缺点。

(1) 单行语句和命令行输出问题。很多时候不能将程序连写成一行,例如

```
import sys;
for i in sys.path:print i
```

而 Perl 和 Awk 就无此限制,可以较为方便地在 Shell 下完成简单程序,不需要像 Python 一样,必须将程序写入一个 py 文件。

(2) Python 具有独特的语法。这也许不应该被称为局限,但是它用缩进来区分语句关系的方式还是给很多初学者带来了困惑。即便是很有经验的 Python 程序员,也可能陷入陷阱当中。

(3) 运行速度慢。这里是指与 C 语言和 C++ 语言相比。

3.3.3 Python Pandas——数据清洗

数据缺失在大部分数据分析应用中都很常见,Pandas 工具使用浮点值 Na 表示浮点和非浮点数组中的缺失数据。

1. 处理 Na 的方法

处理 Na 的方法通常有 3 种: dropna、fillna、is(not)null。

(1) dropna：对于一个 Series，dropna 返回一个仅含非空数据和索引值的 Series。

问题在于 DataFrame 的处理方式，因为一旦 drop 的话，至少要丢掉一行(列)。这里解决方法与前面类似，还是通过一个额外的参数：dropna(axis=0，how='any '，thresh=None)，how 参数可选的值为 any 或者 all，all 仅在切片元素全为 NA 时才抛弃该行(列)。thresh 为整数类型，eg：thresh=3，表示一行当中至少有 3 个 NA 值时才将其保留。

(2) fillna：fillna(value=None，method=None，axis=0)中的 value 除了基本类型外，还可以使用字典，这样可以对不同列填充不同的值。

(3) is(not)null：这一对方法对对象做出元素级的应用，然后返回一个布尔型数组，一般可用于布尔型索引。

2. 处理流程

(1) 过滤数据。

① 对于一个 Series，Dropna 返回一个仅含非空数据和索引值的 Series，处理结果如图 3-2 所示。

```
from pandas import Series,DataFrame
from numpy import nan as NA
data=Series([1,NA,3.5,NA,7])
print(data.dropna())
```

```
D:\Users\yangenneng0\AppData\Local\Programs
0    1.0
2    3.5
4    7.0
dtype: float64

Process finished with exit code 0
```

图 3-2　过滤数据显示结果

② 另一个过滤 DataFrame 行的问题涉及问题序列数据。假设只想保留一部分观察数据，可以用 thresh 参数实现此目的，处理结果如图 3-3 所示。

```
from pandas import Series,DataFrame, np
from numpy import nan as NA
data=DataFrame(np.random.randn(7,3))
data.ix[:4,1]=NA
data.ix[:2,2]=NA
print(data)
print("...........")
print(data.dropna(thresh=2))
```

③ 如果不想滤除缺失的数据，而是通过其他方式填补“空洞”，fillna()是最主要的函数。通过一个常数调用 fillna 就会将缺失值替换为那个常数值，处理结果如图 3-4 所示。

```
from pandas import Series,DataFrame, np

from numpy import nan as NA

data=DataFrame(np.random.randn(7,3))

data.ix[:4,1]=NA

data.ix[:2,2]=NA

print(data)

print("..........")

print(data.fillna(0))
```

```
D:\Users\yangenneng0\AppData\Local\Pro
          0         1         2
0 -0.362989       NaN       NaN
1  0.104062       NaN       NaN
2 -0.478875       NaN       NaN
3 -0.501911       NaN -1.163823
4 -0.236566       NaN  1.572364
5 -1.202140  0.998669 -0.519850
6 -0.521973  0.156062 -2.088459
..........
          0         1         2
3 -0.501911       NaN -1.163823
4 -0.236566       NaN  1.572364
5 -1.202140  0.998669 -0.519850
6 -0.521973  0.156062 -2.088459

Process finished with exit code 0
```

图 3-3 过滤 DataFrame 行的结果

```
D:\Users\yangenneng0\AppData\Local\Programs\P
          0         1         2
0 -1.988129       NaN       NaN
1  0.649393       NaN       NaN
2  0.291714       NaN       NaN
3 -0.217730       NaN -0.151283
4  0.272920       NaN -1.133399
5  0.363155 -0.269356 -0.975531
6  0.126717  1.334043 -1.045276
..........
          0         1         2
0 -1.988129  0.000000  0.000000
1  0.649393  0.000000  0.000000
2  0.291714  0.000000  0.000000
3 -0.217730  0.000000 -0.151283
4  0.272920  0.000000 -1.133399
5  0.363155 -0.269356 -0.975531
6  0.126717  1.334043 -1.045276

Process finished with exit code 0
```

图 3-4 通过 fillna()填补空洞

④ 如果是通过一个字典调用 fillna()，就可以对不同列填充不同的值，处理结果如图 3-5 所示。

```
from pandas import Series,DataFrame, np
from numpy import nan as NA
```

```
data=DataFrame(np.random.randn(7,3))

data.ix[:4,1]=NA

data.ix[:2,2]=NA

print(data)

print("..........")

print(data.fillna({1:111,2:222}))
```

```
D:\Users\yangenneng0\AppData\Local\Programs
          0         1         2
0  0.479313       NaN       NaN
1 -0.527575       NaN       NaN
2 -0.190653       NaN       NaN
3  0.470188       NaN  0.929928
4 -1.534236       NaN  0.642039
5 -0.678035  0.280222  0.648476
6  0.572486 -0.110755 -0.346703
..........
          0           1           2
0  0.479313  111.000000  222.000000
1 -0.527575  111.000000  222.000000
2 -0.190653  111.000000  222.000000
3  0.470188  111.000000    0.929928
4 -1.534236  111.000000    0.642039
5 -0.678035    0.280222    0.648476
6  0.572486   -0.110755   -0.346703

Process finished with exit code 0
```

图 3-5　通过字典调用 fillna()实现填补

⑤ 还可以利用 fillna()实现许多其他功能，比如可以传入 Series 的平均值或中位数，处理结果如图 3-6 所示。

```
from pandas import Series,DataFrame, np

from numpy import nan as NA

data=Series([1.0,NA,3.5,NA,7])

print(data)

print("..........\n")

print(data.fillna(data.mean()))
```

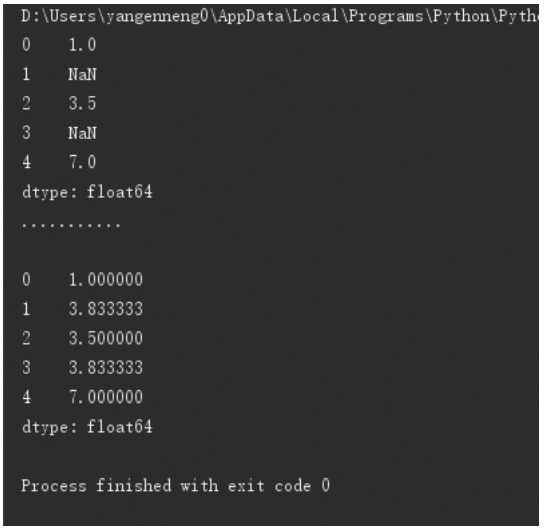

图 3-6 利用 fillna()实现其他功能

(2) 检测和过滤异常值。

异常值(Outlier)的过滤或变换运算在很大程度上就是数组运算。例如,(1000,4)的标准正态分布数组的处理结果如图 3-7 所示。

```
from pandas import Series,DataFrame, np

from numpy import nan as NA

data=DataFrame(np.random.randn(1000,4))

print(data.describe())

print("\n....找出某一列中绝对值大小超过 3 的项...\n")

col=data[3]

print(col[np.abs(col) > 3] )

print("\n....找出全部绝对值超过 3 的值的行...\n")

print(col[(np.abs(data) > 3).any(1)] )
```

(3) 移除重复数据。

① 二维表格型数据结构(DataFrame)的遗传(Duplicated)方法返回一个布尔型 Series,表示各行是否是重复行,处理结果如图 3-8 所示。

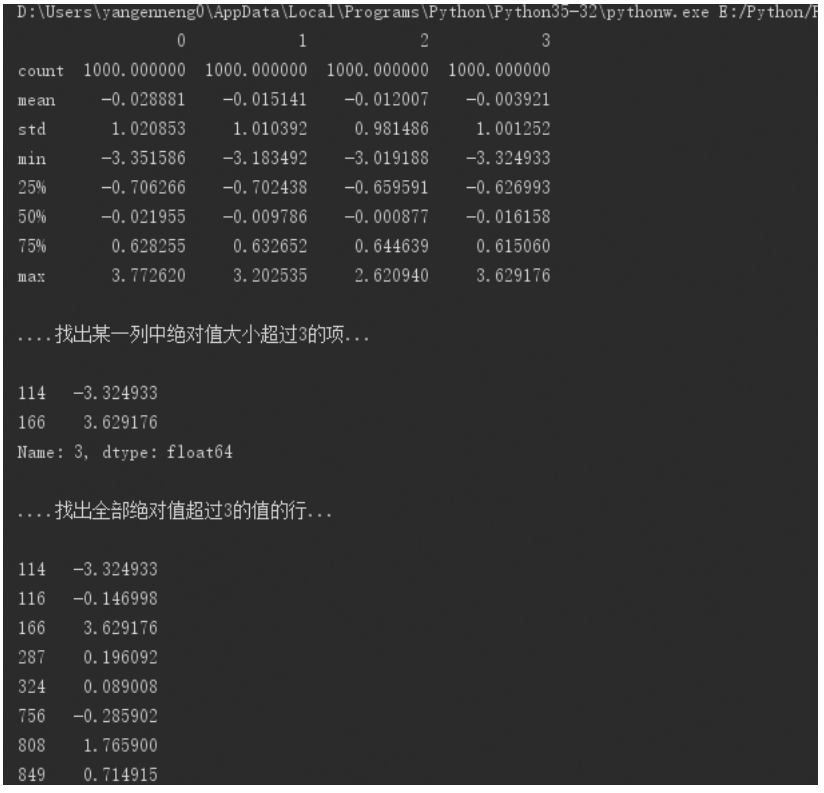

图 3-7　异常值过滤

```
from pandas import Series,DataFrame, np

from numpy import nan as NA

import pandas as pd

import numpy as np

data=pd.DataFrame({'k1':['one']*3+['two']*4, 'k2':[1,1,2,2,3,3,4]})

print(data)

print(".....\n")

print(data.duplicated())
```

② 与此相关的还有一个 drop_duplicated()方法，它用于返回一个移除了重复行的 DataFrame，处理结果如图 3-9 所示。

```
from pandas import Series,DataFrame, np
```

```
from numpy import nan as NA

import pandas as pd

import numpy as np

data=pd.DataFrame({'k1':['one']*3+['two']*4, 'k2':[1,1,2,2,3,3,4]})

print(data)

print(".....\n")

print(data.drop_duplicates())
```

```
D:\Users\yangenneng0\AppData\Local\Programs\
    k1  k2
0  one   1
1  one   1
2  one   2
3  two   2
4  two   3
5  two   3
6  two   4
.....

0    False
1     True
2    False
3    False
4    False
5     True
6    False
dtype: bool

Process finished with exit code 0
```

图 3-8 移除重复数据(1)

```
D:\Users\yangenneng0\AppData\Local\P
    k1  k2
0  one   1
1  one   1
2  one   2
3  two   2
4  two   3
5  two   3
6  two   4
.....

    k1  k2
0  one   1
2  one   2
3  two   2
4  two   3
6  two   4

Process finished with exit code 0
```

图 3-9 移除重复数据(2)

③ 上面两个方法会默认判断全部列，也可以指定判断部分列的重复项，假设还有一列值，而只希望根据 k1 列过滤重复项，处理结果如图 3-10 所示。

```
from pandas import Series,DataFrame, np

from numpy import nan as NA

import pandas as pd

import numpy as np
```

```
data=pd.DataFrame({'k1':['one']*3+['two']*4, 'k2':[1,1,2,2,3,3,4]})

data['v1']=range(7)

print(data)

print(".....\n")

print(data.drop_duplicates(['k1']))
```

```
D:\Users\yangenneng0\AppData\Local\Programs\Pyt
    k1  k2  v1
0  one   1   0
1  one   1   1
2  one   2   2
3  two   2   3
4  two   3   4
5  two   3   5
6  two   4   6
.....

    k1  k2  v1
0  one   1   0
3  two   2   3

Process finished with exit code 0
```

图 3-10　移除重复数据(3)

④ duplicates 和 drop_duplicates 默认保留第一个出现的值组合，传入 take_last＝True 则保留最后一个，处理结果如图 3-11 所示。

```
from pandas import Series,DataFrame, np

from numpy import nan as NA

import pandas as pd

import numpy as np

data=pd.DataFrame({'k1':['one']*3+['two']*4, 'k2':[1,1,2,2,3,3,4]})

data['v1']=range(7)
```

```
print(data)

print(".....\n")

print(data.drop_duplicates(['k1','k2'],take_last=True))
```

```
D:\Users\yangenneng0\AppData\Local\P
    k1  k2  v1
0  one   1   0
1  one   1   1
2  one   2   2
3  two   2   3
4  two   3   4
5  two   3   5
6  two   4   6
.....

E:/Python/PyCharm-WorkSpace/PythonAp
  print(data.drop_duplicates(['k1','
    k1  k2  v1
1  one   1   1
2  one   2   2
3  two   2   3
5  two   3   5
6  two   4   6

Process finished with exit code 0
```

图 3-11 移除重复数据(4)

3.4 数据转换工具

3.4.1 Data Stage

1. Data Stage 的特性

Data Stage 是在构建数据仓库过程中进行数据清洗、数据转换的一套工具。它的工作流程如图 3-12 所示。

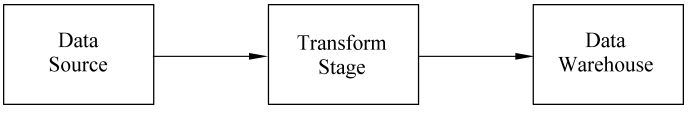

图 3-12 Data Stage 工作流程

Data Stage 包括设计、开发、编译、运行及管理等整套工具。运用 Data Stage 能够对来自一个或多个不同数据源中的数据进行析取、转换,再将结果装载到一个或多个目的